

THIRD EDITION

GONZALO CAMARILLO
MIGUEL A. GARCÍA-MARTÍN

THE 3G IP MULTIMEDIA SUBSYSTEM (IMS)



UICC



MERGING THE INTERNET AND THE CELLULAR WORLDS

 WILEY

The 3G IP Multimedia Subsystem (IMS)

Merging the Internet and the
Cellular Worlds

Third Edition

Gonzalo Camarillo
Ericsson, Finland

Miguel A. García-Martín
Ericsson, Spain



A John Wiley and Sons, Ltd, Publication

The 3G IP Multimedia Subsystem (IMS)

The 3G IP Multimedia Subsystem (IMS)

Merging the Internet and the
Cellular Worlds

Third Edition

Gonzalo Camarillo
Ericsson, Finland

Miguel A. García-Martín
Ericsson, Spain



A John Wiley and Sons, Ltd, Publication

This edition first published 2008
© 2008 John Wiley & Sons Ltd

First edition first published 2004 © 2004 John Wiley & Sons Ltd
Second edition first published 2005 © 2005 John Wiley & Sons Ltd

Registered office

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

For details of our global editorial offices, for customer services and for information about how to apply for permission to reuse the copyright material in this book please see our website at www.wiley.com.

The right of the author to be identified as the author of this work has been asserted in accordance with the Copyright, Designs and Patents Act 1988.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by the UK Copyright, Designs and Patents Act 1988, without the prior permission of the publisher.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The publisher is not associated with any product or vendor mentioned in this book. This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Library of Congress Cataloging-in-Publication Data

Camarillo, Gonzalo.

The 3G IP multimedia subsystem (IMS) : merging the Internet and the cellular worlds / Gonzalo Camarillo, Miguel A. García-Martín. – 3rd ed.

p. cm.

Includes bibliographical reference and index.

ISBN 978-0-470-51662-1 (cloth)

1. Wireless communication systems. 2. Mobile communication systems. 3. Multimedia communications. 4. Internet Protocol multimedia subsystem. I. García-Martín, Miguel A. II. Title.

TK5103.2.C35 2008

621.384—dc22

2008022139

A catalogue record for this book is available from the British Library.

ISBN 978-0-470-51662-1 (HB)

Set in 10/12pt Times by Sunrise Setting Ltd, Torquay, UK.
Printed in Great Britain by Antony Rowe Ltd, Chippenham, England.

To my parents, Anselmo and Isabel; my brothers, Alvaro, Daniel, and Ignacio; and Viviana. They all are a source of energy and motivation in everything I do.

Gonzalo

To my daughter Maria Elizabeth, who was born at the time I started writing this book – she is the sunshine of my life; my wife Jelena, who provided me with all the support and love I needed; my parents, José and Mari-Luz, my aunt Feli, my brother Javier José who, through the distance, encouraged and supported me during this project.

Miguel Angel

Contents

Foreword by Stephen Hayes	xxi
Foreword by Allison Mankin and Jon Peterson	xxiii
About the Authors	xxv
Preface to the Third Edition	xxvii
Preface to the Second Edition	xxix
Preface to the First Edition	xxxi
Acknowledgements	xxxiii

Part I Introduction to the IMS	1
---------------------------------------	----------

1 IMS Vision: Where Do We Want to Go?	5
1.1 The Internet	5
1.2 The Cellular World	6
1.3 Why do we need the IMS?	6
1.4 Relation between IMS and non-IMS Services	8
2 The History of the IMS Standardization	9
2.1 Relations between IMS-related Standardization Bodies	9
2.2 Internet Engineering Task Force	10
2.2.1 Structure of the IETF	10
2.2.2 Working Group Operations	12
2.2.3 Types of RFCs	12
2.3 Third Generation Partnership Project	14
2.3.1 3GPP Structure	15
2.3.2 3GPP Deliverables	15
2.4 Third Generation Partnership Project 2	15
2.4.1 3GPP2 Structure	16
2.4.2 3GPP2 Deliverables	17
2.5 IETF-3GPP/3GPP2 Collaboration	18
2.5.1 Internet Area	18

2.5.2	Operations and Management Area	19
2.5.3	Transport Area	19
2.6	Open Mobile Alliance	20
2.6.1	OMA Releases and Specifications	20
2.6.2	Relationship between OMA and 3GPP/3GPP2	22
2.6.3	Relationship between OMA and the IETF	22
3	General Principles of the IMS Architecture	25
3.1	From Circuit-switched to Packet-switched	25
3.1.1	GSM Circuit-switched	25
3.1.2	GSM Packet-switched	26
3.2	IMS Requirements	26
3.2.1	IP Multimedia Sessions	27
3.2.2	QoS	27
3.2.3	Interworking	28
3.2.4	Roaming	28
3.2.5	Service Control	28
3.2.6	Rapid Service Creation	29
3.2.7	Multiple Access	29
3.3	Overview of Protocols used in the IMS	29
3.3.1	Session Control Protocol	29
3.3.2	The AAA Protocol	30
3.3.3	Other Protocols	31
3.4	Overview of IMS Architecture	31
3.4.1	The Databases: the HSS and the SLF	32
3.4.2	The CSCF	33
3.4.2.1	The P-CSCF	33
3.4.2.2	P-CSCF Location	34
3.4.2.3	The I-CSCF	34
3.4.2.4	I-CSCF Location	35
3.4.2.5	The S-CSCF	35
3.4.2.6	S-CSCF Location	35
3.4.3	The Application Server	35
3.4.3.1	AS Location	37
3.4.4	The MRF	37
3.4.4.1	MRF Location	37
3.4.5	The BGCF	37
3.4.6	The IMS-ALG and the TrGW	37
3.4.7	The PSTN/CS Gateway	39
3.4.8	Home and Visited Networks	40
3.5	Identification in the IMS	42
3.5.1	Public User Identities	42
3.5.2	Private User Identities	43
3.5.3	The Relation between Public User Identities and Private User Identities	43
3.5.4	Public Service Identities	43

3.6	SIM, USIM, and ISIM in 3GPP	45
3.6.1	SIM	45
3.6.2	USIM	46
3.6.3	ISIM	48
3.7	Next Generation Networks (NGN)	49
3.7.1	NGN Overview	49
3.7.2	The Core IMS in NGN	51

Part II The Signaling Plane in the IMS 55

4	Session Control on the Internet	59
4.1	SIP Functionality	59
4.1.1	Session Descriptions and SDP	59
4.1.2	The Offer/Answer Model	60
4.1.3	SIP and SIPS URIs	61
4.1.4	User Location	62
4.2	SIP Entities	63
4.2.1	Forking Proxies	65
4.2.2	Redirect Servers	66
4.3	Message Format	67
4.4	The Start Line in SIP Responses: the Status Line	67
4.5	The Start Line in SIP Requests: the Request Line	68
4.6	Header Fields	68
4.7	Message Body	70
4.8	SIP Transactions	71
4.9	Message Flow for Session Establishment	72
4.10	SIP Dialogs	75
4.10.1	Record-Route, Route, and Contact Header Fields	78
4.11	Extending SIP	78
4.11.1	New Methods	80
4.12	Caller Preferences and User Agent Capabilities	80
4.13	Reliability of Provisional Responses	81
4.14	Preconditions	84
4.15	Event Notification	85
4.15.1	High Notification Rates	87
4.15.1.1	Event Throttling	88
4.15.1.2	Conditional Event Notification	88
4.16	Signaling Compression	89
4.16.1	SigComp Extended Operations	90
4.16.2	Static SIP/SDP Dictionary	91
4.17	Content Indirection	91
4.18	The REFER Method	93
4.19	Globally Routable User Agent URIs (GRUU)	96
4.20	NAT Traversal	100
4.20.1	Types of NAT	101
4.20.2	STUN	105
4.20.3	TURN	105
4.20.4	ICE	106

5 Session Control in the IMS	111
5.1 Prerequisites for Operation in the IMS	112
5.2 IPv4 and IPv6 in the IMS	113
5.3 IP Connectivity Access Network	114
5.4 P-CSCF Discovery	115
5.5 IMS-level Registration	117
5.5.1 IMS Registration with an ISIM	117
5.5.2 IMS Registration with a USIM	125
5.5.2.1 Temporary Private User Identity	126
5.5.2.2 Temporary Public User Identity	127
5.5.2.3 Home Network Domain URI	127
5.5.2.4 Registration Flow	127
5.6 Subscription to the <code>reg</code> Event State	130
5.7 Basic Session Setup	135
5.7.1 The IMS Terminal Sends an INVITE Request	138
5.7.2 The Originating P-CSCF Processes the INVITE Request	145
5.7.3 The Originating S-CSCF Processes the INVITE Request	146
5.7.4 The Terminating I-CSCF Processes the INVITE Request	150
5.7.5 The Terminating S-CSCF Processes the INVITE Request	151
5.7.6 The Terminating P-CSCF Processes the INVITE Request	154
5.7.7 The Callee's Terminal Processes the INVITE Request	154
5.7.8 Processing the 183 Response	158
5.7.9 The Caller's IMS Terminal Processes the 183 Response	159
5.7.10 The Callee's IMS Terminal Processes the PRACK request	159
5.7.11 Alerting the Callee	161
5.8 Application Servers: Providing Services to Users	166
5.8.1 Generalities about Application Servers	167
5.8.2 Types of Application Servers	169
5.8.2.1 The SIP Application Server	169
5.8.2.2 The OSA-SCS	170
5.8.2.3 The IM-SSF Application Server	170
5.8.3 The Session Setup Model through Application Servers	171
5.8.3.1 Application Server Acting as a SIP User Agent	172
5.8.3.2 Application Server Acting as a SIP Proxy Server	174
5.8.3.3 Application Server Acting as a SIP Redirect Server	177
5.8.3.4 Application Server Acting as a SIP B2BUA	178
5.8.4 Filter Criteria	180
5.8.5 An Example of Service Execution	184
5.9 Changes due to Next Generation Networks (NGN)	188
5.9.1 New SIP Functionality in NGN	188
5.9.2 Unneeded IMS Functionality in NGN	189
5.10 Interworking	189
5.10.1 SIP–PSTN Interworking	189
5.10.1.1 Gateway Architecture in the IMS	191
5.10.1.2 The BGCF	192
5.10.2 Interworking with Non-IMS SIP-based Networks	192
5.10.2.1 IPv4/IPv6 Interworking	192

5.11	Combinational Services	196
5.12	Basic Sessions Not Requiring Resource Reservation	197
5.12.1	The Callee Does Not Require Resource Reservation	198
5.12.2	The Caller Does Not Require Resource Reservation	204
5.12.3	Neither the Caller Nor the Callee Require Resource Reservation	206
5.13	Globally Routable User Agent URIs (GRUU) in IMS	206
5.14	IMS Communication Service Identifier (ICSI)	209
5.15	IMS Application Reference Identifier (IARI)	212
5.16	NAT Traversal in the IMS	213
6	AAA on the Internet	215
6.1	Authentication, Authorization, and Accounting	215
6.2	AAA Framework on the Internet	215
6.3	The Diameter Protocol	217
6.3.1	Diameter Sessions	218
6.3.2	The Format of a Diameter Message	219
6.3.3	Attribute-Value Pairs	220
6.3.4	The AAA and AAAS URIs	221
6.3.5	Diameter Base Protocol Commands	222
6.3.5.1	Abort Session Request and Answer (ASR, ASA)	223
6.3.5.2	Accounting Request and Answer (ACR, ACA)	223
6.3.5.3	Capabilities Exchange Request and Answer (CER, CEA) .	223
6.3.5.4	Device Watchdog Request and Answer (DWR, DWA) . . .	223
6.3.5.5	Disconnect Peer Request and Answer (DPR, DPA) . . .	223
6.3.5.6	Re-Authentication Request and Answer (RAR, RAA) . . .	224
6.3.5.7	Session Termination Request and Answer (STR, STA) .	224
6.3.6	Diameter Base Protocol AVPs	224
7	AAA in the IMS	227
7.1	Authentication and Authorization in the IMS	227
7.2	The <i>Cx</i> and <i>Dx</i> Interfaces	227
7.2.1	Command Codes Defined in the Diameter Application for the <i>Cx</i> Interface	229
7.2.1.1	User Authorization Request and Answer (UAR, UAA) .	229
7.2.1.2	Multimedia Auth Request and Answer (MAR, MAA) .	230
7.2.1.3	Server Assignment Request and Answer (SAR, SAA) .	230
7.2.1.4	Location Information Request and Answer (LIR, LIA) .	231
7.2.1.5	Registration Termination Request and Answer (RTR, RTA) .	232
7.2.1.6	Push Profile Request and Answer (PPR, PPA)	232
7.2.2	AVPs Defined in the Diameter Application for the <i>Cx</i> Interface	232
7.2.2.1	Usage of Existing AVPs	236
7.2.3	The User Profile	236
7.3	The <i>Sh</i> Interface	238
7.3.1	Command Codes Defined in the Diameter Application for the <i>Sh</i> Interface	239
7.3.1.1	User Data Request and Answer (UDR, UDA)	239
7.3.1.2	Profile Update Request and Answer (PUR, PUA)	240

7.3.1.3	Subscribe Notifications Request and Answer (SNR, SNA)	240
7.3.1.4	Push Notification Request and Answer (PNR, PNA)	241
7.3.2	AVPs Defined in the Diameter Application for the <i>Sh</i> Interface	241
7.4	Accounting	242
8	Policy and Charging Control in the IMS	243
8.1	PCC Architecture	243
8.1.1	Session Establishment and Policy Control	244
8.1.2	SIP Procedures	246
8.1.3	Proxy Access to SDP Bodies	247
8.1.4	Status of the Signaling Bearer	248
8.1.5	The <i>Rx</i> Interface	249
8.1.6	The <i>Gx</i> Interface	249
8.2	Charging Architecture	251
8.3	Offline Charging Architecture	251
8.3.1	Charging-related SIP Header Fields	253
8.3.2	IMS Terminal in a Visited Network	253
8.3.3	IMS Terminal in its Home Network	255
8.3.4	The <i>Rf</i> Interface	258
8.3.5	The <i>Ga</i> Interface	258
8.4	Online Charging Architecture	260
8.4.1	S-CSCF	260
8.4.2	Application Servers and the MRFC	260
8.4.3	Types of Online Charging	261
8.4.3.1	Unit Determination	262
8.4.3.2	Rating	264
8.4.3.3	Tariff Changes	264
8.4.4	The <i>Ro</i> Interface	265
8.4.5	The <i>Re</i> Interface	265
9	Quality of Service on the Internet	267
9.1	Integrated Services	267
9.1.1	RSVP	267
9.1.2	State in the Network	269
9.2	Differentiated Services	269
10	Quality of Service in the IMS	271
10.1	Policy Control and QoS	271
10.2	Instructions to Perform Resource Reservations	271
10.2.1	Proxy Modifying Bodies	272
10.3	Reservations by the Terminals	274
10.4	QoS in the Network	275

11 Security on the Internet	277
11.1 HTTP Digest Access Authentication	277
11.1.1 Security Properties of Digest	279
11.2 Certificates	280
11.3 TLS	280
11.3.1 SIP Usage	281
11.3.1.1 Client Authentication	282
11.4 S/MIME	282
11.4.1 Self-signed Certificates	284
11.5 Authenticated Identity Body	285
11.6 IPsec	287
11.6.1 ESP and AH	287
11.6.2 Tunnel and Transport Modes	287
11.6.3 Internet Key Exchange	291
11.6.3.1 IKE Security Association Establishment	291
11.6.3.2 IPsec Security Association Establishment	291
11.7 Privacy	291
11.8 Encrypting Media Streams	292
11.8.1 MIKEY	292
12 Security in the IMS	293
12.1 Access Security	293
12.1.1 Authentication and Authorization	294
12.1.1.1 HTTP Digest Access Authentication	294
12.1.1.2 HTTP Digest Access Authentication using AKA	297
12.1.1.2.1 HTTP Digest Access Authentication with AKA: UICC Contains an ISIM	297
12.1.1.2.2 HTTP Digest Access Authentication with AKA: UICC Contains a USIM	300
12.1.2 IPsec Security Association Establishment	300
12.1.3 TLS Connection Establishment	302
12.1.4 IP-CAN Linked Authentication	303
12.1.4.1 Early IMS Security Solution	303
12.1.4.2 NASS-IMS Bundled Authentication	306
12.2 Network Security	308
12.2.1 TLS Usage for Network Security	309
13 Emergency Calls on the Internet	311
13.1 Introduction	311
13.2 Location Acquisition	312
13.2.1 Manual Configuration	313
13.2.2 Location Acquired from DHCP	313
13.2.3 Location Acquired from Layer 2 Protocols	314
13.2.4 Location Acquired from Application Layer Protocols	315
13.2.5 Location Acquired from a GPS	316
13.2.6 Wireless Triangulation	316
13.3 Identifying Emergency Calls	318
13.4 Locating the Closest PSAP	319

13.5	Issuing the Emergency Call	321
13.5.1	The Terminal Acquires its Location	322
13.5.2	The Terminal Does Not Have its Own Location	325
14	Emergency Calls in the IMS	329
14.1	Architecture for Supporting Emergency Calls in IMS	329
14.2	Establishing an Emergency Call in IMS	332
14.3	IMS Registration for Emergency Calls	333
14.4	Call Back from the PSAP to a User	334
14.5	Anonymous Calls	335
14.6	Emergency Calls in Fixed Broadband Accesses	336
Part III	The Media Plane in the IMS	337
15	Media Encoding	341
15.1	Speech Encoding	341
15.1.1	Pulse Code Modulation	342
15.1.2	Linear Prediction	343
15.1.3	GSM-FR	344
15.1.4	AMR	345
15.1.4.1	AMR Modes	345
15.1.4.2	LPC Coefficients Calculation	347
15.1.4.3	Codebooks	347
15.1.4.4	Adaptive Codebook	347
15.1.4.5	Fixed Codebook	348
15.1.4.6	Gains	348
15.1.5	AMR-WB	348
15.1.6	SMV	348
15.2	Video Encoding	353
15.2.1	Common Video Codecs	354
15.2.2	H.263	355
15.2.3	Image Encoding	355
15.2.4	Temporal Correlation	355
15.2.5	Spatial Correlation	356
15.3	Text Encoding	356
15.4	Mandatory Codecs in the IMS	356
16	Media Transport	359
16.1	Reliable Media Transport	359
16.2	Unreliable Media Transport	360
16.2.1	DCCP	360
16.2.2	RTP	361
16.2.3	RTCP	363
16.2.4	SRTP	364
16.3	Media Transport in the IMS	364

Part IV Building Services with the IMS	367
17 Service Configuration on the Internet	371
17.1 The XML Configuration Access Protocol (XCAP)	371
17.1.1 XCAP Application Usage	373
17.2 An Overview of XML	374
17.2.1 XML Namespaces	376
17.3 HTTP URIs that Identify XCAP Resources	376
17.4 XCAP Operations	378
17.4.1 Create or Replace Operations	378
17.4.2 Delete Operations	380
17.4.3 Fetching Operations	380
17.5 Entity Tags and Conditional Operations	380
17.6 Subscriptions to Changes in XML Documents	383
17.7 XML Patch Operations	386
18 Service Configuration in the IMS	389
18.1 XDM architecture	389
18.2 Downloading an XML Document, Attribute, or Element	391
18.3 Directory Retrieval	393
18.4 Data Search with XDM	397
18.5 Subscribing to Changes in XML Documents	403
19 The Presence Service on the Internet	405
19.1 Overview of the Presence Service	405
19.1.1 The <i>pres</i> URI	407
19.2 The Presence Life Cycle	407
19.3 Presence Subscriptions and Notifications	409
19.4 Presence Publication	412
19.5 Presence Information Data Format (PIDF)	412
19.5.1 Contents of the PIDF	413
19.6 The Presence Data Model for SIP	414
19.7 Mapping the SIP Presence Data Model to the PIDF	416
19.8 Rich PIDF	416
19.8.1 Contents of the RPID	417
19.9 CIPID	419
19.10 Timed Presence Extension to the PIDF	419
19.11 Presence Capabilities	421
19.11.1 Service Capabilities	423
19.11.2 Device Capabilities	424
19.11.3 An Example of the Presence Capabilities Document	424
19.12 Geographical Location in Presence	424
19.13 Watcher Information	427
19.14 Watcher Authorization: Presence Authorization Rules	430
19.14.1 Common Policy	430
19.14.2 Presence Authorization Policy Documents	431

19.14.3 Uploading Presence Authorization Policy Documents to the Presence Agent	433
19.14.4 Watcher Authorization: Complete Example	433
19.15 URI-list Services and Resource Lists	434
19.16 Presence Optimizations	437
19.16.1 Partial Notification of Presence Information	437
19.16.2 Partial Presence Publication	438
19.16.3 Event Notification Filtering	439
20 The Presence Service in the IMS	443
20.1 The Foundation of Services	443
20.2 Presence Architecture in the IMS	443
20.3 Presence Publication	444
20.4 Watcher Subscription	446
20.5 Watcher Information and Authorization of Watchers	447
20.6 Presence Optimizations	449
20.7 OMA Extensions to PIDF	450
21 Instant Messaging on the Internet	453
21.1 The <i>im</i> URI	453
21.2 Modes of Instant Messages	454
21.3 Pager-mode Instant Messaging	454
21.3.1 Congestion Control with MESSAGE	454
21.4 Session-based Instant Messaging	455
21.4.1 The MSRP and MSRPS URLs	456
21.4.2 MSRP Overview	456
21.4.3 Extensions to SDP due to MSRP	459
21.4.4 MSRP Core Functionality	460
21.4.5 Status and Reports	461
21.4.6 MSRP Relays	464
21.5 The “isComposing” Indication	468
21.6 Messaging Multiple Parties	470
21.6.1 MESSAGE URI-List Services	470
21.6.2 Chat Rooms	472
21.7 File Transfer	476
22 The Instant Messaging Service in the IMS	477
22.1 Pager-mode Instant Messaging in the IMS	477
22.2 Pager-mode Instant Messaging to Multiple Recipients	478
22.3 Session-based Instant Messaging in the IMS	478
22.4 File Transfer	482
23 Conferencing on the Internet	483
23.1 Conferencing Standardization at the IETF	483
23.2 The SIPPING Conferencing Framework	484
23.2.1 Signaling Architecture	485
23.2.2 Media Architecture	488

23.3	The XCON Conferencing Framework	489
23.3.1	Conference Objects	490
23.3.2	Conference Control Server	491
23.3.3	Foci and Notification Service	492
23.3.4	Floor Control Server	492
23.4	The Binary Floor Control Protocol (BFCP)	493
23.4.1	Contacting the Floor Control Server	494
23.4.1.1	Inside an Offer/Answer Exchange	494
23.4.1.2	Outside an Offer/Answer Exchange	495
23.4.2	BFCP Message Flow	495
23.4.3	BFCP Primitives	497
23.4.4	BFCP Encoding	497
24	Conferencing in the IMS	499
24.1	The IMS Conferencing Service	499
24.1.1	Creating and Joining a Conference	499
24.1.2	Other Actions	502
24.2	Relation with the Work in TISPAN and OMA	502
25	Push-to-talk over Cellular	503
25.1	PoC Standardization	503
25.2	IETF Work Relevant to PoC	504
25.2.1	URI-list Services	504
25.2.1.1	Multiple REFER	505
25.2.1.2	URI-list Format	506
25.2.1.3	Consent-based Communications	506
25.2.2	Event Package for PoC Settings	508
25.2.3	SIP Header Fields	508
25.3	Architecture	508
25.4	Registration	510
25.5	PoC Server Roles	511
25.6	PoC Session Types	512
25.6.1	One-to-one PoC Sessions	513
25.6.2	Ad-hoc PoC Group	514
25.6.3	Pre-arranged PoC Group	514
25.6.4	Chat PoC Group	515
25.7	Adding Users to a PoC Session	516
25.8	Group Advertisements	517
25.9	Session Establishment Types	518
25.10	Answer Modes	520
25.11	Right-to-send-media Indication Types	521
25.12	Participant Information	523
25.13	Barring and Instant Personal Alerts	523
25.14	Full Duplex Call Follow On	523
25.15	The User Plane	523
25.15.1	Media Encoding	524
25.15.2	Media Burst Control Protocol	524
25.15.2.1	Message Encoding	524

25.15.2.2	Message Reliability	525
25.15.2.3	Message Types	525
25.15.2.4	Message Flow	526
25.16	Simultaneous PoC Sessions	527
25.17	Charging in PoC	528
26	Multimedia Telephony Services: PSTN/ISDN Simulation Services	529
26.1	Providing Audible Announcements	530
26.1.1	Announcement at the Time a Session is Being Established	530
26.1.2	Announcement During the Duration of the Session	535
26.1.3	Announcement at the End of the Session	535
26.1.4	Announcement When a Session is Rejected	535
26.2	Communication Diversion (CDIV) and Communication Forwarding	536
26.3	Communication Diversion Notification (CDIVN)	537
26.4	Conference (CONF)	539
26.5	Message Waiting Indication (MWI)	539
26.6	Originating Identification Presentation/Restriction (OIP, OIR)	542
26.7	Terminating Identification Presentation/Restriction (TIP, TIR)	543
26.8	Anonymous Communication Rejection (ACR) and Communication Barring (CB)	543
26.9	Advice of Charge (AoC)	545
26.10	Completion of Communications to Busy Subscriber (CCBS) and Completion of Communications on No Reply (CCNR)	548
26.11	Malicious Communication Identification (MCID)	549
26.12	Communication Hold (HOLD)	551
26.13	Explicit Communication Transfer (ECT)	553
26.14	User Settings in PSTN/ISDN Simulation Services	557
27	Voice Call Continuity	559
27.1	Overview of Voice Call Continuity	559
27.2	VCC Architecture	561
27.3	Registration	563
27.4	Call Origination and Anchoring	563
27.4.1	IMS Originated Call Leg	563
27.4.2	CS Originated Call Leg using CAMEL Services	564
27.4.3	CS Originated Call Leg using CAMEL and ISUP Call Diversion	566
27.5	Call Termination and Anchoring	567
27.5.1	IMS Terminated Call Leg	568
27.5.2	CS Terminated Call Leg	569
27.5.3	CS Originated Call is Terminated in the IMS using CAMEL	570
27.5.4	CS Originated Call is Terminated in the CS Domain	571
27.6	Domain Transfer	572
27.6.1	Transfer from the CS Domain to the IMS Domain	572
27.6.2	Transfer from the IMS Domain to the CS Domain	574

Appendix A List of IMS-related Specifications	577
A.1 Introduction	577
A.2 3GPP Specifications	577
A.3 ETSI NGN Specifications	578
A.4 OMA Specifications	578
References	589
Index	607

Foreword by Stephen Hayes

3GPP, or the Third Generation Partnership Project, was formed in late 1998 to specify the evolution of GSM into a third generation cellular system. Although much focus was placed on new higher bandwidth radio access methods, it was realized that the network infrastructure must also evolve in order to provide the rich services capable of taking advantage of higher bandwidths. The original GSM network infrastructure was very much circuit- and voice-centric. Although data capabilities were added over time the system retained much of its circuit-switched heritage and its inherent limitations. A new approach was needed.

IMS, or the IP Multimedia Subsystem, represented that new approach. The development of IMS was very much a collaborative effort between the leading cellular standards organization (3GPP) and the leading Internet standards organization (IETF). IETF provided the base technology and protocol specifications, while 3GPP developed the architectural framework and protocol integration required to provide the capabilities expected of a world-class mobile system, such as inter-operator roaming, differentiated QoS, and robust charging.

Since the initial specification of IMS, IMS has been adopted by 3GPP2 (the other major cellular standards organization) and it is the leading contender as the base of the ITU work on Next Generation Networks. In the upcoming decades an understanding of IMS will be as important and fundamental for the well-rounded telecom engineer as ISUP knowledge was in previous decades.

IMS is a system. It is designed to provide robust multimedia services across roaming boundaries and over diverse access technologies. To understand IMS, you must understand the underlying protocols and how IMS uses them within its architectural framework. This book facilitates that understanding by explaining first the underlying protocols, such as SIP, and then explaining how IMS makes use of those protocols. This approach allows the user to easily grasp the complex relationship between the protocols and entities as developed in the IETF and their usage and extensions as defined in IMS.

The two authors are uniquely qualified to explain not just the inner workings of IMS but also the rationale and tradeoffs behind the various design choices. Miguel Angel García-Martín was and still is a key contributor within 3GPP. He was one of the principal designers of IMS and authored the initial protocol requirements draft as well as other 3GPP-specific SIP drafts and RFCs. Gonzalo Camarillo was similarly a key contributor within IETF, where he is currently a SIPPING WG co-chair. He has written many RFCs that are key components of IMS. Both authors have been involved with IMS since its inception and do a good job of explaining not only what IMS is but also how it came to be.

Stephen Hayes
Chair – 3GPP Core Network

Foreword by Allison Mankin and Jon Peterson

The Session Initiation Protocol (SIP) is one of the most active initiatives underway in the Internet Engineering Task Force (IETF) today. While the IETF has standardized a number of Internet applications that have turned out to be quite successful (notably, email and the web), few efforts in the IETF have been as ambitious as SIP. Unlike previous attempts to bring telephony over the Internet, which relied extensively on the existing protocols and operational models of the Public Switched Telephone Network (PSTN), SIP elected to use the best parts of email and web technology as its building blocks, and to construct a framework for establishing real-time communication – be it voice, video, instant messaging, or what have you – that is truly native to the Internet.

SIP is a rendezvous protocol – a protocol that allows endpoints on the Internet to discover one another and negotiate the characteristics of a session they would like to share. It converges on the best way for users to communicate, given their preferences, and the capabilities of devices they have at their disposal. Even though it establishes sessions over numerous communications media, it allows policies and services to be provided at the rendezvous level, which greatly simplifies the way end-users and operators manage their needs.

This approach has garnered the attention of almost all of the major vendors and service providers interested in telephony today. But the adoption of SIP by 3GPP has been a special, definitive success for SIP in the global marketplace. 3GPP promises to place SIP firmly in the hands of millions of consumers worldwide, ushering in a whole new paradigm of Internet-based mobile multimedia communications. The IP Multimedia Subsystem (IMS) of 3GPP is the core of this strategy, and it is a SIP-based core.

The IETF has created and continues to develop SIP, and the other protocols for real-time communication and infrastructure: RTP, SDP, DNS, Diameter, ... As 3GPP builds its successive IMS releases, towards a SIP-based multimedia Internet, IETF and 3GPP have grown into a close, working partnership, initiated by our liaison (RFC3113). Both committed to the Internet style afforded by SIP, two worlds with very different perspectives, the 3GPP world of mobile wireless telephony, and the IETF world of the packet Internet, have learned each other's considerations. There remain some differences, in the security models, in some aspects of network control. It's a tribute to the communications, the design work, and not least, to work by the authors of the present volume, that such differences have nonetheless resulted in interoperable SIP, SIP with a coherent character.

Gonzalo Camarillo has been one of the protagonists in SIP's development. In addition to his work editing the core SIP specification (RFC3261) within the IETF, Gonzalo has

chaired the SIPPING Working Group of the IETF (which studies new applications of SIP) and authored numerous documents related to interworking SIP with the traditional telephone network, ensuring that SIP is IPv6 compliant, and using SIP in a wireless context.

Miguel A. García-Martín is one of the principal designers of the IMS, and has also somehow found the time to be one of the main voices for 3GPP within the IETF SIP community. The application of SIP to the mobile handset domain gave rise to numerous new requirements for SIP functionality, many of which would not be obvious to designers unfamiliar with the intricacies of wireless roaming, bandwidth constraints, and so on. As such, Miguel provided some very valuable guidance to the IETF which ensured that SIP is well-tooled to one of its most promising applications.

This book is a milestone presenting the first in-depth coverage of the 3GPP SIP architecture. It is difficult to overestimate the importance of the 3GPP deployment, and this book will position readers to participate in the engineering of that network.

Allison Mankin
Jon Peterson
Directors of the Transport Area of the IETF

About the Authors

Gonzalo Camarillo

Gonzalo Camarillo leads the Multimedia Signaling Research Laboratory of Ericsson in Helsinki, Finland. He is an active participant in the IETF, where he has authored and co-authored several specifications used in the IMS. In particular, he is a co-author of the main SIP specification, RFC 3261. Gonzalo is a member of the IAB (Internet Architecture Board) and the IETF liaison manager to 3GPP. In addition, he co-chairs the IETF SIPPING working group, which handles the requirements from 3GPP and 3GPP2 related to SIP, and the IETF HIP (Host Identity Protocol) working group, which deals with lower-layer mobility and security. He is the Ericsson representative in the SIP Forum and is a regular speaker at different industry conferences. During his stay as a visitor researcher at Columbia University in New York, USA, he published a book entitled “SIP Demystified”. Gonzalo received an M.Sc. degree in Electrical Engineering from Universidad Politecnica de Madrid, Spain, and another M.Sc. degree (also in Electrical Engineering) from the Royal Institute of Technology in Stockholm, Sweden. He is currently continuing his studies as a Ph.D. candidate at Helsinki University of Technology, in Finland.

Miguel A. García-Martín

Miguel A. García-Martín is a System Expert of Ericsson in Madrid, Spain. In the past he has been a Senior Standardization Specialist in the Industry Environment unit of Nokia Siemens Networks in Espoo, Finland and a Principal Research Engineer in the Networking Technologies Laboratory of Nokia Research Center in Helsinki, Finland. Before joining Nokia, Miguel held several positions with Ericsson Finland and Ericsson Spain related to the development of IMS. Miguel is an active participant of the IETF, and for a number of years has been a key contributor in 3GPP. For some time he has also been participating in the specification of NGN in ETSI. In the IETF, he has authored and co-authored several specifications related to the IMS. In 3GPP, he has been a key contributor to the development of the IMS standard. Miguel is also a regular speaker at different industry conferences. Miguel received a B. Eng. degree in Telecommunications Engineering from Universidad de Valladolid, Spain.

Preface to the Third Edition

When 3GPP started standardizing the IMS a few years ago, most analysts expected the number of IMS deployments to grow dramatically as soon the initial IMS specifications were ready (3GPP Release 5 was functionally frozen in the first half of 2002 and completed shortly after that). While those predictions have proven to be too aggressive owing to a number of upheavals hitting the ICT (Information and Communications Technologies) sector, we are now seeing more and more commercial IMS-based service offerings in the market. At the time of writing (May 2008), there are over 30 commercial IMS networks running live traffic, adding up to over 10 million IMS users around the world; the IMS is being deployed globally. In addition, there are plenty of ongoing market activities; it is estimated that over 130 IMS contracts have been awarded to all IMS manufacturers. The number of IMS users will grow substantially as these awarded contracts are launched commercially. At the same time, the number of IMS users in presently deployed networks is steadily increasing as new services are introduced and operators running these networks migrate their non-IMS users to their IMS networks.

On the terminal side, estimations indicate that more than 100 million mobile terminals with support for at least one IMS service will be shipped in 2008. In addition, the fixed version of IMS has made a big effort to be compatible with any standard off-the-shelf SIP-based phone, making the number of available fixed terminals suitable for IMS close to unlimited.

The most common applications running on IMS commercial deployments are IP telephony in fixed and mobile networks, IP centrex, messaging (including text, pictures, and videos), Push-to-talk, video sharing, and presence. However, there is much ongoing work on additional IMS applications. In particular, applications involving machine-to-machine communications (e.g., in sensor networks) are getting much attention. At present, most of the already deployed IMS networks run a specific service instead of using the IMS as the service delivery platform, as the IMS was once envisioned. We expect the market to evolve towards multi-service IMS networks in the following years.

When it comes to current standardization activities in the IMS area, the most relevant activities have to do with multi-access networks. Current IMS networks provide service to endpoints that use several different types of fixed and mobile access technologies, such as WCDMA, WLAN, ADSL and PacketCable. In order to have all these different accesses seemlessly integrated in the IMS architecture, there is still some work to be done to coordinate the specifications coming from 3GPP, 3GPP2, TISPAN, and PacketCable. In the past, there have been a few overlaps between specifications from different organizations. The idea is to minimize those overlaps by clarifying which parts of the architecture each organization should be working on. Additional standardization work includes simplifications of the IMS

architecture and the development of new extensions to implement new services or provide new functionality.

The third edition of this book includes a great deal of new material. We have added new chapters discussing emergency calls, service configuration (XCAP and OMA XDM 2.0), conferencing, and Voice Call Continuity (VCC). We have updated the description of the PCC (Policy and Charging Control) architecture to 3GPP Release 7, OMA Presence 2.0, and PoC (Push-to-talk over Cellular) to OMA PoC 2.0. We have added detailed flow descriptions to each multimedia telephony service (or PSTN/ISDN simulation services). We have included discussions on GRUUs (Globally Routable User agent URIs) and their use in the IMS. We have described new NAT traversal techniques such as ICE (Interactive Connectivity Establishment), protocols such as STUN and TURN, and how they apply to the IMS. We have introduced new service and application identification concepts such as ICSI (IMS Communication Services Identification) and IARI (IMS Application Reference Identifier). We have included a description of combinational services. The Security in the IMS chapter has been updated with HTTP Digest Access Authentication and TLS, Early IMS Security Solution, NASS-IMS bundled authentication, and TLS for Network Security. The Instant Messaging on the Internet chapter now discusses the ‘isComposing’ feature, MESSAGE URI-list services, chat rooms, and file transfer operations with SIP and SDP.

Based on feedback from instructors and lecturers, we have also improved the companion website to this book. We have made all the figures of this book available to our readers in a high-quality format, so that they can be easily imported to slide shows and presentations. Please refer to the companion web site at:

<http://www.wiley.com/go/camarillo>

Overall, we have put a considerable amount of effort into updating the book and creating this third edition. We hope our readers find the new material interesting and easy to understand, and continue finding the book a valuable reference on the IMS and its related Internet technologies.

Preface to the Second Edition

The pace at which new IMS-related technologies have been developed in the last year has been impressive. Based on the deployment experiences of their members and on feedback from several organizations, 3GPP and 3GPP2 have worked extensively to update the IMS architecture so that it supports a wide range of new services.

While many of these updates consist of extensions to provide more functionality, some of them consist of simplifications to the IMS architecture. These simplifications make the IMS architecture more robust and reliable, or increase the performance of services implemented on top of it.

Examples of organizations that provide feedback to 3GPP and 3GPP2 on how to evolve the IMS are the OMA (Open Mobile Alliance) and the standardization bodies involved in the developing of NGN (Next Generation Networks). These organizations use the IMS as a base to provide different types of services.

The second edition of this book, in addition to describing updates to the IMS architecture, includes extensive discussions on the NGN architecture and the services it provides, and on the OMA PoC (Push-to-talk over Cellular) service. We are confident that the reader will find the chapters on these IMS-based services useful.

From the feedback received on the first edition, it seems that many readers found the structure of the book novel and useful. Readers agreed that first describing how a technology works on the Internet before discussing how it applies to the IMS provides a wider perspective than studying the technology in the IMS context alone.

Of course, we have also updated the sections dealing with Internet technologies. These sections include some of the latest protocol extensions developed in the IETF.

Based on the feedback received during the IMS seminars we have given around the world, we have clarified those concepts which were difficult to understand in the first edition.

Finally, also new to the second edition is a companion website on which instructors and lecturers can find electronic versions of the figures. Please go to

<http://www.wiley.com/go/camarillo>

Preface to the First Edition

The IMS (IP Multimedia Subsystem) is the technology that will merge the Internet with the cellular world. It will make Internet technologies, such as the web, email, instant messaging, presence, and videoconferencing available nearly everywhere. We have written this book to help engineers, programmers, business managers, marketing representatives, and technically aware users understand how the IMS works and the business model behind it.

We have distributed the topics in this book into four parts: an introduction, the signaling plane in the IMS, the media plane in the IMS, and IMS service examples. All four parts follow a similar structure; they provide both Internet and IMS perspectives on each topic.

First, we describe how each technology works on the Internet. Then, we see how the same technology is adapted to work in the IMS. Following these two steps for each technology provides the reader with a wider perspective. So, this book is not a commented version of the IMS specifications. It covers a much broader field.

Reading this book will improve anyone's understanding of the Internet technologies used in the IMS. You will know how each technology is used on the Internet and which modifications are needed to make it work in the IMS. This way you will understand how the use of Internet technologies in the IMS will make it easy to take advantage of any current and future Internet service. Finally, you will appreciate how operators can reduce the operational cost of providing new services.

Engineers who are already familiar with the IMS or with any of the IMS-related Internet protocols will also benefit substantially from this book. This way, engineers from the IETF (Internet Engineering Task Force) will understand which special characteristics of the IMS makes it necessary to add or remove certain features from a few Internet protocols so that they can be used in the IMS. On the other hand, engineers from 3GPP (Third Generation Partnership Project) and 3GPP2 will gain a wider perspective on IMS technologies. In addition, any engineer who focuses on a specific technology will gain a better understanding of the system as a whole.

Readers who want to expand their knowledge of any particular topic will find multiple references to 3GPP and 3GPP2 specifications, ITU recommendations, and IETF RFCs and Internet-Drafts in the text. Moreover, Appendix A contains a list with all the 3GPP and 3GPP2 specifications that are relevant to the IMS.

Now, let us look at each part of this book. Part I provides an introduction to the IMS: its goals, its history, and its architecture. We highlight the gains the operators obtain from the IMS. Besides, we discuss what the user can expect from the IMS. In addition, we describe how existing services, such as GPRS, WAP, SMS, MMS, and video-telephony over circuits relate to the IMS.

Part II deals with the signaling plane of the IMS, which includes protocols, such as SIP (Session Initiation Protocol), SDP (Session Description Protocol), Diameter, IPsec, and

COPS (Common Open Policy Service). As we said earlier, we describe each protocol as it is used on the Internet and, then, as it is used in the IMS.

Part III describes the media plane of the IMS. We describe how to convert audio and video into a digital form and how to transport it using protocols, such as RTP (Real-Time Transport Protocol) and RTCP (RTP Control Protocol). Furthermore, we introduce Internet protocols such as DCCP (Datagram Congestion Control Protocol) and SRTP (Secure RTP) that are not currently used in the IMS, but might be in the future.

Finally, Part IV provides IMS service examples, such as presence, instant messaging, and Push-to-talk. These examples illustrate how to build meaningful services using the technologies described in Parts II and III.

Essentially, this book is useful to a wide range of technical and business professionals because it provides a thorough overview of the IMS and its related technologies.

Acknowledgements

Without the encouragement we received from Stephen Hayes we would not have written this book. He was the first to see the need for a book on the IMS that provided the IETF perspective in addition to the 3GPP and 3GPP2 perspectives. In addition, he and Allison Mankin did an outstanding job coordinating the IMS standardization from 3GPP and from the IETF, respectively.

Once we decided, pushed by Stephen, to start writing this book, our management in Ericsson Finland fully supported us in this endeavor. In particular, Stefan Von Schantz, Christian Engblom, Jussi Haapakangas, Rolf Svanback, and Markku Korpi understood from the beginning the importance of the IMS and of spreading knowledge about it.

Our technical reviewers helped us fix technical errors in early versions of the manuscript. Andrew Allen provided useful comments on the whole manuscript. Harri Hakala, Arto Mahkonen, Miguel Angel Pallares, Janne Suotula, Vesa Torvinen, Magnus Westerlund, Brian Williams, Oscar Novo, Jari Urpalainen, Joël Repiquet, Mari Melander, Hannes Tschofenig, Javier Pastor, Jan Holm, Ari Keränen, and Vesa Lehtovirta provided suggestions on different parts of the book. Takuya Sawada and Takuya Kashima performed a thorough review of the manuscript during its translation to Japanese. Anna Reiter provided guidance on language and writing style.

Our editor at John Wiley & Sons, Ltd, Mark Hammond, believed in this book from day one and supported us at every moment.

Part I

Introduction to the IMS

Before we look at how the IMS works in Parts II and III of this book we need to provide some background information on the IMS. This part (Part I) of the book will answer questions on, for example, what the IMS is, why it was created, what it provides, and which organizations are involved in its standardization. In addition, we will describe the IMS architecture and the design principles behind it.

Chapter 1

IMS Vision: Where Do We Want to Go?

Third generation (3G) networks aim to merge two of the most successful paradigms in communications: cellular networks and the Internet. The IP (Internet Protocol) Multimedia Subsystem (IMS) is the key element in the 3G architecture that makes it possible to provide ubiquitous cellular access to all the services that the Internet provides. Picture yourself accessing your favorite web pages, reading your email, watching a movie, or taking part in a videoconference wherever you are by simply pulling a 3G hand-held device out of your pocket. This is the IMS vision.

1.1 The Internet

The Internet has experienced dramatic growth over the last few years. It has evolved from a small network linking a few research sites to a massive worldwide network. The main reason for this growth has been the ability to provide a number of extremely useful services that millions of users like. The best known examples are the World Wide Web and email, but there are many more, such as instant messaging, presence, VoIP (Voice Over IP), videoconferencing, and shared whiteboards.

The Internet is able to provide so many new services because it uses open protocols that are available on the web for any service developer. Moreover, the tools needed to create Internet services are taught at university and are described in large numbers of books.

A widespread knowledge of Internet protocols has an important implication: people who develop new services are the ones who are going to use them. Let us say that a user is interested in chess and would like to play chess over the Internet. This user will be able to program a chess application and make it work over the Internet using an existing transport protocol.

On the other hand, if the protocols were not open and there were few individuals who had access to them, the person programming the chess application would be somebody with deep knowledge of the protocol but little of chess. It is not difficult to guess who would come up with the best chess program: the chess player who understands what to expect from a chess program or the protocol expert. In fact, this is what the Internet has achieved. The number of protocol experts is so high that there is always somebody within a given community

(e.g., the chess community) who understands the requirement of the community and the protocols that need to be involved.

1.2 The Cellular World

At present, cellular telephone networks provide services to over one billion users worldwide. These services include, of course, telephone calls, but are not limited to them. Modern cellular networks provide messaging services ranging from simple text messages (e.g., SMS (Short Messaging Service)) to fancy multimedia messages that include video, audio, and text (e.g., MMS (Multimedia Messaging Service)). Cellular users are able to surf the Internet and read email using data connections, and some operators even offer location services which notify users when a friend or colleague is nearby.

Still, cellular networks did not become so attractive to users only for the services they offered. Their main strength is that users have coverage virtually *everywhere*. Within a country, users can use their terminals not only in cities, but also in the countryside. In addition, there exist international roaming agreements between operators that allow users to access cellular services when they are abroad.

Reduction in terminal size also helped the spread of cellular networks. Old brick-like terminals gave way to modern small terminals that work for several days without having their batteries recharged. This allows people to carry their terminals everywhere with little difficulty.

1.3 Why do we need the IMS?

On the one hand, we have mentioned that the idea of the IMS is to offer Internet services everywhere and at any time using cellular technologies. On the other hand, we have also said that cellular networks already provide a wide range of services, which include some of the most successful Internet services, such as instant messaging. In fact, any cellular user can access the Internet using a data connection and in this way access any services the Internet may provide. So, what do we need the IMS for?

We need to further clarify what we mean by merging the Internet and the cellular worlds and what the real advantages of doing so are. To do that, we need to introduce the different domains in 3G networks, namely the circuit-switched domain and the packet-switched domain.

The circuit-switched domain is an evolution of the technology used in second generation (2G) networks. The circuits in this domain are optimized to transport voice and video, although they can also be used to transport instant messages.

Although circuit-switched technology has been in use since the birth of the telephone, the current trend is to substitute it with more efficient packet-switched technology. Cellular networks follow this trend and, as we said earlier, 3G networks have a packet-switched domain.

The packet-switched domain provides IP access to the Internet. While 2G terminals can act as a modem to transmit IP packets over a circuit, 3G terminals use native packet-switched technology to perform data communications. This way, data transmissions are much faster and the available bandwidth for Internet access increases dramatically. Users can surf the web, read email, download videos, and do virtually everything they can do over any other Internet connection, such as ISDN (Integrated Services Digital Network) or DSL (Digital Subscriber Line). This means that any given user can install a VoIP client in their 3G terminal

and establish VoIP calls over the packet-switched domain. Such a user can take advantage of all the services that service providers on the Internet offer, such as voicemail or conferencing services.

So, again the same question: why do we need the IMS, if all the power of the Internet is already available for 3G users through the packet-switched domain? The answer is threefold: QoS (Quality of Service), charging, and integration of different services.

The main issue with using the packet-switched domain to provide real-time multimedia services is that it provides a best-effort service without QoS; that is, the network offers no guarantees about the amount of bandwidth a user gets for a particular connection or about the delay the packets experience. Consequently, the quality of a VoIP conversation can vary dramatically throughout its duration. At a certain point the voice of the person at the other end of the phone may sound perfectly clear and instants later it can become impossible to understand. Trying to maintain a conversation (or a videoconference) with poor QoS can soon become a nightmare.

So, one of the reasons for creating the IMS was to provide the QoS required for enjoying, rather than suffering, real-time multimedia sessions. The IMS takes care of synchronizing session establishment with QoS provision so that users have a predictable experience.

Another reason for creating the IMS was being able to charge multimedia sessions appropriately. A user involved in a videoconference over the packet-switched domain usually transfers a large amount of information (which consists mainly of encoded audio and video). Depending on the 3G operator, the transfer of such an amount of data may generate large expenses for the user, since operators typically charge by the number of bytes transferred. The user's operator cannot follow a different business model to charge the user because the operator is not aware of the contents of those bytes: they could belong to a VoIP session, to an instant message, to a web page, or to an email.

On the other hand, if the operator is aware of the actual service that the user is using, the operator can provide an alternative charging scheme that may be more beneficial for the user. For instance, the operator might be able to charge a fixed amount for every instant message, regardless of its size. In addition, the operator may charge for a multimedia session based on its duration, independently of the number of bytes transferred.

The IMS does not mandate any particular business model. Instead, it lets operators charge as they think most appropriate. The IMS provides information about the service being invoked by the user, and with this information the operator decides whether to use a flat rate for the service, apply traditional time-based charging, apply QoS-based charging, or perform any new type of charging. As a clarification, by service, in this charging context, we refer to any value offered to the user (e.g., a voice session, an audio/video session, a conference bridge, an instant message, or the provision of presence information about co-workers).

Providing integrated services to users is the third main reason for the existence of the IMS. Although large equipment vendors and operators will develop some multimedia services, operators do not want to restrict themselves to these services. Operators want to be able to use services developed by third parties, combine them, integrate them with services they already have, and provide the user with a completely new service. For example, an operator may have a voicemail service able to store voice messages and a third party develops a text-to-speech conversion service. If the operator buys the text-to-speech service from the third party, it can provide voice versions of incoming text messages for blind users.

The IMS defines the standard interfaces to be used by service developers. This way, operators can take advantage of a powerful multi-vendor service creation industry, avoiding sticking to a single vendor to obtain new services.

Furthermore, the aim of the IMS is not only to provide new services but to provide all the services, current and future, that the Internet provides. In addition, users have to be able to execute all their services when roaming as well as from their home networks. To achieve these goals the IMS uses Internet technologies and Internet protocols. So, a multimedia session between two IMS users, between an IMS user and a user on the Internet, and between two users on the Internet is established using exactly the same protocol. Moreover, the interfaces for service developers we mentioned above are also based on Internet protocols. This is why the IMS truly merges the Internet with the cellular world; it uses cellular technologies to provide ubiquitous access and Internet technologies to provide appealing services.

1.4 Relation between IMS and non-IMS Services

We have just explained that the IMS is needed to provide Internet services (including real-time multimedia services) with an acceptable QoS at an acceptable price. Yet many such services can be provided outside the IMS as well. Two users can establish a videoconference over the circuit-switched domain and send each other multimedia messages using MMS. At the same time they can surf the web and check email over the packet-switched domain (e.g., GPRS (General Packet Radio Service)). They can even access a presence server on the Internet to check the availability of more people who may want to join the videoconference.

Given that all the services just described can be provided with an excellent QoS with no IMS at all, then what does the IMS really provide?

First of all, the IMS provides all the services using packet-switched technology, which is generally more efficient than circuit-switched technology. Nevertheless, the real strength of the IMS when compared with the situation above is that the IMS creates a service environment where any service can access any aspect of the session. This allows service providers to create far richer services than in an environment where all the services are independent of one another.

For example, a service could insert an announcement in a conference based on an event that happens on the Internet, like the change of the presence state of a colleague from busy to available. Another service could, for instance, display on the user's screen the web page of the person who is calling every time a call is received. Moreover, the same service could automatically set the user's presence status to busy and divert incoming calls to an email address instead of to the typical voicemail.

When services in the network can access all the aspects of a session, they can perform many operations (e.g., changing the presence status of the user) without sending any data over the air to the terminal. Spare radio capacity can be used to provide a higher QoS to existing users or to accommodate more users with the same QoS.

Another important advantage of the IMS is that it does not depend on the circuit-switched domain. This way, interworking with devices with no access to this domain, such as laptops connected to the Internet using any videoconferencing software, becomes trivial. This increments dramatically the number of people IMS users are able to communicate with using all types of media.

Chapter 2

The History of the IMS Standardization

In Chapter 1 we mentioned that the IMS (IP Multimedia Subsystem) uses Internet protocols. When the IMS needs a protocol to perform a particular task (e.g., to establish a multimedia session), the standardization bodies standardizing the IMS take the Internet protocol intended for that task and specify its use in the IMS. Still, no matter how simple this may sound, the process of choosing protocols to be used in the IMS can sometimes get tricky. Sometimes, the Internet protocol that is chosen lacks some essential functionality, or does not even exist at all. When this happens the IMS standardization bodies contact the standardization body developing Internet protocols to work together on a solution. We will cover this collaboration in Section 2.5. Nevertheless, before jumping into that we will introduce in Section 2.1 all the standardization bodies involved in IMS development. We need to know who is who and which functions of the IMS each of them performs.

2.1 Relations between IMS-related Standardization Bodies

The ITU (International Telecommunication Union) IMT-2000 (International Mobile Telecommunications-2000) is the global standard for 3G networks. IMT-2000 is the result of the collaboration between different standards bodies. It aims to provide access to telecommunication services using radio links, which include satellite and terrestrial networks.

We will focus on two of the standard bodies involved in IMT-2000: 3GPP (Third Generation Partnership Project) and 3GPP2 (Third Generation Partnership Project 2). However, they are not the only ones working within IMT-2000. Other bodies, such as the ITU-R (ITU-Radiocommunication Sector), for instance, are also involved in IMT-2000 but in different areas from the IMS.

Both 3GPP and 3GPP2 have standardized their own IMS. The 3GPP IMS and the 3GPP2 IMS are fairly similar, but, nevertheless, have a few differences, mostly related to the difference in the cellular aspects of 3GPP and 3GPP2 cellular networks.

An important similarity between the 3GPP IMS and the 3GPP2 IMS is that both use Internet protocols, which have been traditionally standardized by the IETF (Internet Engineering Task Force). Consequently, both 3GPP and 3GPP2 collaborate with the IETF in developing protocols that fulfill their requirements. The following sections introduce the IETF, 3GPP, and 3GPP2 and provide a brief history of the IETF-3GPP/3GPP2 collaboration.

In addition to the standard bodies we have just mentioned, OMA (Open Mobile Alliance [226]) plays an important role in developing IMS services. While 3GPP and 3GPP2 have standardized (or are standardizing) a few IMS services, such as basic video calls or conferencing, OMA focuses on the standardization of service enablers on top of the IMS (of course, other standard bodies and third parties besides OMA may also develop services and service enablers for the IMS).

Lately, additional standardization bodies have come on the scene since IMS made its debut in the fixed broadband access arena. We are referring to Next Generation Networks (NGN) for which IMS forms a substantial part.

In 2004 the ITU-T created an NGN Focus Group (NGN-FG) that for a couple of years studied and advanced the specification work of Next Generation Networks for fixed line accesses based on IMS. In Europe, in 2004, the European Telecommunications Standards Institute (ETSI) created the Telecoms and Internet converged Services and Protocols for Advanced Networks (TISPAN) technical committee, with the goal of standardizing a Next Generation Network for fixed network access based on IMS. ETSI TISPAN is contributing to 3GPP to maintain a single set of IMS specifications. At the end of 2007, the common IMS parts of ETSI TISPAN were transferred to 3GPP and the standardization of these common IMS parts only take place in 3GPP.

In North America, the Alliance for Telecommunications Industry Solutions (ATIS), also in 2004, created the NGN Focus Group to study the applicability of NGN and IMS to North American fixed access networks. The three standardization bodies keep synchronized the definition of NGN and the applicability of IMS to fixed access networks. In addition, they also bring new requirements to 3GPP and 3GPP2 to support fixed broadband access to IMS.

But this is not all. In North America, the relevant initiative for the cable industry is the PacketCable™ initiative led by CableLabs. The PacketCable 2.0 set of specifications defines an application of IMS to cable networks for providing multimedia services. PacketCable has been contributing to 3GPP to maintain a single set of IMS specifications.

2.2 Internet Engineering Task Force

The Internet Engineering Task Force (IETF) is a loosely self-organized collection of network designers, operators, vendors, and research institutions that work together to develop the architecture, protocols, and operation of the public Internet. The IETF is a body that is open to any interested individual. It is not a corporation and, therefore, does not have a board of directors, members, or dues.

The IETF is the standardization body that has developed most of the protocols that are currently used on the Internet. The IETF does not standardize networks, architectures combining different protocols, the internal behavior of nodes, or APIs (Application Programming Interfaces). The IETF is the protocol factory for IP-related protocols.

2.2.1 Structure of the IETF

Work in the IETF is organized in working groups. Each working group is chartered to perform specific tasks, such as the delivery of a precise set of documents. Each working group has from one to three chairs, who ensure that the working group completes its chartered tasks in time. Working groups have a temporary lifetime, so, once they have delivered their documents, either they are rechartered or they cease to exist. Figure 2.1 shows a few, but not

all, of the working groups in the IETF; there are more than 100 active working groups in the IETF. The complete up-to-date list of active working groups is available at:

<http://www.ietf.org/html.charters/wg-dir.html>

Working groups get an acronym name that identifies the chartered task. For instance, SIPPING is the acronym for Session Initiation Protocol Investigation, SIMPLE is the acronym for SIP for Instant Messaging and Presence Leveraging Extensions, and AAA is the acronym for Authentication, Authorization and Accounting.

A collection of working groups form an Area Directorate. Traditionally most of the working groups of interest for the IMS have been part of the Transport Area, but some groups were included in the Application Area or some other area. In March 2006 the IETF created a new Real-Time Applications and Infrastructure (RAI) Area, whose main purpose is to agglutinate all the working groups working around real-time communications, for example, all the SIP-related working groups.

There are currently eight areas in the IETF, as illustrated in Figure 2.1. Note that not all the IETF working groups are shown in Figure 2.1

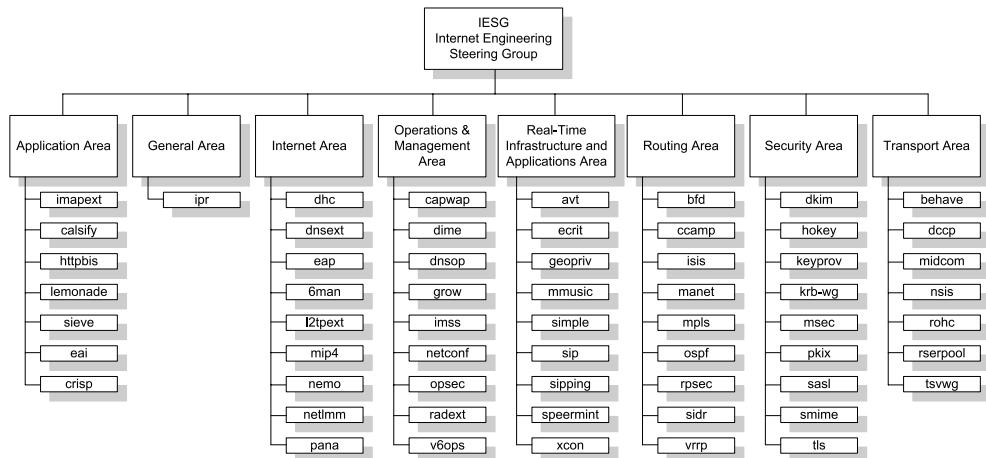


Figure 2.1: The structure of the IETF

Each area has one or two area directors who, together with the IETF chairman, form the IESG (Internet Engineering Steering Group). The IESG is the technical management team of the IETF. They decide on which areas the IETF should work and review all the specifications that are produced.

The following web pages contain the complete list of the working groups of all areas and the charter of the SIPPING working group, respectively:

<http://www.ietf.org/html.charters/wg-dir.html>

<http://www.ietf.org/html.charters/sipping-charter.html>

The IAB (Internet Architecture Board) is the body that provides technical leadership and handles appeals. Its web page is at:

<http://www.iab.org/>

2.2.2 Working Group Operations

The technical work in the IETF is done within the working groups. Working groups do not have any kind of membership; they are formed by a number of volunteers who work as individuals. That is, they do not represent their companies when working for the IETF.

Most of the technical discussions within a working group take place in its mailing list. Even the decisions made at face-to-face meetings (held three times a year) have to be confirmed in the mailing list.

The technical documents used within the working groups are called Internet-Drafts. There are two types: individual submissions and working group items. Individual submissions are technical proposals submitted by an individual or individuals. If the working group decides that an individual submission is a good starting point to work on a particular topic, it becomes a working group item.

Individual submissions and working group items can be distinguished by the name of the file where they are stored. Individual submissions start with:

`draft-author's_name`

while working group items start with:

`draft-ietf-name_of_the_working_group`

A list of all the Internet-Drafts can be found at:

<http://www.ietf.org/internet-drafts/>

When a working group feels that a working group item is ready for publication as an RFC (Request for Comments) the working group chairs send it to the IESG. The IESG may provide feedback to the working group (e.g., ask the working group to change something in the draft) and, eventually, decides whether or not a new RFC is to be published.

Although most of the Internet-Drafts that the IESG receives come from working groups, an individual may also submit an Internet-Draft to the IESG. This usually happens with topics that are not large enough to grant the creation of a working group, but which, nevertheless, are of interest to the Internet community.

It is important to note that Internet-Drafts, even if they are working group items, represent work in progress and should only be referenced as such. Internet-Drafts are temporary documents that expire and cease to exist six months after they have been issued. They can change at any time without backward compatibility issues with existing implementations being taken into consideration. Only when a particular Internet-Draft becomes an RFC can it be considered a stable specification.

2.2.3 Types of RFCs

The technical documents produced by the IETF are called RFCs. According to the contents of the document there are three main types of RFCs:

- standards-track RFCs;
- non-standards-track RFCs;
- BCP (Best Current Practice) RFCs.

Standards-track RFCs typically define protocols and extensions to protocols. According to the maturity of the protocol there are three levels of standards-track RFCs: proposed standard, draft standard, and Internet standard. Standards-track specifications are supposed to advance from proposed to draft and, finally, to Internet standard as they get more and more mature. An important requirement in this process is that a particular specification is implemented by several people to show that different independently built implementations that follow the same specification can successfully inter-operate.

Nevertheless, in practice, only a few RFCs reach the draft standard level and even fewer become Internet standards. At present, the specifications of many protocols that are used extremely frequently on the Internet are proposed standards.

There are three types of non-standards-track RFCs: experimental, informational, and historical (these are called *historic* RFCs). Experimental RFCs specify protocols with a very limited use, while informational RFCs provide information for the Internet community about some topic, such as a requirements document or a process. When a standards-track RFC becomes obsolete, it becomes a historic RFC.

When a document is published as an RFC, a sequential RFC number is permanently assigned to it, independently of the category of the RFC. In addition to the RFC number, a document may also get an additional number in the BCP series or STD series, depending on the category. For example, the Internet Protocol (IP) specified in RFC 791 [256] is also STD 5.

BCP RFCs record the best current practice known to the community for performing a particular task. They may deal with protocol issues or with administrative issues.

Figure 2.2 shows the relations between all the RFC types. A list of all the RFCs published so far and their status can be fetched from:

http://www.ietf.org/iesg/1rfc_index.txt

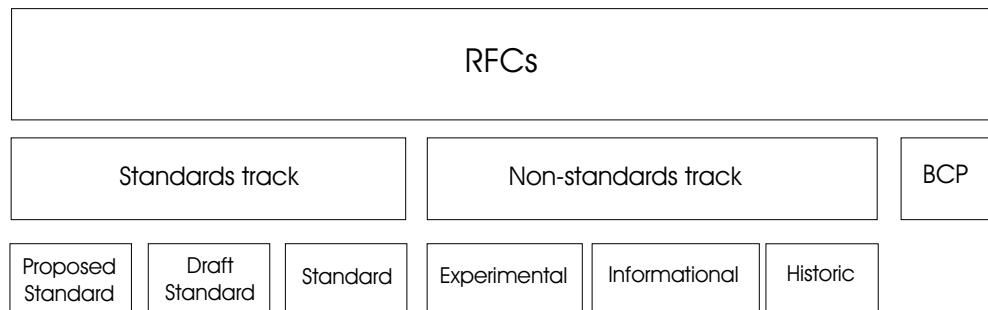


Figure 2.2: RFC types

RFCS can be downloaded from the following web page by just entering the RFC number:

<http://www.ietf.org/rfc.html>

In addition, the RFC Editor offers a web page that allows us to search for RFCs by title, number, author and keywords:

<http://www.rfc-editor.org/rfcsearch.html>

2.3 Third Generation Partnership Project

The Third Generation Partnership Project (3GPP) was born in 1998 as a collaboration agreement between a number of regional telecommunications standards bodies, known as *organizational partners*. The current 3GPP organizational partners are:

1. ARIB (Association of Radio Industries and Business) in Japan,

<http://www.arib.or.jp/english/>

2. CCSA (China Communications Standards Association) in China,

<http://www.ccsa.org.cn/english/>

3. ETSI (European Telecommunications Standards Institute) in Europe,

<http://www.etsi.org/>

4. ATIS (Alliance for Telecommunications Industry Solutions) in the United States of America,

<http://www.atis.org/>

5. TTA (Telecommunications Technology Association) of Korea,

<http://www.tta.or.kr/English/>

6. TTC (Telecommunication Technology Committee) in Japan,

<http://www.ttc.or.jp/e/>

3GPP was originally chartered to develop globally applicable technical specifications and technical reports for a third generation mobile system based on GSM (Global System for Mobile communication). The scope has been reinforced to include maintenance and development of GSM specifications, including the supported and evolved radio networks, technologies, and packet access technologies.

Besides the organizational partners, *market representation partners* provide the partnership with market requirements. Market representation partners include, among others, the IMS Forum, the UMTS Forum, 3G Americas, the GSM Association, the Global mobile Suppliers Association, the TD-SCDMA Forum, and the IPv6 Forum.

3GPP maintains an up-to-date web site at:

<http://www.3gpp.org/>

2.3.1 3GPP Structure

3GPP is organized as a Project Co-ordination Group (PCG) and Technical Specification Groups (TSGs), as illustrated in Figure 2.3. The PCG is responsible for the overall management of 3GPP, time plans, allocation of work, etc. The technical work is produced in the TSGs. At the moment there are four TSGs, responsible for the Core Network and Terminals (CT), System and Services Aspects (SA), GSM EDGE Radio Access Network (GERAN), and Radio Access Network (RAN). Each of the TSGs is further divided into Working Groups. Each of the Working Groups is allocated particular tasks. For instance, CT WG1 is responsible for all the detailed design of the usage of SIP and SDP in the IMS, CT WG3 for interworking aspects, and CT WG4 for all the detailed design of the usage of Diameter. SA WG1 is responsible for the requirements, SA WG2 for the architecture, SA WG3 for the security aspects, SA WG4 for the codecs, and SA WG5 for the operation and maintenance of the network, including charging aspects.

2.3.2 3GPP Deliverables

3GPP working groups do not produce standards. Instead, they produce Technical Specifications (TS) and Technical Reports (TR) that are approved by the TSGs. Once approved, these are submitted to the organizational partners to be submitted to their respective standardization processes. The final part of the process is in the organizational partners' hands when they approve the TSs or TRs as part of their standards procedures. As a result, there is a set of globally developed standards that are ready to be used in a particular region.

3GPP TSs and TRs are numbered according to a sequence of four or five digits that follow the pattern “xx.yyy”. The first two digits “xx” identify the series number, and the last two or three digits “yy” or “yyy” identify a particular specification within a series. For instance, 3GPP TS 23.228 [43] describes the architectural aspects of the IMS.

3GPP groups its specifications in what is called a *Release*. 3GPP Release 5 contains the first version of the IMS. 3GPP Releases 6, 7, and so on contain enhancements and additional functionality to the IMS. The reader must note that the IMS is just a fraction of the 3GPP deliverables in a particular Release, as there are other non-IMS specifications included in a 3GPP Release. 3GPP TSs and TRs include a version number that follows the pattern “x.y.z”, where “x” represents the 3GPP Release where the specification is published, “y” is the version number, and “z” is a sub-version number. So, 3GPP TS 23.228 version 5.8.0 means version 8.0 of the Release 5 version of TS 23.228.

3GPP TSs and TRs are publicly available at the 3GPP web site at either of the following URIs:

<http://www.3gpp.org/specs/specs.htm>

<http://www.3gpp.org/ftp/Specs/archive/>

2.4 Third Generation Partnership Project 2

If 3GPP was created to evolve GSM specifications into a third-generation cellular system, the Third Generation Partnership Project 2 (3GPP2) was born to evolve North American and Asian cellular networks based on ANSI/TIA/EIA-41 standards and CDMA2000® radio access into a third-generation system. 3GPP2, like 3GPP, is a partnership project whose members are also known as *organizational partners*. The current list of organizational

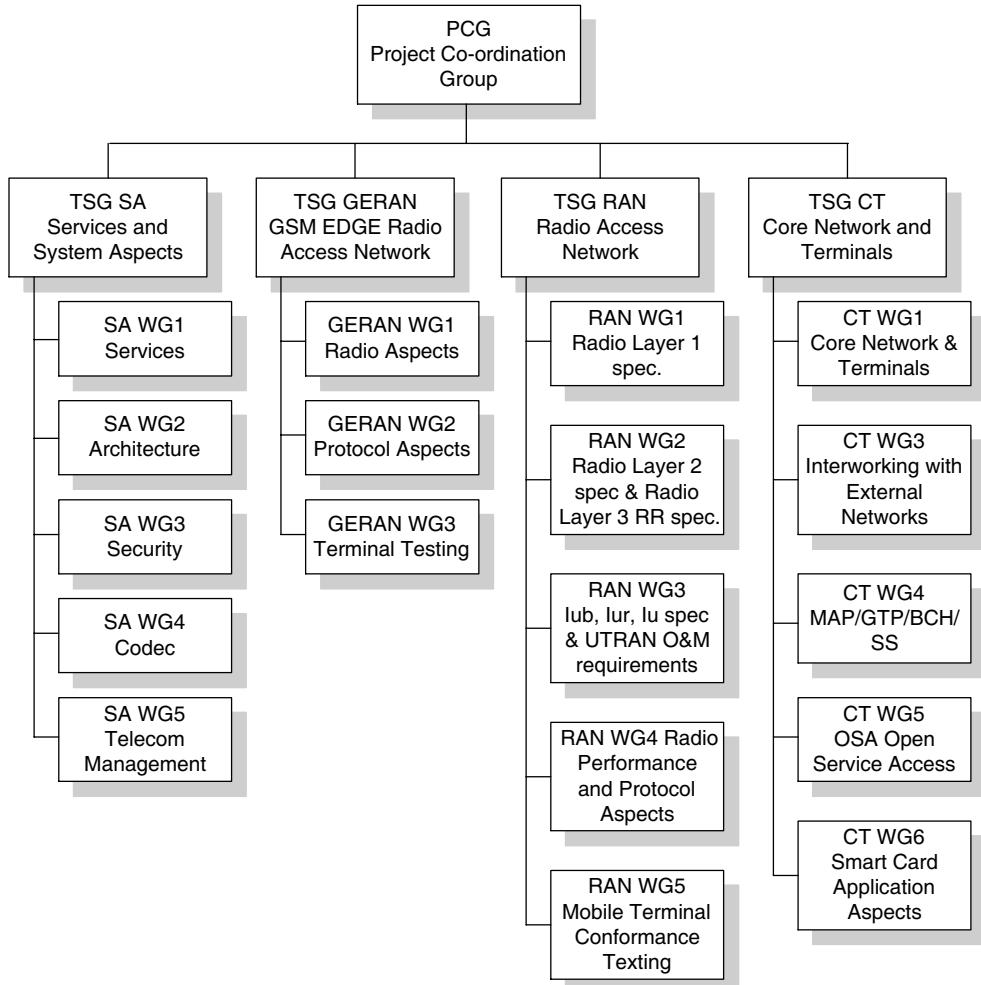


Figure 2.3: The structure of 3GPP

partners include ARIB (Japan), CCSA (China), TIA (Telecommunications Industry Association) (North America), TTA (Korea), and TTC (Japan). Probably the reader has noticed that most of them are also organizational partners of 3GPP.

Like 3GPP, 3GPP2 gets market requirements and advice from *market representation partners*. At the moment the list includes the IPv6 Forum, the CDMA Development Group, and the International 450 Association.

2.4.1 3GPP2 Structure

The 3GPP2 structure mimics the structure of 3GPP, as illustrated in Figure 2.4. The Steering Committee (SC) is responsible for the overall standardization process and the planning. The technical work is done in Technical Specification Groups (TSGs). TSG-A is focused

on the Access Networks Interfaces, TSG-C on CDMA2000 technology, TSG-S on Services and System Aspects, and TSG-X on Intersystems Operations. TSG-X was born as a merger between the former TSG-N (Core Networks) and TSG-P (Packet Data) TSGs, and devotes itself to Core Networks.

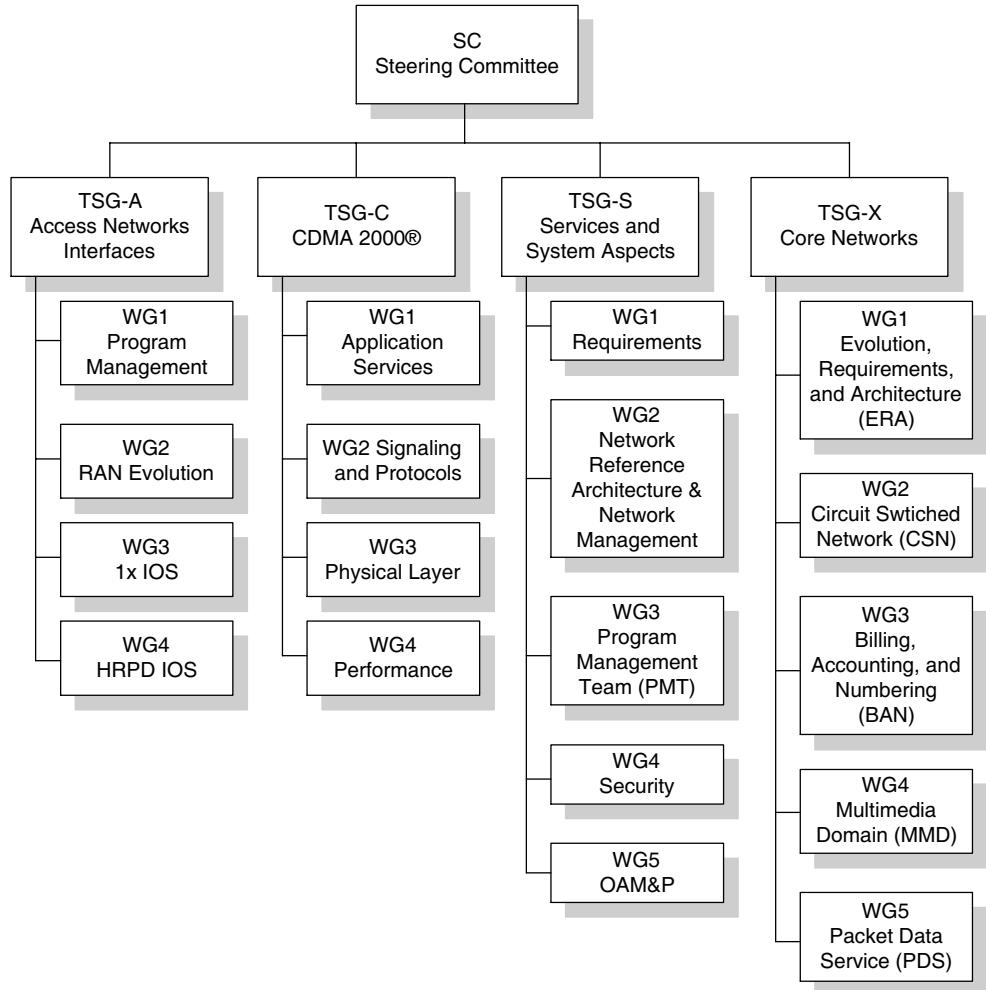


Figure 2.4: The structure of 3GPP2

2.4.2 3GPP2 Deliverables

Like 3GPP, 3GPP2 does not produce standards but, instead, Technical Specifications and Technical Reports. The documents are created by the TSGs and approved by the SC. Then, they are submitted to the organizational partners to be submitted to their respective standardization processes.

3GPP2 TSs and TRs are numbered with a sequence of letters and digits that follows the scheme “A.Bcccc[-ddd]-X” version “y.z” where “A” is a letter that represents the name of the

TSG that delivers the document, “B” can be a ‘P’, ‘R’, or ‘S’ letter to indicate a project, report or specification, respectively. “cccc” is a sequential number allocated to the document. An optional “ddd” sequence of digits is used for multi-part documents. The “X” letter identifies the revision, where “0” is the initial release, “A” is the first revision, and so on. The version number follows the specification and indicates a major and minor version. For instance, the specification X.S0013-002-A v1.0 represents the IP Multimedia Subsystem (IMS) Stage 2, revision A, version 1.0.

3GPP2 TSs and TRs are publicly available at the 3GPP2 web site at:

http://www.3gpp2.org/Public_html/specs/

Since 3GPP2 IMS specifications are based on corresponding 3GPP IMS ones, we focus on the IMS defined by 3GPP. Sometimes, we will highlight differences between the networks, when those differences are relevant to the discussion.

2.5 IETF-3GPP/3GPP2 Collaboration

As we mentioned in Chapter 1 the IMS aims to use Internet protocols. However, some of the protocols chosen for use in the IMS architecture were not completely suitable for the IMS environment. There were even cases where the IETF did not have any solution at all to address some of the issues the IMS was facing.

One possibility would have been to take whatever IETF protocols were already there and modify them to meet the requirements of the IMS. However, the goal of 3GPP and 3GPP2 was clear. They wanted the IMS to use Internet technologies. This way they could take advantage of any future service created for the Internet. Modifying Internet protocols on their own was not an option. Instead, they established a collaboration with the IETF to make sure that the protocols developed there met their requirements.

The terms of this collaboration were documented in RFC 3113 [292] (3GPP-IETF) and in RFC 3131 [91] (3GPP2-IETF). Both 3GPP and 3GPP2 nominated liaisons with the IETF (Ileana Leuca, later Stephen Hayes, and later Hannu Hietalahti from 3GPP, and Tom Hiller and later A. C. Mahendran from 3GPP2), and the IETF nominated a liaison with them (Thomas Narten first and Gonzalo Camarillo later). In any case these collaborations took part mostly at the working group level, without involving the official liaisons most of the time: for example, groups of engineers discussing technical issues in mailing lists, IETF face-to-face meetings, and special workshops. 3G engineers collaborated in providing wireless expertise and requirements from the operators while IETF engineers provided protocol knowledge. The goal was to find solutions that addressed the requirements of the IMS and that, at the same time, were general enough to be used in other environments.

So far, several protocol specifications and protocol extensions have been published in the form of RFCs and Internet-Drafts as the fruit of this collaboration. Most of them do not need to mention the IMS, since they specify protocols with general applicability that are not IMS-specific at all.

The following sections provide a brief history of the areas where the IETF collaborated in developing protocols that are used in the IMS.

2.5.1 *Internet Area*

The area director driving the collaboration in the IETF Internet area was Thomas Narten. The main areas of collaboration were IPv6 and DNS (Domain Name System).

The IPv6 working group produced a specification (RFC 3316 [77]) that provides guidelines on how to implement IPv6 in cellular hosts. When such a host detects that it is using a GPRS access, it follows the guidelines provided in that specification. On the other hand, if the same host is using a different access (e.g., WLAN), it behaves as a regular Internet host. So, terminals behave differently depending on the type of access they are using, not on the type of terminals they are.

In the DNS area there were discussions on how to perform DNS server discovery in the IMS. It was decided not to use DHCP (Dynamic Host Configuration Protocol), but to use GPRS-specific mechanisms instead. At that point there was no working group agreement on stateless DNS server discovery procedures that could be used in the IMS.

2.5.2 Operations and Management Area

The main protocols in the IETF operations and management area where there was collaboration between 3GPP and the IETF were COPS (Common Open Policy Service) and Diameter. Both area directors, Bert Wijnen and Randy Bush, were involved in the discussions; Bert Wijnen in COPS-related discussions and Randy Bush in Diameter-related discussions. Bert Wijnen even participated in 3GPP CN3 meetings as part of this collaboration.

In the COPS area the IMS had decided to use COPS-PR in the *Go* interface, and so 3GPP needed to standardize the *Go* Policy Information Base (PIB). However, in the IETF it was not clear whether using COPS-PR for 3GPP's purposes was a good idea. After a lot of discussions the *Go* PIB was finally created (the IETF produced RFC 3317 [116] and RFC 3318 [293]).

In the Diameter area the IMS needed to define three Diameter applications to support the *Cx*, *Sh*, and *Ro* interfaces. Nevertheless, although new Diameter codes could only be defined in RFCs, there was not enough time to produce an RFC describing these Diameter applications and the new command codes that were needed. At last, the IETF agreed to provide 3GPP with a number of command codes (allocated in RFC 3589 [210]) to be used in 3GPP Release 5 with one condition: 3GPP needed to collaborate with the IETF on improving those Diameter applications until they became general enough. These resulted in 3GPP contributing to the IETF with the Diameter SIP Application [150] and Diameter Credit-Control Application [158]. The IMS is supposed to migrate to these new Diameter applications in future releases.

2.5.3 Transport Area

Collaboration in the transport area was mainly driven by 3GPP (not much from 3GPP2). Two people were essential to the collaboration in this area: Stephen Hayes, initial 3GPP liaison with the IETF and chairman of CN, and Allison Mankin, transport area director in the IETF. They ensured that all the issues related to signaling got the appropriate attention in both organizations.

Everything began when 3GPP decided that SIP was going to be the session control protocol in the IMS. At that point SIP was still an immature protocol that did not meet most of the requirements 3GPP had. At that time SIP was defined in RFC 2543 [161], but there was an Internet-Draft, commonly known as 2543bis, that had fixed some of the issues present in RFC 2543 and was supposed to become the next revision of the protocol specification. However, 2543bis only had two active editors (namely Henning Schulzrinne and Jonathan Rosenberg) and the 3GPP deadlines were extremely tough. A larger team was needed if the SIP working group, where SIP was being developed, wanted to meet those deadlines. That is

how Gonzalo Camarillo, Alan Johnston, Jon Peterson, and Robert Sparks were recruited to edit the SIP specification that resulted in the current RFC 3261 [286].

After very many emails, conference calls, and face-to-face meetings the main outcome of the team was RFC 3261 [286]. However, it was soon clear that 3GPP's requirements were not going to be met with a single protocol. The input 3GPP requirements to SIP were documented in RFC 4083 [202]. A few extensions were needed to fulfill them all. In fact, there were so many requirements and so many extensions needed that the SIP working group was overloaded (other working groups, like MMUSIC, SIMPLE, or ROHC, were also involved, but the main body of the work was tackled by SIP). A new process was needed to handle all of this new work.

The IETF decided to create a new working group to assist SIP in deciding how to best use its resources. The new working group was called SIPPING, and its main function was to gather requirements for SIP, prioritize them and send them to the SIP working group, which was in charge of doing the actual protocol work. This new process was documented in RFC 3427 [214].

At present, most of the protocol extensions related to session establishment needed by 3GPP are finished or quite advanced. As a consequence the 3GPP focus moved towards the SIMPLE working group, which develops SIP extensions for presence and instant messaging. 3GPP members actively participated in the development of the presence and instant messaging specifications, which were adopted by 3GPP, 3GPP2, the Open Mobile Alliance, and other SDOs.

2.6 Open Mobile Alliance

In June 2002, the Open Mobile Alliance (OMA) was created to provide interoperable mobile data services. A number of existing forums at that time, such as the WAP Forum and Wireless Village, were integrated into OMA. Nowadays, OMA includes companies representing most segments of industry. Vendors, service providers, and content providers are all represented in OMA. The OMA web site can be found here:

<http://www.openmobilealliance.org/>

OMA pays special attention to usability and to creating service enablers that allow interoperability of services. That is, OMA service enablers need to be easy to use. In OMA, spending time thinking about how users will interact with a particular service is routine.

Figure 2.5 shows the structure of OMA. The Technical Plenary is responsible for the approval and maintenance of the OMA specifications. It consists of a number of Technical Working Groups and, at the time of writing, two Committees.

The Operations and Processes Committee defines and supports the operational processes of the Technical Plenary. The Release Planning and Management Committee plans and manages the OMA releases, which are based on the specifications developed by the Technical Working Groups.

2.6.1 OMA Releases and Specifications

OMA produces Release Packages. Each of these packages consists of a set of OMA specifications, which are the documents produced by the OMA Technical Working Groups.

For example, the Enabler Release Package for PoC Version 1.0 [232] includes an Enabler Release Definition document [229] that provides a high-level definition of the

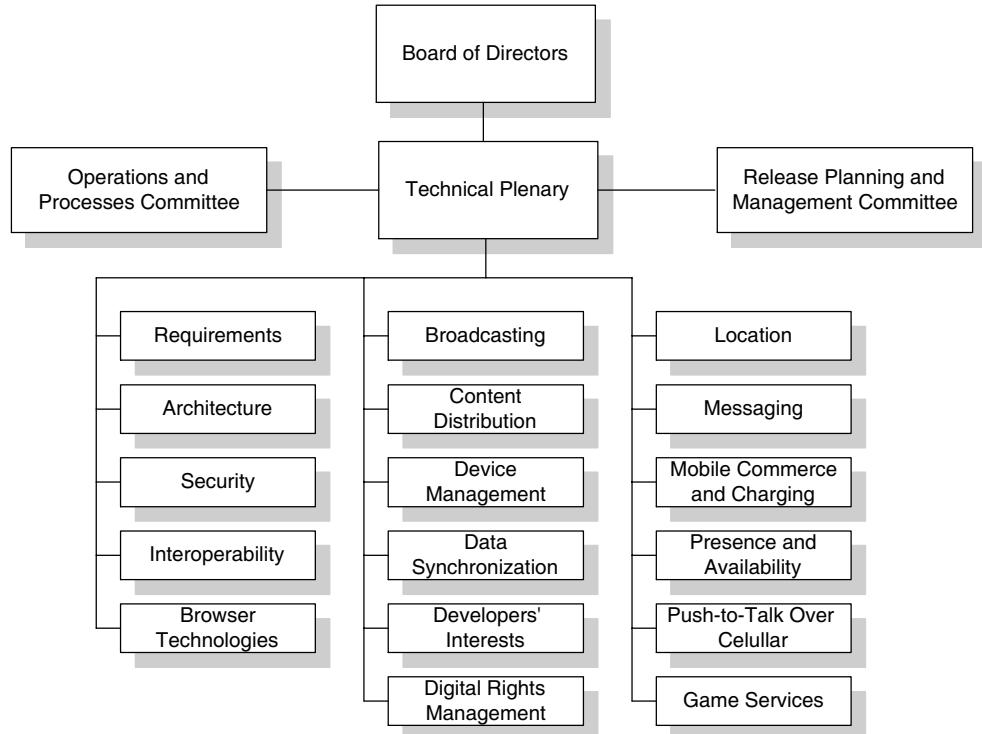


Figure 2.5: OMA structure

PoC (Push-to-talk over Cellular) service and lists the specifications contained in the Enabler Release Package. In addition, the Enabler Release Package includes the following specifications:

- architecture [233]
- requirements [235]
- control plane specification [234]
- user plane specification [236]
- XDM (XML Document Management) specification [237] (which defines data formats and XCAP (XML Configuration Access Protocol) application usages for PoC).

OMA defines maturity levels for its releases. The maturity levels are called *phases* in OMA terminology. Each OMA Release Package can be in one of the following phases:

- Phase 1: Candidate Enabler Release. This is the initial state of the release.
- Phase 2: Approved Enabler Release. The release has successfully passed interoperability tests.

- Phase 3: OMA Interoperability Release. The release has successfully passed exhaustive interoperability tests that may involve interoperability with other OMA service enablers.

As the definitions of the various release phases clearly state, interoperability tests play a key role in OMA. The OMA interoperability tests are referred to as Test Fests and are organized by the Interoperability (IOP) Technical Working Group, which specifies the processes and programs for the Test Fests.

2.6.2 Relationship between OMA and 3GPP/3GPP2

A number of OMA Technical Working Groups use the IMS to some degree. As a consequence, we need to look at the relationship between OMA and some of its Technical Working Groups with 3GPP and 3GPP2 with respect to the IMS.

Some of the OMA work uses the IMS as a base. Therefore there are situations where an OMA Technical Working Group comes up with new requirements on the IMS that need to be met in order to implement a new service.

In general, the agreement between 3GPP, 3GPP2 and OMA is that OMA generates requirements on the IMS, and 3GPP and 3GPP2 extend the IMS to meet these new requirements. This agreement tries to avoid having different versions of the IMS: the 3GPP IMS and the IMS as extended by OMA. Having a single organization managing and maintaining the specifications of the IMS ensures interoperability between the IMS implementations of different vendors.

Still, there is no clear-cut distinction between the IMS and the services on top of it. A multimedia session between two participants may be considered a service by some, but it is part of the IMS, as discussed in Chapter 5. Conferencing can also be considered to be a service, but it is specified by 3GPP as part of the IMS. Presence is an interesting area as well, because both 3GPP and OMA have ongoing work related to presence.

However, even when both 3GPP and OMA work on similar issues, such as presence, they aim to have compatible specifications. For example, the OMA specifications developed by the Presence and Availability Technical Working Group focus on different aspects of presence from the 3GPP presence-related specifications. Nevertheless, all these specifications are compatible.

Another example of an area where both OMA and 3GPP perform activities is messaging. While 3GPP focuses on specifying instant messaging services for the IMS using the work of the IETF SIMPLE WG as a base (e.g., the SIP MESSAGE method and MSRP), OMA focuses on the interworking between SIMPLE-based and Wireless Village-based instant messaging and on the evolution of MMS (Multimedia Messaging Service).

In order to ensure that every OMA service uses the IMS (as specified by 3GPP and 3GPP2) in a consistent and interoperable way, OMA has produced the IMSinOMA Enabler Release Package [230]. This release package includes an Enabler Release Definition document [228], a Requirements document [239] and an Architecture document [238]. This release package also describes how non-IMS-based OMA services can interoperate with IMS-based OMA services.

2.6.3 Relationship between OMA and the IETF

In the same way as the 3GPP and 3GPP2 IMS specifications refer to IETF protocols and extensions, OMA specifications also include references to IETF documents. The

standardization collaboration between OMA and the IETF (documented in RFC 3975 [169]) consists mainly of working-group-level communications. A set of engineers collaborate with both OMA and the IETF. They bring OMA requirements to the relevant IETF working groups, which analyze them and develop appropriate solutions.

However, sometimes communications at the working group level are insufficient. To handle these cases, both OMA and the IETF have appointed a liaison to each other. At the time of writing, the OMA liaison to the IETF is Ileana Leuca and the IETF liaison to OMA is Dean Willis.

OMA maintains a web page at the following address that allows both organizations to track the status of the IETF Internet-Drafts that the OMA Technical Working Groups need:

<http://www.openmobilealliance.org/Technical/IETF.aspx>

Chapter 3

General Principles of the IMS Architecture

In Chapter 1 we introduced the circuit-switched and the packet-switched domains and described why we need the IMS to provide rich Internet services. Chapter 2 introduced the players standardizing the IMS and defining its architecture. In this chapter we will describe the history of the circuit-switched and the packet-switched domains. In addition, we will introduce the design principles that lay behind the IMS architecture and its protocols. We will also tackle in this chapter the IMS network nodes and the different ways in which users are identified in the IMS.

3.1 From Circuit-switched to Packet-switched

Let us look at how cellular networks have evolved from circuit-switched networks to packet-switched networks and how the IMS is the next step in this evolution. We will start with a brief introduction to the history of the 3G circuit-switched and packet-switched domains.

The Third Generation Partnership Project (3GPP) is chartered to develop specifications for the evolution of GSM. That is, 3GPP uses the GSM specifications as a design base for a third generation mobile system.

GSM has two different modes of operation: circuit-switched and packet-switched. The 3G circuit-switched and packet-switched domains are based on these GSM modes of operation.

3.1.1 *GSM Circuit-switched*

Not surprisingly, the GSM circuit-switched network uses circuit-switched technologies, which are also used in the PSTN (Public Switched Telephone Network). Circuit-switched networks have two different planes: the signaling plane and the media plane.

The signaling plane includes the protocols used to establish a circuit-switched path between terminals. In addition, service invocation also occurs in the signaling plane.

The media plane includes the data transmitted over the circuit-switched path between the terminals. The encoded voice exchanged between users belongs to the media plane.

Signaling and media planes followed the same path in early circuit-switched networks. Nevertheless, at a certain point in time the PSTN started to differentiate the paths the signaling

plane and the media plane follow. This differentiation was triggered by the introduction of services based on IN (Intelligent Network). Calls to toll-free numbers are an example of an IN service. The GSM version of IN services is known as CAMEL services (Customized Applications for Mobile network Enhanced Logic).

In both IN and CAMEL the signaling plane follows the media plane until there is a point where the call is temporarily suspended. At that point the signaling plane performs a database query (e.g., a query for a routing number for an 800 number) and receives a response. When the signaling plane receives the response to the query the call setup is resumed and both the signaling plane and the media plane follow the same path until they reach the destination.

3GPP has gone a step further in the separation of signaling and media planes with the introduction of the split architecture for the MSC (Mobile Switching Center). The MSC is split into an MSC server and a media gateway. The MSC server handles the signaling plane and the media gateway handles the media plane. The split architecture was introduced in Release 4 of the 3GPP specifications.

We will see that the IMS also keeps signaling and media paths separate, but goes even further in this separation. The only nodes that need to handle both signaling and media are the IMS terminals; no network node needs to handle both.

3.1.2 *GSM Packet-switched*

The GSM packet-switched network, also known as GPRS (General Packet Radio Service, specified in 3GPP TS 23.060 [35]) was the base for the 3GPP Release 4 packet-switched domain. This domain allows users to connect to the Internet using native packet-switched technologies.

Initially, there were three applications designed to boost the usage of the packet-switched domain:

- the Wireless Application Protocol (WAP) [314];
- access to corporate networks;
- access to the public Internet.

Nevertheless, none of these applications was attracting enough customers to justify the enormous cost of deploying packet-switched mobile networks.

3.2 IMS Requirements

The situation that operators were facing right before the conception of the IMS was not encouraging at all. The circuit-switched voice market had become a commodity, and operators found it difficult to make a profit by only providing and charging for voice calls. On the other hand, packet-switched services had not taken off yet, so operators were not making much money from them either.

Thus, operators needed a way to provide more attractive packet-switched services to attract users to the packet-switched domain. That is, the mobile Internet needed to become more attractive to its users. In this way the IMS (IP Multimedia Subsystem) was born. With the vision described in Chapter 1 in mind, equipment vendors and operators started designing the IMS.

So, the IMS aims to:

- combine the latest trends in technology;
- make the *mobile Internet* paradigm come true;
- create a common platform to develop diverse multimedia services;
- create a mechanism to boost margins due to extra usage of mobile packet-switched networks.

Let us look at the requirements that led to the design of the 3GPP IMS (captured in 3GPP TS 22.228 [53] Release 5). In these requirements the IMS is defined as an architectural framework created for the purpose of delivering IP multimedia services to end users. This framework needs to meet the following requirements:

- support for establishing IP Multimedia Sessions;
- support for a mechanism to negotiate Quality of Service (QoS);
- support for interworking with the Internet and circuit-switched networks;
- support for roaming;
- support for strong control imposed by the operator with respect to the services delivered to the end user;
- support for rapid service creation without requiring standardization.

The Release 6 version of 3GPP TS 22.228 [53] added a new requirement to support access from networks other than GPRS. This is the so-called *access independence* of the IMS, since the IMS provides support for different access networks.

3.2.1 IP Multimedia Sessions

The IMS can deliver a broad range of services. However, there is one service of special importance to users: audio and video communications. This requirement stresses the need to support the main service to be delivered by the IMS: multimedia sessions over packet-switched networks. Multimedia refers to the simultaneous existence of several media types. The media types in this case are audio and video.

Multimedia communications were already standardized in previous 3GPP releases, but those multimedia communications take place over the circuit-switched network rather than the packet-switched network.

3.2.2 QoS

Continuing with the analysis of the requirements we find the requirement to negotiate a certain QoS (Quality of Service). This is a key component of the IMS.

The QoS for a particular session is determined by a number of factors, such as the maximum bandwidth that can be allocated to the user based on the user's subscription or the current state of the network. The IMS allows operators to control the QoS a user gets, so that operators can differentiate certain groups of customers from others.

3.2.3 *Interworking*

Support for interworking with the Internet is an obvious requirement, given that the Internet offers millions of potential destinations for multimedia sessions initiated in the IMS. By the requirement to interwork with the Internet, the number of potential sources and destinations for multimedia sessions is dramatically expanded.

The IMS is also required to interwork with circuit-switched networks, such as the PSTN (Public Switched Telephone Network), or existing cellular networks. The first audio/video IMS terminals that will reach the market will be able to connect to both circuit-switched and packet-switched networks. So, when a user wants to call a phone in the PSTN or a cellular phone the IMS terminal chooses to use the circuit-switched domain.

Thus, interworking with circuit-switched networks is not strictly required, although, effectively, most of the IMS terminals will also support the circuit-switched domain.¹ The requirement to support interworking with circuit-switched networks can be considered a long-term requirement. This requirement will be implemented when it is possible to build IMS terminals with packet-switched support only.

3.2.4 *Roaming*

Roaming support has been a general requirement since the second generation of cellular networks; users have to be able to roam to different networks (e.g., if a user is visiting a foreign country). Obviously the IMS inherits this requirement, so it should be possible for users to roam to different countries (subject to the existence of a roaming agreement signed between the home and the visited network).

3.2.5 *Service Control*

Operators typically want to impose policies on the services delivered to the user. We can divide these policies into two categories:

- general policies applicable to all the users in the network;
- individual policies that apply to a particular user.

The first type of policy comprises a set of restrictions that apply to all users in the network. For instance, operators may want to restrict the usage of high-bandwidth audio codecs, such as G.711 (ITU-T Recommendation G.711 [177]), in their networks. Instead, they may want to promote lower bandwidth codecs such as AMR (Adaptive Multi Rate, specified in 3GPP TS 26.071 [7]).

The second type of policy includes a set of policies which are tailored to each user. For instance, a user may have some subscription to use IMS services that do not include the use of video. The IMS terminal will most likely support video capabilities, but if the user attempts to initiate a multimedia session that includes video, the operator will prevent that session being set up. This policy is modeled on a user-by-user basis, as it is dependent on the terms of usage in the user's subscription.

¹IMS terminals supporting audio capabilities are required to support the circuit-switched domain because of the inability of IMS Releases 5 and 6 to provide support for emergency calls. So, in IMS Releases 5 and 6, emergency calls are placed over the circuit-switched domain. 3GPP Release 7 provides support for emergency calls over IMS. Emergency calls are further analyzed in Chapters 13 and 14.

3.2.6 Rapid Service Creation

The requirement about service creation had a strong impact on the design of IMS architecture. This requirement states that IMS services do not need to be standardized.

This requirement represents a milestone in cellular design, because in the past, every single service was either standardized or had a proprietary implementation. Even when services were standardized there was no guarantee that the service would work when roaming to another network. The reader may already have experienced the lack of support for call diversion to voicemail in GSM networks when the user is visiting another country.

The IMS aims to reduce the time it takes to introduce a new service. In the past the standardization of the service and interoperability tests caused a significant delay. The IMS reduces this delay by standardizing *service capabilities* instead of *services*.

3.2.7 Multiple Access

The multiple access requirement introduces other means of access than GPRS. The IMS is just an IP network and, like any other IP network, it is lower-layer and access-independent. Any access network can in principle provide access to the IMS. For instance, the IMS can be accessed using a WLAN (Wireless Local Access Network), an ADSL (Asymmetric Digital Subscriber Line), an HFC (Hybrid Fiber Coax), or a cable modem.

Still, 3GPP, as a project committed to developing solutions for the evolution of GSM, has focused on GPRS access (both in GSM and UMTS (Universal Mobile Telecommunications System)) for the first release of the IMS (i.e., Release 5). Future releases will study other accesses, such as WLAN.

3.3 Overview of Protocols used in the IMS

When the European Telecommunications Standards Institute (ETSI) developed the GSM standard, most of its protocols were specially designed for GSM (especially those dealing with the radio interface and with mobility management). ETSI reused only a few protocols developed by the International Telecommunication Union-Telecommunications (ITU-T). Most of the protocols were developed from scratch because there were no existing protocols to take as a base.

A few years later, 3GPP began developing the IMS, a system based on IP protocols, which had been traditionally developed by the IETF (Internet Engineering Task Force). 3GPP analyzed the work done in the past by ETSI in developing its own protocols and decided to reuse protocols which had already been developed (or were under development at that time) in other standards development organizations (SDOs) such as the IETF or ITU-T. This way, 3GPP takes advantage of the experience of the IETF and the ITU-T in designing robust protocols, reducing at the same time standardization and development costs.

3.3.1 Session Control Protocol

The protocols that control the calls play a key role in any telephony system. In circuit-switched networks the most common call control protocols are TUP (Telephony User Part, ITU-T Recommendation Q.721 [176]), ISUP (ISDN User Part, ITU-T Recommendation Q.761 [185]), and the more modern BICC (Bearer Independent Call Control, ITU-T Recommendation Q.1901 [186]). The protocols considered for use as the session control protocol for the IMS were obviously all based on IP. The candidates were as follows.

Bearer Independent Call Control (BICC). BICC (specified in ITU-T Recommendation Q.1901 [186]) is an evolution of ISUP. Unlike ISUP, BICC separates the signaling plane from the media plane, so that signaling can traverse a separate set of nodes from the media plane. In addition, BICC supports and can run over a different set of technologies, such as IP, SS7 (Signaling System No. 7, ITU-T Recommendation Q.700 [180]), or ATM (Asynchronous Transfer Mode).

H.323. Like BICC, H.323 (ITU-T Recommendation H.323 [191]) is an ITU-T protocol. H.323 defines a new protocol to establish multimedia sessions. Unlike BICC, H.323 was designed from scratch to support IP technologies. In H.323, signaling and the media do not need to traverse the same set of hosts.

SIP (Session Initiation Protocol, RFC 3261 [286]). Specified by the IETF as a protocol to establish and manage multimedia sessions over IP networks, SIP was gaining momentum at the time when 3GPP was choosing its session control protocol. SIP follows the well-known client–server model, much used by many protocols developed by the IETF. SIP designers borrowed design principles from SMTP (Simple Mail Transfer Protocol, RFC 2821 [201]) and especially from HTTP (Hypertext Transfer Protocol, RFC 2616 [144]). SIP inherits most of its characteristics from these two protocols. This is an important strength of SIP, because HTTP and SMTP are the most successful protocols on the Internet. SIP, unlike BICC and H.323, does not differentiate the User-to-Network Interface (UNI) from a Network-to-Network Interface (NNI). In SIP there is just a single protocol that works end-to-end. Unlike BICC and H.323, SIP is a text-based protocol. This means that it is easier to extend, debug, and use to build services.

SIP was chosen as the session control protocol for the IMS. The fact that SIP makes it easy to create new services carried great weight in this decision. Since SIP is based on HTTP, SIP service developers can use all the service frameworks developed for HTTP, such as CGI (Common Gateway Interface) and Java servlets.

3.3.2 *The AAA Protocol*

In addition to the session control protocol there are a number of other protocols that play important roles in the IMS. Diameter (whose base protocol is specified in RFC 3588 [96]) was chosen to be the AAA (Authentication, Authorization, and Accounting) protocol in the IMS.

Diameter is an evolution of RADIUS (specified in RFC 2865 [262]), which is a protocol that is widely used on the Internet to perform AAA. For instance, when a user dials up to an Internet Service Provider (ISP) the network access server uses RADIUS to authenticate and authorize the user accessing the network.

Diameter consists of a base protocol that is complemented with so-called *Diameter applications*. Diameter applications are customizations or extensions to Diameter to suit a particular application in a given environment.

The IMS uses Diameter in a number of interfaces, although not all the interfaces use the same Diameter application. For instance, the IMS defines a Diameter application to interact with SIP during session setup and another one to perform credit control accounting.

3.3.3 Other Protocols

In addition to SIP and Diameter there are other protocols that are used in the IMS. H.248 (ITU-T Recommendation H.248 [189]) and its packages are used by signaling nodes to control nodes in the media plane (e.g., a media gateway controller controlling a media gateway). H.248 was jointly developed by ITU-T and IETF and is also referred to as the MEGACO (MEdia GAteway COntrol) protocol.

RTP (Real-Time Transport Protocol, defined in RFC 3550 [301]) and RTCP (RTP Control Protocol, defined in RFC 3550 [301] as well) are used to transport real-time media, such as video and audio.

We have mentioned a few application-layer protocols used in the IMS. We will describe these in Parts II and III of this book, along with other application-layer Internet protocols that may be used in the IMS in the future, and other protocols that belong to other layers.

3.4 Overview of IMS Architecture

Before exploring the general architecture in the IMS we should keep in mind that 3GPP does not standardize *nodes*, but *functions*. This means that the IMS architecture is a collection of functions linked by standardized interfaces. Implementers are free to combine two functions into a single node (e.g., into a single physical box). Similarly, implementers can split a single function into two or more nodes.

In general, most vendors follow the IMS architecture closely and implement each function into a single node. Still, it is possible to find nodes implementing more than one function and functions distributed over more than one node.

Figure 3.1 provides an overview of the IMS architecture as standardized by 3GPP. The figure shows most of the signaling interfaces in the IMS, typically referred to by a two- or three-letter code. We do not include all the interfaces defined in the IMS, but only the most relevant ones. The reader can refer to 3GPP TS 23.002 [17] to find a complete list of all the interfaces.

On the left side of Figure 3.1 we can see the IMS mobile terminal, typically referred to as the User Equipment (UE). The IMS terminal attaches to a packet network, such as the GPRS network, through a radio link.

Note that, although the figure shows an IMS terminal attaching to the network using a radio link, the IMS supports other types of device and access. PDAs (personal digital assistants) and computers are examples of devices that can connect to the IMS. Examples of alternative accesses are WLAN or ADSL.

The remainder of Figure 3.1 shows the nodes included in the so-called IP Multimedia Core Network Subsystem. These nodes are:

- one or more user databases, called HSSs (Home Subscriber Servers) and SLFs (Subscriber Location Functions);
- one or more SIP servers, collectively known as CSCFs (Call/Session Control Functions);
- one or more ASes (Application Servers);
- one or more MRFs (Media Resource Functions), each one further divided into MRFCs (Media Resource Function Controllers) and MRFPs (Media Resource Function Processors);

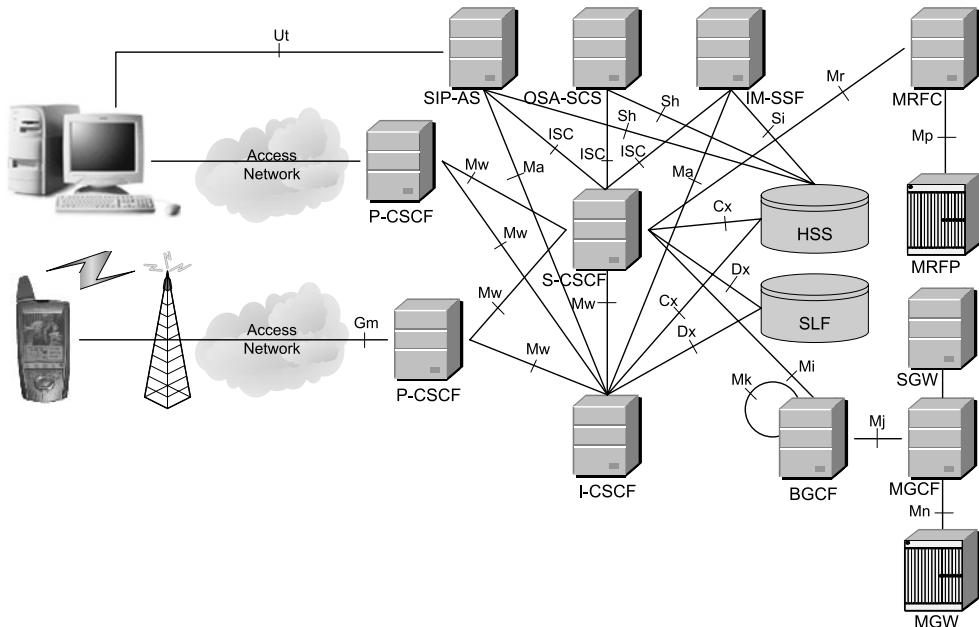


Figure 3.1: 3GPP IMS architecture overview

- one or more BGCFs (Breakout Gateway Control Functions);
- one or more PSTN gateways, each one decomposed into an SGW (Signaling Gateway), an MGCF (Media Gateway Controller Function), and an MGW (Media Gateway).

Note that Figure 3.1 does not contain a reference to charging collector functions. These are described in Section 7.4.

3.4.1 The Databases: the HSS and the SLF

The Home Subscriber Server (HSS) is the central repository for user-related information. Technically, the HSS is an evolution of the HLR (Home Location Register), which is a GSM node. The HSS contains all the user-related subscription data required to handle multimedia sessions. These data include, among other items, location information, security information (including both authentication and authorization information), user profile information (including the services that the user is subscribed to), and the S-CSCF (Serving-CSCF) allocated to the user.

A network may contain more than one HSS, in case the number of subscribers is too high to be handled by a single HSS. However, all the data related to a particular user are stored in a single HSS.²

Networks with a single HSS do not need a Subscription Locator Function (SLF). On the other hand, networks with more than one HSS do require an SLF.

²The HSS is typically implemented using a redundant configuration, to avoid a single point of failure. Nevertheless, we consider a redundant configuration of HSSs as being a single logical node.

The SLF is a simple database that maps users' addresses to HSSs. A node that queries the SLF, with a user's address as the input, obtains the HSS that contains all the information related to that user as the output.

Both the HSS and the SLF implement the Diameter protocol (RFC 3588 [96]) with an IMS-specific Diameter application.

3.4.2 *The CSCF*

The CSCF (Call/Session Control Function), which is a SIP server, is an essential node in the IMS. The CSCF processes SIP signaling in the IMS. There are three types of CSCF, depending on the functionality they provide. All of them are collectively known as CSCFs, but an individual CSCF belongs to one of the following three categories:

- P-CSCF (Proxy-CSCF);
- I-CSCF (Interrogating-CSCF);
- S-CSCF (Serving-CSCF).

3.4.2.1 **The P-CSCF**

The P-CSCF is the first point of contact (in the signaling plane) between the IMS terminal and the IMS network. From the SIP point of view the P-CSCF is acting as an outbound/inbound SIP proxy server. This means that all the requests initiated by the IMS terminal or destined for the IMS terminal traverse the P-CSCF. The P-CSCF forwards SIP requests and responses in the appropriate direction (i.e., toward the IMS terminal or toward the IMS network).

The P-CSCF is allocated to the IMS terminal during IMS registration and does not change for the duration of the registration (i.e., the IMS terminal communicates with a single P-CSCF during the registration).

The P-CSCF includes several functions, some of which are related to security. First, it establishes a number of IPsec security associations toward the IMS terminal. These IPsec security associations offer integrity protection (i.e., the ability to detect whether the contents of the message have changed since its creation).

Once the P-CSCF authenticates the user (as part of security association establishment) the P-CSCF asserts the identity of the user to the rest of the nodes in the network. This way, other nodes do not need to further authenticate the user, because they trust the P-CSCF. The rest of the nodes in the network use this identity (asserted by the P-CSCF) for a number of purposes, such as providing personalized services and generating account records.

In addition, the P-CSCF verifies the correctness of SIP requests sent by the IMS terminal. This verification keeps IMS terminals from creating SIP requests that are not built according to SIP rules.

The P-CSCF also includes a compressor and a decompressor of SIP messages (IMS terminals include both as well). SIP messages can be large, given that SIP is a text-based protocol. While a SIP message can be transmitted over a broadband connection in a fairly short time, transmitting large SIP messages over a narrowband channel, such as some radio links, may take a few seconds. The mechanism used to reduce the time to transmit a SIP

message is to compress the message, send it over the air interface, and decompress it at the other end.³

The P-CSCF may include a PDF (Policy Decision Function). The PDF may be integrated with the P-CSCF or be implemented as a stand-alone unit. The PDF authorizes media plane resources and manages Quality of Service over the media plane.

The P-CSCF also generates charging information toward a charging collection node.

An IMS network usually includes a number of P-CSCFs for the sake of scalability and redundancy. Each P-CSCF serves a number of IMS terminals, depending on the capacity of the node.

3.4.2.2 P-CSCF Location

The P-CSCF may be located either in the visited network or in the home network. When the underlying packet network is based on GPRS, the P-CSCF is always located in the network where the GGSN (Gateway GPRS Support Node) is located. So both the P-CSCF and GGSN are either located in the visited network or in the home network. Owing to current deployments of GPRS, it is expected that the first IMS networks will inherit this mode and will be configured with the GGSN and P-CSCF in the home network. It is also expected that once IMS reaches the mass market, operators will migrate the configuration and will locate the P-CSCF and the GGSN in the visited network.

3.4.2.3 The I-CSCF

The I-CSCF is a SIP proxy located at the edge of an administrative domain. The address of the I-CSCF is listed in the DNS (Domain Name System) records of the domain. When a SIP server follows SIP procedures (described in RFC 3263 [285]) to find the next SIP hop for a particular message, the SIP server obtains the address of an I-CSCF of the destination domain.

Besides the SIP proxy server functionality, the I-CSCF has an interface to the SLF and the HSS. This interface is based on the Diameter protocol (RFC 3588 [96]). The I-CSCF retrieves user location information and routes the SIP request to the appropriate destination (typically an S-CSCF).

The I-CSCF also implements an interface to Application Servers, in order to route requests that are addressed to services rather than regular users.

In addition, the I-CSCF may optionally encrypt the parts of the SIP messages that contain sensitive information about the domain, such as the number of servers in the domain, their DNS names, or their capacity. This functionality is referred to as THIG (Topology Hiding Inter-network Gateway). THIG functionality is optional and is not likely to be deployed by most networks.

A network will include typically a number of I-CSCFs for the sake of scalability and redundancy.

³There is a misconception that compression between the IMS terminal and the P-CSCF is enabled just to save a few bytes over the air interface. This is not the motivation lying behind compression. In particular, it is not worth saving a few bytes of signaling when the IMS terminal will be establishing a multimedia session (e.g., audio, video) that will use much more bandwidth than the signaling. The main motivation for compression is to reduce the time taken to transmit SIP messages over the air interface.

3.4.2.4 I-CSCF Location

The I-CSCF is usually located in the home network, although in some special cases, such as an I-CSCF(THIG), it may be located in a visited network as well.

3.4.2.5 The S-CSCF

The S-CSCF is the central node of the signaling plane. The S-CSCF is essentially a SIP server, but it performs session control as well. In addition to SIP server functionality the S-CSCF also acts as a SIP registrar. This means that it maintains a binding between the user location (e.g., the IP address of the terminal the user is logged onto) and the user's SIP address of record (also known as a Public User Identity).

Like the I-CSCF, the S-CSCF also implements a Diameter (RFC 3588 [96]) interface to the HSS. The main reasons to interface the HSS are as follows.

- To download the authentication vectors of the user who is trying to access the IMS from the HSS. The S-CSCF uses these vectors to authenticate the user.
- To download the user profile from the HSS. The user profile includes the service profile, which is a set of triggers that may cause a SIP message to be routed through one or more ASes.
- To inform the HSS that this is the S-CSCF allocated to the user for the duration of the registration.

All the SIP signaling the IMS terminals sends, and all the SIP signaling the IMS terminal receives, traverses the allocated S-CSCF. The S-CSCF inspects every SIP message and determines whether the SIP signaling should visit one or more ASes en route toward the final destination. Those ASes would potentially provide a service to the user.

One of the main functions of the S-CSCF is to provide SIP routing services. If the user dials a telephone number instead of a SIP URI (Uniform Resource Identifier), the S-CSCF provides translation services, typically based on DNS E.164 Number Translation (as described in RFC 2916 [143]).

The S-CSCF also enforces the policy of the network operator. For example, a user may not be authorized to establish certain types of session. The S-CSCF keeps users from performing unauthorized operations.

A network usually includes a number of S-CSCFs for the sake of scalability and redundancy. Each S-CSCF serves a number of IMS terminals, depending on the capacity of the node.

3.4.2.6 S-CSCF Location

The S-CSCF is always located in the home network.

3.4.3 *The Application Server*

The Application Server (AS) is a SIP entity that hosts and executes services. Depending on the actual service the AS can operate in SIP proxy mode, SIP UA (User Agent) mode (i.e., endpoint), or SIP B2BUA (Back-to-Back User Agent) mode (i.e., a concatenation of two SIP User Agents). The AS interfaces the S-CSCF and the I-CSCF using SIP and the

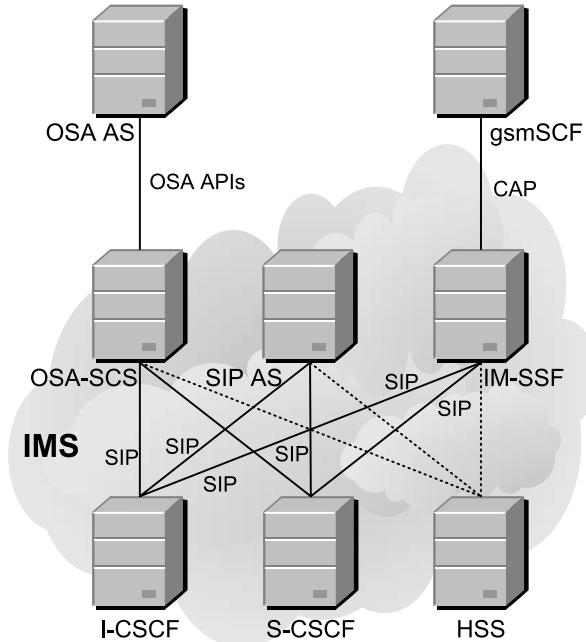


Figure 3.2: Three types of Application Server

HSS using Diameter. In addition, ASes can provide IMS terminals with an interface that is used for configuration purposes.

Figure 3.2 depicts three different types of Application Server.

SIP AS (Application Server). This is the native AS that hosts and executes IP Multimedia Services based on SIP. It is expected that new IMS-specific services will probably be developed in SIP ASes.

OSA-SCS (Open Service Access–Service Capability Server). This AS provides an interface to the OSA framework AS. It inherits all the OSA capabilities, especially the capability to access the IMS securely from external networks. This node acts as an AS on one side (interfacing the S-CSCF with SIP) and as an interface between the OSA AS and the OSA Application Programming Interface (API, described in 3GPP TS 29.198 [18]).

IM-SSF (IP Multimedia Service Switching Function). This specialized AS allows us to reuse CAMEL (Customized Applications for Mobile network Enhanced Logic) services that were developed for GSM in the IMS. The IM-SSF allows a gsmSCF (GSM Service Control Function) to control an IMS session. The IM-SSF acts as an AS on one side (interfacing the S-CSCF with SIP). On the other side, it acts as an SSF (Service Switching Function), interfacing the gsmSCF with a protocol based on CAP (CAMEL Application Part, defined in 3GPP TS 29.278 [1]).

All three types of AS behave as SIP ASes toward the IMS network (i.e., they act as a SIP proxy server, a SIP User Agent, a SIP redirect server, or a SIP Back-to-Back User Agent).

The IM-SSF AS and the OSA-SCS AS have other roles when interfacing CAMEL or OSA, respectively.

In addition to the SIP interface the AS may optionally provide an interface to the HSS. The SIP-AS and OSA-SCS interfaces toward the HSS are based on the Diameter protocol (RFC 3588 [96]) and are used to download or upload data related to a user stored in the HSS. The IM-SSF interface toward the HSS is based on MAP (Mobile Application Part, defined in 3GPP TS 29.002 [46]).

3.4.3.1 AS Location

The AS can be located either in the home network or in an external third-party network to which the home operator maintains a service agreement. In any case, if the AS is located outside the home network, it does not interface the HSS.

3.4.4 The MRF

The MRF (Media Resource Function) provides a source of media in the home network. The MRF provides the home network with the ability to play announcements, mix media streams (e.g., in a centralized conference bridge), transcode between different codecs, obtain statistics, and do any sort of media analysis.

The MRF is further divided into a signaling plane node called the MRFC (Media Resource Function Controller) and a media plane node called the MRFP (Media Resource Function Processor). The MRFC acts as a SIP User Agent and contains a SIP interface towards the S-CSCF. The MRFC controls the resources in the MRFP via an H.248 interface.

The MRFP implements all the media-related functions, such as playing and mixing media.

3.4.4.1 MRF Location

The MRF is always located in the home network.

3.4.5 The BGCF

The BGCF is essentially a SIP server that includes routing functionality based on telephone numbers. The BGCF is only used in sessions that are initiated by an IMS terminal and addressed to a user in a circuit-switched network, such as the PSTN or the PLMN. The main functionality of the BGCF is to do one of the following:

- select an appropriate network where interworking with the circuit-switched domain is to occur;
- select an appropriate PSTN/CS gateway if interworking is to occur in the network where the BGCF is located.

3.4.6 The IMS-ALG and the TrGW

As we describe later in Section 5.2, IMS supports two IP versions, namely IP version 4 (IPv4, specified in RFC 791 [256]) and IP version 6 (IPv6, specified in RFC 2460 [119]). At some point in an IP multimedia session or communication, interworking between the

two versions may occur. In order to facilitate interworking between IPv4 and IPv6 without requiring terminal support, the IMS adds two new functional entities that provide translation between both protocols. These new entities are the IMS Application Layer Gateway (IMS-ALG) and the Transition Gateway (TrGW). The former processes control plane signaling (e.g., SIP and SDP messages); the latter processes media plane traffic (e.g., RTP, RTCP). (The IMS-ALG functionality actually resides in an IBCF; see Section 3.7.2.)

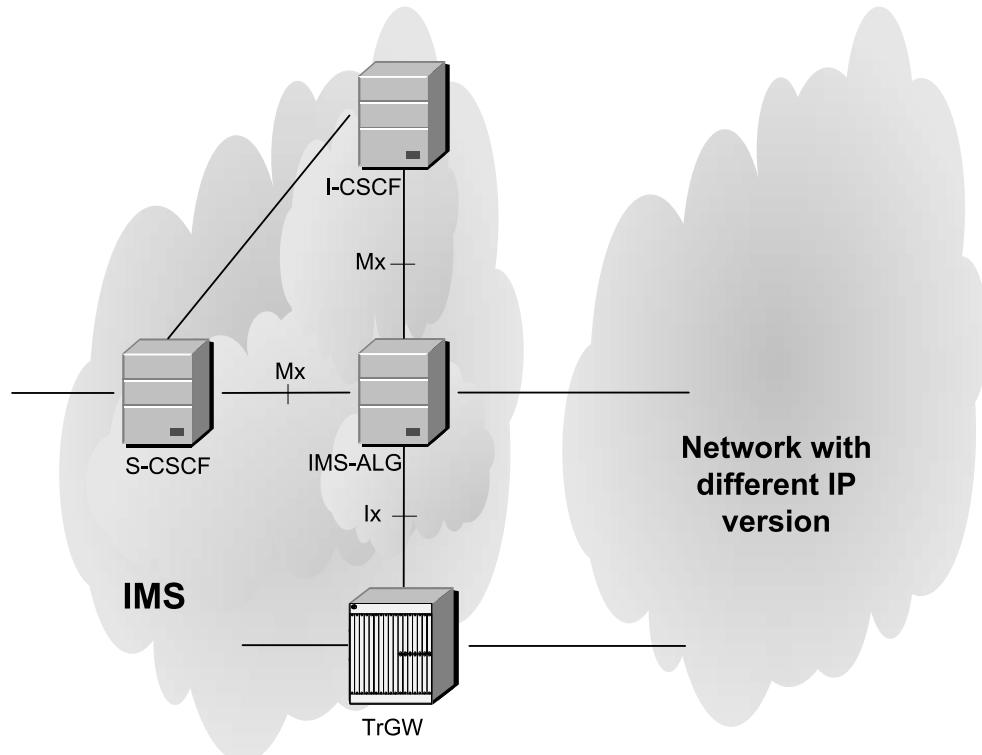


Figure 3.3: The IMS-ALG and the TrGW

Figure 3.3 shows the relation of the IMS-ALG with the TrGW and the rest of the IMS nodes. The IMS-ALG acts as a SIP B2BUA by maintaining two independent signaling legs: one toward the internal IMS network and the other toward the other network. Each of these legs are running over a different IP version. In addition, the IMS-ALG rewrites the SDP by changing the IP addresses and port numbers created by the terminal with one or more IP addresses and port numbers allocated to the TrGW. This allows the media plane traffic to be routed to the TrGW. The IMS-ALG interfaces the TrGW through the *Ix* interface. The IMS-ALG interfaces the I-CSCF for incoming traffic and the S-CSCF for outgoing traffic through the *Mx* interface.

The TrGW is effectively a NAT-PT/NAPT-PT (Network Address Port Translator–Protocol Translator). The TrGW is configured with a pool of IPv4 addresses that are dynamically allocated for a given session. The TrGW does the translation of IPv4 and IPv6 at the media level (e.g., RTP, RTCP). 3GPP standardizes the details of the IPv4/IPv6 interworking of the IMS-ALG and TrGW in 3GPP TS 29.162 [4].

ICE (see Section 4.20.4) can also be used by dual-stack terminals to decide whether to use IPv4 or IPv6. In Section 5.16, we discuss the interactions between different NAT traversal techniques.

3.4.7 The PSTN/CS Gateway

The PSTN gateway provides an interface toward a circuit-switched network, allowing IMS terminals to make and receive calls to and from the PSTN (or any other circuit-switched network).

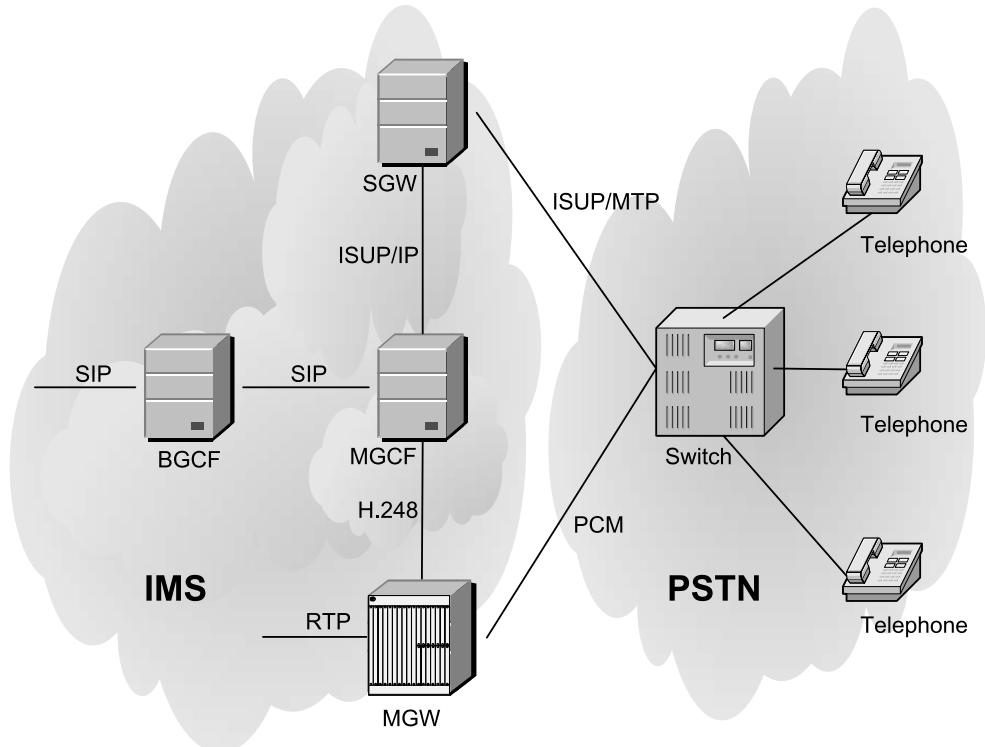


Figure 3.4: The PSTN/CS gateway interfacing a CS network

Figure 3.4 shows a BGCF and a decomposed PSTN gateway that interfaces the PSTN. The PSTN gateway is decomposed into the following functions.

SGW (Signaling Gateway). The Signaling Gateway interfaces the signaling plane of the CS network (e.g., the PSTN). The SGW performs lower-layer protocol conversion. For instance, an SGW is responsible for replacing the lower MTP (ITU-T Recommendation Q.701 [179]) transport with SCTP (Stream Control Transmission Protocol, defined in RFC 2960 [308]) over IP. So, the SGW transforms ISUP (ITU-T Recommendation Q.761 [185]) or BICC (ITU-T Recommendation Q.1901 [186]) over MTP into ISUP or BICC over SCTP/IP.

MGCF (Media Gateway Control Function). The MGCF is the central node of the PSTN/CS gateway. It implements a state machine that does protocol conversion and maps SIP (the call control protocol on the IMS side) to either ISUP over IP or BICC over IP (both BICC and ISUP are call control protocols in circuit-switched networks). In addition to the call control protocol conversion the MGCF controls the resources in an MGW (Media Gateway). The protocol used between the MGCF and the MGW is H.248 (ITU-T Recommendation H.248 [189]).

MGW (Media Gateway). The Media Gateway interfaces the media plane of the PSTN or CS network. On one side the MGW is able to send and receive IMS media over the Real-Time Transport Protocol (RTP) (RFC 3550 [301]). On the other side the MGW uses one or more PCM (Pulse Code Modulation) time slots to connect to the CS network. In addition, the MGW performs transcoding when the IMS terminal does not support the codec used by the CS side. A common scenario occurs when the IMS terminal is using the AMR (3GPP TS 26.071 [7]) codec and the PSTN terminal is using the G.711 codec (ITU-T Recommendation G.711 [177]).

3.4.8 Home and Visited Networks

The IMS borrows a few concepts from GSM and GPRS, such as having a home and a visited network. In the cellular model, when we use our cellphones in the area where we reside, we are using the infrastructure provided by our network operator. This infrastructure forms the so-called home network.

On the other hand, if we roam outside the area of coverage of our home network (e.g., when we visit another country), we use an infrastructure provided not by our operator, but by another operator. This infrastructure is what we call the visited network, because effectively we are a *visitor* in this network.

In order for us to use a visited network, the visited network operator has to have signed a roaming agreement with our home network operator. In these agreements both operators negotiate some aspects of the service provided to the user, such as the price of calls, the quality of service, or how to exchange accounting records.

The IMS reuses the same concept of having a visited and a home network. Most of the IMS nodes are located in the home network, but there is a node that can be located either in the home or the visited network. That node is the P-CSCF (Proxy-CSCF). The IMS allows two different configurations, depending on whether the P-CSCF is located in the home or the visited network.

In addition, when the IP-CAN (IP Connectivity Access Network) is GPRS, the location of the P-CSCF is subordinated to the location of the GGSN. In roaming scenarios, GPRS allows location of the GGSN either in the home or in the visited network (the SGSN is always located in the visited network).

In the IMS, both the GGSN and the P-CSCF share the same network. This allows the P-CSCF to control the GGSN over the so-called *Rx* and *Gx* interfaces. As both the P-CSCF and the GGSN are located in the same network, the *Rx* and *Gx* interfaces are always intra-operator interfaces, which makes their operation simpler.

Figure 3.5 shows a configuration where the P-CSCF (and the GGSN) is located in the visited network. This configuration represents a longer-term vision of the IMS, because it requires IMS support from the visited network (i.e., the GGSN has to be upgraded to be 3GPP Release 5-compliant).

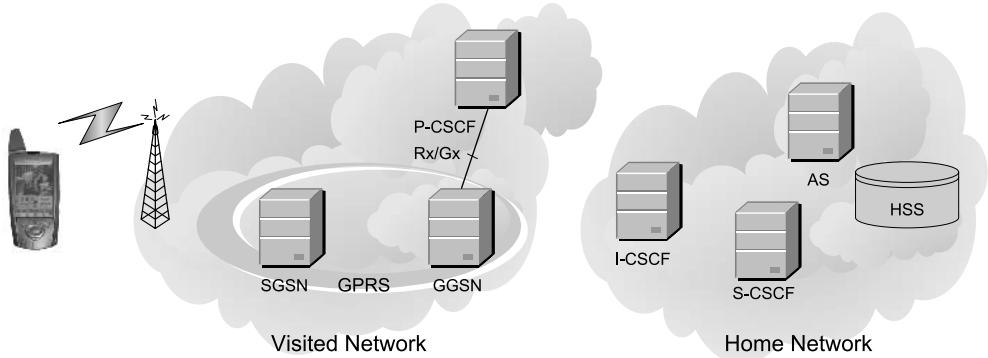


Figure 3.5: The P-CSCF located in the visited network

It is not expected that all networks in the world will deploy IMS simultaneously. Consequently, it is not expected that all roaming partners will upgrade their GGSNs to a Release 5 GGSN at the same time as the home network operator starts to provide the IMS service. Therefore, we expect that early IMS deployments will locate the P-CSCF in the home network, as shown in Figure 3.6.

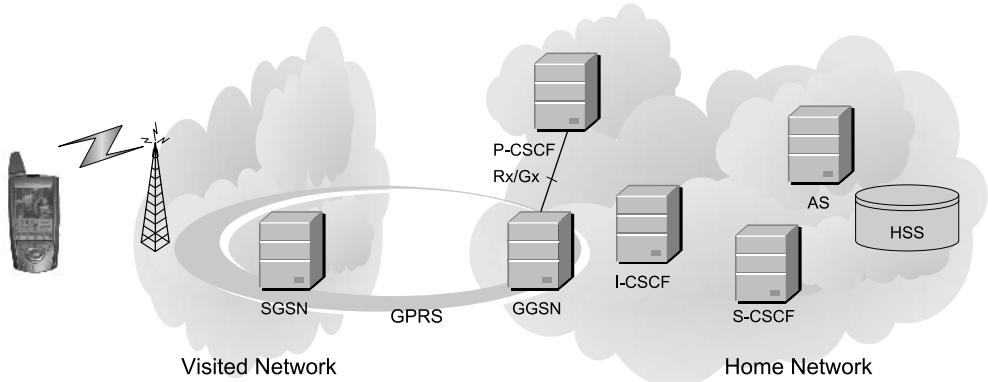


Figure 3.6: The P-CSCF located in the home network

Figure 3.6 shows a near-term configuration where both the P-CSCF and the GGSN are located in the home network. This configuration does not require any IMS support from the visited network. In particular, the visited network does not need to have a 3GPP Release 5-compliant GGSN. The visited network only provides the radio bearers and the SGSN. So, this configuration can be deployed from the very first day of the IMS. As a consequence, it is expected that this will be the most common configuration in the early years of IMS deployments.

Even so, this configuration has a severe disadvantage with respect to the configuration where the P-CSCF and GGSN are located in the visited network. Since the media plane traverses the GGSN and the GGSN is located in the home network, the media are first routed

to the home network and then to their destination. This creates an undesired trombone effect that causes delays in the media plane.

3.5 Identification in the IMS

In a network of any kind, it must be possible to uniquely identify users. This is the property that allows a particular phone to ring (as opposed to a different telephone) when we dial a sequence of digits in the PSTN (Public Switched Telephone Network).

Central to any network is the ability of the operator to identify users, so that calls can be directed to the appropriate user. In the PSTN, users are identified by a telephone number (i.e., a collection of ordered digits that identify the telephone subscriber). The telephone number that identifies a subscriber may be represented in different formats: a local short number, a long-distance number, or an international number. In essence, these are just different representations of the same telephone subscriber. The number of digits depends on the destination of the call (e.g., same area, another region, or another country).

In addition, when a service is provided, sometimes there is a need to identify the service. In the PSTN, services are identified by special numbers, typically through a special prefix, such as 800 numbers. IMS also provides mechanisms to identify services.

3.5.1 Public User Identities

In the IMS there is also a deterministic way to identify users. An IMS user is allocated one or more *Public User Identities*. The home operator is responsible for allocating these Public User Identities to each IMS subscriber. A Public User Identity is either a SIP URI (as defined in RFC 3261 [286]) or a TEL URI (as defined in RFC 3966 [295]). Public User Identities are used as contact information on business cards. In the IMS, Public User Identities are used to route SIP signaling. If we compare the IMS with GSM, a Public User Identity is to the IMS what an MSISDN (Mobile Subscriber ISDN Number) is to GSM.

When the Public User Identity contains a SIP URI, it typically takes the form of `sip:first.last@operator.com`, although IMS operators are able to change this scheme and address their own needs. In addition, it is possible to include a telephone number in a SIP URI using the following format:

```
sip:+1-212-555-0293@operator.com;user=phone
```

This format is needed because SIP requires that the URI under registration be a SIP URI. So, it is not possible to register a TEL URI in SIP, although it is possible to register a SIP URI that contains a telephone number.

The TEL URI is the other format that a Public User Identity can take. The following is a TEL URI representing a phone number in international format:

```
URI}tel:+1-212-555-0293
```

TEL URIs are needed to make a call from an IMS terminal to a PSTN phone, because PSTN numbers are represented only by digits. On the other hand, TEL URIs are also needed if a PSTN subscriber wants to make a call to an IMS user, because a PSTN user can only dial digits.

We envision that operators will allocate at least one SIP URI and one TEL URI per user. There are reasons for allocating more than one Public User Identity to a user, such as having

the ability to differentiate personal (e.g., private) identities that are known to friends and family from business Public User Identities (that are known to colleagues), or for triggering a different set of services.

The IMS offers an interesting concept: *a set of implicitly registered public user identities*. In regular SIP operation, each identity that needs to be registered requires a SIP REGISTER request. In the IMS, it is possible to register several Public User Identities in one message, saving time and bandwidth (the complete mechanism is described in Section 5.6).

3.5.2 Private User Identities

Each IMS subscriber is assigned a *Private User Identity*. Unlike Public User Identities, Private User Identities are not SIP URIs or TEL URIs; instead, they take the format of an NAI (Network Access Identifier, specified in RFC 2486 [72]). The format of an NAI is `username@operator.com`.

Unlike Public User Identities, Private User Identities are not used for routing SIP requests; instead, they are exclusively used for subscription identification and authentication purposes. A Private User Identity performs a similar function in the IMS to that which an IMSI (International Mobile Subscriber Identifier) does in GSM. A Private User Identity need not be known by the user, because it might be stored in a smart card, in the same way that an IMSI is stored in a SIM (Subscriber Identity Module).

3.5.3 The Relation between Public User Identities and Private User Identities

Operators assign one or more Public User Identities and a Private User Identity to each user. In the case of GSM/UMTS (Universal Mobile Telecommunications System), the smart card stores the Private User Identity and at least one Public User Identity. The HSS, as a general database for all the data related to a subscriber, stores the Private User Identity and the collection of Public User Identities allocated to the user. The HSS and the S-CSCF also correlate the Public User Identities and Private User Identities.

The relation between an IMS subscriber, the Private User Identity and the Public User Identities is shown in Figure 3.7. An IMS subscriber is assigned one Private User Identity and a number of Public User Identities. This is the case of the IMS as standardized in 3GPP Release 5.

3GPP Release 6 has extended the relationship of Private User Identities and Public User Identities, as shown in Figure 3.8. An IMS subscriber is allocated not one, but a number of Private User Identities. In the case of UMTS, only one Private User Identity is stored in the smart card, but users may have different smart cards that they insert in different IMS terminals. It might be possible that some of those Public User Identities are used in combination with more than a single Private User Identity. This is the case of Public User Identity 2 in Figure 3.8, because it is assigned to Private User Identities 1 and 2. This allows Public User Identity 2 to be used simultaneously from two IMS terminals, each one assigned a different Private User Identity (e.g., different smart cards are inserted in different terminals).

3.5.4 Public Service Identities

The concept of Public Service Identities (PSIs) is introduced in Release 6 of the 3GPP specifications. Unlike Public User Identities, which are allocated to users, a PSI is an identity

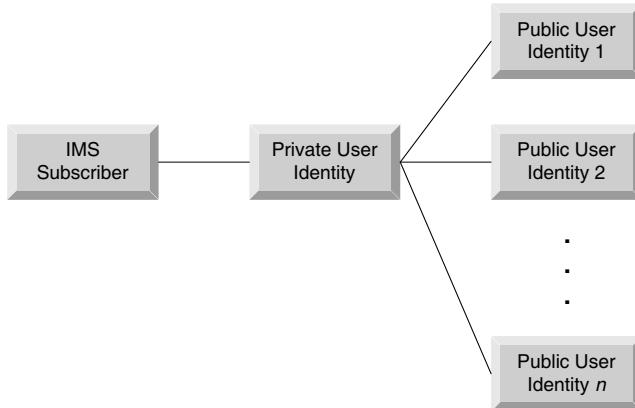


Figure 3.7: Relation of Private User Identity and Public User Identities in 3GPP R5

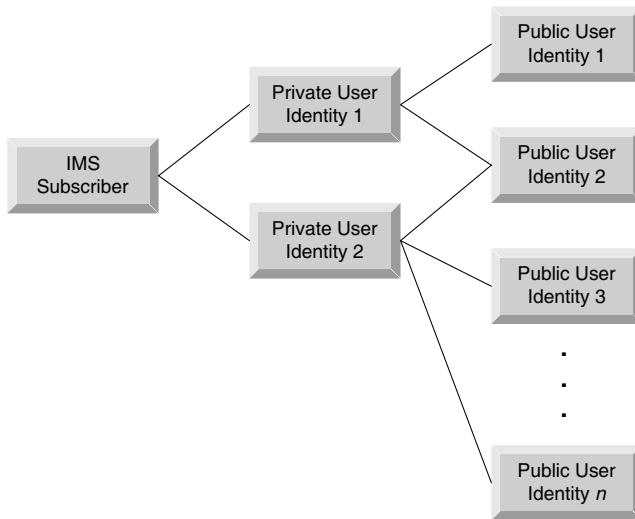


Figure 3.8: Relation of Private User Identities and Public User Identities in 3GPP R6

allocated to a service hosted in an AS. For instance, an AS hosting a chat room is identified by a PSI. Like Public User Identities, PSIs may take the format of a SIP URI or a TEL URI.

Unlike Public User Identities, PSIs do not have an associated Private User Identity. This is because the Private User Identity is used for user authentication purposes. PSIs are not applicable to users.

Since PSIs are service identities that directly resolve to an AS, I-CSCFs directly interface ASes in order to route incoming SIP requests addressed to PSIs towards ASes.

3.6 SIM, USIM, and ISIM in 3GPP

Central to the design of 3GPP terminals is the presence of a UICC (Universal Integrated Circuit Card). The UICC is a removable smart card that contains limited storage of data. The UICC is used to store, among other things, subscription information, authentication keys, a phonebook, and messages.

GSM and 3GPP specifications rely on the presence of a UICC in the terminal for its operation. Without a UICC present in the terminal the user can only make emergency calls (and this is a country-dependent feature; there are countries where a UICC is required to place even an emergency call).

The UICC allows users to easily move their user subscriptions (including the phonebook) from one terminal to another. The user simply removes the smart card from a terminal and inserts it into another terminal.

UICC is a generic term that defines the physical characteristics of the smart card (like the number and disposition of pins, voltage values, etc.). The interface between the UICC and the terminal is standardized.

A UICC may contain several logical applications, such as a SIM (Subscriber Identity Module), a USIM (Universal Subscriber Identity Module), and an ISIM (IP multimedia Services Identity Module). In addition, a UICC can contain other applications, such as a telephone book. Figure 3.9 shows a UICC that contains several applications.

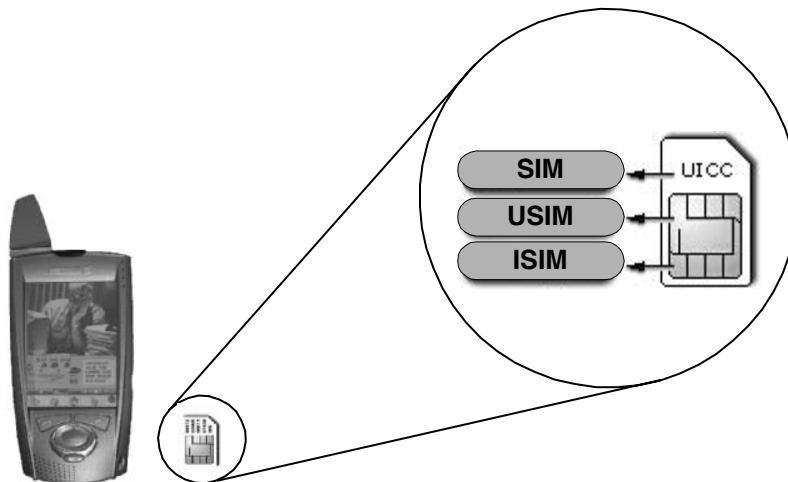


Figure 3.9: SIM, USIM, and ISIM in the UICC of 3GPP IMS terminals

3.6.1 SIM

SIM provides storage for a collection of parameters (e.g., user subscription information, user preferences, authentication keys, and storage of messages) that are essential for the operation of terminals in GSM networks. Although the terms UICC and SIM are often interchanged, UICC refers to the physical card, whereas SIM refers to a single application residing in the UICC that collects GSM user subscription information. SIM is widely used in 2G (second generation) networks, such as GSM networks.

The SIM application was standardized in the early stages of GSM. 3GPP inherited the specifications (currently SIM is specified in 3GPP TS 11.11 [21] and 3GPP TS 51.011 [2]).

3.6.2 USIM

USIM (standardized in 3GPP TS 31.102 [31]) is another example of an application that resides in third generation UICCs. USIM provides another set of parameters (similar in nature, but different from those provided by SIM), which include user subscriber information, authentication information, payment methods, and storage for messages. USIM is used to access UMTS networks, the third generation evolution of GSM.

A USIM is required if a circuit-switched or packet-switched terminal needs to operate in a 3G (third generation) network. Obviously, both SIM and USIM can co-exist in the same UICC, so that if the terminal is capable, it can use both GSM and UMTS networks.

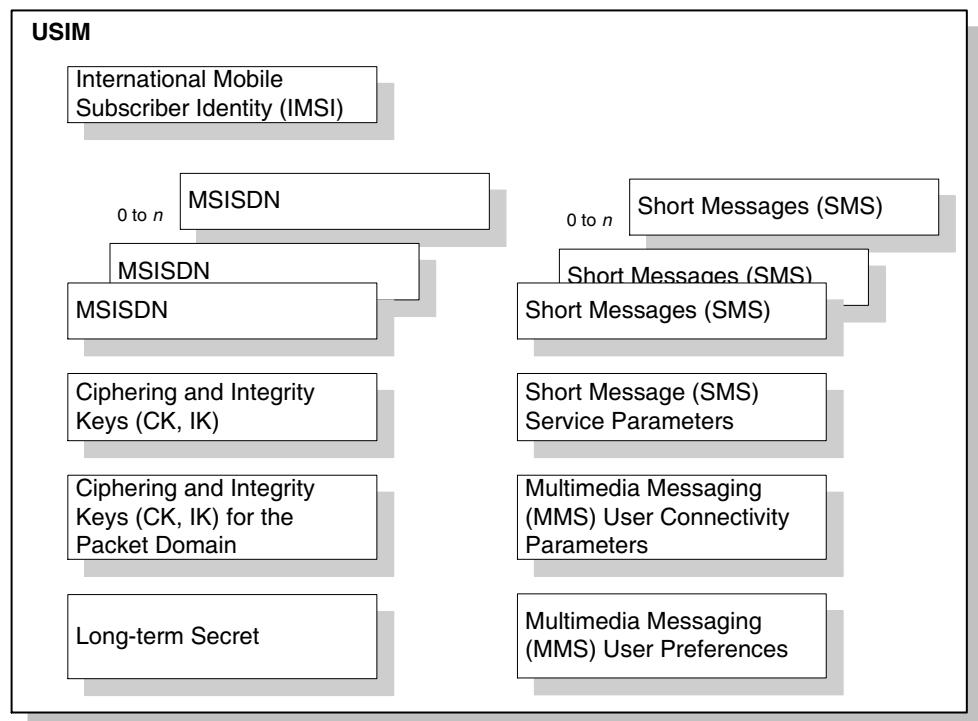


Figure 3.10: Simplified representation of the structure of the USIM application

Figure 3.10 shows a simplified version of the structure of USIM. USIM stores, among others, the following parameters.

IMSI (International Mobile Subscriber Identity). IMSI is an identity which is assigned to each user. This identity is not visible to the users themselves, but only to the network. IMSI is used as the user identification for authentication purposes. The Private User Identity is the equivalent of the IMSI in IMS.

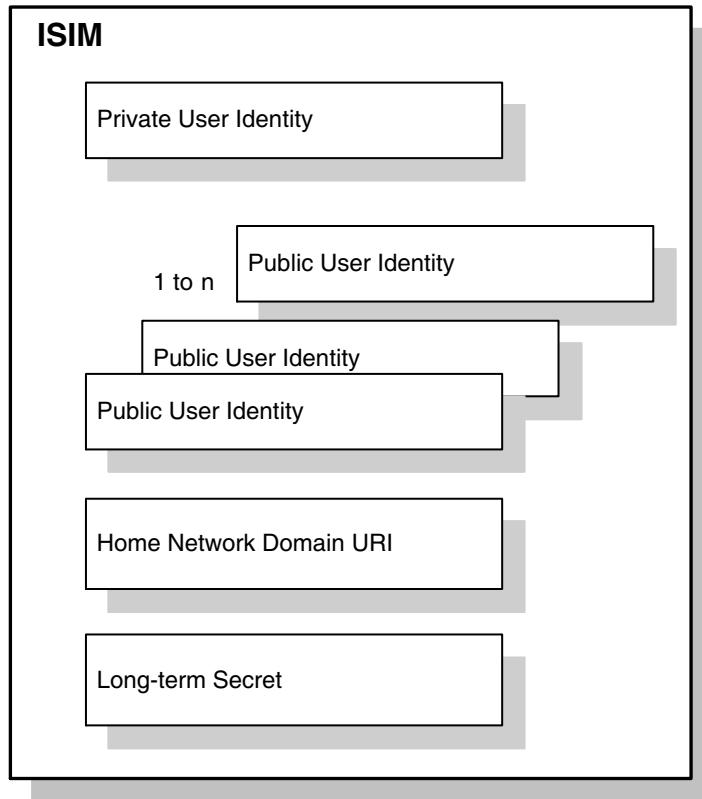


Figure 3.11: Structure of an ISIM application

MSISDN (Mobile Subscriber ISDN Number). This field stores one or more telephone numbers allocated to the user. A Public User Identity is the equivalent of the MSISDN in the IMS.

CK (Ciphering Key) and IK (Integrity Key). These are the keys used for ciphering and integrity protection of data over the air interface. USIM separately stores the keys used in circuit-switched and packet-switched networks.

Long-term secret. USIM stores a long-term secret that is used for authentication purposes and for calculating the integrity and cipher keys used between the terminal and the network.

SMS (Short Messages Service). USIM provides a storage for short messages and their associated data (e.g., sender, receiver and status).

SMS parameters. This field in the USIM stores configuration data related to the SMS service, such as the address of the SMS center or the protocols that are supported.

MMS (Multimedia Messaging Service) user connectivity parameters. This field stores configuration data related to the MMS service, such as the address of the MMS server and the address of the MMS gateway.

MMS user preferences. This field stores the user preferences related to the MMS service, such as the delivery report flag, read-reply preference, priority, and time of expiration.

3.6.3 ISIM

A third application that may be present in the UICC is ISIM (standardized in 3GPP TS 31.103 [10]). ISIM is of especial importance for the IMS, because it contains the collection of parameters that are used for user identification, user authentication and terminal configuration when the terminal operates in the IMS. ISIM can co-exist with a SIM, a USIM, or both applications in the same UICC.

Figure 3.11 depicts the structure of the ISIM application. The relevant parameters stored in ISIM are as follows.

Private User Identity. ISIM stores the Private User Identity allocated to the user. There can only be one Private User Identity stored in ISIM.

Public User Identity. ISIM stores one or more SIP URIs of Public User Identities allocated to the user.

Home Network Domain URI. ISIM stores the SIP URI that contains the home network domain name. This is used to find the address of the home network during the registration procedure. There can only be one home network domain name URI stored in ISIM.

Long-term secret. ISIM stores a long-term secret that is used for authentication purposes and for calculating the integrity and cipher keys used between the terminal and the network. The IMS terminal uses the integrity key to integrity-protect the SIP signaling that the IMS terminal sends to or receives from the P-CSCF. If the signaling is ciphered, the IMS terminal uses the cipher key to encrypt and decrypt the SIP signaling that the IMS terminal sends to or receives from the P-CSCF.

All of the above-mentioned fields are read-only, meaning that the user cannot modify the values of the parameters.

From the description of the fields contained in ISIM the reader has probably realized that ISIM is important for authenticating users. We describe in detail the access to the IMS and the authentication of users with an ISIM in Section 12.1.1.2.1.

Access to a 3GPP IMS network relies on the presence of either an ISIM or a USIM application in the UICC. ISIM is preferred because it is tailored to the IMS, although access with USIM is also possible. This allows operation in an IMS network of users who have not upgraded their UICCs to IMS-specific ones that contain an ISIM application. We describe in detail the access to the IMS and authentication with a USIM in Section 12.1.1.2.2.

Because of the lower degree of security contained in a SIM application, access to a 3GPP IMS network with a SIM application is not allowed. Non-3GPP IMS networks that do not support UICC in the IMS terminals (e.g., 3GPP2) store the parameters contained in the ISIM as part of the terminal's configuration or in the terminal's built-in memory.

3GPP2 IMS networks also allow the above-mentioned parameters to be stored in an R-UIM (Removable User Identity Module). The R-UIM is a smart card secure storage, equivalent to a 3GPP UICC with an ISIM application.

3.7 Next Generation Networks (NGN)

We earlier discussed that IMS is access-network independent, although 3GPP and 3GPP2 focused on making sure that their radio access networks were ready to accept IMS services. This section focuses on Next Generation Networks (NGN). NGN offers access to IMS services from fixed broadband accesses such as Asymmetric Digital Subscriber Lines (ADSL). Since NGN is a broad topic, we describe the main concepts of NGN. We then focus on the IMS aspects of NGN, and especially on the particulars of access to IMS from fixed broadband accesses.

NGN was originally standardized in the European Telecommunications Standards Institute (ETSI), in a standardization body named TISPAN (Telecoms and Internet converged Services and Protocols for Advanced Networks). A few changes to the common IMS architecture were required, and those were also discussed and eventually agreed in 3GPP. Because of the overlapping of the two bodies, and with the goal of having a single IMS standard, 3GPP and ETSI agreed in year 2007 that the common IMS standardization will take place in 3GPP. ETSI TISPAN will continue standardizing aspects of IMS which are not related to the common IMS (for example, the so-called PSTN/ISDN emulation subsystem).

This chapter focuses on the applicability of the IMS to Next Generation Networks defined by the European Telecommunications Standards Institute (ETSI) and later agreed by 3GPP. A detailed list of the ETSI specifications related to NGN can be found in Appendix A.3.

3.7.1 NGN Overview

We describe the general architecture of Next Generation Networks with the help of Figure 3.12. The figure schematically shows the existence of terminals that connect to an NGN. The network is divided into two main layers, namely the *service layer* and the *transport layer*. Each of the layers is composed of a number of subsystems that can be modularly plugged in as required, and a number of common functions. Some of the subsystems may also contain common functional elements that provide functions to more than a subsystem.

The NGN architecture allows for any distribution of the elements and subsystems in different networks. As such, it provides existence for an access network, a visited network, and a home network, each one providing a different type of service.

The transport layer is responsible for providing the layer 2 connectivity, IP connectivity, and transport control. The transport layer is further divided into the Network Attachment Subsystem (NASS), the Resource and Admission Control Subsystem (RACS), and a number of common transfer functions.

NASS is responsible for supplying the terminal with an IP address, together with configuration parameters, providing authentication at the IP layer, authorizing network access and access network configuration based on users' profiles, and for the location manager at the IP layer.

RACS is responsible for providing resource management and admission control. Among other functions, RACS provides gate control functionality, policy enforcement, and admission control based on user profiles.

The transfer functions contain a number of functional elements that are visible and sometimes controlled by functional elements of the NASS or RACS. For instance, media gateways, border gateways, etc., are examples of transfer functions.

The service layer contains a number of subsystems that provide the platform for enabling services to the user. Prior to describing each of the subsystems, we need to define the concepts of PSTN/ISDN *emulation* and PSTN/ISDN *simulation*.

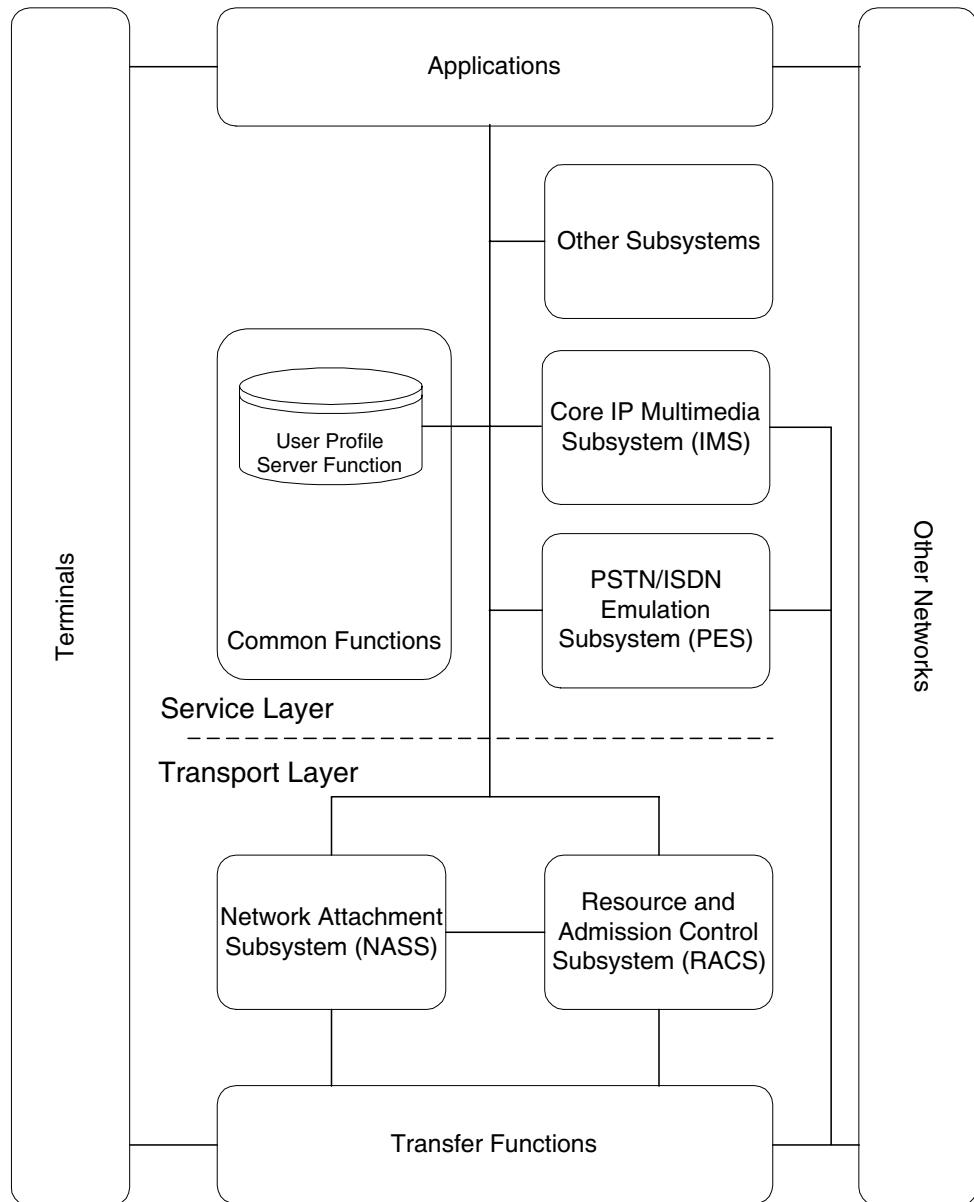


Figure 3.12: NGN architecture overview

The term *PSTN/ISDN emulation* is used to refer to an NGN that implements the same services that are today provided in the PSTN and ISDN. Therefore, PSTN/ISDN emulation implementations aim to replace the PSTN/ISDN core network without replacing the terminals. Users connected to an NGN providing PSTN/ISDN emulation have exactly the same services as they have in the regular PSTN/ISDN without noticing that an NGN is actually delivering the service.

The term *PSTN/ISDN simulation* is used to refer to an NGN that is providing telecommunication services compatible with the PSTN/ISDN, but not necessarily being exactly the same. The concept also indicates not only a replacement of the PSTN/ISDN, but also a replacement of the terminals that support further capabilities compared with a regular PSTN/ISDN phone.

So the service layer comprises a number of subsystems, of which two are defined in Release 1 of NGN; others are being defined in future releases as the need arises.

The PSTN/ISDN Emulation Subsystem (PES) implements the PSTN/ISDN emulation concept. PES allows users to receive the same services that they are currently receiving in PSTN/ISDN networks with the existing PSTN/ISDN terminals. PES can be implemented either as a monolithic softswitch or as a distributed IMS. In the case of the distributed IMS, since services do not change, SIP requests and responses carry ISUP bodies. Network elements that provide services (e.g., Application Servers) read the ISUP body to provide a service to the user.

The core IMS implements the PSTN/ISDN simulation concept. The core IMS enables SIP-based multimedia services to NGN terminals. The core IMS enables services, some of which may be new multimedia services (such as presence, instant messaging, etc.), while others might be more traditional telephony services. The core IMS is largely based on the 3GPP IMS specifications. We describe in more detail the core IMS and its services in Section 3.7.2.

The service layer in NGN also provides for the existence of applications that are typically implemented in Application Server Functions (ASFs).

A number of common functions provide functional services to several subsystems. This is the case with the User Profile Server Function (UPSF), which is a database that contains user-specific information; this is similar to what the HSS is to the IMS.

Other subsystems beyond the PES and the core IMS can be standardized in the future. For example, NGN could support a streaming subsystem or a content broadcasting subsystem.

3.7.2 *The Core IMS in NGN*

As we mentioned before, the core IMS is largely based on the 3GPP IMS specifications; however the core IMS in NGN considers only SIP network elements such as CSCFs, BGCF, MGCF, and MRFC. In particular, ASes, MRFP, MGW, user databases, etc., are considered to be outside the core IMS, although they are all present in NGN either as part of the common functions or as part any of the subsystems of the transport layer.

In this section we focus on the new functionality that has been added to the IMS owing to fixed broadband access. Typically, 3GPP has accepted changes to its IMS specifications to adopt new functionality due to fixed broadband access, so on most occasions the latest version of the 3GPP specifications describe the case of accesses to IMS over broadband fixed access.

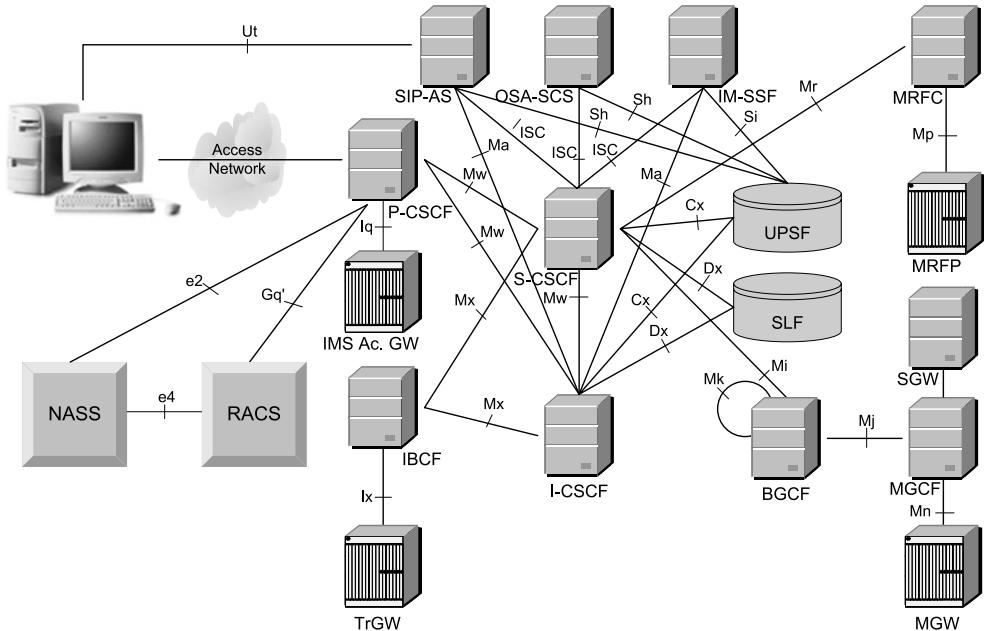


Figure 3.13: NGN: core IMS architecture

Figure 3.13 shows the core IMS architecture in NGN and the interfaces with adjacent nodes. As expected, the core IMS architecture is very similar to the 3GPP IMS architecture, because most of the nodes are present in the core IMS.

The core IMS adds new interfaces to the P-CSCF to communicate with the functional elements of the low transport-layer subsystems, the RACS and NASS. The P-CSCF includes a new Gq' interface towards the RACS for the purpose of requesting authorization of QoS resources, reserving resources, and providing control of gates in the transport layer. The Gq' interface is based on the 3GPP Gq interface specified in 3GPP TS 29.209 [19], although IMS has evolved and Gq has been replaced with the Rx interface. The P-CSCF also implements a new $e2$ interface towards the NASS for the purpose of retrieving user's location information. Both Gq' and $e2$ interfaces are based on the Diameter protocol.

In most cases, especially in fixed networks, an ADSL router located in the customer premises acts as a NAT (Network Address Translation). The NAT is located in between the terminal and the first signaling point: the P-CSCF. The function of the NAT is to translate IP addresses allocated from a private space numbering to globally routable IP addresses (see Section 4.20). Sometimes, not only the IP address but also the port number changes when an IP packet traverses a NAT. Thus, a NAT rewrites the source or destination IP address and perhaps also the source and destination port numbers present in the IP header of an IP packet. What NATs typically do not do is to appropriately rewrite IP addresses and port numbers that appear in SIP header fields (e.g., Contact, Via), SDP lines such as the $c=$ or $m=$ lines, and various other places where IP addresses and port numbers are explicitly signaled.

To assist in the task of NAT traversal for those terminals that do not support NAT traversal capabilities, the P-CSCF can include an IMS Application Level Gateway (IMS-ALG). The IMS-ALG translates IPv4 addresses and port numbers from the private to the public address

space (and vice versa), and provides control for the network address and port translator functions located in the IMS Access Gateway. Figure 3.14 shows the IMS-ALG embedded in the P-CSCF controlling the IMS Access Gateway.

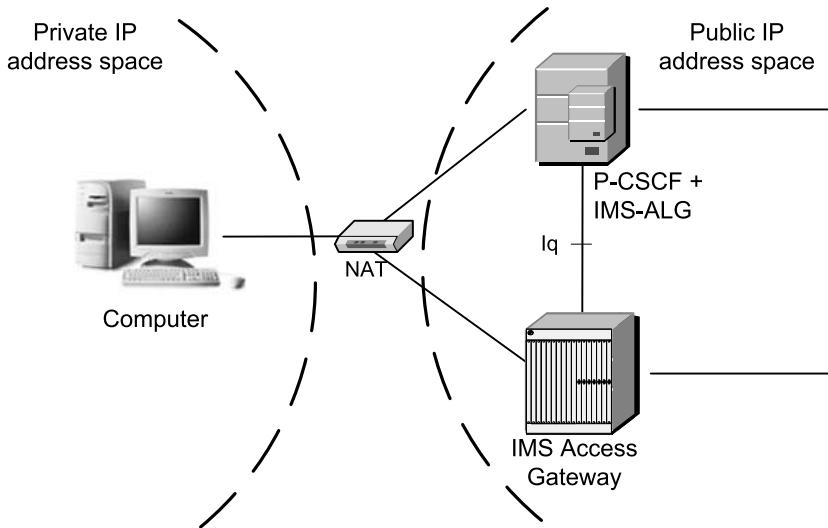


Figure 3.14: An IMS-ALG embedded in a P-CSCF supports NAT traversal

Outside the core IMS, but still present in the NGN architecture, are the User Profile Server Function (UPSF), the Subscriptor Locator Function (SLF), the Interconnection Border Control Function (IBCF), the Interworking Function (IWF), and Application Server Functions (ASFs). The distinction between what is considered to be part of IMS and what is outside IMS but still part of NGN is a bit fuzzy, so do not be surprised if the considerations change as time passes.

The UPSF in NGN is similar to the HSS in IMS. The main difference lies in the fact that the HSS, since it is an evolution of the GSM HLR, includes an HLR/AUC that provides mobility management. This is not required in NGN; therefore the UPSF is limited to the IMS-specific parts of the HSS. Besides that, the external interfaces are common to the HSS and the UPSF, and the database function for IMS users is the same in both cases.

The SLF has the same functionality as its counterpart in the IMS.

The IBCF is a new functional entity introduced by NGN, which, as with most changes to IMS, has been also adopted by 3GPP IMS. The IBCF acts as a separation between two different domains. Effectively, the IBCF is a session border controller, typically the first SIP node that gets a SIP request from an external domain, replacing the I-CSCF as the original first entry point. The IBCF can also be the last node in the signaling path prior to the forwarding of the SIP request to an external domain. The main functionality of the IBCF is to screen the signaling, and obfuscate those SIP headers that the operator considers dangerous to expose externally. The IBCF may also integrate an Interworking Function (IWF) which can provide interworking with other signaling protocols, such as H.323.

In case there is a need for IP version interworking, the IBCF may provide functionality similar to the IMS-ALG for translating, for example, IPv4 addresses to IPv6 addresses, and vice versa. In this scenario, the IBCF is also responsible for inserting a Transition Gateway

(TrGW) in the media path when it is needed. The Transition Gateway is responsible for the translation of IPv4 to IPv6 in the media path (e.g., RTP). Figure 3.15 represents an IBCF located at the edge of the internal network, controlling a Transition Gateway.

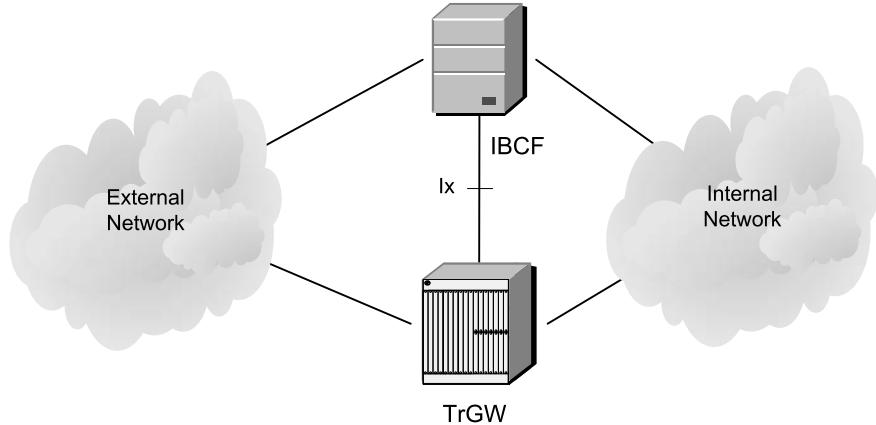


Figure 3.15: An IBCF is the border between internal and external domains

Application Server Functions (ASFs) execute services. NGN identifies two types of ASF, namely ASF Type 1 and Type 2. The ASF Type 1 may interact with the RACS when providing a service to the user. ASF Type 2 merely relies on the call control protocol to provide the service. ASF Type 2 is functionally equivalent to the AS in the IMS.

In addition to the mentioned nodes, NGN provide for the existence of charging and data collection functions. These include data collection and mediation functions to a billing system. NGN reuses most of the charging functionality available in the IMS.

Part II

The Signaling Plane in the IMS

As we saw in Chapter 3 the IMS has a signaling plane and a media plane that traverse different paths. This part of the book (Part II) tackles the signaling plane in the IMS, while Part III tackles the media plane.

We explained in Chapter 2 that one of the goals of the IMS is to use Internet technologies and protocols as much as possible. One of the main advantages of Internet protocols is that they are designed to be general enough to work within a wide range of architectures. While we are mainly interested in showing how these protocols are used in the IMS, it is essential to understand how they are used in other environments as well.

So, we will divide the explanation of every protocol into two chapters. First, we describe how each protocol works in the Internet environment and how it can be extended for use in architectures with different requirements from the public Internet (e.g., the IMS). Once we have introduced a particular protocol and its design principles in a chapter, we can study how that protocol is used in the IMS in the following chapter, which protocol extensions are used, and how that protocol fits into the IMS architecture. Following these steps in the explanation of every protocol gives the reader a broad perspective of the Internet technologies used in the IMS. Understanding the similarities and differences in the operation of a particular protocol on the Internet and in the IMS will help the reader to understand how the IMS provides Internet services to its users.

Chapter 4

Session Control on the Internet

Many think that the most important component of the signaling plane is the protocol that performs session control. The protocol chosen to perform this task in the IMS is the Session Initiation Protocol (SIP) (defined in RFC 3261 [286]).

SIP was originally developed within the SIP working group in the IETF. Even though SIP was initially designed to invite users to existing multimedia conferences, today it is mainly used to create, modify and terminate multimedia sessions. In addition, there exist SIP extensions to deliver instant messages and to handle subscriptions to events. We will first look at the core protocol (used to manage multimedia sessions), and then we will deal with the most important extensions.

4.1 SIP Functionality

Protocols developed by the IETF have a well-defined scope. The functionality to be provided by a particular protocol is carefully defined in advance before any working group starts working on it. In our case the main goal of SIP is to deliver a session description to a user at their current location. Once the user has been located and the initial session description delivered, SIP can deliver new session descriptions to modify the characteristics of the ongoing sessions and terminate the session whenever the user wants.

4.1.1 Session Descriptions and SDP

A session description is, as its name indicates, a description of the session to be established. It contains enough information for the remote user to join the session. In multimedia sessions over the Internet this information includes the IP address and port number where the media needs to be sent, and the codecs used to encode the voice and the images of the participants.

Session descriptions are created using standard formats. The most common format for describing multimedia sessions is the Session Description Protocol (SDP), defined in RFC 2327 [160]. Note that although the “P” in SDP stands for “Protocol”, SDP is simply a textual format to describe multimedia sessions. Figure 4.1 shows an example of an SDP session description that Alice sent to Bob. It contains, among other things, the subject of the conversation (swimming techniques), Alice’s IP address (192.0.0.1), the port number where Alice wants to receive audio (20000), the port number where Alice wants to receive video

```
v=0
o=Alice 2790844676 2867892807 IN IP4 192.0.0.1
s=Let's talk about swimming techniques
c=IN IP4 192.0.0.1
t=0 0
m=audio 20000 RTP/AVP 0
a=sendrecv
m=video 20002 RTP/AVP 31
a=sendrecv
```

Figure 4.1: Example of an SDP session description

(20002), and the audio and video codecs that Alice supports (0 corresponds to the audio codec G.711 μ -law and 31 corresponds to the video codec H.261).

As we can see in Figure 4.1 an SDP description consists of two parts: session-level information and media-level information. The session-level information applies to the whole session and comes before the *m*= lines. In our example, the first five lines correspond to session-level information. They provide version and user identifiers (*v*= and *o*= lines), the subject of the session (*s*= line), Alice’s IP address (*c*= line), and the time of the session (*t*= line). Note that this session is supposed to take place at the moment when this session description is received. That is why the *t*= line is *t*=0 0.

The media-level information is media-stream specific and consists of an *m*= line and a number of optional *a*= lines that provide further information about the media stream. Our example has two media streams and, thus, has two *m*= lines. The *a*= lines indicate that the streams are bidirectional (i.e., users send and receive media).

As Figure 4.1 illustrates, the format of all the SDP lines consists of *type=value*, where type is always one character long. Table 4.1 shows all the types defined by SDP.

Even if SDP is the most common format to describe multimedia sessions, SIP does not depend on it. SIP is session-description format independent. That is, SIP can deliver a description of a session written in SDP or in any other format. For example, after the video conversation above about swimming techniques, Alice feels like inviting Bob to a real training session this evening in the swimming pool next to her place. She uses a session description format for swimming sessions to create a session description and uses SIP to send it to Bob. Alice’s session description looks something like the one in Figure 4.2.

This example is intended to stress that SIP is completely independent of the format of the objects it transports. Those objects may be session descriptions written in different formats or any other piece of information. We will see in subsequent sections that SIP is also used to deliver instant messages, which of course are written using a different format from SDP and from our description format for swimming sessions.

4.1.2 The Offer/Answer Model

In the SDP example in Figure 4.1, Alice sent a session description to Bob that contained Alice’s transport addresses (IP address plus port numbers). Obviously, this is not enough to establish a session between them. Alice needs to know Bob’s transport addresses as well. SIP provides a two-way session-description exchange called the offer/answer model (which is described in RFC 3264 [283]). One of the users (the offerer) generates a session description

Table 4.1: SDP types

Type	Meaning
v	Protocol version
b	Bandwidth information
o	Owner of the session and session identifier
z	Time zone adjustments
s	Name of the session
k	Encryption key
i	Information about the session
a	Attribute lines
u	URL containing a description of the session
t	Time when the session is active
e	Email address to obtain information about the session
t	Times when the session will be repeated
p	Phone number to obtain information about the session
m	Media line
c	Connection information
i	Information about the media line

Subject: Swimming Training Session
 Time: Today from 20:00 to 21:00
 Place: Lane number 4 of the swimming-pool near my place

Figure 4.2: Example of a session description without SDP being used

(the offer) and sends it to the remote user (the answerer), who then generates a new session description (the answer) and sends it to the offerer.

RFC 3264 [283] provides the rules for offer and answer generation. After the offer/answer exchange, both users have a common view of the session to be established. They know, at least, the formats they can use (i.e., formats that the remote end understands) and the transport addresses for the session. The offer/answer exchange can also provide extra information, such as cryptographic keys to encrypt traffic.

Figure 4.3 shows the answer that Bob sent to Alice after having received Alice's offer in Figure 4.1. Bob's IP address is 192.0.0.2, the port number where Bob will receive audio is 30000, the port number where Bob will receive video is 30002, and, fortunately, Bob supports the same audio and video codecs as Alice (G.711 μ -law and H.261). After this offer/answer exchange, all they have left to do is to have a nice video conversation.

4.1.3 SIP and SIPS URIs

SIP identifies users using SIP URIs, which are similar to email addresses; they consist of a username and a domain name. In addition, SIP URIs can contain a number of parameters

```
v=0
o=Bob 234562566 236376607 IN IP4 192.0.0.2
s=Let's talk about swimming techniques
c=IN IP4 192.0.0.2
t=0 0
m=audio 30000 RTP/AVP 0
a=sendrecv
m=video 30002 RTP/AVP 31
a=sendrecv
```

Figure 4.3: Bob's SDP session description

(e.g., transport), which are encoded using semicolons. The following are examples of SIP URIs:

```
sip:Alice.Smith@domain.com
sip:Bob.Brown@example.com
sip:carol@ws1234.domain2.com;transport=tcp
```

In addition, users can be identified using SIPS URIs. Entities contacting a SIPS URI use TLS (Transport Layer Security, see Section 11.3) to secure their messages. The following are examples of SIPS URIs:

```
URI}sips:Alice.Smith@domain.com
URI}sips:Bob.Brown@example.com
```

4.1.4 User Location

We said earlier that the main purpose of SIP is to deliver a session description to a user at their current location, and we have already seen what a session description looks like. Now let us look at how SIP tracks the location of a given user.

SIP provides personal mobility. That is, users can be reached using the same identifier no matter where they are. For example, Alice can be reached at

```
sip:Alice.Smith@domain.com
```

regardless of her current location. This is her public URI, also known as her AoR (Address of Record).

Nevertheless, when Alice is logged in at work her SIP URI is

```
sip:asmith@ws1234.company.com
```

and when she is working at her computer at the university her SIP URI is

```
sip:alice@pc12.university.edu
```

Therefore, we need a way to map Alice's public URI

```
sip:Alice.Smith@domain.com
```

to her current URI (at work or at the university) at any given moment.

To do this, SIP introduces a network element called the registrar of a particular domain. A registrar handles requests addressed to its domain. Thus, SIP requests sent to

`sip:Alice.Smith@domain.com`

will be handled by the SIP registrar at domain.com.

Every time Alice logs into a new location, she registers her new location with the registrar at domain.com, as shown in Figure 4.4. This way the registrar at domain.com can always forward incoming requests to Alice wherever she is.

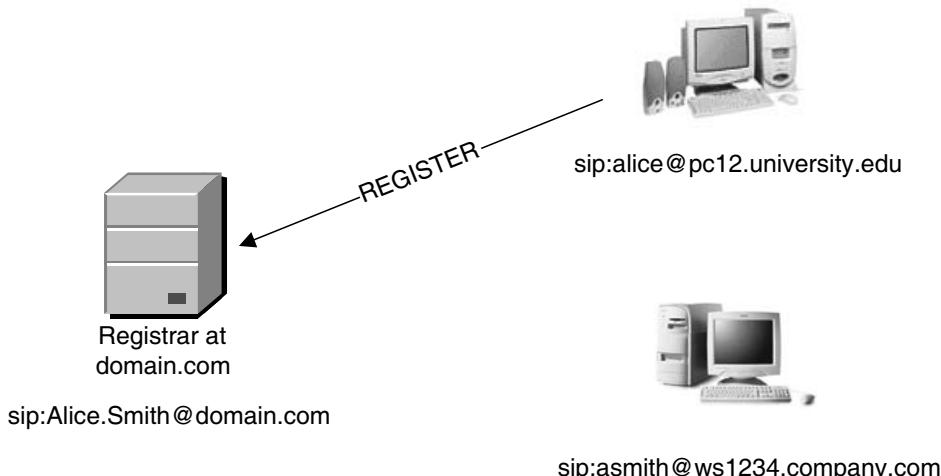


Figure 4.4: Alice registers her location with the domain.com registrar

On reception of the registration the registrar at domain.com can store the mapping between Alice's public URI and her current location in two ways: it can use a local database or it can upload this mapping into a location server. If the registrar uses a location server, it will need to consult it when it receives a request for Alice. Note that the interface between the registrar and the location server is not based on SIP, but on other protocols.

4.2 SIP Entities

Besides the registrars, which were introduced in the previous section, SIP defines user agents, proxy servers, and redirect servers. UAs (user agents) are SIP endpoints that are usually handled by a user. In any case, user agents can also establish sessions automatically with no user intervention (e.g., a SIP voicemail). Sessions are typically established between user agents.

User agents come in all types of flavors. Some are software running on a computer, others, like the commercial SIP phones shown in Figure 4.5, look like desktop phones, and others still are embedded in mobile devices like laptops, PDAs, or mobile phones. Some of them are not even used for telephony and do not have speakers or microphones.

Proxy servers, typically referred to as proxies, are SIP routers. A proxy receives a SIP message from a user agent or from another proxy and routes it toward its destination.



Figure 4.5: Three examples of commercial SIP phones

Routing the request involves relaying the message to the destination user agent or to another proxy in the path.

It is important to understand fully how SIP routing works, because it is one of the key components of the protocol. A given user can be available at several user agents at the same time. For instance, Alice can be reachable on her computer at the university

`sip:alice@pc12.university.edu`

and on her PDA with a wireless connection

`sip:alice@pda.com`

She has registered both locations with the registrar at domain.com. If the registrar receives a SIP message addressed to Alice's public URI

`sip:Alice.Smith@domain.com`

it has to decide whether to route it to Alice's computer or to Alice's PDA. In this case, Alice has programmed the registrar to route SIP messages to her computer between 8:00 and 13:00

and to her PDA from 13:00 to 14:00. The registrar simply checks the current time and routes the SIP message accordingly.

Being able to route SIP messages on the basis of any criteria is a very powerful tool for building services that are specially tailored to the needs of each user. Users typically choose to route SIP messages based on the sender, the time of the day, whether the subject is business-related or personal, the type of session (e.g., route video calls to the computer with the big screen), etc.; the combinations are infinite.

In the previous example we saw that the registrar routed the SIP message to Alice's user agent. Yet the entities handling routing of messages are called proxies. Proxies and registrars are only logical roles. In our example, the same physical box acted as a registrar when Alice registered her current location and as a proxy when it was routing SIP messages toward Alice's user agent. This configuration is shown in Figure 4.6.

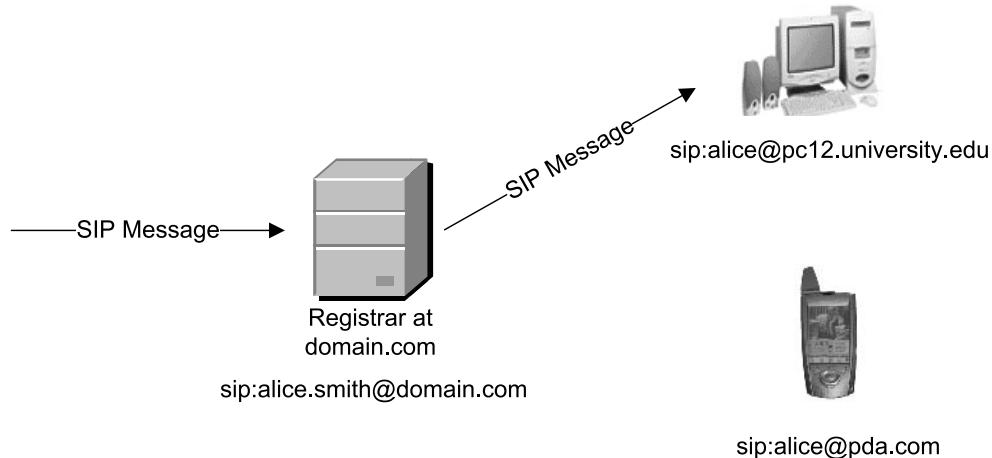


Figure 4.6: Proxy co-located with the registrar of the domain

A different configuration could consist of using a separate physical box for each role, as shown in Figure 4.7. Here, the proxy needs to access the information about Alice's location that the registrar got in the first place. This is resolved by adding a location server. The registrar uploads Alice's location to the location server, and the proxy consults the location server in order to route incoming messages.

4.2.1 Forking Proxies

In the previous examples the proxy chose a single user agent as the destination of the SIP message. However, sometimes it is useful to receive calls on several user agents at the same time. For instance, in a house with a single line, all the telephones ring at once, giving us the chance to pick up the call in the kitchen or in the living room. SIP proxy servers that route messages to more than one destination are called forking proxies, as shown in Figure 4.8.

A forking proxy can route messages in parallel or in sequence. An example of parallel forking is the simultaneous ringing of all the telephones in a house. Sequential forking consists of the proxy trying the different locations one after the other. A proxy can, for example, let a user agent ring for a certain period of time and, if the user does not pick up, try a new user agent.

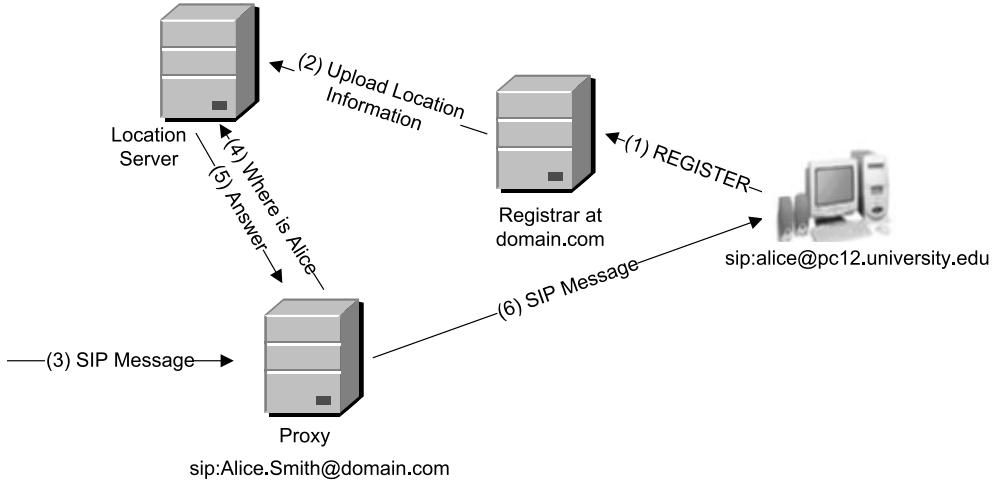


Figure 4.7: Proxy and registrar kept separate

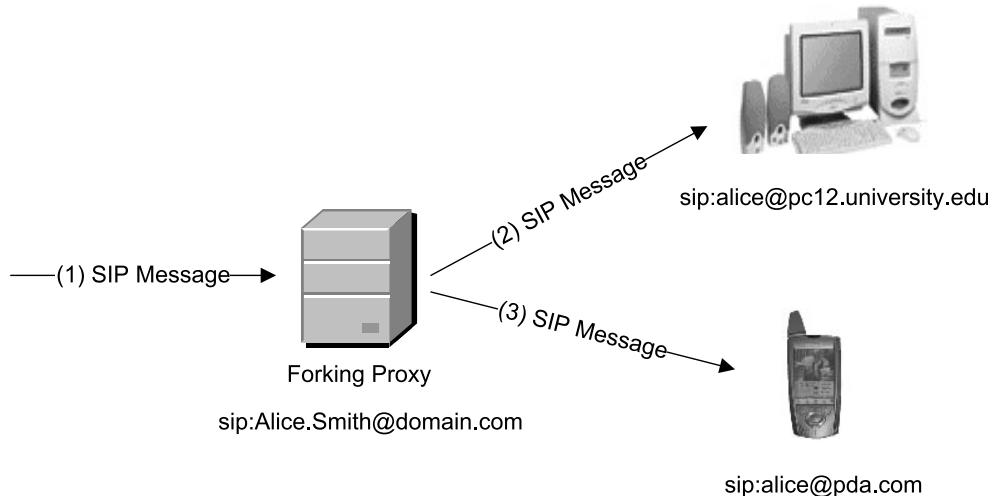


Figure 4.8: Forking proxy operation

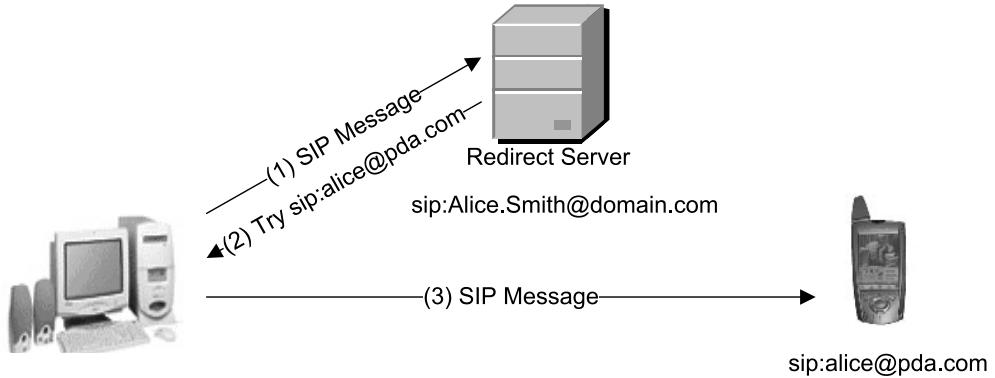
4.2.2 Redirect Servers

Redirect servers are also used to route SIP messages, but they do not relay the message to its destination as proxies do. Redirect servers instruct the entity that sent the message (a user agent or a proxy) to try a new location instead. Figure 4.9 shows how redirect servers work. A user agent sends a SIP message to

`sip:Alice.Smith@domain.com`

and the redirect server tells it to try the alternative address

`sip:alice@pda.com`

**Figure 4.9:** Redirect server operation

4.3 Message Format

SIP is based on HTTP [144] and so it is a textual request-response protocol. Clients send requests, and servers answer with responses. A SIP transaction consists of a request from a client, zero or more provisional responses, and a final response from a server. We will introduce the format of SIP requests and responses before explaining, in Section 4.8, the types of transactions that SIP defines.

Figure 4.10 shows the format of SIP messages. They start with the *start line*, which is called the *request line* in requests and the *status line* in responses. The *start line* is followed by a number of header fields that follow the format *name:value* and an empty line that separates the header fields from the optional message body.

```

Start line
A number of header fields
Empty line
Optional message body
  
```

Figure 4.10: SIP message format

4.4 The Start Line in SIP Responses: the Status Line

As we said earlier the start line of a response is referred to as the status line. The status line contains the protocol version (SIP/2.0) and the status of the transaction, which is given in numerical (status code) and human-readable (reason phrase) formats. The following is an example of a status line:

SIP/2.0 180 Ringing

The protocol version is always set to SIP/2.0 (a history of previous versions of the protocol is given in *SIP Demystified* [97]). We will see in Section 4.11 how SIP is extended without it being necessary to increase its protocol version.

The status code 180 indicates that the remote user is being alerted. Ringing is the reason phrase and it is intended to be read by a human (e.g., displayed to the user). Since it is intended for human consumption the reason phrase can be written in any language.

Responses are classified by their status codes, which are integers that range from 100 to 699. Table 4.2 shows how status codes are classified according to their values.

Table 4.2: Status code ranges

Status code range	Meaning
100–199	Provisional (also called informational)
200–299	Success
300–399	Redirection
400–499	Client error
500–599	Server error
600–699	Global failure

Apart from the start line (status line in responses and request line in requests) the format of requests and responses is identical, as shown in Figure 4.10. So, let us now tackle the format of the request line and then the format of the rest of the message.

4.5 The Start Line in SIP Requests: the Request Line

The start line in requests is referred to as the request line. It consists of a *method name*, the *Request-URI*, and the protocol version SIP/2.0. The method name indicates the purpose of the request and the *Request-URI* contains the destination of the request. Below, is an example of a request line:

```
INVITE sip:Alice.Smith@domain.com SIP/2.0
```

The method name in this example is INVITE. It indicates that the purpose of this request is to invite a user to a session. The *Request-URI* shows that this request is intended for Alice.

Table 4.3 shows the methods that are currently defined in SIP and their meaning.

Figure 4.11 shows a SIP transaction. The user agent client (UAC) sends a BYE request, and the user agent server (UAS) sends back a 200 (OK) response. Note that, usually, SIP message flows only show the method name of the request and the status code and the reason phrase of the response. These pieces of information are usually enough for any message flow to be understood.

Before explaining the types of SIP transactions and how to use them, we will study the formats of SIP header fields and bodies. After that, we will provide the readers with some message flows that will help them to understand how to perform useful tasks, such as establishing a session using SIP.

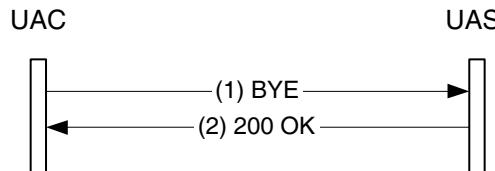
4.6 Header Fields

Right after the start line, SIP messages (both requests and responses) contain a set of header fields (see Figure 4.10). There are mandatory header fields that appear in every message and optional header fields that only appear when needed. A header field consists of the header field's name, a colon, and the header field's value, as shown in the example below:

```
To: Alice Smith <sip:Alice.Smith@domain.com>;tag=1234
```

Table 4.3: SIP methods

Method name	Meaning
ACK	Acknowledges the reception of a final response for an INVITE
BYE	Terminates a session
CANCEL	Cancels a pending request
INFO	Transports PSTN telephony signaling
INVITE	Establishes a session
NOTIFY	Notifies the user agent about a particular event
OPTIONS	Queries a server about its capabilities
PRACK	Acknowledges the reception of a provisional response
PUBLISH	Uploads information to a server
REGISTER	Maps a public URI with the current location of the user
SUBSCRIBE	Requests to be notified about a particular event
UPDATE	Modifies some characteristics of a session
MESSAGE	Carries an instant message
REFER	Instructs a server to send a request

**Figure 4.11:** SIP transaction

As we can see, the value of a header field can consist of multiple items. The To header field above contains a display name (Alice Smith), a URI

`sip: Alice.Smith@domain.com`

and a tag parameter.

Some header fields can have more than one entry in the same message, as shown in the example below:

```
Route: <sip:p1.domain1.com>
Route: <sip:p34.domain2.com>
```

Multi-entry header fields can appear in a single-value-per-line form, as shown above, or in a comma-separated value form, as shown below. Both formats are equivalent.

`Route: <sip:p1.domain1.com>, <sip:p34.domain2.com>`

Note that in all the examples so far there is a space between the colon and the value of the header field. In the example above, we can also see a space after the comma separating the Route entries. SIP parsers ignore these spaces, but they are typically included in the messages to improve their readability for humans.

Let us have a look at the most important SIP header fields: the six mandatory header fields that appear in every SIP message. They are **To**, **From**, **Cseq**, **Call-ID**, **Max-Forwards**, and **Via**.

To. The **To** header field contains the URI of the destination of the request. However, this URI is not used to route the request. It is intended for human consumption and for filtering purposes. For example, a user can have a private URI and a professional URI and requests can be filtered depending on which URI appears in the **To** field. The **tag** parameter is used to distinguish, in the presence of forking proxies, different user agents that are identified with the same URI.

From. The **From** header field contains the URI of the originator of the request. Like the **To** header field, it is mainly used for human consumption and for filtering purposes.

Cseq. The **Cseq** header field contains a sequence number and a method name. They are used to match requests and responses.

Call-ID. The **Call-ID** provides a unique identifier for a SIP message exchange.

Max-Forwards. The **Max-Forwards** header field is used to avoid routing loops. Every proxy that handles a request decrements its value by one, and if it reaches zero, the request is discarded.

Via. The **Via** header field keeps track of all the proxies a request has traversed. The response uses these **Via** entries so that it traverses the same proxies as the request did in the opposite direction.

4.7 Message Body

As Figure 4.10 shows, the message body is separated from the header fields by an empty line. SIP messages can carry any type of body and even multipart bodies using MIME (Multipurpose Internet Mail Extensions) encoding.

RFC 2045 [146] defines the MIME format which allows us to send emails with multiple attachments in different formats. For example, a given email message can carry a JPEG picture and an MPEG video as attachments.

SIP uses MIME to encode its message bodies. Consequently, SIP bodies are described in the same way as attachments to an email message. A set of header fields provide information about the body: its length, its format, and how it should be handled. For example, the header fields below describe the SDP session description of Figure 4.1:

```
Content-Disposition: session
Content-Type: application/sdp
Content-Length: 193
```

The **Content-Disposition** indicates that the body is a session description, the **Content-Type** indicates that the session description uses the SDP format, and the **Content-Length** contains the length of the body in bytes.

Figure 4.12 shows an example of a multipart body encoded using MIME. The first body part is an SDP session description and the second body part consists of the text “This is the second body part”. Note that the **Content-Type** for the whole body is **multipart/mixed**

```

Content-Type: multipart/mixed; boundary="0806040504000805090"
Content-Length: 384

--0806040504000805090
Content-Type: application/sdp
Content-Disposition: session

v=0
o=Alice 2790844676 2867892807 IN IP4 192.0.0.1
s=Let's talk about swimming techniques
c=IN IP4 192.0.0.1
t=0 0
m=audio 20000 RTP/AVP 0
a=sendrecv
m=video 20002 RTP/AVP 31
a=sendrecv
--0806040504000805090--
Content-Type: text/plain

This is the second body part
--0806040504000805090--

```

Figure 4.12: MIME encoding of a multipart body

and that each body part has its own Content-Type, namely application/sdp and text/plain.

An important property of bodies is that they are transmitted end-to-end. That is, proxies do not need to parse the message body in order to route the message. In fact, the user agents may choose to encrypt the contents of the message body end-to-end. In this case, proxies would not even be able to tell which type of session was being established between both user agents.

4.8 SIP Transactions

Now that we know all the elements in a SIP network and the elements of SIP messages, we can study the three types of transaction that SIP defines: regular transactions, INVITE–ACK transactions, and CANCEL transactions. The type of a particular transaction depends on the request initiating it.

Regular transactions are initiated by any request except INVITE, ACK, or CANCEL. Figure 4.13 shows a regular BYE transaction. In a regular transaction, the user agent server receives a request and generates a final response that terminates the transaction. In theory, it would be possible for the user agent server to generate one or more provisional responses before generating the final response, although, in practice, provisional responses are seldom sent within a regular transaction.

An INVITE–ACK transaction involves two transactions: an INVITE transaction and an ACK transaction, as shown in Figure 4.14. The user agent server receives an INVITE request and generates zero or more provisional responses and a final response. When the user agent

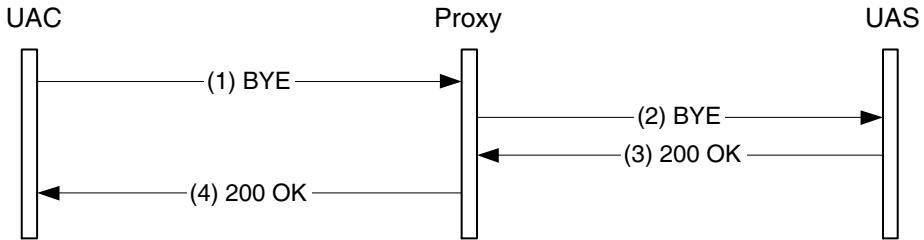


Figure 4.13: Regular transaction

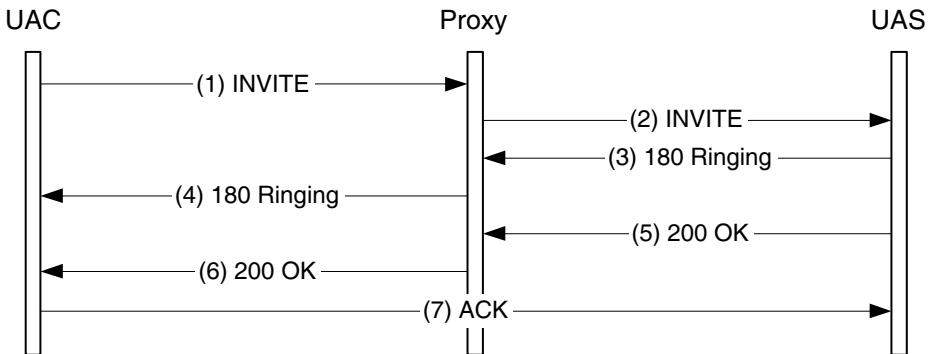


Figure 4.14: INVITE-ACK transaction

client receives the final response, it generates an ACK request, which does not have any response associated with it.

CANCEL transactions are initiated by a CANCEL request and are always connected to a previous transaction (i.e., the transaction to be cancelled). CANCEL transactions are similar to regular transactions, with the difference that the final response is generated by the next SIP hop (typically a proxy) instead of by the user agent server. Figure 4.15 shows a CANCEL transaction cancelling an INVITE transaction. Note that the INVITE transaction, once it is cancelled, terminates as usual (i.e., final response plus ACK).

4.9 Message Flow for Session Establishment

Now that we have introduced the different types of SIP transaction, let us see how we can use SIP to establish a multimedia session. First of all, Alice registers her current location

`sip:alice@pda.com`

with the registrar at domain.com, as shown in Figure 4.16. To do this, Alice sends a REGISTER request (Figure 4.17) indicating that requests addressed to the URI in the To header field

`sip:Alice.Smith@domain.com`

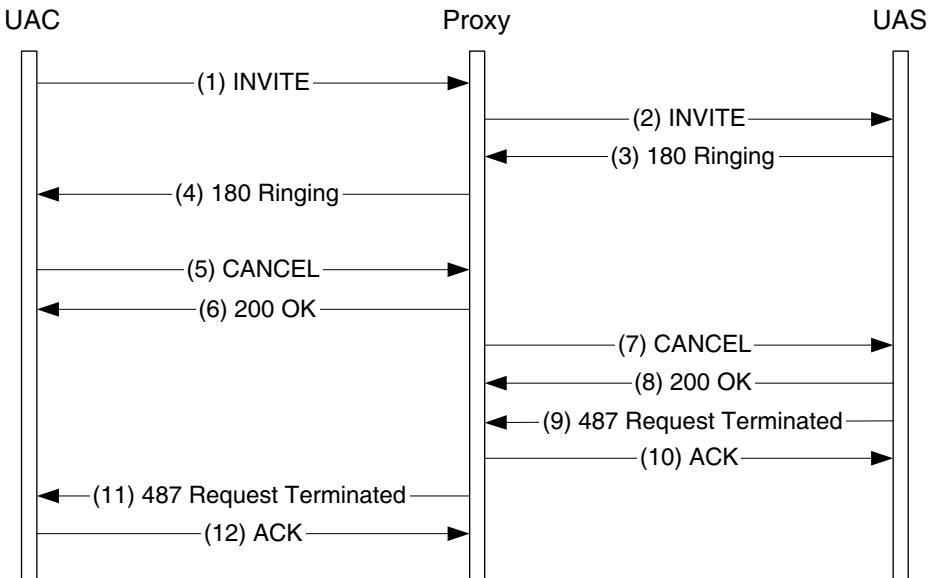


Figure 4.15: CANCEL transaction



Figure 4.16: Alice registers her location

should be relayed to the URI in the Contact header field

`sip:alice@pda.com`

The *Request-URI* of the REGISTER request contains the domain of the registrar (domain.com). The registrar responds with a 200 (OK) response (Figure 4.18) indicating that the transaction was successfully completed.

At a later time, Bob invites Alice to an audio session. Figure 4.19 shows the establishment of the audio session between Bob and Alice through the proxy server at domain.com.

Bob sends an INVITE request (Figure 4.20) using Alice's public URI

`sip:Alice.Smith@domain.com`

as the *Request-URI*. The proxy at domain.com relays the INVITE request (Figure 4.21) to Alice at her current location (her PDA). Alice accepts the invitation sending a 200 (OK) response (Figure 4.22), which is relayed by the proxy to Bob (Figure 4.23).

```

REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 192.0.0.1:5060;branch=z9hG4bKna43f
Max-Forwards: 70
To: <sip:Alice.Smith@domain.com>
From: <sip:Alice@pda.com>;tag=453448
Call-ID: 843528637684230998sdasdsgt
Cseq: 1 REGISTER
Contact: <sip:alice@pda.com>
Expires: 7200
Content-Length: 0

```

Figure 4.17: (1) REGISTER

```

SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.0.1:5060;branch=z9hG4bKna43f
;received=192.0.0.1
To: <sip:Alice.Smith@domain.com>;tag=54262
From: <sip:Alice@pda.com>;tag=453448
Call-ID: 843528637684230998sdasdsgt
Cseq: 1 REGISTER
Contact: <sip:alice@pda.com>;expires=7200
Date: Sat, 25 Mar 2006 17:38:00 GMT
Content-Length: 0

```

Figure 4.18: (2) 200 OK

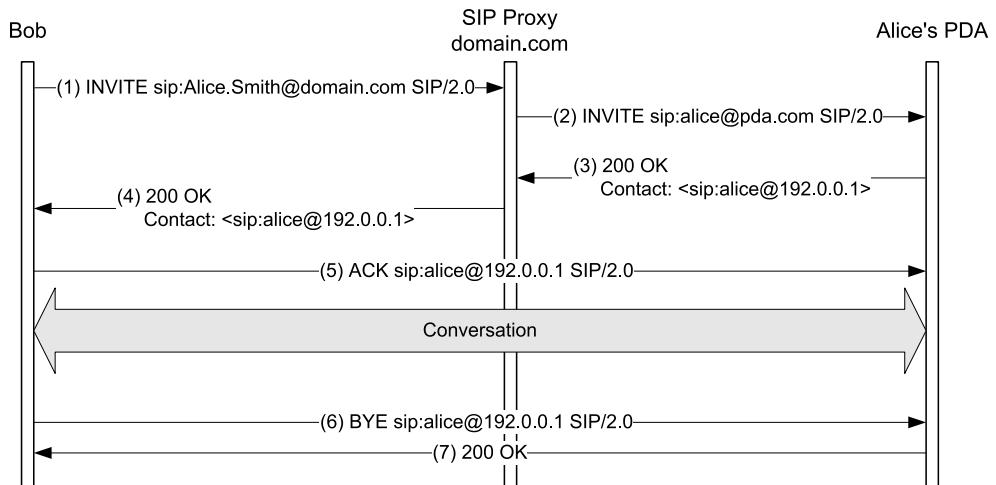


Figure 4.19: Session establishment through a proxy

```

INVITE sip:Alice.Smith@domain.com SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:bob@192.0.100.2>
Content-Type: application/sdp
Content-Length: 138

v=0
o=bob 2890844526 2890844526 IN IP4 ws1.domain2.com
s=-
c=IN IP4 192.0.100.2
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000

```

Figure 4.20: (1) INVITE

Note that Alice has included a Contact header field in her 200 (OK) response. This header field is used by Bob to send subsequent messages to Alice. This way, once the proxy at domain.com has helped Bob locate Alice, Bob and Alice can exchange messages directly between them.

Bob uses the URI in the Contact header field of the 200 (OK) response to send his ACK (Figure 4.24). Now that the session (i.e., an audio stream) is established, Bob and Alice can talk about whatever they want. If, in the middle of the session, they wanted to make any changes to the session (e.g., add video), all they would need to do would be to issue another INVITE request with an updated session description. INVITE requests sent within an ongoing session are usually referred to as re-INVITEs. (UPDATE requests can also be used to modify ongoing sessions. In any case, UPDATEs are used when no interactions with the callee are expected. In this case, we use re-INVITE because the callee is typically prompted before adding video to a session.)

When Bob and Alice finish their conversation, Bob sends a BYE request to Alice (Figure 4.25). Note that, as with the ACK, this request is sent directly to Alice, without the intervention of the proxy. Alice responds with a 200 (OK) response to the BYE request (Figure 4.26).

4.10 SIP Dialogs

In Figure 4.19, Bob and Alice exchange a number of SIP messages in order to establish (and terminate) a session. The exchange of a set of SIP messages between two user agents is referred to as a SIP dialog. In our example the SIP dialog is established by the “INVITE–200 OK” transaction and is terminated by the “BYE–200 OK” transaction. Note, however, that, in addition to INVITE, there are other methods that can create dialogs as well (e.g., SUBSCRIBE). We will study them in later sections.

```

INVITE sip:Alice.Smith@domain.com SIP/2.0
Via: SIP/2.0/UDP p1.domain.com:5060;branch=z9hG4bK543fg
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
    ;received=192.0.100.2
Max-Forwards: 69
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576sl
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:bob@192.0.100.2>
Content-Type: application/sdp
Content-Length: 138

v=0
o=bob 2890844526 2890844526 IN IP4 ws1.domain2.com
s=-
c=IN IP4 192.0.100.2
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000

```

Figure 4.21: (2) INVITE

```

SIP/2.0 200 OK
Via: SIP/2.0/UDP p1.domain.com:5060;branch=z9hG4bK543fg
    ;received=192.1.0.1
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
    ;received=192.0.100.2
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576sl
To: Alice <sip:Alice.Smith@domain.com>;tag=1df345fkj
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:alice@192.0.0.1>
Content-Type: application/sdp
Content-Length: 132

v=0
o=alice 2890844545 2890844545 IN IP4 192.0.0.1
s=-
c=IN IP4 192.0.0.1
t=0 0
m=audio 30000 RTP/AVP 0
a=rtpmap:0 PCMU/8000

```

Figure 4.22: (3) 200 OK

```

SIP/2.0 200 OK
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
      ;received=192.0.100.2
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>;tag=1df345fkj
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:alice@192.0.0.1>
Content-Type: application/sdp
Content-Length: 132

v=0
o=alice 2890844545 2890844545 IN IP4 192.0.0.1
s=-
c=IN IP4 192.0.0.1
t=0 0
m=audio 30000 RTP/AVP 0
a=rtpmap:0 PCMU/8000

```

Figure 4.23: (4) 200 OK

```

ACK sip:alice@192.0.0.1 SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74765
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>;tag=1df345fkj
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 ACK
Contact: <sip:bob@192.0.100.2>
Content-Length: 0

```

Figure 4.24: (5) ACK

```

BYE sip:alice@192.0.0.1 SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK745gh
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>;tag=1df345fkj
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 2 BYE
Content-Length: 0

```

Figure 4.25: (6) BYE

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK745gh
;received=192.0.100.2
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>;tag=1df345fkj
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 2 BYE
Content-Length: 0
```

Figure 4.26: (7) 200 OK

When a SIP dialog is established (e.g., with an INVITE transaction), all the subsequent requests within that dialog follow the same path. In our example, all the requests after the INVITE (the ACK (5) and the BYE (6)) are sent end-to-end between the user agents. However, some proxies choose to remain in the signaling path for subsequent requests within a dialog instead of routing the first INVITE request and stepping down after the 200 (OK) response. Let us study the mechanism used by proxies to stay in the path after the first INVITE request. It consists of three header fields: Record-Route, Route, and Contact.

4.10.1 Record-Route, Route, and Contact Header Fields

Figure 4.27 shows a message flow where the proxy at domain.com remains in the path for all the requests sent within the dialog. The proxy requests to remain in the path by adding a Record-Route header field to the INVITE request (2). The `lr` parameter that appears at the end of the URI indicates that this proxy is RFC 3261-compliant (older proxies used a different routing mechanism).

Alice obtains the Record-Route header field with the proxy's URI in the INVITE request (2), and Bob obtains it in the 200 (OK) response (4). From that point on, both Bob and Alice insert a Route header field in their requests, indicating that the proxy at domain.com needs to be visited. The ACK (5 and 6) is an example of a request with a Route header field sent from Bob to Alice. The BYE (7 and 8) shows that requests in the opposite direction (i.e., from Alice to Bob) use the same Route mechanism.

4.11 Extending SIP

So far, we have focused on describing the core SIP protocol, as defined in RFC 3261 [286]. Now that the main SIP concepts (such as registrars, proxies, redirect servers, forking, SIP encoding, and SIP routing) are clear, it is time to study how SIP is extended.

SIP's extension negotiation mechanism uses three header fields: Supported, Require, and Unsupported. When a SIP dialog is being established the user agent client lists all the names of the extensions it wants to use for that dialog in a Require header field, and all the names of the extensions it supports not listed previously in a Supported header field. The names of the extensions are referred to as *option tags*.

The user agent server inspects the Require header field and, if it does not support any of the extensions listed there, it sends back an error response indicating that the dialog could not be established. This error response contains an Unsupported header field listing the extensions the user agent server did not support.

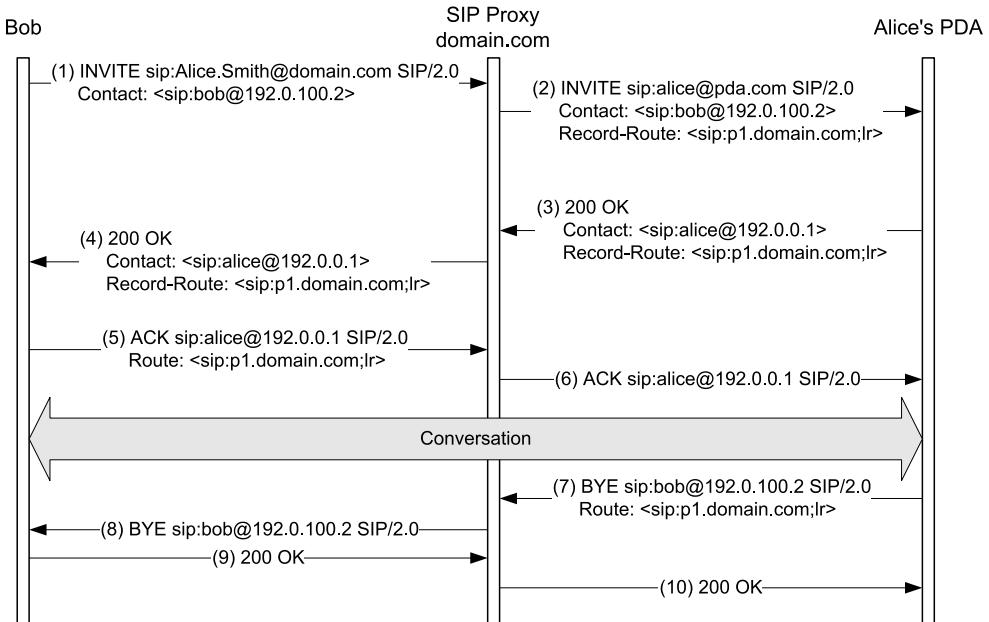


Figure 4.27: Usage of Record-Route, Route, and Contact

If the user agent server supports (and is willing to use) all the required extensions, it should decide whether or not it wants to use any extra extension for this dialog and, if so, it includes the option tag for the extension in the `Require` header field of its response. If this option tag was included in the `Supported` header field of the client, the dialog will be established. Otherwise, the client does not support the extension (or is not willing to use it). In this case the user agent server includes the extension which is required by the server in a `Require` header field of an error response. Such an error response terminates the establishment of the dialog.

Figure 4.28 shows a successful extension negotiation between Bob and Alice. They end up using the extensions whose option tags are `foo1`, `foo2`, and `foo4`.

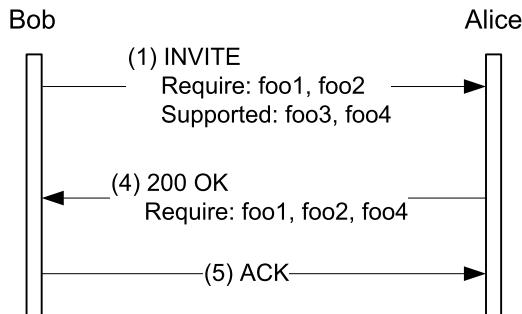


Figure 4.28: Extension negotiation in SIP

4.11.1 New Methods

In addition to option tags, SIP can be extended by defining new methods. We saw in Table 4.3 that there are many SIP methods, but that the core protocol only uses a subset of them. The rest of the methods are defined in SIP extensions.

In a SIP dialog the user agents need to know which methods the other end understands. For this purpose, each of the user agents include an `Allow` header field in their messages listing all the methods it supports. An example of an `Allow` header field is

```
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
```

As we can see, the `Allow` header field lets user agents advertise the methods they support, but it cannot be used to express the fact that a particular method is required for a particular dialog. To provide such functionality, an option tag associated with the method required is defined. This way a user agent can include the option tag in its `Require` header field and force the remote end to apply the extension and, so, to understand the method. The extension for reliable provisional responses described in Section 4.13 is an example of an option tag associated with a method.

4.12 Caller Preferences and User Agent Capabilities

We saw in Section 4.9 how Alice's user agents can register their location in a registrar using a `REGISTER` request. When a proxy server in the same domain as the registrar receives a request for Alice, it relays the request to all the locations registered by Alice's user agents. However, Alice might not want to receive personal calls on her office phone or business calls on her home phone. Moreover, the person calling Alice may not want to talk to her, but only leave a message on her voicemail. The following two SIP extensions make it possible to do what we have just described.

The user agent capabilities extension (defined in RFC 3840 [288]) allows user agents to provide more information about themselves when they register. A user agent can indicate, among other things: the SIP methods it supports; whether or not it supports video, audio, and text communications; whether it is used for business or for personal communications; and whether it is handled by a human or by an automaton (e.g., voicemail). Figure 4.29 shows a `REGISTER` that carries user agent capabilities in its `Contact` header field. In this case the user agent registering supports both audio and video, is fixed (as opposed to mobile), and implements the following SIP methods: `INVITE`, `BYE`, `OPTIONS`, `ACK`, and `CANCEL`.

The user agent capabilities defined originally by the IETF consisted of simple properties, such as support for audio or video or being a mobile or a fixed device. By contrast, the current trend in the industry is to define whole services as single capabilities. For instance, a user agent can inform the registrar that it supports the conferencing service provided by the operator. Supporting such a conferencing service may include supporting a particular floor control protocol and stereophonic audio, but both capabilities are contained in a single user agent capability: the conferencing service capability.

The caller preferences extension (defined in RFC 3841 [287]) allows callers to indicate the type of user agent they want to reach. For instance, a caller may only want to speak to a human (no voicemails) or may want to reach a user agent with video capabilities. The caller preferences are carried in the `Accept-Contact`, `Reject-Contact`, and `Request-Disposition` header fields.

```

REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 192.0.0.1:5060;branch=z9hG4bKna43f
Max-Forwards: 70
To: <sip:Alice.Smith@domain.com>
From: <sip:Alice@pda.com>;tag=453448
Call-ID: 843528637684230@998sdasdsgt
Cseq: 1 REGISTER
Contact: <sip:alice@pda.com>
           ;audio;video;mobility="fixed"
           ;methods="INVITE,BYE,OPTIONS,ACK,CANCEL"
Expires: 7200
Content-Length: 0

```

Figure 4.29: REGISTER carrying UA capabilities

The Accept-Contact header field contains a description of the destination user agents where it is OK to send the request. On the other hand, the Reject-Contact header field contains a description of the destination user agents where it is *not* OK to send the request. The Request-Disposition header field indicates how servers dealing with the request should handle it: whether they should proxy or redirect and whether they should perform sequential or parallel searches for the user.

Figure 4.30 shows an INVITE request that carries caller preferences. The caller that sent this request wants to reach a mobile user agent that implements the INVITE, OPTIONS, BYE, CANCEL, ACK, and MESSAGE methods and that does not support video. In addition, the caller wants proxies to perform parallel searches for the callee.

4.13 Reliability of Provisional Responses

When only the core protocol is used, SIP provisional responses are not transmitted reliably. Only requests and final responses are considered important and, thus, transmitted reliably. However, some applications need to ensure that provisional responses are delivered to the user agent client. For example, a telephony application may find it important to let the caller know whether or not the callee is being alerted. Since SIP transmits this information in a 180 (Ringing) provisional response this telephony application needs to use an extension for reliable provisional responses.

However, before describing such an extension, let us study how requests and final responses are transmitted. SIP is transport-protocol agnostic and, thus, can run over reliable transport protocols, such as TCP (Transport Control Protocol), and over unreliable transport protocols, such as UDP (User Datagram Protocol). The reader can find in the *IEEE Network article* [112] an evaluation of transport protocols for SIP that analyzes the pros and cons of UDP, TCP, and SCTP [308] to be used underneath SIP.

Regardless of the transport protocol, SIP provides an application-layer acknowledgement message that confirms the reception of the original message by the other end. When unreliable transport protocols are used, messages are retransmitted at the application layer until the acknowledge message arrives.

Some SIP messages are retransmitted hop-by-hop (e.g., INVITE requests), while others are retransmitted end-to-end (e.g., 200 (OK) responses for an INVITE request). Figure 4.31

```

INVITE sip:Alice.Smith@domain.com SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Request-Disposition: proxy, parallel
Accept-Contact: *;mobility="mobile"
           ;methods="INVITE,OPTIONS,BYE,CANCEL,ACK,MESSAGE"
Reject-Contact: *;video
Contact: <sip:bob@192.0.100.2>
Content-Type: application/sdp
Content-Length: 138

v=0
o=bob 2890844526 2890844526 IN IP4 ws1.domain2.com
s=-
c=IN IP4 192.0.100.2
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000

```

Figure 4.30: INVITE carrying caller preferences

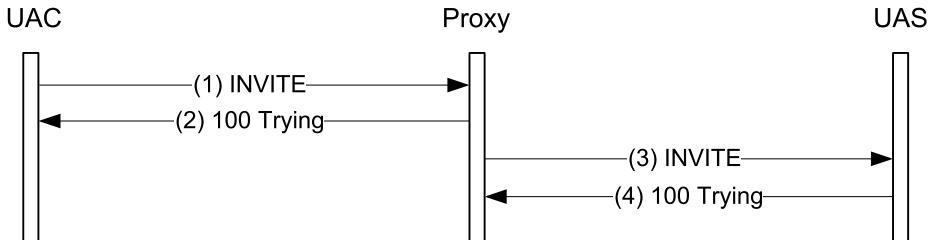


Figure 4.31: Hop-by-hop transmission in SIP

shows a message that is transmitted hop-by-hop. Upon reception of the 100 Trying response the user agent client knows that the next hop (i.e., the proxy) has received the request.

Figure 4.32 shows the previous hop-by-hop message followed by an end-to-end message. In the end-to-end message the user agent server, upon reception of the ACK request, knows that the remote end (i.e., the user agent client, as opposed to the proxy) has received the response. Note that “end-to-end” here refers to the fact that a reliable transmission for the message is provided end-to-end (by the user agents) and not by the proxy servers in the path. Still, all proxy servers in the path handle the message, as shown in Figure 4.32.

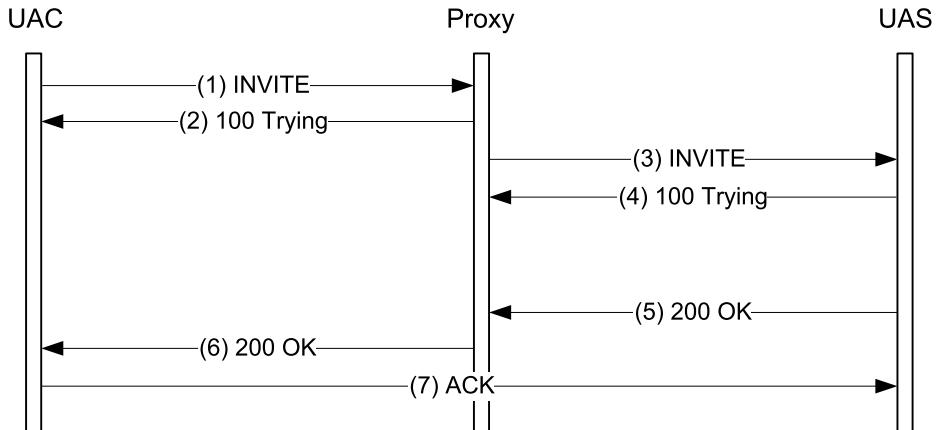


Figure 4.32: End-to-end transmission in SIP

Coming back to the provisional responses (other than 100 Trying), there is no application-layer acknowledgement message for them in core SIP. Therefore, there is an extension (defined in RFC 3262 [284]) whose option tag is 100rel that creates such a message: a PRACK request. Figure 4.33 shows how this works.

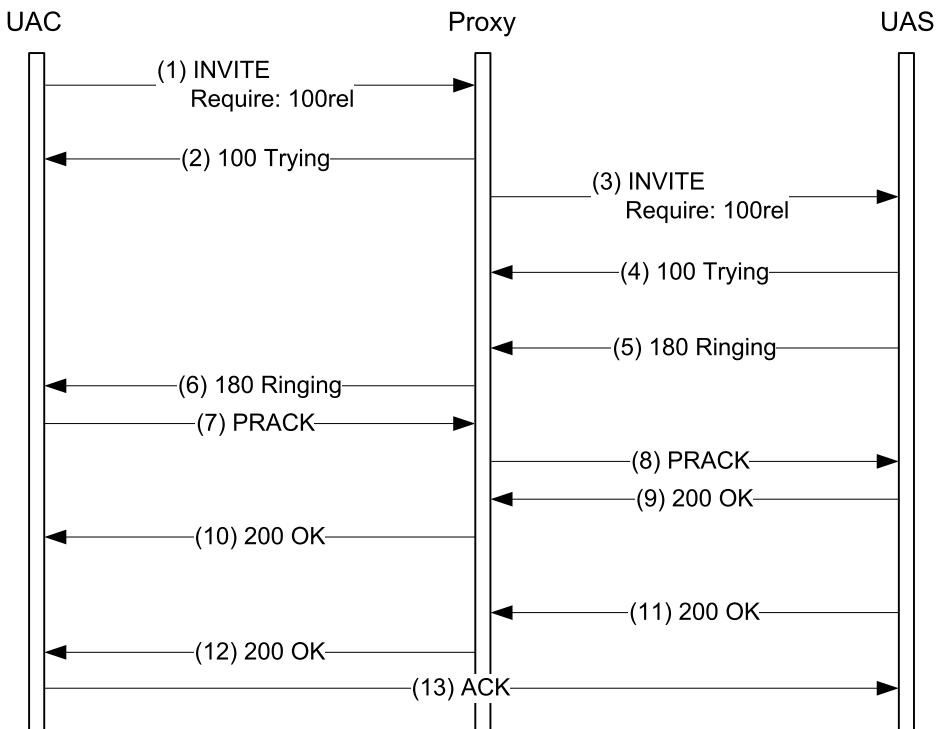


Figure 4.33: Reliable provisional responses and PRACK

The INVITE request in Figure 4.33 contains the `100rel` option tag, which requests the use of reliable provisional responses. When the user agent server sends a 180 (Ringing) provisional response (5), it will retransmit it until the PRACK request arrives. At this point the user agent server knows that the provisional response was received by the user agent client (i.e., end-to-end transmission). It is worth mentioning that the 200 (OK) response (9) for the PRACK request would not be needed to provide reliability. However, in order not to make a special case for this method, it was designed to work like any other method. The only SIP method that does not have a response associated with it is ACK, for historical reasons.

As a last note about transport protocols for SIP, the reader must bear in mind that, since SIP can run over different transports, a single SIP dialog may involve different transport protocols. It is common that, within the same dialog, the transport protocol between a user agent and a proxy is not the same as the one between that proxy and the other user agent or the next proxy. This leads to situations where user agents retransmit messages over a reliable transport protocol to cope with possible losses in the leg that uses an unreliable transport protocol such as UDP.

4.14 Preconditions

In all the examples we have seen so far the only conditions for a session to be established were that the callee accepted the invitation and that the user agent server supported the extensions required by the user agent client. However, some clients require that some preconditions are met before establishing a session. For example, a user may be willing to speak to another user as long as the voice quality is acceptable. In this case, if the network cannot ensure a certain QoS (Quality of Service) for the duration of the session, the caller prefers not to establish the session at all.

The extension that allows user agents to express preconditions is defined in RFC 3312 [103]. This extension, whose option tag is `precondition`, is a mixture of a SIP extension and an SDP extension; it defines a SIP option tag and new SDP attributes. Therefore, this extension can only be used with sessions described using SDP. Other session description formats may define their own extension for preconditions in the future.

When a user agent receives an offer with preconditions, it does not alert the user until those preconditions are met. The preconditions are encoded, as mentioned earlier, in the SDP body. There are two types of precondition: access preconditions and end-to-end preconditions. Figure 4.34 shows an SDP with access preconditions. The user agent that generated this session description is requesting (`a=des:qos` means desired QoS) QoS in both directions (`sendrecv`) in both accesses: its local access (local) and the remote access (remote). The `a=curr:qos` lines indicate that, currently, there is no QoS in either of the accesses.

```
m=audio 20000 RTP/AVP 0
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
```

Figure 4.34: Access preconditions

Figure 4.35 shows an SDP with end-to-end preconditions. The user agent that generated this session description is requesting optional end-to-end (e2e) QoS in both directions (sendrecv).

```
m=audio 20000 RTP/AVP 0
a=curr:qos e2e none
a=des:qos optional e2e sendrecv
```

Figure 4.35: End-to-end preconditions

When mandatory preconditions appear in a session description the callee is alerted only when the current QoS conditions are equal to or better than the desired conditions. Therefore, the INVITE request is typically answered with a 183 (Session Progress) provisional response that does not imply either alerting (180 (Ringing) does) or acceptance of the session (200 (OK) does).

In order for them to know when all the preconditions are met, both user agents need to exchange session descriptions. For example, when the user agent that generated the session description in Figure 4.34 obtains QoS in its access network, it will update the a=curr:qos line from none to sendrecv and send a session description as shown in Figure 4.36. This session description is sent using the UPDATE method, defined in RFC 3311 [267], as shown in Figure 4.37. Using this method, both user agents keep each other up to date on the status of the preconditions.

```
m=audio 20000 RTP/AVP 0
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
```

Figure 4.36: Updated current QoS conditions

In Figure 4.37, once the first session descriptions have been exchanged, both user agents perform QoS reservations in their respective accesses. When the user agent client finishes its reservation, it sends an UPDATE request (5) informing the user agent server (a=curr:qos local sendrecv). When the user agent server finishes its own QoS reservations, all the preconditions are met and the callee is alerted (7).

4.15 Event Notification

So far, we have seen how to use SIP to establish sessions. Now we will see how to use SIP to obtain the status of a given resource and track changes in that status. For example, at a given moment, the state of Alice's presence is "online". When she logs off from her computer her presence status changes to "offline". In this example the resource is Alice and the status information is her presence information.

RFC 3265 [264] defines a framework for event notification in SIP. It uses the SUBSCRIBE and NOTIFY methods. The entity interested in the status information of a resource *subscribes* to that information. The entity that keeps track of the resource state will send a NOTIFY request with the current status information of the resource and a new NOTIFY

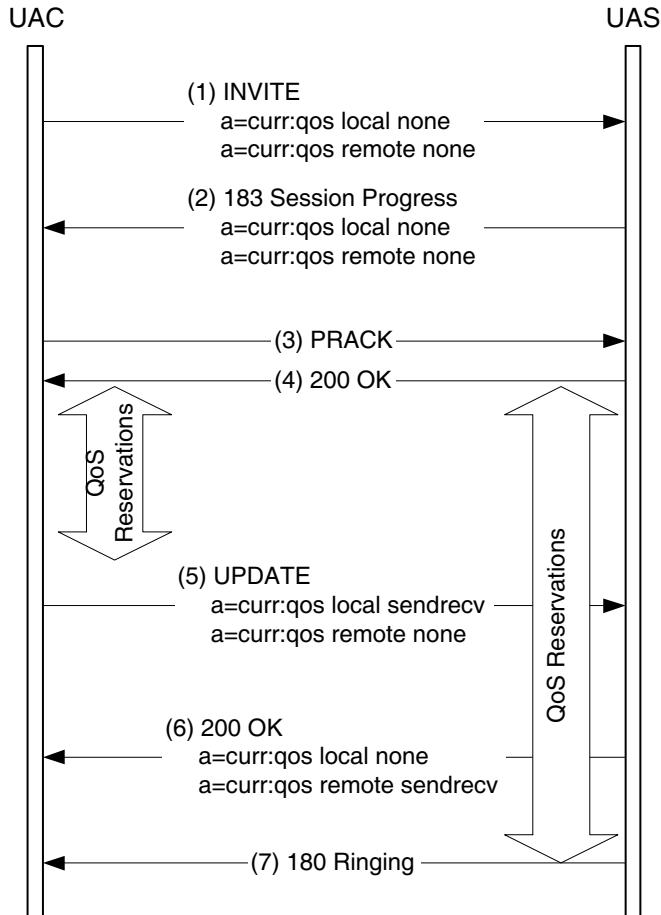


Figure 4.37: The UPDATE method

request every time the status changes. The type of status information is defined by an Event header field. Specifications defining new values for the Event header field are called *event packages*.

The event notification framework defines two new roles in SIP: the *subscriber* and the *notifier*. A subscriber is a SIP UA that sends a SUBSCRIBE request for a particular event. A subscriber gets NOTIFY requests containing state information related to the subscribed event. A notifier is a SIP UA that receives SUBSCRIBE requests for a particular event and generates a NOTIFY request containing the state information related to the subscribed event. A notifier keeps a subscription state for each of the subscribers.

Figure 4.38 shows an example where Alice, acting in the role of a subscriber, subscribes to her voicemail. This application is described in RFC 3842 [212]. The voicemail server is acting as a notifier. In this case, the status information she is interested in is the number of messages that have arrived at the voicemail. This corresponds to an Event header field of value `message-summary`.

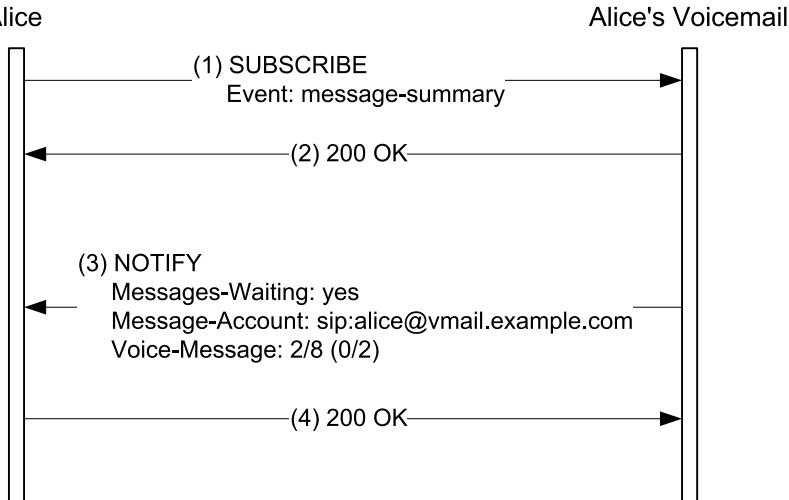


Figure 4.38: Voicemail status information

Note that the 200 (OK) response (2) to the SUBSCRIBE request only indicates that the SUBSCRIBE transaction has been successful. That is, the subscription has been accepted by the voicemail. The information about the resource always arrives in a NOTIFY transaction. The NOTIFY request (3) in Figure 4.38 shows the body of the NOTIFY. It indicates that Alice's voicemail has two new messages and eight old messages, of which none of the new and two of the old messages are urgent (figures enclosed in parentheses).

If, before the subscription expires, Alice's voicemail receives a new message, it will send a new NOTIFY request to Alice informing her about the new arrival.

4.15.1 High Notification Rates

The event notification framework offers a powerful tool that allows a subscriber to be informed about changes in the state of a resource. However, in some situations the amount of information that the subscriber has to process might be large. Imagine, for instance, a subscriber to the presence information of a user who is driving on a highway.

Our subscriber gets frequent updates, because the user's geographical position changes rapidly. In any case, our subscriber might not be interested in receiving accurate real-time information. The consequence of having very detailed and accurate information is that the bandwidth needed to transport all of this information increases and so does the power needed to process it.

The event framework provides a mechanism, called event throttling, to limit the rate at which notifications are sent to a subscriber. This mechanism is helpful for devices with low processing-power capabilities, limited battery life, or low-bandwidth accesses. Event throttling is used to build such services as presence (we describe the presence service in Chapter 19).

4.15.1.1 Event Throttling

The event framework allows the notifier of the actual event package to setup a policy for the notification rate. While most packages use a default policy, there are situations where subscribers want to communicate to the notifier the amount of information they are willing to receive.

The event-throttling mechanism allows a subscriber to an event package to indicate the minimum period of time between two consecutive notifications. So, if the state information changes rapidly, the notifier holds those notifications until the throttling timer has expired, at which point the notifier sends a single notification to the subscriber.

With this mechanism the watcher does not have a real-time view of the subscribe-state information, but it has approximate information. This approximation is good enough for a number of applications.

4.15.1.2 Conditional Event Notification

There are certain cases where a subscriber loses connectivity for a short period of time. When the connectivity is restored, it subscribes again to the event package notification it was previously subscribed to. The notifier immediately sends a NOTIFY request that contains the latest event state, which in most cases has not changed, and is the same that the subscriber had before loss of connectivity. A mechanism that allows the suppression of the body of the NOTIFY request, or even the whole NOTIFY request (including its body), when the subscriber already has the state information is desired for this type of scenario. This is what the conditional event notification, specified in the Internet-Draft “An extension to Session Initialization protocol (SIP) events for conditional event notification” [220], provides.

The mechanism provides a mechanism to create versions of the body (state) of NOTIFY requests. These versions are in the format of entity-tags. When the subscriber creates a SUBSCRIBE request, if it already has the state pertaining to that event package, it includes a Suppress-If-Match header field that contains the value of the last entity-tag associated to the state. If the state has not changed, then the entity-tag of the Suppress-If-Match header is exactly the same one as the state that, otherwise, would be included in the body of the NOTIFY request. Since both entity-tags are equal, the notifier answers the SUBSCRIBE request with a new 204 (No Notification) response and does not issue the classical NOTIFY request.

However, if the entity-tag included in the Suppress-If-Match header field of the SUBSCRIBE request does not match the entity-tag of the current state, the notifier acts in the classical way: it answers the SUBSCRIBE request with a 200 (OK) response and an immediate NOTIFY request that contains the state. This NOTIFY request contains a SIP-ETag header field that contains the value of the entity-tag associated to the body of the NOTIFY request.

The decision on whether to suppress the body of the NOTIFY request or the complete NOTIFY request is in the notifier’s hands. If something has to be reported in the headers of the NOTIFY request, such as a change in the subscription state (Subscription-State header field), only the body is suppressed. Otherwise, if there are no changes to the headers of the NOTIFY request, the complete NOTIFY request is suppressed.

Figure 4.39 depicts a flow where the body of the NOTIFY request is suppressed. A subscriber sends a SUBSCRIBE request (1) and gets a 202 (Accepted) response followed by a NOTIFY request (3) that contains a body conveying the event state, and a SIP-ETag that

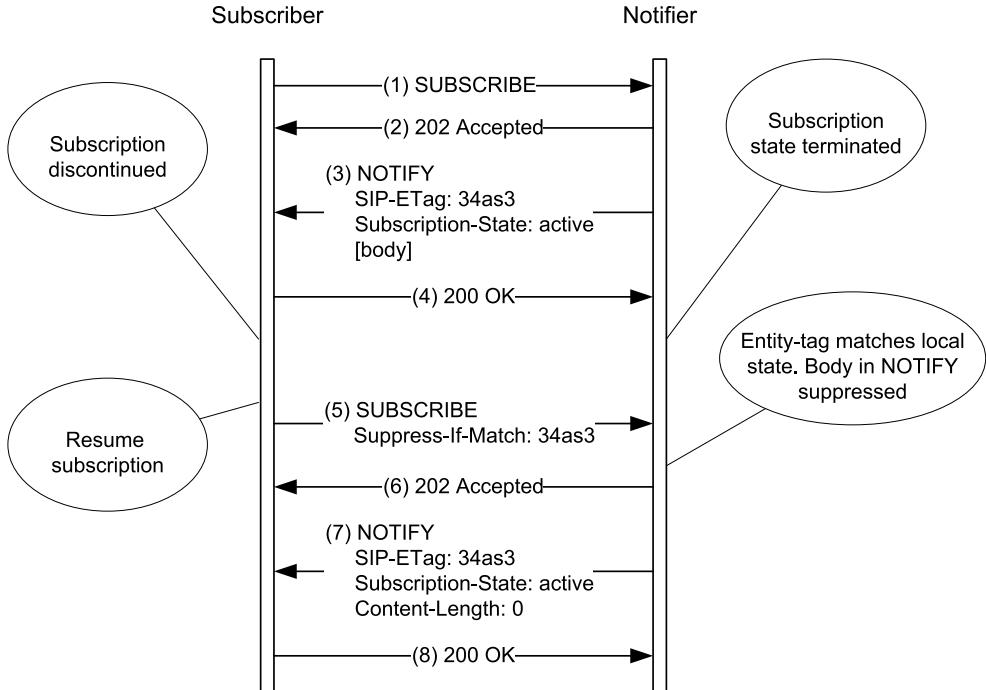


Figure 4.39: Suppression of the body of a NOTIFY request

contains the value of the entity-tag associated to that version of the event state. The subscriber stores the entity-tag along with the event state itself.

At some point in time, the subscriber, for whatever reason, loses the subscription, which is detected on the notifier side, which sets the subscription state to terminated. When the subscriber recovers from the failure, it needs to re-subscribe to the event state changes, so it sends a new SUBSCRIBE request (5) that contains a Suppress-If-Match header field that lists the value of the entity-tag associated with the stored event state. The notifier finds a match with the value of the local entity-tag of the event state. Since the subscription state has changed from terminated to active, this needs to be communicated in the headers of the NOTIFY request, so only the body of the NOTIFY request (7) can be suppressed. The notifier sends this NOTIFY request (7) that contains the Subscription-State and SIP-ETag header fields, but not a body.

4.16 Signaling Compression

As we explained in Section 4.15.1, users with low-bandwidth access need to minimize the amount of data they send and receive if they want an acceptable user experience. Otherwise, performing a simple operation would take so long that users would lose their patience and stop using the service. However, SIP is not an efficient protocol with respect to message size. Its textual encoding makes SIP messages grow dramatically as soon as several extensions are used at the same time. Fortunately, we have signaling compression to help us.

RFC 3486 [98] describes how to signal that a SIP message needs to be compressed. It defines the `comp=sigcomp` parameter to be used in URIs (if a request is to be compressed) or in `Via` entries (if a response is to be compressed). When this parameter appears in a message the message is compressed using the mechanism described in RFC 3320 [258], which is known as SigComp (signaling compression). Figure 4.40 shows how to use the `comp=sigcomp` parameter to signal SIP traffic compression between a user agent and a proxy. The `Route` header field indicates that the requests (INVITE and ACK) need to be compressed, and the `Via` header field indicates that the response (200 OK) needs to be compressed.

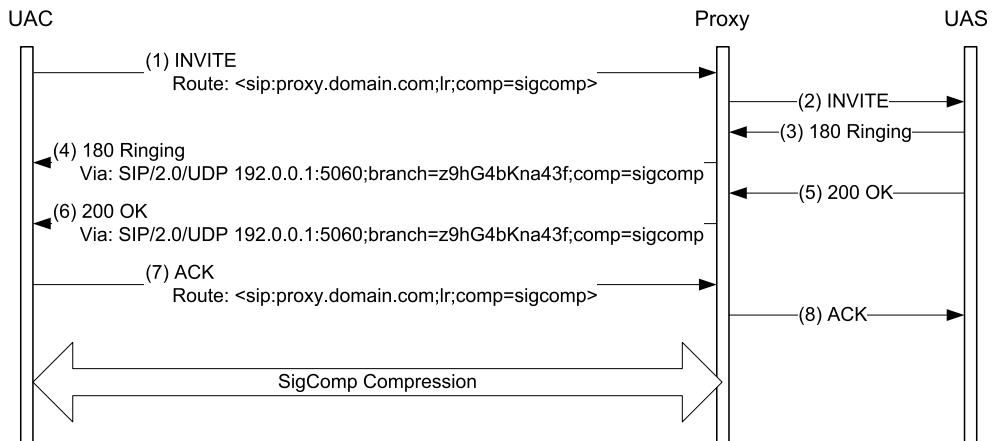


Figure 4.40: SigComp compression between a UA and a proxy

The SigComp specification, RFC 3320 [258], defines a Universal Decompressor Virtual Machine (UDVM) intended to run decompression algorithms. The entity compressing the message provides the entity receiving it with the algorithm to be used to decompress the message. This way the sender is free to choose any compression algorithm to compress its messages.

Compression algorithms substitute long expressions that are used often in the message with shorter pointers to those expressions. The compressor builds a dictionary that maps the long expressions to short pointers and sends this dictionary to the decompressor, as shown in Figure 4.41.

Figure 4.41 shows how a very simple compression algorithm works. This algorithm substitutes words with letters (note that advanced compression algorithms can substitute much longer expressions by pointers). We can see that every message that is compressed is sent together with a dictionary.

4.16.1 *SigComp Extended Operations*

However, we can see that dictionaries corresponding to different messages have many words in common. If we could use a single dictionary this redundancy would disappear. RFC 3321 [162], on SigComp extended operations, defines a way to use a shared dictionary. Figure 4.42 shows how SigComp extended operations work. The first message is compressed in the same way as in Figure 4.41, but subsequent messages reference previous terms in the shared dictionary. We can see that the compression ratio achieved using extended operations is better than that achieved with regular SigComp compression.

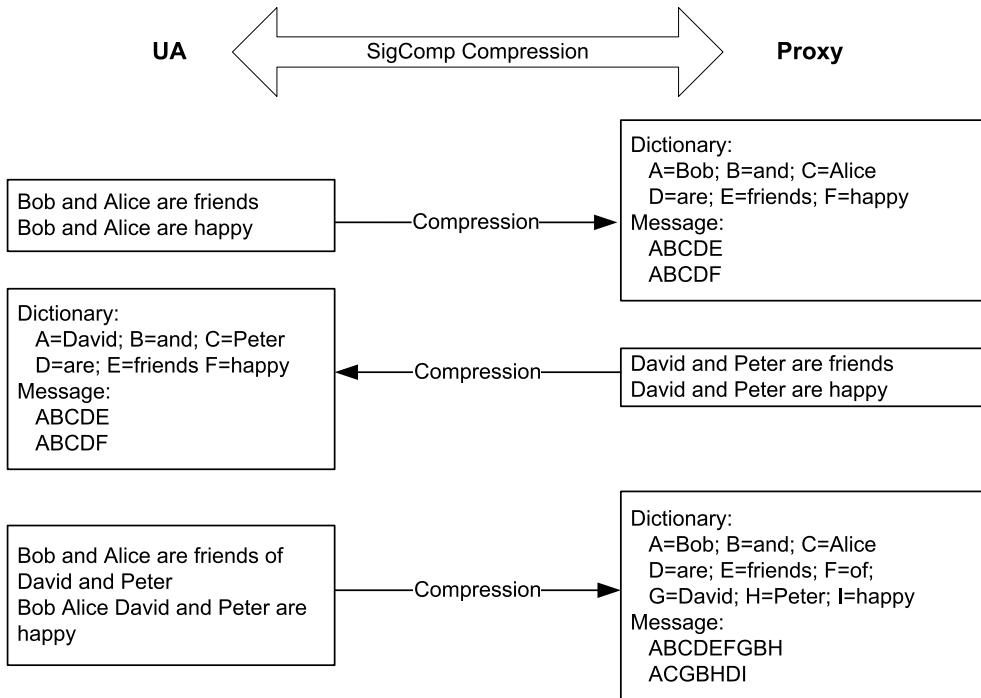


Figure 4.41: Regular SigComp compression

4.16.2 Static SIP/SDP Dictionary

As we have seen in Figure 4.42, the dictionary is built up as the compressors get more input: the longer the input the better the dictionary and the more efficient the compression. However, this way of building up the decompression dictionary does not suit SIP. The first message a user agent sends to establish a session is typically an INVITE request. At this point the user is waiting for the other party to answer in a reasonable time. If sending the INVITE request takes too long the user will have a poor user experience. Unfortunately, this INVITE request cannot take advantage of any previously built dictionary and so it will not be compressed as much as subsequent messages will (e.g., BYE). Therefore, this way of building up the dictionary gives us slow INVITE and fast BYE requests. This means that the user has a bad user experience.

To improve the compression efficiency for the first message, SIP provides a static SIP/SDP dictionary (defined in RFC 3485 [151]), which is supported to be present in every implementation that supports SigComp. This dictionary contains the SIP and SDP terms that are used most often. Figure 4.43 shows how the dictionary is used. The compression efficiency improves significantly compared with Figure 4.42.

4.17 Content Indirection

Compressing SIP using the techniques described in Section 4.16 helps to reduce the size of the messages. However, sometimes SIP message bodies are just too large, even after compression.

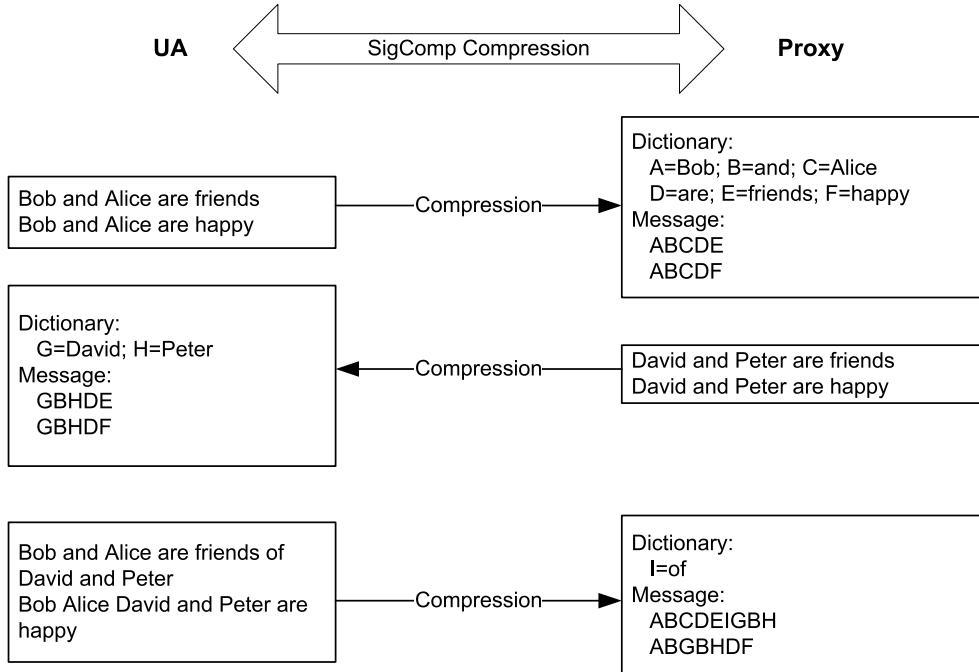


Figure 4.42: SigComp extended operations

Large messages have two important disadvantages: service behavior is too slow on low-bandwidth links and, more importantly, messages get fragmented when transported over UDP. We will describe the problems associated with IP fragmentation and then look at a SIP extension that resolves this issue.

While TCP provides transport-layer fragmentation, UDP does not. This means that if an application-layer message (e.g., a SIP message) is larger than the maximum size that link layers in the path can handle, the message will be fragmented at the IP layer.

If one of the fragments of this message gets lost, the sender needs to retransmit the whole message, which is clearly quite an inefficient way to perform packet loss recovery. Moreover, some port-based firewalls and NATs (Network Address Translators) cannot handle fragments. This is because only the first fragment carries the port numbers of the datagram carrying the message. When a firewall or a NAT receives a fragment which is not the first one, it cannot find the port number of the datagram and simply discards the packet.

So, in some situations (e.g., a UA behind a NAT that cannot handle fragments), it might be impossible to transmit large SIP messages. In this situation a SIP extension called *content indirection* (specified in RFC 4483 [94]) is our friend.

Content indirection allows us to replace a MIME body part with an external reference, which is typically an HTTP URI. The destination UA fetches the contents of that MIME body part using the references contained in the SIP message.

Figures 4.44 and 4.45 show an INVITE request carrying a session description in the body and as an external reference, respectively. In the second case, Bob will fetch Alice's SDP using the HTTP URI provided in the INVITE request.

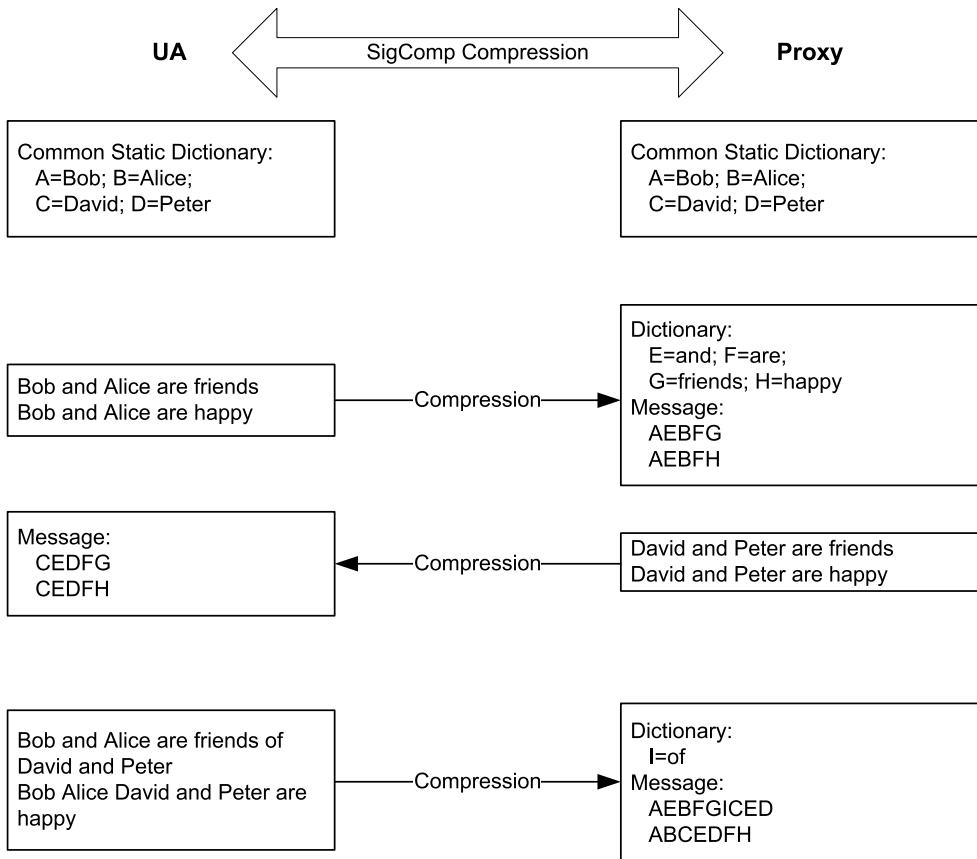


Figure 4.43: SigComp with SIP/SDP static dictionary

Content indirection is especially useful for optional body parts. For example, if Alice uses content indirection to include her photo in her INVITEs, the callees can choose whether or not they want to fetch it. A callee on a low-bandwidth link can probably live without seeing Alice's photo, while another callee using a high-speed access will most likely enjoy seeing it. In addition, SIP proxies in the path will not need to route large messages, which is important for proxies handling a large number of users.

4.18 The REFER Method

The REFER method (specified in RFC 3515 [306]) is used to request a user agent to contact a resource. This resource is identified by a URI, which may or may not be a SIP URI.

The most common usage of REFER is call transfer. For example, Bob is in a call (i.e., an audio session) with Alice. Bob is using his mobile SIP phone because he is walking on the street. When Bob arrives at his office, he wants to transfer the session with Alice to his fixed SIP phone. Bob sends a REFER request to Alice's user agent requesting it to contact Bob's fixed phone's SIP URI. Once the transfer is finished, Bob continues with this call on this fixed phone.

```

INVITE sip:Alice.Smith@domain.com SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:bob@192.0.100.2>
Content-Type: application/sdp
Content-Length: 138

v=0
o=bob 2890844526 2890844526 IN IP4 ws1.domain2.com
s=-
c=IN IP4 192.0.100.2
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000

```

Figure 4.44: SDP carried in the message body

```

INVITE sip:Alice.Smith@domain.com SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:bob@192.0.100.2>
Content-Type: message/external-body;
    ACCESS-TYPE=URL;
    URL="http://www.domain.com/mysdp";
    size=138
Content-Length: 31

Content-Type: application/sdp

```

Figure 4.45: SDP provided as a external reference

Bob would use the same REFER mechanism to transfer his call with Alice to another person (Alice may want to talk to Bob's boss after her conversation with Bob). In this case, Bob would like to make sure that Alice has been able to contact Bob's boss before hanging up. The REFER mechanism includes an implicit subscription to the result of the operation initiated by the recipient of the REFER. That is, Alice will inform Bob (with a NOTIFY request) as to whether or not she managed to contact Bob's boss successfully.

Bob can also use REFER to request Alice's user agent to contact a non-SIP URI. For example, if Bob wants Alice to have a look at his new personal web page, he sends a REFER

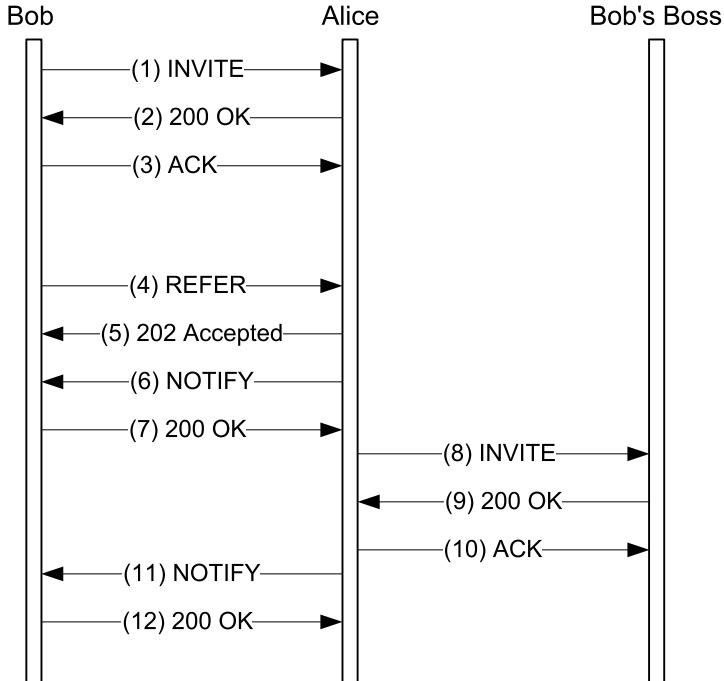


Figure 4.46: The REFER method

request to Alice's user agent requesting it to contact his personal web page's HTTP URI. In this case, Alice will inform Bob when she downloads the web page.

Figure 4.46 shows the message flow involved in the example described earlier where Alice is requested to contact Bob's boss. The message flow shows how the REFER request creates an implicit subscription to the result of the operation (in this case, contacting Bob's boss). This subscription has the same properties as subscriptions created with a SUBSCRIBE request (see Section 4.15). That is, Bob receives NOTIFY requests carrying information about the result of the transaction initiated by Alice toward Bob's boss.

On receiving a REFER request, a user agent always generates a NOTIFY request immediately (6). At a later point, when the operation requested in the REFER request has been attempted, further NOTIFY requests are sent. The initial NOTIFY request is needed to handle forking scenarios.

Even if a REFER request forks and arrives at several user agent servers, the user agent client that sent the REFER request only receives one final response (routing rules for proxies ensure this property for non-INVITE transactions). Such a user agent client discovers all the user agent servers that received the REFER request when it receives the NOTIFY requests.

Although the implicit subscription associated with REFER is useful in many situations, some applications using the REFER method do not need the implicit subscription. It is possible to suppress the implicit subscription of a particular REFER request by using a REFER extension (specified in RFC 4488 [207]).

4.19 Globally Routable User Agent URIs (GRUU)

In most SIP terminals several applications are implemented and coexist. For example, a SIP terminal can include video telephony, instant messaging, presence, image sharing, file transfer, and many other SIP applications. Furthermore, SIP allows all these applications to be identified with the same SIP URI, such as `sip:alice@example.com`. Incoming sessions are received by the SIP stack, which, based on the contents of the SIP request, dispatches the message to the appropriate application. In some cases a mechanism to identify a particular instance of an application or a given endpoint device is required.

On the other hand, we saw earlier that it is common for the same user to be registered from different terminals simultaneously. The user is identified by a SIP URI, such as `sip:alice@example.com`, but the URI is not specific to any device. If Alice's proxy receives an incoming SIP request, it will most likely fork it to all the terminals from which Alice is registered.

There are some cases, though, when it is required to identify a single SIP application among those running simultaneously in the terminal, and a single terminal among those from where the user is registered. This is the case of a call transfer service. Assume that Alice is engaged in a call with Bob. Alice wants to transfer the call to Charlie, so that at the end of the operation, a session is setup between Bob and Charlie. According to what we saw in Section 4.18, Alice sends to Charlie a REFER request which contains a `Refer-To` header including Bob's SIP URI. Then Charlie can send an INVITE request to Bob's SIP URI. The problem arises if several SIP applications are running in Bob's terminal or if Bob is registered in several terminals simultaneously, because then all Bob's terminals will ring (which is not the desired effect), or the INVITE request might be dispatched to an incorrect SIP application in Bob's terminal.

To dismiss this problem, the concept of a Globally Routable User Agent URI (GRUU) (specified in the Internet-Draft “Obtaining and using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)” [275]) is introduced. A GRUU is a SIP URI that has the following properties:

- it is logically linked to a SIP Address of Record (AoR);
- it contains enough information to route a SIP request to a given SIP instance running in a specific SIP terminal;
- it is globally reachable, or, at least, it has the same reachability as the SIP AoR it is linked to.

SIP GRUUs are characterized by the presence of a “`gr`” parameter in the SIP URI. The “`gr`” parameter may have a value or may be empty.

GRUUs pose a threat to anonymity. The fundamental concept behind anonymity is to avoid disclosing private information, such as the user's SIP AoR, IP address of the terminal, location information, etc. Since GRUUs are logically connected to SIP AoRs and the specific SIP instance at a specific user's terminal, they get properties opposite to those desired for anonymity. Because of this, there are two types of GRUU: those which expose the linked SIP AoR and those that do not. If privacy of the user's identity is desired, the latter are used.

Assume Alice has a SIP AoR of

`sip:alice@example.com`

An example of a SIP GRUU that exposes the linked SIP AoR is

```
sip:alice@example.com;gr=49dnsasdka
```

Her SIP AoR is exposed because the GRUU is simply formed by appending the “gr” parameter to a value that identifies the SIP instance.

An example of a SIP GRUU that does not expose her linked SIP AoR is

```
sip:9dxkasjd5djm@example.com;gr
```

Her SIP AoR is not disclosed. The GRUU is identified as such because of the presence of the “gr” parameter, although in this case, the SIP instance is identified in the username part of the SIP URI.

There are two more concepts related to GRUUs. On the one hand, a SIP instance running in a UA needs to *obtain* a GRUU, a step that needs to be done in coordination with its SIP registrar. On the other hand, the SIP UA needs to announce its GRUU during a session, so that in the event the correspondent invokes a service such as call transfer, he already knows the GRUU to use to direct the call transfer.

There are three ways for the UA to obtain a GRUU:

- during the registration process;
- self-made at the UA;
- pre-configured or via a similar out-of-band mechanism.

Perhaps the most common case is for UAs to obtain a GRUU during the registration process. The process is schematically illustrated in Figure 4.47. A user agent sends a REGISTER request that, among other things, contains a Supported header field that adds the value gruu. The REGISTER request also contains a Contact with the contact URI, and a sip.instance media feature tag, whose value is set to a Uniform Resource Name (URN) that uniquely identifies the SIP instance at that UA. Typically the URN included in the sip.instance media feature tag is a Universally Unique Identifier (UUID), specified in RFC 4122 [205]. Figure 4.48 shows an example of a Contact header field used in a registration.

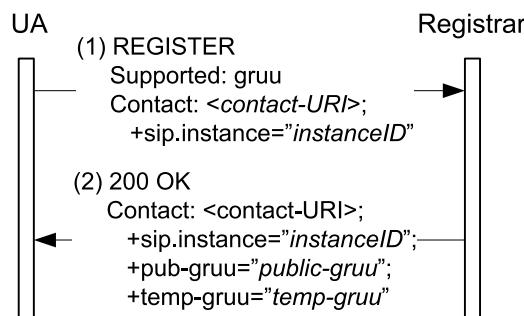


Figure 4.47: A pair of GRUUs obtained at registration time

```
Contact: <sip:alice1@pc.example.com>;  
;+sip.instance=<urn:uuid:a56d41bf-e0c9-4c80-abd3-82c40b973806>"
```

Figure 4.48: Contact header field in the SIP REGISTER request (1)

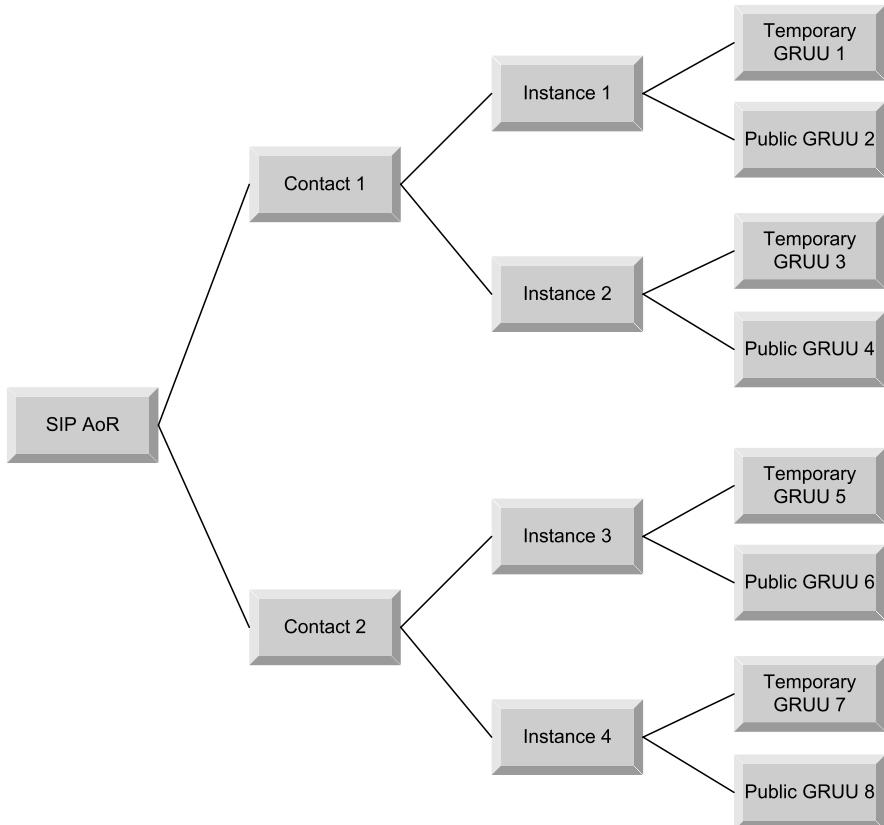


Figure 4.49: Data structure at the registrar

Then the registrar generates a public GRUU and a temporary GRUU, whose values are bound to the SIP AoR and instance ID. So, if the instance ID changes (such as when the same user is registering from a different terminal), then the registrar will allocate a different pair of GRUUs. Supporting GRUU in the registrar requires that the registrar keep track of all the combinations of SIP AoRs, Contact values, instance values, and public and temporary GRUUs. Figure 4.49 shows the logical relations between all these data in a registrar. Fortunately, if GRUUs are smartly generated, the storage requirement for the registrar can be minimized. Appendix A of the Internet-Draft “Obtaining and using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)” [275] contains a discussion on the topic.

Once a pair of temporary and public GRUUs have been generated, the registrar then includes both in the Contact header field of the 200 (OK) response to the REGISTER request. An example is given in Figure 4.50. The public GRUU is built by the concatenation

```
Contact: <sip:alice1@pc.example.com>;
pub-gruu="sip:alice@example.com;gr=urn:uuid:
a56d41bf-e0c9-4c80-abd3-82c40b973806";
temp-gruu="sip.tgruu.2098dakds2dammksdf9hko@example.com;gr";
+sip.instance=<urn:uuid:a56d41bf-e0c9-4c80-abd3-82c40b973806>;
expires=3600
```

Figure 4.50: GRUUs included in the Contact header field of a 200 (OK) response (2)

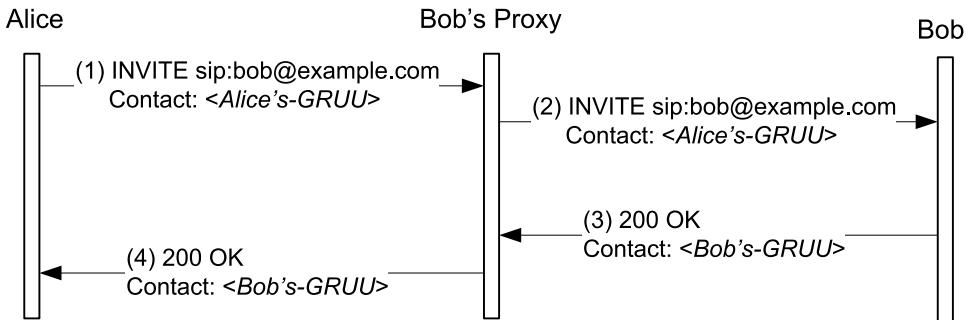


Figure 4.51: Session setup with GRUUs

of Alice's SIP Aor and the gr parameter with a value that equals the UUID URN received in the SIP instance media feature tag. The temporary GRUU does not reveal Alice's SIP Aor, although it is routable to Alice's domain. Public GRUUs are used when users do not require privacy for their identity, because the GRUU can track down the real SIP Aor. Since the SIP Aor cannot be derived from a temporary GRUU, temporary GRUUs are used when users require privacy for their identity.

Now that the user agent has acquired a couple of GRUUs, let us analyze how they are put into operation. Figure 4.51 depicts the most relevant aspects of a session setup with GRUUs. The figure assumes that both user agents supports GRUU, although this might not always be the case. To advertise a GRUU, the caller replaces the value of the regular Contact header field with a GRUU, typically learnt at registration time, in the INVITE request (1, 2). If the caller requests privacy of its identity, it includes a temporary GRUU in the Contact header field. Otherwise it includes a public GRUU. The callee eventually replies with a 200 (OK) response (3, 4) whose Contact header field contains one of the callee's GRUUs rather than a regular contact value.

Assume now that Bob is registered from two or more terminals. Assume also that Alice wants to transfer the call to Charlie. A simplified call flow is depicted in Figure 4.52. When the session is established, Alice sends a REFER request (1) to Charlie. The request contains a Refer-To header field that contains Bob's GRUU, which was learnt through the Contact header field at session setup. The GRUU has the property of pointing to exactly the same instance and terminal that Bob used to established the session with Alice. Charlie replies with a 202 (Accepted) response (2) and sends an INVITE request (3) to the value included in the Refer-To header field of the REFER request (1), i.e., to Bob's GRUU. The INVITE request (3) resolves to Bob's proxy server, which receives the request. Then Bob's proxy analyzes the Request-URI field and determines that it is a valid GRUU (e.g., non-expired)

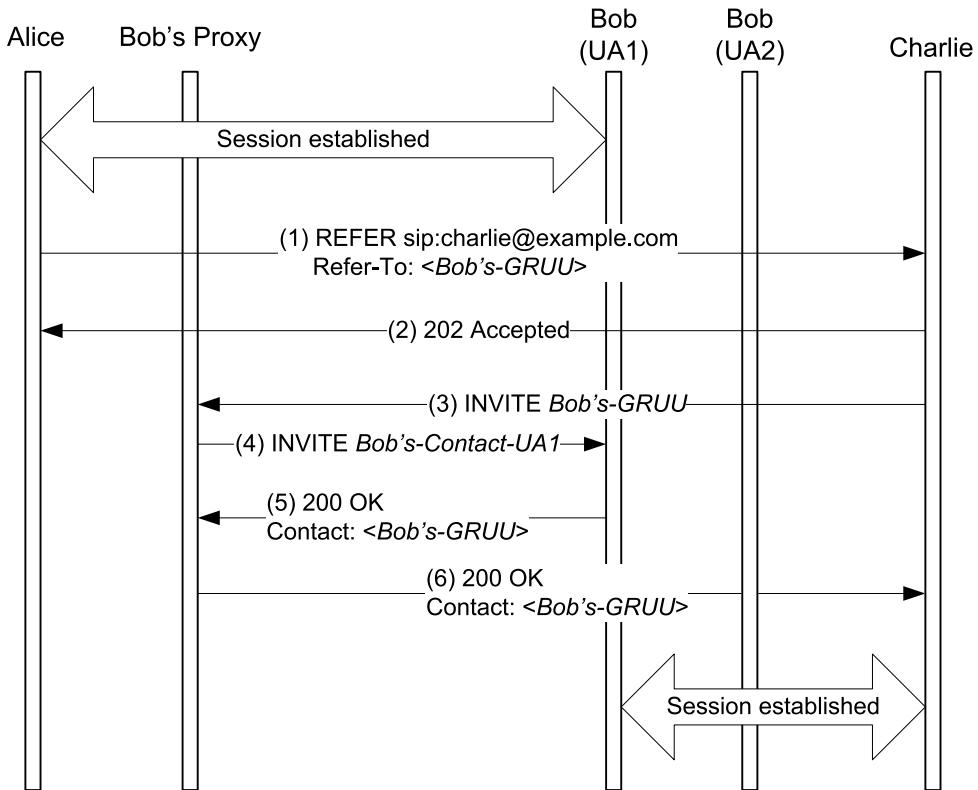


Figure 4.52: Simplified call flow for a transfer operation with GRUUs

bound to a given Bob's instance at UA1. The proxy does regular re-writing of the Request-URI, e.g., replaces the GRUU in the incoming INVITE request (3) with the corresponding contact registered at registration and bound to this GRUU, i.e., UA1. The GRUU provides the SIP instance indication at UA1. Without the GRUU, the proxy should have forked the INVITE request (3) to all registered Bob's contacts, making both UA1 and UA2 ring, which is not the desired effect. So, Bob's proxy forwards the INVITE request (4) to Bob's UA1 only. Note that this INVITE request (4) does not contain a GRUU in the Request-URI, but just Bob's contact URI learnt during registration. Eventually Bob accepts the new INVITE request (4), which also has protocol indications to replace the existing session with Alice. Bob answers with a 200 (OK) response (5), which is forwarded to Charlie. The session is now established between Bob and Charlie. The initial session between Alice and Bob is ended with normal procedures (not shown in Figure 4.52).

4.20 NAT Traversal

Many users of the current public Internet are behind NATs (Network Address Translators). NATs were designed to minimize the number of public IP addresses (i.e., IP addresses that are globally routable on the public Internet) needed by a site by assigning private addresses to the hosts within the site. A NAT with (at least) a public IP address handles the traffic between

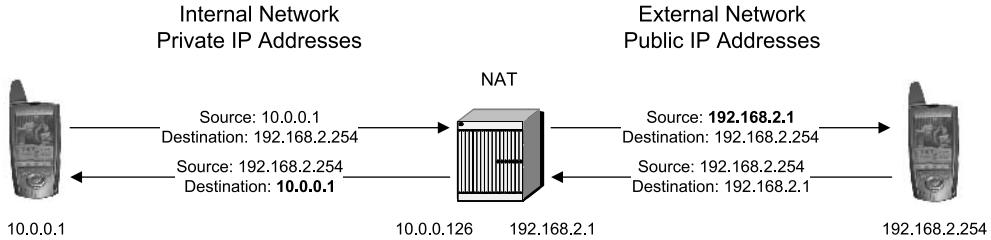


Figure 4.53: NAT operation

hosts within the site and hosts outside it (e.g., in the public Internet). Figure 4.53 illustrates the operation of a NAT. The NAT has two interfaces: an internal interface with a private IP address (i.e., 10.0.0.126) and an external interface with a public IP address (i.e., 192.168.2.1). When an internal host (whose IP address is 10.0.0.1) sends traffic to an external one (whose IP address is 192.168.2.254), the NAT performs the needed address translations (in bold in the figure) so that internal private addresses are not exposed on the external interface.

4.20.1 Types of NAT

In addition to NATs that translate IP addresses, such as the one in Figure 4.53, most NATs also translate port numbers. This way, NATs can make better use of their external address space. NATs can be classified by how they perform address and port mappings (see RFC 4787 [80]). The types of mapping found on NATs are endpoint-independent, address-dependent, and address- and port-dependent.

Figure 4.54 shows a NAT performing endpoint-independent mapping. Packets from the same source address and port are translated to the same public address and port regardless of their destination. In the figure, the packets sent to 192.168.2.254:80 and to 192.168.2.62:80 both have the same source address and port (10.0.0.1:20000). Consequently, the NAT maps the source address and port of both packets to the same public address and port (192.168.2.1:25000).

Figure 4.55 shows a NAT performing address-dependent mapping. Packets from the same source address and port are translated to the same public address and port as long as the packets are sent to the same host (i.e., to the same destination IP address). In the figure, the packets sent to 192.168.2.254:80 and to 192.168.2.254:8080 both have the same source address and port (10.0.0.1:20000). Consequently, the NAT maps the source address and port of both packets to the same public address and port (192.168.2.1:25000). The packet sent to 192.168.2.62:80 has the same source address and port (10.0.0.1:20000) as the previous packets but is not addressed to the same host. Therefore, the NAT maps its source address and port differently (192.168.2.1:30000).

Figure 4.56 shows a NAT performing address- and port-dependent mapping. Packets from the same source address and port are translated to the same public address and port only if they are sent to the same IP address and port. In the figure, the packets sent to 192.168.2.254:80 and to 192.168.2.254:8080 both have the same source address and port (10.0.0.1:20000). However, since they are sent to different ports, even if their destination addresses are the same, they are mapped differently (192.168.2.1:25000 and 192.168.2.1:27000).

NATs can also be classified by how they filter incoming traffic. The types of filtering behaviors found on NATs are endpoint-independent, address-dependent, and address- and

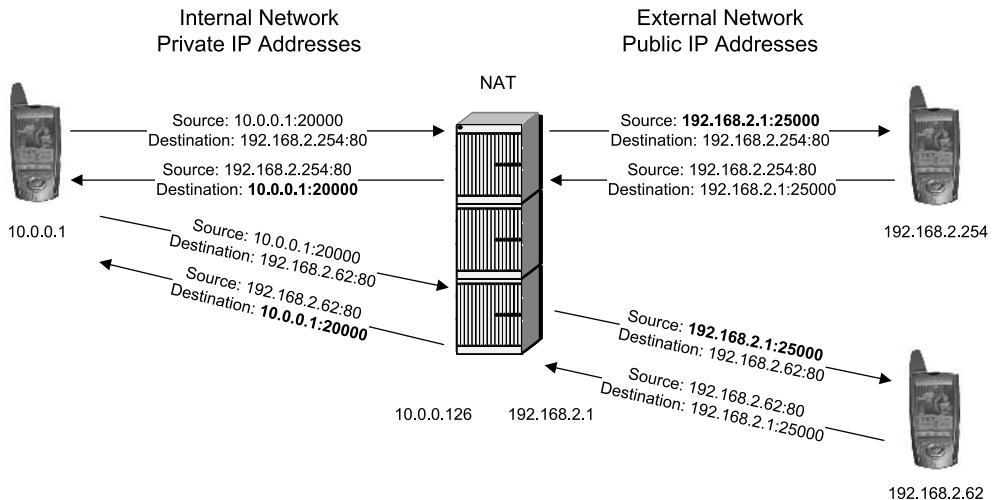


Figure 4.54: Endpoint-independent mapping

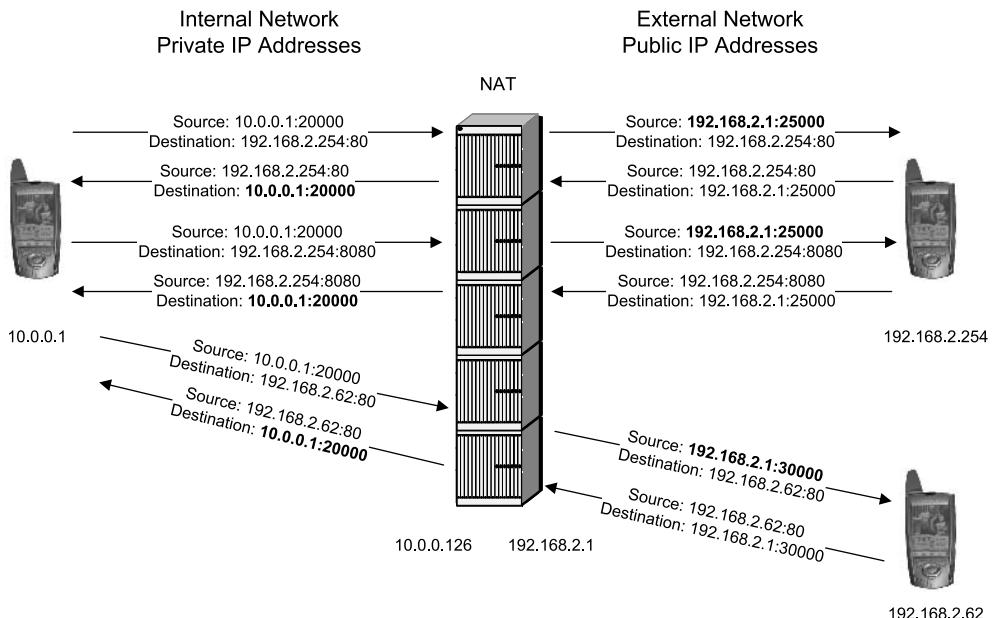
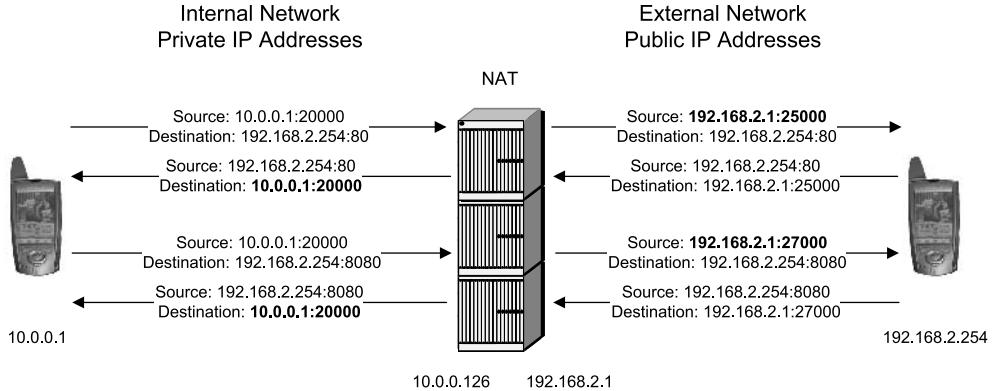
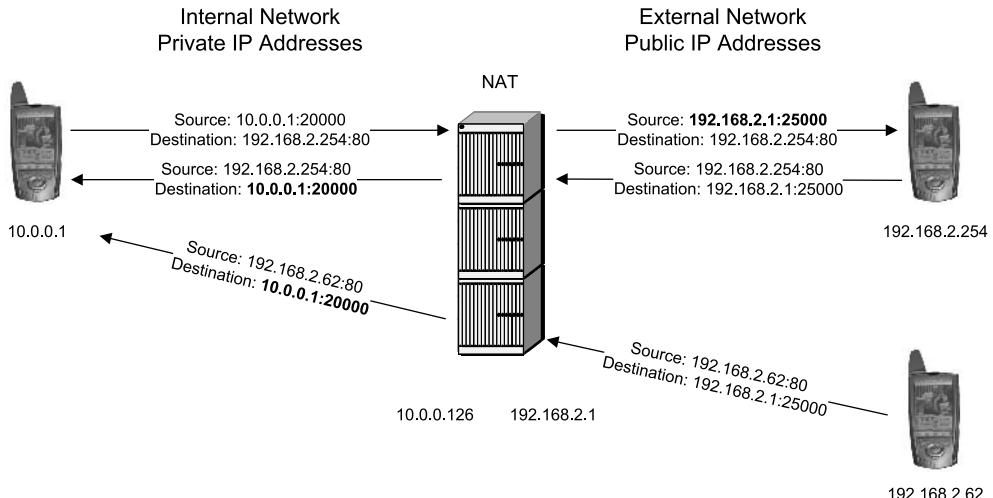


Figure 4.55: Address-dependent mapping

**Figure 4.56:** Address- and port-dependent mapping**Figure 4.57:** Endpoint-independent filtering

port-dependent. Note that a NAT's filtering behavior is independent from its mapping behavior. That is, a NAT can follow different policies for mapping and for filtering.

Figure 4.57 shows a NAT performing endpoint-independent filtering. When the NAT allocates an external public IP address and port for a binding, it accepts packets from any address to the allocated address and port. In the figure, the NAT allocates 192.168.2.1:25000 for the binding. The NAT accepts packets from both 192.168.2.254:80 and 192.168.2.62:80.

Figure 4.58 shows a NAT performing address-dependent filtering. When the NAT allocates an external public IP address and port for a binding, it accepts packets only from the host (i.e., IP address) for which the binding was created. In the figure, the NAT allocates 192.168.2.1:25000 for the binding. The NAT accepts packets from both 192.168.2.254:80 and 192.168.2.254:8080 but not from 192.168.2.62:80.

Figure 4.59 shows a NAT performing address and port dependent filtering. When the NAT allocates an external public IP address and port for a binding, it accepts packets only from

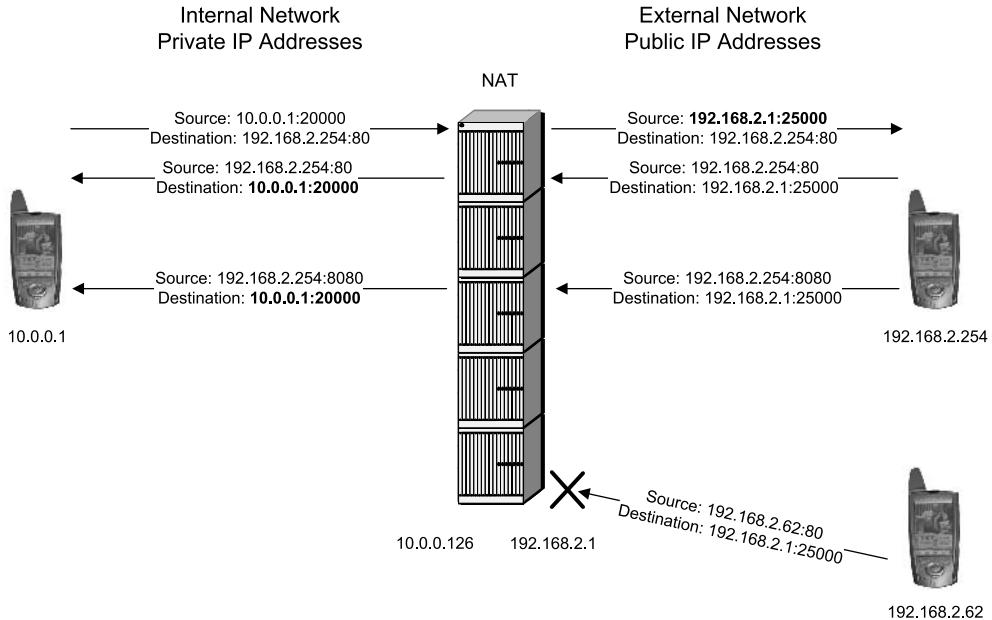


Figure 4.58: Address-dependent filtering

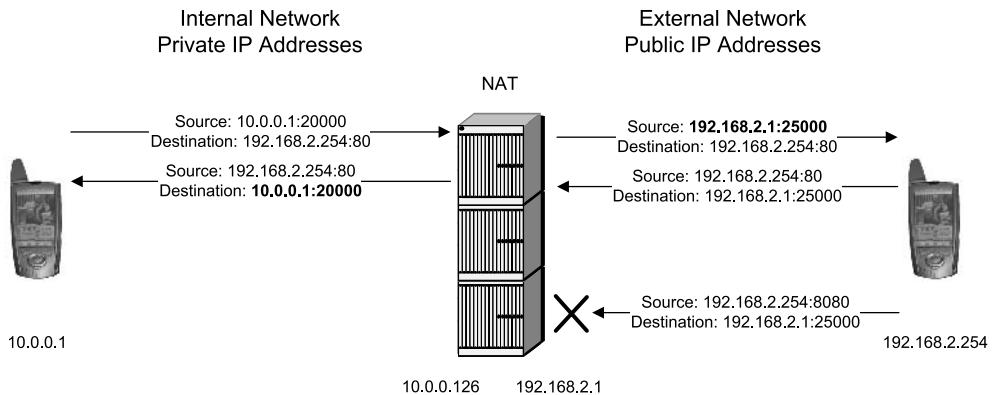


Figure 4.59: Address- and port-dependent filtering

the IP address and port for which the binding was created. In the figure, the NAT allocates 192.168.2.1:25000 for the binding. The NAT accepts packets from 192.168.2.254:80 but not from 192.168.2.254:8080.

In addition to their mapping and filtering behaviors, NATs can be classified in many different ways. However, just by looking at these two classifications, it is clear that the difficulty of traversing one or more NATs will depend on the types of those NATs. For example, a NAT with endpoint-independent mapping and filtering behaviors will be easier to traverse than one with address- and port-dependent behavior. Initial attempts to develop NAT traversal mechanisms worked well in certain scenarios, but had shortcomings in other

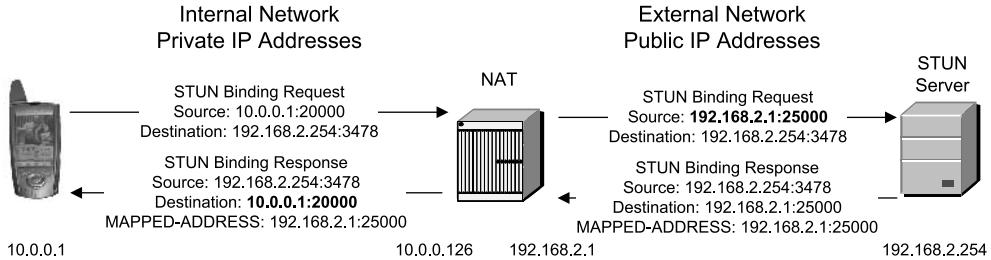


Figure 4.60: STUN operation

scenarios that involved different types of NAT. STUN and TURN, which are described in the following sections (Sections 4.20.2 and 4.20.3), are two such mechanisms. ICE, which is described in Section 4.20.4, combines both STUN and TURN in a methodology that, although quite complex, works in most scenarios.

4.20.2 STUN

STUN (specified in the Internet-Draft “Session traversal utilities for NAT (STUN)” [281]) is a protocol that allows a STUN client to discover its IP address as seen by a STUN server. Effectively, by deploying the STUN server on the public Internet, the STUN client can discover the IP address the NAT has allocated for the client. Figure 4.60 shows how STUN works. The STUN client sends a STUN binding request to the STUN server. On receiving the request, the STUN server gets the source IP address and port of the request and places it in a MAPPED-ADDRESS attribute. The server then returns a STUN binding response that carries the attribute. On receiving the response, the STUN client knows that there are one or more NATs between itself and the STUN server and that the outermost NAT has allocated 192.168.2.1:25000 for the client’s binding.

Some NATs inspect the payload of the packets whose IP addresses they translate. If they find the addresses being translated in the payload, they also translate them there. This means that if such a NAT handles a STUN binding response, it translates the address inside the MAPPED-ADDRESS attribute. In the example in Figure 4.60, the NAT would translate 192.168.2.1:25000 into 10.0.0.1:20000 (i.e., the same mapping as it performed in the IP header of the packet). To overcome this issue, STUN servers also include an XOR-MAPPED-ADDRESS attribute in their responses. This attribute contains the same address as the MAPPED-ADDRESS attribute but in an obfuscated form, so that NATs are not able to identify the address in the attribute. Effectively, STUN clients use the XOR-MAPPED-ADDRESS attribute. The MAPPED-ADDRESS attribute is still used to achieve backwards compatibility with older implementations that pre-date the definition of the XOR-MAPPED-ADDRESS attribute.

4.20.3 TURN

TURN (specified in the Internet-Draft “Traversal using relays around NAT (TURN): Relay extensions to session traversal utilities for NAT (STUN)” [280]) is an extension to STUN that allows TURN clients to allocate an IP address and port at a TURN relay. The TURN relay, unsurprisingly, relays traffic received on the allocated IP address and port to the TURN client.

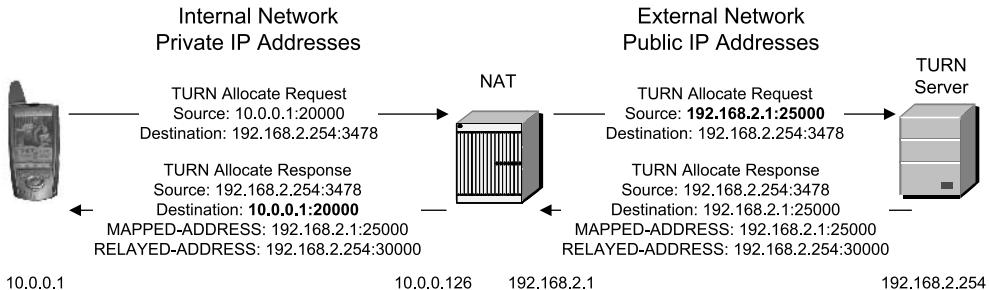


Figure 4.61: TURN operation

Figure 4.61 shows how TURN works. The TURN client sends a TURN allocate request to the TURN relay. On receiving the request, the TURN relay gets the source IP address and port of the request and places it in a MAPPED-ADDRESS attribute, as if it was a STUN server. In addition, it allocates an address for the client and places it in a RELAYED-ADDRESS attribute. The TURN relay then returns a TURN allocate response that carries both attributes. On receiving the response, the TURN client knows that there are one or more NATs between itself and the TURN server and that the outermost NAT has allocated 192.168.2.1:25000 for the client's binding. In addition, the client knows that traffic received by the relay on 192.168.2.254:30000 will be relayed to the client.

TURN relays encapsulate the traffic to be relayed to the client so that the client knows where the traffic comes from. In addition, the client can send traffic to remote endpoints through the TURN relay.

4.20.4 ICE

ICE (specified in the Internet-Draft “Interactive Connectivity Establishment (ICE): A protocol for NAT traversal for offer/answer protocols” [274]) is a NAT traversal methodology that combines different mechanisms such as STUN and TURN. The ICE methodology can be used by any protocol that implements offer/answer exchanges. Of course, SIP implements the offer/answer model and thus can use ICE.

ICE is used between two endpoints (e.g., two SIP User Agents) willing to establish a connection between them. First, the offerer gathers the candidate addresses that it can use to communicate with the remote endpoint. These candidate addresses are transport addresses. That is, each candidate address consists of an IP address and a port. The candidate addresses the offerer gathers can include local addresses, addresses obtained using STUN, addresses obtained using TURN, and addresses obtained using any other mechanism, such as a VPN (Virtual Private Network) protocol. Addresses obtained using STUN are called server-reflexive addresses and addresses obtained using TURN are called relayed addresses. Once the offerer gathers all its candidate addresses, it orders them by priority and sends them in its offer to the answerer.

On receiving the offer, the answerer also gathers the candidate addresses it can use to communicate with the offerer. The answerer can also gather different types of address during this process. Once the answerer has gathered all its candidate addresses, it orders them by priority and sends them in its answer to the offerer.

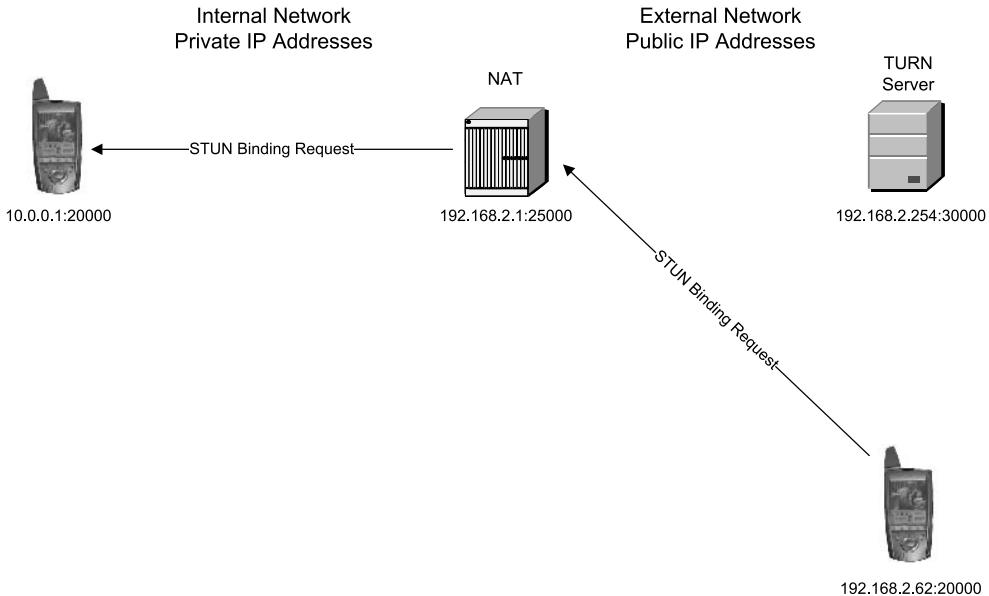


Figure 4.62: Address- and port-dependent filtering

Endpoints typically assign the highest priority to local addresses, then to server-reflexive addresses, and then to relayed addresses. This way of assigning priorities attempts to first minimize the number of relays (e.g., TURN relays) in the path, and second minimize the number of NATs in the path. A local address is preferred over a server-reflexive address that involves traversing a NAT; a server-reflexive address is preferred over a relayed address that involves traversing a relay (which is typically heavier than traversing a NAT). Nevertheless, user agents are free to use their own priority schemes. For example, an endpoint may assign a higher priority to an address where it can receive traffic for free rather than to an address where receiving traffic is expensive.

Once both user agents have ordered candidate address lists, they pair them to form candidate address pairs. Then, the endpoints use end-to-end STUN binding requests to test which candidate address pairs work and which do not. For a candidate address pair to work, both user agents should be able to send a STUN binding request to the remote address and receive the corresponding STUN binding response.

Once the endpoints start testing candidate address pairs by sending STUN probes, they can follow different strategies to select the address pair to be used for their communication from all the candidate address pairs that work. A common strategy is to test candidate address pairs sequentially in order of priority, and use for their communication the first pair that works.

For example, let us assume that the endpoint in Figure 4.61 wants to communicate with an endpoint that has a public IP address as shown in Figure 4.62. The candidate addresses of the endpoint behind a NAT are 10.0.0.1:20000 (a local address), 192.168.2.1:25000 (a server-reflexive address), and 192.168.2.254:30000 (a relayed address). The candidate address of the other endpoint is 192.168.2.62:20000 (a local address).

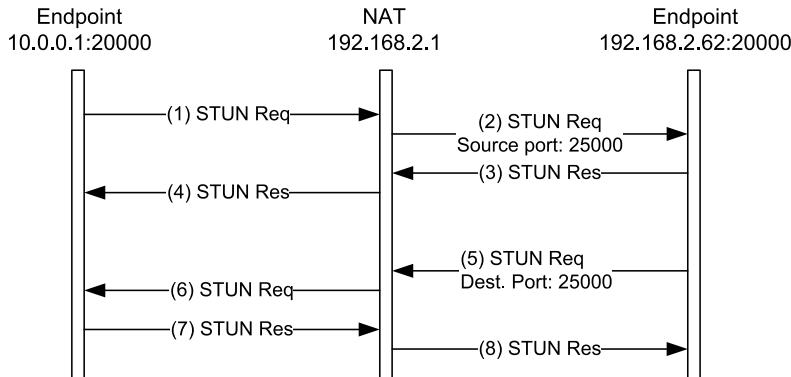


Figure 4.63: Endpoint-independent mapping

The endpoint with the public IP address starts testing the addresses it got for the remote endpoint. The local address 10.0.0.1:20000 is unreachable because it belongs to a private network. The relayed address works all the time but has the lowest priority because it involves the use of a relay. Whether or not the server-reflexive address works will depend on the type of NAT in our example. If the NAT implements endpoint-independent filtering, it will accept the incoming STUN binding request (which was sent to 192.168.2.1:25000) and deliver it to the endpoint, as shown in Figure 4.62. However, if the NAT implements address-dependent filtering, the NAT will drop the STUN binding request. For this STUN binding request to be accepted by a NAT with such a filtering policy, the endpoint in the private network needs to first send a STUN binding request itself as part of its own address testing procedure.

Figure 4.63 shows what happens when the endpoint in the private network sends first its STUN binding request and the NAT implements endpoint-independent mapping. The NAT maps this STUN binding request (1) in the same way as it mapped the previous STUN binding request sent to the TURN server in Figure 4.61 (i.e., 192.168.2.1:25000). The endpoint with the public IP address sends its own STUN binding request (5) to the candidate address it got in the offer/answer exchange (192.168.2.1:25000), and everything works.

Figure 4.64 shows what happens when the endpoint in the private network sends first its STUN binding request and the NAT implements address-dependent mapping. The NAT maps this STUN binding request (1) in a different way from how it mapped the previous STUN binding request sent to the TURN server in Figure 4.61. This time, the NAT maps it to 192.168.2.1:27000. When the endpoint with the public IP address sends its own STUN binding request (5) to the candidate address it got in the offer/answer exchange (192.168.2.1:25000), the NAT drops the request because it implements address-dependent filtering. However, when the endpoint with the public address received the previous STUN binding request (2), it learned a new address that it had not received in the offer/answer exchange (i.e., 192.168.2.1:27000). When the endpoint tries it, everything works. This last scenario illustrates the fact that the end-to-end STUN connectivity checks can help endpoints discover new addresses that were unknown when the initial offer/answer exchange was performed.

This example, which is based on a fairly simple network architecture (i.e., one endpoint behind a NAT and one with a public IP address), illustrates how many different scenarios endpoints trying to communicate can face, depending on the types of NAT deployed between

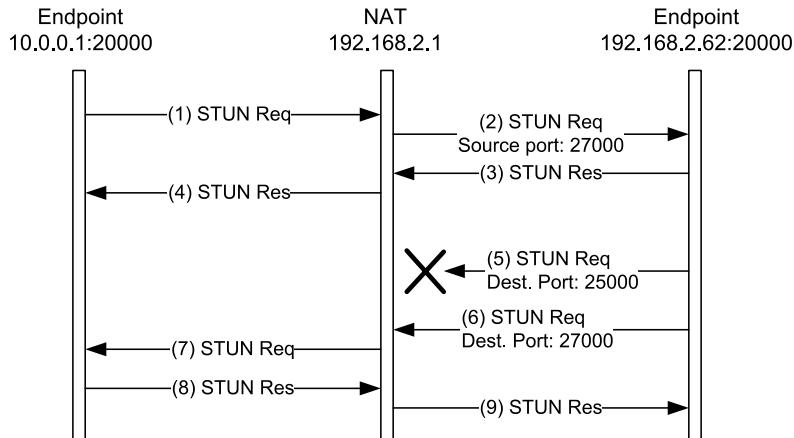


Figure 4.64: Address-dependent mapping

them. ICE makes no assumptions about the types of NAT there may be between two particular endpoints; instead, it tries all different possible candidate address pairs until it finds one that works.

Chapter 5

Session Control in the IMS

We saw in Section 4.1 how SIP is used in a public Internet environment. We have also explored the core SIP functionality and a few important extensions that SIP User Agents may support. Each implementation of SIP is free to implement the options or SIP extensions that the particular application requires.

3GPP is one of those particular applications of SIP where SIP is used in a wireless environment. In the case of 3GPP, SIP is used over an underlying packet network that defines a number of constraints. The result of the evaluation of SIP in wireless environments led to the definition of a set of requirements that accommodates SIP in 3GPP networks. The implementation of solutions to these wireless requirements led 3GPP to mandate the use of a number of options and extensions to SIP and other protocols. We can consider 3GPP's function as creating a profile of utilization of SIP and other protocols in the IP Multimedia Subsystem. The 3GPP SIP profile utilization for IMS is specified in 3GPP TS 24.229 [37]. We call it a *profile* because there are no differences with respect to the usage of SIP on the public Internet. However, 3GPP has mandated the implementation of a number of extensions and options in both the IMS network nodes and IMS terminals. This section focuses on describing how SIP is used in the IMS as well as highlighting differences in the utilization of SIP with respect to the public Internet.

When 3GPP began the work on session control for the IMS, SIP was chosen as the protocol to control sessions. At that time the IETF (Internet Engineering Task Force) was working on a revision of SIP that led to the migration and extension of the protocol from RFC 2543 [161] to RFC 3261 [286] and other RFCs. Previously, the performance of SIP in wireless environments had never been evaluated.

Wireless environments have a number of strict requirements for session control protocols like SIP. These requirements range from extra security requirements to the capability of providing the same services no matter whether the mobile station is located in the home network or roaming to a visited network. The IETF analyzed these requirements and took most of them into consideration. This led to the design of a number of SIP solutions that were either included in the core SIP specification in RFC 3261 [286] or documented as separate extensions to SIP. We will analyze these extensions when we delve deeper into session control in the IMS.

5.1 Prerequisites for Operation in the IMS

Before an IMS terminal starts any IMS-related operation there are a number of prerequisites that have to be met. Figure 5.1 shows a high-level view of the required prerequisites.

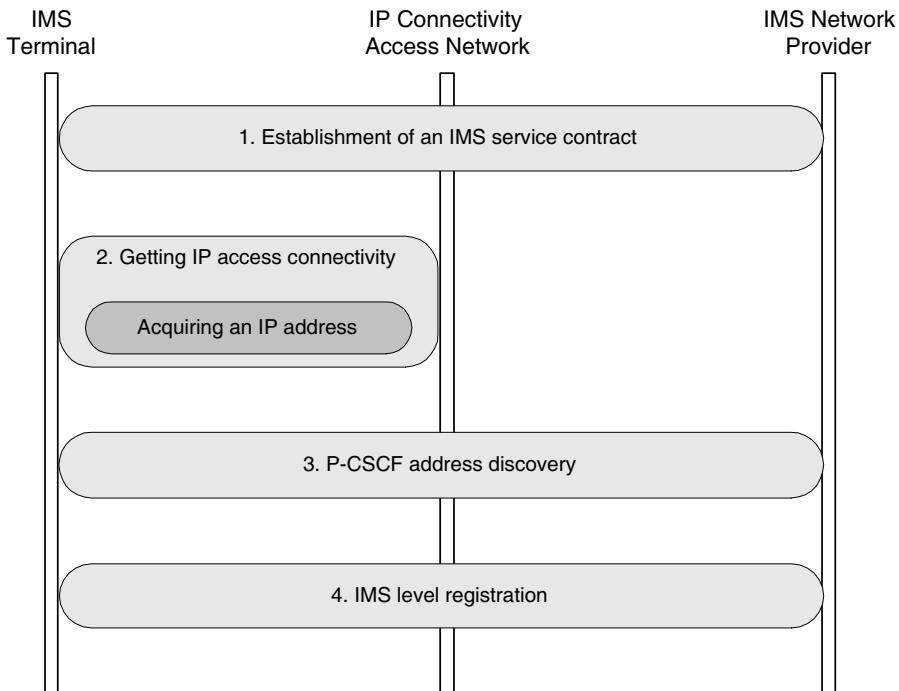


Figure 5.1: Prerequisites to get the IMS service

First, the IMS service provider has to authorize the end user to use the IMS service. This typically requires a subscription or a contract signed between the IMS network operator and the user. This contract is similar to the subscription that authorizes an end-user to receive and establish telephone calls over a wireless network.

Second, the IMS terminal needs to get access to an IP-CAN (IP Connectivity Access Network) such as GPRS (in GSM/UMTS networks), ADSL (Asymmetric Digital Subscriber Line), or WLAN (Wireless Local Access Network). IP-CAN provides access to the IMS home network or to an IMS visited network. As part of this prerequisite the IMS terminal needs to acquire an IP address (the procedures for GPRS access are described in 3GPP TS 23.060 [35]). This IP address is typically dynamically allocated by the IP-CAN operator for a determined period of time.

When these two prerequisites are fulfilled the IMS terminal needs to discover the IP address of the P-CSCF that will be acting as an outbound/inbound SIP proxy server. All the SIP signaling sent by the IMS terminal traverses this P-CSCF. When the P-CSCF discovery procedure has been completed the IMS terminal is able to send and receive SIP signaling to or from the P-CSCF. The P-CSCF is allocated permanently for the duration of IMS registration, a procedure that is typically triggered when the IMS terminal is switched on or off.

Depending on the IP Connectivity Access Network in use, the P-CSCF discovery procedure may take place as part of the process to obtain IP-CAN connectivity or as a separate procedure. A separate procedure is achieved by means of DHCP (Dynamic Host Configuration Protocol, specified in RFC 2131 [123]) or DHCPv6 (DHCP for IPv6, specified in RFC 3315 [124]).

When the previous prerequisites are fulfilled the IMS terminal registers at the SIP application level to the IMS network. This is accomplished by regular SIP registration. IMS terminals need to register with the IMS before initiating or receiving any other SIP signaling. As the IMS is modeled in different layers, the IP-CAN layer is independent of the IMS application (SIP) layer. Therefore, registration at the IMS level is independent of registration with IP-CAN (e.g., attachment to a GPRS network). The IMS registration procedure allows the IMS network to locate the user (i.e., the IMS obtains the terminal's IP address). It also allows the IMS network to authenticate the user, establish security associations, and authorize the establishment of sessions. We describe the security functions of the IMS in Chapter 12.

5.2 IPv4 and IPv6 in the IMS

When 3GPP was designing the IMS, version 6 of the Internet Protocol (also known as IPv6) was being standardized in the IETF. 3GPP did an analysis of the applicability of IPv6 for IMS and concluded that, by the time the first IMS implementations would go into operation, IPv6 would most likely be the common IP version in the Internet. Any large scale deployment of IPv4 required the allocation of private IP addresses and the presence of some type of Network Address Translator (NAT) in the path of the communication.

SIP and its associated protocols (SDP, RTP, RTCP, etc.) were known examples of protocols that suffered problems when traversing NATs. Allowing IPv4 for IMS would have required a major analysis of NAT traversal techniques. For all these reasons 3GPP decided to select IPv6 as the only allowed version of IP for IMS connectivity.

Unfortunately, when the first IMS products reached the market, the situation was quite different from the one foreseen a few years earlier. IPv6 had not taken off, IPv4 and NATs were becoming ubiquitous, and work had been done in the field of helping SIP and associated protocols to traverse NATs easily.

In June 2004 3GPP re-examined the IPv4/IPv6 dilemma once more. Market indications at that time revealed that IPv6 had not yet gone into the mainstream. Most of the Internet was still running IPv4, and only a few mobile networks were ready to start a big IPv6 deployment like the one IMS would require. Furthermore, the work on NAT traversal for SIP had progressed substantially, and SIP was a friendly NAT-traversal protocol. IPv4 was already implemented in most of the early IMS products since it did not require an extra effort.

Based on all these indications 3GPP decided to allow early deployments of IPv4 for IMS, starting even from the very first IMS release, 3GPP Release 5. The work describing the support for IPv4 in IMS was collected in 3GPP TR 23.981 [16]. The main 3GPP architecture document, 3GPP TS 23.221 [30], was amended to refer to 3GPP TR 23.981 [16] for those early IMS implementations that supported IPv4.

Dual stack implementations (IPv4 and IPv6) in both IMS terminals and network nodes are now allowed, as well as single IP version implementations. Because of this, two new nodes, the IMS-ALG (Application Layer Gateway) and the Transition Gateway (TrGW) were added to the IMS architecture. The former deals with SIP interworking and the latter with RTP interworking, both between IPv4 and IPv6 (and vice versa).

The consequence of adding IPv4 to the IMS is a delay in deploying IPv6 for the public Internet. Having IMS as the main driver of IPv6 would have accelerated the deployment of IPv6 in the Internet. While mostly everyone agrees that IPv6 will, one day, be the common version of IP in the Internet, it has not happened yet, and IMS has to go along with that fact.

The rest of this book, while considering both IPv4 and IPv6 IMS, gives precedence in examples to IPv6. We believe that IPv6 is a future-proof protocol, and we believe most of our readers will appreciate our effort to devoting a more careful analysis to IPv6 IMS.

5.3 IP Connectivity Access Network

There are multiple types of IP Connectivity Access Network. Examples of IP Connectivity Access Networks in fixed environments are Digital Subscriber Lines (DSL), dial-up lines, and enterprise Local Access Networks (LAN), etc. In wireless environments we have packet data access networks, such as GPRS or Wireless Local Access Networks (WLAN). The procedures to register and acquire an IP address are different for different IP Connectivity Access Networks.

For instance, in GPRS, the IMS terminal first undertakes a set of procedures, globally known as *GPRS attach procedures*. These procedures involve several nodes, ranging from the SGSN to the HLR and the GGSN. The procedures are illustrated in Figure 5.2. Once these procedures are complete the terminal sends an *Activate PDP Context Request* message to the SGSN requesting connection to either an IPv4 or an IPv6 network. The message includes a request for connectivity to a particular APN (Access Point Name) and packet connection type. The APN identifies the network to connect to and the address space where the IP address belongs. In the case of an IMS terminal the APN indicates a desired connection to the IMS network and the connectivity type indicates either IPv4 or IPv6. The SGSN, depending on the APN and the type of network connection, chooses an appropriate GGSN. The SGSN sends a *Create PDP Context Request* message to the GGSN. The GGSN is responsible for allocating IP addresses.

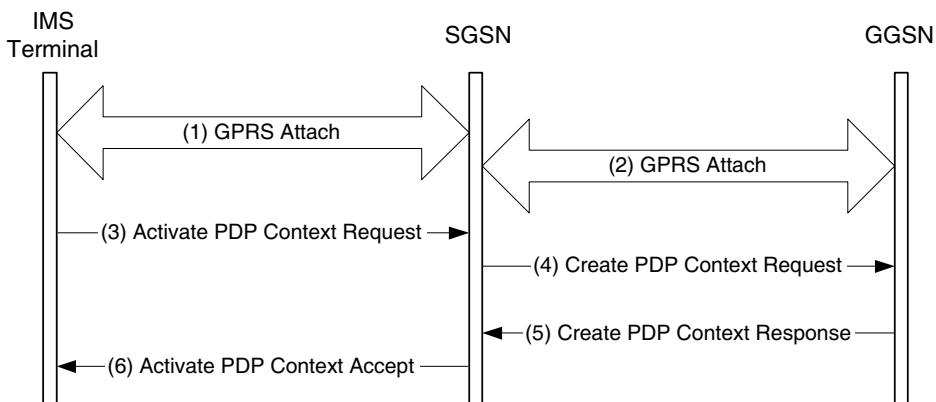


Figure 5.2: Getting IP connectivity in GPRS

If the terminal requested an IPv6 connection, the GGSN does not provide the terminal with a full IPv6 address belonging to the IMS address space. Instead, the GGSN provides the terminal with a 64-bit IPv6 prefix and includes it in a *Create PDP Context Response* message.

The SGSN transparently forwards this IPv6 prefix in an *Activate PDP Context Accept*. When the procedure is completed the IMS terminal has got a 64-bit IPv6 prefix. The terminal is able to choose any 64-bit IPv6 suffix. Together they form a 128-bit IPv6 address (i.e., the IPv6 address that the terminal will use for its IMS traffic).

If the terminal requested an IPv4 connection, the GGSN provides the terminal with its IPv4 address.

When the IP Connectivity Access Network is not GPRS the protocol used to configure the IMS terminal will most likely be DHCP (specified in RFC 2131 [123]) or DHCP for IPv6 (DHCPv6, specified in RFC 3315 [124]). DHCP is used to send configuration parameters to a terminal. Its main purpose is to provide the terminal with an IP address, although the DHCP server can also send, if requested by the terminal, other types of configuration data such as the address of an outbound SIP proxy server or the address of an HTTP proxy server.

Sometimes, the procedure to get an IP Connectivity Access Network requires a registration and a payment of some form. It has become very popular to provide Wireless LAN access at hot spots, such as airports and hotels. Getting access to these networks typically requires some form of subscription to the service, and some form of payment. It may be required that the user log into a web page and introduce a username and password, or some credit card data for payment service usage. In other cases, a 3GPP Wireless LAN access network can be used.

5.4 P-CSCF Discovery

P-CSCF discovery is the procedure by which an IMS terminal obtains the IP address of a P-CSCF. This is the P-CSCF that acts as an outbound/inbound SIP proxy server toward the IMS terminal (i.e., all the SIP signaling sent by or destined for the IMS terminal traverses the P-CSCF).

P-CSCF discovery may take place in various ways:

- integrated into the procedure that gives access to the IP-CAN;
- as a stand-alone procedure.

The integrated version of P-CSCF discovery depends on the type of IP Connectivity Access Network. If IP-CAN is a GPRS network, once the *GPRS attach procedures* are complete, the terminal is authorized to use the GPRS network. Then, the IMS terminal does a so-called *Activate PDP Context procedure*. The main goal of the procedure is to configure the IMS terminal with an IP address,⁴ but in this case the IMS terminal also discovers the IP address of the P-CSCF to which to send SIP requests.

The stand-alone version of the P-CSCF discovery procedure is based on the use of DHCP (specified in RFC 2131 [123]), or DHCPv6, for IPv6 (specified in RFC 3315 [124]), and DNS (Domain Name System, specified in RFC 1034 [217]).

In DHCPv6 the terminal does not need to know the address of the DHCP server, because it can send its DHCP messages to a reserved multicast address. If DHCP is used (for IPv4), the terminal broadcasts a discover message on its local physical subnet. In some configurations

⁴If IPv6 is used, in fact, the IMS terminal is equipped with an IPv6 prefix of 64 bits. The IMS terminal is free to select any 64-bit suffix, completing the 128 bits in an IPv6 address.

a DHCP relay may be required to relay DHCP messages to an appropriate network, although the presence of the DHCP relay is transparent to the terminal.

The procedure for DHCPv6 is shown in steps 1 and 2 of Figure 5.3. Once the IMS terminal has got connectivity to the IP-CAN, the IMS terminal sends a DHCPv6 Information-Request (1) where it requests the DHCPv6 options for SIP servers (specified in RFC 3319 [305]). In the case of the IMS the P-CSCF performs the role of an outbound/inbound SIP proxy server, so the DHCP server returns a DHCP Reply message (2) that contains one or more domain names and/or IP addresses of one or more P-CSCFs.

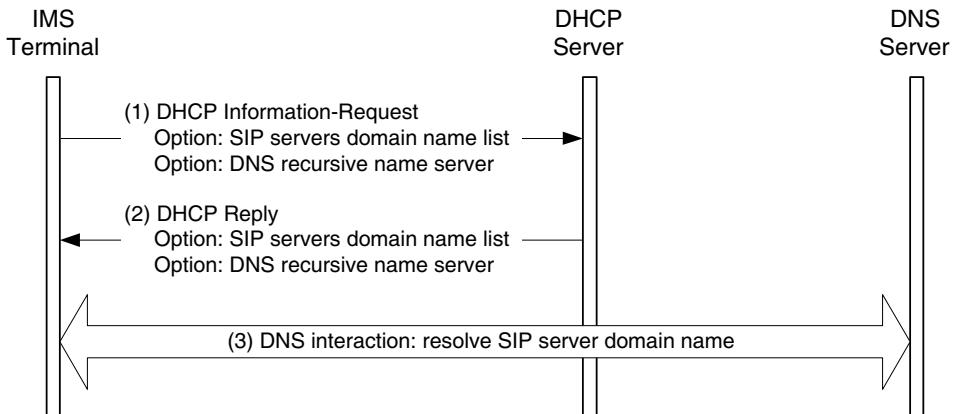


Figure 5.3: P-CSCF discovery procedure based on DHCP and DNS

At the discretion of the IMS terminal implementation, there are two possible ways in which the IMS terminal can specify the request for the DHCPv6 option for SIP servers.

- The IMS terminal requests the *SIP servers domain name list* option in the DHCPv6 Information-Request message.⁵ The DHCPv6 Reply message contains a list of the domain names of potential P-CSCFs. The IMS terminal needs to resolve at least one of these domain names into an IPv6 address. A query-response dialog with DNS resolves the P-CSCF domain name, but prior to any DNS interaction, the IMS terminal also needs to get the address of one or more DNS servers to send its DNS messages. To solve this problem the DHCP Information-Request message, (1) in Figure 5.3, not only contains a request for the option for SIP servers, but also includes a request for the *DNS recursive name server option*. The DHCPv6 Reply message (2) contains a list of IPv6 addresses of DNS servers, in addition to the domain name of the P-CSCF. Then, the IMS terminal queries the just learnt DNS server in order to resolve the P-CSCF domain name into one or more IPv6 addresses. The procedures to resolve a SIP server into one or more IP addresses are standardized in RFC 3263 [285].
- The alternative consists of the IMS terminal requesting the *SIP servers IPv6 address list* option in the DHCPv6 Information-Request message. The DHCP server answers in

⁵The DHCPv6 option for SIP servers differs from a similar option in DHCPv4. In DHCPv6 there are two different option codes to request: either the domain name or the IP address of the SIP server. In DHCPv4 there is a single option code, with two possible answers: domain names or IPv4 addresses. It seems that the maximum number of DHCPv4 options is limited to 256, whereas in DHCPv6 the maximum number of options is 65535. This gives enough room to allocate the two option codes needed.

a DHCP Reply message that contains a list of IPv6 addresses of the P-CSCF allocated to the IMS terminal. In this case, no interaction with DNS is needed, because the IMS terminal directly gets one or more IPv6 addresses.

These two ways are not mutually exclusive. It is possible, although not required, that an IMS terminal requests both the *SIP servers domain name list* and the *SIP servers IPv6 address list*. The DHCP server may be configured to answer both or just one of the options, but if the DHCP server answers with both lists the IMS terminal should give precedence to the *SIP servers domain name list*. Handling of all these conflicts is described in RFC 3263 [285].

Another way to provide the address of the P-CSCF relies in some means of configuration. This can be, for example, an SMS sent to the terminal for the purpose of configuration or the client provisioning [227] or device management [231] specified by the Open Mobile Alliance (OMA).

Eventually, the IMS terminal discovers the IP address of its P-CSCF and can send SIP signaling to its allocated P-CSCF. The P-CSCF takes care of forwarding the SIP signaling to the next SIP hop. The P-CSCF allocated to the IMS terminal does not change until the next P-CSCF discovery procedure. This procedure typically takes place when the terminal is switched on or during severe error conditions. The important aspect to highlight is that the IMS terminal does not need to worry about possible changes of address of the P-CSCF, because its address is not variable.

5.5 IMS-level Registration

Once the IMS terminal has followed the procedures of getting access to an IP Connectivity Access Network, has acquired an IPv4 address or built an IPv6 address, and has discovered the IPv4 or IPv6 address of its P-CSCF, the IMS terminal can begin registration at the IMS level.

IMS-level registration is the procedure where the IMS user requests authorization to use the IMS services in the IMS network. The IMS network authenticates and authorizes the user to access the IMS network.

IMS-level registration is accomplished by a SIP REGISTER request. We explained in Section 4.1.4 that a SIP registration is the procedure whereby a user binds his *public URI* to a URI that contains the host name or IP address of the terminal where the user is logged in. Unlike regular SIP procedures, registration with the IMS is mandatory before the IMS terminal can establish a session.

The IMS registration procedure uses a SIP REGISTER request. However, this procedure is heavily overloaded in the IMS, for the sake of fulfilling the 3GPP requirement of a minimum number of round trips. The goal is achieved and the procedure completes after two round trips, as illustrated in Figure 5.4.⁶

5.5.1 IMS Registration with an ISIM

We explained in Section 3.6 that in order to authenticate users, when they access the IMS network, the IMS terminal needs to be equipped with a UICC. The UICC can include an ISIM application, a USIM application, or both. The parameters stored in both applications are completely different, since the ISIM is IMS-specific and the USIM was already available

⁶Note that, for the sake of simplicity, Figure 5.4 does not show a Subscriber Location Function (SLF). An SLF is needed if there is more than one HSS in the home network of the subscriber.

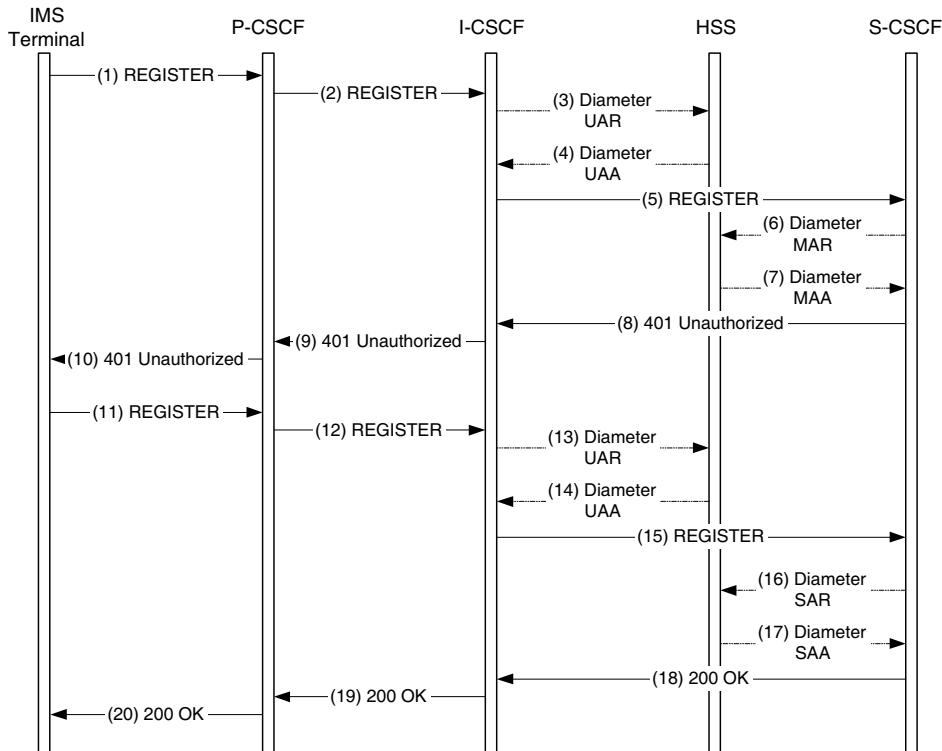


Figure 5.4: Registration at the IMS level

for circuit-switched and packet-switched networks before the IMS was designed. Although the registration procedure is quite similar, independently of the presence of an ISIM or USIM, there are certainly detailed differences. In this section we describe access to the IMS with an ISIM application in the UICC. Section 5.5.2 describes the registration procedures when the UICC only contains a USIM.

The IMS registration procedure satisfies the following requirements in two round trips:

- the user binds a Public User Identity to a contact address – this is the main purpose of a SIP REGISTER request;
- the home network authenticates the user;
- the user authenticates the home network;
- the home network authorizes the SIP registration and the usage of IMS resources;
- if the P-CSCF is located in a visited network, the home network verifies that there is an existing roaming agreement between the home and the visited network and authorizes the usage of the P-CSCF;
- the home network informs the user about other possible identities that the home network operator has allocated exclusively to that user;

- the IMS terminal and the P-CSCF negotiate the security mechanism that will be in place for subsequent signaling;
- the P-CSCF and the IMS terminal establish a set of security associations that protect the integrity of SIP messages sent between the P-CSCF and the terminal;
- both the IMS terminal and the P-CSCF upload to each other the algorithms used for compression of SIP messages.

In this section we focus on a few aspects of the IMS registration procedure. Section 12.1.1 describes the security aspects that relate to registration.

Before creating the initial SIP REGISTER request, the IMS terminal retrieves from ISIM the Private User Identity, a Public User Identity, and the home network domain URI. Then, the IMS terminal creates a SIP REGISTER request and includes the following four parameters.

The registration URI. This is the SIP URI that identifies the home network domain used to address the SIP REGISTER request. This is a URI that typically points to the home network, but it could be any subdomain of the home network. The registration URI is included in the *Request-URI* of the REGISTER request.

The Public User Identity. This is a SIP URI that represents the user ID under registration. In SIP, it is known as the SIP *Address-of-Record* (i.e., the SIP URI that users print in their business cards). It is included in the To header field value of the REGISTER request.

The Private User Identity. This is an identity that is used for authentication purposes only, not for routing. It is equivalent to what in GSM is known as IMSI (International Mobile Subscriber Identity); it is never displayed to the user. It is included in the *username* parameter of the Authorization header field value included in the SIP REGISTER request.

The Contact address. This is a SIP URI that includes the IP address of the IMS terminal or the host name where the user is reachable. It is included in the SIP Contact header field value of the REGISTER request.

According to Figure 5.4 the IMS creates a SIP REGISTER request including the above-mentioned information. Figure 5.5 shows an example of the REGISTER request that the IMS terminal sends to the P-CSCF. It must be noted that the P-CSCF may be either located in a visited or a home network. So, in general, the P-CSCF may not be located in the same network as the home network, and it needs to locate an entry point into the home network by executing the DNS procedures specified in RFC 3263 [285]. These procedures provide the P-CSCF with the SIP URI of an I-CSCF. That I-CSCF is located at the entrance to the home network. The P-CSCF inserts a P-Visited-Network-ID that contains an identifier of the network where the P-CSCF is located. The home network requires this header field to validate the existence of a roaming agreement between the home and visited networks. The P-CSCF also inserts a Path header field with its own SIP URI to request the home network to forward all SIP requests through this P-CSCF. Eventually, the P-CSCF forwards the SIP REGISTER request to an I-CSCF in the home network, (2) in Figure 5.4.

The I-CSCF does not keep a registration state, mainly because it is typically configured in DNS to be serving a load-balancing function. When a SIP proxy needs to contact a SIP proxy

```

REGISTER sip:home1.net SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A];comp=sigcomp;
      branch=z9hG4bK9h9ab
Max-Forwards: 70
From: <sip:alice@home1.net>;tag=s8732n
To: <sip:alice@home1.net>
Contact: <sip:[1080::8:800:200C:417A];comp=sigcomp>
      ;expires=600000
Call-ID: 23fi57lju
Authorization: Digest username="alice_private@home1.net",
      realm="home1.net", nonce="",
      uri="sip:home1.net", response=""
Security-Client: ipsec-3gpp; alg=hmac-sha-1-96;
      spi-c=3929102; spi-s=0293020;
      port-c:3333; port-s=5059
Require: sec-agree
Proxy-Require: sec-agree
Cseq: 1 REGISTER
Supported: path
Content-Length: 0

```

Figure 5.5: (1) REGISTER

located in another network, it gets a different IP address of an I-CSCF, because of the DNS load-balancing mechanisms. As a consequence, I-CSCFs do not keep any state associated to registration. In particular, I-CSCFs are not aware of whether an S-CSCF is allocated to the user and what the address of such an S-CSCF would be.

In order to carry out a first-step authorization and to discover whether there is an S-CSCF already allocated to the user, the I-CSCF sends a Diameter User-Authentication-Request (UAR) to the HSS, (3) in Figure 5.4. The I-CSCF transfers to the HSS the Public User Identity and Private User Identity and the visited network identifier, all of which are extracted from the SIP REGISTER request. The HSS authorizes the user to roam the visited network and validates that the Private User Identity is allocated to the Public User Identity under registration. The HSS answers with a Diameter User-Authentication-Answer (UAA) (4). The HSS also includes the SIP URI of a previously allocated S-CSCF in the Diameter UAA message, if there was an S-CSCF already allocated to the user. However, if this was the first registration (e.g., after the user switched the IMS terminal on), there will most likely not be an S-CSCF allocated to the user. Instead, the HSS returns a set of S-CSCF capabilities that are the input for the I-CSCF when selecting the S-CSCF.

Let us assume for the time being that the user switches his IMS terminal on and the S-CSCF is unallocated as yet. The I-CSCF needs to perform an S-CSCF selection procedure based on the S-CSCF capabilities that the HSS returned in the Diameter UAA message. These capabilities are divided into mandatory capabilities, or capabilities that the chosen S-CSCF has to fulfill, and optional capabilities, or capabilities that the chosen S-CSCF may or may not fulfill. The standard does not indicate *what* these capabilities are and *how* they are specified. Instead, capabilities are represented by an integer and have semantics only in a particular home network. As an example, let us assume that operator A assigns

the semantics of *capability 1* to the S-CSCF that provides detailed charging information, whereas *capability 2* indicates support in the S-CSCF for SIP calling preferences. Then, the operator can configure the user data in the HSS to indicate that for this particular subscriber it is mandatory that the S-CSCF supports capability 2 and, optionally, that it may support capability 1. Because the Diameter interface defined between the I-CSCF and the HSS is an intra-operator interface, mapping the capabilities to the semantics is a matter of operator configuration. In different networks, capabilities 1 and 2 will have different semantics.

S-CSCF selection is based on the capabilities received from the HSS in the Diameter UAA. The I-CSCF has a configurable table of S-CSCFs operating in the home network and the capabilities supported by each one. This allows the I-CSCF to choose an appropriate S-CSCF for this particular user. Then, the I-CSCF continues with the process by proxying the SIP REGISTER request to the chosen S-CSCF, (5) in Figure 5.4. An example of such a REGISTER request is shown in Figure 5.6. The S-CSCF receives the REGISTER request and authenticates the user. Initial registrations are always authenticated in the IMS. Other registrations may or may not be authenticated, depending on a number of security issues. Only REGISTER requests are authenticated in the IMS. Other SIP requests, such as INVITE, are never authenticated by the IMS.

```

REGISTER sip:scscf1.home1.net SIP/2.0
Via: SIP/2.0/UDP icscf1.home1.net;branch=z9hG4bKea1dof,
      SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz,
      SIP/2.0/UDP [1080::8:800:200C:417A];comp=sigcomp;
      branch=z9hG4bK9h9ab
Max-Forwards: 68
From: <sip:alice@home1.net>;tag=s8732n
To: <sip:alice@home1.net>
Contact: <sip:[1080::8:800:200C:417A];comp=sigcomp>
          ;expires=600000
Call-ID: 23fi57lju
Authorization: Digest username="alice_private@home1.net",
               realm="home1.net", nonce="",
               uri="sip:home1.net", response="",
               integrity-protected="no"
Require: path
Supported: path
Path: <sip:term@pcscf1.visited1.net;lr>
P-Visited-Network-ID: "Visited 1 Network"
P-Charging-Vector: icid-value="W34h6dlg"
Cseq: 1 REGISTER
Content-Length: 0

```

Figure 5.6: (5) REGISTER

The S-CSCF then contacts the HSS for a double purpose. On the one hand, the S-CSCF needs to download authentication data to perform authentication for this particular user. On the other hand, the S-CSCF needs to save the S-CSCF URI in the HSS, so that any further query to the HSS for the same user will return routing information pointing to this S-CSCF.

For this purpose the S-CSCF creates a Diameter Multimedia-Auth-Request (MAR) message, (6) in Figure 5.4. The HSS stores the S-CSCF URI in the user data and answers in a Diameter Multimedia-Auth-Answer (MAA) message, (7) in Figure 5.4. In 3GPP IMS, users are authenticated by the S-CSCF with data provided by the HSS. These authentication data are known as *authentication vectors*. The HSS includes one or more authentication vectors in the Diameter MAA message, so that the S-CSCF can properly authenticate the user. Then, the S-CSCF creates a SIP 401 (Unauthorized) response, (8) in Figure 5.4. This response includes a challenge in the WWW-Authenticate header field that the IMS terminal should answer.

The SIP 401 (Unauthorized) response is forwarded, according to regular SIP procedures, via the I-CSCF and P-CSCF. An example of such a response is shown in Figure 5.7. When the IMS terminal receives the SIP 401 (Unauthorized) response, it realizes that there is a challenge included and produces an appropriate response to that challenge. The response to the challenge (sometimes known as credentials) is included in a new SIP REGISTER request, (11) in Figure 5.4. The actual contents of the credentials depend on the IMS network. If we are dealing with a 3GPP IMS terminal, then the terminal stores authentication information in a smart card (UICC (Universal Integrated Circuit Card)). The IMS terminal extracts or derives the parameters stored in the smart card to build the credentials, and does so transparently to the user. Chapter 12 is devoted to security in IMS networks.

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP [1080::8:800:200C:417A];comp=sigcomp;
      branch=z9hG4bK9h9ab
From: <sip:alice@home1.net>;tag=s8732n
To: <sip:alice@home1.net>;tag=409sp3
Call-ID: 23fi57lju
WWW-Authenticate: Digest realm="home1.net",
                  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                  algorithm=AKAv1-MD5
Security-Server: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96;
                  spi-c=909767; spi-s=421909;
                  port-c:4444; port-s=5058
Cseq: 1 REGISTER
Content-Length: 0
```

Figure 5.7: (10) 401 Unauthorized

In response, the IMS terminal sends a new SIP REGISTER request to the P-CSCF, (11) in Figure 5.4. An example of this new SIP REGISTER request is shown in Figure 5.8.

The P-CSCF does the same operation as for the first REGISTER request; that is, it determines the entry point in the network stored in the *Request-URI* of the REGISTER request and finds an I-CSCF in the home network. It must be noted that this I-CSCF, because of DNS load-balancing mechanisms, may not be the same I-CSCF that the first REGISTER request, (2) in Figure 5.4, traversed.

The I-CSCF sends a new Diameter UAR message, (13) in Figure 5.4, for the same reasons as explained before. The difference in this situation is that the Diameter UAA message (14) includes routing information: the SIP URI of the S-CSCF allocated to the user. The HSS stored this URI when it received a Diameter MAR message (6). Therefore, no matter whether the I-CSCF is the same I-CSCF the first REGISTER request traversed or not, the second

```

REGISTER sip:home1.net SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Max-Forwards: 70
P-Access-Network-Info: 3GPP-UTRAN-TDD;
      utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=s8732n
To: <sip:alice@home1.net>
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
      ;expires=600000
Call-ID: 23fi571ju
Authorization: Digest username="alice_private@home1.net",
      realm="home1.net",
      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
      algorithm=AKAv1-MD5,
      uri="sip:home1.net",
      response="6629fae49393a05397450978507c4ef1"
Security-Verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96;
      spi-c=909767; spi-s=421909;
      port-c:4444; port-s=5058
Require: sec-agree
Proxy-Require: sec-agree
Cseq: 2 REGISTER
Supported: path
Content-Length: 0

```

Figure 5.8: (11) REGISTER

REGISTER request ends up in the same S-CSCF, the one that was allocated to the user at the time of the registration. The S-CSCF receives the REGISTER request (15) that includes the user credentials. An example of this REGISTER request is displayed in Figure 5.9. The S-CSCF then validates these credentials against the authentication vectors provided by the HSS in a Diameter MAA message (7) in Figure 5.4. If authentication is successful, then the S-CSCF sends a Diameter SAR message to the HSS (16) to inform the HSS that the user is now registered and to download the user profile (17). The user profile is an important piece of information that includes, among other things, the collection of all the Public User Identities allocated for authentication of the Private User Identity. It also indicates to the S-CSCF which of these Public User Identities are automatically registered in the S-CSCF in a *set of implicitly registered Public User Identities*. In addition, the user profile also contains the *initial filter criteria*, which is the collection of triggers that determine when a SIP request is forwarded to the Application Server that will be providing the service.

At this stage the S-CSCF has stored the contact URI for this user, as it was present in the Contact header field of the SIP REGISTER request. It has also stored the list of URIs included in the Path header field. This list always includes the P-CSCF URI and may optionally include an I-CSCF URI. Later, the S-CSCF will route initial SIP requests addressed to the user via the list of URIs included in the Path header field and the contact address in this order.

```

REGISTER sip:scscf1.home1.net SIP/2.0
Via: SIP/2.0/UDP icscf1.home1.net;branch=z9hG4bKea1dof
Via: SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
branch=z9hG4bK9h9ab
Max-Forwards: 68
P-Access-Network-Info: 3GPP-UTRAN-TDD;
utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=s8732n
To: <sip:alice@home1.net>
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
;expires=600000
Call-ID: 23fi57lju
Authorization: Digest username="alice_private@home1.net",
realm="home1.net",
nonce="dc98b7102dd2f0e8b11d0f600bfb0c093",
algorithm=AKAv1-MD5,
uri="sip:home1.net",
response="6629fae49393a05397450978507c4ef1",
integrity-protected="yes"
Require: path
Supported: path
Path: <sip:term@pcscf1.visited1.net;lr>
P-Visited-Network-ID: "Visited 1 Network"
P-Charging-Vector: icid-value="W34h6dlg"
Cseq: 2 REGISTER
Content-Length: 0

```

Figure 5.9: (15) REGISTER

Last, but not least, the S-CSCF sends a 200 (OK) response to the REGISTER request, to indicate the success of the REGISTER request, (18) in Figure 5.4. An example of this response is displayed in Figure 5.10. The 200 (OK) response includes a P-Associated-URI header field that contains the list of implicitly registered Public User Identities, including the one under registration. The only exception is barred Public User Identities, i.e., those that can be used for registration purposes but not for regular traffic, which are never present in the P-Associated-URI header field.

The 200 (OK) response also contains a Service-Route header field that includes a list of SIP server URIs. Future SIP requests (excluding REGISTER requests, which are always routed according to the instructions received from the HSS) that the IMS terminal sends will be routed via these SIP servers, in addition to the outbound proxy (P-CSCF). In the IMS the Service-Route header field value always contains the address of the S-CSCF of the user and may also contain the address of an I-CSCF in the home network.

The 200 (OK) response traverses the same I-CSCF and P-CSCF that the REGISTER request traversed. Eventually, the IMS terminal gets the 200 (OK) response, (20) in Figure 5.4. At this stage the registration procedure is complete. The IMS terminal is

```

SIP/2.0 200 OK
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Path: <sip:term@pcscf1.visited1.net;lr>
Service-Route: <sip:orig@scscf1.home1.net;lr>
From: <sip:alice@home1.net>;tag=s8732n
To: <sip:alice@home1.net>;tag=409sp3
Call-ID: 23fi57lju
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
          ;expires=600000
Cseq: 2 REGISTER
Date: Wed, 21 January 2004 18:19:20 GMT
P-Associated-URI: <sip:alice@home1.net>,
                  <sip:alice-family@home1.net>,
                  <sip:+1-212-555-1234@home1.net;user=phone>
Content-Length: 0

```

Figure 5.10: (20) 200 OK

registered with the IMS for the duration of time indicated in the `expires` parameter of the `Contact` header field.

As the reader may have noticed, the registration process with the IMS is based on SIP `REGISTER` requests that are authenticated. The procedure is overloaded with some extra functionality. For instance, the SIP `Path` header field extension (specified in RFC 3327 [316]) informs the S-CSCF of a P-CSCF (and perhaps an I-CSCF) via which SIP requests addressed to the IMS terminal should be routed. In the opposite direction the SIP `Service-Route` header field extension (specified in RFC 3608 [317]) provides the IMS terminal with a sequence of proxies via which future SIP requests have to be routed (e.g., S-CSCF), in addition to the outbound SIP server (P-CSCF). We also indicated the existence of a SIP `P-Associated-URI` header field (specified in RFC 3455 [154]⁷). The S-CSCF populates the `P-Associated-URI` header field with the list of non-barred implicitly set of registered Public User Identities. The first URI included in this list is the default Public User Identity, i.e., the identity that is used when the user does not express a preference for any of their identities when it initiates a session or SIP dialog.

We want to stress that the IMS terminal is able to register a Public User Identity at any time. For instance, when the IMS application in the terminal is switched on, the IMS terminal may be configured to register one Public User Identity. Other Public User Identities may be registered later, at any time, upon explicit indication of the user. For example, this allows us to register independently a personal Public User Identity from a business Public User Identity.

5.5.2 IMS Registration with a USIM

If the IMS terminal is equipped with a UICC that does not contain an ISIM application, perhaps because the card was acquired before the IMS service came into operation, the user can still register with the IMS network, but there are a few problems.

⁷Note that 3GPP TS 24.229 changes the semantics of the `P-Associated-URI` header field with respect to those stated in RFC 3455. The interested reader should consult the former for the most updated semantics.

The first problem is that the IMS terminal is unable to extract or derive the Private User Identity, a Public User Identity and the home network domain URI to address the SIP REGISTER request to, because all of these parameters are stored in the ISIM, not the USIM. However, the IMS terminal can access a USIM. Of special interest in the USIM is the IMSI, which is a collection of a maximum of 15 decimal digits that globally represent the identity of a mobile subscriber, including their country and mobile network operator. The home operator allocates an IMSI to each subscriber.

Figure 5.11 depicts the structure of an IMSI. Beginning from the left side, the first three digits form the *Mobile Country Code (MCC)*. The MCC represents the country of operation of the home network. The MCC is followed by two or three digits that constitute the *Mobile Network Code (MNC)*. The MNC represents the home operator within the country of the MCC. The remaining digits constitute the *Mobile Subscriber Identification Number (MSIN)*.

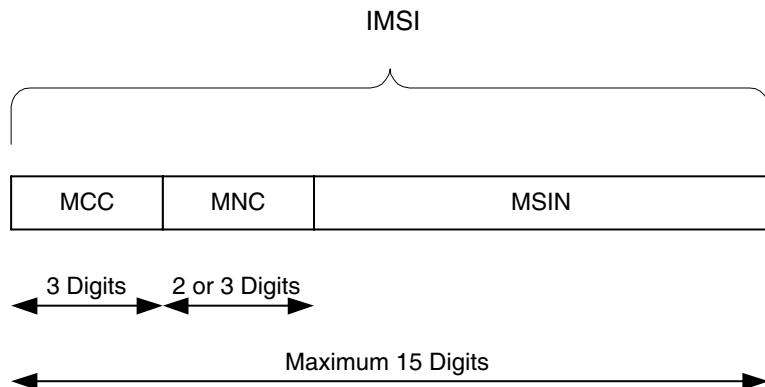


Figure 5.11: Structure of an IMSI

The IMSI is never used for routing calls in cellular networks; it is just used for identification of subscribers and their data stored in the network, authentication, and authorization purposes.

When an IMS terminal is loaded with a UICC that does not contain an ISIM, the terminal extracts the IMSI from the USIM in order to build a *temporary Private User Identity*, a *temporary Public User Identity*, and a *home network domain URI* that allows it to build a SIP REGISTER request and route it to the home network. These three parameters are only used during registration, re-registration, and deregistration procedures. When the user is eventually registered, the S-CSCF sends a collection of the regular Public User Identities allocated to the user. The IMS terminal only uses these Public User Identities for any SIP traffic other than REGISTER requests. Consequently, the temporary identities are never known or used outside the home network (e.g., in a session setup).

5.5.2.1 Temporary Private User Identity

The temporary Private User Identity is derived from the IMSI. A regular Private User Identity has the format `username@realm`. A temporary Private User Identity has the same format. On building a temporary Private User Identity the IMS terminal inserts the complete IMSI as

the `username` of the Private User Identity. The `realm` gets split into subrealms separated by dots (like a DNS subdomain), where the first subrealm is the string “ims”, the next subrealm contains the string “mnc” followed by the three digits of the MNC in the IMSI, the next subrealm contains the string “mcc” followed by the three digits of the MCC in the IMSI, and the rest is the fixed string `.3gppnetwork.org`. As an example, assume an IMSI: 2483235551234, where the MCC is 248, the MNC is 323, and the MSIN is 5551234. According to the explanation above, the derived temporary Private User Identity from that IMSI is

```
2483235551234@ims.mnc323.mcc248.3gppnetwork.org
```

Note that Private User Identities are not SIP URIs.

5.5.2.2 Temporary Public User Identity

An IMS terminal without an ISIM also needs to build a temporary Public User Identity to be registered. A regular Public User Identity during registration is a SIP URI that takes the format `sip:user@domain`. It is very simple to build a temporary Public User Identity, because it takes the same format as the temporary Private User Identity, but now, it is prepended by the string: “`sip:`”, since the identity is a SIP URI. So, if we take as an example the same IMSI that we chose to illustrate a temporary Private User Identity, the corresponding temporary Public User Identity is

```
sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org
```

5.5.2.3 Home Network Domain URI

When an IMS terminal equipped only with a USIM needs to build a home network domain URI for inclusion in the *Request-URI* of the SIP REGISTER request, the terminal just removes the “user” part of the temporary Public User Identity. According to the example that we have been following the home network domain URI is set to

```
sip:ims.mnc323.mcc248.3gppnetwork.org
```

5.5.2.4 Registration Flow

Registration flow is the same no matter whether the IMS terminal is provided with an ISIM or a USIM application in the UICC. Figure 5.4 was discussed when we described registration with an ISIM and it is still valid with a USIM. However, the contents of the messages change, since the messages convey a temporary Private User Identity and a temporary Public User Identity.

Figure 5.12 shows an example of a REGISTER request when the IMS terminal is equipped with a USIM in the UICC. The *Request-URI* and the `uri` parameter of the `Authorization` header are set to the home network domain derived from the IMSI. The `From` and `To` headers are set to the temporary Public User Identity derived from the IMSI. The `username` parameter in the `Authorization` header is set to the temporary Private User Identity also derived from the IMSI.

When the P-CSCF receives the REGISTER request (1), it performs its regular procedures to discover an I-CSCF in the home network. In this case, since the *Request-URI* of the REGISTER request is set to `ims.mnc323.mcc248.3gppnetwork.org`, the P-CSCF uses

```

REGISTER sip:ims.mnc323.mcc248.3gppnetwork.org SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A];comp=sigcomp;
     branch=z9hG4bK9h9ab
Max-Forwards: 70
P-Access-Network-Info: 3GPP-UTRAN-TDD;
     utran-cell-id-3gpp=24450289A3299239
From: <sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org>;tag=4fa3
To: <sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org>
Contact: <sip:[1080::8:800:200C:417A];comp=sigcomp>
     ;expires=600000
Call-ID: 23fi57lju
Authorization: Digest
     username="2483235551234@ims.mnc323.mcc248.3gppnetwork.org",
     realm="ims.mnc323.mcc248.3gppnetwork.org", nonce="",
     uri="sip:ims.mnc323.mcc248.3gppnetwork.org", response=""
Security-Client: ipsec-3gpp; alg=hmac-sha-1-96;
     spi-c=3929102; spi-s=0293020;
     port-c:3333; port-s=5059
Require: sec-agree
Proxy-Require: sec-agree
Cseq: 1 REGISTER
Supported: path
Content-Length: 0

```

Figure 5.12: (1) REGISTER

this domain as the input to the DNS procedures (which are specified in RFC 3263 [285]). This requires that home network operators have configured appropriately not only the DNS entries corresponding to their own domain name but also the DNS entries corresponding to a DNS domain that follows the 3gppnetwork.org pattern.

The I-CSCF queries the HSS with a Diameter UAR message (3) that contains the temporary Private User Identities and Public User Identities extracted from the REGISTER request. Eventually, the I-CSCF forwards the REGISTER request (5) to the S-CSCF. The S-CSCF downloads the authentication vectors from the HSS with a Diameter MAA message (7). These vectors are synchronized with the USIM in the UICC, since there is no ISIM present. The S-CSCF challenges the IMS terminal in a 401 (Unauthorized) response (8), which is forwarded to the IMS terminal via the I-CSCF (9) and the P-CSCF (10). Figure 5.13 shows an example of the contents of the 401 (Unauthorized) response (10).

The IMS terminal computes the challenge within the USIM, builds a new SIP REGISTER request (11) that contains an answer to the challenge, and sends it again to the P-CSCF, as shown in Figure 5.14. The message is forwarded to the S-CSCF.

When the S-CSCF gets the REGISTER request (15), it verifies the response and providing it was correct, it contacts the HSS to download the user profile. Part of the user profile contains the list of Public User Identities that are equivalent, or alias to it, and some of them are implicitly registered when the user registers any of them. The HSS also sends an indication of whether the temporary Public User Identity derived from the IMSI is barred,

```

SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP [1080::8:800:200C:417A];comp=sigcomp;
      branch=z9hG4bK9h9ab
From: <sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org>;tag=4fa3
To: <sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org>;tag=409sp3
Call-ID: 23fi57lju
WWW-Authenticate: Digest realm="ims.mnc323.mcc248.3gppnetwork.org",
                  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                  algorithm=AKAv1-MD5
Security-Server: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96;
                  spi-c=909767; spi-s=421909;
                  port-c:4444; port-s=5058
Cseq: 1 REGISTER
Content-Length: 0

```

Figure 5.13: (10) 401 Unauthorized

```

REGISTER sip:ims.mnc323.mcc248.3gppnetwork.org SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Max-Forwards: 70
P-Access-Network-Info: 3GPP-UTRAN-TDD;
      utran-cell-id-3gpp=24450289A3299239
From: <sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org>;tag=4fa3
To: <sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org>
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
          ;expires=600000
Call-ID: 23fi57lju
Authorization: Digest
              username="2483235551234@ims.mnc323.mcc248.3gppnetwork.org",
              realm="ims.mnc323.mcc248.3gppnetwork.org",
              nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
              algorithm=AKAv1-MD5,
              uri="sip:ims.mnc323.mcc248.3gppnetwork.org",
              response="6629fae49393a05397450978507c4ef1"
Security-Verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96;
                  spi-c=909767; spi-s=421909;
                  port-c:4444; port-s=5058
Require: sec-agree
Proxy-Require: sec-agree
Cseq: 2 REGISTER
Supported: path
Content-Length: 0

```

Figure 5.14: (11) REGISTER

so that the user cannot originate or receive any SIP traffic with that temporary Public User Identity other than for registrations.

The S-CSCF inserts all the non-barred Public User Identities that are associated with this registered Public User Identity in the P-Associated-URI header field value. This header field indicates which identities have been automatically registered under the so-called set of implicitly registered Public User Identities. The S-CSCF sends the 200 (OK) response, (18) in Figure 5.4, via the I-CSCF and P-CSCF to the IMS terminal (20). Figure 5.15 shows an example of the 200 (OK) response, (20) in Figure 5.4, that the IMS terminal receives. In this example, the temporary Public User Identity is barred, since it is not included in the P-Associated-URI header field.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Path: <sip:term@pcscf1.visited1.net;lr>
Service-Route: <sip:orig@scscf1.home1.net;lr>
From: <sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org>;tag=4fa3
To: <sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org>;tag=409sp3
Call-ID: 23fi57lju
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
          ;expires=600000
Cseq: 2 REGISTER
Date: Wed, 21 January 2004 18:19:20 GMT
P-Associated-URI: <sip:alice@home1.net>,
                  <sip:alice-family@home1.net>,
                  <sip:+1-212-555-1234@home1.net;user=phone>
Content-Length: 0
```

Figure 5.15: (20) 200 OK

Once the registration process is complete the IMS terminal has got a set of regular Public User Identities that can be used for subscriptions, establishing sessions, etc. The temporary identities are used only for registration purposes.

5.6 Subscription to the reg Event State

Let us consider for a second the operation of SIP in a general Internet context rather than in the IMS. In SIP a user agent can register with a registrar. The registrar accepts the registration and creates a registration state. When registering, the SIP User Agent publishes its contact address (i.e., the location where the user is available). The registrar stores this information for a determined length of time, according to the expiration timer of the SIP REGISTER request. Now, let us imagine for a second that the registrar gracefully shuts down, or simply that the operator of the registrar wants, for whatever reason, to clear the registration state of the SIP User Agent in the registrar. Would the SIP User Agent be informed about this hypothetical deregistration? The answer is no, because basic SIP functionality does not offer a mechanism for the UA to be informed that a deregistration has occurred.

In the IMS there is a clear requirement, perhaps inherited from the GSM age, for the user to be informed about whether he is reachable or not (i.e., whether the terminal is

under radio coverage and whether the user is registered with the network or not). Most existing GSM phones provide information to the user on whether the phone is under radio coverage or not and whether the terminal is registered to the network. The core SIP specified in RFC 3261 [286] does not offer a solution to this requirement. The solution that the IETF worked out and the IMS adopted was to create a registration package for the SIP event framework. The solution (specified in RFC 3680 [269]), allows an IMS terminal to subscribe to its registration-state information stored in the S-CSCF. When the IMS terminal has completed its registration, it sends a SUBSCRIBE request for the `reg` event. This request is addressed to the same Public User Identity that the SIP User Agent just registered. The S-CSCF receives the request and installs that subscription (i.e., the S-CSCF takes the role of a notifier, according to RFC 3265 [264]). According to the same RFC, the S-CSCF sends a NOTIFY request to the user (e.g., the IMS terminal). This request includes an XML document in its body (specified in RFC 3680 [269]) that contains a list of all the Public User Identities allocated to the user, along with the user's registration state. The IMS terminal now knows whether the user is registered or not, and which Public User Identities the user is registered with (remember that the user may be registered with the IMS network with more than a single Public User Identity). The IMS terminal may decide to display a list of all the Public User Identities that the operator has allocated to the user. Furthermore, the IMS terminal can display different icons for registered and non-registered Public User Identities that belong to the user.

If the S-CSCF has to shut down or if the operator needs to manually deregister a Public User Identity, the registration information will be changed. This will provoke the S-CSCF into informing each subscriber of the `reg` event of that user.

In addition to the IMS terminal subscribing to its own registration state, the P-CSCF also subscribes to the registration state of the user. This registration allows the P-CSCF to be informed in real time about which Public User Identities are registered. If there is administrative action taking place on one or all of them, the P-CSCF can delete any state it may have.⁸

Other entities that may require a subscription to the `reg` state of a user are Application Servers. However, this subscription is not compulsory, as it depends on the type of service offered to the user. For instance, an Application Server offering a welcome message when a user switches on their IMS terminal may subscribe to the `reg` state of the user, so that when the user connects to the IMS network, the Application Server is informed that the user has now switched their IMS terminal on. In response, the Application Server sends an instant message to the user, giving a welcome message or any other information of interest to the user.

Figure 5.16 shows the complete registration flow.⁹ This figure expands the contents of Figure 5.4 by including additionally the subscriptions to the `reg` event of both the P-CSCF and the IMS terminal. Messages 1–20 in Figure 5.16 have already been described previously when dealing with Figure 5.4.

Figure 5.17 shows the contents of the SUBSCRIBE request (27) that the IMS terminal sends to the user's registration state. You may notice that the *Request-URI* contains the Public User Identity that was just registered in the previous REGISTER requests. The IMS terminal sends this SUBSCRIBE request to its P-CSCF. The P-CSCF honors

⁸Although the P-CSCF does not keep a registration state that relates to SIP registrations, it keeps states that relate to the compression of signaling (SigComp) and the establishment of security associations.

⁹Note that the SLF, which is required if there is more than one HSS in the home network, is not shown.

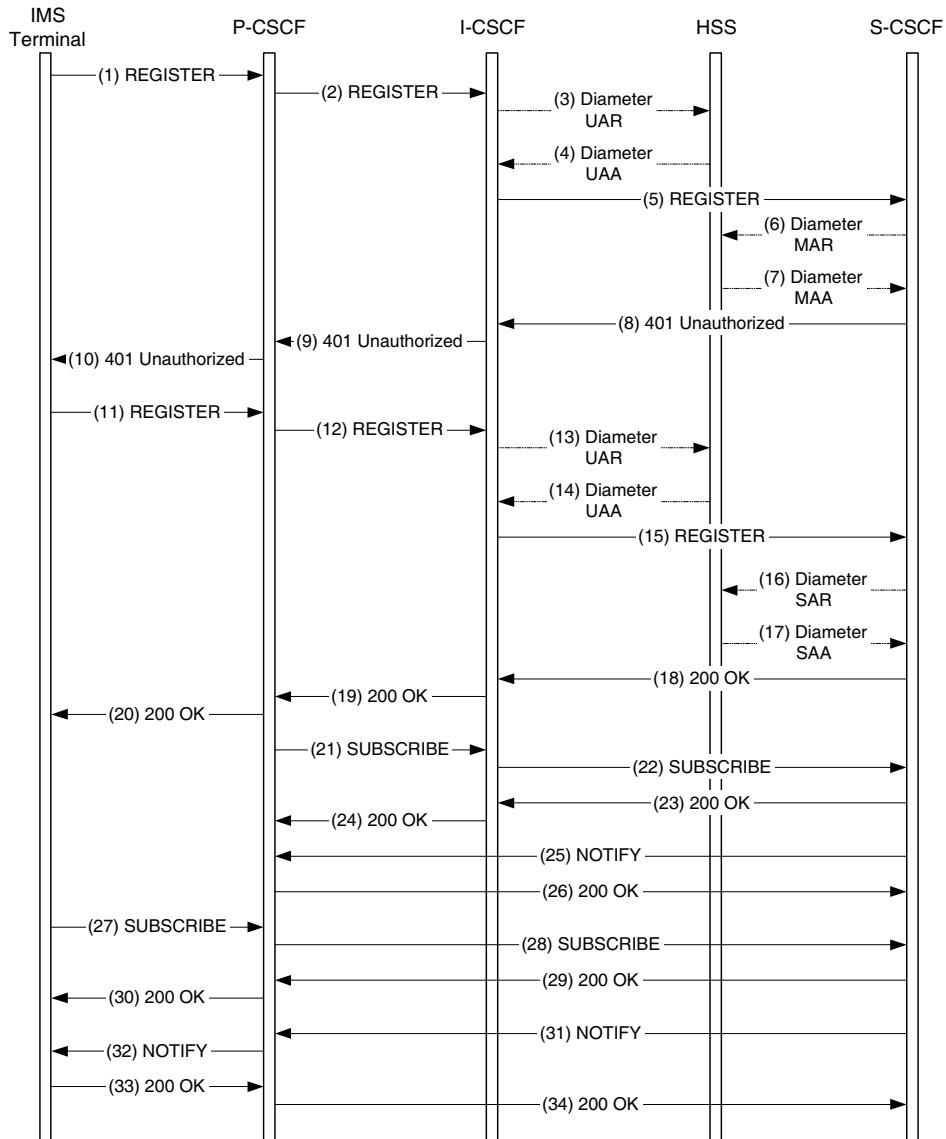


Figure 5.16: Complete registration flow in the IMS, including subscription to the `reg` event state

```

SUBSCRIBE sip:alice@home1.net SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Max-Forwards: 70
Route: <sip:pcscf1.visited1.net:5058;lr;comp=sigcomp>,
        <sip:orig@scscf1.home1.net;lr>
P-Preferred-Identity: "Alice Bell" <sip:alice@home1.net>
Privacy: none
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                      utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=d9211
To: <sip:alice@home1.net>
Call-ID: b89rjhnedlrfjfslsj40a222
Require: sec-agree
Proxy-Require: sec-agree
Cseq: 61 SUBSCRIBE
Event: reg
Expires: 600000
Accept: application/reginfo+xml
Security-Verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96;
                  spi-c=98765432; spi-s=909767;
                  port-c=5057; port-s=5058
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
Content-Length: 0

```

Figure 5.17: (27) SUBSCRIBE

the Route header field and proxies the request to the S-CSCF, (28) in Figure 5.16. The S-CSCF acts as a SIP notifier and, as it is also a registrar for the Public User Identity of interest, accepts the subscription by sending a 200 (OK) response (29). The P-CSCF forwards the response (30) to the IMS terminal. In addition, the S-CSCF sends a NOTIFY request (31) that contains a well-formed and valid XML document that contains registration information. The P-CSCF forwards the NOTIFY request to the IMS terminal. Figure 5.18 shows an example of this NOTIFY request (32), as received at the IMS terminal. If we focus on the XML registration information document, we observe that there is a set of Public User Identities allocated to this user. In this case, these Public User Identities are `sip:alice@home1.net`, `sip:alice-family@home1.net` and `sip:+12125551234@home1.net;user=phone`. They are indicated in the `aor` attribute of the `registration` element. The S-CSCF also indicates in the `state` attribute that all of these Public User Identities are active. The contact address of each Public User Identity is also supplied. In this example, each contact element contains a `uri` element that includes the address of the different identities that point to the same SIP URI. This URI contains the IP address allocated to the IMS terminal. An important piece of information is included in the `event` attribute of the contact address. The `event` attribute indicates which event triggered the change in state. In this example the S-CSCF indicates that a SIP registration made the first Public User Identity change to active, because the `event` attribute of the Public User Identity of `sip:alice@home1.net` was set to `registered`. However, the `event` attribute

```

NOTIFY sip:[1080::8:800:200C:417A]:5059;comp=sigcomp SIP/2.0
Via: SIP/2.0/UDP pcscf1.visited1.net:5058;comp=sigcomp;
      branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP scscf1.home1.net;branch=z9hG4bKs1pp0
Max-Forwards: 69
Route: <sip:pcscf1.home1.net;lr>
From: <sip:alice@home1.net>;tag=d9211
To: <sip:alice@home1.net>;tag=151170
Call-ID: b89rjhnedlrfjfslj40a222
Cseq: 42 NOTIFY
Subscription-State: active;expires=600000
Event: reg
Content-Type: application/reginfo+xml
Contact: <sip:scscf1.home1.net>
Content-Length: 873

<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo"
          version="1" state="full">
    <registration aor="sip:alice@home1.net"
                  id="11a" state="active">
        <contact id="542" state="active" event="registered"
                  duration-registered="1" expires="599999">
            <uri>sip:[1080::8:800:200C:417A]:5059;comp=sigcomp</uri>
        </contact>
    </registration>
    <registration aor="sip:alice-family@home1.net"
                  id="11b" state="active">
        <contact id="543" state="active" event="created"
                  duration-registered="1" expires="599999">
            <uri>sip:[1080::8:800:200C:417A]:5059;comp=sigcomp</uri>
        </contact>
    </registration>
    <registration aor="sip:+12125551234;user=phone"
                  id="11c" state="active">
        <contact id="544" state="active" event="created"
                  duration-registered="1" expires="599999">
            <uri>sip:[1080::8:800:200C:417A]:5059;comp=sigcomp</uri>
        </contact>
    </registration>
</reginfo>

```

Figure 5.18: (30) NOTIFY

of the other two Public User Identities was set to `created`, indicating that these two Public User Identities were created administratively. These two administratively created Public User Identities form what is known as the *set of implicitly registered Public User Identities*. This is an interesting feature of IMS that allows a user to configure his registration so that whenever one Public User Identity is registered, other Public User Identities are automatically registered. The same applies for deregistration: if a Public User Identity is deregistered, the set of implicitly registered Public User Identities are automatically deregistered. This feature can be used to speed up the registration process, as now this process is independent of the number of identities allocated to the user. In addition, any other identities that the operator has reserved for the user's disposal but are not registered are also signaled in the "reg" event document. This allows the user to learn other allocated but not registered Public User Identities.

5.7 Basic Session Setup

This section explores the process of establishing a basic session in the IMS. For the sake of clarity we will assume that an IMS terminal is establishing a session toward another IMS terminal. As both terminals are IMS terminals they will support the same sort of capabilities.

For the sake of simplicity, we assume that neither the calling party nor the called party have any services associated with the session. We describe in Section 5.8.3 how an Application Server is involved and how Application Servers can provide valuable services to the user.

Figures 5.19 and 5.20 are description flow charts of the signaling involved in a basic session setup. At first glance the reader may think that there are a lot of SIP messages flowing around. In addition, there are many nodes involved in setting up the session. The evaluation is probably right; SIP is a rich protocol in its expression at the cost of a high number of messages.

However, let us focus on Figures 5.19 and 5.20. We are assuming that both users are roaming to a network outside their respective home networks, such as when both users are outside their respective countries. This leads to having two different visited networks in the figures. We also assume that each of the users has a different business relationship with his respective operator; therefore, there are two different home networks in the figures. In addition, we assume that the P-CSCF is located in a visited network. When we consider all the roaming/non-roaming scenarios, different home networks, etc., this is the most complete and complicated case. Once the reader is familiar with this scenario, variations of it are just subsets or simplifications of it.

For the sake of clarity we refer to the *originating P-CSCF* and *originating S-CSCF* as the P-CSCF and S-CSCF that are serving the caller. Similarly, we refer to the *terminating P-CSCF* and *terminating S-CSCF* as the P-CSCF and S-CSCF that are serving the callee.

The first thing the reader might have noticed is the complete separation of the signaling and media planes. Signaling (i.e., SIP) traverses a set of CSCFs, whereas media is sent end-to-end (i.e., from an IMS terminal to another IMS terminal), only traversing IP routers and, if applicable, GGSNs.

Another observation is that all SIP signaling traverses both the originating P-CSCF and the originating S-CSCF in all circumstances. This is a significant difference with respect to other cellular networks, where, if the user is roaming, signaling does not traverse the home network. The P-CSCF must be present in all the signaling exchanged with the terminal because it compresses/decompresses SIP in the interface toward the terminal.

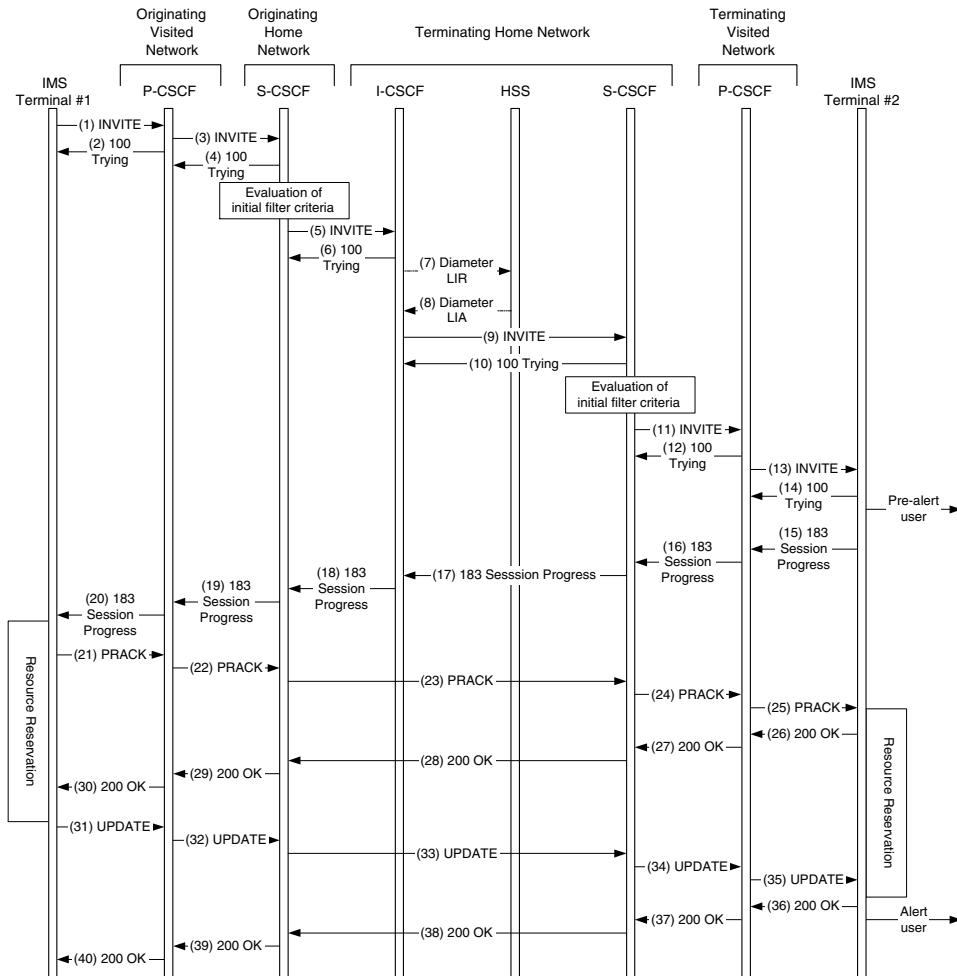


Figure 5.19: Basic session setup (part 1)

The S-CSCF is traversed in all requests to allow the triggering of services that the user may have requested. The S-CSCF plays an important role in service provision by involving one or more Application Servers that implement the service logic. Since the S-CSCF is always located in the home network, services are always available to the user regardless of whether the user is roaming or not. We describe the triggering of services in Section 5.8.3. Note, however, that traversing the CSCFs only affects the signaling plane. It does not affect, in principle, the media plane, which is transmitted end-to-end.

If we continue with the examination of the figures, we discover that the flow follows the model described in “Integration of resource management and Session Initiation Protocol (SIP)” (RFC 3312 [103]), sometimes known as *preconditions*. We have already explored the model in Section 4.14.

While the use of preconditions was mandatory in 3GPP Release 5, Release 6 allows session establishment with no preconditions when the remote terminal does not support them

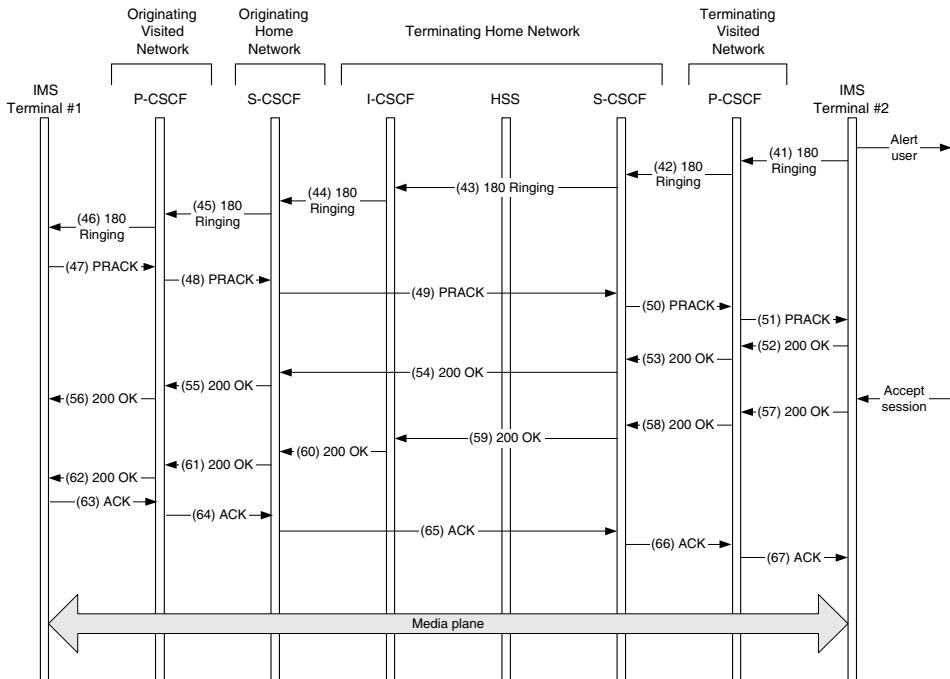


Figure 5.20: Basic session setup (part 2)

or when a particular service does not require them. Push-to-talk over Cellular (PoC), which is described in Chapter 25, is an example of a service that does not use preconditions.

For regular sessions between two IMS terminals, the requirement to use the preconditions call-flow model comes from the fact that in cellular networks, radio resources for the media plane are not always available.¹⁰ If the preconditions extension was not used, when the called party accepted the call, the radio bearers at both the calling party and called party would not be ready. This means that the terminals would not be able to transmit and receive media-plane traffic (e.g., Real-Time Transport Protocol). As a consequence, the first words the callee says may be lost.

If we take a look at Figure 5.19 we can see a Diameter interaction between the I-CSCF and the HSS in the terminating network. This is required because the I-CSCF needs to discover the address of the S-CSCF serving the destination user. However, it must be noted that there is no interaction between the HSS and any other node in the originating network. In other words, the flow is asymmetric, considering the originating and terminating sides of it. The advantage is that the HSS is relieved of involvement in extra messages that are not required in the originating call leg.¹¹

By examining Figures 5.19 and 5.20, we can see that all SIP signaling traverses the originating and terminating P-CSCF and S-CSCF nodes (four nodes in total). This requires

¹⁰It must be noted that radio resources for the signaling plane are always allocated, so that sessions and other types of SIP signaling can be addressed to the IMS terminal. However, the situation is not the same for the radio resources that the media plane will use.

¹¹Note that the HSS in typical implementations is a high-capacity node, serving hundreds of thousands of users. Saving a couple of messages on each session attempt is a significant saving in capacity at the HSS.

that each of these nodes inserts a Record-Route header field that contains the SIP URI of the node. This guarantees that future signaling (e.g., a PRACK or BYE request) will visit that node. However, the I-CSCF in the terminating network plays a different role and may or may not insert a Record-Route header field pointing to itself. This means that the INVITE request and all its responses (both provisional and the final one) will visit the I-CSCF, but other requests such as PRACK, ACK, or BYE will not visit the I-CSCF.

5.7.1 The IMS Terminal Sends an INVITE Request

Figure 5.21 shows an example of the INVITE request (1) that the IMS terminal sends to the network. We will devote a few pages to this INVITE request and analyze it in depth.

The *Request-URI* contains the Public User Identity of the intended destination. In this case the *Request-URI* points to Bob's identity, which belongs to an operator known as home2.net.

Figure 5.22 reproduces the contents of the *Via* header field displayed in Figure 5.21. The *Via* header field contains the IP address and port number where the IMS terminal is supposed to receive the responses to the INVITE request. The port number is of importance because it is the port number bound to the security association that the P-CSCF has established toward the IMS terminal (we describe security associations in detail in Section 12.1.2). The P-CSCF will forward responses to the INVITE request to the IP address and port number declared in the *Via* header field. If the IMS terminal fails to include a port number bound to the security association at the P-CSCF, the IPsec layer at the P-CSCF will discard the packet. In addition, the header also indicates the willingness of IMS terminals to receive compressed signaling using signaling compression (SigComp, RFC 3320 [258]). The contents of the *Contact* header field are quite similar, as they indicate the SIP URI where the IMS terminal is willing to receive subsequent requests belonging the same SIP dialog as the INVITE request.

The *Via* header field also indicates the transport protocol that is used to transport SIP messages to the next hop. SIP supports any transport protocol (e.g. UDP, TCP, or SCTP). So, every node is free to choose the more appropriate transport protocol according to the guidelines given in RFC 3261 [286].

The reader can observe that there is a preloaded *Route* header field (Figure 5.23). The value of the *Route* header field points to the P-CSCF in the visited network and the S-CSCF in the home network. Typically, in SIP, the *Route* header fields are learnt during the initial INVITE transaction. In normal circumstances the SIP UA obtains the route out of the *Record-Route* header field value, and the set of proxies to traverse will be operational for subsequent transactions that belong to the dialog created with the INVITE request.

In the IMS, the first messages need to traverse a set of SIP proxies, which serve the IMS terminal. The SIP UA needs to create a list of proxies to be traversed, and place this list into the *Route* header field. This set of proxies is created from the concatenation of the outbound SIP proxy, learnt during the P-CSCF discovery procedure (see Section 5.4), and the value of the *Service-Route* header field received in the 200 (OK) response for a REGISTER request. We already saw in Figure 5.10 what this 200 (OK) response looks like. In our example the *Service-Route* header field contained a single node to visit, the S-CSCF in the home network. So, when the IMS terminal creates a request other than a subsequent request within an existing dialog, it inserts the preloaded *Route* header field pointing to the mentioned nodes.

If we continue with the examination of the relevant header fields in Figure 5.21, we can observe the presence of a *P-Preferred-Identity* header field with the value set to

```

INVITE sip:bob@home2.net SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;
      comp=sigcomp;branch=z9hG4bK9h9ab
Max-Forwards: 70
Route: <sip:pcscf1.visited1.net:5058;lr;comp=sigcomp>,
        <sip:orig@scscf1.home1.net;lr>
P-Preferred-Identity: "Alice Smith" <sip:alice@home1.net>
Privacy: none
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                        utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>
Call-ID: 3s09cs03
Cseq: 127 INVITE
Require: sec-agree
Proxy-Require: sec-agree
Supported: 100rel, precondition
Security-Verify: ipsec-3gpp; q=0.1;
                  alg=hmac-sha-1-96;
                  spi-c=98765432; spi-s=909767;
                  port-c=5057; port-s=5058
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Content-Type: application/sdp
Content-Length: 591

v=0
o=- 1073055600 1073055600 IN IP6 1080::8:800:200C:417A
s=-
c=IN IP6 1080::8:800:200C:417A
t=0 0
m=video 8382 RTP/AVP 98 99
b=AS:75
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=inactive
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
a=rtpmap:99 MP4V-ES
m=audio 8283 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=inactive
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.21: (1) INVITE

```
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;
comp=sigcomp;branch=z9hG4bK9h9ab
```

Figure 5.22: The Via header field

```
Route: <sip:pcscf1.visited1.net:5058;lr;comp=sigcomp>,
<sip:orig@scscf1.home1.net;lr>
```

Figure 5.23: The Route header field

Alice Smith and a SIP URI. This header field is a SIP extension (which is specified in the “Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks”, RFC 3325 [194]). The reason for this header field is that a user may have several Public User Identities. When the user initiates a session, they need to indicate which one of their identities should be used for this session. The user is identified with this identity in the network charging records and, if applicable, at the callee. In addition, the identity is also used to trigger services; so, different identities may trigger different services. The user chooses a preferred identity and inserts the value in the P-Preferred-Identity header field of the INVITE request. When the P-CSCF receives the INVITE request, it verifies that the INVITE request was received within an already established security association with the terminal that registered the identity of the P-Preferred-Identity. The P-CSCF also verifies the possible identities that the user can use in requests received through that security association. Provided that all the verifications are correct, the P-CSCF, when forwarding the INVITE request toward the home network, replaces the P-Preferred-Identity with a P-Asserted-Identity header field that contains the same value. If the P-CSCF considers that the preferred identity is not allocated to the user or is not registered to the IMS network, the P-CSCF removes the P-Preferred-Identity header field and inserts a P-Asserted-Identity header field that contains a value the P-CSCF considers to be one of the valid identities of the user. In addition, if the user did not indicate a P-Preferred-Identity header field, then the P-CSCF chooses a default identity and inserts it in a P-Asserted-identity header field. In any case the SIP request that the P-CSCF forwards always includes a P-Asserted-Identity header field that includes an authenticated Public User Identity value.

The next header field that we see in Figure 5.21 is the P-Access-Network-Info header field. This header field is an extension created by the IETF as a consequence of 3GPP requirements (this header field is documented in the “Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)”, RFC 3455 [154]). Figure 5.24 shows an example of its utilization. The header field provides two types of access information.

- (1) *The type of layer 2/3 technology used by the IMS terminal.* In our example, the IMS terminal indicates that is using the UTRAN (UMTS Terrestrial Radio Access Network). This information may be valuable to Application Servers, as they may use it to customize the service, depending on the characteristics of the access network. The available bandwidth to the terminal is also determined by the type of access network. For instance, a wireless LAN access typically provides higher rates than the GERAN (GSM/Edge Radio Access Network).

```
P-Access-Network-Info: 3GPP-UTRAN-TDD;
utran-cell-id-3gpp=24450289A3299239
```

Figure 5.24: The P-Access-Network-Info header field

- (2) *The identity of the radio cell where the IMS terminal is connected.* This parameter is optional, as it may or may not be available in the SIP application layer. But, when the SIP application layer can read the cell identity from the lower radio layers, the IMS terminal will insert it. The base-station transceiver broadcasts its cell identity over the radio channels, and the radio layer at the terminal stores the last received cell identity. The IMS application reads that information and sticks it in the P-Access-Network-Info header field. On the other hand, the cell ID implicitly contains some rough location information that may be used to provide a personalized service to the user (e.g., an announcement giving a list of the closest Italian restaurants in the same location area).

Because of the sensitive private information contained in the P-Access-Network-Info header field the IMS terminal only inserts the header field when a security association towards the P-CSCF is already in place. Furthermore, the header field is transmitted from the IMS terminal, via the visited network, until the last hop in the home network, but it is never transferred to the callee's home network. This scheme allows Application Servers in the home network to receive the header field when they process a SIP message. Referring to Figure 5.19, the INVITE request in steps (1)–(4) contains the header field, but the INVITE request in step (5) does not contain the header field, as this INVITE request is destined for the callee's home network.

The Privacy header field in Figure 5.21 shows the willingness of the user to reveal certain privacy information to non-trusted parties (such as the called party). In this example the user does not have any requirement regarding the privacy of the session. In other cases the user may have indicated that they do not want to reveal the contents of some header fields to nontrusted parties. One of these header fields is the P-Asserted-Identity, which, as we previously described, is inserted not by the IMS terminal but by the P-CSCF.

The From, To, Call-ID, and CSeq header fields are set according to the SIP procedures described in the SIP core specification, RFC 3261 [286]. It must be noted that these header fields transport end-to-end information and that network nodes do not assert, inspect, or take any action on the value of these header fields. It is especially important to highlight that the From header field is not policed, so the IMS terminal can insert any value there, including a SIP URI that does not belong to the user. For the purpose of identifying the user the P-Asserted-Identity header field is used instead.

The Require, Proxy-Require, and Supported header fields are shown in Figure 5.25. These header fields include those values that declare the mandatory or optional requirements of certain capabilities or SIP extensions. The mandatory capabilities are required in the Require and Proxy-Require header fields, whereas the optional capabilities are declared in the Supported header field. In the example in Figure 5.25 the SIP security agreement sec-agree is the mandatory capability that has to be supported, whereas precondition and the reliability of provisional response 100rel extensions are the optional capabilities. We describe the SIP security agreement in Chapter 12.

The precondition value in the Supported header field is an indication to the remote terminal that the SIP preconditions extension can be used, if required by the callee. All IMS

```

Require: sec-agree
Proxy-Require: sec-agree
Supported: 100rel, precondition

```

Figure 5.25: The Require, Proxy-Require, and Supported header fields

terminals implement this extension, which determines the call flow. In particular, because the preconditions extension is used, the call flow contains 183 (Session Progress) responses, PRACK, and UPDATE requests. We described the generalities of the preconditions extension in Section 4.14.

In the IMS we are interested in the quality of service aspects of this extension. The problem in cellular networks is that resources are not preallocated before the usage is needed. So, in most cases, both IMS terminals do not have reserved resources prior to the session establishment, so each terminal has to reserve resources during the session establishment, because resource reservation typically requires knowledge of the bandwidth of the media (which is determined by the agreed codec) and the IP address and port number of the remote endpoint. However, there are cases where one or both IMS terminals are not using an IP-CAN that requires resource reservation (WLAN is typically one of those IP-CANs), in which case the flow is simplified. These simplified cases are described in Section 5.12.

Let us come back to the general case, in which both IMS terminals need to reserve resources. The calling party cannot freely request the network to establish a session comprising high-bandwidth audio and video if, at the end of the SIP/SDP negotiation, the remote party just supports a low-bandwidth codec or, even worse, declines the session establishment. The preconditions extension allows the terminals to have a first exchange of SDP in an INVITE request and a 183 (Session Progress) response. After that, both parties are aware of the capabilities and willingness of the remote party to establish a particular set of media streams with a particular set of codecs, as well as the remote IP address and port number to send the media streams to. This is the moment when the IMS terminal can request a guaranteed quality of service from the network to establish the appropriate media streams.

The preconditions extension defines two models of operation with respect to quality of service. In the IMS the so-called local segmentation model for quality of service reservation is used. Each of the terminals is responsible for maintaining the appropriate resource reservation in its respective local segment. In the case of a GPRS access network, the local segment is defined between the IMS terminal and the GGSN. A particular quality of service is requested using the PDP context activation procedure.

The Supported header field indicates all those SIP extensions that the IMS terminal supports and are appropriate for the invited party to use in the response. In the example we saw that, in addition to the precondition extension, the IMS terminal declares its support for the reliable provisional responses extension 100rel. All IMS terminals support this extension, which, in any case, is already required by the preconditions extension.

The Security-Verify header field is related to security features of the signaling. We describe this header field in detail in Section 12.1.2 together with other security features of the IMS.

The Contact header field in Figure 5.26 indicates the SIP URI where the IMS terminal wants to receive further SIP requests pertaining to the same dialog. Typically, the IMS terminal is not aware of its own host name (unless it does a reverse DNS query to find its own host name), and it is really not required. So, this URI typically contains an IP address

rather than a host name. The port number that follows the IP address refers to the port number bound to the security association that the P-CSCF has established toward the IMS terminal. If the IMS terminal fails to indicate a port number that falls into the security association that the P-CSCF has established toward the IMS terminal, the IPsec layer in the P-CSCF discards packets and the IMS terminal does not receive the desired SIP signaling. In addition, the URI declares the willingness to provide signaling compression, because of the inclusion of the `comp=sigcomp` parameter.

```
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
```

Figure 5.26: The Contact header field

The next header field the IMS terminal includes in the INVITE request in Figure 5.21 is the `Allow` header field, reproduced in Figure 5.27. The `Allow` header field is optional in INVITE requests, although highly recommended, as it gives a hint to the peer terminal about the SIP methods the IMS terminal supports. In our example, in Figure 5.27 the IMS terminal declares its support for the core SIP methods (e.g., INVITE, ACK, CANCEL, BYE), the methods supported by the precondition extension (RFC 3312 [103]) (e.g., PRACK, UPDATE), the REFER method (RFC 3515 [306]), and the MESSAGE method (RFC 3428 [115]). When the session is established, the peer terminal grants support for REFER or MESSAGE methods that, otherwise, could or could not be supported, as they are extensions to the SIP core protocol.

```
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
```

Figure 5.27: The Allow header field

`Content-Type` and `Content-Length` are the last two header fields in Figure 5.21. The values of these header fields depend on the type of body included in the INVITE request. SDP (Session Description Protocol), whose core protocol is specified in RFC 2327 [160]¹² is the only protocol used in the IMS to describe sessions. Furthermore, although it is possible for INVITE requests not to include any body at all, IMS-compliant terminals, when creating INVITE requests, always insert an SDP offer describing the session they are trying to establish.

Figure 5.28 reproduces the `Content-Type` and `Content-Length` header fields. These headers indicate that the accompanying body is an SDP body of a certain length. If we take a look at the SDP body, we can see a number of lines that indicate the type of session that the IMS terminal wants to create. In particular, the `c=` line indicates the IP address where the terminal wants to receive media streams. In this example the IMS terminal wants to establish two media streams, indicated by the presence of two `m=` lines: one is an audio stream and the other is a video stream. It is worth noting that IMS terminals always include a `b=` line per audio or video media stream, indicating the bandwidth requirements for that particular media stream. The `b=` line is otherwise optional in SDP, but IMS terminals include it to give a hint to the network nodes that need to control resources (such as bandwidth allocation).

The `m=` lines in SDP include the port number where the IMS terminal wants to receive the media stream and a list of codecs that the terminal is willing to support for this media

¹²There is work in progress in the IETF to revise SDP, RFC 2327, into a new document that will contain some clarifications about the protocol, support for IPv6, etc. The document will be allocated a new RFC number.

```

Content-Type: application/sdp
Content-Length: 567

v=0
o=- 1073055600 1073055600 IN IP6 1080::8:800:200C:417A
s=- 
c=IN IP6 1080::8:800:200C:417A
t=0 0
m=video 8382 RTP/AVP 98 99
b=AS:75
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
a=rtpmap:99 MP4V-ES
m=audio 8283 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.28: The Content-Type and Content-Length header fields and the SDP body

stream and for this session. The first `m=` line in Figure 5.28 indicates the desire to establish a video stream and it includes the supported and desired video codecs for this session. In this example the IMS terminal indicates support for the H.263 [183] and MPEG-4 Visual [174] codecs. The H.263 [183] baseline is the mandatory codec for 3GPP IMS terminals. 3GPP also recommends IMS terminals to implement H.263 Version 2 Interactive and Streaming Wireless Profile (Profile 3) Level 10 [188] and MPEG-4 Visual Simple Profile at Level 0 [174]. Other codecs may be supported at the discretion of the IMS terminal manufacturer. 3GPP2 supports a different set of codecs that include EVRC (Enhanced Variable Rate Codec) and SMV (Selectable Mode Vocoder).

The second `m=` line in Figure 5.28 indicates the wish to establish an audio stream and it includes the supported and desired audio codecs for this session. In this case the IMS terminal has indicated support for the AMR narrowband speech codec (specified in 3GPP TS 26.071 [7]) and the telephone-event RTP payload format (used to transmit key presses during a session), as specified in RFC 2833 [303]. Both the AMR and the telephone-event RTP payload format are mandatory audio codecs for a 3GPP IMS terminal. If the IMS terminal supports wideband speech codecs, the mandatory codec to implement is the AMR wideband codec, as specified in 3GPP TS 26.171 [5], working at a 16 kHz sampling frequency.

We also observe that the SDP in Figure 5.28 indicates the use of the precondition extension, observable through the presence of a few `a=curr:qos` and `a=des:qos` lines directly below each `m=` line. These attributes indicate the current and desired local quality of service bearers for the respective media streams. Typically the IMS terminal has not established a radio bearer to send and receive the media streams when it sends the INVITE request; therefore, the quality of service is set to none. In addition, each of the media streams also set the stream in `inactive` mode, as a mechanism to indicate that resources have not been reserved yet, and thus the other party should not start answer with a 200-class response and start sending media all the way. At a later point, when resources have been reserved, the media streams will be marked as `sendrecv` (or not marked at all, which defaults to `sendrecv`), indicating that media can be sent bi-directionally.

This concludes the examination of the INVITE request that the IMS terminals sends out. But what happens to that request? How does it reach the peer terminal? What other transformations occur en route to its final destination? We analyze them in the next section.

5.7.2 *The Originating P-CSCF Processes the INVITE Request*

So far we have analyzed all the SIP header fields and SDP that the IMS terminal populates. When this INVITE is ready the terminal sends it to the P-CSCF discovered during the P-CSCF discovery procedures. The P-CSCF receives the INVITE request through an established security association (we describe security associations in Section 12.1.2). Assuming that the security functions are correctly performed, the P-CSCF takes a few actions that we describe in the following paragraphs.

First, the P-CSCF verifies that the IMS terminal is acting correctly, according to IMS routing requirements. For instance, the P-CSCF verifies that the `Route` header is correctly populated; in other words, it contains the contents of the `Service-Route` header field that the IMS terminal received in a 200 (OK) response to a REGISTER request (during the last registration procedure). This guarantees that the SIP signaling will traverse the S-CSCF in the home network, as requested during registration in the above-mentioned `Service-Route` header. If the P-CSCF considers that the `Route` header field does not include the value of the `Service-Route` header field received during the registration process, either the P-CSCF replaces the `Route` header according to what the P-CSCF considers to be correct, or the P-CSCF answers the INVITE request with a 400 (Bad Request) response and does not forward the INVITE request.

The P-CSCF inspects the SDP offer. The P-CSCF is configured with the local policy of the network where the P-CSCF is operating. Such a policy may indicate that some media parameters are not allowed in the network. For instance, the policy may indicate that G.711 codecs are not allowed. This is likely to be prohibited, since G.711 requires a bandwidth of 64 kb/s, whereas AMR requires only a maximum of 14 kb/s. Basically, if an IMS terminal uses G.711 as an audio codec, then it is using a radio spectrum that could potentially be used by four users. As radio is a scarce resource, and since AMR is mandated in all the 3GPP IMS implementations, it sounds natural that operators will prevent IMS terminal misuse of such expensive resources. The local policy is dependent on each operator's needs, network topology, charging models, roaming agreements, etc.

If the P-CSCF, when policing the SDP, discovers a media parameter that does not fit in the current local policy, the P-CSCF generates a 488 (Not Acceptable Here) response containing an SDP body, and it does not forward the INVITE request. The SDP body included in the

488 response indicates the media types, codecs, and other SDP parameters which are allowed according to local policy.

Then, the P-CSCF looks to see whether a P-Preferred-Identity header is included in the INVITE request. If it is, as in our example in Figure 5.21, then the P-CSCF verifies whether the value in that header corresponds to one of the implicitly or explicitly registered Public User Identities. The P-CSCF is aware of the security association related to this request, so only that IMS terminal, and not an impersonating agent, is at the other side of the security association sending the INVITE request. The P-CSCF learnt, during the registration procedure (see Section 5.6 for a complete description of the subscription to the reg state), all the Public User Identities registered to the user. The P-CSCF deletes the P-Preferred-Identity header in the INVITE request and inserts a P-Asserted-Identity header following the procedures specified in RFC 3325 [194]. The value of the P-Asserted-Identity header field is set to a valid SIP registered Public User Identity of the user; for example, the same value of the P-Preferred-Identity header field or any other registered Public User Identity of that user, if the one included in the P-Preferred-Identity header field is not registered or the P-Preferred-Identity header field is not present in the request. It must be noted that, in asserting the identity of the user, the P-CSCF never takes into account the From header field. The From header field is considered an end-to-end header.

Once this is done, the P-CSCF removes and modifies the headers that relate to the security agreement. These procedures are described in Section 12.1.2. The P-CSCF also inserts charging headers. This procedure is described in detail in Chapter 7.

The P-CSCF always records the route, because all the SIP requests and responses have to traverse the P-CSCF, so that the P-CSCF can apply compression, security, and policing, as mentioned previously. When a SIP proxy wants to remain in the path for a subsequent exchange of requests pertaining to the same SIP dialog, it inserts a Record-Route header field with its own SIP URI as a value. And this is exactly what the P-CSCF does: it inserts a Record-Route header field that contains the P-CSCF's SIP URI. The reader must note that this Record-Route header does not contain the comp=sigcomp parameter or the ports derived from the security associations. This is because the INVITE request is transmitted toward the core network, where there is no need for SIP compression or IPsec connections.

In addition to recording the route the P-CSCF modifies the Via, Route, and Max-Forwards header field values as per regular SIP procedures.

Figure 5.29 shows the INVITE request that the P-CSCF forwards to the S-CSCF. This INVITE corresponds to step (3) in Figure 5.19. The P-CSCF forwards the request to the next hop included in the Route header. In this example, that node is an S-CSCF, but it could have been an I-CSCF in the home network, if that was decided by the home network operator.

5.7.3 The Originating S-CSCF Processes the INVITE Request

The S-CSCF allocated to the caller receives the INVITE request and examines the P-Asserted-Identity header to identify the user who originated the INVITE request. The S-CSCF downloaded the user profile at registration. Among other information, the user profile contains the so-called filter criteria. The filter criteria contain the collection of triggers that determine whether a request has to traverse one or more Application Servers that provide services to the user. It must be noted that the S-CSCF does not execute services, but triggers services to be executed by Application Servers.

```

INVITE sip:bob@home2.net SIP/2.0
Via: SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;
      comp=sigcomp;branch=z9hG4bK9h9ab
Max-Forwards: 69
Route: <sip:orig@scscf1.home1.net;lr>
Record-Route: <sip:pcscf1.visited1.net;lr>
P-Asserted-Identity: "Alice Smith" <sip:alice@home1.net>
Privacy: none
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                           utran-cell-id-3gpp=24450289A3299239
P-Charging-Vector: icid-value="W34h6dlg"
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>
Call-ID: 3s09cs03
Cseq: 127 INVITE
Supported: 100rel, precondition
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Content-Type: application/sdp
Content-Length: 591

v=0
o=- 1073055600 1073055600 IN IP6 1080::8:800:200C:417A
s=-
c=IN IP6 1080::8:800:200C:417A
t=0 0
m=video 8382 RTP/AVP 98 99
b=AS:75
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=inactive
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
a=rtpmap:99 MP4V-ES
m=audio 8283 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=inactive
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.29: (3) INVITE

The S-CSCF always evaluates the initial filter criteria in initial SIP requests.¹³ For instance, the S-CSCF evaluates the filter criteria when it receives an initial INVITE or SUBSCRIBE request, because INVITE and SUBSCRIBE requests create a dialog. The S-CSCF also evaluates the filter criteria when it receives OPTIONS or MESSAGE requests outside any other dialog (they can be sent within or outside an existing dialog). However, the S-CSCF does not evaluate the filter criteria when it receives a PRACK or UPDATE request, as this is a subsequent request sent within a SIP dialog.

Having said that, we will not stop now to find out how filter criteria work. We assume for the time being that our user, named `sip:alice@home1.net`, does not have any services associated with the origination of a SIP session. We will revisit this assumption in Section 5.8.4 when we describe how filter criteria work and how Application Servers are involved in the signaling path.

The S-CSCF, like the P-CSCF, polices SDP. Like the P-CSCF, the S-CSCF polices those SDP media parameters that are not set according to local policy. Unlike the P-CSCF the S-CSCF uses the HSS user-related information (the user profile). The S-CSCF also polices the SDP for user-related information. This allows operators, for instance, to offer cheap subscriptions that do not allow the use of certain media streams (e.g., video) or certain premium codecs. If the request does not fit with the policy, the S-CSCF may not process the INVITE request and may answer it with a 488 (Not Acceptable Here) response indicating the media types, codecs, and other SDP parameters which are allowed according to the policy.

The S-CSCF of the originating user is the first node that tries to route the SIP request based on the destination (the callee) in the *Request-URI*. The originating P-CSCF (and I-CSCF if it was traversed) were never interested in the destination of the session setup and they did not look at the *Request-URI* field of the request. However, the S-CSCF in the originating home network is the first node that takes a look at the destination (*Request-URI*). In doing so the S-CSCF may find two different types of contents for the *Request-URI*: a SIP URI or a TEL URI.

If the S-CSCF finds a SIP URI in the *Request-URI*, then regular SIP routing procedures apply. Basically, given a domain name, the S-CSCF has to find a SIP server in that domain name. The procedures are normatively described in RFC 3263 [285]. In our example the *Request-URI* is set to `sip:bob@home2.net` and the S-CSCF has to find a SIP server (typically an I-CSCF) in the network named *home2.net*. The S-CSCF extracts the domain name *home2.net* and does a number of DNS queries, as shown in Figure 5.30. The first DNS query (1) tries to find the transport protocols supported by SIP services in *home2.net*. The result (2) reveals that *home2.net* is implementing SIP services on both TCP and UDP. Then the S-CSCF, as it is interested in UDP services, queries the DNS requesting information about SIP services over UDP, hence the `_sip._udp.home2.net` query (3). The result (4) returns the host names and port numbers of two different SIP servers, *icscf1* and *icscf2*. Having determined the supported transport protocol host name and port number, the S-CSCF queries the DNS to find the IP address of that SIP proxy (5). The result is returned in step (6). At this stage the S-CSCF has learnt all the information required to build the INVITE request and to forward it to the appropriate I-CSCF at the terminating home network.

Some may think that so many queries to the DNS will have a severe impact on the session setup time. DNS offers a way around this problem. On one side, DNS typically tries to guess future queries based on the responses offered by DNS. If the information is available,

¹³In this context we refer to *initial requests* as those SIP requests that either create a dialog or are stand-alone requests (i.e., those that are not subsequent requests).

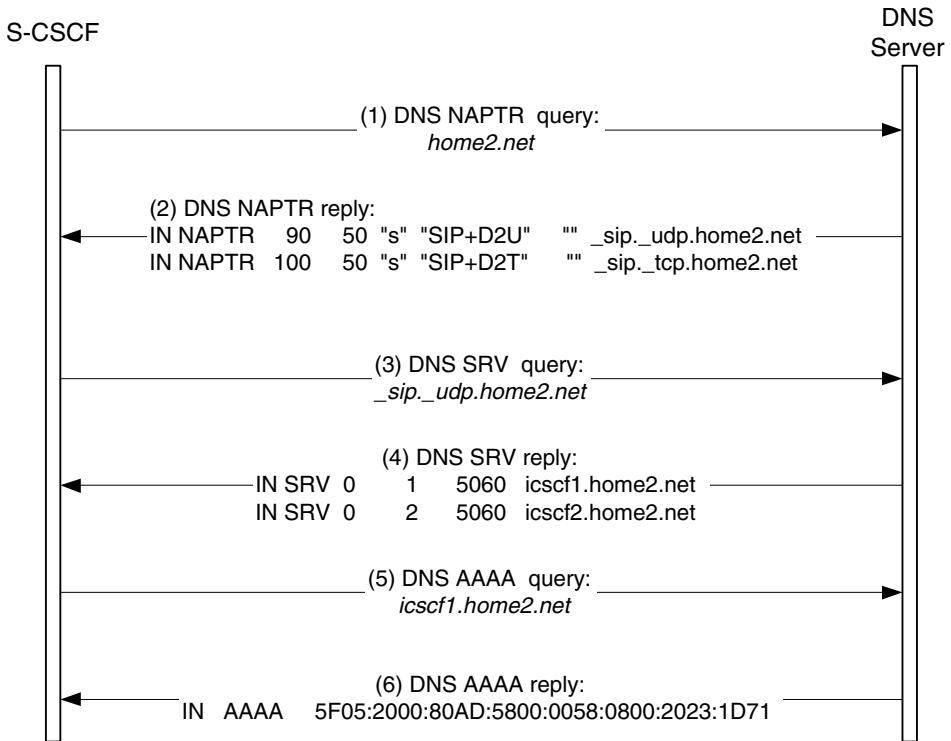


Figure 5.30: DNS procedures to locate a SIP server in the terminating home network

DNS returns not only the required information, but also the information that is likely to be requested at a later point. This avoids subsequent queries to the DNS. On the other hand, the responses offered by DNS are cached for a certain length of time. This means that, if within a certain configurable period of time the S-CSCF has to query the DNS with the same input query (e.g., the same domain name), then all the data are locally cached and no interaction with DNS is required.

We indicated that the *Request-URI* of the INVITE request may contain either a SIP URI or a TEL URI. In the case where the *Request-URI* contains a TEL URI (specified in RFC 3966 [295]), there are two possibilities. The first possibility is that the telephone number contained in the TEL URI “belongs” to an IMS user; therefore, there is most likely a SIP URI allocated to that user that maps the TEL URI to a SIP URI. The other possibility is that the telephone number does not belong to an IMS user. This could be the case for a telephone number that belongs to a PSTN user or other circuit-switched network (e.g., GSM). The S-CSCF, being a SIP proxy server, implements routing based on SIP, and not on telephone numbers. So, the S-CSCF first tries to map the TEL URI into a SIP URI, typically by using another DNS service: the ENUM service (specified in RFC 2916 [143]).

DNS ENUM specifies a mechanism that consists of reversing the order of telephone number digits, adding a dot in between two digits, adding the domain name `.e164.arpa`, and performing a DNS NAPTR query with the resulting string. For instance, if the TEL URI is `tel:+1-212-555-1234`, the DNS query for the NAPTR record is sent to

4.3.2.1.5.5.2.1.2.1.e164.arpa. In the case where the telephone number is mapped to a SIP URI, DNS answers typically with a list of URIs including, but not restricted to, the SIP URI. The S-CSCF uses any of the SIP URIs as an entry to the routing process.

However, if the telephone number is not associated with a SIP URI, there is no mapping to DNS, so DNS returns a negative response, indicating that no record was available for that telephone number. This is an indication that the user is neither an IMS user nor a user defined at any other SIP domain. This may be the case when the user resides in the PSTN or a traditional circuit-switched network. As a result the S-CSCF is unable to forward the SIP request to its destination. Instead, the S-CSCF requires the services of a BGCF (Breakout Gateway Control Function). The BGCF is specialized in routing SIP requests based on telephone numbers, typically addressed to a circuit-switched network. We describe the interactions with circuit-switched networks in more detail in Section 5.10.1.

Once the S-CSCF has made a determination of how to route the INVITE request, it adds a new value to the existing P-Asserted-Identity header field. We saw how the P-CSCF inserts a P-Asserted-Identity header field that contains, as a value, one of the multiple SIP URIs that the user registered. The S-CSCF receives this value and identifies the user. Before forwarding the INVITE request, the S-CSCF adds a new value with a TEL URI associated with the user. The assumption here is that every user has a TEL URI associated to a SIP URI. The SIP URI enables SIP communications, whereas the TEL URI allows voice calls originated on or destined to a circuit-switched network, such as the PSTN. If the S-CSCF does not insert the TEL URI in the P-Asserted-Identity header field and the session terminates in the PSTN (perhaps due to a forwarding), then the PSTN network is unable to identify who is calling. Figure 5.31 shows the contents of the P-Asserted-Identity when the S-CSCF forwards the INVITE request to the final destination.

```
P-Asserted-Identity: "Alice Smith" <sip:alice@home1.net>,
<URI}tel:+1-212-555-1234>
```

Figure 5.31: The P-Asserted-Identity header field

The S-CSCF also takes action on the P-Access-Network-Info header field. If the SIP request is forwarded outside the home domain, then the S-CSCF removes the header, since it may contain sensitive private information (e.g., the cell ID). If the request is forwarded to an Application Server (AS) that is part of the home network, then the S-CSCF keeps the P-Access-Network-Info header field in the request, so that the AS can read the contents.

In 3GPP Release 5, the S-CSCF always remains in the path for subsequent SIP signaling sent within the SIP dialog created by the INVITE request. 3GPP Release 6 adds further flexibility by allowing an S-CSCF not to remain in the path for subsequent signaling. This can be used, for example, if the SIP request is SUBSCRIBE request forwarded or terminated by an AS that is keeping charging records and perhaps other data. If the SIP request is an INVITE request, the S-CSCF typically remains in the path for subsequent requests and adds its own SIP URI to the Record-Route header field value.

5.7.4 The Terminating I-CSCF Processes the INVITE Request

Let us return to Figure 5.19 and follow up the INVITE request. The S-CSCF has found, through DNS procedures, a SIP server in the destination home network. This SIP server in the IMS is an I-CSCF. So the I-CSCF in the destination home network receives the INVITE

request, (5) in Figure 5.19. The I-CSCF is not interested in the identity of the caller, identified in the P-Asserted-Identity header field. The I-CSCF is, however, interested in the callee, identified in the *Request-URI* of the INVITE request. The I-CSCF has to forward the SIP request to the S-CSCF allocated to the callee. For the sake of simplicity, let us assume that the callee is registered to the network and, so an S-CSCF is allocated to them. The I-CSCF is not aware of the address of the S-CSCF allocated to the callee, so the I-CSCF has to discover it. The S-CSCF saved its own address in the HSS during the registration procedure. Therefore, the I-CSCF queries the HSS with a Diameter Location-Information-Request (LIR) message (7).

The I-CSCF includes the address of the callee, copied from the *Request-URI* of the SIP request, in a Public-Identity Attribute-Value Pair (AVP) of the Diameter LIR request. In the INVITE request example we have been following, the *Request-URI* is set to `sip:bob@home2.net`, and so is the Public-Identity AVP of the Diameter LIR request. The HSS receives the Diameter LIR request, inspects the Public-Identity AVP, searches for the data associated with that user, gets the stored S-CSCF address, and inserts it into a Server-Name AVP in a Diameter Location-Information-Answer (LIA) message, which is step (8) in Figure 5.19.

Once the I-CSCF receives the Diameter LIA message, it knows where to route the INVITE request (5), whose routing was temporarily suspended. The I-CSCF at this stage does not modify or add any SIP header field, other than the SIP routing header fields (e.g., Route, Max-Forwards, Via, etc.). So, the I-CSCF forwards the INVITE request to the S-CSCF that is assigned to the callee, (9) in Figure 5.19.

The I-CSCF may or may not remain in the path for subsequent SIP signaling sent within the SIP dialog created by the INVITE request. It depends on the network topology and the configuration. For instance, the security policy in the network may dictate that SIP signaling has to traverse an I-CSCF, perhaps because the operator does not want to expose the S-CSCF to SIP signaling generated outside the home network. In such a case the I-CSCF adds its own SIP URI to the Record-Route header field value. Otherwise, the I-CSCF does not add itself to the Record-Route header field value, and once the INVITE transaction is complete the I-CSCF will not receive further SIP requests and responses belonging to the dialog created by the INVITE request.

5.7.5 The Terminating S-CSCF Processes the INVITE Request

The S-CSCF in the terminating network that takes care of the callee receives the INVITE request, (9) in Figure 5.19. The S-CSCF first identifies the callee by the *Request-URI* in the INVITE request. Then, the S-CSCF evaluates the initial filter criteria of the called user. This evaluation is the same as that undertaken by the S-CSCF in the originating side with respect to the calling user, with the only exception being that in this case, the terminating S-CSCF is looking for services that apply to sessions established toward the user, rather than sessions originated by the user. For the sake of simplicity, we assume at this stage that the callee does not have any service that requires the involvement of an Application Server.

In 3GPP Release 5, the terminating S-CSCF always remains in the path for subsequent SIP signaling sent within the SIP dialog created by the INVITE request. 3GPP Release 6 adds the possibility of the S-CSCF not remaining in the path. The procedure is most likely applicable to non-INVITE requests that are terminated in an Application Server. This may be the case with the presence service. However, for INVITE requests, the terminating S-CSCF

typically remains in the path; therefore, it adds its own SIP URI to the `Record-Route` header field value.

Then, the S-CSCF continues with the processing of the INVITE request. The goal is to forward the INVITE request to the callee's IMS terminal, by means of traversing a set of proxies determined during the registration process of the callee. This set of proxies will always include a P-CSCF and may include one or more I-CSCFs. In the example in Figure 5.19 we assume that SIP has to traverse only a P-CSCF.

The S-CSCF, therefore, creates a new *Request-URI* with the contents of the `Contact` header field value registered by the callee during registration. The S-CSCF also sets the value of the `Route` header field to that of the `Path` header field learnt during the same registration process. In our case, as the SIP requests have to traverse the P-CSCF that is assigned to the user, the `Route` header contains the P-CSCF URI learnt during user registration. Figure 5.32 shows the details of the `Route` header field.

```
Route: <sip:pcscf2.visited2.net;lr>
```

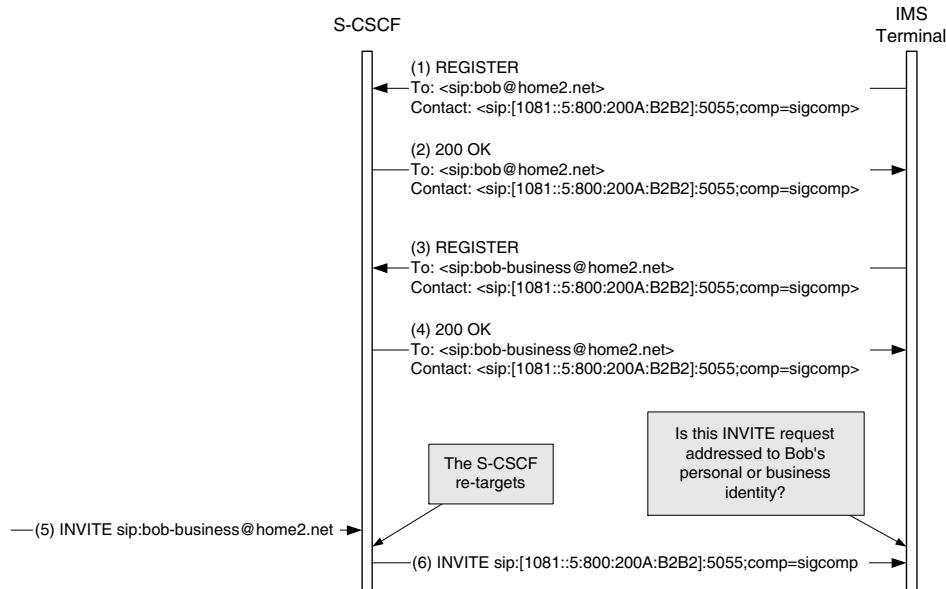
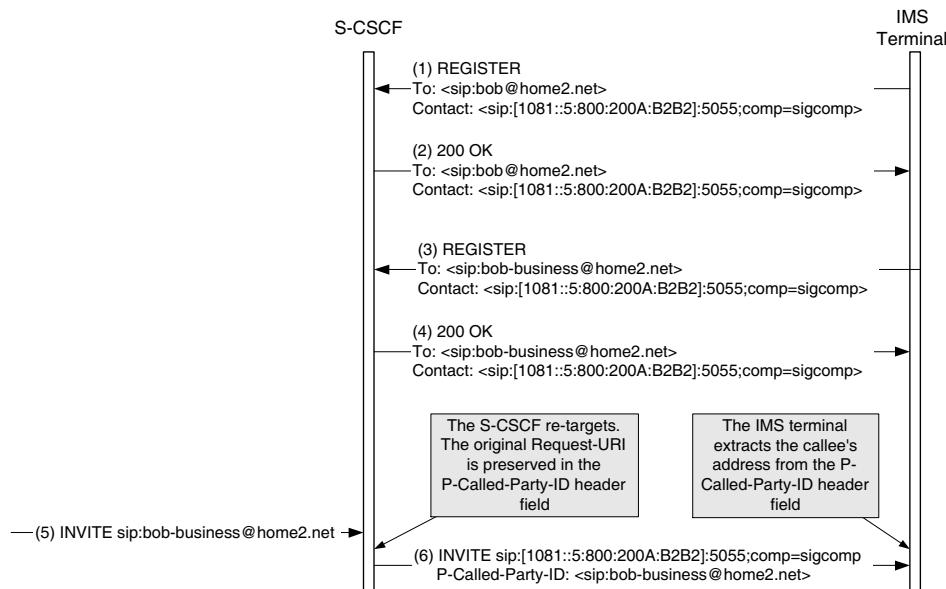
Figure 5.32: The `Route` header field

3GPP had a requirement for SIP that led to an extension in the form of the `P-Called-Party-ID` header field. Let us take a look at the problem description with the help of Figure 5.33. We saw in Section 5.5 that the IMS terminal can register a new Public User Identity at any time. Steps (1)–(4) in Figure 5.33 describe an IMS user registering two different Public User Identities, identified by the `To` header field value of the SIP REGISTER request. In step (1), Bob is registering the Public User Identity `sip:bob@home2.net`, whereas in step (3) Bob is registering the identity `sip:bob-business@home2.net`. In both REGISTER requests the `Contact` header field contains the same value: the SIP URI of the IMS terminal.

We have seen in previous paragraphs that the S-CSCF, on receiving an INVITE request, such as step (5) of Figure 5.33, replaces the contents of the original *Request-URI* header field with more accurate information extracted from the `Contact` header during the registration of the user. This procedure in SIP parlance is known as *retarget*. So, the S-CSCF retargets the *Request-URI* in the INVITE request. When the INVITE request is delivered to the IMS terminal, as in step (6) of Figure 5.33, Bob does not know to which of the several Public User Identities this INVITE request corresponds. Was this INVITE sent toward Bob's personal or business identity? Perhaps the IMS terminal wants to apply a different alerting tone, depending on the callee's address, but this is not possible because the callee's address was lost during the retarget process.

The solution led to the creation of the `P-Called-Party-ID` header field in SIP, which is sketched in Figure 5.34. The `P-Called-Party-ID` header field is documented in 3GPP extensions to SIP (RFC 3455 [154]). The mechanism is quite simple: when the S-CSCF retargets, it also inserts a `P-Called-Party-ID` header field, whose value is set to the original *Request-URI*. Then, the S-CSCF forwards the INVITE request, (6) in Figure 5.34, including the `P-Called-Party-ID` header field. When the IMS terminal receives the INVITE request, it inspects the contents of the `P-Called-Party-ID` header field in order to determine which of the various identities the INVITE request is destined for. Then, the IMS terminal may apply any distinguished alerting tone or similar service.

Returning to our example in Figure 5.19, the S-CSCF eventually forwards the INVITE request to the P-CSCF, (11) in Figure 5.19.

**Figure 5.33:** The S-CSCF re-targets**Figure 5.34:** Usage of the P-Called-Party-ID header field

5.7.6 The Terminating P-CSCF Processes the INVITE Request

The P-CSCF receives the INVITE request, (11) in Figure 5.19, and does not need to take any routing decision, because the *Request-URI* already contains a SIP URI that includes the IP address (or host name) of the IMS terminal. The P-CSCF needs to identify the Public User Identity of the callee in order to find the proper security association established with the IMS terminal of the user. The P-CSCF extracts this Public User Identity from the P-Called-Party-ID header of the SIP INVITE request.

The P-CSCF always remains in the path for subsequent SIP signaling sent within the SIP dialog created by the INVITE request. Therefore, the P-CSCF adds its own SIP URI to the Record-Route header field value.

The terminating P-CSCF inspects the contents of the Privacy header field and, if needed, acts to remove the P-Asserted-Identity header field. For instance, if the Privacy header field is set to a value of *id*, then the P-CSCF removes the P-Asserted-Identity header field from the INVITE request that it sends to the callee's IMS terminal. This gives the originating user some degree of privacy, as the terminating user is unable to determine who the caller is. In the example we have been following, the caller set the Privacy header field to *none*, indicating that there is no requirement to keep any privacy; therefore, the P-CSCF keeps the P-Asserted-Identity header field in the INVITE request.

The P-CSCF needs to carry out a number of functions that are related to policy control, security aspects, compression of SIP signaling, etc. But there are no other SIP-specific functions other than acting as a SIP proxy. We describe all of these functions in their respective sections.

Figure 5.35 shows an example of the INVITE request that the P-CSCF sends to the callee.

5.7.7 The Callee's Terminal Processes the INVITE Request

The INVITE request in Figure 5.35 is received at the callee's IMS terminal. The INVITE carries an SDP offer generated at the caller's terminal. The SDP offer indicates the IP address and port numbers where the caller wants to receive media streams, the desired and supported codecs for each of the media streams, etc.

The INVITE request indicates in the Supported header field the caller's support for the *precondition* extension. In this example, we assume that both the caller and the callee have to reserve resources (Section 5.12 describes cases when it is not the case that both terminals have to reserve resources. If the callee has to reserve resources, then the callee needs to know when the caller has completed his resource reservation. Therefore, the caller turns the initially optional usage of the preconditions extension into mandatory usage.

The precondition extension model requires that the callee respond with a 183 (Session Progress) response that contains an SDP answer. The SDP answer contains the media streams and codecs that the callee is able to accept for this session. We expect IMS terminals to be configured to answer this INVITE request automatically without requiring any interaction with the user. For instance, users may have configured their IMS terminals to automatically accept sessions that require support for the AMR audio codec and the H.263 video codec. However, this is terminal-implementation dependent, and different terminal manufacturers may include different features. In any case, if automatic configuration is not available, the terminal has to alert the user to take a decision about how to answer the SDP with the preferred streams and codecs.

The callee's IMS terminal could start its resource reservation process at this stage, because it knows all the parameters needed (all included in the SDP) to start resource reservation.

```

INVITE sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp SIP/2.0
Via: SIP/2.0/UDP pcscf2.visited2.net:5056;comp=sigcomp;
branch=z9hG4bK2a2qr
Via: SIP/2.0/UDP scscf2.home2.net;branch=z9hG4bKvp2ym1
Via: SIP/2.0/UDP icscf2.home2.net;branch=z9hG4bKra1ar
Via: SIP/2.0/UDP scscf1.home1.net;branch=z9hG4bKs1pp0
Via: SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
branch=z9hG4bK9h9ab
Max-Forwards: 65
Record-Route: <sip:pcscf2.visited2.net:5056;lr;comp=sigcomp>
Record-Route: <sip:scscf2.home2.net;lr>
Record-Route: <sip:scscf1.home1.net;lr>
Record-Route: <sip:pcscf1.visited1.net;lr>
P-Asserted-Identity: "Alice Smith" <sip:alice@home1.net>,
<URI;tel:+1-212-555-1234>
Privacy: none
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>
Call-ID: 3s09cs03
Cseq: 127 INVITE
Supported: 100rel, precondition
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Content-Type: application/sdp
Content-Length: 591

v=0
o=- 1073055600 1073055600 IN IP6 1080::8:800:200C:417A
s=-
c=IN IP6 1080::8:800:200C:417A
t=0 0
m=video 8382 RTP/AVP 98 99
b=AS:75
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=inactive
a=rtpmap:98 H263
a=fmtpt:98 profile-level-id=0
a=rtpmap:99 MP4V-ES
m=audio 8283 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=inactive
a=rtpmap:97 AMR
a=fmtpt:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.35: (13) INVITE

However, if the terminal started resource reservation at this stage, it would need to reserve resources for the higher-demand codec, when several codecs are possible within a particular media stream. For instance, the SDP in the INVITE request declares its support for two video codecs: H.263 and MPEG-4. It is not worth reserving resources for a codec that requires a higher bandwidth, when that codec may not be used at all during the session. If the session charge depends on the reserved resources, then it is easier to justify the short period of time that the terminal needs to wait for a potential new SDP offer. If the SDP offer contains one codec per media stream, then resource reservation can start straight away.

The IMS terminal inspects the P-Asserted-Identity header field, if present, and extracts the identity of the caller. The P-CSCF may remove this header field if the privacy requirement indicates this. The IMS terminal also determines, by inspecting the value of the P-Called-Party-ID header field, which of the several identities of its user the INVITE request is addressed to. The combination of caller and callee identities may be used at this or a later stage to play a personalized ringing tone, display a picture of the calling party, etc.

Figure 5.36 shows an example of a 183 (Session Progress) provisional response that the callee's IMS terminal sends back to the originating side. The IMS terminal inserts a P-Access-Network-Info header field in a similar way to that the originating terminal used to insert the same header field in the INVITE request.

The terminal also inserts a Require header field with the values 100rel and precondition. The latter is required because the terminal has to reserve resources, and needs to follow the model for resource reservation described in the precondition extension. The former is required because the *precondition* extension requires that provisional responses be transmitted reliably. If a reliable provisional responses extension is not used, and if UDP is used as a transport protocol in between any two hops in the path, then there is no guarantee that the response will ever arrive at the remote party. It can be lost, and the UAS that issued the provisional response will never notice it. To solve the problem the “Reliability of Provisional Responses in the Session Initiation Protocol (SIP)” (specified in RFC 3262 [284]) guarantees that the reception of a provisional response is acknowledged by a PRACK request. If the callee does not receive a PRACK request to confirm the reception of the provisional response within a determined time, then it will retransmit the provisional response.

The *precondition* extension requires that SDP be included in a provisional response and that support for the reliable provisional responses mechanism is given to guarantee that provisional responses are received at the originating terminal. This is the reason why the callee's IMS terminal inserts a Require header field with the tag 100rel in the 183 (Session Progress) provisional response.

Figure 5.36 shows a Privacy header field set with the value none. Hence the user does not have a requirement to maintain privacy with respect to those header fields that reveal any identity or private information. However, the reader may notice the absence of a P-Preferred-Identity header field. This header field is used to give a hint to the network node about the identity that the user wants to reveal. The network already knows this information, because the INVITE request was addressed to that identity. So, there is no need for the IMS terminal to indicate to which identity the INVITE request was addressed.

The SDP in the 183 (Session Progress) provisional response also includes an a=conf : qos line. This line requests callers to send an updated SDP when they have reserved resources in their local segment. We described earlier that the IMS uses the so-called segmented model within the precondition model. According to the local segmented quality of service, in this model each terminal is responsible for reserving resources in its own local segment. A condition for the terminating terminal to alert the user is that resources are reserved at

```

SIP/2.0 183 Session Progress
Via: SIP/2.0/UDP pcscf2.visited2.net:5056;comp=sigcomp;
      branch=z9hG4bK2a2qr
Via: SIP/2.0/UDP scscf2.home2.net;branch=z9hG4bKvp2ym1
Via: SIP/2.0/UDP icscf2.home2.net;branch=z9hG4bKra1ar
Via: SIP/2.0/UDP scscf1.home1.net;branch=z9hG4bKs1ppo
Via: SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Record-Route: <sip:pcscf2.visited2.net:5056;lr;comp=sigcomp>,
               <sip:scscf2.home2.net;lr>,
               <sip:scscf1.home1.net;lr>,
               <sip:pcscf1.visited1.net;lr>
Privacy: none
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                       utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 127 INVITE
Require: 100rel, precondition
Contact: <sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
RSeq: 9021
Content-Type: application/sdp
Content-Length: 658

v=0
o=- 3021393216 3021393216 IN IP6 1081::5:800:200A:B2B2
s=-
c=IN IP6 1081::5:800:200A:B2B2
t=0 0
m=video 14401 RTP/AVP 98 99
b=AS:75
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=conf:qos remote sendrecv
a=inactive
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
a=rtpmap:99 MP4V-ES
m=audio 6544 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=conf:qos remote sendrecv
a=inactive
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.36: (15) 183 Session Progress

both the originating and at the terminating side. If the originating side reserves resources but does not inform the terminating side, the callee will never be alerted. To solve this problem, the terminating terminal requests the originating terminal to inform it when the originating terminal has its resources available, and this is indicated in the mentioned `a=conf:qos` line in SDP.

Since the IMS terminal has not reserved resources in its local segment, it sets all the media streams in “inactive” mode by keeping the existing `a=inactive` line in each of the media streams in SDP.

The IMS terminal also inserts a `Contact` header field whose value is a SIP URI that includes the IP address or Fully Qualified Domain Name of the IMS terminal. In addition, the SIP URI also includes the port number bound to the security association established between the P-CSCF and the IMS terminal. Also, the IMS terminal declares the willingness to compress SIP from and to the P-CSCF in the `comp=sigcomp` parameter.

In addition, the callee’s IMS terminal carries out the regular SIP processing to generate the 183 Session Progress toward the caller’s IMS terminal.

5.7.8 Processing the 183 Response

The 183 (Session Progress) response traverses, according to SIP procedures to route responses, the same set of proxies that the corresponding INVITE request traversed.

When the P-CSCF receives the 183 response, (15) in Figure 5.19, it verifies that the IMS terminal has formulated the response correctly. For example, it verifies that the `Via` and `Record-Route` header field contain the values that the IMS terminal should have used in a response to the INVITE request. This avoids “spoof” IMS terminals that “forget” to include the address of an S-CSCF or P-CSCF in a `Record-Route` header field to avoid any account of the activity of the IMS terminal. In the case where the P-CSCF finds a malformed `Via` or `Record-Route` header field, the P-CSCF discards the response (why bother for a “spoof” terminal) or rewrites that header field with the appropriate values.

When the P-CSCF received the INVITE request, it saved the value of the `P-Called-Party-ID` header. The value of that header indicated the Public User Identity of the receiver side. Now, when the P-CSCF receives the 183 response from the IMS terminal, the P-CSCF inserts a `P-Asserted-Identity` header field, whose value is the same as that included in the `P-Called-Party-ID` header field of the INVITE request. This mechanism allows the remaining trusted network nodes to get the Public User Identity of the callee being used for this session.

Eventually, the P-CSCF forwards the 183 response, based on the contents of the value of the topmost `Via` header field, to the S-CSCF, (16) in Figure 5.19.

The S-CSCF receives the 183 response. The S-CSCF removes the `P-Access-Network-Info` header field prior to forwarding the response to the I-CSCF (17).

The I-CSCF does not undertake any special processing in the 183 response. It just forwards it to the S-CSCF in the originating network (17).

The originating S-CSCF receives the 183 response (18) and may remove the `P-Asserted-Identity` header field if privacy requirements indicate so. In the example we are following, as the callee set the `Privacy` header field to the value `none`, the originating S-CSCF does not take any action on the `P-Asserted-Identity` header field.

Eventually, the S-CSCF forwards the 183 response (19) to the P-CSCF. The P-CSCF forwards the response (20) to the caller’s IMS terminal.

5.7.9 *The Caller's IMS Terminal Processes the 183 Response*

The caller's IMS terminal receives the 183 response and focuses particularly on the SDP answer. The SDP contains the IP address of the remote terminal (i.e., the destination IP address where the media streams are sent). It also contains an indication of whether the callee accepted establishment of a session with these media streams. For instance, the callee could have accepted the establishment of an audio session without video. However, this is not the case in our example, because the callee accepted all the media streams proposed in the SDP offer, as can be seen in the SDP answer.

The SDP answer also includes an indication from the callee that it wants to be notified when the caller's IMS terminal has completed its resource reservation process. This is indicated by the presence of the `a=conf : qos` line somewhere below each `m=` line.

The SDP answer indicates which of the offered codecs are supported and desired for this session at the callee. We have seen an example in Figure 5.36 in which the callee supported the two offered video codecs (H.263 and MPEG-4) and the audio codec (AMR). An IMS terminal typically tries to narrow down the number of negotiated codecs to just one codec per media stream. This allows accurate resource reservation which only requires reservation of the bandwidth of that codec. If several codecs were negotiated, the terminal would reserve the maximum bandwidth required by the most demanding codec, even though during the session the actual codec used could require less bandwidth. This leads to an inefficient use of resource management and, potentially, the user being charged for bandwidth that was not actually used. For this reason, IMS terminals, once a first SDP exchange has taken place, trim down the number of codecs per media stream to just a single codec.

Hence the caller's IMS terminal creates a new SDP offer, whose only difference is the removal of the MPEG-4 codec from the video line in SDP in favor of the H.263 codec.

As the 183 response required the response to be acknowledged, because of the presence of the `100rel` tag in the `Require` header field, the caller's IMS terminal creates a PRACK request. The new SDP offer is included in the PRACK request. Figure 5.37 shows the PRACK request that carries a piggybacked SDP offer.

In parallel with the generation of the PRACK request, the IMS terminal starts the resource reservation mechanism. The actual procedure is dependent on the underlying IP Connectivity Access Network. Typically, it will require some sort of dialog exchange with the packet and radio nodes.

The PRACK request traverses all the proxies that asked to remain in the path for subsequent signaling. Typically, this path will be a subset of the proxies that the INVITE request traversed. The path is determined by the `Route` header field included in the PRACK request. The value of the `Record-Route` header field included in the 183 response sets that path.

5.7.10 *The Callee's IMS Terminal Processes the PRACK request*

When the PRACK request, (25) in Figure 5.19, is received at the callee the IMS terminal generates a 200 (OK) response (26) that contains an SDP answer. The 200 (OK) response is an answer to the PRACK request and it should not be confused with a 200 (OK) for the INVITE that will occur later.

Because this is the second offer/answer exchange during the establishment of this particular session, there is almost nothing to negotiate, just confirmation of the media streams

```

PRACK sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Max-Forwards: 70
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                        utran-cell-id-3gpp=24450289A3299239
Route: <sip:pcscf1.visited1.net:5058;lr;comp=sigcomp>,
       <sip:scscf1.home1.net;lr>,
       <sip:scscf2.home2.net;lr>,
       <sip:pcscf2.visited2.net;lr>
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 128 PRACK
Require: precondition, sec-agree
Proxy-Require: sec-agree
Security-Verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96;
                  spi-c=98765432; spi-s=909767;
                  port-c=5057; port-s=5058
RAck: 9021 127 INVITE
Content-Type: application/sdp
Content-Length: 577

v=0
o=- 1073055600 1073055602 IN IP6 1080::8:800:200C:417A
s=-
c=IN IP6 1080::8:800:200C:417A
t=0 0
m=video 8382 RTP/AVP 98
b=AS:75
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=inactive
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
m=audio 8283 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=inactive
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.37: (21) PRACK

and codecs of the session. At this time, the terminal may still be involved in its resource reservation process, and most likely at this stage the resource reservation will not be complete. Therefore, the SDP answer generated by the callee typically indicates that there are no resources available yet at the local segment (`a=curr:qos local` and `a=des:qos local`). The IMS terminal still indicates that it wants to receive an indication when the caller is ready with the resource reservation. Figure 5.38 shows the 200 (OK) response, (26) in Figure 5.19.

At the same time, the callee starts resource reservation in its own segment. Typically, this is a process that involves the terminal, the radio nodes, and the packet nodes. If the IP-CAN is a GPRS network, it involves the SGSN and GGSN on the packet network as well as the radio nodes.

The 200 (OK) response to the PRACK request traverses the same set of SIP proxies that the PRACK request traversed. Eventually, the 200 (OK) response, (30) in Figure 5.19, arrives at the caller's IMS terminal. At this stage the caller's IMS terminal is most likely still engaged in its resource reservation process.

Once the caller's IMS terminal has got the required resources from the network, it honors the request for a confirmation message, present in the form of the `a=conf` line in the SDP answer, (20) and (30) in Figure 5.19. The IMS terminal sends an UPDATE request that visits the same set of proxies as the PRACK request (i.e., those proxies that asked to receive subsequent SIP messages by adding their own SIP URIs to the Record-Route header field of the INVITE request, (1)–(13) in Figure 5.19).

Figure 5.39 shows an example of the contents of the UPDATE request, (31) in Figure 5.19. The UPDATE request contains another SDP offer, in which the caller's IMS terminal indicates that resources are reserved at his local segment. The presence of the `a=curr:qos local sendrecv` line conveys this information. In addition, since resources are already available at the caller's side, each media stream is now qualified with the line `a=sendrecv`, indicating the ability to send and receive media. Since `a=sendrecv` is the default bi-directional state for a media stream in SDP, it can be safely omitted.

Eventually, the callee's IMS terminal receives the UPDATE request (35). The IMS terminal will generate a 200 (OK) response, (36) in Figure 5.19. According to the SDP offer/answer model a successful SDP offer has to be answered with an SDP answer. So, the callee's IMS terminal includes an SDP answer in the 200 (OK) response. At this stage the callee's IMS terminal may have already finished its resource reservation, or not, depending on how much time it takes to complete the process. Therefore, the callee's IMS terminal indicates its own local quality-of-service status, which may either be complete, or not. This is indicated once more in the `a=curr:qos local` line in SDP. Figure 5.40 shows an example of the 200 (OK) response that the callee's IMS terminal makes when resources are already available to it.

The 200 (OK) response will follow the same path as the UPDATE request. Eventually, the caller's IMS terminal will receive it and the UPDATE transaction will be completed.

5.7.11 Alerting the Callee

On the callee's side there are two conditions that have to be met before the callee is alerted. On one side, the terminal needs to complete its local resource reservation process. On the other side, the callee's terminal has to get the information that the caller's terminal has also completed its local resource reservation process. The latter information is conveyed in the SDP of an UPDATE request. Resource reservation and reception of the UPDATE request are two independent processes that can happen in any order, but the idea is that the callee

```

SIP/2.0 200 OK
Via: SIP/2.0/UDP pcscf2.visited2.net:5056;comp=sigcomp;
      branch=z9hG4bK2a2qr
Via: SIP/2.0/UDP scscf2.home2.net;branch=z9hG4bKvp2yml
Via: SIP/2.0/UDP scscf1.home1.net;branch=z9hG4bKs1pp0
Via: SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
P-Access-Network-Info: 3GPP-UTRAN-TDD;
      utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 128 PRACK
Content-Type: application/sdp
Content-Length: 634

v=0
o=- 3021393216 3021393218 IN IP6 1081::5:800:200A:B2B2
s=-
c=IN IP6 1081::5:800:200A:B2B2
t=0 0
m=video 14401 RTP/AVP 98
b=AS:75
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=conf:qos remote sendrecv
a=inactive
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
m=audio 6544 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=conf:qos remote sendrecv
a=inactive
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.38: (26) 200 OK

```

UPDATE sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
branch=z9hG4bK9h9ab
Max-Forwards: 70
P-Access-Network-Info: 3GPP-UTRAN-TDD;
utran-cell-id-3gpp=24450289A3299239
Route: <sip:pcscf1.visited1.net:5058;lr;comp=sigcomp>,
<sip:scscf1.home1.net;lr>,
<sip:scscf2.home2.net;lr>,
<sip:pcscf2.visited2.net;lr>
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 129 UPDATE
Require: sec-agree
Proxy-Require: sec-agree
Security-Verify: ipsec-3gpp; q=0.1;
alg=hmac-sha-1-96;
spi-c=98765432; spi-s=909767;
port-c=5057; port-s=5058
Content-Type: application/sdp
Content-Length: 585

v=0
o=- 1073055600 1073055604 IN IP6 1080::8:800:200C:417A
s=-
c=IN IP6 1080::8:800:200C:417A
t=0 0
m=video 8382 RTP/AVP 98
b=AS:75
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=sendrecv
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
m=audio 8283 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.39: (31) UPDATE

```

SIP/2.0 200 OK
Via: SIP/2.0/UDP pcscf2.visited2.net:5056;comp=sigcomp;
      branch=z9hG4bK2a2qr
Via: SIP/2.0/UDP scscf2.home2.net;branch=z9hG4bKvp2yml
Via: SIP/2.0/UDP scscf1.home1.net;branch=z9hG4bKs1pp0
Via: SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                      utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 129 UPDATE
Content-Type: application/sdp
Content-Length: 594

v=0
o=- 3021393216 3021393220 IN IP6 1081::5:800:200A:B2B2
s=-
c=IN IP6 1081::5:800:200A:B2B2
t=0 0
m=video 14401 RTP/AVP 98
b=AS:75
a=curr:qos local sendrecv
a=curr:qos remote sendrecv
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=sendrecv
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
m=audio 6544 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote sendrecv
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.40: (36) 200 OK

should not be alerted before resources are available at both sides of the session (e.g., caller and callee).

When the callee's IMS terminal rings, it will also generate a 180 (Ringing) provisional response, (41) in Figure 5.20. The response is sent to the caller's terminal and traverses those proxies that the INVITE request traversed. Figure 5.41 shows an example of the 180 (Ringing) response. The response typically does not contain SDP, since all the session parameters, such as media streams and codecs, have already been negotiated in the previous exchanges. Because of the lack of SDP, the 180 (Ringing) response does not really require to be confirmed with a PRACK request. In this example we show it requiring a PRACK request confirmation, because of the presence of the `Require` header field with the value `100rel` in it. Section 5.12 shows other examples where the 180 (Ringing) response is unreliable and does not require a PRACK request confirmation.

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP pcscf2.visited2.net:5056;comp=sigcomp;
      branch=z9hG4bK2a2qr
Via: SIP/2.0/UDP scscf2.home2.net;branch=z9hG4bKvp2ym1
Via: SIP/2.0/UDP icscf2.home2.net;branch=z9hG4bKra1ar
Via: SIP/2.0/UDP scscf1.home1.net;branch=z9hG4bKs1pp0
Via: SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Record-Route: <sip:pcscf2.visited2.net:5056;lr;comp=sigcomp>,
               <sip:scscf2.home2.net;lr>,
               <sip:scscf1.home1.net;lr>,
               <sip:pcscf1.visited1.net;lr>
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                       utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 127 INVITE
Require: 100rel
Contact: <sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
RSeq: 9022
Content-Length: 0
```

Figure 5.41: (41) 180 Ringing

When the caller's IMS terminal receives the 180 (Ringing) response, (46) in Figure 5.20, it will likely generate a locally stored ring-back tone to indicate to the caller that the peer terminal is ringing. Then, since the response contains a `Require` header field containing the value `100rel`, the caller's IMS terminal generates a PRACK request, (47) in Figure 5.20, and sends it to the callee. This PRACK request does not typically contain SDP. The PRACK request visits the same proxies as the previous PRACK and UPDATE requests

(i.e., those which recorded the route during the INVITE transaction). Eventually, the callee's IMS terminal will receive the PRACK request (51) and will answer it with a 200 (OK) response (52). This 200 (OK) response for the PRACK request does not typically contain SDP. The caller's IMS terminal will eventually receive this response, (56) in Figure 5.20.

When the callee finally accepts the session the IMS terminal sends a 200 (OK) response, (57) in Figure 5.20. This 200 (OK) response completes the INVITE transaction. The response does not typically contain SDP. Figure 5.42 shows an example of this response. Eventually, the caller's IMS terminal receives the response, (62) in Figure 5.20, and starts generating media-plane traffic. The caller's IMS terminal also sends an ACK request, (63) in Figure 5.20, to confirm receipt of the 200 (OK) response. The ACK request is routed back to the callee's IMS terminal. Figure 5.43 shows an example of the ACK request sent from the caller's IMS terminal.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pcscf2.visited2.net:5056;comp=sigcomp;
      branch=z9hG4bK2a2qr
Via: SIP/2.0/UDP scscf2.home2.net;branch=z9hG4bKvp2ym1
Via: SIP/2.0/UDP icscf2.home2.net;branch=z9hG4bKra1ar
Via: SIP/2.0/UDP scscf1.home1.net;branch=z9hG4bKs1pp0
Via: SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Record-Route: <sip:pcscf2.visited2.net:5056;lr;comp=sigcomp>,
               <sip:scscf2.home2.net;lr>,
               <sip:scscf1.home1.net;lr>,
               <sip:pcscf1.visited1.net;lr>
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                      utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 127 INVITE
Contact: <sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Content-Length: 0
```

Figure 5.42: (57) 200 OK

The session setup is complete, and both users can generate their respective audio and video media streams. These media streams are in general sent end-to-end (e.g., from the caller to the callee's IMS terminals and vice versa) via IP-CAN routers.

5.8 Application Servers: Providing Services to Users

The exploration of the basic session setup in Section 5.7 has provided the reader with an overview of a basic audio/video session established between two IMS users, where none of the users have any services executed (other than the audio/video session). This goes against the spirit of IMS, where the basic idea is to consider the IMS as a common platform for

```

ACK sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Max-Forwards: 70
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                      utran-cell-id-3gpp=24450289A3299239
Route: <sip:pcscf1.visited1.net:5058;lr;comp=sigcomp>,
       <sip:scscf1.home1.net;lr>,
       <sip:scscf2.home2.net;lr>,
       <sip:pcscf2.visited2.net;lr>
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 127 ACK
Content-Length: 0

```

Figure 5.43: (63) ACK

providing innovative services. The basic session setup example, which provided no extra services, was just an educational exercise created for the sole purpose of understanding the basics of the IMS, the basics of SIP routing in the IMS, and the peculiarities of SIP when used in the IMS.

A more realistic example includes providing one or more services to the caller, the callee, or even both of them.

In this section we describe the generalities of Application Servers (ASes), the different types of AS that we can find in the IMS, the different modes of operation of an AS, how they get involved during session setup, and how they can provide a service to the user.

5.8.1 Generalities about Application Servers

In a network there is generally more than one AS. Typically, there will be several ASes, each specialized in providing a particular service. Some Application Servers will implement some technologies, such as Java technology, SIP servlets, or SIP CGI (Common Gateway Interface). All of these ASes are characterized by implementing a SIP interface toward the S-CSCF. For historical reasons the interface defined between the S-CSCF and the AS is known as the IMS Service Control (*ISC*) interface. The name was adopted prior to 3GPP agreeing to use SIP over that interface. That is the reason for its different name.

Furthermore, 3GPP specifications still define the interface as “SIP plus some possible extensions”, despite the interface being pure SIP and, with the passage of time, the need not arising to define extensions to SIP on that interface, in contrast to all other interfaces arriving at the S-CSCF.

Figure 5.44 shows the possible combinations of ASes and the different interfaces. ASes can be located in the home network or in a third-party service provider network. But it is up to the S-CSCF to decide whether to involve an AS in the session setup or not.

In addition, any AS can implement other protocols such as HTTP (Hypertext Transfer Protocol, specified in RFC 2616 [144]) or WAP (Wireless Application Protocol [314]), although this option is not described in the IMS specifications (e.g., to provide a graphical

interface to the user, so that the user may configure some specific details of a service over HTTP or WAP).

For instance, an AS may carry out the same task as a call-forwarding service. The user may log onto the web page of the AS to configure the SIP URI where all the sessions will be forwarded. The graphical user interface provided by a web page is an improvement to the end user's experience when configuring services, compared with the current mechanism of configuring services in circuit-switched networks.¹⁴

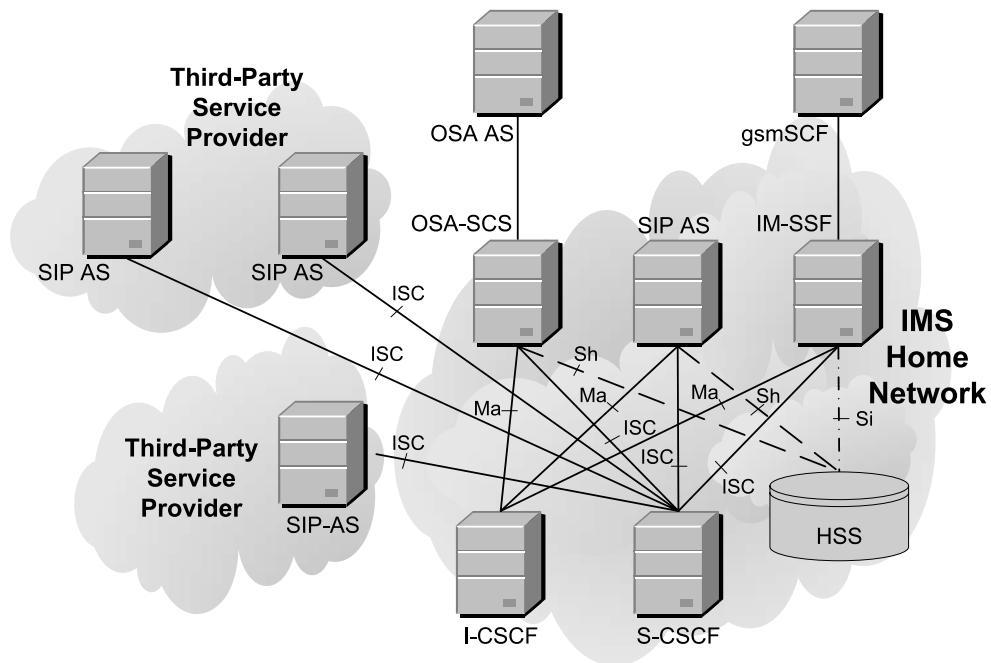


Figure 5.44: Interfaces to the Application Server

3GPP Release 6 adds yet another optional standardized interface toward an Application Server. This interface is code-named *Ut* and is used to provide the user with a protocol to configure and manage groups, policies, etc. So, this is not an interface used for live traffic. The protocol on the *Ut* interface is based on the XML Configuration Access Protocol (XCAP), specified in RFC 4825 [277]. XCAP defines how to use HTTP to create, modify, and delete an XML document, element, attribute, or value of a XML element. Figure 5.45 shows the *Ut* interface defined between the IMS terminal and ASes. We describe XCAP in greater detail in Chapter 17.

In addition, I-CSCFs provide the *Ma* interface towards Application Servers. This interface allows an I-CSCF to receive an incoming request addressed to a Public Service Identity (PSI) that resolves to an Application Server. The I-CSCF routes the request directly to the Application Server via the *Ma* interface. While the *Ma* interface is independent of the location of the Application Server, since the PSI needs to resolve to the home network, so that

¹⁴The mechanism to configure services in circuit-switched networks typically consists of the user dialing a collection of digits, such as *21*5551122#.

an I-CSCF in the home network receives the request, it implies that the *Ma* interface makes sense in Application Servers that belong to the home network.

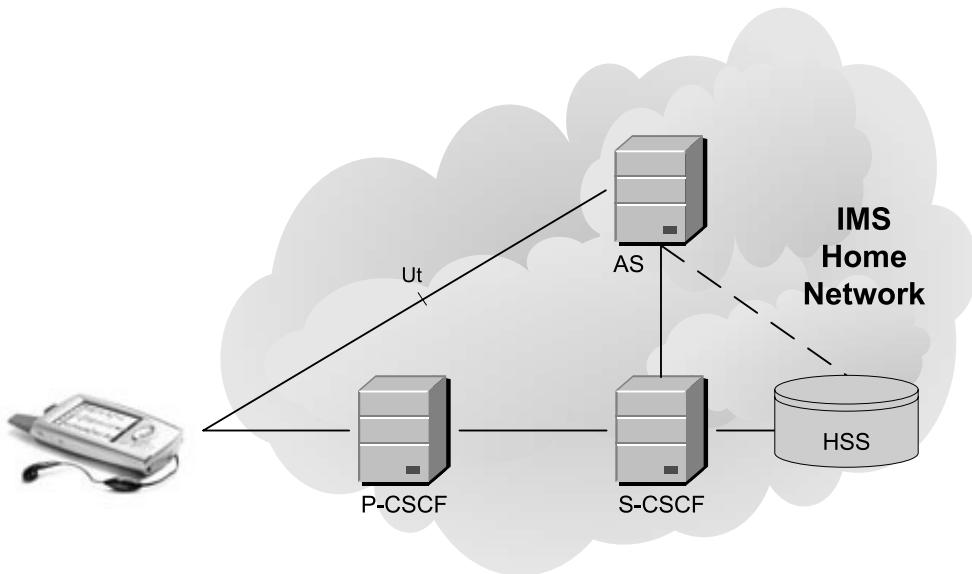


Figure 5.45: The *Ut* interface

5.8.2 Types of Application Servers

The IMS defines three different types of Application Server, depending on their functionality. Let us take a look at the characteristics of each of those types of AS.

5.8.2.1 The SIP Application Server

The SIP Application Server is the native AS in the IMS. New services exclusively developed for the IMS are likely to be executed by SIP ASes.

Figure 5.46 shows the relation between SIP ASes and the rest of the network. When a SIP AS is located in the home network, it can optionally implement an interface to the HSS. The implementation of the interface depends on whether the actual service logic needs to further interact with the HSS or not. The optional interface from the AS to the HSS is code-named *Sh*, and the protocol is based on Diameter (specified in RFC 3588 [96]) and an application specified in 3GPP TS 29.328 [42] and 3GPP TS 29.329 [54].

Diameter is a highly extensible protocol designed to provide authentication, authorization, and accounting (AAA) services. We study the Diameter protocol in depth when we describe the *AAA on the Internet* in Section 6.3. Because the *Sh* interface is based on Diameter, we describe the *Sh* interface in Section 7.3.

We already mentioned that SIP ASes can be located in the home network or in a third-party service provider network. If the SIP AS is located in a third-party service provider network, it cannot implement the *Sh* interface in the HSS, as *Sh* is just an intra-operator interface.

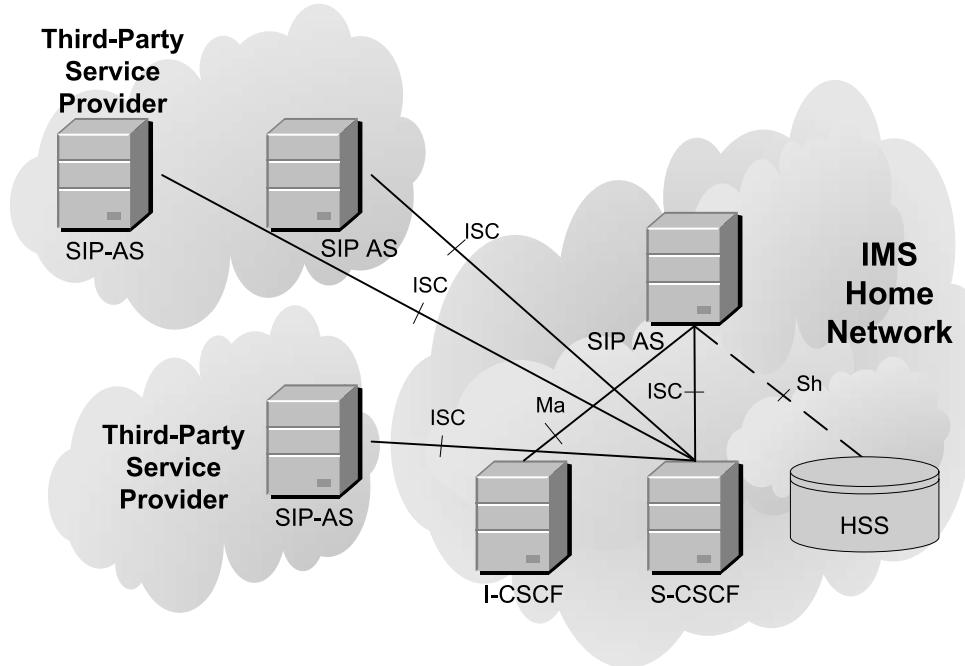


Figure 5.46: The SIP Application Server

5.8.2.2 The OSA-SCS

The Open Service Access–Service Capability Server (OSA-SCS) provides the gateway functionality to execute OSA services in the IMS. Most likely, new IMS services will not be developed in the OSA framework Application Server, but in SIP ASes instead. However, there is an existing base of services implemented in the OSA framework AS. So, in order to provide access to those services from the IMS a gateway is needed. This gateway is the OSA-SCS.

Figure 5.47 shows the OSA-SCS in the context of the IMS. The OSA-SCS provides an interface, external to the IMS, toward the OSA framework AS. This interface is based on the OSA series of specifications defined in 3GPP TS 29.198 [18]. The OSA-SCS AS also provides the *ISC* interface toward the S-CSCF based on SIP, and may use the optional *Sh* interface toward the HSS. We describe the *Sh* interface in Section 7.3.

The OSA-SCS is located in the home network, although the OSA framework allows secure third-party access; therefore, it allows the OSA framework AS to be located in a third-party service provider network.

From the perspective of the S-CSCF, the OSA-SCS appears and behaves as a SIP AS. The S-CSCF cannot differentiate between an OSA-SCS and a SIP AS.

5.8.2.3 The IM-SSF Application Server

The third type of Application Server is the IP Multimedia Service Switching Function (IM-SSF). This AS provides a gateway to legacy service networks that implement CAMEL (Customized Applications for Mobile network Enhanced Logic) services, which are widely

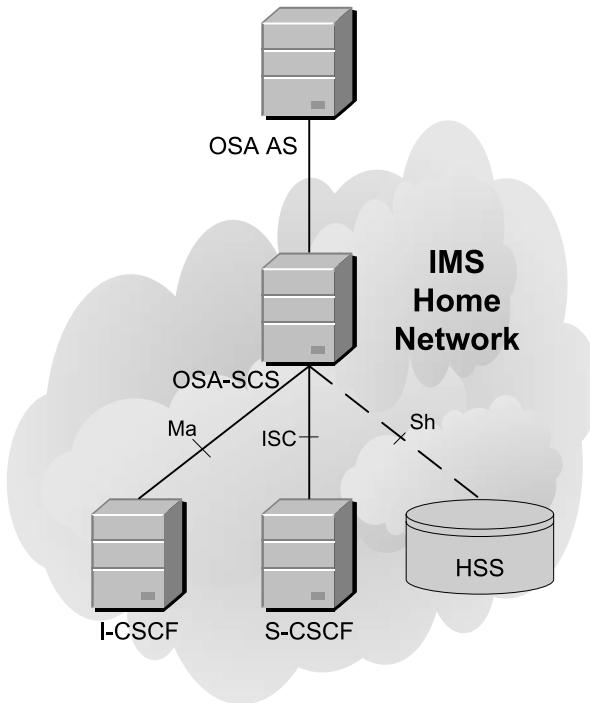


Figure 5.47: The OSA-SCS

deployed in GSM networks. The IM-SSF acts as a gateway between SIP and CAMEL services, thus allowing CAMEL services to be invoked from the IMS.

Figure 5.48 shows the IM-SSF in the context of the IMS. On the IMS side, the IM-SSF interfaces the S-CSCF via the ISC interface, with SIP as the protocol. The IM-SSF also interfaces the HSS with an interface code-named *Si*. The protocol on the *Si* interface is MAP (Mobile Application Part), an existing non-IMS protocol used in GSM networks. MAP is specified in 3GPP TS 29.002 [46]. 3GPP TS 23.278 [3] defines the interactions of CAMEL with the IMS.

Outside the IMS the IM-SSF interfaces the gsmSCF (GSM Service Control Function), which is part of the Camel Service Environment (CSE). The protocol on this interface is CAP (CAMEL Application Part), a non-IMS protocol specified in 3GPP TS 29.278 [1].

The IM-SSF is located in the home network.

From the perspective of the S-CSCF, the IM-SSF appears and behaves as a SIP AS. The S-CSCF cannot differentiate between an IM-SSF and a SIP AS.

5.8.3 The Session Setup Model through Application Servers

Although we have not yet described how the S-CSCF decides when and how to involve a particular Application Server in the signaling path (we will describe it soon in Section 5.8.4), we need to investigate how ASes fit in the whole session setup path, and the different SIP behavior they can have. This description is applicable to all of the different types of AS.

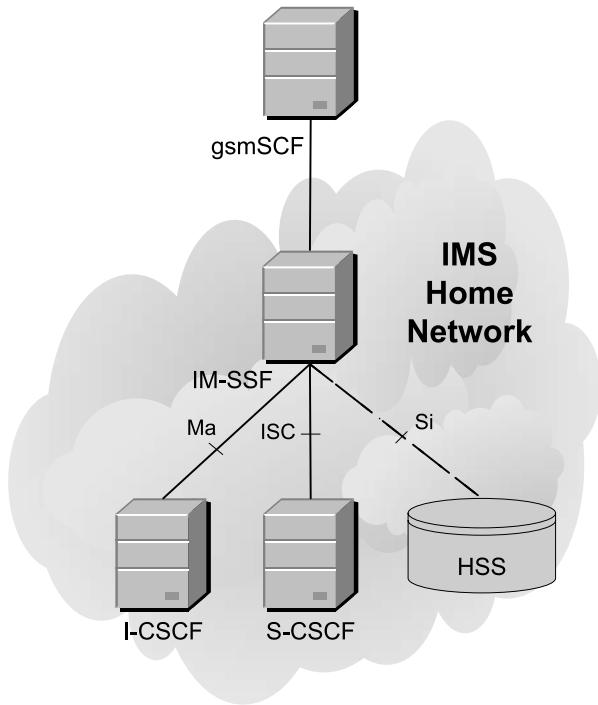


Figure 5.48: The IM-SSF Application Server

From the point of view of SIP an AS can act as either an originating or a terminating SIP User Agent (i.e., an endpoint that originates or terminates SIP), a SIP proxy server, a SIP redirect server, or a SIP B2BUA (Back-to-Back User Agent). An AS may sometimes act as a SIP proxy server and at other times as a SIP User Agent, depending on the service provided to the user. If the AS decides not to provide a service, then it acts as a SIP proxy server. This guarantees that the S-CSCF receives the SIP request and continues with the process. If an AS does not want to provide a service, then it should not record the route, so that it is out of the signaling path once the SIP transaction is complete.

5.8.3.1 Application Server Acting as a SIP User Agent

Figure 5.49 shows a session setup model when the Application Server is acting as a terminating user agent. In this example we have chosen to locate the P-CSCF in the home network, but it could be located in the visited network as well. Figure 5.49 shows an example of an AS acting as a terminating user agent (it is acting as the destination of the session). The service is provided on the originating side (i.e., the service is provided to the caller).

We can see in Figure 5.49 an IMS terminal sending a SIP INVITE request (1) that arrives at the originating P-CSCF and the originating S-CSCF (2). The S-CSCF decides to forward the INVITE to an AS (3). The AS behaves as a SIP UA and responds with a 200 (OK) response (4) that is sent through the S-CSCF (5) and the P-CSCF (6) back to the IMS terminal.

In Figure 5.49 and subsequent figures we show examples in which the SIP request is an INVITE request. However, the SIP request could be any other initial SIP request that is

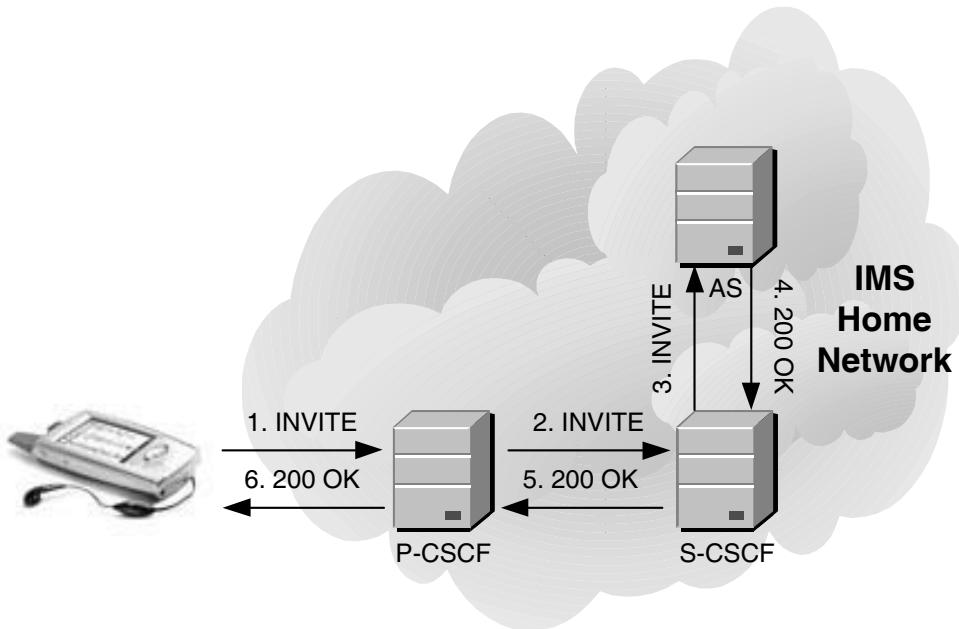


Figure 5.49: An Application Server, acting as a terminating SIP UA, is providing services in the originating call leg

subject to treatment by filter criteria (see Section 5.8.4 for a description of which requests are subject to such treatment, like SUBSCRIBE or PUBLISH requests).

For the sake of clarity, in Figure 5.49 and subsequent figures, we have omitted all the signaling that may take place in between the SIP INVITE request and the 200 (OK) response. For instance, if the SIP INVITE request requires use of the precondition extension, there will be a few more messages in between the INVITE request and the 200 (OK) response.

Figure 5.50 shows a similar scenario in which an AS is also acting as a terminating SIP User Agent. However, in this case the service is provided in the terminating call leg (i.e., to the callee). It is worth noting that the AS effectively “hijacks” the session and the user never receives the SIP message (e.g., the INVITE request). As shown in Figure 5.50, a user agent has sent an INVITE request (1) that is received at the I-CSCF. The I-CSCF forwards the INVITE request (2) to the S-CSCF. The S-CSCF decides to forward the INVITE request (3) to an AS. The AS acts as a user agent and establishes the session; that is, it answers with a 200 (OK) response (4). The response is sent back, in (5) and (6), via the I-CSCF.

An example of a service that uses this model is any service that requires a server to handle the SIP request on behalf of the user. This is the model used in the presence service (e.g., when a watcher subscribes to the presence information of the presentity, or user, because in spite of the SUBSCRIBE request being addressed to the end user, the request is diverted to the presence server). The presence server (which is a specialized AS) treats the request appropriately.

Figures 5.49 and 5.50 show the case when the SIP AS acts as a terminating SIP User Agent (i.e., the AS is replacing or acting as the destination of the session). A different example is shown in Figure 5.51, where the AS is acting as a caller (i.e., the AS initiates

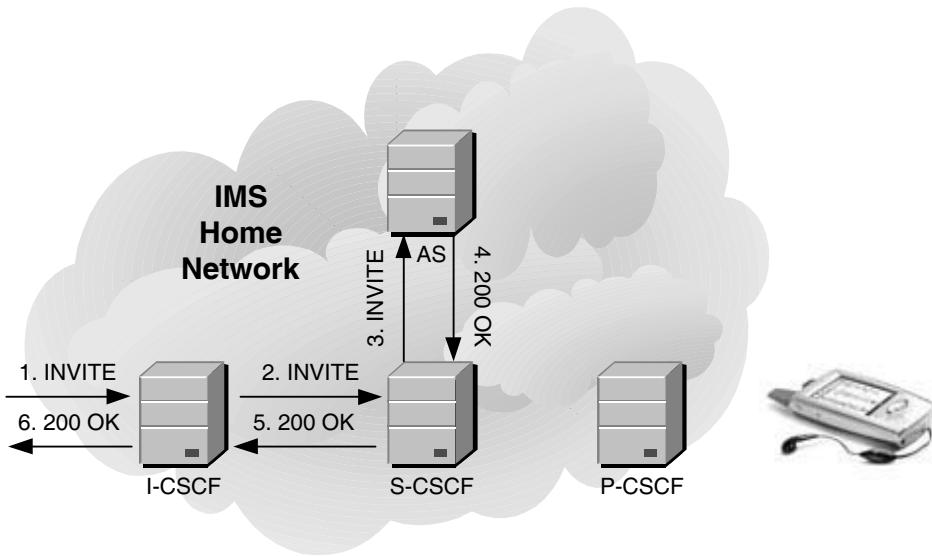


Figure 5.50: An Application Server, acting as a terminating SIP UA, is providing services in the terminating call leg

the SIP session). In this case the service does not have an originating/terminating distinction, because the user receives the session setup.

An example of this service is a wake-up call service, where the AS, at a certain pre-configured time, initiates a session toward the IMS terminal to provide the wake-up call.

5.8.3.2 Application Server Acting as a SIP Proxy Server

In another configuration, an Application Server may need to act as a SIP proxy server to provide the service. The configuration is shown in Figure 5.52 in an originating call leg. An IMS terminal sends a SIP INVITE request (or any other request subject to filter criteria assessment) that traverses the P-CSCF (1) and the S-CSCF (2). The S-CSCF decides to involve an AS, and forwards the INVITE request (3) to that AS. The S-CSCF inserts in the INVITE request a Route header field that points to the AS in the first place and the S-CSCF in the second place. This is to allow the AS to forward the request back to the same S-CSCF (4). Furthermore, the S-CSCF needs to insert some sort of state information in its own SIP URI in the Route header field. This allows the S-CSCF to determine whether the SIP request has already been received from the IMS terminal and sent once to that AS.

Figure 5.53 shows an example of what the Route header field would look like in any SIP request sent to an AS. The header contains two SIP URIs; the first one points to the AS that is receiving the SIP request. This is regular behavior according to SIP loose routing rules. The second URI is pointing to the same S-CSCF, thus allowing the AS to forward the request back to the S-CSCF. There is state information contained in the “username” part of the SIP URI (state34 in our example in Figure 5.53). When the S-CSCF receives the SIP request from the AS, (4) in Figure 5.52, the S-CSCF reads the “username” part of the SIP URI in the Route header field value and is able to determine the point in the process where it needs to continue.

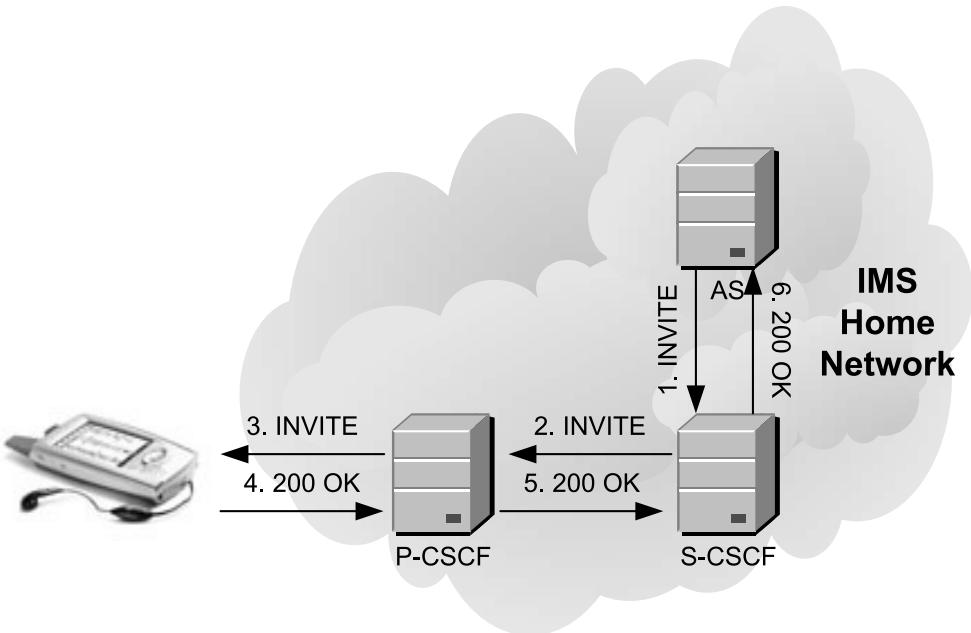


Figure 5.51: An Application Server, acting as an originating SIP UA, is providing services to the user

In our example in Figure 5.53 the S-CSCF writes the state information as part of the “username” part of the SIP URI. The S-CSCF can freely choose any other mechanism to convey this state, such as including a parameter in its own URI or choosing a different port number to receive the request from the AS. The rule is that the S-CSCF writes its own SIP URI. Since the S-CSCF itself is the only entity that will need to parse the “username” at a later point it can encode the state information it wishes to read later.

Last but not least, we need to mention that the S-CSCF is not aware of the SIP behavior of the AS. The S-CSCF, when forwarding a request to an AS, always inserts this double Route header field, no matter whether the AS is operating as a SIP User Agent, a SIP proxy server, etc. If the AS operates as a SIP User Agent the AS simply does not use the Route header field, because the AS answers the request.

Let us return to our examination of Figure 5.52, where we have a SIP INVITE request received at the AS (3). The AS takes different decisions and actions depending on the actual service provided to the user. For instance, the AS needs to decide whether it wants to remain in the signaling path for subsequent SIP signaling related to this session. In other words, the AS needs to decide whether it needs to insert its own SIP URI in the value of the Record-Route header field. The S-CSCF will not forward the remaining SIP requests that belong to the same SIP dialog if the AS does not record its own route. The behavior is different with responses, since they always follow the same path as their corresponding requests traversed. Therefore, responses to this initial request that reached the AS are always forwarded via the AS, irrespective of whether the AS recorded the route or not.

The AS may need to change a header field value in the SIP request. For instance, let us imagine that the AS is providing a speed-dial service, whereby the user only dials a

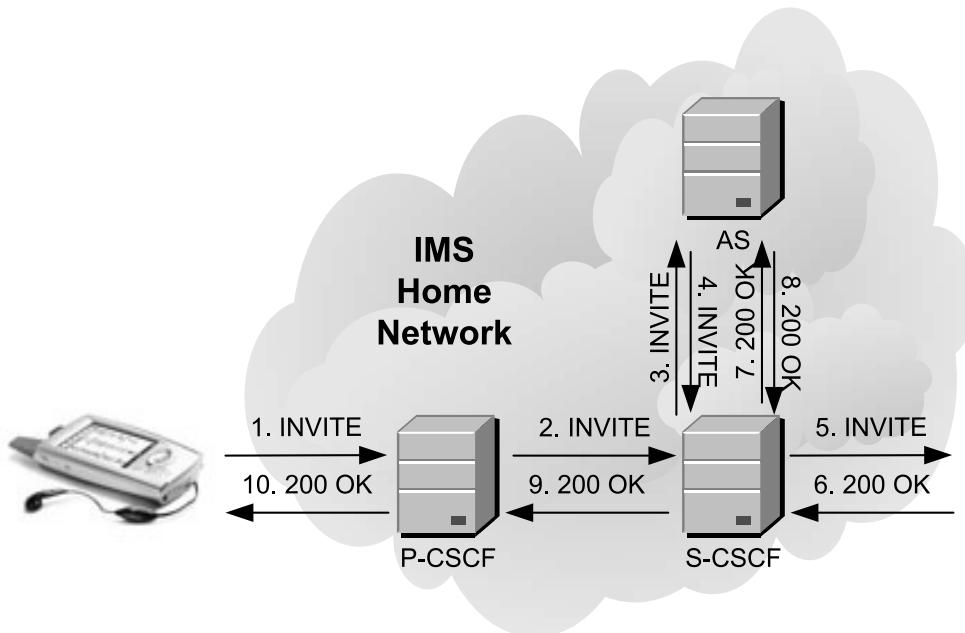


Figure 5.52: An Application Server, acting as a SIP proxy server, is providing services in the originating call leg

Route: <sip:as33.home1.net;lr>, <sip:state34@scscf1.home1.net;lr>

Figure 5.53: The Route header field in a SIP request forwarded to an AS

short string of digits, such as `tel:123;phone-context=example.com`. The AS contains a personalized list per user that expands the dialed number with the real URI (TEL URI or SIP URI). In this case the service the AS provides consists of just rewriting the *Request-URI* and replacing `tel:123;phone-context=example.com` with the real destination URI. There are no other actions that the AS is requested to do. Furthermore, in the speed-dial service example the AS does not even need to *Record-Route*, because the service is provided entirely in the INVITE request.

Eventually, the AS honors the *Route* header field contained in the received INVITE request and forwards the INVITE request, with all the needed changes, to the SIP URI indicated in the *Route* header field; that is, to the S-CSCF (4).

When the S-CSCF receives the INVITE request for the second time (4), because of the inclusion of the state information, it is able to determine the next required action. In the example in Figure 5.52 the S-CSCF forwards the INVITE request toward the callee's network (5). Eventually, a 200 (OK) response is received (6). The S-CSCF, according to regular SIP routing procedures for SIP responses, forwards the 200 (OK) response (7) to the AS. The AS forwards it back to the S-CSCF (8). The S-CSCF forwards it to the P-CSCF (9) which in turns forwards it to the IMS terminal (10).

In the configuration shown in Figure 5.54 the AS is still acting as a SIP proxy server, but it is providing a service to the callee. The INVITE request (1) is received at an I-CSCF.

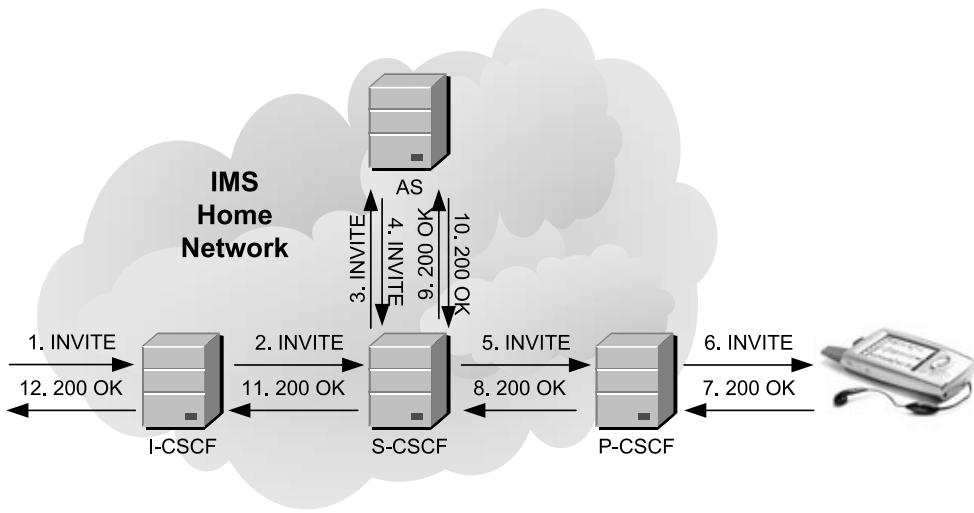


Figure 5.54: An Application Server, acting as a SIP proxy server, is providing services in the terminating call leg

The I-CSCF forwards the INVITE request (2) to the S-CSCF that is handling the user. The S-CSCF decides to involve an AS that may provide a service, so, it forwards the INVITE request (3) to the AS. The AS executes the service, perhaps modifying some headers in the SIP request, being compliant with SIP rules which state that header fields must be modified by a proxy, and then forwards the INVITE request (4) back to the S-CSCF, because of the presence of a Route header field pointing to the S-CSCF. The S-CSCF forwards the INVITE request to the IMS terminal via the P-CSCF, (5) and (6). The responses will traverse the same set of proxies in the reverse direction.

The call-forwarding service is an easy service that can be implemented using this configuration. The AS needs to rewrite the *Request-URI* to point to the new destination URI.

5.8.3.3 Application Server Acting as a SIP Redirect Server

An Application Server may be acting as a SIP redirect server. Figure 5.55 shows the high-level signaling flow for an AS providing services in the terminating call leg (i.e., to the callee). According to Figure 5.55 an I-CSCF in a home network receives an INVITE request (1). The I-CSCF forwards it to the S-CSCF (2). The S-CSCF involves an AS and forwards the INVITE request to it (3). The AS acting as a SIP redirect server generates a 302 (Moved Temporarily) final response (4). The response contains a Contact header field that includes the new URI to contact. The response is forwarded back to the originator, (5) and (6). When the originator of the session receives the 302 (Moved Temporarily) response, it generates a new INVITE request whose *Request-URI* (the destination address) is the Contact header field value received in the 302 (Moved Temporarily) response. This new INVITE request may not even reach the same IMS home network (e.g., if the new destination address belongs to a different domain name).

A typical example of the applicability of SIP redirect servers is the provision of call-forwarding services. They can also be used to provide number portability services. In

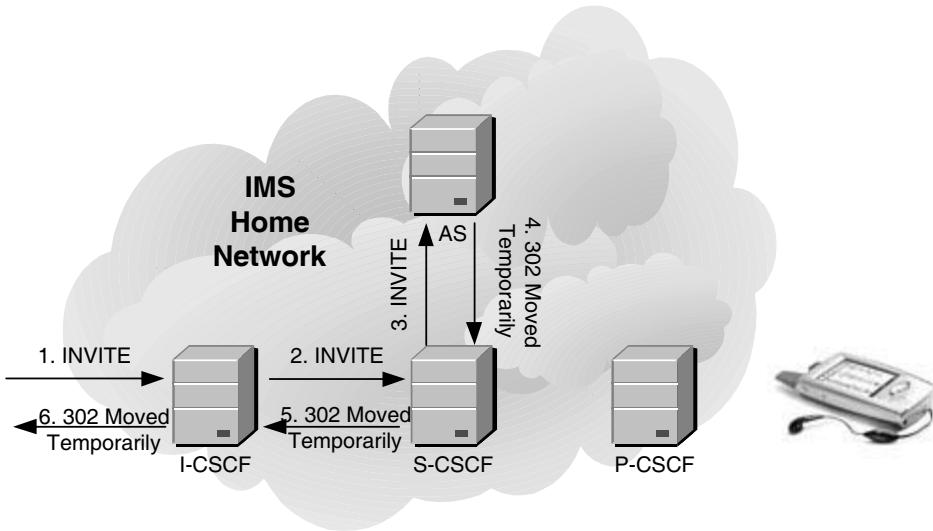


Figure 5.55: An Application Server, acting as a SIP redirect server, is providing services in the terminating call leg

general, SIP redirect servers can be used whenever the session is forwarded somewhere and the operator is not interested in being part of the session.

5.8.3.4 Application Server Acting as a SIP B2BUA

The last mode of operation of an Application Server is the SIP B2BUA (Back-to-Back User Agent). A B2BUA is just two SIP User Agents connected by some application-specific logic. In general, a B2BUA performs similar actions to a SIP proxy server. It receives requests and forwards those requests somewhere else; it receives responses and relays them back to the originating entity. However, there are differences between a SIP proxy and a B2BUA. These differences are related to the type of actions that both are allowed to perform and the consequences of performing those actions.

Figure 5.56 shows the logical view of a SIP B2BUA. A SIP request A is received in one of the SIP User Agents, which will pass the request to the application-specific logic. The application-specific logic is responsible for generating a response A and creating a new SIP request B, which is partly related to request A. Requests A and B are only partly related because the application-specific logic may change any header, including those header fields that a SIP proxy server cannot modify, such as `From`, `To`, `Call-ID`, etc. The application-specific logic may even change the method in the SIP request. The application-specific logic may also change SDP as well, another action that is not allowed to be taken by SIP proxies. The B2BUA can even asynchronously generate a SIP request on one of the call legs without having received any stimulus on the other call leg or create multiple call legs, such as a third-party call controller.

Because a SIP B2BUA is just a collection of SIP User Agents, B2BUAs need to understand all the methods, extensions, etc. that in normal circumstances only two endpoints would need to understand. For example, in the IMS context, a SIP B2BUA

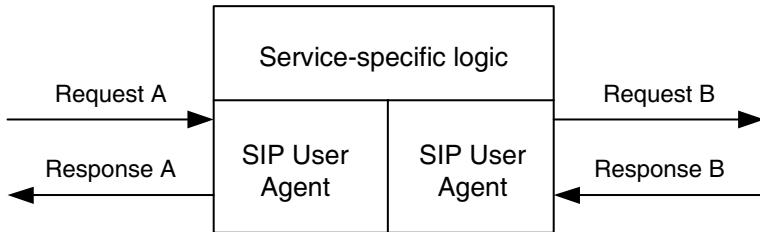


Figure 5.56: Logical view of a SIP Back-to-Back User Agent

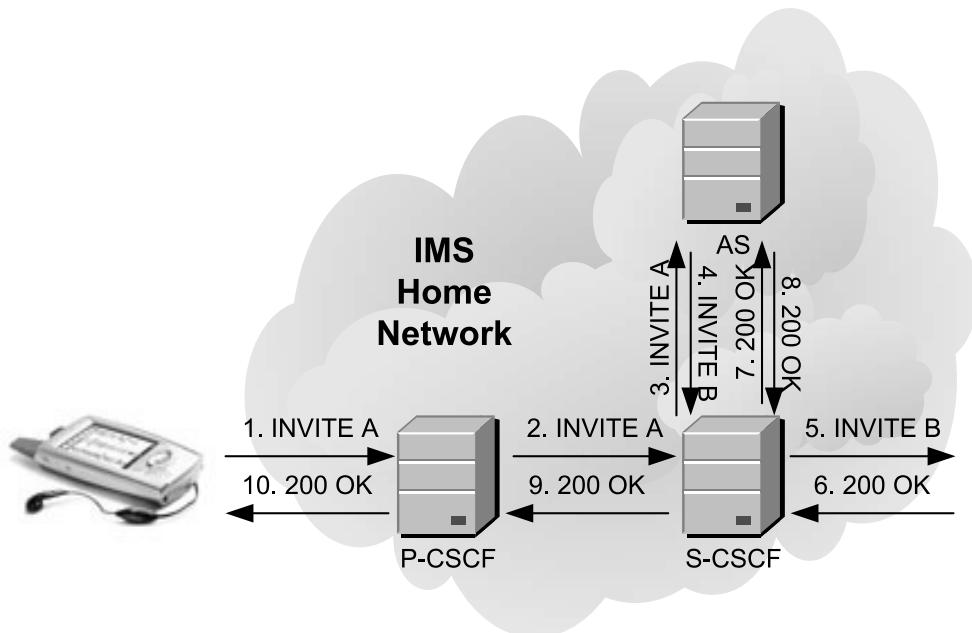


Figure 5.57: An Application Server, acting as a SIP B2BUA, is providing services in the originating call leg

needs to understand the SIP preconditions extension. Proxies need not understand this extension.

Figure 5.57 shows the high-level signaling flow of an AS acting as a B2BUA and providing services to the originating party of a session. In this example “INVITE A” represents the INVITE request received by the AS, and “INVITE B” represents the INVITE request that the AS sends, in an effort to indicate that these INVITE requests are uncorrelated and the only point of correlation is the SIP B2BUA.

The logic used by a B2BUA will also dictate how requests and responses are mapped. One possibility is that responses received on one side of the B2BUA are simply regenerated at the other side. Another is that when INVITE A is received, the B2BUA generates the corresponding response A, and then generates INVITE request B. This behavior depends on the actual service provided to the user.

An example of an AS acting as a B2BUA on the originating call leg is a prepaid AS. The prepaid AS first verifies that the user has enough credit in his account to pay for the cost generated by the session. If the user does not have enough credit the B2BUA will simply reject the call, but if it has enough credit it will proceed. Once the session is established, if the user runs out of credit, the B2BUA sends a BYE request to each of the parties to tear down the session.

Figure 5.58 shows how an AS, acting as a B2BUA, provides services to the callee. We have also distinguished the two different uncorrelated INVITE requests and named them INVITE A and INVITE B. In step (4) the AS generates a new INVITE B that is partly related to INVITE A.

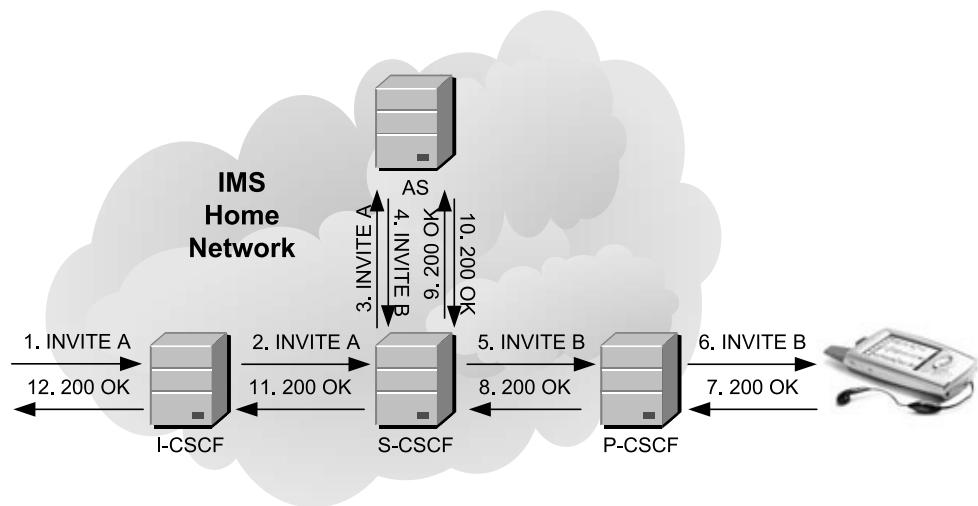


Figure 5.58: An Application Server acting as a SIP B2BUA is providing services in the terminating call leg

A privacy AS is a good example of an AS that, acting as a SIP B2BUA, provides a service to the callee. On receiving a SIP request or response, the B2BUA obfuscates those header fields that reveal information related to the user (e.g., From, Contact, etc.) and may even change the SDP to avoid peers becoming aware of the IP address of the originator of the session. So, the callee cannot see any SIP header, IP address, etc. related to the caller, and vice versa.

5.8.4 Filter Criteria

Filter criteria are among the most important pieces of user information stored in the network, because they determine the services that will be provided to each user. Filter criteria contain a collection of user-related information that helps the S-CSCF to decide when to involve (e.g., forward the SIP request to) a particular Application Server to provide the service.

3GPP TS 23.218 [39] specifies, for historical reasons, two sets of filter criteria: *initial filter criteria* and *subsequent filter criteria*. Only initial filter criteria are used. Subsequent filter criteria constituted a theoretical exercise, since the implementation of subsequent filter criteria by the S-CSCF would conflict with the SIP routing rules for proxies.

Initial filter criteria are supposed to be evaluated with those SIP initial requests that either create a dialog or are stand-alone requests (i.e., those that are not subsequent requests within a SIP dialog). For instance, the S-CSCF evaluates initial filter criteria when it receives a first SUBSCRIBE request, INVITE, OPTIONS, or any such requests that either create a dialog or are sent outside any dialog. The S-CSCF does not evaluate initial filter criteria when it receives a PRACK, NOTIFY, UPDATE, or BYE request, since they are always sent as part of an existing SIP dialog.

The subsequent filter criteria concept was that the S-CSCF would evaluate subsequent filter criteria when it received a subsequent request within a SIP dialog. However, the result of evaluating subsequent filter criteria would lead the S-CSCF to forward a subsequent SIP request to an AS, an action that is in contradiction to the routing procedures for a subsequent request in a SIP proxy. In addition, in the event that an AS received, this subsequent request the AS would have not (most likely) received the initial SIP request that created the SIP dialog. Therefore, the AS would reject the request and would ignore it. Consequently, the decision was made not to implement subsequent filter criteria.

The only implemented filter criteria in the specifications are the initial filter criteria. As the subsequent filter criteria do not exist, the terms *initial filter criteria* and *filter criteria* are usually interchangeable.

The HSS stores all the data related to a user in a data structure named the *user profile*. Figure 5.59 shows a high-level simplified structure of the user profile. We describe the user profile in detail in Section 7.2.3, but for the time being it is enough to mention that the user profile contains the *Private User Identity* to which the user profile is applicable and one or more *service profiles*. Each service profile contains one or more *Public User Identities* to which the service profile is applicable and zero or more *filter criteria*. We focus on filter criteria in this section.

When the user registers with the S-CSCF, the S-CSCF contacts the HSS and downloads the user profile that includes the filter criteria. So, filter criteria are available in the S-CSCF at the moment the user registers.

Filter criteria determine the services that are applicable to the collection of Public User Identities listed in the Service profile. Filter criteria are further structured as sketched in Figure 5.60.

The first field in the filter criteria structure is the *Priority*. The Priority field determines the order in which these filter criteria will be assessed, compared with the remaining filter criteria that are part of the same service profile. The S-CSCF will first choose filter criteria that have higher priority, indicated by a lower figure in the Priority (i.e., priority 1 is the highest priority). After evaluating it the S-CSCF continues with the filter criteria of the next Priority number (e.g., 2, 3, etc.). The Priority field of filter criteria is a unique number with respect to all the filter criteria that belong to the same service profile. In any case, numbers do not need to be consecutive. For instance, the highest priority of the first set of filter criteria that the S-CSCF evaluates could be Priority 100. The second could be Priority 200 and so on. This leaves room to accommodate new services in between them.

After the Priority field, there can be zero or one *Trigger Points*. A Trigger Point is an expression that needs to be evaluated in order to determine whether or not the SIP request will be forwarded to a particular AS. A trigger point is a collection of individual filters called *Service Point Triggers*. For instance, a Trigger Point can express

```
(Method = INVITE) AND (URI]Request-URI = sip:user@example.com)
```

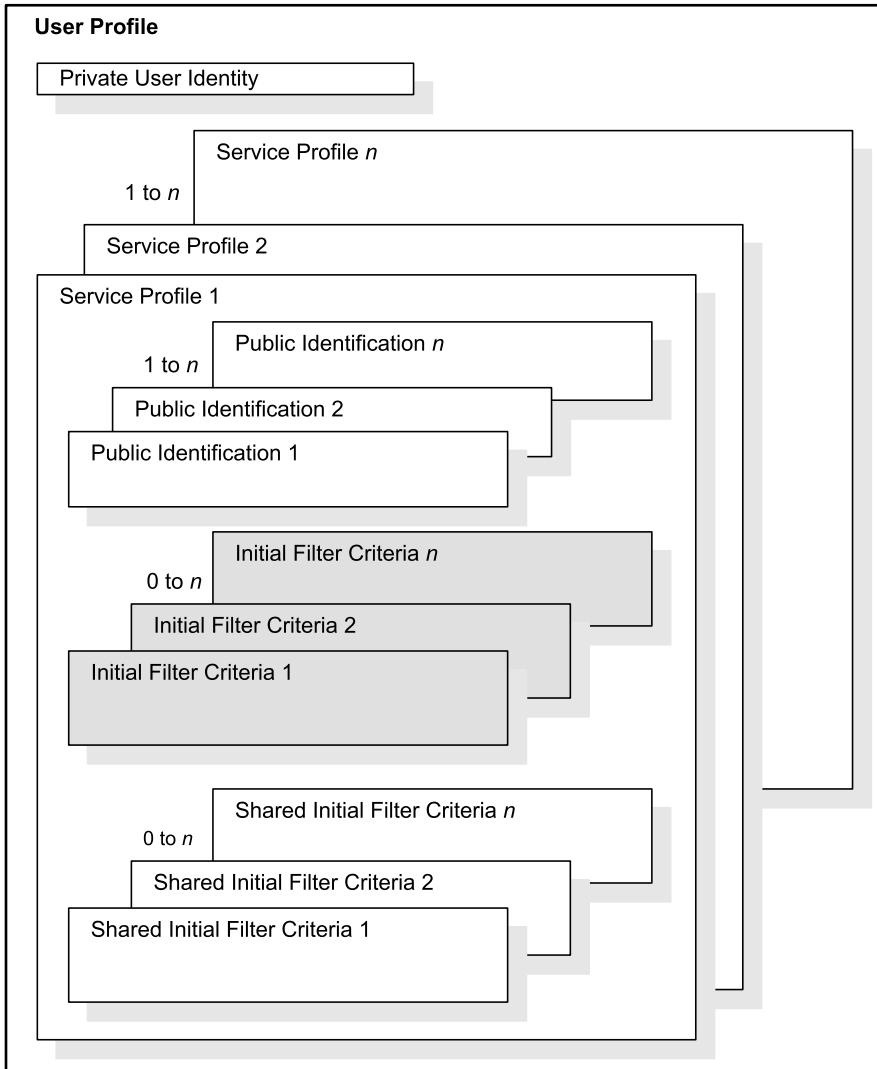


Figure 5.59: Simplified representation of the structure of the user profile

In this example, one Service Point Trigger is Method = INVITE and the other is Request-URI = `sip:user@example.com`.

The Service Point Trigger allows us to access the information stored in different fields of the SIP request:

- the value of *Request-URI*;
- the method of the SIP request (e.g., INVITE, OPTIONS, SUBSCRIBE, etc.);
- the presence or absence of any SIP header;
- a partial or full match between the contents of any SIP header;

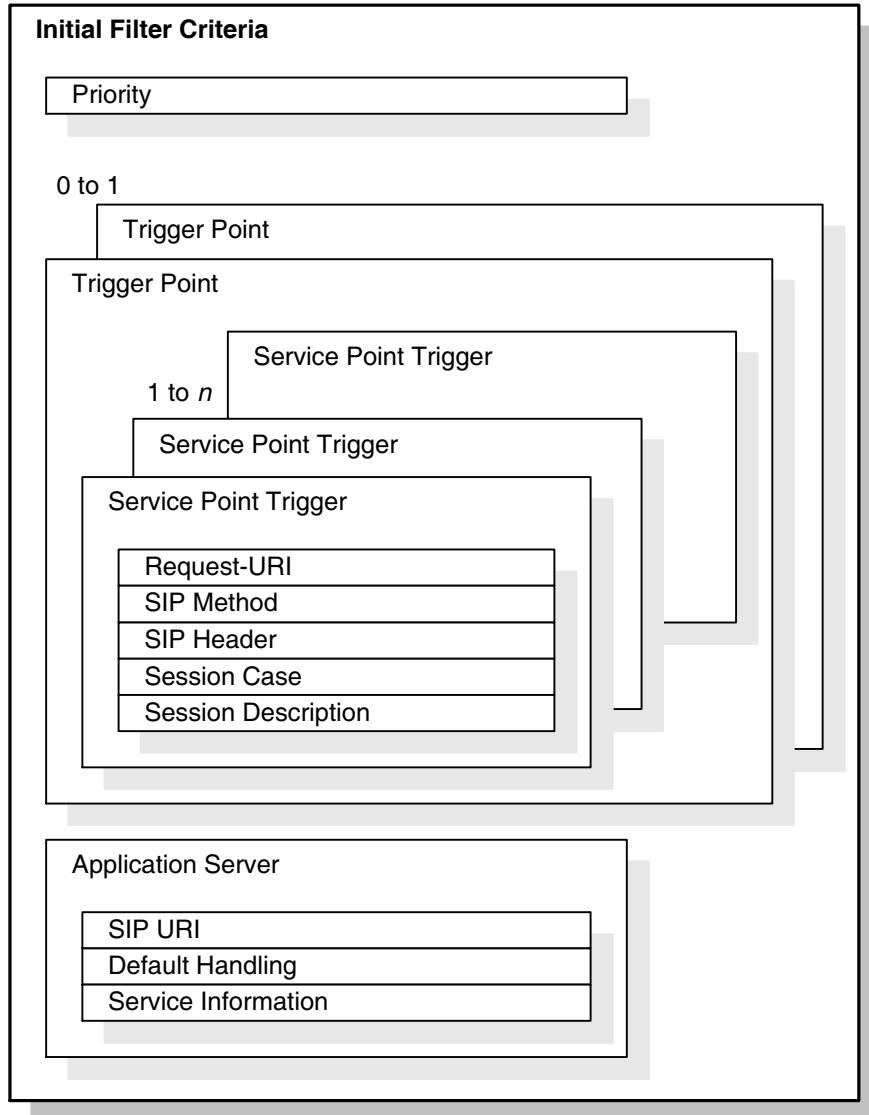


Figure 5.60: Structure of initial filter criteria

- the session case (i.e., whether the SIP request is originated by the served user, addressed to the served registered user, or addressed to the served unregistered user);
- the session description (i.e., any partial or full match on any SDP line);

If there is no Trigger Point, then any SIP request is unconditionally forwarded to the AS.

After the Trigger Points, which include one or more Service Point Triggers, the initial filter criteria contain the AS SIP URI. This is the address of the AS that will receive the SIP request if the conditions described in the Trigger Points are met. There is a *Default*

Handling field that indicates the action to be taken if the S-CSCF cannot contact, for whatever reason, that AS. The possible actions are to continue processing the SIP request or to abort the process.

The *Service Information* field contains some transparent data (i.e., transparent to the HSS and S-CSCF) that the AS may need to process the request. The use of this field is restricted to SIP REGISTER requests or any other request where the S-CSCF is acting as a SIP User Agent Client. The reason is that these data are appended in a body to the SIP request. This action is not allowed in SIP proxies. Therefore, the only possibility of using this information is when the S-CSCF, due to initial filter criteria triggering, acting as a SIP User Agent Client generates, a third-party SIP REGISTER request to the AS. Such a REGISTER request can contain the *Service Information* (if the AS needs it), whose purpose is the transfer of the IMSI to the IM-SSF of the subscriber, so that the IMSI can be used by the IM-SSF.

Lastly, the user profile is encoded using the Extensible Markup Language (XML). An XML schema defining initial filter criteria is specified in 3GPP TS 29.228 [40]. Initial filter criteria are transported from the HSS to the S-CSCF over Diameter messages (Diameter is explained in Section 6.3, and the usage of Diameter in IMS in Section 7.2).

5.8.5 An Example of Service Execution

We illustrate the execution of a service, including the filter criteria that trigger that service, with an example. Let us imagine a service provided to a particular user (`sip:goodguy@example.com`). The service automatically diverts a session setup attempt to an answering machine (e.g., MRF) when the caller is listed in a black list (e.g., `sip:badguy@example.com`). In order to model the service, it is required that the service profile applicable to `sip:goodguy@example.com` contains initial filter criteria that have a Trigger Point that filters all the terminating INVITE requests whose caller is the bad guy. The Trigger Point is represented as

```
(method = INVITE) AND
(P-Asserted-Identity = sip:badguy@example.com) AND
(Session Case = Terminating)
```

When the conditions are met, the S-CSCF should send the request to an Application Server identified by its SIP URI: `sip:as33.example.com`. The AS contains the logic to provide the diversion service to the MRFC, instructing the MRFC to play a pre-recorded announcement.

If we assume that the good guy in our example has only subscribed to this service, his user profile encoded in XML would look like the one represented in Figure 5.61.

Figure 5.62 shows the example in action. The bad guy sends an INVITE request (1) that is routed to the S-CSCF (2). The S-CSCF evaluates all the filter criteria that are part of the good guy's service profile. In this example we assume that there is a single filter criterion, which instructs the S-CSCF to forward to AS `sip:as33.example.com` those terminating INVITE requests that are sent by the bad guy (i.e., whose P-Asserted-Identity is `sip:badguy@example.com`).

The S-CSCF inserts a double Route header in the INVITE request, one value being the SIP URI of the AS and the other its own S-CSCF SIP URI, and then the S-CSCF forwards the INVITE request (3) to the AS. The AS receives the INVITE request and applies the service. In this case, applying the service means forwarding it to a new destination, which

```
<?xml version="1.0" encoding="UTF-8"?>
<testDatatype xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <IMSSubscription>
        <PrivateID>privategoodguy@example.com</PrivateID>
        <ServiceProfile>
            <PublicIdentity>
                <Identity>sip:goodguy@example.com</Identity>
            </PublicIdentity>
            <InitialFilterCriteria>
                <Priority>0</Priority>
                <TriggerPoint>
                    <ConditionTypeCNF>1</ConditionTypeCNF>
                    <SPT>
                        <ConditionNegated>0</ConditionNegated>
                        <Group>0</Group>
                        <Method>INVITE</Method>
                    </SPT>
                    <SPT>
                        <ConditionNegated>0</ConditionNegated>
                        <Group>0</Group>
                        <SIPHeader>
                            <Header>P-Asserted-Identity</Header>
                            <Content>
                                "sip:badguy@example.com"
                            </Content>
                        </SIPHeader>
                    </SPT>
                    <SPT>
                        <ConditionNegated>0</ConditionNegated>
                        <Group>0</Group>
                        <SessionCase>1</SessionCase>
                    </SPT>
                </TriggerPoint>
                <ApplicationServer>
                    <ServerName>sip:as33.example.com</ServerName>
                    <DefaultHandling>1</DefaultHandling>
                </ApplicationServer>
            </InitialFilterCriteria>
            <ServiceProfile>
        </IMSSubscription>
    </testDatatype>
```

Figure 5.61: An example of a user profile

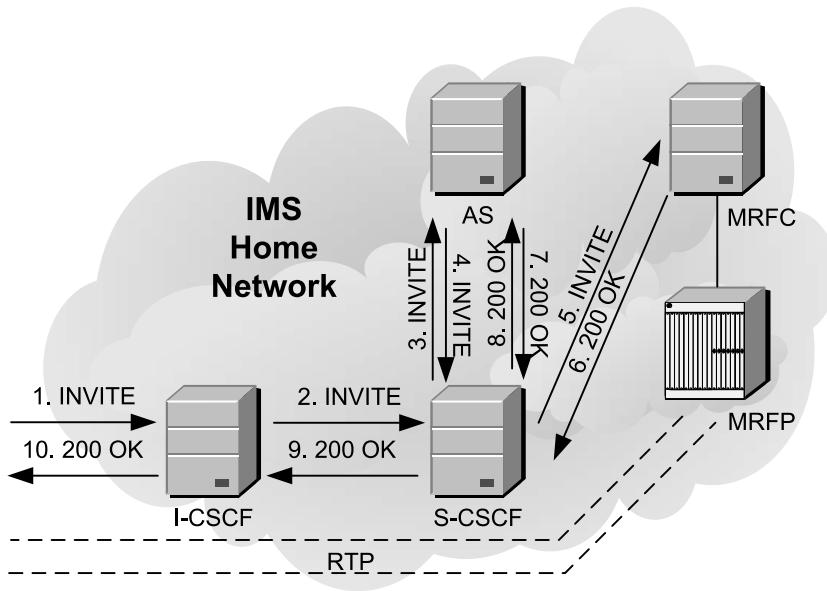


Figure 5.62: An example of a service

is the MRFC. This can be accomplished by the AS acting as a SIP proxy and replacing the *Request-URI* in the forwarded INVITE request.

Therefore, the AS replaces the *Request-URI* of the SIP INVITE request with the SIP URI of the MRFC, `sip:announcementBadGuy@mrfc.example.com`. The SIP URI contains a username part in the URI that gives a hint to the MRFC about which announcement to play. The AS honors the *Route* header in the received INVITE request (3) and forwards the INVITE request to the S-CSCF (4).

On receiving the INVITE request again, the S-CSCF detects that the *Request-URI* has changed with respect to the INVITE request initially received, so it does not assess other possible filter criteria. Instead, it forwards the request to its new destination, in this case the MRFC (5). The MRFC examines the username part contained in the *Request-URI* in order to find out what announcement to play. In this case, “`announcementBadGuy`” is a hint to the MRFC to play the appropriate stored announcement. The MRFC eventually sends a 200 (OK) response back to the originator. It also sends H.248 commands to the MRFP to play the stored announcement. The MRFC plays the announcement using the negotiated codec in the SDP.

In this example the tasks of the Application Server are relatively simple. In real life scenarios, ASes carry out much more complex tasks. For instance, they can provide an administrative HTTP interface that allows the user to log into the AS and manage their own black list. Every time the user adds or removes an identity from their black list, the AS retrieves the filter criteria from the HSS, over the *Sh* interface, modifies them with the new addition or removal, and sends them back to the HSS. The HSS pushes the updated user profile (which includes the new updated filter criteria) to the S-CSCF, so the S-CSCF applies the new configuration of the service in real time.

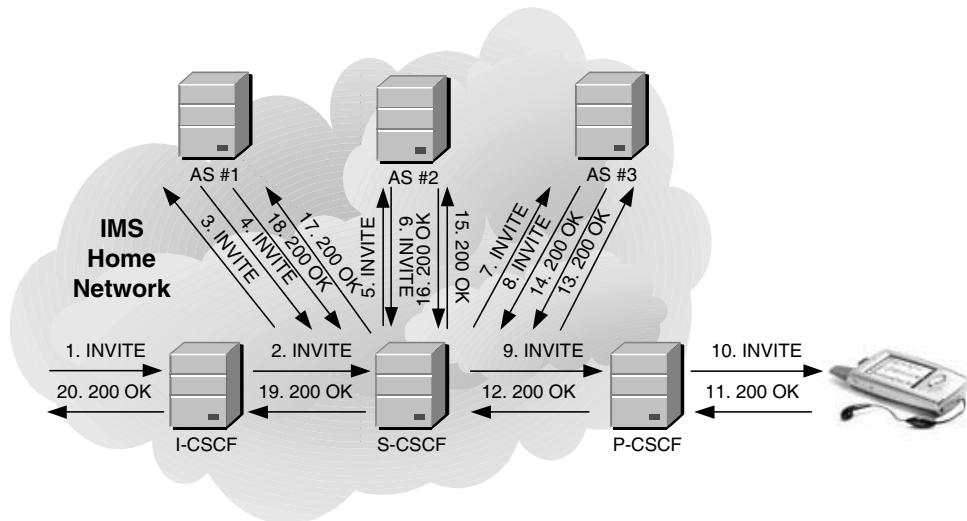


Figure 5.63: A few Application Servers providing services

So far we have shown a simple example where only a single AS is involved in session setup. Typically, there will be several services that potentially can be provided to a user. So, in most cases there will be several ASes involved in the signaling path. Figure 5.63 shows a more complex example, where the S-CSCF is evaluating all the filter criteria for the user and, as a result of this evaluation, several ASes receive the SIP INVITE request. In the example in Figure 5.63, all the ASes are acting as SIP proxies, but any other combination is possible as well.

According to Figure 5.63, the S-CSCF receives an INVITE request (2), evaluates the filter criterion that has highest priority (the one with the lowest figure in the priority field), and forwards the INVITE request (3) to AS #1. When the S-CSCF receives back the INVITE request (4) the S-CSCF evaluates the next filter criterion with the next priority and, as a result, forwards the INVITE request (5) to AS #2. The S-CSCF repeats the operation with the next filter criterion (ordered by priority) when it next receives the INVITE request (6). As a result of this last filter criteria evaluation, the S-CSCF forwards the INVITE request (7) to AS #3. Once the S-CSCF receives back the INVITE request (8) and once the S-CSCF has evaluated all the different filter criteria, the S-CSCF forwards the INVITE request to the P-CSCF and the IMS terminal, (9) and (10). Of course, at any time the S-CSCF could have evaluated a filter criterion for which there was not a match in the Service Point Triggers. Consequently, the S-CSCF would have not forwarded the INVITE request to the AS indicated in that filter criterion.

Figure 5.63 indicates a few important aspects about filter criteria. First, the order in which services are executed is important. For instance, if AS #1 were an answering machine service that was automatically connected at some time of day, then AS #1 would act as a SIP User Agent and, consequently, AS #2 and AS #3 would not receive the INVITE request. A second aspect is that every time a new AS is involved an extra session setup delay is added to the global session setup delay. If a large number of ASes were involved, then the caller might give up because of the delay in session setup time.

5.9 Changes due to Next Generation Networks (NGN)

5.9.1 New SIP Functionality in NGN

The new functionality required by IMS to operate in an NGN comprises the usage of a few new SIP header fields, namely the `History-Info` header field and the SIP URIs for Applications, specified in RFC 4458 [192]. These header fields had been also adopted by 3GPP IMS.

The `History-Info` header field, specified in RFC 4244 [81], allows proxies to record retarget or redirect information in SIP requests and responses. This allows a SIP UA to find out if the original SIP requests issued by the caller have been redirected or retargeted by network elements.

IMS already provided a small similar piece of information in the format of the `P-Called-Party-ID` header field. We have already described the `P-Called-Party-ID` header field in Section 5.7. In short, this header field allows SIP requests to retain the public user identity included in the `Request-URI` when the S-CSCF retargets and replaces the `Request-URI` with the contact information of the user. The limitation of the `P-Called-Party-ID` header field lies in the fact that it is only applicable to the last retarget, i.e., the one that takes place at the terminating S-CSCF that has got contact information from the user's registration.

The `History-Info` header field not only records this last retarget, but also any other previous retarget that could have taken place, for instance, because of forwarding from one user to another, or due to redirections. In this way, both the caller and callee can gather information about the history of transformations that the `Request-URI` of a SIP request suffered en route to its destination.

IMS in NGN brings two new ways of authenticating users. On one side, NGN allows users to be authenticated with traditional HTTP Digest Authentication, which is specified in RFC 3261 [286]. The adoption of traditional HTTP Digest Authentication requires a few changes in the `Cx` interface, so that the I-CSCF and S-CSCF can transfer HTTP Digest parameters to the HSS. Notice that HTTP Digest Authentication merely requires a username and a password, so it does not require a UICC (sometimes called a SIM) card or similar in the terminal.

The other authentication mechanism that NGN brings to IMS is the so-called *NASS-bundled authentication*. In essence, the NASS-bundled authentication mechanism consists in authenticating the user at the IP level with the NASS subsystem when he connects his ADSL modem or router, and then exporting that authentication to the upper IMS layer. We can think of the NASS-bundled authentication as a kind of *line authentication*, because the user or his devices are never authenticated; instead, their fixed line, via their ADSL modem or router, is authenticated, and then that authentication is reused in IMS. In fact, this is similar to the plain old telephone lines, where the physical line is authenticated and allocated with a phone number and a billing identity, no matter which terminal or user is connected at the other side of the line. Like the HTTP Digest Authentication, the NASS-bundled authentication mechanism does not require a UICC card in the terminal.

When NASS-bundled authentication is applied, the terminal builds a `REGISTER` request that contains his Private User Identity in the `Authorization` header field. Then the terminal sends the `REGISTER` request to the P-CSCF. The P-CSCF contacts the NASS subsystem and requests the DSL location assigned to the IP address where the `REGISTER` request was received from. The DSL location is a kind of unique line identifier for each DSL customer in the network. Then the P-CSCF inserts a `P-Access-Network-Info` header field

that contains two new parameters: `dsl-location` and `network-provided`. The former contains the DSL location (or line identifier received from the NASS). The latter is just a flag to indicate that this information has been supplied by the network, as opposed to regular `P-Access-Network-Info` header fields, which are normally supplied by the terminal. Then the P-CSCF forwards the REGISTER request to the S-CSCF, which contacts the UPSF (or HSS) to retrieve user information. The user data returned by the UPSF contains, among other things, the DSL location token (or line identifier) of this particular user. If the DLS location received from the UPSF matches the one inserted by the P-CSCF, then it means that the user is using his regular DSL line, their line has been authenticated, and thus the user can also be authenticated at the IMS layer.

As a consequence of the NASS-bundled authentication of the HTTP Digest Authentication, no security associations are created between the terminal and the P-CSCF.

5.9.2 *Unneeded IMS Functionality in NGN*

The IMS in general, and the 3GPP SIP profile specified in 3GPP TS 24.229 [37], provide optimizations for access to the IMS from low-bandwidth environments or terminals with limited memory capabilities. These limitations are not usually present when the IMS is accessed from a fixed broadband access. Therefore, the IMS contains some functionality that is disabled when the IMS operates in an NGN environment.

One of these unneeded functions is compression of SIP messages, also known as SigComp. We described SigComp in Section 4.16. Since the goal of SigComp is to transmit a SIP message faster over a low-bandwidth channel, and since NGN assumes broadband access, there is no need to compress SIP.

Another function that is not really required is the extended timers that the IMS has set for the IMS terminal and the P-CSCF. The P-CSCF and the IMS terminal implement longer timers for SIP than those recommended by RFC 3261 [286]. This would allow an IMS terminal behind a low-bandwidth channel to avoid retransmissions of SIP requests and responses. Certainly, these extended timers are not required if the access is a high-bandwidth channel.

5.10 Interworking

The IMS will become even more attractive when its users are able to communicate with persons located in the PSTN (Public Switched Telephone Network) and on the Internet. Release 5 specifications do not define any type of interworking with other networks, but Release 6 specifications define interworking with the PSTN (3GPP TS 29.163 [38]) and with non-IMS SIP-based networks (3GPP TS 29.162 [4]).

5.10.1 *SIP–PSTN Interworking*

Even though the PSTN offers video services in addition to the traditional audio calls, the existing work on SIP–PSTN interworking focuses on audio-only calls. So, we will describe how to establish audio sessions between SIP User Agents and PSTN terminals.

Audio calls in the PSTN are established using two types of signaling: network-to-network signaling and user-to-network signaling. Network-to-network signaling (e.g., SS7) is used between telephone switches, while user-to-network signaling (e.g., DSS-1) is used between terminals and their local exchanges. We focus our description on the interworking between

SIP and network-to-network signaling protocols, because this interworking is more likely to be used in the IMS in the future.

There are two levels of interworking in a call between the PSTN and a SIP network: the signaling and the media levels. They are usually handled by a distributed gateway with three components: signaling gateway, media gateway controller, and media gateway (the architecture for PSTN-to-SIP interworking is specified in RFC 3372 [313]). Figure 5.64 shows a logical gateway with these three elements.

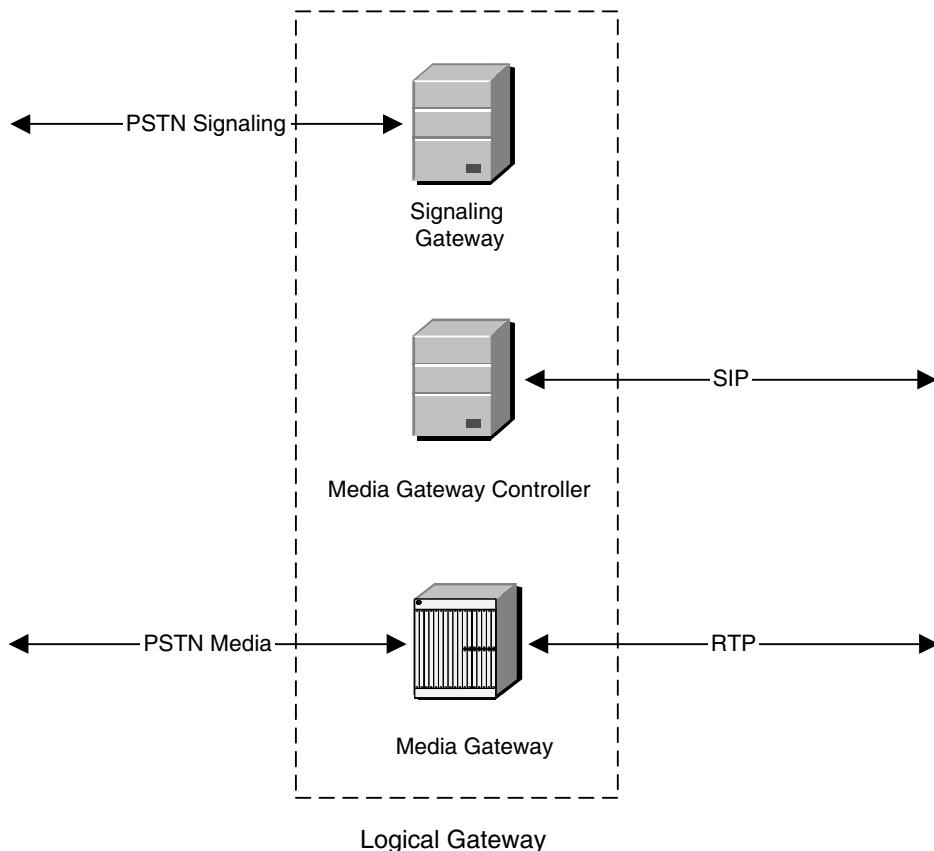


Figure 5.64: Architecture of a logical gateway

The signaling gateway receives PSTN signaling and sends it to the media gateway controller over an IP network (and vice versa). The media gateway controller performs the mapping between PSTN signaling and SIP, and controls the media gateway (ISUP-to-SIP interworking is specified in RFC 3398 [109] and RFC 3578 [110]). The media gateway performs media transcoding, if needed, and encapsulates the media received from the PSTN side in RTP packets (and vice versa).

The traffic between the signaling gateway and the media gateway controller is typically transported over SCTP (specified in RFC 2960 [308]). Keeping these two elements separate allows a single media gateway controller to handle the traffic of many signaling gateways that

are usually physically distributed. Nevertheless, these two elements are sometimes co-located (i.e., they are implemented in the same physical box). This makes it unnecessary to use any transport protocol between them. Media gateway controllers use control protocols such as H.248 [189] or MGCP (Media Gateway Control Protocol, specified in RFC 2705 [75]), to control media gateways.

5.10.1.1 Gateway Architecture in the IMS

The IMS uses the gateway architecture we have just described. Figure 5.65 shows the different functions and interfaces that are involved in interworking with the PSTN (specified in 3GPP TS 29.163 [38]).

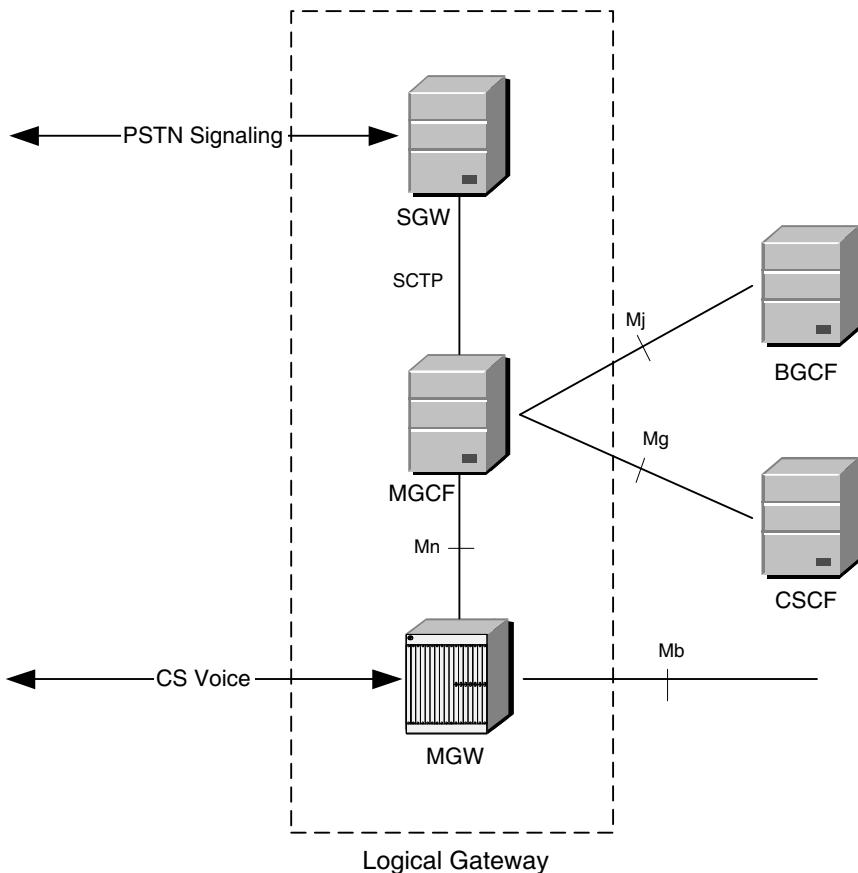


Figure 5.65: IMS–PSTN interworking architecture

The SGW (signaling gateway) exchanges ISUP or BICC over SCTP with the MGCF (Media Gateway Control Function). The MGCF performs BICC/ISUP-to-SIP interworking (and SIP-to-BICC/ISUP) and exchanges SIP signaling with the CSCFs using the *Mg* interface. CSCFs receiving SIP signaling from an MGCF regard the SIP signaling as coming from an S-CSCF.

The MGCF interacts with the MGW (media gateway) through the *Mn* interface, which is based on H.248 [189]. (The *Mn* interface is described in 3GPP TS 29.332 [44].)

The MGW performs media conversions between the PSTN and the IMS. It exchanges RTP packets with the IMS over the *Mb* interface and exchanges circuit-switched voice data with the PSTN.

5.10.1.2 The BGCF

When a session terminates in the PSTN the BGCF (Breakout Gateway Control Function) determines which MGCF handles it. The BGCF receives a request from an S-CSCF and, based on an analysis of the destination address and on any agreements the operator may have for calls terminating in the PSTN, the BGCF decides whether the session should be handled by a local MGCF or by a remote MGCF.

If the session is to be handled by a local MGCF, the BGCF relays the request to one of the MGCFs in its network, as shown in Figure 5.66. The BGCF may or may not Record-Route. If it does, it remains in the signaling path for the rest of the session. If it does not Record-Route, it steps out from the signaling path. The BGCF in Figure 5.66 does not Record-Route.

If the session is to be handled by a remote MGCF, the BGCF relays the request to another BGCF in the remote network, as shown in Figure 5.67. The BGCF in the remote network chooses an MGCF and relays the request to it. None of the BGCFs in Figure 5.67 Record-Route.

Sessions originating in the PSTN and terminating in the IMS do not traverse the BGCF, because MGCF selection is performed in the circuit-switched domain instead. Figure 5.68 illustrates this point. The MGCF sends the initial INVITE request to an I-CSCF because the MGCF does not know which S-CSCF has been assigned to the destination user.

5.10.2 Interworking with Non-IMS SIP-based Networks

Interworking between the IMS and non-IMS SIP-based networks (specified in 3GPP TS 29.162 [4]) is still not mature. There is still a need to define the behavior of IMS entities when a SIP entity on the Internet does not support some of the extensions used in the IMS (e.g., preconditions).

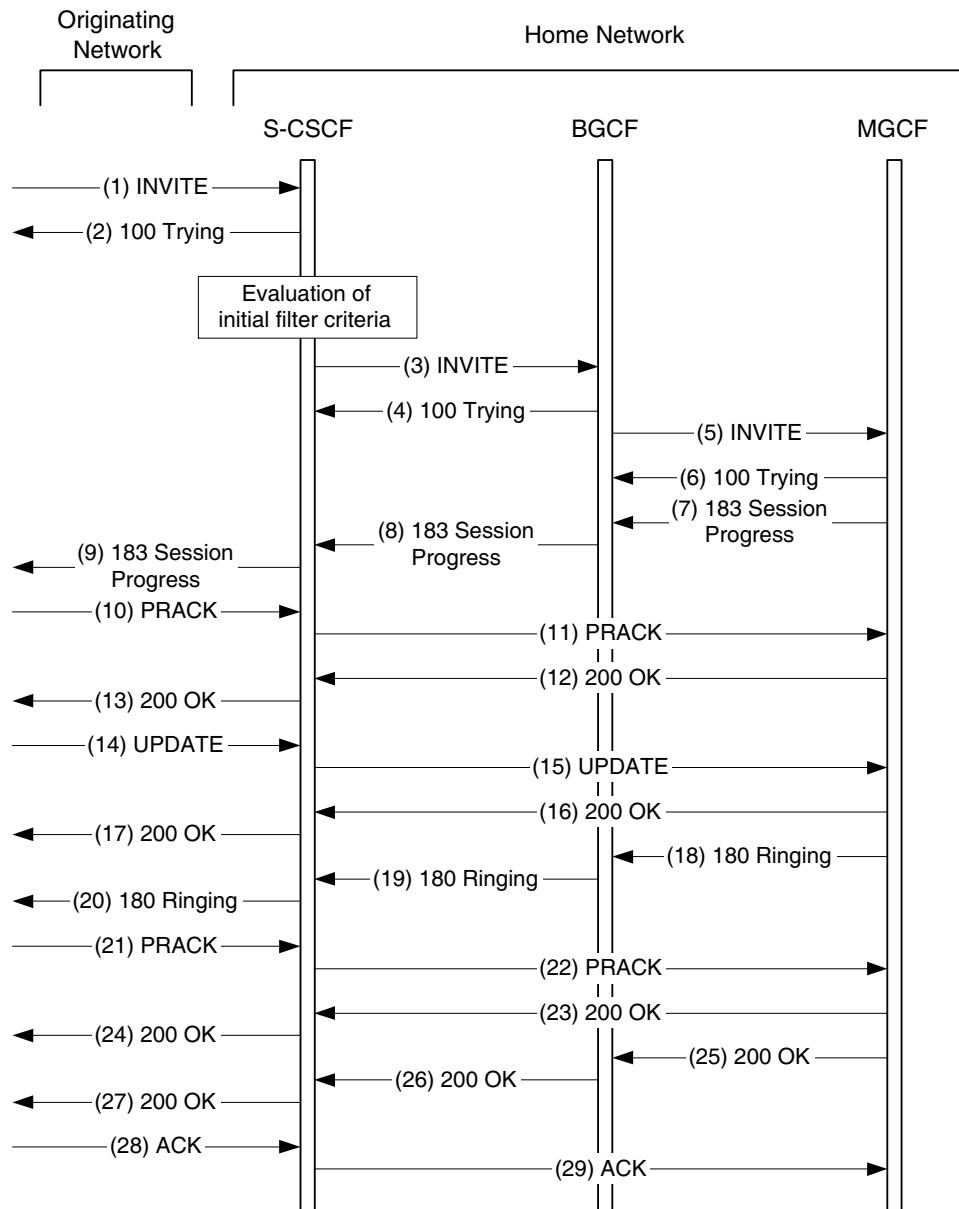
In any case, one of the most important issues in this interworking is how to establish sessions between IPv6 IMS terminals and IPv4 SIP User Agents on the Internet. Let us now look at how the IETF resolves this issue. The IETF IPv4/IPv6 translation mechanisms we will describe may be adopted by 3GPP and 3GPP2 in the future.

5.10.2.1 IPv4/IPv6 Interworking

Although IPv6 is the future of the Internet, IPv4 is currently the most widespread network-layer protocol. Even so, IMS terminals cannot talk directly to IPv4 Internet hosts, because the IMS is based on IPv6.

IPv6 user agents need to exchange both signaling and media with IPv4 user agents on the Internet. The signaling part is handled by SIP and the media part is typically handled using NATs (Network Address Translators), as shown in Figure 5.69.

IPv6–IPv4 interworking at the SIP layer is handled by proxy servers at the edge of every domain, which need to have dual IPv4–IPv6 stacks. Proxies within an IPv6 (or an IPv4)

**Figure 5.66:** Session handled by a local MGCF

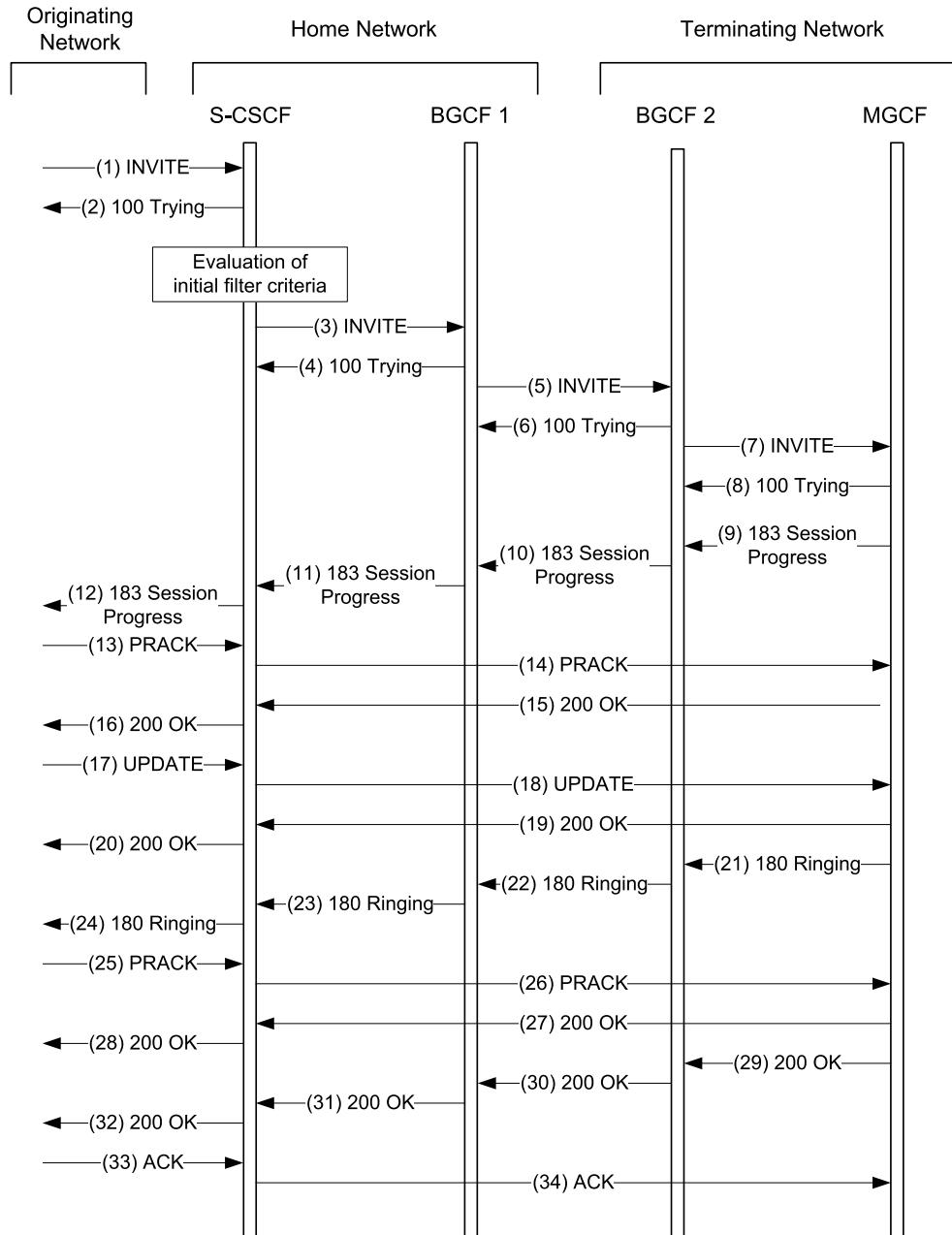


Figure 5.67: Session handled by a remote MGCF

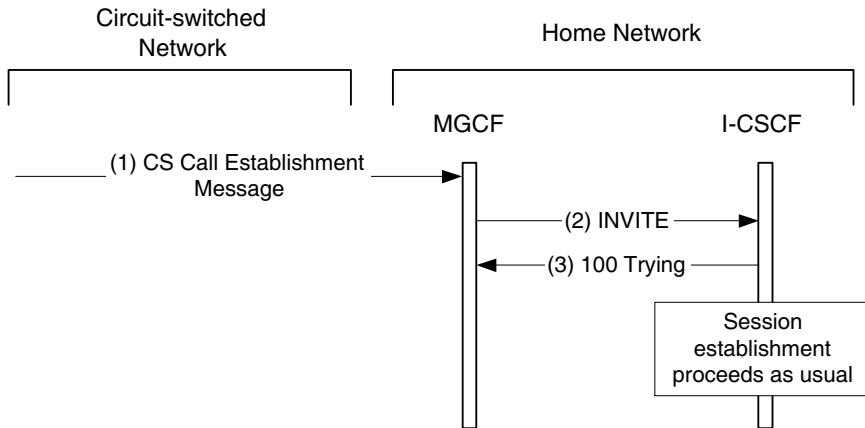


Figure 5.68: PSTN call terminating at the IMS

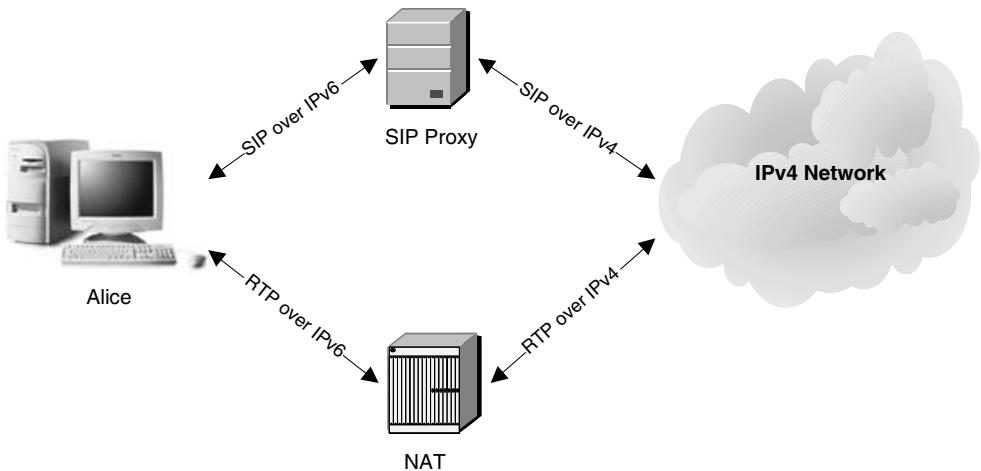


Figure 5.69: IPv4–IPv6 interworking

network always receive and send SIP messages using IPv6 (or IPv4). Nevertheless, proxies at the edge of a network may receive a SIP message over IPv6 whose next hop is an IPv4 host or a SIP message over IPv4 whose next hop is an IPv6 host. In this case the proxy handling the SIP message **Record-Routes**, so that it remains in the path of all subsequent requests and responses related to that dialog. This way, it can relay messages from IPv4 to IPv6 and from IPv6 to IPv4.

Proxies at the edge of a domain need to have both IPv4 and IPv6 DNS entries in order to handle incoming sessions. IPv4 hosts that send a message to the domain use the proxy IPv4 DNS entry while IPv6 hosts use the IPv6 entry. The proxy decides whether or not it has to **Record-Route** based on the IP version of the incoming datagram.

IPv4–IPv6 interworking at the media level is performed by media intermediaries. These media intermediaries can be controlled by another entity (e.g., a SIP session border controller) or they can be stand-alone like a NAT (see Section 4.20). There have been many discussions on whether NAT-based IPv4–IPv6 transition techniques are appropriate because they have a set of shortcomings. We mention them anyway since, at the time of writing, it is not clear that they will be abandoned.

When NAT-based IPv4–IPv6 transition techniques are in use, user agents face a typical NAT traversal problem. User agents need to somehow obtain the address of the NAT to place it in their offers and answers. In the example in Figure 5.70, Alice uses IPv6 while Bob uses IPv4. Alice sends an INVITE with an offer to Bob to establish a session. Nevertheless, Alice cannot use her IPv6 address in her offer, because Bob would not understand it; she needs to use the IPv4 address of the NAT (192.0.10.1). In Section 5.16, we discuss the interactions between different ways to deal with NAT traversal.

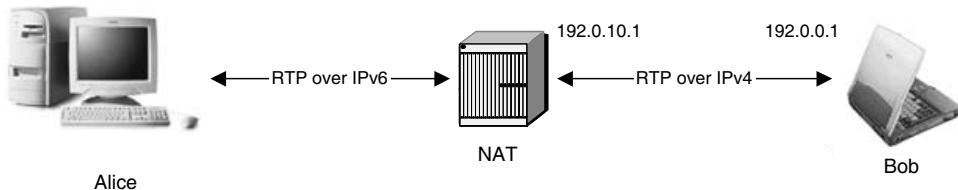


Figure 5.70: IPv4–IPv6 NAT

5.11 Combinational Services

Combinational services (specified in 3GPP TS 23.279 [11] and 3GPP TS 24.279 [12]) combine a CS (circuit-switched) session and an IMS session between the same two terminals into a single CSI (combination of CS and IMS services) session to provide the user with a unified experience, as shown in Figure 5.71. Of course, combinational services can only be implemented in terminals that can simultaneously access a CS network and a PS network. These services are especially useful when the terminal uses a low-bandwidth or high-delay PS (packet-switched) access that is not suitable for real-time interactive media. In such a case, the CS network is used to transport the real-time components of the session (e.g., audio, video, or both) and the PS network is used to transport non-real-time components (e.g., instant messaging). This way, users can apply IMS services to their non-real-time components. Note, however, that time synchronization cannot be provided between media transported over different domains.

Terminals discover advertise and discover support for combinational services using feature tags (as specified in 3GPP TS 24.279 [12]; general use of feature tags is described in Section 4.12). The feature tags used to indicate support for combinational services involving CS audio and CS video are `3gpp.cs-voice` and `3gpp.cs-video`, respectively. OPTIONS requests can be used to discover whether or not a remote terminal supports these feature tags.

A CSI session can be established in two ways: an IMS session followed by a CS session, or a CS session followed by an IMS session. In both cases, the terminal establishing the second session can use the MSISNN of the remote terminal to establish the session. If the

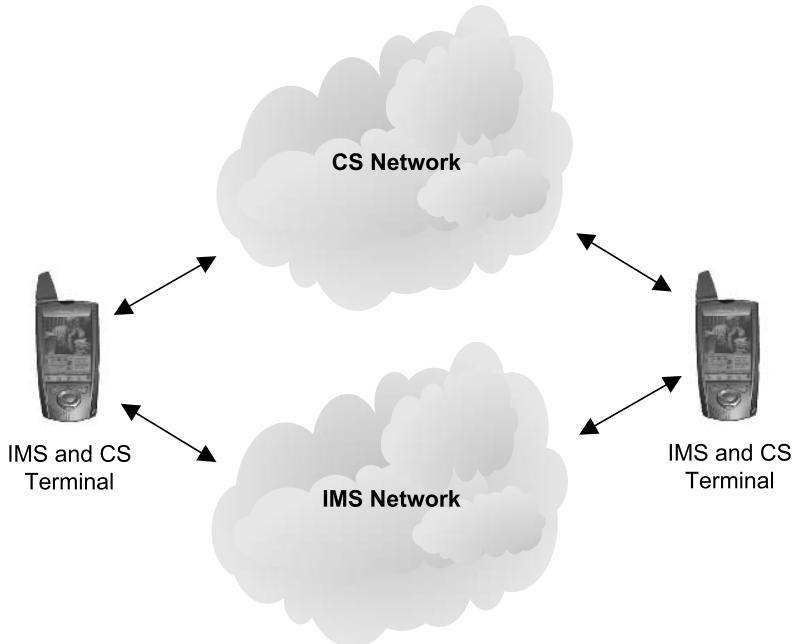


Figure 5.71: Combinational services

second session is the IMS session, the MSISDN can be formatted as a `tel` URI. The terminal receiving a session establishment (CS or IMS) for the second session correlates this session establishment with the ongoing session (which was established earlier) using the `P-Asserted-Identity` of the IMS session and the calling party number of the CS session.

Another scenario where combinational services are useful is when an IMS terminal using only a PS access communicates with a terminal that uses a CS access for real-time interactive media and a PS access for other media, as shown in Figure 5.72. A CSI AS is defined for this purpose. The CSI AS takes care of splitting and combining the sessions towards the terminals, since one terminal only handles a single IMS session while the other handles two sessions: an IMS session and a CS session.

5.12 Basic Sessions Not Requiring Resource Reservation

We saw in Section 5.7 the general session establishment where both IMS terminals need to reserve resources during the session establishment phase. This is typically the case when both terminals are using a cellular IP-CAN, such as GPRS. However, there might be cases where resource reservation has been already achieved, or is not even needed due to the characteristics of the IP-CAN, in which case the session establishment flow is greatly simplified. Depending on whether one or both terminals can skip the resource reservation phase, the flows are slightly different. Let us take a deeper look at these cases.

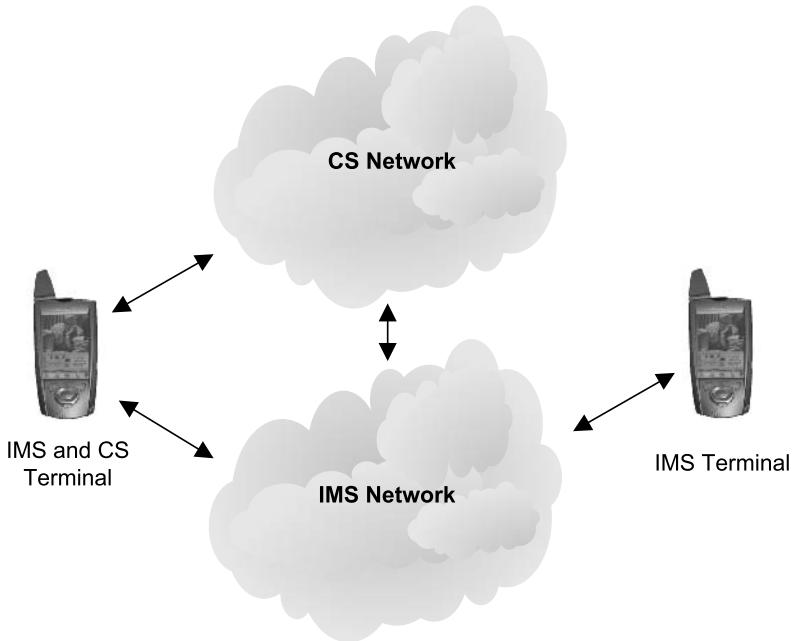


Figure 5.72: Combinational services involving an IMS-only terminal

5.12.1 *The Callee Does Not Require Resource Reservation*

Figure 5.73 depicts the first part of a session establishment flow when the caller needs to reserve local resources and the callee does not need to reserve local resources. Figure 5.74 depicts the rest of the flow. Let us analyze the relevant messages. Figure 5.75 shows the INVITE request as it leaves the caller IMS terminal. This INVITE request is exactly the same as the one we described in the regular basic session setup in Figure 5.21. Bear in mind that at this stage, the caller knows it needs to reserve resources but knows nothing about the callee situation, so the requests are, as one can expect, the same. The caller has indicated the support for the reliability of provisional responses and preconditions extensions by inserting the tags `100rel` and `precondition` in the `Supported` header field. In addition, the caller has declared the lack of resources in the `a=curr:qos local none` SDP line, and its desired result in `a=des:qos mandatory local sendrecv` SDP line. In addition, it has set each of the media streams in inactive mode by setting the `a=inactive` SDP line as a mechanism to prevent the remote party from starting to send media that, otherwise, would not be received by the caller. Also, in this example, the caller has selected a single video codec and a single audio codec (because the `telephone-events` is not considered a real codec), so at this stage, there is a single selected codec for each of video and audio.

This INVITE request is forwarded all the way to the callee, which answers with the 180 (Session Progress) response shown in Figure 5.76. The callee is requiring the usage of the preconditions extension and the reliable provisional responses extension, as can be seen in the `Require` header field. The callee also indicates in each SDP media stream that he has local resources already available, as we can see in the `a=curr:qos local sendrecv` SDP lines. It also adds an `a=conf:qos remote sendrecv` line in each media stream to request

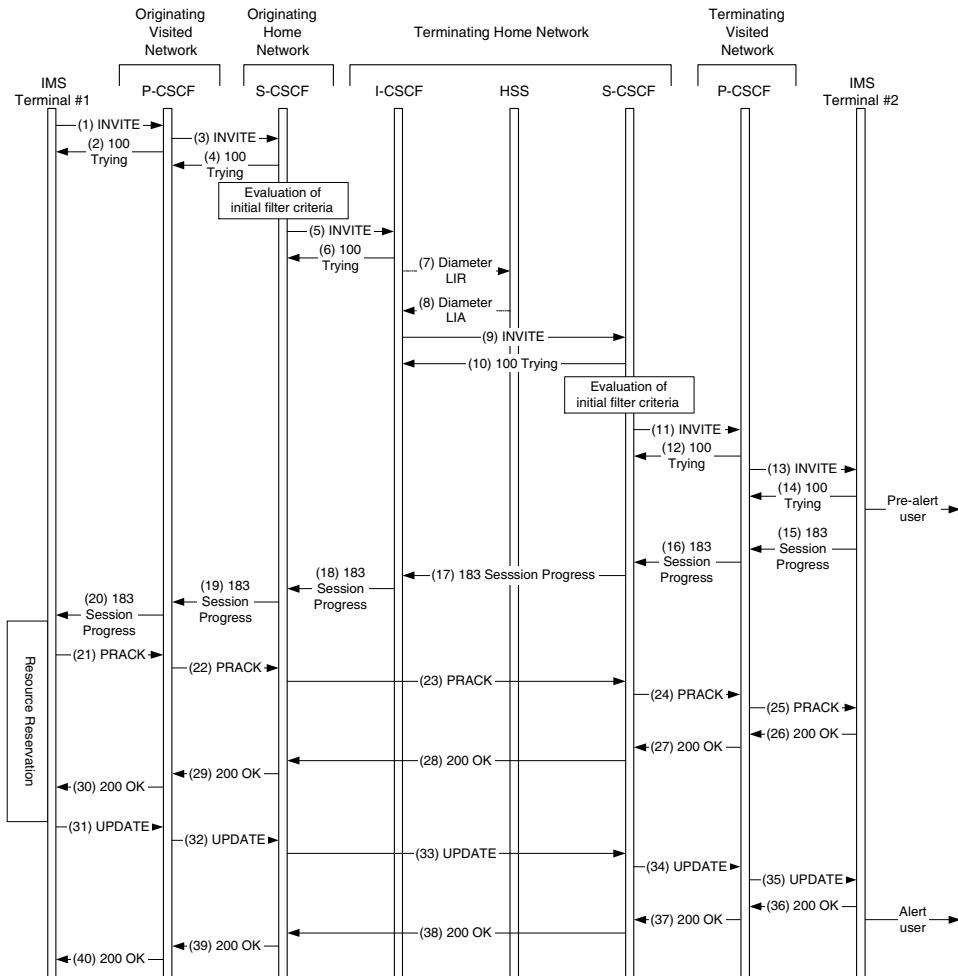


Figure 5.73: Basic session setup, callee does not require resource reservation (part 1)

the caller to confirm when he has got his resources available. However, media streams are kept in inactive mode, `a=inactive`, because the caller does not have resources so media packets cannot be sent through the media path.

The 183 (Session Progress) response is sent back all the way through the signaling path to the caller, which also confirms the final agreed single codec for each of the media streams, the remote IP addresses and port numbers, etc., so it starts the resource reservation procedure. The caller also confirms the reception of the 183 (Session Progress) response with a PRACK request, according to the reliable provisional responses mechanism. Since there is a single codec selected for each media stream in the session, the SDP has not changed; thus, the caller does not insert SDP in the PRACK request. The callee receives the PRACK request in step (25) and generates a 200 (OK) response in step (26), which does not contain SDP either. The 200 (OK) response to the PRACK request is forwarded and received, in step (30) by the caller.

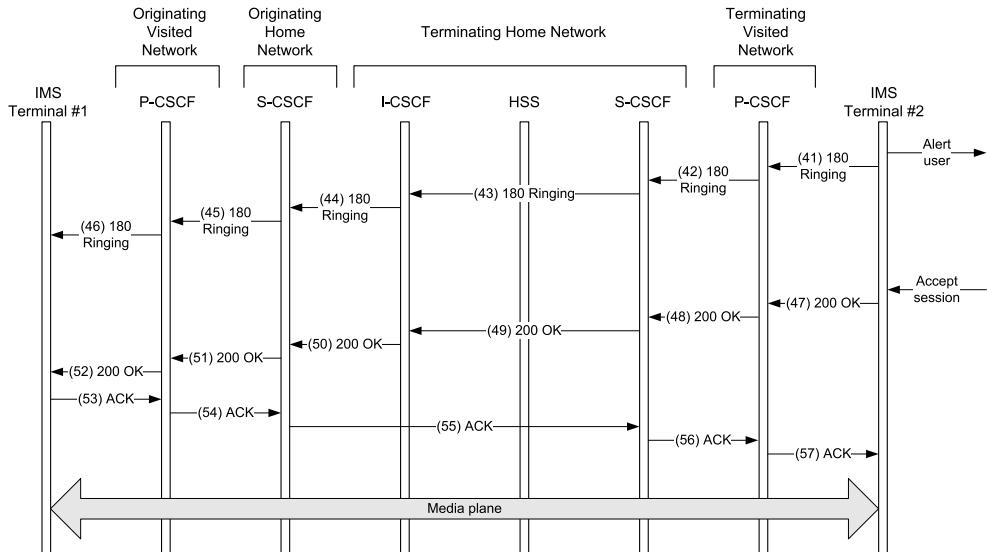


Figure 5.74: Basic session setup, callee does not require resource reservation (part 2)

When the caller finalizes its resource reservation process, it honors the resource confirmation requested in the `a=confirm` line in SDP for each of the media streams, and generates an UPDATE request (step 31) that contains a new SDP offer. The purpose of this SDP offer is to inform the callee that resources are now available on the caller's local side. Figure 5.77 shows this UPDATE request. Since both sides now have available resources, each media stream is attributed with the `a=sendrecv` line to indicate the ability to send and receive media over that media path.

The UPDATE request is received in step 35 at the callee, which generates a 200 (OK) response (step 36) that contains an SDP answer. This SDP answer is very similar to the one in the UPDATE request, since both sides of the media path have available resources at their disposal. The callee examines the SDP in the UPDATE request (35) and observes that the resources are already available at the caller, because of the presence of `a=curr:qos` local `sendrecv` lines in each media stream definition. Since the callee always had available resources at his disposal, and the caller now has them, the callee alerts the user and generates the corresponding 180 (Ringing) provisional response. This corresponds to step 41 in Figure 5.74. Since this provisional response does not contain SDP, it need not be sent reliably. This is achieved by not including a `Require` header field with the value `100rel` or an `RSeq` header field in the 180 (Ringing) response (41).

When the callee eventually accepts the session, it generates a 200 (OK) response, step 46 in Figure 5.74, which is received by the caller and indicates that the session has been accepted. The 200 (OK) response does not contain SDP, since the SDP has not changed since the previous exchange. The 200 (OK) response (51) is confirmed with an ACK request (52). Media can be exchanged between the two endpoints at this stage.

```

INVITE sip:bob@home2.net SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;
      comp=sigcomp;branch=z9hG4bK9h9ab
Max-Forwards: 70
Route: <sip:pcscf1.visited1.net:5058;lr;comp=sigcomp>,
        <sip:orig@scscf1.home1.net;lr>
P-Preferred-Identity: "Alice Smith" <sip:alice@home1.net>
Privacy: none
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                        utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>
Call-ID: 3s09cs03
Cseq: 127 INVITE
Require: sec-agree
Proxy-Require: sec-agree
Supported: 100rel, precondition
Security-Verify: ipsec-3gpp; q=0.1;
                  alg=hmac-sha-1-96;
                  spi-c=98765432; spi-s=909767;
                  port-c=5057; port-s=5058
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Content-Type: application/sdp
Content-Length: 567

v=0
o=- 1073055600 1073055600 IN IP6 1080::8:800:200C:417A
s=-
c=IN IP6 1080::8:800:200C:417A
t=0 0
m=video 8382 RTP/AVP 98
b=AS:75
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=inactive
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
m=audio 8283 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=inactive
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.75: (1) INVITE

```

SIP/2.0 183 Session Progress
Via: SIP/2.0/UDP pcscf2.visited2.net:5056;comp=sigcomp;
      branch=z9hG4bK2a2qr
Via: SIP/2.0/UDP scscf2.home2.net;branch=z9hG4bKvp2ym1
Via: SIP/2.0/UDP icscf2.home2.net;branch=z9hG4bKra1ar
Via: SIP/2.0/UDP scscf1.home1.net;branch=z9hG4bKs1pp0
Via: SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Record-Route: <sip:pcscf2.visited2.net:5056;lr;comp=sigcomp>,
               <sip:scscf2.home2.net;lr>,
               <sip:scscf1.home1.net;lr>,
               <sip:pcscf1.visited1.net;lr>
Privacy: none
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                      utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 127 INVITE
Require: 100rel, precondition
Contact: <sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
RSeq: 9021
Content-Type: application/sdp
Content-Length: 642

v=0
o=- 3021393216 3021393216 IN IP6 1081::5:800:200A:B2B2
s=-
c=IN IP6 1081::5:800:200A:B2B2
t=0 0
m=video 14401 RTP/AVP 98
b=AS:75
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=conf:qos remote sendrecv
a=inactive
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
m=audio 6544 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=conf:qos remote sendrecv
a=inactive
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.76: (15) 183 Session Progress

```

UPDATE sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Max-Forwards: 70
P-Access-Network-Info: 3GPP-UTRAN-TDD;
      utran-cell-id-3gpp=24450289A3299239
Route: <sip:pcscf1.visited1.net:5058;lr;comp=sigcomp>,
      <sip:scscf1.home1.net;lr>,
      <sip:scscf2.home2.net;lr>,
      <sip:pcscf2.visited2.net;lr>
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 129 UPDATE
Require: sec-agree
Proxy-Require: sec-agree
Security-Verify: ipsec-3gpp; q=0.1;
      alg=hmac-sha-1-96;
      spi-c=98765432; spi-s=909767;
      port-c=5057; port-s=5058
Content-Type: application/sdp
Content-Length: 593

v=0
o=- 1073055600 1073055604 IN IP6 1080::8:800:200C:417A
s=-
c=IN IP6 1080::8:800:200C:417A
t=0 0
m=video 8382 RTP/AVP 98
b=AS:75
a=curr:qos local sendrecv
a=curr:qos remote sendrecv
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=sendrecv
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
m=audio 8283 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote sendrecv
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.77: (31) UPDATE

5.12.2 The Caller Does Not Require Resource Reservation

A different and even simpler case occurs when the caller does not need to reserve resources, but the callee requires resource reservation. In the case of the caller, it may be that the IP-CAN the caller is connected to does not require resource reservation, or it might be that the caller has been able to reserve resources before the actual session has taken place. In any case, the caller does not need to reserve resources during the session. Figure 5.78 shows the corresponding flow.

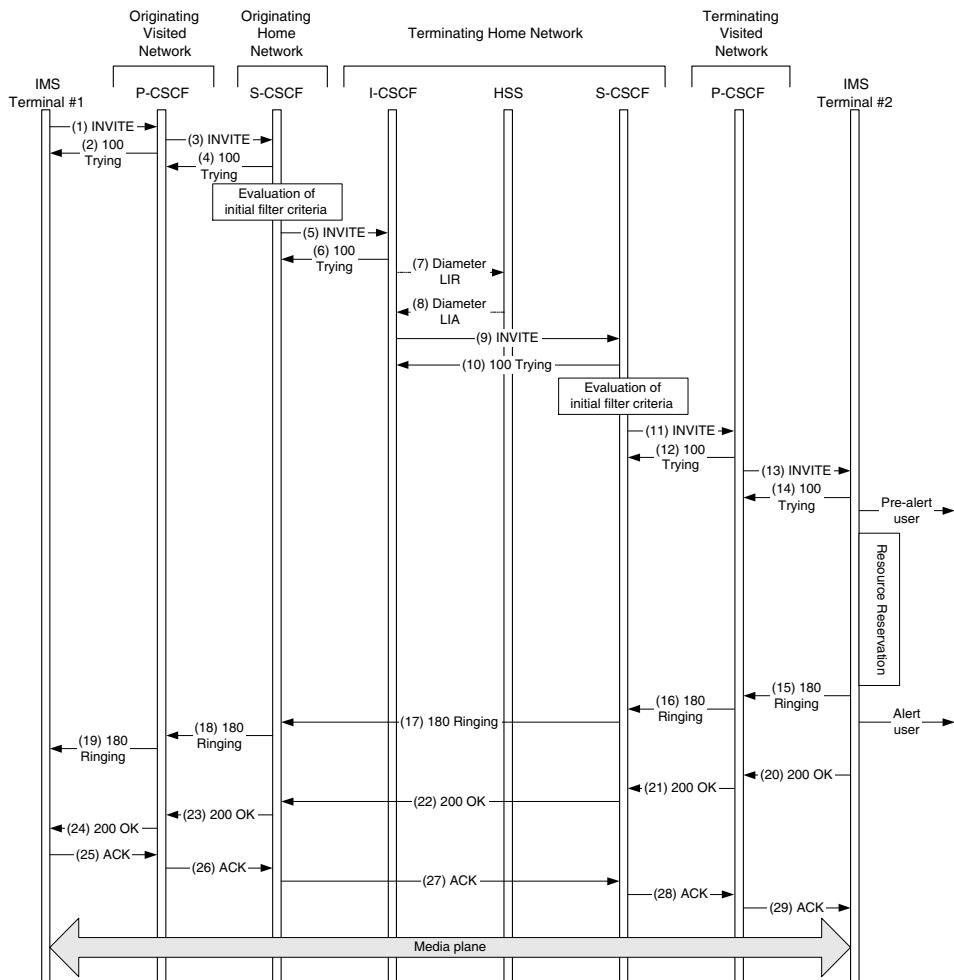


Figure 5.78: Basic session setup, caller does not require resource reservation

The flow starts by the caller sending an INVITE request (1) (see Figure 5.79), where it indicates in the `a=curr:qos local sendrecv` line in SDP that resources are already available. Since this is the case, the media streams are set in “sendrecv” mode as well (which is the default mode, and does not require explicit signaling). In addition, there is a single offered codec per media stream (we do not count the telephone-event pseudo-codec).

```

INVITE sip:bob@home2.net SIP/2.0
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;
    comp=sigcomp;branch=z9hG4bK9h9ab
Max-Forwards: 70
Route: <sip:pcscf1.visited1.net:5058;lr;comp=sigcomp>,
    <sip:orig@scscf1.home1.net;lr>
P-Preferred-Identity: "Alice Smith" <sip:alice@home1.net>
Privacy: none
P-Access-Network-Info: 3GPP-UTRAN-TDD;
    utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>
Call-ID: 3s09cs03
Cseq: 127 INVITE
Require: sec-agree
Proxy-Require: sec-agree
Supported: 100rel, precondition
Security-Verify: ipsec-3gpp; q=0.1;
    alg=hmac-sha-1-96;
    spi-c=98765432; spi-s=909767;
    port-c=5057; port-s=5058
Contact: <sip:[1080::8:800:200C:417A]:5059;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Content-Type: application/sdp
Content-Length: 563

v=0
o=- 1073055600 1073055600 IN IP6 1080::8:800:200C:417A
s=-
c=IN IP6 1080::8:800:200C:417A
t=0 0
m=video 8382 RTP/AVP 98
b=AS:75
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:98 H263
a=fmtpt:98 profile-level-id=0
m=audio 8283 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=inactive
a=rtpmap:97 AMR
a=fmtpt:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.79: (1) INVITE

The INVITE request is forwarded following the regular process and eventually is received by the callee (step 13). The callee may pre-alert the user at this stage, like indicating that an incoming session is being set up. The callee inspects the SDP, and notices that resources are available at the caller side, and there is a single codec agreed for each media stream, so it has all the bandwidth parameters, IP addresses and port numbers required to start its resource reservation process. So, the callee reserves resources, and when it finishes, it alerts the user and generates a 180 (Ringing) provisional response in step 15, which does not contain SDP, and thus need not be sent reliably. Eventually, the callee accepts the session, at the moment in which a 200 (OK) final response is generated (step 20). This 200 (OK) final response contains the SDP answer to the initial INVITE request, indicating that resources are also available now at the callee's side. Figure 5.80 shows the detailed contents of the 200 (OK) final response (step 20).

When the caller receives the 200 (OK) final response (24), it generates an ACK request (25) to confirm the reception. Media plane traffic can start at this moment.

There is a slight variation of the flow as shown in Figure 5.78. The callee may decide to include SDP in the 180 (Ringing) response, in which case, it has to be sent reliably, and therefore needs to be confirmed with a PRACK request (and its corresponding 200 (OK) response).

5.12.3 Neither the Caller Nor the Callee Require Resource Reservation

There are scenarios where neither the caller nor the callee require to reserve resources during the session setup up. Push-to-talk over Cellular (PoC) is an example of this application. We further describe the PoC service in Chapter 25.

The signaling flow that applies when neither of the endpoints require resource reservation is equal to the case when only the callee requires resource reservation, i.e., the flow is as shown in Figure 5.78, with the exception that the callee will not reserve local resources. Because of the similarities of these flows, we shall not devote further analysis to this case.

5.13 Globally Routable User Agent URIs (GRUU) in IMS

We described in Section 4.19 the need and general mechanism for handling GRUUs in the Internet. Let us take a look at how IMS deals with GRUUs.

In IMS, the S-CSCF is the registrar, and therefore the S-CSCF generates GRUUs to IMS terminals. In an initial registration, the S-CSCF generates a public GRUU and a temporary GRUU for the Public User Identity under registration, and sends both GRUUs in the Contact header field of the 200 (OK) response to the REGISTER request. Furthermore, if the Public User Identity under registration automatically registers other Public User Identities that are part of the set of implicitly registered Public User Identities, then the S-CSCF generates a public GRUU and a temporary GRUU per implicitly registered Public User Identity. However, these GRUUs that are bound to implicitly registered Public User Identities are not conveyed in the 200 (OK) response to the REGISTER request, but instead, learnt via the `reg` event package notification (we described the subscriptions to the `reg` event package in Section 5.6).

Figure 5.81 shows an example of the XML document included in a NOTIFY request for the `reg` event package, but now including also GRUU information. The XML document provides information about two registered Public User Identities in `<registration>` elements. One Public User Identity was explicitly registered, whereas the other one, which

```

SIP/2.0 200 OK
Via: SIP/2.0/UDP pcscf2.visited2.net:5056;comp=sigcomp;
      branch=z9hG4bK2a2qr
Via: SIP/2.0/UDP scscf2.home2.net;branch=z9hG4bKvp2ym1
Via: SIP/2.0/UDP icscf2.home2.net;branch=z9hG4bKra1ar
Via: SIP/2.0/UDP scscf1.home1.net;branch=z9hG4bKs1ppo
Via: SIP/2.0/UDP pcscf1.visited1.net;branch=z9hG4bKoh2qrz
Via: SIP/2.0/UDP [1080::8:800:200C:417A]:5059;comp=sigcomp;
      branch=z9hG4bK9h9ab
Record-Route: <sip:pcscf2.visited2.net:5056;lr;comp=sigcomp>,
               <sip:scscf2.home2.net;lr>,
               <sip:scscf1.home1.net;lr>,
               <sip:pcscf1.visited1.net;lr>
Privacy: none
P-Access-Network-Info: 3GPP-UTRAN-TDD;
                      utran-cell-id-3gpp=24450289A3299239
From: <sip:alice@home1.net>;tag=ty20s
To: <sip:bob@home2.net>;tag=d92119
Call-ID: 3s09cs03
Cseq: 127 INVITE
Contact: <sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp>
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Content-Type: application/sdp
Content-Length: 570

v=0
o=- 3021393216 3021393216 IN IP6 1081::5:800:200A:B2B2
s=-
c=IN IP6 1081::5:800:200A:B2B2
t=0 0
m=video 14401 RTP/AVP 98
b=AS:75
a=curr:qos local sendrecv
a=curr:qos remote sendrecv
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=rtpmap:98 H263
a=fmtp:98 profile-level-id=0
m=audio 6544 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote sendrecv
a=des:qos mandatory local sendrecv
a=des:qos mandatory remote sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 5.80: (20) 200 OK

```

<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo"
           xmlns:gr="urn:ietf:params:xml:ns:gruuinfo"
           version="1" state="full">
    <registration aor="sip:alice@home1.net" id="11a" state="active">
        <contact id="542" state="active" event="registered"
                  duration-registered="1" expires="599999"
                  callid="23fi57lju" cseq="2">
            <uri>sip:[1080::8:800:200C:417A]:5059;comp=sigcomp</uri>
            <unknown-param name="+sip.instance">
                &lt;&lt;urn:uui:a56d41bf-e0c9-4c80-abd3-82c40b973806>&gt;;
            </unknown-param>
            <gr:pub-gruu uri="sip:alice@home1.net;gr=urn:uuid:
                           a56d41bf-e0c9-4c80-abd3-82c40b973806"/>
            <gr:temp-gruu uri="sip.tgruu.208dkds2dam@ckjs.home1.net;gr"
                           first-cseq="2"/>
        </contact>
    </registration>
    <registration aor="sip:+12125551234@home1.net;user=phone"
                  id="11c" state="active">
        <contact id="544" state="active" event="created"
                  duration-registered="1" expires="599999"
                  callid="23fi57lju" cseq="2">
            <uri>sip:[1080::8:800:200C:417A]:5059;comp=sigcomp</uri>
            <unknown-param name="+sip.instance">
                &lt;&lt;urn:uui:a56d41bf-e0c9-4c80-abd3-82c40b973806>&gt;;
            </unknown-param>
            <gr:pub-gruu uri="sip:+12125551234@home1.net;user=phone
                           gr=urn:uuid:a56d41bf-e0c9-4c80-abd3-82c40b973806"/>
            <gr:temp-gruu uri="sip.tgruu.n0wkdsj@ckjs.home1.net;gr"
                           first-cseq="2"/>
        </contact>
    </registration>
</reginfo>

```

Figure 5.81: XML reg event document including GRUUs

represents a telephone number, was implicitly registered. Information about each Public User Identity comprises the contact information, instance ID, public GRUU, temporary GRUU, and the Call-ID and CSeq header fields of the REGISTER request that generated the registration. The IMS terminal can now determine not only the set of implicitly registered Public User Identities, but also the temporary and public GRUU allocated to each of them.

A public GRUU is persistent and valid for the period of time the user is registered, i.e., from the initial registration until a deregistration (either explicit or administrative). Therefore, public GRUUs are only generated in an initial registration. In a re-registration, the S-CSCF returns the same public GRUU that was created during the initial registration.

Temporary GRUUs have a more limited lifespan. Temporary GRUUs are generated in an initial registration or in a re-registration, and are valid until the user deregisters. In other words, in an initial registration, the S-CSCF creates a public GRUU and a first temporary GRUU. Before the registration expires, when the IMS terminal re-registers, the S-CSCF creates an additional temporary GRUU, while the public GRUU and the first temporary GRUU are still valid. Every time the IMS terminal re-registers without discontinuing the existing registration, an additional temporary GRUU is created and returned to the IMS terminal without invalidating the previous ones. The goal is to provide some sort of anonymity via temporary GRUUs, while still permitting slightly older temporary GRUUs to be used without jeopardizing their functionality. In general, the IMS terminal needs to store the public GRUU and a handful of temporary GRUUs per identity. If it needs to advertise a temporary GRUU, it should always use the last acquired one.

Temporary GRUUs are created in the S-CSCF and are not communicated to the HSS. This has an implication for the S-CSCF to create temporary GRUUs that can resolve directly to the S-CSCF. There are several mechanisms to achieve this property. One of them consists of the S-CSCF host name being included in temporary GRUUs created at that S-CSCF. Another consists of configuring temporary GRUUs created at a S-CSCF as Public Service Identities, and configuring the mapping of the PSI to the proper S-CSCF. The exact mechanism for creating temporary GRUUs is not standardized at the time of writing.

5.14 IMS Communication Service Identifier (ICSI)

3GPP Release 7 introduced the notion of the *IMS Communication Service Identifier (ICSI)*, which is a mechanism that allows the terminal and the network to identify which service is intended to be invoked with the SIP signaling. The SIP signaling contains a tag that identifies the intended service. This allows a terminal that receives a SIP request to dispatch it to the intended application. In the network, the identification of the service can help in triggering the signaling to the correct Application Server.

The ICSI is effectively a Uniform Resource Name (URN) that identifies the service. For example, an imaginary “game foo” application could be identified with the following URN:

```
urn:urn-xxx:3gpp-service.ims.icsi.game-foo
```

The string xxx represents a number that the Internet Assigned Numbers Authority (IANA) has to allocate. At the time of this writing, this allocation has not been done yet. The URN is further split in several trees. One of them is the 3gpp-service, which is the one designated to include an ICSI.

We saw in Section 4.12 how terminals can use media feature tags to indicate supported features at registration time. Terminals can also indicate in requests (such as INVITE) the feature that they require to establish the session. 3GPP has defined an Application Reference feature-tag named g.3gpp.app_ref that can hold, among other items, URNs that represent ICSIs. The g.3gpp.app_ref feature tag is included in the Contact header field of any standalone SIP request or request that initiates a dialog and contains a Contact header field, for example REGISTER, initial INVITE, initial SUBSCRIBE, etc., but not MESSAGE request, which does not contain a Contact header field.

ICSI are a valuable tool when the user is simultaneously registered from different terminals, and each one has implemented or enabled different services. The S-CSCF can use this information to dispatch requests to the appropriate terminal. Figure 5.82 shows an

```
Contact: <sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp>
:+g.3gpp.app_ref="urn:urn-xxx:3gpp-service.ims.icsi.game-foo"
```

Figure 5.82: ICSI included in the Contact header field

example of a Contact header field of a SIP request that contains the 3GPP Application Reference feature-tag that includes an ICSI.

In addition to the g.3gpp.app_ref feature tag, there are two new header fields that can hold ICSIs. These are the P-Preferred-Service and P-Asserted-Service header fields, both of them specified in the Internet-Draft “A Session Initiation Protocol (SIP) extension for the identification of services” [122]. These header fields work in a similar way to the P-Preferred-Identity and P-Asserted-Identity header fields that we discussed earlier in Section 5.7.1, but now with respect to intended services.

When the IMS terminal sends a SIP request that initiates a new session, or a standalone SIP request (e.g., MESSAGE), the IMS terminal includes a P-Preferred-Service header field that contains the URN that represents the service associated to such a request. Figure 5.83 shows an example of this header field. In addition, if this SIP request contains a Contact header field (notice that the MESSAGE request does not contain a Contact header field), then it also contains an ICSI encoded in the g.3gpp.app_ref feature tag of the Contact header field. Then the IMS terminals sends the SIP request to the P-CSCF, as per regular procedures.

```
P-Preferred-Service: urn:urn-xxx:3gpp-service.ims.icsi.game-foo
```

Figure 5.83: The P-Preferred-Service header field

The S-CSCF receives the SIP request, evaluates the preferred service indicated by the IMS terminal and the rest of the SIP signaling, decides the value of the ICSI, inserts the ICSI in a new P-Asserted-Service header field, and removes the existing P-Preferred-Service header field. This ICSI value is typically the same as was received in the P-Preferred-Service header field, but local policy makes the S-CSCF change it to a different one. Figure 5.84 shows an example of the P-Asserted-Service header field.

```
P-Asserted-Service: urn:urn-xxx:3gpp-service.ims.icsi.game-foo
```

Figure 5.84: The P-Asserted-Service header field

In general, the value of the ICSI in the g.3gpp.app_ref feature tag is used by the IMS terminals, whereas the value of the ICSI in the P-Asserted-Service header field is used by network elements, such as S-CSCFs and Application Servers.

Figure 5.85 shows the registration of an IMS terminal that includes its ICSI in the g.3gpp.app_ref feature tag, and then a session initiated by another IMS terminal towards this one, using the ICSI. The flow starts when IMS Terminal 2 sends a REGISTER request (1) that contains a Contact header field describing one or more supported services in the g.3gpp.app_ref feature tag. We assume that this REGISTER request (1) contains the credentials needed for authentication. The REGISTER request (1) is forwarded to an I-CSCF (2) and eventually reaches the S-CSCF allocated to the user (5). The S-CSCF is now informed of the services supported by this terminal. If the user is simultaneously registered

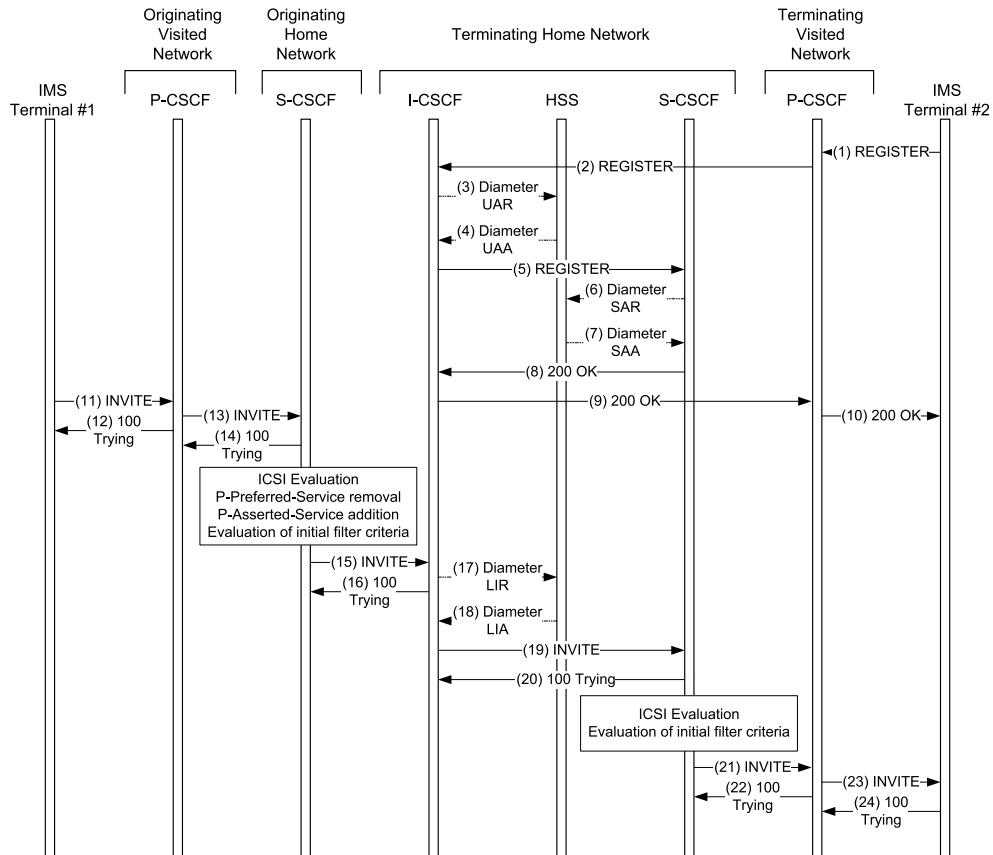


Figure 5.85: Registration and session initiation flow supporting ICSI

from different terminals, the S-CSCF is able to distinguish the services supported by each terminal.

At a later point in time, IMS Terminal 1 sends an INVITE request (11) towards the user of IMS Terminal 2. The INVITE request (11) contains a P-Preferred-Service header field that includes the ICSI that identifies the intended communication service, and a Contact header field that also contains the ICSI encoded in the g.3gpp.app_ref feature tag. This INVITE request (11) is forwarded (13) to the originating S-CSCF, which examines the SIP message, authorizes the originating user to use the service identified by the ICSI, and replaces the P-Preferred-Service header field with a P-Asserted-Service header field, prior to evaluating the initial filter criteria. Then it forwards the INVITE request (15) towards the terminating network. Assuming trust between the originating and terminating networks, the S-CSCF in the terminating network receives the INVITE request (19) and evaluates whether the user authorizes the terminating user for using the service identified by the ICSI, and then determines which of the various terminals where the user is registered is more suitable for executing the service. Finally, the S-CSCF forwards the INVITE request (21) towards that IMS terminal, which may inspect the value of the g.3gpp.app_ref feature tag in the Contact header field for identification of the intended service.

At the time of writing, there is progress in standardizing the value of URNs for a number of communication services, including multimedia telephony services, instant messaging, and others. 3GPP maintains a registry of allocated ICSIs on the following web page:

<http://www.3gpp.org/tb/0ther/URN/URN.htm>

The P-Asserted-Service header field requires the existence of trust between two networks. Otherwise, an IBCF located at the edge of the network will remove this header field. A similar behaviour for trust is in place with respect to the P-Asserted-Identity header field as well.

5.15 IMS Application Reference Identifier (IARI)

The IMS Communication Service Identifier (ICSI) provides an indication of the service that the user is intending to invoke. However, it does not indicate which application, inside the IMS terminal, is invoking that service. Actually, there can be a number of applications that use the same service. Assume an instant messaging service. The service can be invoked by a networked game or by a presence and instant messaging application. While the core of the communication service is granted by the ICSI, there might be additional (or even proprietary) features that are only available when both IMS terminals are using the same application.

To address this issue, 3GPP has introduced in Release 7 the notion of the *IMS Application Reference Identifier (IARI)*, which is a token that identifies the application running in the terminal. This information is valuable to endpoints, e.g., the IMS terminals, or an Application Server that is acting as an endpoint (e.g., a SIP User Agent), but unlike the ICSI, it is not valuable to the rest of the network nodes.

The IARI can help the terminating endpoint to dispatch a SIP request to exactly the same application that has sent the SIP request in the originating endpoint, allowing a better user experience to be provided to both users engaged in the communication.

Like the ICSI, the IARI is actually a URN as well, and it is in fact encoded in the g.3gpp.app_ref feature tag, like the ICSI. However, unlike ICSI entries, IARI entries are part of the 3gpp-application tree in the URN. The following is an example of the URN of a IARI representing a hypothetical “game application 1”:

urn:urn-xxx:3gpp-application.ims.iari.game-appl-1

IARIs are encoded in the g.3gpp.app_ref feature tag of the Contact header field of any standalone SIP request or request that initiates a dialog and that contains a Contact header field. For example, REGISTER, initial INVITE, initial SUBSCRIBE request, etc., encode IARIs. Figure 5.86 shows an example of a hypothetical game application IARI encoded in the Contact header field of any of the mentioned SIP requests.

```
Contact: <sip:[1081::5:800:200A:B2B2]:5055;comp=sigcomp>
;+g.3gpp.app_ref="urn:urn-xxx:3gpp-application.ims.iari.game-appl-1"
```

Figure 5.86: IARI included in the Contact header field

3GPP maintains a registry of allocated IARIs on the same web page as the ICSI registry:

<http://www.3gpp.org/tb/0ther/URN/URN.htm>

5.16 NAT Traversal in the IMS

In Section 3.7.2, we discussed how a P-CSCF can control an IMS Access Gateway over the Iq interface (see Figure 3.14) and how an IBCF can control a TrGW (see Figure 3.15). Both the IMS Access Gateway and the TrGW typically have public IP addresses that can serve as anchor points to help clients traverse NATs. In Section 4.20.4, we described how clients can use ICE to traverse NATs without any explicit assistance from the network. The interactions between session border gateways such as the IMS Access Gateway and the TrGw and ICE have been defined to avoid conflicts that could lead to inter-operability problems (they have been specified in 3GPP TS 24.229 [37]).

P-CSCFs and IBCFs inspect the SDP bodies of the SIP messages coming through them to see if ICE is being used. If they control a session border gateway, they can decide to act as ICE endpoints so that all the ICE procedures (including STUN-based connectivity checks) are terminated in the session border gateway. Whether or not a P-CSCF or an IBCF decide to terminate ICE procedures in their gateway or let ICE run end-to-end is up to the operator's local policy.

Note that ICE can also be used by dual-stack terminals to decide whether to use IPv4 or IPv6. Therefore, the use of ICE also impacts on IPv4–IPv6 transition techniques.

Chapter 6

AAA on the Internet

6.1 Authentication, Authorization, and Accounting

The term *AAA* has been traditionally used to refer to *Authentication, Authorization, and Accounting* activities. All of those activities are of crucial importance for the operation of an IP network, although typically they are not so visible to the end user.

The importance of AAA functions lies in the fact that they provide the required protection and control in accessing a network. As a consequence, the administrator of the network can bill the end user for services used. By services we are referring to any type of services related to the access of the network, such as high bandwidth, provision of routing services, gateway services, etc.

Before we proceed with this chapter, let us agree on a common terminology.

Authentication. This is the act of verifying the identity of an entity (subject).

Authorization. This is the act of determining whether a requesting entity (subject) will be allowed access to a resource (object) (e.g., network access, certain amount of bandwidth, etc.).

Accounting. This is the act of collecting information on resource usage for the purposes of capacity planning, auditing, billing, or cost allocation.

All of these concepts are intimately linked. For instance, it is not feasible to record the usage of a resource when the entity (subject) making usage of the resource (object) is not yet known. Therefore, in order to account for the usage of a resource the entity has to be authenticated. Once the subject is authenticated, it can be authorized to access the resource. Here, we are speaking generically. A resource could be access to a network, a radio resource, or access to a conference bridge.

The rest of this chapter describes the Internet architecture needed to provide the network functions of AAA. We will learn about the protocols that the IETF has developed to provide the mentioned functions.

6.2 AAA Framework on the Internet

At the beginning of 1997 the IETF defined the Remote Authentication Dial In User Service (RADIUS, RFC 2058 [260]) as the protocol to perform AAA functions on the Internet.

The IETF revised the protocol in mid-1997 in RFC 2138 [261] and again in 2000 in RFC 2865 [262].

RADIUS offers a Network Access Server (NAS) the possibility of requesting authentication and authorization to a centralized RADIUS server. A typical example of the usage of RADIUS is shown in Figure 6.1. A user has established an agreement to access the Internet with an operator that provides a collection of dial-up access servers. A computer equipped with a modem dials up a Network Access Server. A circuit-switched connection is established between the computer (actually, the modem in the computer) and the Network Access Server. The Network Access Server does not contain a list of users who can access the network, since there may be a large collection of servers that are geographically widely spread and it would not be feasible to manage the list in all access servers. Instead, the Network Access Server is configured to request authentication and authorization from an AAA server, using an AAA protocol like RADIUS. The AAA server contains all the data needed to authenticate and then authorize the user (e.g., a password). Once the user is authenticated and authorized the user can get access to the network. The Network Access Server will be providing accounting information reports to the AAA server, so that the network operator can appropriately bill the user.

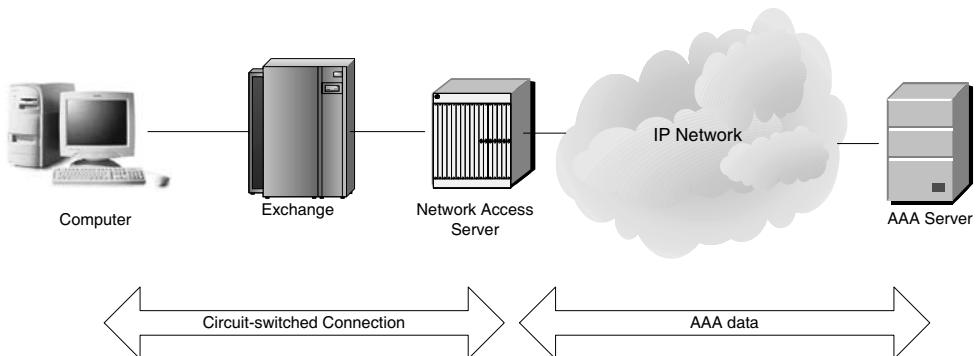


Figure 6.1: AAA functions in a dial-up scenario

The RADIUS protocol performs relatively well in small-scale configurations and for the particular application that it was designed for, that is, a user dials into a dial-up server and the dial-up server requests authentication and authorization from an AAA server. RADIUS offers problems in large environments where congestion and lost data can occur. RADIUS runs over UDP and, therefore, lacks congestion control. RADIUS lacks some functionality that is required in certain applications or networks, such as the ability of the AAA server to send an unsolicited message to the access server.

For all of these reasons the IETF has come up with an improved version of RADIUS named Diameter (the IETF specified the Diameter base protocol in RFC 3588 [96] at the end of 2003). The IMS has selected the modern Diameter as the protocol to perform AAA functions. As the IMS relies on Diameter rather than RADIUS, we will focus in this chapter on Diameter.

6.3 The Diameter Protocol

Diameter is specified as a base protocol and a set of Diameter applications that complement the base protocol functionality. The base protocol contains the basic functionality and is implemented in all Diameter nodes, independently of any specific application. Applications are extensions to the basic functionality that are tailored for a particular usage of Diameter in a particular environment. For instance, there is an application tailored for Network Access Server configurations, another for Mobile IPv4, another for Credit Control, and even one for SIP. Applications are developed as extensions, so new applications are developed when needed. Figure 6.2 depicts the relation between the Diameter base protocol and a few applications.

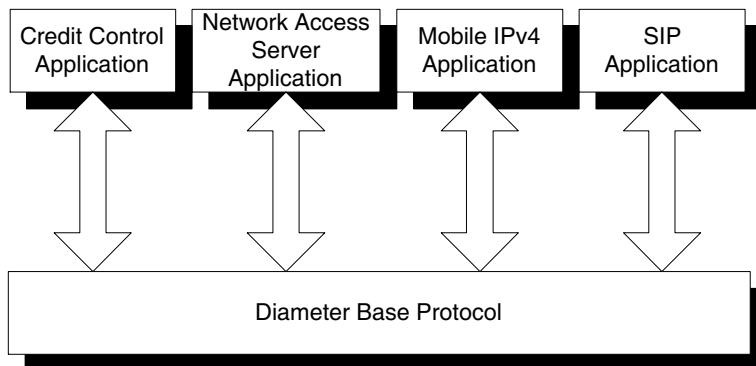


Figure 6.2: Diameter base protocol and applications

Diameter runs over a reliable transport that offers congestion control (e.g., TCP, SCTP). In particular, Diameter does not run over UDP. Unlike RADIUS, lost Diameter messages are retransmitted at each hop. Diameter provides an application-level heartbeat message that monitors the status of the connection and allows recovery in failure situations. Diameter also allows accounting messages to be routed to a different server from authentication/authorization messages (this is actually the case in the IMS).

The Diameter base protocol defines different functional entities for the purpose of performing AAA functions. These are as follows.

Diameter client. This is a functional entity, typically located at the edge of the network, which performs access control. Examples of Diameter clients are Network Access Servers and, in Mobile IP, mobility agents (Foreign Agents).

Diameter server. This is a functional entity that handles authentication, authorization, and accounting requests for a particular realm.

Realm. This is an administrative domain.

Proxy. This is a functional entity that, in addition to forwarding Diameter messages, makes policy decisions relating to resource usage and provisioning. A proxy may modify messages to implement policy decisions, such as controlling resource usage, providing admission control, and provisioning.

Relay. This is a functional entity that forwards Diameter messages, based on routing-related information and realm-routing table entries. A relay is typically transparent. It can modify Diameter messages only by inserting or removing routing-related data, but cannot modify other data.

Redirect agent. This is a functional entity that refers clients to servers and allows them to communicate directly.

Translation agent. This is a functional entity that performs protocol translation between Diameter and other AAA protocols, such as RADIUS.

Diameter node. This is a functional entity that implements the Diameter protocol and acts either as a Diameter client, Diameter server, proxy, relay, redirect agent, or translation agent.

Diameter is a peer-to-peer protocol, rather than the common client/server protocol. This means that, unlike in protocols that follow the client/server model, in Diameter any of the peers can asynchronously send a request to the other peer. Note that, unlike in client/server protocols, a Diameter client is *not* the functional entity that sends a request and a Diameter server is *not* the functional entity that sends an answer to the request. Instead, a Diameter client is a functional entity that performs access control, whereas a Diameter server is the functional entity that performs authentication and authorization. In Diameter, both a Diameter client and a Diameter server can send or receive requests and responses.

Diameter messages are either requests or answers. A request is answered by a single answer. Except for rare occasions, Diameter requests are always answered, so the sender of the request always gets accurate information about the fate of the request and, in the case of error, the sender can recover easily. Diameter is a binary-encoded protocol.

6.3.1 Diameter Sessions

We are used to using the term *session* in the context of SIP/SDP and with the meaning of *multimedia session*. According to SDP (RFC 2327 [160]), a *multimedia session* is:

“a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session.”

Typically, a multimedia session in SIP is delimited by INVITE and BYE transactions.

Diameter also introduces the same term with a broader meaning, and care must be taken to avoid confusion between both terms. According to the Diameter base protocol (RFC 3588 [96]), a *Diameter session* is:

“a related progression of events devoted to a particular activity.”

For instance, in the context of a user dialing up a Network Access Server, the session is composed of all the Diameter messages exchanged between the Network Access Server and the Diameter server from the moment the user dials until the connection is dropped. In the case of the IMS a Diameter session might be composed of all the messages exchanged between the S-CSCF (acting as a Diameter client) and the HSS (acting as a Diameter server) from the time the user registers in the IMS until the user is no longer registered.

Whenever we use the term *session* in the context of Diameter, we refer to a *Diameter session*, unless it is explicitly referred to as a *multimedia session*.

6.3.2 The Format of a Diameter Message

Figure 6.3 shows the format of a Diameter message. A Diameter message consists of a 20-octet header and a number of *Attribute-Value Pairs* (AVPs). The length of the header is fixed; it is always present in any Diameter message. The number of AVPs is variable, as it depends on the actual Diameter message. An AVP is a container of data (typically authentication, authorization, or accounting data).

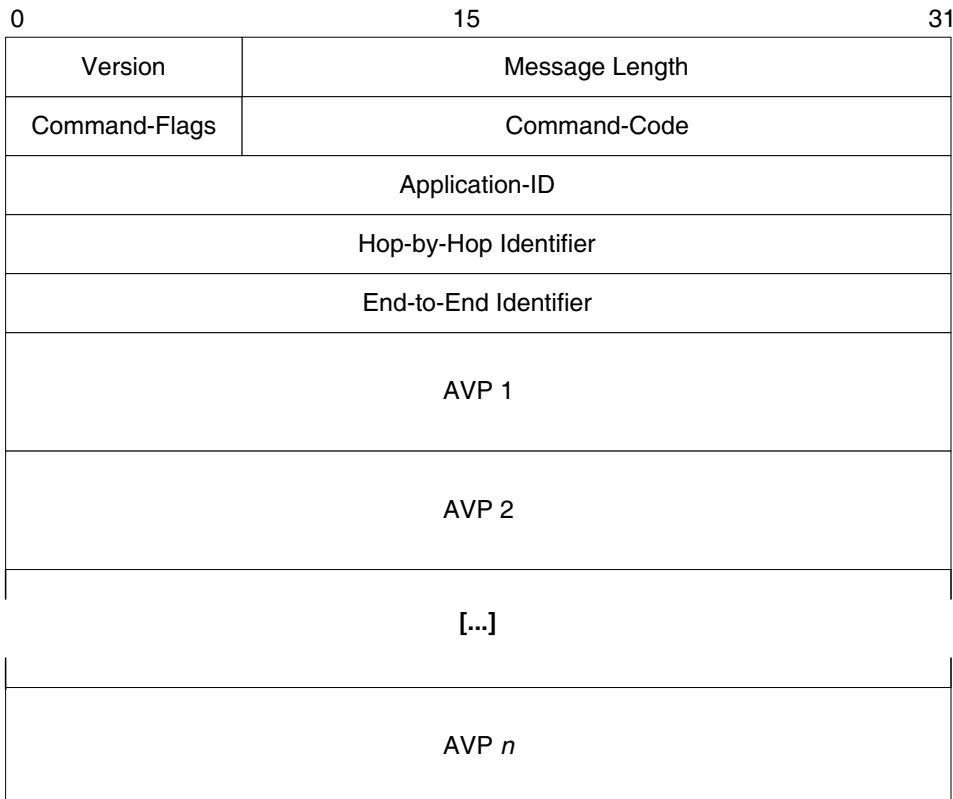


Figure 6.3: Format of a Diameter message

The Diameter header is split into *fields*. According to Figure 6.3 a Diameter header starts with the **Version** field. For the time being, there is only Version 1. A **Message-Length** field containing the length of the Diameter message including all the headers and AVPs follows in the Diameter header.

The **Command-Flags** field indicate:

- whether the message is a request or an answer;
- whether the message is proxiable or not;

- whether the message contains a protocol error according to the format of a Diameter message;
- whether the message is a potentially retransmitted message.

The Command-Code field identifies the actual command (i.e., the actual request or answer). Requests and answers share the same Command-Code address space. A flag present in the Command-Flags field indicates whether the message is a request or an answer.

The Application-ID field identifies the Diameter application that is sending the message. For instance, the Application-ID field can identify the application as the Diameter base protocol, a Network Access Server application, a Mobile IPv4 application, or any other Diameter application.

The Hop-by-Hop Identifier field contains a value that each hop sets when sending a request. The answer has the same Hop-by-Hop Identifier, so a Diameter node can easily correlate the answer with the corresponding request. Each Diameter node that treats a Diameter request changes the value of the Hop-by-Hop Identifier.

The sender of the request sets the value of the End-to-End Identifier field, which is a static value that does not change when Diameter nodes proxy the request. Together with the origin's host identity, the End-to-End Identifier is used to detect duplicate requests. The Diameter node that generates an answer keeps the same value as was found in the request.

6.3.3 Attribute-Value Pairs

Diameter messages, like RADIUS messages, transport a collection of Attribute-Value Pairs (AVPs). An AVP is a container of data. Figure 6.4 depicts the structure of an AVP. Each AVP contains an AVP Code, Flags, an AVP Length, an optional Vendor-ID, and Data.

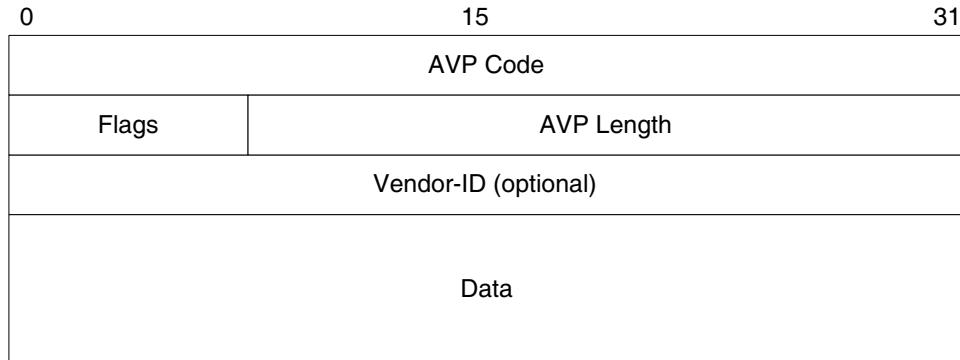


Figure 6.4: Structure of an AVP

The AVP Code, in conjunction with the Vendor-ID field, if present, uniquely identifies the attribute. The absence of a Vendor-ID field or a Vendor-ID field with a value set to zero indicates a standard AVP specified in an IETF specification. AVP code numbers ranging from 1 to 255 identify attributes imported from or already defined by RADIUS. AVP numbers 256 and higher identify Diameter-specific attributes.

The Flags field indicates:

- the need for encryption to guarantee end-to-end security;
- whether support for the AVP is mandatory or optional; if the sender indicates that support for the AVP is mandatory and the receiver does not understand the AVP, the Diameter request is rejected;
- whether the optional Vendor-ID field is present or not.

The AVP Length indicates the length of the AVP, including the AVP Code, AVP Length, Flags, Vendor-ID (if present), and the Data field.

The Data field contains some data specific to the attribute. The field has a length of zero or more octets. The length of the data is derived from the AVP Length field.

The Diameter base protocol specifies a number of formats of the Data field: *OctetString*, *Integer32*, *Integer64*, *Unsigned32*, *Unsigned64*, *Float32*, *Float64*, and *Grouped*. Most of them are self-explanatory. A Grouped AVP is an AVP whose Data field is, in turn, a sequence of other AVPs.

Diameter allows applications to derive AVP data formats. The base protocol already defines a few derived AVPs, the most important ones being the following:

- *Address* to convey an IPv4 or IPv6 address;
- *Time* to represent the date and time;
- *UTF8String* to represent a UTF-8 encoded string;
- *DiameterIdentity* to convey the fully qualified domain name of a Diameter node;
- *DiameterURI* to convey an AAA or AAAS URI;
- *Enumerated*, a numerical value that represents some semantics.

6.3.4 The AAA and AAAS URIs

AAA protocols are able to use an *aaa* or an *aaas* URI to identify AAA resources. The *aaas* URI indicates that transport security must be used. The syntax of these URIs is

```
"aaa://" FQDN [ port ] [ transport ] [ protocol ]
"aaas://" FQDN [ port ] [ transport ] [ protocol ]

port      = ":" 1*DIGIT
transport = ";transport=" transport-protocol
protocol  = ";protocol=" aaa-protocol

transport-protocol = ( "tcp" / "sctp" / "udp" )
aaa-protocol      = ( "diameter" / "radius" /
                      "tacacs+" )
```

where FQDN is a Fully Qualified Domain Name. The URIs might be appended by an optional port number, an optional *transport* protocol, or an optional *protocol* to access the AAA resource. If the port number is not present, the default Diameter port number (3868) is assumed. If the *transport* parameter is not present, then the default SCTP protocol is assumed. If the *protocol* parameter is not present, Diameter is assumed. The reader should note that the *aaa* and *aaas* URIs are able to accommodate Diameter, RADIUS, and other protocols.

Examples of *aaa* or *aaas* URIs include

```
aaa://server.home1.net
aaas://server.home1.net
aaa://server.home1.net:8868
aaa://server.home1.net;transport=tcp;protocol=diameter
```

6.3.5 Diameter Base Protocol Commands

We have seen that Diameter messages are either requests or answers. A request and its corresponding answer are identified by a common *Command-Code* in the Diameter header. The *Command-Code* is a number that indicates the action the Diameter server needs to take. As a request and its corresponding answer share the same *Command-Code* address space, we need to refer to the *Command-Flags* to find out if the command is a request or an answer.

The Diameter base protocol (RFC 3588 [96]) specifies an initial number of command codes. An application is able to extend these basic commands and add new application-specific ones. Table 6.1 shows the initial set of requests and answers defined in the Diameter base protocol.

Table 6.1: Diameter base commands

<i>Command-Name</i>	<i>Abbreviation</i>	<i>Command-Code</i>
Abort-Session-Request	ASR	274
Abort-Session-Answer	ASA	274
Accounting-Request	ACR	271
Accounting-Answer	ACA	271
Capabilities-Exchange-Request	CER	275
Capabilities-Exchange-Answer	CEA	275
Device-Watchdog-Request	DWR	280
Device-Watchdog-Answer	DWA	280
Disconnect-Peer-Request	DPR	282
Disconnect-Peer-Answer	DPA	282
Re-Auth-Request	RAR	258
Re-Auth-Answer	RAA	258
Session-Termination-Request	STR	275
Session-Termination-Answer	STA	275

Typically, every message is abbreviated by its initials. For instance, the Abort-Session-Request message is typically referred to as the ASR message. Let us take a look at the semantics of the main Diameter messages.

6.3.5.1 Abort Session Request and Answer (ASR, ASA)

It might be necessary for a Diameter server to stop the service provided to the user (e.g., network access) because, say, there are new reasons that were not anticipated when the session was authorized. Among others, lack of credit, security reasons, or just an administrative order may be reasons to abort an ongoing Diameter session.

When a Diameter server decides to instruct the Diameter client to stop providing a service, the Diameter server sends an Abort-Session-Request (ASR) message to the Diameter client. The Diameter client reports the execution of the command in an Abort-Session-Answer (ASA).

6.3.5.2 Accounting Request and Answer (ACR, ACA)

A Diameter node may need to report accounting events to a Diameter server that provides accounting services. Diameter provides the Accounting-Request (ACR) command, whereby a Diameter client reports usages of the service to a Diameter server. The command includes information that helps the Diameter server to record the one-time event that generated the command or the beginning or end of a service (e.g., access to a network).

6.3.5.3 Capabilities Exchange Request and Answer (CER, CEA)

The first Diameter messages that two Diameter nodes exchange, once the transport connection is established, are the Capabilities-Exchange-Request (CER) and the Capabilities-Exchange-Answer (CEA). The messages carry the node's identity and its capabilities (protocol version, the supported Diameter applications, the supported security mechanisms, etc.).

6.3.5.4 Device Watchdog Request and Answer (DWR, DWA)

It is essential for Diameter to detect transport- and application-layer failures as soon as possible, in order to take corrective action. The mechanism that Diameter provides to detect these failures is based on an application-layer watchdog. During periods of traffic between two Diameter nodes, if a node sends a request and no answer is received within a certain time period, that is enough to detect a failure either at the transport or application layer. However, in the absence of regular traffic it is not possible to detect such a potential failure. Diameter solves the problem by probing the transport and application layer by means of a Diameter node sending a DWR message. The absence of the receipt of a DWA message is enough for it to be concluded that a failure has occurred.

6.3.5.5 Disconnect Peer Request and Answer (DPR, DPA)

A Diameter node that has established a transport connection with a peer Diameter node may want to close the transport connection, (e.g., if it does not foresee more traffic toward the peer node). In this case the Diameter node sends a Disconnect-Peer-Request (DPR) to the peer node to indicate the imminent disconnection of the transport protocol. The DPR message also conveys the semantics of requesting the peer not to re-establish the connection unless it is essential (e.g., to forward a message).

6.3.5.6 Re-Authentication Request and Answer (RAR, RAA)

At any time, but especially in sessions that last a long time, the Diameter server may request a re-authentication of the user, just to confirm that there is no possible fraud. A Diameter server that wants to re-authenticate a user sends a Re-Auth-Request message to a Diameter client. The client responds with a Re-Auth-Answer message.

6.3.5.7 Session Termination Request and Answer (STR, STA)

A Diameter client reports to the Diameter server that a user is no longer making use of the service by sending a Session-Termination-Request (STR) message. The Diameter server answers with a Session-Termination-Answer (STA) message.

For instance, if the dial-up server reports that the dial-up connection has dropped, then the Diameter client sends the STR message to the Diameter server.

6.3.6 *Diameter Base Protocol AVPs*

Each request and answer defines which Attribute-Value Pairs (AVPs) are present in the message. Some AVPs may be optional in a particular request or answer; others may be mandatory.

The presence or absence of standard defined AVPs are dependent on the actual request or response. For instance, the `Authorization-Lifetime` AVP indicates the period of time for which the authorization of a user is valid. Once the authorization lifetime is reached, the Diameter client will re-authenticate the user. This AVP is only present in authorization answer messages.

The list of Diameter base protocol AVPs is quite large; we refer the reader to RFC 3588 [96] to get the complete list. We describe in the following paragraphs a few important AVPs that are very often found in Diameter messages.

The `User-Name` AVP indicates the username under which the user is known in the realm. The `User-Name` AVP follows the format of the Network Access Identifier (NAI) specified in RFC 2486 [72]. An NAI has either the format of `username` or `username@realm`. In the IMS the `User-Name` AVP carries the Private User Identity.

Every Diameter answer message carries a `Result-Code` AVP. The value of the `Result-Code` AVP indicates whether the request was successfully completed or not and it gives a list of possible values of the AVP that depend on the actual request and answer.

The `Origin-Host` AVP conveys the fully qualified domain name of the Diameter node that generates the request. The node also includes the realm of the Diameter node in the `Origin-Realm` AVP.

The `Destination-Host` AVP indicates the fully qualified domain name of the Diameter server where the username is defined. Sometimes the user does not know the actual host name of the server, but does know the administrative domain where the username is valid. In that case the `Destination-Realm` AVP is used.

Diameter request messages can be proxiable or non-proxiable. There is a flag in the Diameter header that indicates whether the message is proxiable or not. Proxiable messages can be routed through proxies toward the destination realm. Therefore, a proxiable request always contains a `Destination-Realm` AVP. Non-proxiable messages are just routed to the next hop and are never forwarded.

Another interesting AVP is the `Session-ID` AVP. It contains a global identifier of the session. All messages pertaining to the same session contain the same `Session-ID` AVP value. Section 6.3.1 describes the concept of session in the context of Diameter.

The **Vendor-Specific-Application-Id** is a grouped AVP that conveys the identity of a Diameter application that is vendor-specific (e.g., not standardized in the IETF). The **Vendor-Specific-Application-Id** contains either an **Auth-Application-Id** or an **Acct-Application-id** AVP, although only one of them can be present at the same time. The former identifies the authentication and authorization portion of the application, whereas the latter identifies the accounting portion of the application. The **Vendor-Specific-Application-Id** AVP can also contain a **Vendor-Id** AVP that identifies the vendor.

The **Auth-Session-State** AVP indicates whether the Diameter client wants to maintain a state for a particular Diameter session. The Diameter client uses this AVP as a request, and the Diameter server replies with the same AVP in the answer.

The **Proxy-Info** is a grouped AVP that contains a **Proxy-Host** and a **Proxy-State** AVP; it may also contain any other AVP. It allows stateless Diameter nodes to include a state in a request. The corresponding answer will contain the same AVP, so the stateless Diameter node can retrieve the state information and proceed with the Diameter session. The **Proxy-Host** AVP contains the host name of the proxy that inserted the information. The **Proxy-State** AVP contains the opaque data that are written and then read by the proxy itself.

A relay or proxy agent appends a **Route-Record** AVP to all the requests. The **Route-Record** AVP contains the identity of the Diameter node the request was received from. This allows the detection of loops. It also allows the Diameter server to verify and authorize the path a request took.

Chapter 7

AAA in the IMS

Authentication and authorization are generally linked in the IMS. In contrast, accounting is a separate function executed by different nodes. This was the reason why we decided to separate the description of authentication and authorization from the description of accounting. Section 7.1 discusses authentication and authorization, and Section 7.4 discusses accounting.

7.1 Authentication and Authorization in the IMS

Figure 7.1 shows the IMS architecture for performing authentication and authorization functions. There are three interfaces over which authentication and authorization actions are performed (namely the *Cx*, *Dx*, and *Sh* interfaces).

The *Cx* interface is specified between a Home Subscriber Server (HSS) and either an I-CSCF or an S-CSCF. When more than a single HSS is present in a home network there is a need for a Subscription Locator Function (SLF) to help the I-CSCF or S-CSCF to determine which HSS stores the data for a certain user. The *Dx* interface connects an I-CSCF or S-CSCF to an SLF running in Diameter redirect mode.

The *Sh* interface is specified between an HSS and either a SIP Application Server or an OSA Service Capability Server (for a complete description of the different types of Application Server in the IMS, see Section 5.8.2).

In all of these interfaces the protocol used between any two nodes is Diameter (specified in RFC 3588 [96]) with an IMS-specific tailored application. Such a Diameter application defines new Diameter command codes and new Attribute-Value Pairs (AVPs).

7.2 The *Cx* and *Dx* Interfaces

The *Cx* interface is specified between an I-CSCF and an HSS or between an S-CSCF and an HSS. Similarly, the *Dx* interface is specified between an I-CSCF and an SLF or between an S-CSCF and an SLF. The only difference between these interfaces is that the SLF implements the functionality of a Diameter redirect agent, whereas the HSS acts as a Diameter server. In either case, both the I-CSCFs and S-CSCFs act as Diameter clients.

In networks where there is more than a single HSS, Diameter clients (S-CSCF, I-CSCF) need to contact the SLF to find out which of the several HSSs in the network stores the user information for the user identified by the Public User Identity. The Diameter command the

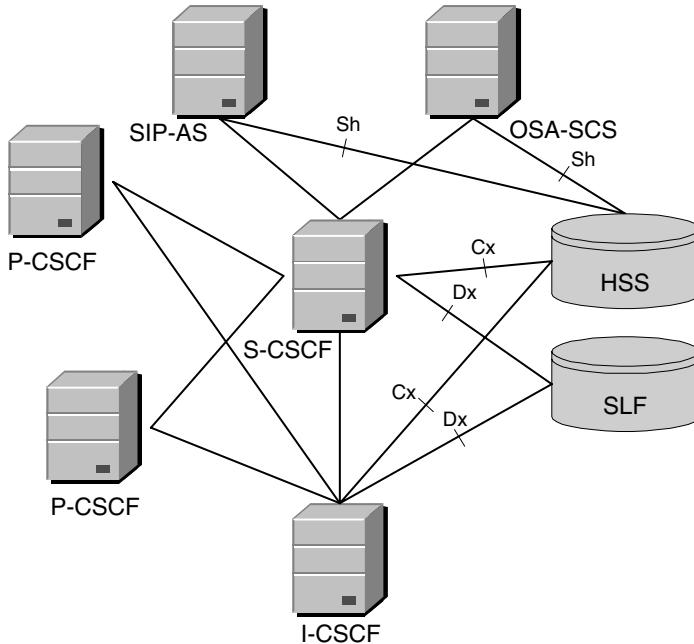


Figure 7.1: Architecture for authentication and authorization in the IMS

S-CSCF or the I-CSCF sends is the same, no matter whether the message is addressed to the SLF or the HSS. The SLF acts as an enhanced Diameter redirect agent and contains a table that maps Public User Identities to the address of the HSS that contains the user information. The SLF then includes a *Redirect-Host* AVP in the answer. The *Redirect-Host* AVP contains the address of the particular HSS that the I-CSCF or S-CSCF needs to contact. The I-CSCF or S-CSCF then forwards the Diameter message addressed to that HSS.

Because Diameter messages over the *Cx* and *Dx* interfaces are the same, the *Dx* interface can be considered as transparent to describe the interactions over the *Cx* interface. In this chapter we typically refer to the *Cx* interface and the HSS, but the description applies in a similar manner to the *Dx* interface and the SLF.

Note that the P-CSCF implements neither the *Cx* nor the *Dx* interface.

For a particular user the I-CSCF and S-CSCF use the *Cx* and *Dx* interfaces to perform the following functions:

- to locate an already allocated S-CSCF to the user;
- to download the authentication vectors of the user; these vectors are stored in the HSS;
- to authorize the user to roam in a visited network;
- to record in the HSS the address of the S-CSCF allocated to the user;
- to inform the HSS about the registration state of a user's identity;
- to download from the HSS the user profile that includes the filter criteria;

- to push the user profile from the HSS to the S-CSCF when the user profile has changed;
- to provide the I-CSCF with the necessary information to select an S-CSCF.

The *Cx* and *Dx* interfaces implement a vendor-specific Diameter application called the *Diameter Application for the Cx Interface*. This application is specified in 3GPP TS 29.228 [40] and 3GPP TS 29.229 [33]. The Diameter Application for the *Cx* Interface is not standardized in the IETF, but the IETF has authorized, for 3GPP Release 5, this application (as specified in RFC 3589 [210]).

The Diameter Application for the *Cx* Interface has formed the basis of a generic AAA application for SIP servers (documented in RFC 4740 [150]). It is foreseen that in future 3GPP releases the vendor-specific Diameter application for the *Cx* Interface will migrate to the IETF standardized Diameter SIP application.

7.2.1 Command Codes Defined in the Diameter Application for the Cx Interface

As previously mentioned, the I-CSCF and S-CSCF perform a number of functions over the *Cx* and *Dx* interfaces. In order to perform these functions the Diameter Application for the *Cx* Interface has defined a number of new commands (requests and answers). Table 7.1 lists the new commands specified in the Diameter Application for the *Cx* Interface.

Table 7.1: List of commands defined by the Diameter Application for the *Cx* Interface

Command-Name	Abbreviation	Command-Code
User-Authorization-Request	UAR	300
User-Authorization-Answer	UAA	300
Server-Assignment-Request	SAR	301
Server-Assignment-Answer	SAA	301
Location-Info-Request	LIR	302
Location-Info-Answer	LIA	302
Multimedia-Auth-Request	MAR	303
Multimedia-Auth-Answer	MAA	303
Registration-Termination-Request	RTR	304
Registration-Termination-Answer	RTA	304
Push-Profile-Request	PPR	305
Push-Profile-Answer	PPA	305

7.2.1.1 User Authorization Request and Answer (UAR, UAA)

An I-CSCF sends a User-Authorization-Request (UAR) message when the I-CSCF receives a SIP REGISTER request from an IMS terminal. There are a few reasons why the I-CSCF sends the Diameter UAR message to the HSS.

- The HSS first filters the Public User Identity contained in the SIP REGISTER request. For instance, the HSS verifies that the Public User Identity is allocated to a legitimate

subscriber of the home network and that the user is a regular non-blocked user (i.e., not blocked for lack of payment or any other reason).

- The HSS also verifies that the home network has a roaming agreement with the network where the P-CSCF is operating. This allows the P-CSCF network to exchange charging records with the home network.
- The I-CSCF also needs to determine whether there is an S-CSCF already allocated to the Public User Identity under registration, before the I-CSCF forwards the SIP REGISTER request to that S-CSCF. If there is not an S-CSCF allocated to the Public User Identity, the I-CSCF will receive the set of capabilities required in the S-CSCF, so that the I-CSCF is able to select an S-CSCF with those capabilities.
- The SIP REGISTER request typically carries the Private User Identity and the Public User Identity of the user. The HSS checks that the Public User Identity can use that Private User Identity for authentication purposes.

Figure 7.2 depicts a typical registration flow. When the I-CSCF receives a SIP REGISTER request (2) it sends the Diameter UAR message to the HSS (3). The HSS sends a Diameter User-Authorization-Answer (UAA) message (4) and then the I-CSCF proceeds with the registration process. The operation is also repeated in (13) and (14), since the I-CSCF does not keep a state in between these registrations. Furthermore, because of DNS load-balancing mechanisms, the I-CSCF that receives the first SIP REGISTER request (2) might not be the same I-CSCF that receives the second SIP REGISTER request (12).

The HSS includes a Result-Code AVP in the UAA message that helps the I-CSCF to determine whether to continue with the registration or to reject it. If the registration is authorized, the UAA message contains AVPs that help the I-CSCF to determine whether there is an S-CSCF already allocated to the user or whether the I-CSCF has to select a new S-CSCF.

7.2.1.2 Multimedia Auth Request and Answer (MAR, MAA)

Figure 7.2 is also valid for describing the Diameter Multimedia-Auth-Request (MAR) and Multimedia-Auth-Answer (MAA) messages. When the S-CSCF receives an initial SIP REGISTER request, it has to authenticate the IMS user. However, in an initial registration the S-CSCF does not have authentication vectors to authenticate the user. These vectors are stored in the HSS. The S-CSCF sends a Diameter MAR message to the HSS with the purpose of retrieving the authentication vectors. In addition, the S-CSCF records its own SIP URI in the user-related data stored in the HSS, so that other CSCFs (e.g., I-CSCFs) or ASes are able to get the URI of the S-CSCF allocated to that particular user by interrogating the HSS.

7.2.1.3 Server Assignment Request and Answer (SAR, SAA)

Figure 7.2 also shows the use of the SAR and SAA messages. When the S-CSCF eventually authenticates the user (actually, the Private User Identity), the Public User Identity is registered and bound to a contact address. At that time the S-CSCF sends the SAR message to the HSS to inform the HSS that the user is currently registered in that S-CSCF. The S-CSCF also requests the user profile associated with that user. The HSS attaches the user profile in the SAA message.

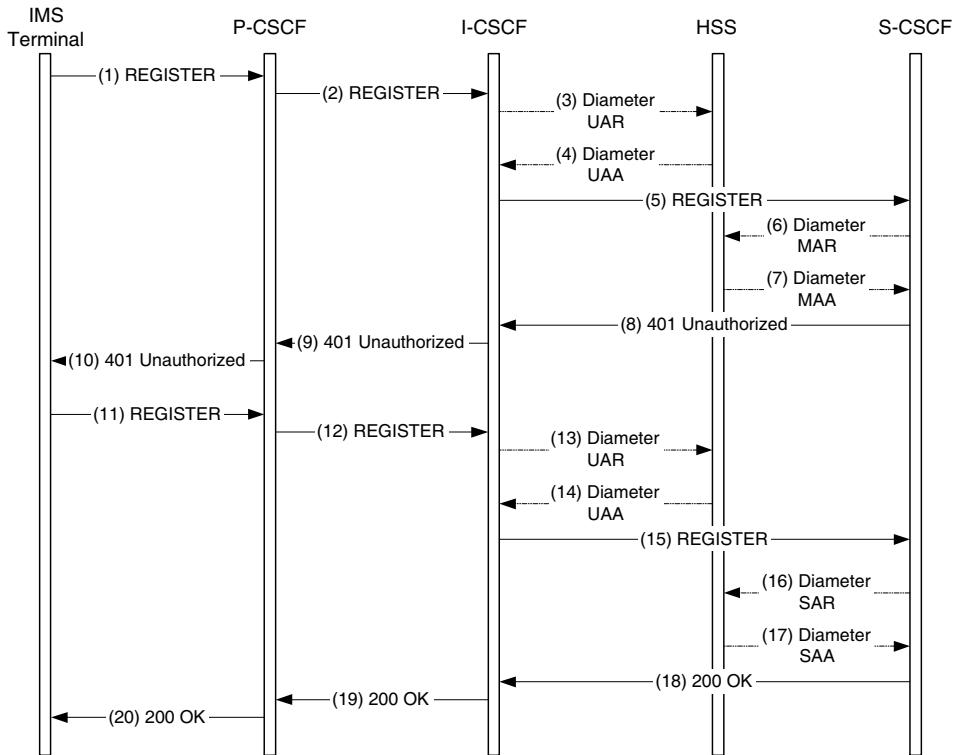


Figure 7.2: UAR/UAA, MAR/MAA, and SAR/SAA messages during registration

The S-CSCF also sends a SAR message to the HSS when the user is no longer registered in that S-CSCF, so that the HSS is aware of the user registration status. The S-CSCF might also request the HSS to continue to be the S-CSCF allocated to the user, even when the user is not registered. The final word belongs to the HSS, because it authorizes whether the S-CSCF keeps the S-CSCF allocation or not. Keeping the S-CSCF allocation allows the S-CSCF to keep the user profile information. Subsequent registrations would not require downloading such information from the HSS.

7.2.1.4 Location Information Request and Answer (LIR, LIA)

An I-CSCF that receives a SIP request that does not contain a Route header field that points to the next SIP hop (S-CSCF) needs to find out which S-CSCF (if any) is allocated to the user. On receiving the SIP request the I-CSCF sends a Location-Info-Request (LIR) to the HSS. The HSS replies with a Location-Info-Answer (LIA). The LIA command indicates the SIP URI of the S-CSCF allocated to the user or, if there is no S-CSCF allocated to the user, then the HSS will include the set of capabilities that are required by the S-CSCF, so that the I-CSCF is able to select an S-CSCF for this user (similar to the selection that takes place during initial registration).

According to Figure 7.3 an I-CSCF that receives a SIP request that does not contain routing information sends a Diameter LIR message to the HSS. The HSS replies with a

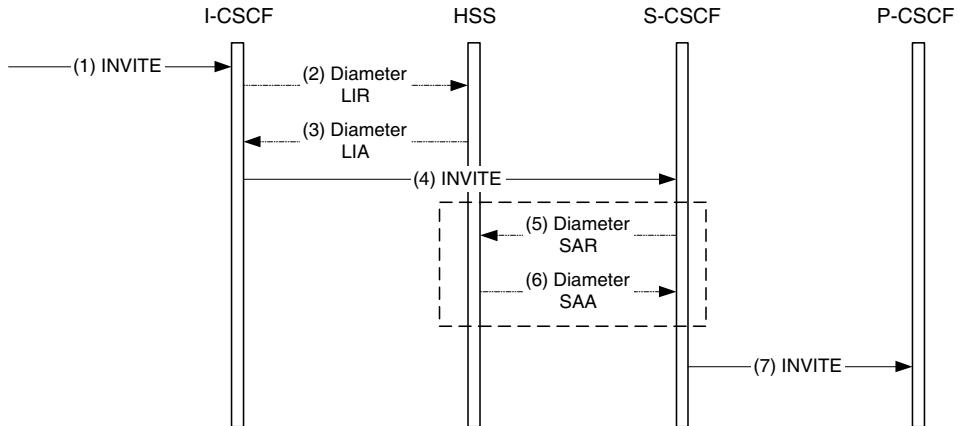


Figure 7.3: LIR/LIA and SAR/SAA messages

Diameter LIA message that contains the address of the S-CSCF that is allocated to the user; therefore, the I-CSCF forwards the INVITE request to that S-CSCF.

7.2.1.5 Registration Termination Request and Answer (RTR, RTA)

Because of administrative action, the operator of the home network may wish to deregister one or more registered Public User Identities allocated to a user. When this happens the HSS sends a Registration-Termination-Request (RTR) message to the S-CSCF where the user is registered.

Figure 7.4 depicts an HSS sending an RTR message to the S-CSCF with the purpose of deregistering one or more Public User Identities. The S-CSCF notifies all the subscribers of the user's reg state; in this case, the P-CSCF and the IMS terminal. In this example both the P-CSCF and the IMS terminal have subscribed to the user's reg state, so the S-CSCF notifies the P-CSCF (3) and the IMS terminal (5) and (6).

7.2.1.6 Push Profile Request and Answer (PPR, PPA)

The HSS may change the user profile at any time, such as when a new service is available to the user; this action typically requires the addition of new filter criteria to the user profile. When the user profile is updated the HSS sends a Push-Profile-Request message to the S-CSCF allocated to the user, with the message containing an updated copy of the user profile. Figure 7.5 shows an example of the PPR and PPA Diameter messages. These messages are not connected to any SIP signaling.

7.2.2 AVPs Defined in the Diameter Application for the Cx Interface

The Diameter Application for the Cx Interface defines a number of new Attribute-Value Pairs. Table 7.2 lists these new attributes together with their AVP code. The Vendor-Id field of all of these AVPs is set to the value 10415, which identifies 3GPP.

The Visited-Network-Identifier AVP conveys an identifier of the network where the P-CSCF is located. An I-CSCF maps the P-Visited-Network-ID header field included

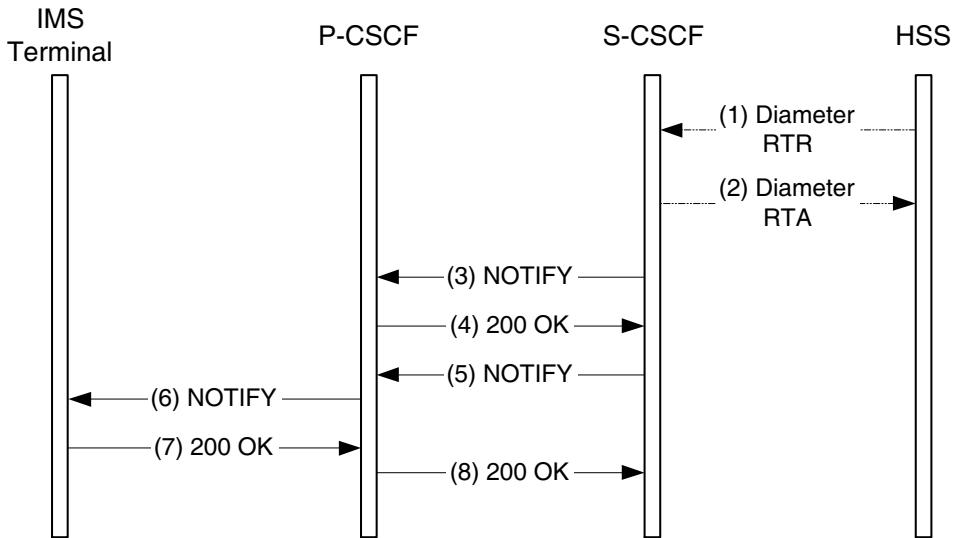


Figure 7.4: RTR/RTA messages

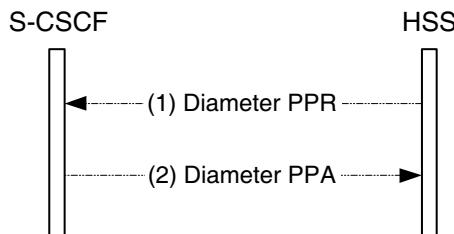


Figure 7.5: PPR/PPA messages

in a SIP REGISTER request to the Visited-Network-Identifier AVP. The home network is able to authorize the IMS terminal to use a P-CSCF located in that network.

The Public-Identity AVP carries a Public User Identity (SIP URI or TEL URI).

The Server-Name AVP contains the URI of the SIP server node (e.g., the S-CSCF URI).

Server-Capabilities is a grouped AVP whose main purpose is to convey the required capabilities of the S-CSCF that will be serving the user. The HSS conveys these capabilities to the I-CSCF, so that the I-CSCF can select an adequate S-CSCF for that user. As this is a grouped AVP it contains, in turn, other AVPs: Mandatory-Capability, Optional-Capability, and Server-Name.

The Mandatory-Capability AVP indicates a capability that the chosen S-CSCF has to implement, whereas the Optional-Capability contains a capability that the S-CSCF may optionally implement. Both AVPs can be repeated a number of times to allow several capabilities to be expressed. Capabilities are represented by an integer. The home operator allocates the semantics of the capabilities to each integer, which is a valid action because the Cx interface is only used inside the home network. For instance, the capability of the S-CSCF

Table 7.2: AVPs defined by the Diameter Application for the *Cx* Interface

Attribute name	AVP code
Visited-Network-Identifier	600
Public-Identity	601
Server-Name	602
Server-Capabilities	603
Mandatory-Capability	604
Optional-Capability	605
User-Data	606
SIP-Number-Auth-Items	607
SIP-Authentication-Scheme	608
SIP-Authenticate	609
SIP-Authorization	610
SIP-Authentication-Context	611
SIP-Auth-Data-Item	612
SIP-Item-Number	613
Server-Assignment-Type	614
Deregistration-Reason	615
Reason-Code	616
Reason-Info	617
Charging-Information	618
Primary-Event-Charging-Function-Name	619
Secondary-Event-Charging-Function-Name	620
Primary-Charging-Collection-Function-Name	621
Secondary-Charging-Collection-Function-Name	622
User-Authorization-Type	623
User-Data-Already-Available	624
Confidentiality-Key	625
Integrity-Key	626
User-Data-Request-Type	627

to execute Java code can be assigned to capability 1, the capability of the S-CSCF to run SIP CGI scripts can be assigned to capability 2, and so on.

The User-Data AVP carries the user profile. The user profile is described in detail in Section 7.2.3.

The S-CSCF indicates how many authentication vectors it wants to receive from the HSS for a particular user in the SIP-Number-Auth-Items AVP. The HSS also uses this AVP to indicate how many authentication vectors it is actually sending.

The SIP-Auth-Data-Item is a grouped AVP that contains the following AVPs: SIP-Item-Number, SIP-Authentication-Scheme, SIP-Authenticate, SIP-Authorization, SIP-Authentication-Context, Confidentiality-Key, and Integrity-Key.

When a Diameter message carries more than one SIP-Auth-Data-Item AVP and the S-CSCF has to consider the order in which to process them, then the HSS includes a

sequential number in the SIP-Item-Number AVP that is included in each SIP-Auth-Data-Item.

The SIP-Authentication-Scheme AVP indicates the authentication scheme that is used for the authentication of SIP messages. 3GPP Release 5 only defines Digest-AKAv1-MD5 as an authentication scheme.

The SIP-Authenticate AVP is used by the HSS to send the value that the S-CSCF inserts in the SIP WWW-Authenticate header field of a 401 Unauthorized response. When the user is authenticated the IMS terminal sends a SIP request that contains an Authorization header field. The value of this header is not sent to the HSS unless there is a failure of synchronization, and in that case the S-CSCF copies the SIP Authorization header to the SIP-Authorization AVP.

The SIP-Authentication-Context is used to carry part of or the complete SIP request to the S-CSCF for certain authentication mechanisms (e.g., the HTTP Digest with quality of protection set to auth-int, specified in RFC 2617 [145]).

Confidentiality-Key and Integrity-Key AVPs contain, respectively, the keys that the P-CSCF needs to encrypt/decrypt or protect/verify the SIP messages sent to or from the IMS terminal. The HSS sends these keys to the S-CSCF in the mentioned AVPs, the S-CSCF inserts these keys as parameters of the Digest scheme in the SIP WWW-Authenticate header field, and then the P-CSCF removes them.

The S-CSCF indicates in the Server-Assignment-Type AVP the reason for the S-CSCF contacting the HSS. Possible reasons are: the S-CSCF requires the user profile; the user has registered, re-registered, or deregistered; a timeout during registration; administrative deregistration; an authentication failure or timeout; etc.

When the HSS deregisters a user, it sends the information to the S-CSCF in a Deregistration-Reason AVP. The Deregistration-Reason is a grouped AVP that contains a Reason-Code AVP and, optionally, a Reason-Info AVP. The Reason-Code AVP is a numerical code that identifies the reason for network-initiated deregistration, such as a permanent termination of the IMS subscription or because a new S-CSCF has been allocated to the user. The Reason-Info contains a readable text string that describes the reason for deregistration.

The Charging-Information is a grouped AVP that conveys to the S-CSCF the AAA URIs of the Event Charging Function (ECF) and Charging Collection Function (CCF) nodes. The AAA URIs of the primary and secondary nodes are sent to the S-CSCF, and, the secondary nodes are used in case of the failure of the corresponding primary nodes. The Charging-Information AVP contains any of the following AVPs:

- Primary-Event-Charging-Function-Name
- Secondary-Event-Charging-Function-Name
- Primary-Charging-Collection-Function-Name
- Secondary-Charging-Collection-Function-Name.

The User-Authorization-Type AVP indicates the type of authorization that an I-CSCF requests in a UAR message. The value can indicate an initial registration or re-registration, a deregistration, or “registration and capabilities”. The I-CSCF uses the “registration and capabilities” value when the current S-CSCF allocated to the user is not reachable and the I-CSCF requests the capabilities of the S-CSCF in order to make a new S-CSCF selection.

In a SAR message the S-CSCF can also indicate to the HSS whether the S-CSCF has already got the user profile. The S-CSCF does this by including a User-Data-Already-Available AVP in the SAR message.

7.2.2.1 Usage of Existing AVPs

Besides the AVPs that 3GPP created to support the Diameter Application for the Cx Interface, the requests and answers of this application also make use of existing AVPs defined in the Diameter base protocol (RFC 3588 [96]). The most important AVPs that 3GPP uses are described in Section 6.3.6. Also important is the User-Name AVP, which in the IMS always carries the Private User Identity.

7.2.3 The User Profile

The user profile, which is stored in the HSS, contains a lot of information related to a particular user. The S-CSCF downloads the user profile when the user registers for the first time with that S-CSCF. The S-CSCF receives the user profile in a User-Data AVP included in a Diameter SAA message. If the user profile changes in the HSS while the user is registered to the network, then the HSS sends the updated user profile in a User-Data AVP included in a Diameter PPR message.

Figure 7.6 shows the structure of the user profile. A user profile is bound to a Private User Identity and to the collection of Public User Identities that are, in turn, associated with that Private User Identity.

The user profile contains a plurality of *service profiles*. Each service profile defines the service triggers that are applicable to a collection of Public User Identities. The service profile is divided into four parts: a collection of one or more *public identifications*, an optional *core network service authorization*, zero or more *initial filter criteria*, and zero or more *shared initial filter criteria*.

The public identifications included in the service profile contain the Public User Identities associated with that service profile. The service profile is applicable to all the identities listed in public identifications. Each public identification contains a tag to indicate whether the Public User Identity is barred or not. A barred Public User Identity can be used for registration purposes, but not for any other SIP traffic (such as establishing a session). A public identification contains either a SIP URI or a TEL URI.

A service profile can also contain a core network service authorization, which, in turn, contains a *subscribed media profile identifier*. The subscribed media profile identifier contains a value that identifies the set of SDP parameters that the user is authorized to request. The identifier, which is stored in the service profile, is just an integer value; the actual SDP profile is stored in the S-CSCF. The S-CSCF uses the subscribed media profile identifier to apply a particular SDP profile that helps the S-CSCF to police SDP in user-initiated requests. For instance, a user might not be authorized to use video. In this case, if the user initiates a session whose SDP contains a video stream, the S-CSCF will reject the session attempt when the S-CSCF evaluates the SDP against the subscribed media profile.

The third part of the information stored in the service profile is a collection of *initial filter criteria*. These determine which SIP requests must visit a certain Application Server so that a particular service can be provided. The initial filter criteria are described in detail in Section 5.8.4.

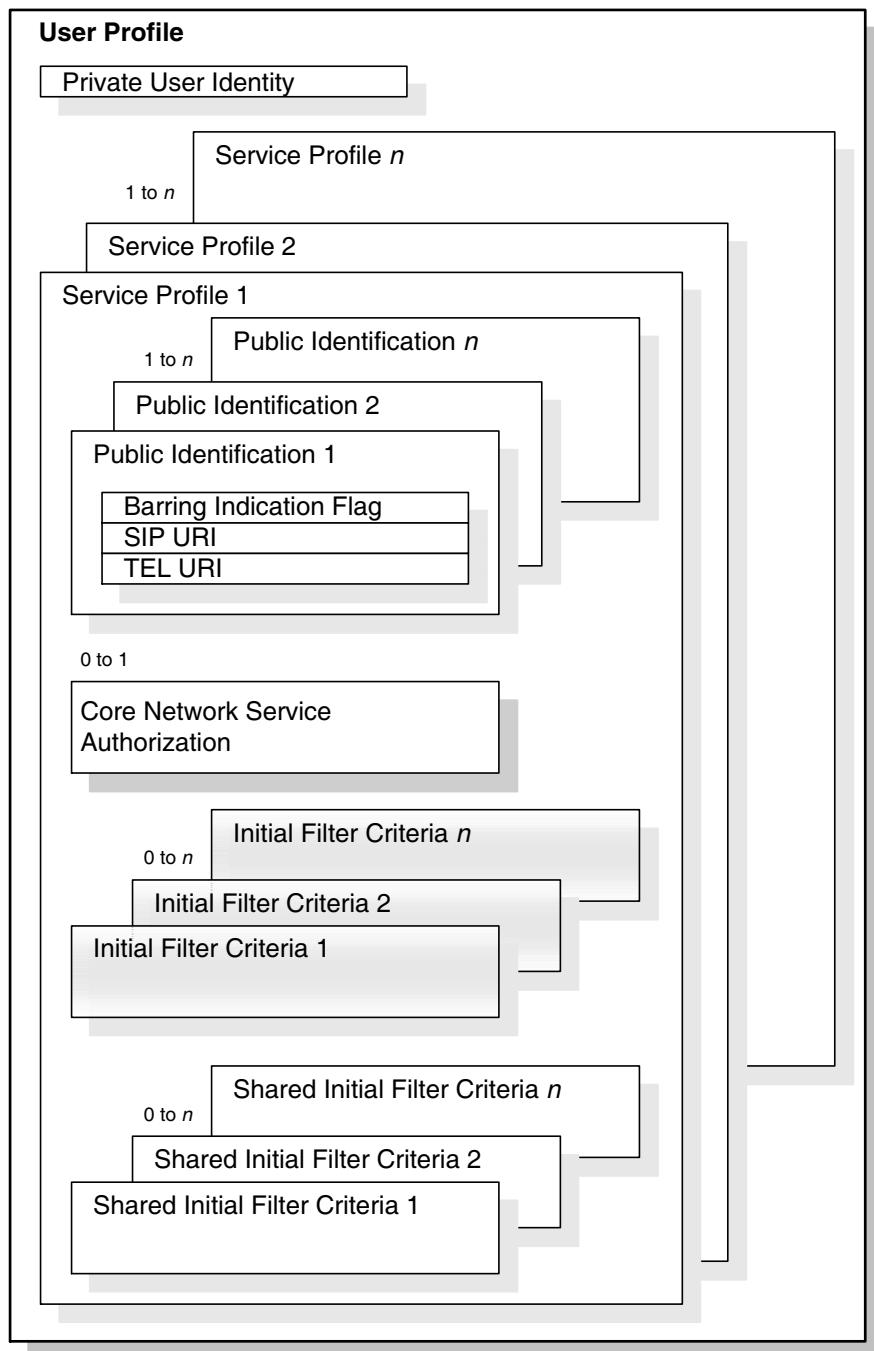


Figure 7.6: Structure of the user profile

The last part of the service profile are the *shared initial filter criteria*. This is an optional feature that requires support by both the S-CSCF and the HSS. Typically, many users in a network might share a collection of initial filter criteria. It is not that optimal if every time a user registers to a S-CSCF, it downloads an initial filter criterion that has already been downloaded previously. Shared initial filter criteria allow the creation of a database of initial filter criteria that are common to a collection of users. The database is stored in both S-CSCFs and HSS. Each shared initial filter criterion is identified by a unique identifier. When a user's service profile contains one or more shared initial filter criteria, only the identifiers are downloaded to the S-CSCF; the S-CSCF has previously stored the shared initial filter criteria in its internal database.

7.3 The *Sh* Interface

The *Sh* interface is defined between a SIP AS or an OSA-SCS and the HSS. It provides a data storage and retrieval type of functionality, such as an Application Server downloading data from the HSS or an Application Server uploading data to the HSS. These data could be service execution scripts or configuration parameters applicable to the user and to a particular service. The *Sh* interface also provides a subscription and notification service, so that the AS can subscribe to changes in the data stored in the HSS. When such data change, the HSS notifies the Application Server.

The implementation of the *Sh* interface is optional in an Application Server and it depends on the nature of the service provided by the Application Server: some services require interaction with the HSS, others do not.

The protocol over the *Sh* interface is Diameter (specified in RFC 3588 [96]) with the addition of a Diameter application (specified in 3GPP TS 29.328 [42] and 3GPP TS 29.329 [54]). This application is known as the *Diameter Application for the Sh Interface*. This is a vendor-specific Diameter application where 3GPP is the vendor; it defines new command codes and AVPs to support the required functionality over the *Sh* interface.

The *Sh* interface introduces the term *user data* to refer to diverse types of data. Most of the Diameter messages over the *Sh* interface operate over some variant of *user data*. User data, in the *Sh* interface context, can refer to any of the following.

Repository data. The AS uses the HSS to store transparent data. The data are only understood by those Application Servers that implement the service. The data are different from user to user and from service to service.

Public identifiers. This is the list of Public User Identities allocated to the user.

IMS user state. This is the registration state of the user in the IMS. This can be registered, unregistered, pending while being authenticated, or unregistered, but an S-CSCF is allocated to trigger services for unregistered users.

S-CSCF name. This contains the address of the S-CSCF allocated to the user.

Initial filter criteria. These contain the triggering information for a service. An Application Server can only get the initial filter criteria that route SIP requests to the requesting Application Server.

Location information. This contains the location of the user in the circuit-switched or packet-switched domains.

User state. This contains the state of the user in the circuit-switched or packet-switched domains.

Charging information. This contains the addresses of the charging functions (primary and secondary event charging function or charging collection function).

7.3.1 Command Codes Defined in the Diameter Application for the *Sh* Interface

The *Sh* interface defines eight new Diameter messages to support the required functionality of the interface. Table 7.3 lists the new commands defined in the Diameter Application for the *Sh* Interface.

Table 7.3: List of commands defined by the Diameter Application for the *Sh* Interface

<i>Command-Name</i>	Abbreviation	<i>Command-Code</i>
User-Data-Request	UDR	306
User-Data-Answer	UDA	306
Profile-Update-Request	PUR	307
Profile-Update-Answer	PUA	307
Subscribe-Notifications-Request	SNR	308
Subscribe-Notifications-Answer	SNA	308
Push-Notifications-Request	PNR	309
Push-Notifications-Answer	PNA	309

7.3.1.1 User Data Request and Answer (UDR, UDA)

An Application Server sends a User-Data-Request (UDR) to the HSS to request user data for a particular user. The user data can be of a type defined over the *Sh* interface. The HSS returns the requested type of data in a Diameter User-Data-Answer (UDA) message. Figure 7.7 depicts the applicable call flow.

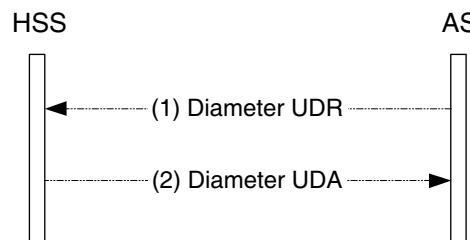


Figure 7.7: UDR/UDA messages

7.3.1.2 Profile Update Request and Answer (PUR, PUA)

An Application Server may modify repository-type user data and store them in the HSS. In doing so the Application Server sends a Profile-Update-Request (PUR) to the HSS. The HSS returns the result of the storage operation in a Diameter Profile-Update-Answer (PUA) message. Figure 7.8 depicts the applicable call flow.

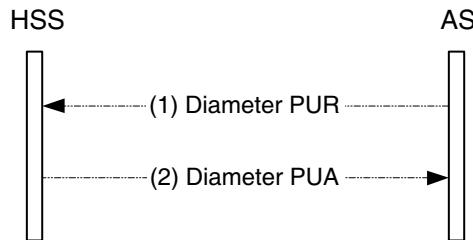


Figure 7.8: PUR/PUA messages

7.3.1.3 Subscribe Notifications Request and Answer (SNR, SNA)

An Application Server can subscribe to changes in the user data by sending a Subscribe-Notifications-Request (SNR) message to the HSS. The types of user data where notifications are allowed are repository data, IMS user state, S-CSCF name, and initial filter criteria. The HSS informs the Application Server of the result of the subscription operation in a Subscribe-Notifications-Answer (SNA). Figure 7.9 shows the flow.

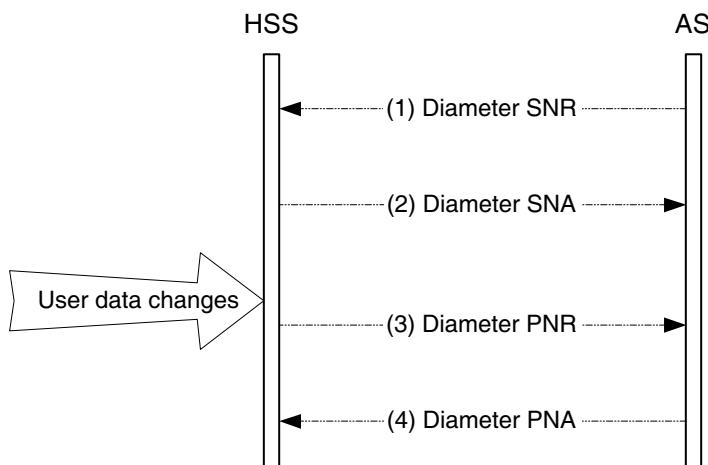


Figure 7.9: SNR/SNA and PNR/PNA messages

7.3.1.4 Push Notification Request and Answer (PNR, PNA)

When changes occur in user data stored in the HSS and an Application Server is subscribed to changes in these user data, the HSS sends a Push-Notification-Request (PNR) to the subscribed Application Server. The PNR message includes a copy of the changed data. The Application Server answers with a Push-Notification-Answer (PNA). The flow is also depicted in Figure 7.9.

7.3.2 AVPs Defined in the Diameter Application for the *Sh* Interface

The Diameter Application for the *Sh* Interface defines a number of new Attribute-Value Pairs. Table 7.4 lists these new attributes together with the AVP code.

Table 7.4: AVPs defined by the Diameter Application for the *Sh* Interface

Attribute name	AVP code
User-Identity	100
MSISDN	101
User-Data	102
Data-Reference	103
Service-Indication	104
Subs-Req-Type	105
Requested-Domain	106
Current-Location	107
Server-Name	108

The User-Identity is a grouped AVP that contains the identity of the user either as a Public User Identity, in which case it contains a Public-Identity AVP (borrowed from the Diameter Application for the *Cx* Interface), or as a Mobile Subscriber Integrated Services Digital Network (MSISDN) number, in which case it contains an MSISDN AVP.

The User-Data AVP contains the user data according to the definition of user data in the *Sh* interface. The type of user data is specified in a Data-Reference AVP, which can contain a value that represents any of the different types of user data.

The Service-Indication AVP contains an identifier of the repository data stored in the HSS. This allows an Application Server that implements several services to store data for each of the services in the HSS and still be able to distinguish and associate each data set with each corresponding service.

The Subs-Req-Type AVP contains an indication of whether the Application Server subscribes to the notification service in the HSS.

The Requested-Domain AVP indicates whether the Application Server is interested in accessing circuit-switched domain data or packet-switched domain data.

The Current-Location AVP indicates whether a procedure called “active location retrieval” should be initiated.

The Server-Name AVP mirrors the AVP with the same name in the Diameter Application for the *Cx* Interface.

7.4 Accounting

Accounting is defined as the collection of resource consumption data for the purposes of capacity and trend analysis, cost allocation, auditing, and billing. Although we will be focusing in this section on the charging (i.e., billing) aspects of accounting, we should keep in mind that accounting records can be used for other purposes as well.

The IMS uses the Diameter protocol to transfer the accounting information that charging in the IMS is based on. The CSCFs inform the charging system about the type and the length of the sessions each user establishes. In addition, the routers (e.g., the GGSN) inform the charging system about media activity during those sessions. The charging system assembles all the accounting information related to each user in order to charge them accordingly.

Charging systems use unique identifiers that are used to correlate the accounting data applying to a particular session received from different entities. So, the accounting records generated by a router and by a CSCF about the same session have the same unique identifier. In this chapter, we describe how these identifiers are delivered to the relevant network elements and how these elements generate accounting information.

Chapter 8

Policy and Charging Control in the IMS

This chapter describes the PCC (Policy and Charging Control) architecture of the IMS. Policy and charging are specified as part of the same architecture (PCC) because they are very much related. Policy control is needed to enforce charging decisions (e.g., when a user runs out of credit the ongoing sessions the user is involved in may have to be terminated). However, there are aspects of policy control that are not related to charging and aspects of charging that are not related to policy control. Policy control also includes QoS control (e.g., users cannot use more than a certain bandwidth during peak hours) and charging also includes accounting (e.g., used for network dimensioning). Policy control in earlier IMS releases (e.g., Release 5) was based on the *Go* interface, which was based on the COPS protocol (specified in RFC 2748 [126]). COPS was used between a Policy Decision Point (PDP) (e.g., a SIP entity) and a Policy Enforcement Point (PEP) (e.g., a gateway) to transmit policy-related information. COPS is not used any longer in later IMS releases. That is why we do not include a Policy on the Internet chapter describing COPS before this chapter.

8.1 PCC Architecture

Figure 8.1 shows the PCC architecture. The PCRF (Policy and Charging Rules Function) is the point where policy decisions are made. In order to make decisions, the PCRF receives information from AFs (Application Functions) and from the SPR (Subscription Profile Repository). An AF (i.e., a P-CSCF or an Application Server) provides the PCRF with information obtained from the session signaling (especially from the SDP in the offer/answer exchanges) over the Diameter-based *Rx* interface. The SPR provides the PCRF with QoS-related information about the user's subscription over the *Sp* interface. The *Sp* interface has not been standardized yet but current deployments use LDAP or Diameter on that interface.

The decisions made by the PCRF are enforced by a gateway (e.g., a GGSN). More precisely, decisions are enforced by the PCEF (Policy and Charging Enforcement Function), which is a logical function within the gateway. The PCRF communicates with the PCEF over the Diameter-based *Gx* interface.

The PCEF communicates with the OCS (Online Charging System) over the Diameter-based *Gy* interface. The OCS performs online credit control over that interface. That is, it can instruct the gateway to close a particular gate if the user runs out of credit.

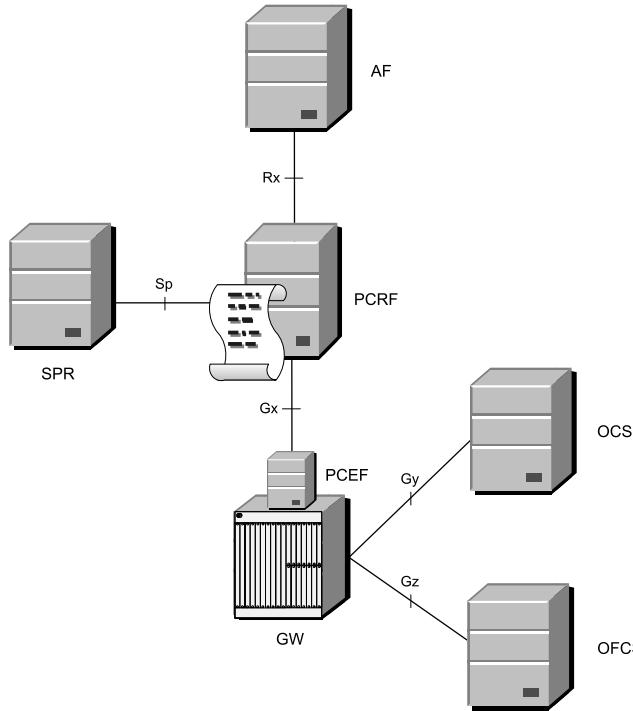


Figure 8.1: PCC architecture

The PCEF communicates with the OFCS (Offline Charging System) over the Diameter-based G_z interface. The OFCS obtains information about data flows to be used for charging purposes.

8.1.1 Session Establishment and Policy Control

Let us see policy control in operation at session establishment. Figures 8.2 and 8.3 show a P-CSCF receiving an INVITE request (1) addressed to one of its terminals. When the P-CSCF obtains both the offer and the answer, it sends an AAR message to the PCRF describing the session to be established. With this description, the PCRF selects a set of PCC rules for the session (in order to select the PCC rules, the PCRF can optionally contact the SPR as well). The PCRF returns an AAA message to the P-CSCF.

Once the PCRF has a description of the session and the set of PCC rules that need to be applied to it, it can use two modes of operation: push or pull. In the push mode, the PCRF pushes the PCC rules to the gateway that will be handling the session. When the gateway receives the PCC rules, it instructs the terminal to establish a bearer. (This procedure is referred to as network-initiated bearer establishment; it triggers the terminal to send a bearer-establishment message.) When the gateway receives the bearer establishment message from the terminal, the gateway applies the PCC rules it has received. In the pull mode, the PCRF does not contact the gateway immediately. Instead, it waits for the gateway to make the first contact. When the gateway receives a bearer establishment message from the terminal, the

gateway requests a set of PCC rules for the session from the PCRF (i.e., the gateway pulls the rules from the PCRF).

Figure 8.2 illustrates the push mode. The PCRF installs the PCC rules in the PCEF by sending a RAR message (7). The PCEF installs the PCC rules, returns a RAA message (8), and instructs the terminal to initiate the PDP Context Activation procedure (9). When the terminal performs the PDP Context Activation procedure (10), the gateway applies the PCC rules to the media flow. The gateway recognizes the media flow using the description (e.g., IP addresses and ports) received from the PCRF earlier. Optionally, the PCEF can send notifications to the PCRF about the media flow using CCR messages (which are responded to with CCA messages by the PCRF).

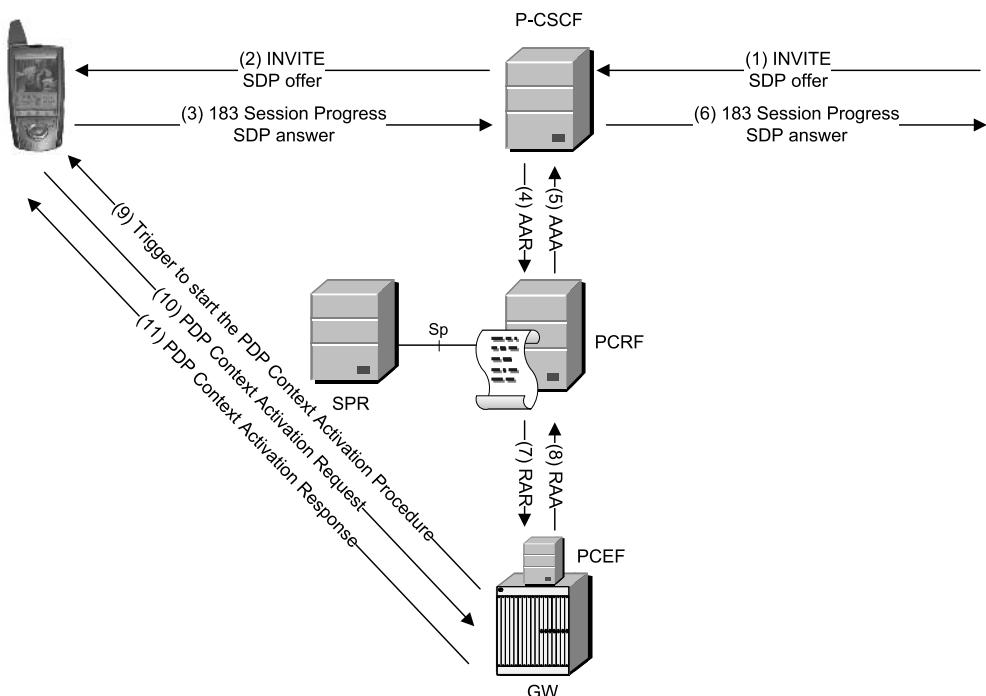


Figure 8.2: Push mode policy control for an incoming INVITE request

Figure 8.3 illustrates the pull mode. When the terminal performs the PDP Context Activation procedure (7), the gateway requests the PCC rules to be applied from the PCRF using a CCR message (8). The PCRF installs the PCC rules in the PCEF by returning a CAA message.

Policy control works in a similar way for an outgoing INVITE request. Figures 8.4 and 8.5 show a P-CSCF receiving an INVITE request (1) generated by one of its terminals. Once the P-CSCF obtains both the offer and the answer (3), the procedures in push and pull modes are identical to the one in Figures 8.2 and 8.3, respectively.

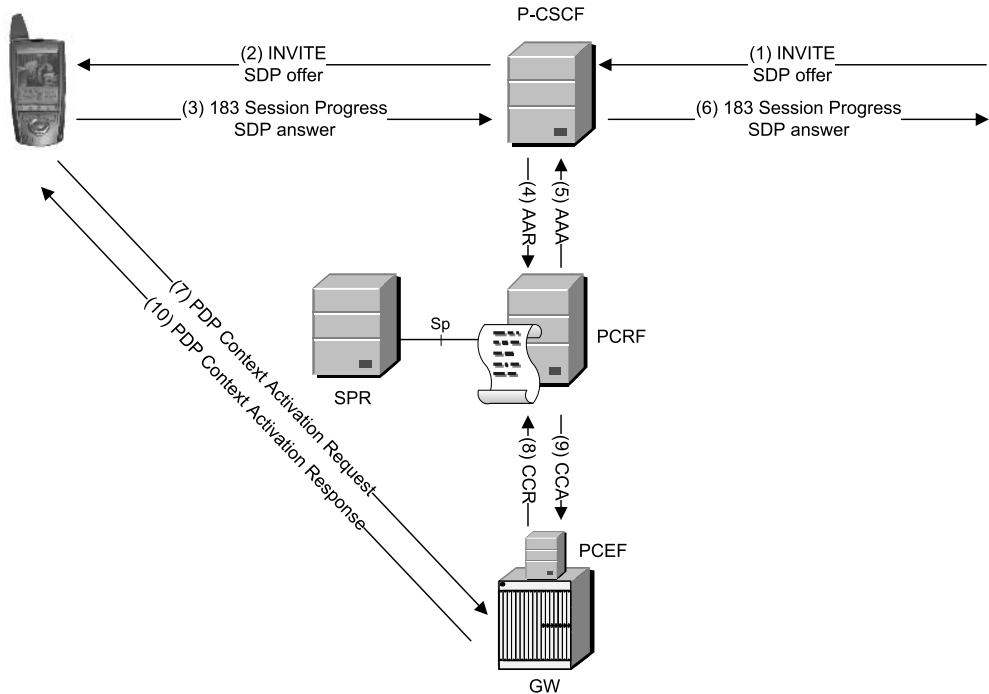


Figure 8.3: Pull mode policy control for an incoming INVITE request

8.1.2 SIP Procedures

There are two types of limitation on the type of sessions a terminal can establish over an IMS network: user-specific limitations and general network policies. An example of a user-specific limitation is an audio-only subscription. A user with such a subscription is not authorized to establish video streams. General network policies are not user-specific, but apply to all the users in the network. For instance, an IMS network may not allow high-bandwidth audio codecs like G.711.

The offer/answer exchanges that a terminal performs need to take into account these limitations. For instance, if a terminal is not allowed to establish video sessions, it should not perform offer/answer exchanges that include video media streams. The P-CSCF and the S-CSCF ensure that unauthorized offer/answer exchanges do not take place.

Both the P-CSCF and the S-CSCF use the same mechanism to keep user agents from performing unauthorized offer/answer exchanges. They rely on having access to the SDP bodies that contain the offer/answer exchanges between user agents. If an offer contains information against the policy (e.g., a forbidden codec) the CSCF returns a 488 (Not Acceptable Here) response with an SDP body describing either the policy or a subset of it. (The P-CSCF can consult the PCRF in order to make this type of policy decision; this process is called early authorization.) For instance, the SDP body can contain a session description that would have been an acceptable offer (e.g., the forbidden codec is removed from the list of codecs). Figure 8.6 shows an S-CSCF using this mechanism. The S-CSCF receives an INVITE request (2) with the session description indicated in Figure 8.7, which contains a video stream in addition to an audio stream. The S-CSCF returns a 488 response (3),

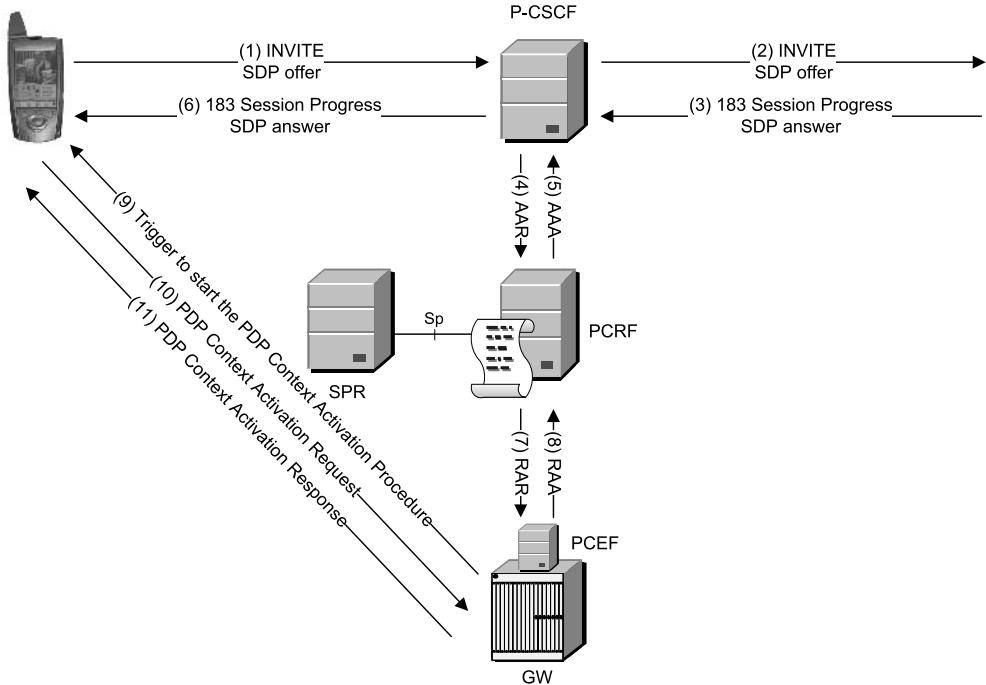


Figure 8.4: Push mode policy control for an outgoing INVITE request

containing the session description indicated in Figure 8.8. This session description includes only an audio stream, indicating that the user cannot establish video streams.

A P-CSCF may send a 488 (Not Acceptable) here response if the AAA response it gets from the PCRF (see Figure 8.4) does not authorize the session the user is trying to establish.

8.1.3 Proxy Access to SDP Bodies

All the IMS policy mechanisms that we have described assume that AFs and S-CSCFs have access to the SDP bodies of the SIP messages. Consequently, the use of end-to-end encryption mechanisms like S/MIME (described in Section 11.4) is not allowed in the IMS. Moreover, user agents are forced to use SDP as their session description format. Otherwise, the AFs and S-CFCSS would not be able to understand which type of session was being established.

Furthermore, Section 10.2 describes how the P-CSCF modifies SDP bodies to convey QoS information to the terminals (e.g., the P-CSCF can instruct the terminal to group several media flows into a single QoS reservation flow). So, end-to-end integrity protection mechanisms are not allowed in the IMS either.

The IETF is working on extensions to allow proxy servers to communicate different policies to the user agents without accessing end-to-end bodies such as session descriptions (see the Internet-Draft “A framework for Session Initiation Protocol (SIP) session policies” [166]). Unfortunately, these extensions will not be ready for some time. So we do not expect the IMS to adopt them in the immediate future.

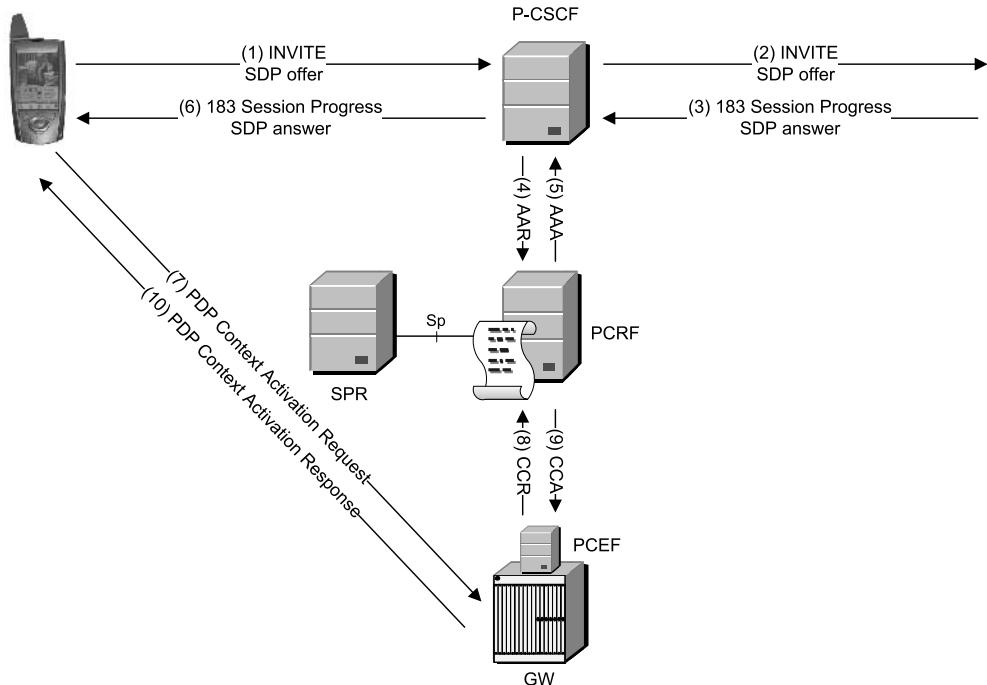


Figure 8.5: Pull mode policy control for an outgoing INVITE request

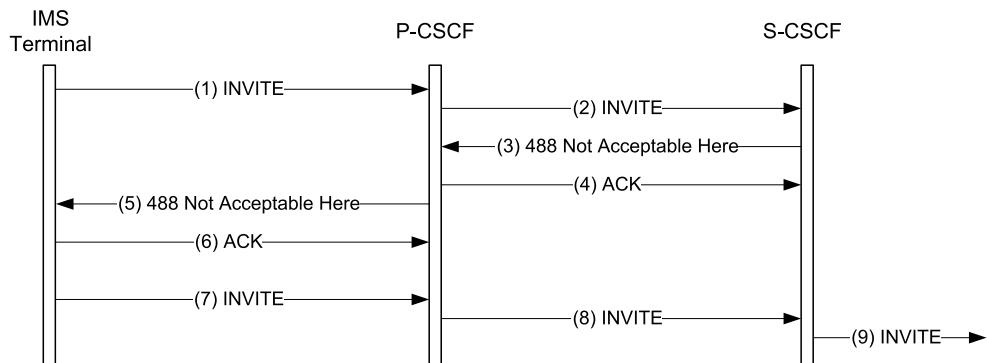


Figure 8.6: The S-CSCF does not accept the terminal's offer

8.1.4 Status of the Signaling Bearer

The PCC architecture also supports the delivery of status information about the bearer used by a given terminal to send and receive SIP signaling (this is specified in 3GPP TS 29.214 [50]). A P-CSCF can subscribe to the status of a terminal's signaling bearer using the *Rx* interface. If the PCRF informs the P-CSCF that the terminal's signaling bearer has been lost, the P-CSCF rejects incoming messages intended for the terminal (as specified in 3GPP TS 24.229 [37]).

```

v=0
o=- 2790844676 2867892807 IN IP6 1080::8:800:200C:417A
s=-
c=IN IP6 1080::8:800:200C:417A
t=0 0
m=audio 20000 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local none
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event
m=video 20002 RTP/AVP 31

```

Figure 8.7: SDP in the INVITE request

```

v=0
o=S-CSCF 2790844634 2867892823 IN IP6 1080::8:800:200C:2A2F
s=-
c=IN IP6 1080::8:800:200C:2A2F
t=0 0
m=audio 0 RTP/AVP 97 96
b=AS:25.4
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 8.8: SDP in the 488 response

The P-CSCF can also terminate an ongoing session by sending a BYE request if the terminal's signaling bearer is lost.

8.1.5 The Rx Interface

As Figure 8.1 shows, the *Rx* interface (which is specified in 3GPP TS 29.214 [50]) is defined between the PCRF and an AF. The Diameter Application for the *Rx* Interface is a vendor-specific Diameter application whose vendor identifier is 10415 (this identifier identifies 3GPP) and whose application identifier is 16777236.

Table 8.1 shows the Diameter commands used over the *Rx* interface. Table 8.2 shows the AVPs specific to the *Rx* application. Table 8.3 shows the AVPs that are defined by other applications and are reused in the *Rx* application.

8.1.6 The Gx Interface

As Figure 8.1 shows, the *Gx* interface (which is specified in 3GPP TS 29.212 [49]) is defined between the PCRF and a PCEF. The Diameter application for the *Gx* interface is a vendor-

Table 8.1: List of commands used by the Diameter Application for the *Rx* Interface

<i>Command-Name</i>	Abbreviation
AA-Request	AAR
AA-Answer	AAA
Re-Auth-Request	RAR
Re-Auth-Answer	RAA
Session-Termination-Request	STR
Session-Termination-Answer	STA
Abort-Session-Request	ASR
Abort-Session-Answer	ASA

Table 8.2: AVPs specific to the Diameter Application for the *Rx* Interface

Attribute name
Abort-Cause
Access-Network-Charging-Address
Access-Network-Charging-Identifier
Access-Network-Charging-Identifier-Value
AF-Application-Identifier
AF-Charging-Identifier
Codec-Data
Flow-Description
Flow-Number
Flows
Flow-Status
Flow-Usage
Specific-Action
Max-Requested-Bandwidth-DL
Max-Requested-Bandwidth-UL
Media-Component-Description
Media-Component-Number
Media-Sub-Component
Media-Type
RR-Bandwidth
RS-Bandwidth
SIP-Forking-Indication
Service-URN
Acceptable-Service-Info
Service-Info-Status-AVP

Table 8.3: AVPs reused in the Diameter Application for the *Rx* Interface

Attribute name
Reservation-Priority
Framed-IP-address
Framed-IPv6-Prefix
3GPP-RAT-Type
IP-CAN-Type

specific Diameter application whose vendor identifier is 10415 (this identifier identifies 3GPP) and whose application identifier is 16777238.

Table 8.4 shows the Diameter commands used over the *Gx* interface. Table 8.5 shows the AVPs specific to the *Gx* application. Table 8.6 shows the AVPs that are defined by other applications and are reused in the *Gx* application.

Table 8.4: List of commands used by the Diameter Application for the *Gx* Interface

Command-Name	Abbreviation
Credit-Control-Request	CCR
Credit-Control-Answer	CCA
Re-Auth-Request	RAR
Re-Auth-Answer	RAA

8.2 Charging Architecture

The IMS charging architecture (described in 3GPP TS 32.240 [55]) defines two charging models: offline charging and online charging. Offline charging is applied to users who pay for their services periodically (e.g., at the end of the month). Online charging, also known as credit-based charging, is used to charge for prepaid services. Both the online and the offline charging models may be applied to the same session. A user may, for example, have a permanent subscription for making voice calls and a prepaid card with credit worth ten minutes of video.

8.3 Offline Charging Architecture

Figure 8.9 shows the IMS offline charging architecture. Offline charging can be applied to sessions or to events (e.g., the sending of an instant message). Network elements that can collect accounting metrics implement the CTF (Charging Trigger Function). The CTF generates charging events based on its accounting metrics and sends these charging events to the CDF (Charging Data Function) over the Diameter-based *Rf* interface. In Figure 8.9, the P-CSCF, the S-CSCF, the I-CSCF, the AS, the MRFC, the MGCF, and the BGCF all implement the CTF.

The CDF uses the information in the charging events received over the *Rf* interface to generate CDRs (Charging Data Records). The CDF sends these CDRs to the CGF (Charging

Table 8.5: AVPs specific to the Diameter Application for the *Gx* Interface

Attribute name
Bearer-Usage
Charging-Rule-Install
Charging-Rule-Remove
Charging-Rule-Definition
Charging-Rule-Base-Name
Charging-Rule-Name
Event-Trigger
Metering-Method
Offline
Online
Precedence
Reporting-Level
TFT-Filter
TFT-Packet-Filter-Information
ToS-Traffic-Class
QoS-Information
QoS-Class-Identifier
Charging-Rule-Report
PCC-Rule-Status
Bearer-Identifier
Bearer-Operation
Access-Network-Charging-Identifier-Gx
Bearer-Control-Mode
Network-Request-Support
Guaranteed-Bitrate-DL
Guaranteed-Bitrate-UL
IP-CAN-Type

Gateway Function) over the GTP (GRPS Protocol) based *Ga* interface. The CGF acts as a gateway between the IMS and the billing domain, with which the CGF communicates using the *Bx* interface. The *Bx* interface is based on a file transfer protocol (e.g., FTP) that is used to transfer the CDRs (as specified in 3GPP TS 32.297 [56]).

The OFCS (Offline Charging System) does not only collect data from IMS nodes and ASes. The OFCS can also collect data from non-IMS nodes such as a GGSN. Figure 8.9 shows a GGSN that communicates to the CGF over the *Ga* interface. However, note that in Figure 8.1, the reference point between the GGSN (GW) and the OFCS was *Gz*. In this type of charging, the *Ga* and the *Gz* interfaces are equivalent.

The contents and the format of the CDRs depend on the node providing the charging events (the format for IMS nodes is specified in 3GPP TS 32.260 [59]). Different billing domains exchange information using non-standard means.

Table 8.6: AVPs reused in the Diameter Application for the Gx Interface

Attribute name
3GPP-RAT-Type
3GPP-SGSN-Address
3GPP-SGSN-IPv6-Address
3GPP-SGSN-MCC-MNC
3GPP-User-Location-Info
Access-Network-Charging-Address
Access-Network-Charging-Identifier-Value
AF-Charging-Identifier
Called-Station-ID
CC-Request-Number
CC-Request-Type
Charging-Information
Flow-Description
Flows
Flow-Status
Framed-IP-Address
Framed-IPv6-Prefix
Max-Requested-Bandwidth-UL
Max-Requested-Bandwidth-DL
Rating-Group
Service-Identifier
Subscription-Id
User-Equipment-Info

8.3.1 Charging-related SIP Header Fields

Let us see how all the different architectural elements interact when a session is established. In addition to the interfaces described previously, SIP entities exchange charging information among each other using SIP header fields. There are two SIP header fields (both specified in RFC 3455 [154]) that carry charging-related information in the IMS: P-Charging-Vector and P-Charging-Function-Addresses.

8.3.2 IMS Terminal in a Visited Network

Figure 8.10 shows a session establishment message flow involving a roaming user when offline charging and push mode policy control are used. We only show the caller’s home and visited domains, since the flow in the callee’s home and visited domains is similar to that of the caller.

The P-CSCF receives an initial INVITE (1) and generates a globally unique identifier called ICID (IMS Charging Identity). This ICID identifies the session that is going to be established for charging purposes. The P-CSCF places the ICID in the P-Charging-Vector header field shown in Figure 8.11 and relays the INVITE to the S-CSCF (2).

The S-CSCF inserts the caller’s home network IOI (Inter-Operator Identifier) into the originating IOI parameter (`orig-roi`) in the P-Charging-Vector header field. The originating IOI helps the callee’s home network to identify the caller’s network so that both

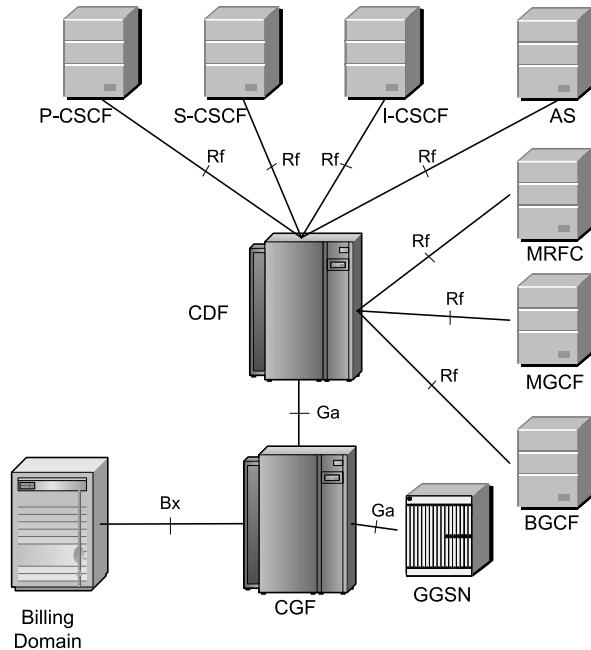


Figure 8.9: IMS offline charging architecture

networks can exchange charging records. The INVITE (3) sent to the callee's home network contains both the ICID and the originating IOI, as shown in Figure 8.12.

The S-CSCF in the callee's home network inserts a terminating IOI parameter (`term-ioi`) into the P-Charging-Vector header field of the 183 (Session Progress) (4) response. The `term-ioi` parameter helps the caller's home network to identify the callee's home network so that both networks can exchange charging information. An example of this P-Charging-Vector header field is shown in Figure 8.13.

The S-CSCF removes the IOIs from the P-Charging-Vector header field before relaying the 183 (Session Progress) response to the P-CSCF in the caller's visited network. The resulting P-Charging-Vector header field in the 183 (Session Progress) (5) response is shown in Figure 8.14.

The CDF only provides an intra-operator interface. That is, only nodes located in the same administrative domain as the CDF report to it. Since the P-CSCF and the S-CSCF in our example are located in different networks, they send accounting information related to the session to different CDFs. The way the P-CSCF chooses an appropriate CDF for the session is not specified, but this choice is typically based on local configuration. The S-CSCF receives a number of CDF addresses from the HSS as part of the Diameter SAA message when the user registers with the network. The S-CSCF uses one of these addresses to send accounting information about the session. The remaining addresses are backup addresses for reliability purposes.

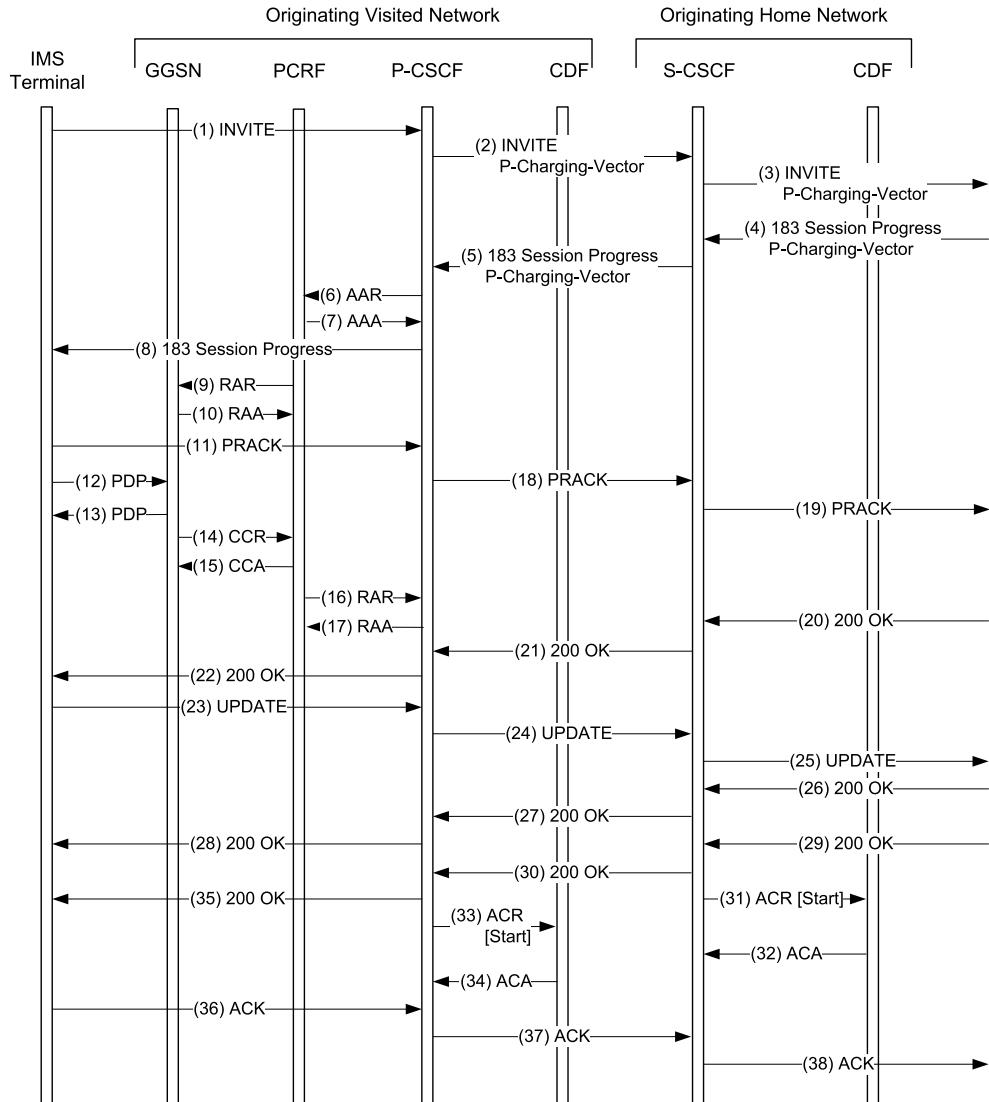


Figure 8.10: Session establishment flow

P-Charging-Vector: `icid-value="AyretyU0dm+602IrT5tAFrbHLso="`

Figure 8.11: P-Charging-Vector value in INVITE (2)

8.3.3 IMS Terminal in its Home Network

Figure 8.15 shows a session establishment message flow for a user in their home network when offline charging and push mode policy control are applied. We only show the caller's home domain, since the flows in the callee's domain are similar.

```
P-Charging-Vector: icid-value="AyretyU0dm+602IrT5tAFrbHLso=";
orig-ioi=home1.net
```

Figure 8.12: P-Charging-Vector value in INVITE (3)

```
P-Charging-Vector: icid-value="AyretyU0dm+602IrT5tAFrbHLso=";
orig-ioi=home1.net;
term-ioi=home2.net
```

Figure 8.13: P-Charging-Vector value in 183 (Session Progress) (4)

```
P-Charging-Vector: icid-value="AyretyU0dm+602IrT5tAFrbHLso="
```

Figure 8.14: P-Charging-Vector value in 183 (Session Progress) (5)

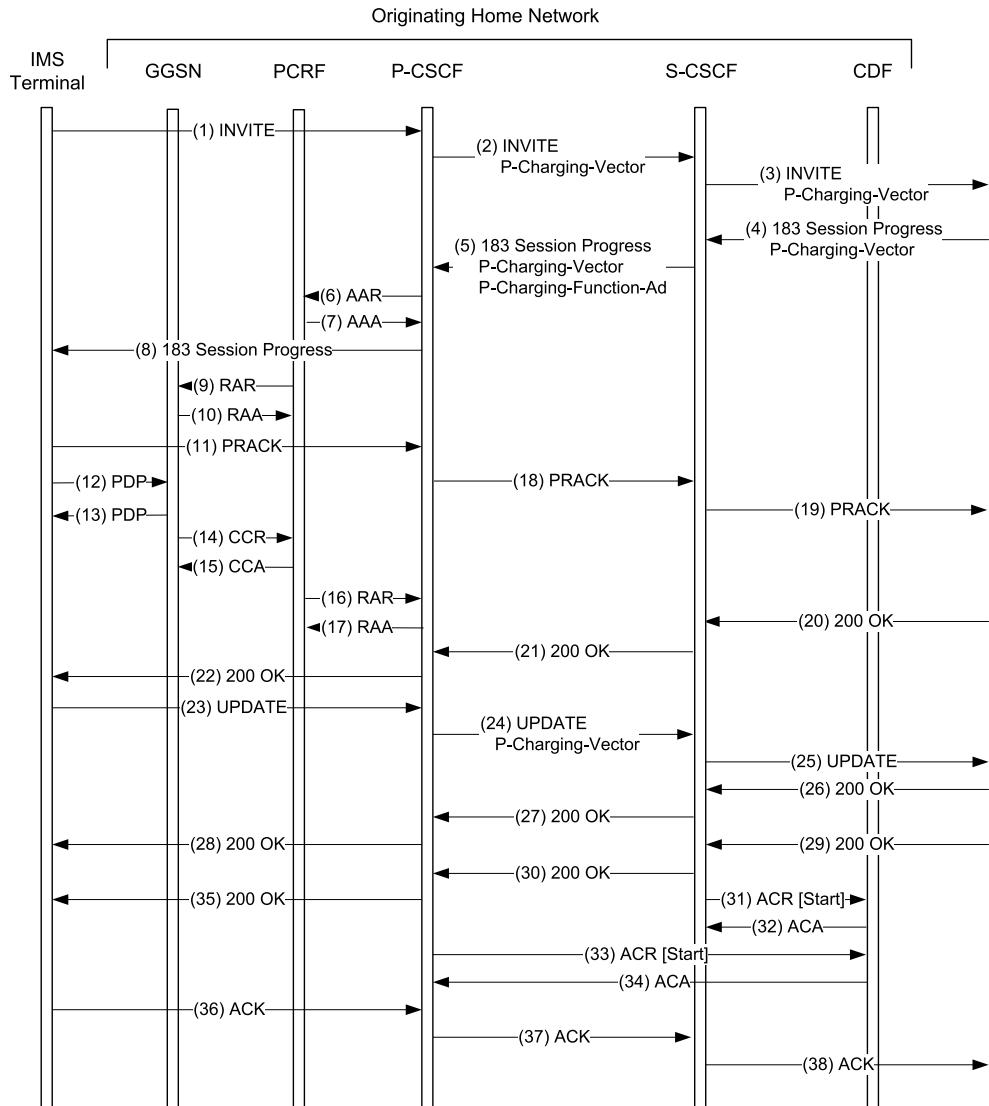
The difference between Figure 8.15 and Figure 8.10 is that the P-CSCF and the S-CSCF in the former illustration are located in the same administrative domain. This implies that they share more information than the P-CSCF and the S-CSCF in Figure 8.10, which belong to different domains. In particular, the S-CSCF provides the P-CSCF with the addresses of the CDFs (main and backup nodes) for the session in a 183 (Session Progress) response (5). In turn, the P-CSCF provides the S-CSCF with information on how the GGSN is handling the media flows in an UPDATE request (23).

The S-CSCF inserts a P-Charging-Function-Addresses header field in the 183 (Session Progress) response (5) of Figure 8.15. Figure 8.16 shows the contents of this header field: two CDF addresses (the parameter that carries the CDF addresses is called CCF because it was defined for earlier IMS releases that used CCF nodes instead of CDFs).

Basically, the S-CSCF inserts a P-Charging-Function-Addresses header field whenever the SIP request or response is addressed to a node that is located in the same network. For instance, if both caller and callee are users of the same operator and neither of them are roaming, the two P-CSCFs and two S-CSCFs involved in the session report charging events to the same CDFs, whose addresses are distributed by one of the S-CSCFs.

Once the PDP Context Activation procedure completes (13), the GGSN informs the PCRF (14) about the completion of the procedure and the parameters used on it. The PCRF, in turn, informs the P-CSCF (16) about the parameters used for the session. The P-CSCF piggybacks this information to the UPDATE request (23) sent by the terminal at a later time. The P-CSCF adds the P-Charging-Vector header field shown in Figure 8.17 to the UPDATE request (23).

The P-Charging-Vector header field in Figure 8.17 contains several parameters (these parameters are specified in 3GPP TS 24.229 [37], which extends the definition of the P-Charging-Vector header field in RFC 3455 [154]). The ggsn parameter contains the address of the GGSN handling the session. The pdp-info parameter contains a collection of information related to the PDP contexts established by the terminal. The pdp-info parameter is a quoted string that, in turn, encloses more PDP context-related parameters. The pdp-item parameter, which contains a sequential number of the PDP context information within the P-Charging-Vector header field, indicates that the following parameters pertain to a particular PDP context used in the session. There is a pdp-item parameter per PDP context established by the terminal for the session. The pdp-sig parameter indicates whether or not the PDP context is used for signaling (in our example the PDP context is not used

**Figure 8.15:** Session establishment flow

P-Charging-Function-Addresses: `ccf=[5555::b99:c88:d77:e66];
ccf=[5555::a55:b44:c33:d22]`

Figure 8.16: P-Charging-Function-Addresses value in 183 (Session Progress) (5)

```
P-Charging-Vector: icid-value="AyretU0dm+602IrT5tAFrbHLso=";
ggsn=[5555::4b4:3c3:2d2:1e1];
pdp-info="pdp-item=1; pdp-sig=no
gcid=39B26CDE;
flow-id=(\{1,1\},\{1,2\})"
```

Figure 8.17: P-Charging-Vector value in UPDATE (19)

for signaling, but for media). The `gcid` parameter contains the GPRS Charging Identifier (GCID), which is the charging identifier of the PDP context at the GGSN (the GGSN generates a different `gcid` for each new PDP context created at the GGSN).

The `flow-id` parameter indicates the flows that were sent or received over the PDP context. The `flow-id` parameter contains a collection of pairs of digits $\{x, y\}$ that refer to a particular media stream in the SDP. The “ x ” digit indicates the order of media line ($m=$) in the SDP, so that a 1 refers to the first media line ($m=$), a 2 to the second media line ($m=$), etc. The “ y ” digit indicates the sequential order of the port number of the flow within the media line. This is because a media stream typically is composed of more than a single flow and, therefore, each flow is sent to a different port number. For instance, if RTP (specified in RFC 3550 [301]) is the media transport protocol, RTP is sent to a port number and RTCP (also specified in RFC 3550 [301]) to the following port number. In our previous example, since the `flow-id` parameter contains two pairs, there are two different flows. Both are flows of the first media stream in the SDP (represented by a “1” in the “ x ” component of each tuple). The first pair refers to the RTP flow (“1” in the “ y ” component of the first tuple); the second refers to the RTCP flow of the same media stream (“2” in the “ y ” component of the first tuple).

8.3.4 The Rf Interface

As Figure 8.9 shows, the *Rf* interface (which is specified in 3GPP TS 32.299 [58]) is defined between a CTF and a CDF. The Diameter messages used over this interface are Accounting-Request (ACR) and Accounting-Answer (ACA), which are part of the Diameter base protocol (specified in RFC 3588 [96]). (See Section 6.3.5 for a description of these and other Diameter base protocol commands.) Tables 8.7 and 8.8 show the AVPs that each message can carry.

The ACR and ACA messages are used to report accounting information to the CDF. ACR messages are typically triggered by the receipt of a SIP message, such as a response to an INVITE. In Figures 8.10 and 8.15 the S-CSCF exchanges an ACR (31) and an ACA (32) message with a CDF. In the same figures the P-CSCF also exchanges an ACR (33) and an ACA (34) with the same or a different CDF. Both figures show the value of the Accounting-Record-Type AVP in the ACR message. In this case the value of this AVP is START_RECORD. The value INTERIM_RECORD is used during ongoing sessions, and the value STOP_RECORD is used when a session is terminated. The value EVENT_RECORD is used for events that are not related to the management of a session, such as the reception of a SUBSCRIBE request, as shown in Figure 8.18.

8.3.5 The Ga Interface

As Figure 8.9 shows, the *Ga* interface (which is specified in 3GPP TS 32.295 [23]) is defined between a CDF and a CGF. The CDRs sent by the CDFs to the CGF over this interface

Table 8.7: AVPs in an ACR message

Attribute name
Session-Id
Origin-Host
Origin-Realm
Destination-Realm
Accounting-Record-Type
Accounting-Record-Number
Acct-Application-Id
User-Name
Acct-Interim-Interval
Origin-State-Id
Event-Timestamp
Proxy-Info
Proxy-Host
Proxy-State
Route-Record
Service-Context-Id
Service-Information
AVP

Table 8.8: AVPs in an ACA message

Attribute name
Session-Id
Result-Code
Origin-Host
Origin-Realm
Accounting-Record-Type
Accounting-Record-Number
Acct-Application-Id
User-Name
Error-Reporting-Host
Acct-Interim-Interval
Origin-State-Id
Event-Timestamp
Proxy-Info
Proxy-Host
Proxy-State

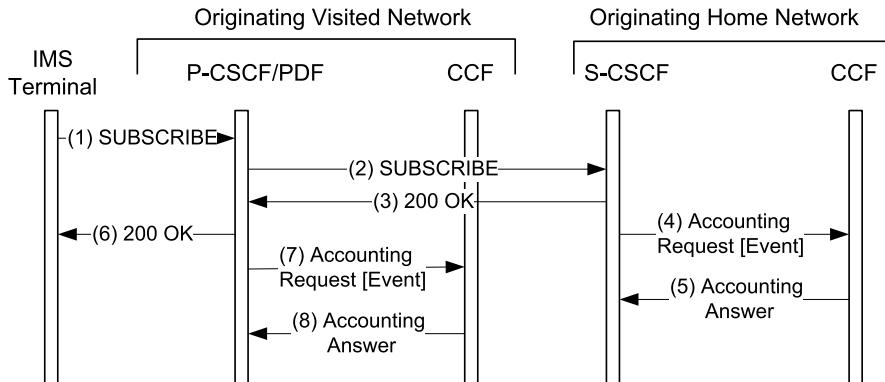


Figure 8.18: Use case for the EVENT_RECORD value in ACR messages

are transported over GTP (GPRS Protocol; GTP is specified in 3GPP TS 29.060 [34]; 3GPP TS 32.295 [23] extends GTP to specify GTP).

8.4 Online Charging Architecture

Figure 8.19 shows the IMS online, or credit-based, charging architecture. Network elements with information about resource usage report that information to the OCF (Online Charging Function) over the Diameter-based *Ro* interface or using CAP (depending on the element). The IMS GWF (IMS Gateway Function) acts as an AS and communicates with the S-CSCF over the *ISC* interface, like any other AS. The OCF consists of two modules: the SBCF (Session Based Charging Function) and the EBCF (Event Based Charging Function).

The OCF communicates with the ABMF (Account Balance Management Function) over the *Rc* interface (which has not been standardized yet) and with the RF (Rating Function) over the *Re* interface (which is specified in 3GPP TS 32.296 [24]).

8.4.1 S-CSCF

Network operators configure those sessions to which online charging has to be applied by defining a filter criterion in the user profile. The filter criterion sends all the SIP requests to the Application Server that is acting as IMS-GWF.

The IMS-GWF looks like any other AS to the S-CSCF, but it does not provide services for the user in the usual sense. Instead, the IMS-GWF reports accounting information to the OCF using the *Ro* interface. If the user runs out of credit during a session, the OCF informs the IMS-GWF (using the *Ro* interface) and the IMS-GWF terminates the session by acting as a B2BUA and sending two BYE requests, one toward each terminal.

8.4.2 Application Servers and the MRFC

The filter criterion in the user profile may get the S-CSCF to relay a SIP message to an AS or to the MRFC that will provide the service to the user. If such a service implies online charging, the AS or the MRFC uses the *Ro* interface to send charging information to the OCF. The AS or the MRFC receives the address or addresses of the OCF from the S-CSCF

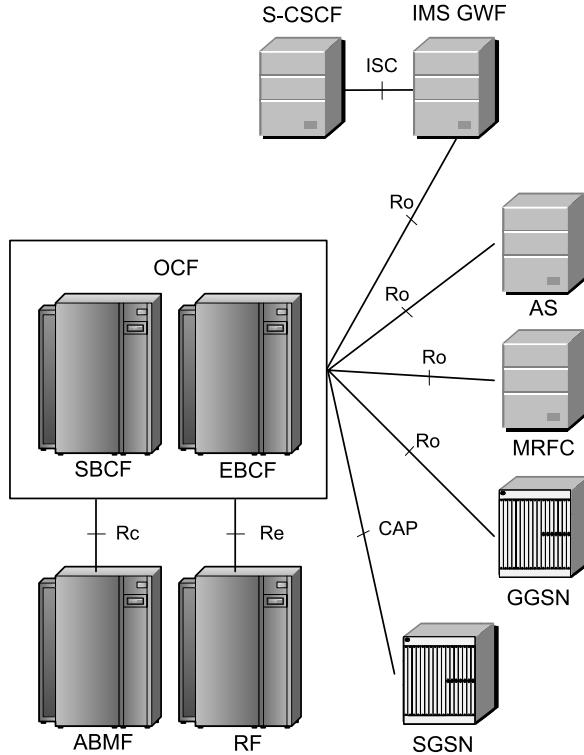


Figure 8.19: IMS online charging architecture

P-Charging-Function-Addresses: `ecf=[5555::b99:c88:d77:e66]` ;
`ecf=[5555::a55:b44:c33:d22]`

Figure 8.20: P-Charging-Function-Addresses header field

in the P-Charging-Function-Addresses header field of the SIP message, as shown in Figure 8.20 (the parameter that carries the OCF addresses is called ECF because it was defined for earlier IMS releases that used ECF nodes instead of OCFs).

8.4.3 Types of Online Charging

Online charging is based on credit units. Services are paid for by credit units, and users can enjoy a particular service as long as they have enough credit units in their accounts. For example, if Bob has two credit units in his account and ten minutes of streaming video costs one credit unit, Bob will be authorized to receive streaming video in his terminal for 20 minutes.

There are three types of online charging in the IMS: Immediate Event Charging (IEC), Event Charging with Unit Reservation (ECUR), and Session Charging with Unit Reservation (SCUR). When IEC is used, the OCF deducts a number of credit units from the user's account and then authorizes the CTF to provide the service to the user. When ECUR or SCUR are used, the OCF reserves a number of credit units in the user's account and authorizes the CTF

to provide the service to the user. If a particular service costs more credit units than those originally reserved by the OCF, the CTF can contact the OCF to request further credit unit reservations. When the service is over the CTF reports to the OCF the number of credit units that the user spent. At this point the OCF returns to the user's account all the credit units that were reserved but not used. ECUR applies to event-based charging while SCUR applies to session-based charging.

Figure 8.21 shows an example of IEC. The value of the CC-Request-Type AVP in CCR messages used for IEC is EVENT_REQUEST. The OCF receives a CCR (1) message, deducts the appropriate number of credit units from the user's account, and returns a CCA (2) message. At this point the CTF starts delivering the service requested by the user.

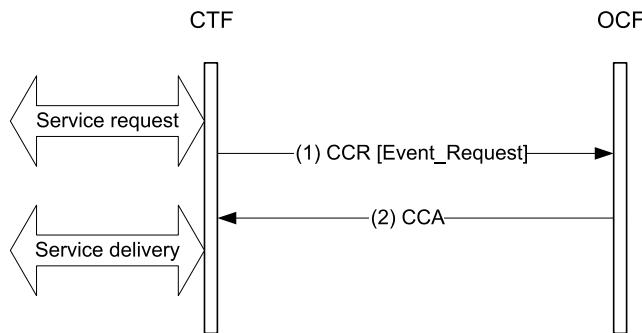


Figure 8.21: Immediate Event Charging

Figure 8.22 shows an example of ECUR. The values of the CC-Request-Type AVP are INITIAL_REQUEST or TERMINATION_REQUEST when CCR messages are used for ECUR. The OCF receives a CCR (1) message and reserves a number of credit units in the user's account. The OCF returns a CCA (2) message and the OCF start delivering the service.

The CTF sends a CCR message (3) whose CC-Request-Type AVP value is TERMINATION_REQUEST, once service delivery has ended. At this point the OCF can return to the user's account the credit units that were reserved but not used.

Figure 8.23 shows an example of SCUR. When CCR messages are used for SCUR, the values of the CC-Request-Type AVP are INITIAL_REQUEST, UPDATE_REQUEST, or TERMINATION_REQUEST. The OCF receives a CCR (1) message and reserves a number of credit units in the user's account. The OCF returns a CCA (2) message and the OCF start delivering the service.

When the user has spent most of the reserved credit units, the CTF sends a CCR message (3) whose CC-Request-Type AVP value is UPDATE_REQUEST. The OCF reserves more credit units and returns a CAA (4) message.

The CTF sends a CCR message (5) whose CC-Request-Type AVP value is TERMINATION_REQUEST, once service delivery has ended. At this point the OCF can return to the user's account the credit units that were reserved but not used.

8.4.3.1 Unit Determination

Unit determination refers to the process of determining how many credit units need to be reserved or withdrawn from the user's account for a given service. When unit determination is performed by the EFC it is referred to as *centralized unit determination*. When unit

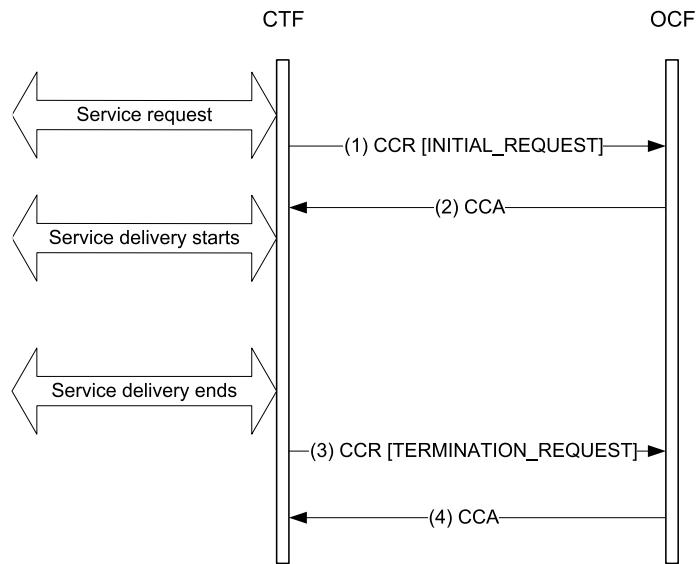
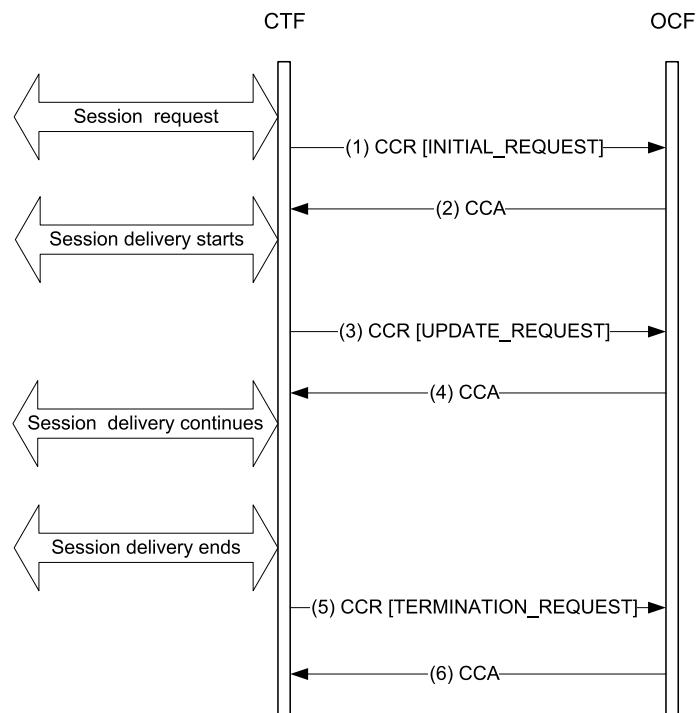
**Figure 8.22:** Event Charging with Unit Reservation**Figure 8.23:** Session Charging with Unit Reservation

Table 8.9: List of commands used by the Diameter Application for the *Ro* Interface

<i>Command-Name</i>	Abbreviation
Credit-Control-Request	CCR
Credit-Control-Answer	CCA
Re-Auth-Request	RAR
Re-Auth-Answer	RAA
Capabilities-Exchange-Request	CER
Capabilities Exchange Answer	CEA
Device-Watchdog-Request	DWR
Device-Watchdog-Answer	DWA
Disconnect-Peer-Request	DPR
Disconnect-Peer-Answer	DPA
Abort-Session-Request	ASR
Abort-Session-Answer	ASA

Table 8.10: List of commands used by the Diameter Application for the *Re* Interface

<i>Command-Name</i>	Abbreviation
PriceRequest	PRQ
PriceResponse	PRS
TariffRequest	TRQ
TariffResponse	TRS
ServiceUsageRequest	SUQ
ServiceUsageResponse	SUS

determination is performed by the MRFC or the AS it is referred to as *decentralized unit determination*.

8.4.3.2 Rating

The conversion of credit units into monetary units is referred to as rating. That is, the price of the service is calculated based on inputs fed into the rating function. If rating is performed by the OCF it is referred to as *centralized rating*, and if rating is performed by the MRFC or the AS it is referred to as *decentralized rating*.

Centralized unit determination implies centralized rating, because it is the OCF that determines the number of credit units to be charged. Neither the MRFC nor the AS has this information.

8.4.3.3 Tariff Changes

The OCF can inform the MRFC and the AS about changes in the tariff of a given service using the Tariff-Switch-Definition AVP in ACA messages. The OCF can use the same ACA message to grant some credit units to be used before the tariff change and some to be used after.

Table 8.11: AVPs specific to the Diameter Application for the *Re* Interface

Attribute name
ActualTime
AllowedUnits
BasicPrice
BasicPriceTimeStamp
BeginTime
BillingInfo
ConsumedUnits
ConsumedUnitsAfterTariffSwitch
Counter
CounterChange
CounterChangeForFirstChargeableTimeUnit
CounterChangeForFirstChargeableTimeUnitAfterSwitch
CounterChangePerChargeableVolumeUnit
CounterChangePerChargeableVolumeUnitAfterSwitch
CounterChangePerConsumedServiceUnit
CounterChangePerSession
CounterChangePerSubsequentChargeableTimeUnit
CounterChangePerSubsequentChargeableTimeUnitAfterSwitch
CounterExpiryDate
CounterID
CounterPrice
CounterTariff
CounterThreshold
CounterType
CounterValue
CounterValueBegin
CounterValueChange
CounterValueEnd
DestinationID
DestinationIDData
DestinationIDType

8.4.4 The *Ro* Interface

As Figure 8.19 shows, the *Ro* interface (which is specified in 3GPP TS 32.299 [58]) is defined between CTF and an OCS. The Diameter application for the *Ro* interface is the credit control application (specified in RFC 4006 [158]). Table 8.9 shows the Diameter commands used over the *Ro* interface.

8.4.5 The *Re* Interface

As Figure 8.19 shows, the *Re* interface (which is specified in 3GPP TS 32.296 [24]) is defined between the OCF and the RF. The Diameter Application for the *Re* Interface is a vendor-

Table 8.11: Continued

Attribute name
EParameterE1
EParameterE2
EParameterE3
EParameterE4
EParameterE5
EParameterE6
EParameterE7
ExpiryTime
Extension
FirstRequest
ImpactOnCounter
MinimalRequestedUnits
MonetaryTariff
MonetaryTariffAfterValidUnits
MonetaryQuota
Service-Rating
NextMonetaryTariff
Price
RequestedCounter
RequestedUnits
RequestSubType
Service-Identifier
ServiceInformation
SetCounterTo
Subscription-Id
Subscription-Id-Data
Subscription-Id-Type
TariffSwitchTime
ValidUnits

specific Diameter application whose vendor identifier is 10415 (this identifier identifies 3GPP) and whose application identifier is 16777218.

Table 8.10 shows the Diameter commands used over the *Re* interface. Table 8.11 shows the AVPs specific to the *Rx* application.

Chapter 9

Quality of Service on the Internet

Although the Internet has been traditionally a best-effort network, the ability to provide a certain level of QoS for certain packet flows is essential for some applications. For example, while the user of a file transfer application may accept a longer transfer delay when the network is congested, a multimedia user may find *trying* to maintain a conversation with a long round-trip delay irritating. Such users would probably request a higher QoS for their multimedia flows than for the rest of their flows.

However, QoS is not only about requesting a better treatment for certain flows; users also want to know if the network will be able to provide them with the requested QoS. If there is a long delay or a high packet loss rate, some users may prefer to exchange instant messages instead of having a VoIP (Voice over IP) conversation.

There are two models that provide QoS on the Internet: the *Integrated Services* model and the *Differentiated Services* (DiffServ) model. We cover the former in Section 9.1 and the latter in Section 9.2.

9.1 Integrated Services

The Integrated Services architecture (specified in RFC 1633 [89]) was designed to provide end-to-end QoS. Endpoints request a certain level of QoS for their packet flows and, if the network grants it, their routers treat those flows accordingly. There are two different services available in this architecture: the controlled load service and the guaranteed service.

The controlled load service ensures that packets are treated as if the network was under moderate load. Flows using this service are not affected by network congestion when this appears. Nevertheless, the network does not guarantee a certain bandwidth or a certain delay. This service can be seen as a better-than-best-effort service.

The guaranteed service guarantees a certain bandwidth or a certain delay threshold. In practice, it is not common to see this service in use because the controlled load service is often good enough and is easier to manage.

9.1.1 RSVP

The *Integrated Services* architecture uses RSVP (Resource ReSerVation Protocol, specified in RFC 2205 [90]) as the resource reservation protocol. Endpoints send RSVP messages requesting a certain QoS (e.g., a certain bandwidth) for a flow. Routers receiving these

messages obtain a description of the flow (e.g., source and destination transport addresses) as well, so that they can apply the correct treatment to all the packets that belong to it. Obviously, RSVP needs to ensure that the routers receiving resource reservation requests for a flow are the routers that will route the packets of that flow. That is, RSVP messages need to follow the same path as the packets of the flow (e.g., RTP packets carrying voice). Let us see how RSVP achieves this.

An RSVP reservation consists of a two-way handshake: a PATH message is sent and a RESV message is received, as shown in Figure 9.1. The PATH message is sent from endpoint A to endpoint B, and the network routes it as for any other IP packet. At a later time, when endpoint A sends RTP packets with voice to endpoint B, they will follow the same path as the PATH message did.

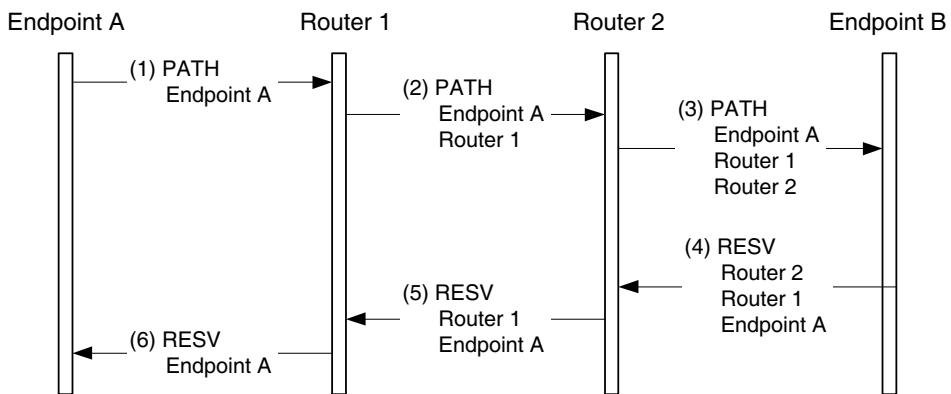


Figure 9.1: RSVP operation

PATH messages store the nodes they traverse. This allows RESV messages to be routed back to endpoint A, following the same path as the PATH message followed but in the opposite direction. In short, PATH messages leave a trail of bread crumbs so that RESV messages can find their way home. Note that SIP uses the same mechanism to route responses back to the UAC (see Section 4.6). SIP uses the *Via* header field to store the trail left by the request.

Resource reservation actually takes place when routers receive RESV messages. Therefore, resource reservation is actually performed by endpoint B; that is, the receiver of the flow (e.g., the RTP packets). PATH messages only mark the path that RESV messages have to follow.

Note, however, that the packets of the flow follow the same path as the PATH message as long as there are no routing changes in the network. If, as a consequence of a change in the network topology, packets from endpoint A to endpoint B start following a different path, a new resource reservation is needed. RSVP tackles this using soft states. Reservation soft states created by RESV messages are kept in routers only for a period of time. If they timeout before they are refreshed by a new RESV message, routers just delete them.

Endpoints periodically exchange PATH and RESV messages while the flow (e.g., the RTP packets) is active. This way, after a change in the routing logic in the network, those routers that no longer remain in the path between the endpoints do not get any new refreshes and,

consequently, remove their reservation state. New routers start receiving RESV messages that install the reservation state needed for the flow.

Note that RSVP is an admission control protocol, in addition to being a resource reservation protocol. A router can reject a resource reservation request, either because the router does not have enough resources or because the user is not allowed to reserve them.

9.1.2 State in the Network

The main problem with the integrated services architecture is that the network needs to store a lot of state information. When a packet arrives at a router, the router needs to check all the reservations it is currently handling to see whether the packet belongs to any of them. This means that routers need to store state information about every flow and need to perform lookups before routing any packet. Even though RSVP supports aggregation of reservations for multicast sessions in order to reduce the state the network needs to keep, the general feeling is that RSVP does not scale well when implemented in the core network. On the other hand, RSVP can be used to perform admission control or to connect DiffServ clouds without these scalability problems.

9.2 Differentiated Services

The DiffServ architecture (specified in RFC 2475 [87] and RFC 3260 [157]) addresses some of the problems in the integrated services architecture. DiffServ routers need to keep a minimal state, enabling them to decide which treatment a packet needs more quickly.

DiffServ routers know what treatments a packet can get. These treatments are referred to as Per-Hop Behaviors (PHBs) and are identified by 8-bit codes called Differentiated Services Codepoints (DSCPs). IP packets are marked at the edge of the network with a certain DSCP so that routers in the path apply the correct PHB to them. Two examples of standard PHBs are *expedited forwarding* (specified in RFC 3246 [117]) and *assured forwarding* (specified in RFC 2597 [165]). Packets to which the expedited forwarding PHB is applied do not see any congestion in the network. Effectively, they are treated as if they were transmitted over a TDM (Time Division Multiplexing) circuit that was exclusively reserved for them. The assured forwarding PHB provides different drop precedence levels, so that low-priority packets are discarded before high-priority ones under congestion conditions.

0	15	31
Version	Header Length	Type of Service
Identification		Flags
Time to Live		Fragment Offset
Protocol		Header Checksum
Source Address		
Destination Address		

Figure 9.2: DSCP is encoded in the *Type of Service* field in IPv4

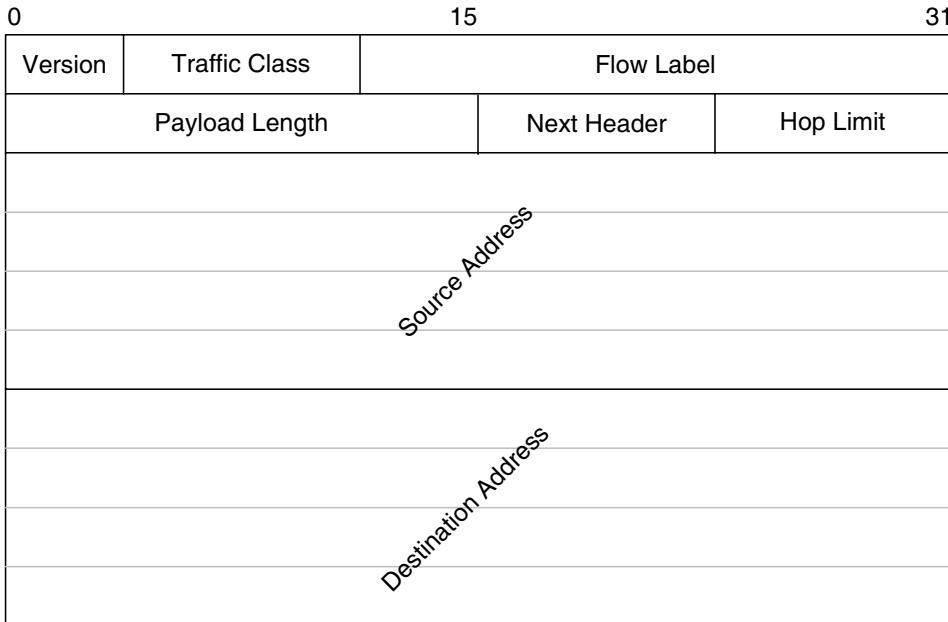


Figure 9.3: DSCP is encoded in the *Traffic Class* field in IPv6

IP packets carry their DSCPs in their IP header. The format of IPv4 and IPv6 headers is shown in Figures 9.2 and 9.3, respectively. The DSCP for a packet is placed in the *Type of Service* field in IPv4 and in the *Traffic Class* field in IPv6.

Chapter 10

Quality of Service in the IMS

The IMS supports several end-to-end QoS models (described in 3GPP TS 23.207 [13]). Terminals can use link-layer resource reservation protocols (e.g., PDP Context Activation), RSVP, or DiffServ codes directly. Networks can use DiffServ or RSVP. The most common model when cellular terminals are involved is to have terminals use link-layer protocols and to have the GGSN map link-layer resource reservation flows to DiffServ codes in the network. As mentioned in Chapter 8, the PCC (Policy and Charging Control) architecture includes QoS control. That is, PCC can be used to enforce QoS-related policy decisions such as how much bandwidth is allocated to a given session.

10.1 Policy Control and QoS

Section 8.1 describes how the PCC architecture can be used to enforce policies. The PCRF makes policy decisions based on the information received from the AF. Those decisions are enforced by the PCEF (which in a cellular network can be located at the GGSN). Policy decisions can be related to QoS and, thus, the PCC architecture is used to enforce them. Therefore, the message flows shown in this chapter are a simplified version of those in Figures 8.2, 8.3, 8.4, and 8.5.

10.2 Instructions to Perform Resource Reservations

Terminals need to be able to map the media streams of a session into resource reservation flows. A terminal that establishes an audio and a video stream may choose to request a single reservation flow for both streams or to request two reservation flows, one for video and one for audio. Requesting a reservation flow may consist of creating a secondary PDP context or sending RSVP PATH messages, for instance.

The PCC architecture supports instructing terminals on how to perform resource reservations. To do so the P-CSCF (acting as an AF) uses the SRF (Single Reservation Flow) semantics (specified in RFC 3524 [104]) of the SDP grouping framework. (Note that the entity making the decision as to how to perform resource reservations is the P-CSCF, not the PCRF as some readers could have expected given that it is a policy-related decision.)

The SDP grouping framework (specified in RFC 3388 [101]) allows us to group media streams and to describe the semantics of the group. For example, LS (lip synchronization) semantics indicate that the play-out of media streams in the group needs to be synchronized.

LS semantics are typically used to group an audio and a video stream, as shown in Figure 10.1. The `a=group` line carries the semantics of the group (LS in this case) and the identifiers of the streams (the `a=mid` line in the streams).

```
v=0
o=- 289083124 289083124 IN IP6 1080::8:800:200C:417A
t=0 0
c=IN IP6 1080::8:800:200C:417A
a=group:LS 1 2
m=audio 20000 RTP/AVP 0
a=mid:1
m=video 20002 RTP/AVP 31
a=mid:2
```

Figure 10.1: Grouping streams using LS semantics

SRF semantics indicate that all the streams in the group should use the same resource reservation flow. Consequently, the two audio streams of the session description in Figure 10.2 would use the same PDP context (assuming a GPRS access), while the video stream would use its own PDP context.

```
v=0
o=- 289083124 289083124 IN IP6 1080::8:800:200C:417A
t=0 0
c=IN IP6 1080::8:800:200C:417A
a=group:SRF 1 2
a=group:SRF 3
m=audio 20000 RTP/AVP 0
a=mid:1
m=audio 20002 RTP/AVP 0
a=mid:2
m=video 20004 RTP/AVP 31
a=mid:3
```

Figure 10.2: Grouping streams using SRF semantics

The P-CSCF adds `a=mid` and `a=group:SRF` lines to the session descriptions before relaying them to the terminals (as described in 3GPP TS 24.229 [37]). The terminals use this information to perform resource reservation. Figures 10.3 and 10.4 illustrate this point.

The P-CSCF may or may not use the mechanism we have just described in this section. It may happen, therefore, that the SDP that the IMS terminal receives does not contain any instructions to perform resource reservation. In that case the IMS terminal is free to decide how to group media streams into reservation flows.

10.2.1 Proxy Modifying Bodies

The mechanism just described assumes that the P-CSCF can both understand and modify the session descriptions exchanged by the terminals. This implies that terminals can only use SDP as their session description format and that end-to-end integrity protection or

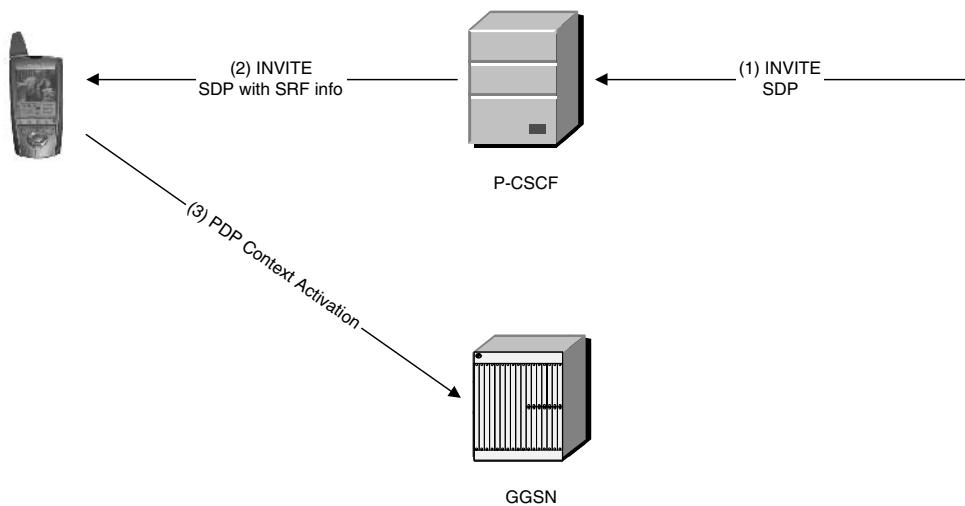


Figure 10.3: P-CSCF adds SRF info to an incoming INVITE request

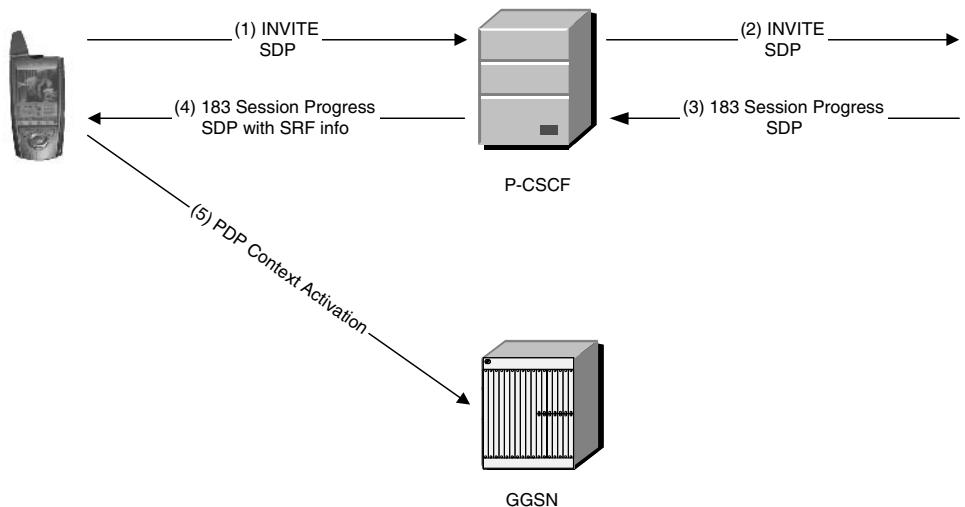


Figure 10.4: P-CSCF adds SRF info to an incoming 183 response

encryption mechanisms like S/MIME (described in Section 11.4) are not allowed in the IMS. Section 8.1.3 provides further reasons for these restrictions.

The IETF is working on extensions to allow proxy servers to communicate information, such as resource reservation instructions to user agents without accessing end-to-end bodies such as session descriptions (see the Internet-Draft “A Framework for Session Initiation Protocol (SIP) Session Policies” [166]). As mentioned in Section 8.1.3, these extensions will not be ready for some time. So, we do not expect the IMS to adopt them in the immediate future.

10.3 Reservations by the Terminals

Terminals use the SRF information received in session descriptions to determine the number of resource reservation flows that need to be established. When the access network is GPRS, a resource reservation flow is a PDP context. Let us see how the terminals establish PDP contexts (this is described in 3GPP TS 24.008 [47]).

A terminal establishes a PDP context to exchange SIP signaling right after performing a GPRS attach, as shown in Figure 10.5. The information stored by the network regarding this PDP context includes the terminal’s IP address and the PDP context’s QoS characteristics, including its traffic class. There exist four traffic classes: best effort, interactive, streaming, and conversational. The PDP contexts used for SIP signaling are always conversational.

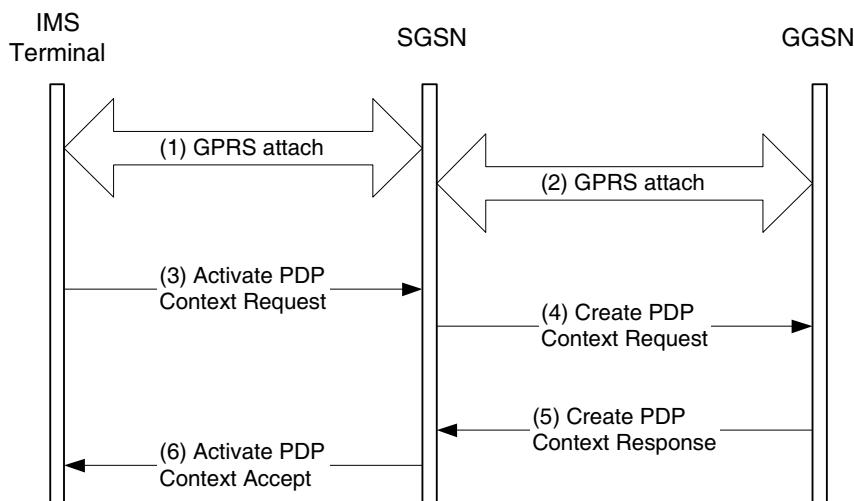


Figure 10.5: PDP context activation

IMS terminals establish additional PDP contexts to send and receive media, as shown in Figure 10.6. The number of additional PDP contexts, referred to as *secondary PDP contexts*, depends on the instructions received from the P-CSCF in the form of `a=group:SRF` lines. Secondary PDP contexts use the same IP address as the primary PDP context, but may have different QoS characteristics.

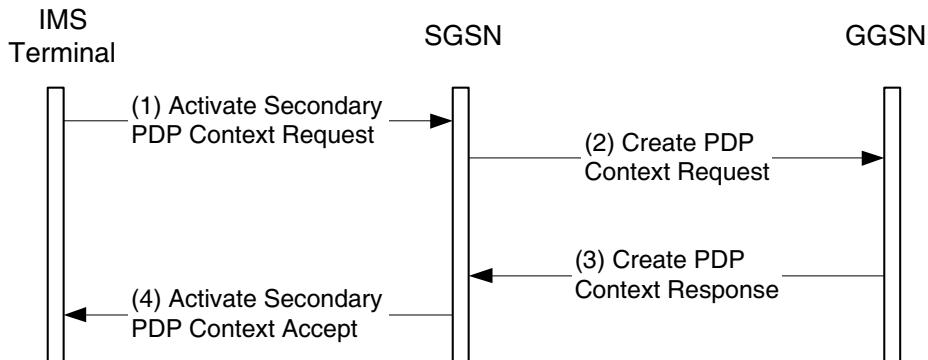


Figure 10.6: Secondary PDP context activation

10.4 QoS in the Network

The GGSN receives traffic from a given terminal over a PDP context, assigns it an appropriate DSCP (Differentiated Services Codepoint), and sends it out into a DiffServ-enabled network, as shown in Figure 10.7. In short, the GGSN implements the DiffServ edge function.

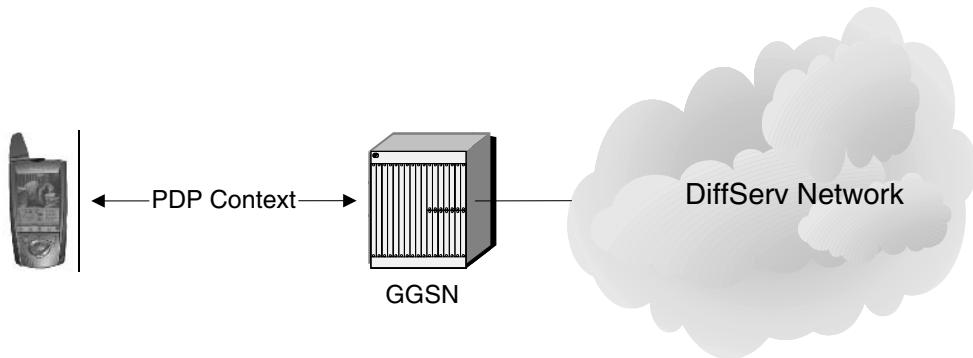


Figure 10.7: Mapping PDP context information to DSCPs by the GGSN

The DSCP that corresponds to a particular PDP context is typically assigned based on statically configured rules in the GGSN. Still, when RSVP is used, the GGSN may use the information carried in RSVP signaling to decide which DSCP to use.

Chapter 11

Security on the Internet

According to the traditional definition, network security comprises integrity, confidentiality, and availability. Message integrity ensures that if an unauthorized party modifies a message between the sender and the receiver, the receiver is able to detect this modification. In addition to message integrity, integrity mechanisms always provide some type of proof of data origin. Knowing that a message has not been modified without knowing who initially created the message would be useless.

Confidentiality mechanisms keep unauthorized parties from gaining access to the contents of a message. Confidentiality is typically achieved through encryption.

Denial of Service (DoS) attacks compromise the system's availability by keeping authorized users from accessing a particular service. The most common DoS attacks consist of keeping the servers busy performing an operation or sending the servers more traffic than they can handle.

SIP provides several security mechanisms to address integrity, confidentiality, and availability. Some of the security mechanisms come from the world of the web, some come from the world of email, and some of them are SIP-specific. We analyze these mechanisms in the following sections and describe how they relate to the three security properties just described.

11.1 HTTP Digest Access Authentication

The first problem a SIP server faces is authenticating users who are requesting services. SIP has inherited an authentication mechanism from HTTP called *HTTP Digest Access Authentication* (specified in RFC 2617 [145]). In the SIP context the server authenticating the user (i.e., the caller) can be a proxy, a registrar, a redirect server, or a user agent (the callee's user agent). The `WWW-Authenticate` and `Authorization` header fields are used with registrars, redirect servers, and user agents, and the `Proxy-Authenticate` and `Proxy-Authorization` header fields are used with proxies.

When using HTTP Digest Access Authentication the client and the server have a shared secret (e.g., a password), which is exchanged using an out-of-band mechanism. When a server at a given domain receives a request from a client the server challenges the client to provide valid credentials for that domain. At that point the client provides the server with a username and proves that the client knows the shared secret.

An obvious way for the client to prove that it knows the shared secret would be to send it to the server in clear text (i.e., without any encryption). In fact, this is what HTTP basic access authentication (also specified in RFC 2617 [145]) does. Nevertheless, the security risks created by sending passwords in clear text are obvious. Any attacker that manages to gain access to the message carrying the shared secret gains access to the shared secret itself. Previous SIP specifications allowed the use of basic authentication, but it has now been deprecated for some time. The use of HTTP Digest Access Authentication is currently recommended instead.

Clients using digest can prove that they know the shared secret without sending it over the network. Digest uses hashes and nonces for this purpose. A hash algorithm is a one-way function that takes an argument of an arbitrary length and produces a fixed length result, as shown in Figure 11.1. The fact that hash algorithms are one-way functions means that it is computationally infeasible to obtain the original argument from the result. Two popular hash algorithms are MD5 (specified in RFC 1321 [263]) and SHA1 (specified in RFC 3174 [127]). A nonce is a random value that is used only once.

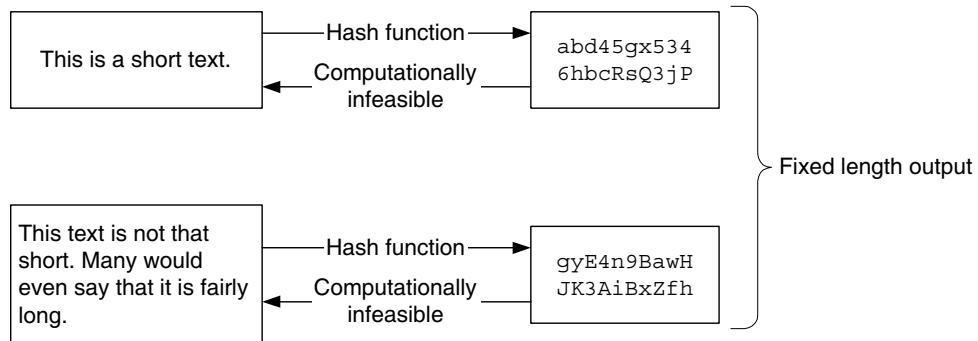


Figure 11.1: Hash function

Figure 11.2 shows how digest uses hashes and nonces. Alice sends an INVITE (1) request addressed to Bob through her outbound proxy (at domain.com). The proxy challenges the INVITE with a 407 (Proxy Authentication Required) response (2). The proxy includes a `Proxy-Authenticate` header field with a set of parameters. The `realm` parameter indicates the domain of the proxy server, so that the client knows which password to use. The `qop` (quality of protection) parameter indicates that the server supports integrity protection for either the request line alone (`auth`) or for both the request line and the message body (`auth-int`). The server provides the client with a random nonce in the `nonce` parameter. The `algorithm` parameter identifies the hash function (MD5, in this example).

When the client receives the response it issues a new INVITE (3) with a `Proxy-Authorization` header field. The `Proxy-Authorization` header field contains a set of parameters. The `response` parameter is especially interesting. It contains a hash comprising, among other things, the username, the password, the server's nonce, the client's nonce (`cnonce` parameter), and the request line. When the `auth-int` `qop` is chosen the message body is also fed into the hash algorithm to generate the `response` parameter.

When the server receives this `Proxy-Authorization` header field it calculates another hash value using the same input as the client, but using the shared secret the server has. If the result matches the value in the `response` parameter of the INVITE request the server

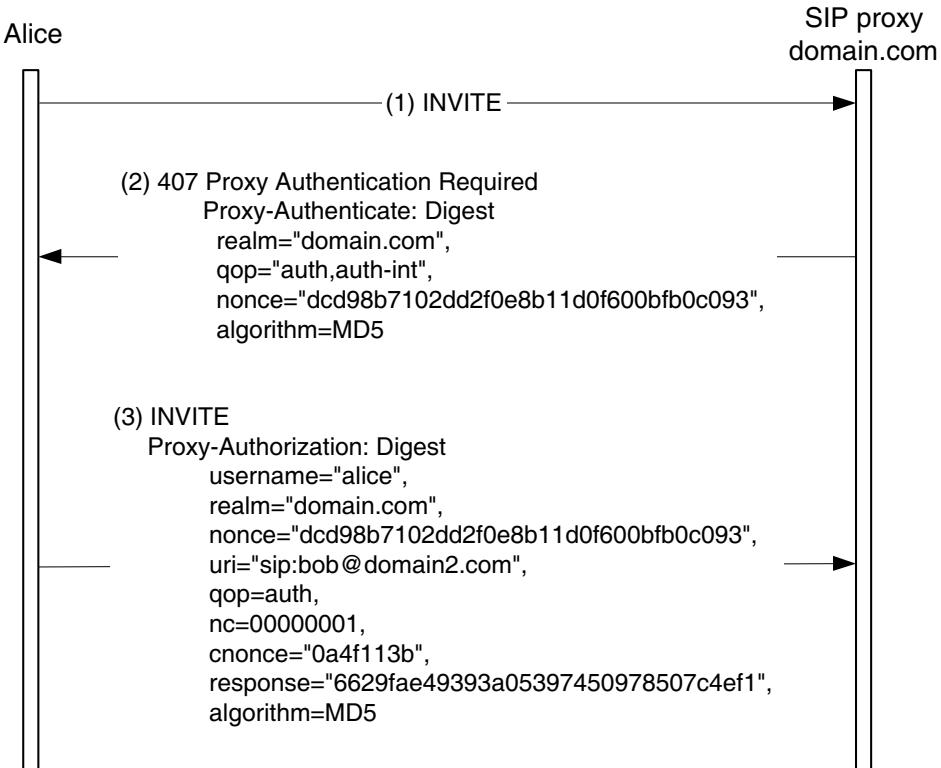


Figure 11.2: HTTP Digest Access Authentication operation

considers the authentication successful and keeps on processing the INVITE. Otherwise, the server will challenge this INVITE again.

The inclusion of random nonces chosen by the server in the hash prevents replay attacks. Even if an eavesdropper manages to obtain the correct hash value for a particular nonce it will not be able to access the server, since the server will challenge it with a different random nonce.

11.1.1 Security Properties of Digest

HTTP Digest Access Authentication provides authentication of users and a limited degree of integrity protection (no confidentiality). Digest integrity protects the request line (i.e., method and *Request-URI*) and, potentially, the message body. Still, it does not integrity-protect the header fields of the message. This lack of header field protection is an important drawback, because, as we saw in previous chapters, header fields in SIP carry important information which attackers can easily manipulate if digest is used by itself. In brief, digest access authentication is vulnerable to Man-in-the-Middle (MitM) attacks.

Regarding availability, digest offers good DoS protection. Servers issuing challenges using digest can remain stateless until the new request arrives. So, an attacker issuing a large number of requests to a server will not block any resources for a long period of time.

In addition to client authentication, digest can also provide server authentication. That is, the server can prove to the client that the server knows their shared secret as well. Nevertheless, this feature is not used by SIP proxies.

Digest client authentication is common because user agents acting as clients do not typically have certificates to use more advanced and secure certificate-based authentication mechanisms. On the other hand, domain administrators typically provide their proxy servers with site certificates, which is why certificate-based proxy authentication is used instead of digest server authentication.

11.2 Certificates

Certificates are key to understanding TLS and S/MIME (which are described in the following sections). A certificate is a statement about the truth of something signed by a trusted party. A passport is an example of a certificate. The government of a country certifies the identity of the passport holder. Everyone who trusts the government will be convinced of the identity of the holder.

Certificates used in SIP bundle together a public key with a user (e.g., `sip:Alice.Smith@domain.com`), a domain (e.g., `domain.com`), or a host (e.g., `ws1234.domain.com`). Domain certificates, also known as site certificates, are used by network elements (i.e., proxies, redirect servers, and registrars) and are usually signed by a certification authority whose public key is well known. This way, any SIP entity that knows the public key of the certification authority can check the validity of the certificates presented by another SIP entity. Within a domain, host certificates can be used to authenticate different network nodes (e.g., two proxies).

Since building and managing a certification authority hierarchy to provide signed certificates to every user has been proven to be a difficult task for a number of reasons (technical and nontechnical), SIP allows the use of self-signed certificates: that is, certificates that are not signed by any certification authority. Section 11.4 describes how self-signed certificates are used in SIP.

11.3 TLS

TLS (Transport Layer Security, specified in RFC 2246 [120]) is based on SSL (Secure Sockets Layer). SSL was designed to protect web communications and is currently implemented in most Internet browsers. TLS is more generic than SSL and can be used to protect any type of reliable connection. So, using TLS to protect SIP traffic is yet another security solution that comes from the world of the web.

TLS provides data integrity and confidentiality for reliable connections (e.g., a TCP connection). It can also provide compression, although that feature is not typically used. TLS consists of two layers: namely, the TLS handshake layer and the TLS record layer.

The TLS handshake layer handles the authentication of peers, using public keys and certificates, and the negotiation of the algorithm and the keys to encrypt the actual data transmission. Once a reliable connection at the transport layer is established (e.g., a TCP connection) a TLS handshake takes place. The handshake starts with an exchange of `hello` messages and usually takes two RTTs (Round Trip Times) to complete. During authentication

the server provides the client with the server's certificates (e.g., X.509 certificates [187]) and can optionally request the client to provide its certificates as well. The server chooses the encryption algorithm to be used from among those supported by the client.

The TLS handshake also allows peers to generate a so-called premaster secret (e.g., using Diffie–Hellman) and to exchange random values. The premaster secret and the random values are used by the TLS record layer to generate the encryption key.

The TLS record layer handles the encryption of the data. It typically uses a symmetric encryption algorithm whose key is generated from the values provided by the handshake layer.

As we can see in Figure 11.3, when TLS is used over TCP peers need to wait for three RTTs (one for the TCP handshake and two for the TLS handshake) before exchanging secure data. Note that in Figure 11.3 we assume that the TLS Client Hello (3) is piggybacked on the TCP ACK that completes the TCP three-way handshake, although many TCP implementations send them in parallel rather than in a single datagram.

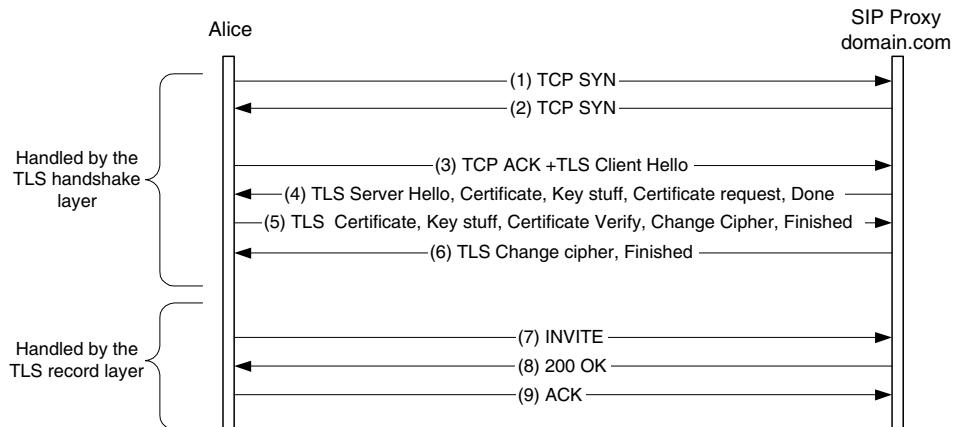


Figure 11.3: TLS connection establishment

11.3.1 SIP Usage

From the SIP point of view, TLS is a hop-by-hop security solution. That is, a TLS connection protects the messages exchanged between a SIP entity and the SIP entity one hop away (e.g., a user agent and a proxy). For example, if Alice's outbound proxy needs to send a SIP message to Bob through his proxy server at domain2.com, Alice's proxy will perform a DNS lookup for domain2.com. If the DNS records for domain2.com show that the proxy at domain2.com prefers to use TLS, they will exchange SIP traffic using a TLS connection; however, establishing this TLS connection is out of Alice's control. When Alice sent the original message to her outbound proxy she did not know anything about the domain2.com DNS records. If these DNS records had said that UDP with no security was preferred over TLS, the proxies would not have used any kind of encryption. So, Alice needs a way to tell her proxy that she wants her message to be protected all of the time. That is, she wants all of the proxies in the path to exchange data using TLS while handling her request. SIP provides

a mechanism for user agents to request this type of treatment for a request: the SIPS URI scheme.

When a user agent sends a request to a SIPS URI it is requesting that every proxy in the path uses TLS to relay the request to the next proxy. Still, using SIPS URIs does not imply that the last proxy will use TLS to relay the request to the destination user agent. The last proxy is free to use whatever security mechanism it has agreed on with the user agent (e.g., a manually configured IPsec association). This is because users of a particular domain can agree on the security mechanisms to use within that domain fairly easily. On the other hand, doing so for inter-domain communications would be difficult to manage. It is better to use a default solution for this case (i.e., TLS). Of course, the use of SIPS URIs also implies that proxies route the responses to the request using TLS as well.

11.3.1.1 Client Authentication

As we said earlier, TLS supports certificate-based client and server authentication. This type of authentication is used when a proxy acts as a client sending a request to another proxy that acts as a server. In any case, while proxies typically have site certificates, at present SIP User Agents do not usually have certificates. One way to provide client and server authentication without providing user agents with certificates is to use digest access authentication over TLS. The server is authenticated using certificates at the TLS handshake layer, and the user agent (the client) is authenticated using digest once the TLS connection has been established.

11.4 S/MIME

SIPS URIs ensure that messages are secured hop by hop. Still, every SIP proxy handling a message with a SIPS URI has access to *all* of the contents of that message. Of course, proxies need to have access to some header fields (i.e., `Route`, `Request-URI`, `Via`, etc.), but user agents may want to exchange some information they do not want proxies to have access to. For example, Alice and Bob may not want the proxies to be aware of which type of session (e.g., audio or video) they are establishing, or they may want to exchange cryptographic keys to exchange media traffic in a secure way. The fact that Alice and Bob trust their proxies to route messages does not mean that they want the proxies' administrators to be able to listen to their conversation. Some type of end-to-end security is needed. SIP uses an end-to-end security solution from the email world: S/MIME.

Section 4.7 described how SIP uses the Multipurpose Internet Mail Extensions (MIME) format specified in RFC 2045 [146] to encode multipart SIP bodies. S/MIME (Secure/MIME) allows us, among other things, to sign and encrypt message parts and encode them using MIME. These message parts are usually message bodies (e.g., a session description), although sometimes header fields carrying sensitive information are protected as well.

The CMS (Cryptographic Message Syntax, specified in RFC 3369 [168]) is a binary format to encode, among other things, signed messages, encrypted messages, and the certificates related to those messages. (RFC 2633 [259] describes how to encode CMS binary objects using MIME.)

Figure 11.4 shows an example of an encrypted body using S/MIME. Message bodies are encrypted using a symmetric content encryption key, which is encrypted using the public key of the recipient (symmetric encryption is much faster than public key-based encryption), as shown in Figure 11.5. The recipient decrypts the content encryption key using their private key and then decrypts the message with the content encryption key, as shown in Figure 11.6.

```

INVITE sip:Alice.Smith@domain.com SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576sl
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:bob@192.0.100.2>
Content-Length: 197
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
    name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m;
    handling=required

rfvbnj756tbBghyHhHUujhJhjH77n8HHGT9HG4VQpfyF467GhIGfHfYT6
7n8HHGghyHhHUujhJh4VQpfyF467GhIGfHfYGTrvbnjT6jH7756tbB9H
f8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
OGhIGfHfQbnj756YT64V

```

Figure 11.4: Message body encrypted using S/MIME

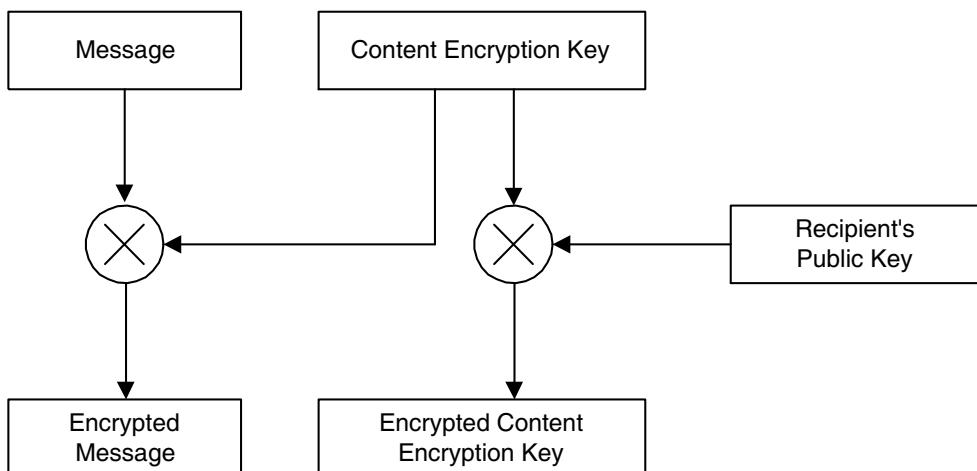


Figure 11.5: Encrypting a body using S/MIME

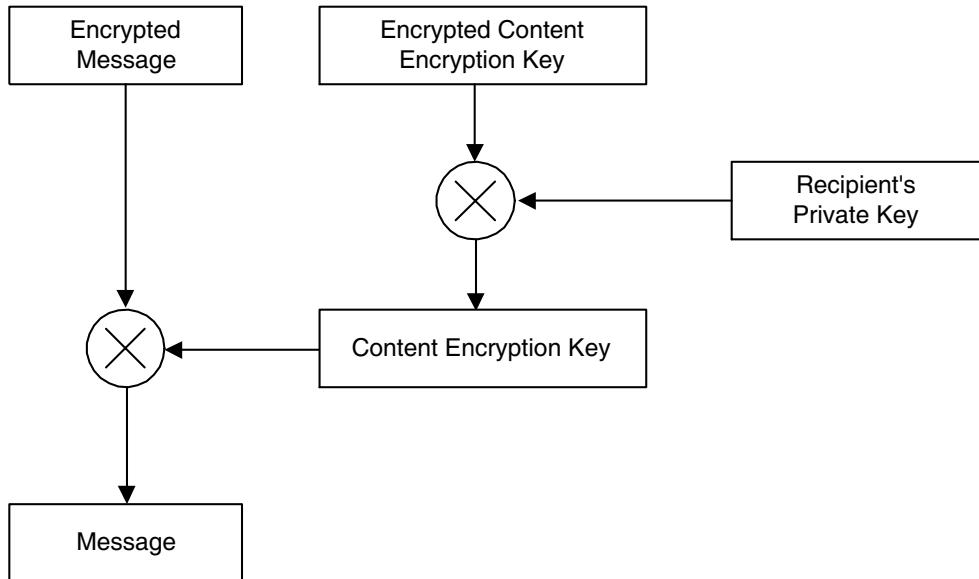


Figure 11.6: Decrypting a body using S/MIME

Figure 11.7 shows an example of a signed body using S/MIME. Message bodies are signed with the sender’s private key. Signed message bodies usually carry the sender’s certificate. In this way the receiver obtains the sender’s public key, which should be used to check the validity of the signature.

If a message body needs to be signed *and* encrypted the message bodies of Figures 11.4 and 11.7 can be nested. The encrypted message body can be signed or the signed message body can be encrypted.

The format in Figure 11.7 works well when the recipient of the message supports S/MIME. Nevertheless, at present many user agents do not support it. Such a user agent would not be able to understand Figure 11.7, since it uses the CMS binary format. There is an alternative way to sign messages that makes them understandable for S/MIME-unaware user agents (described in RFC 1847 [147]). Figure 11.8 shows an example of this format. The message body (an SDP session description) is encoded as an independent MIME part, and the signature is encoded using the S/MIME binary format. This type of signature is referred to as a CMS-detached signature, because it is not encoded together with the message body that has been signed. Even if S/MIME-unaware implementations were able to parse the session description, they of course would not be able to verify the signature. In that case the integrity of the message cannot be verified.

11.4.1 Self-signed Certificates

As explained in Section 11.2, in the absence of a certification authority S/MIME bodies can carry self-signed certificates that bundle a public key with a user. However, at first sight this seems to provide no security at all. An attacker can sign or encrypt a message with her private key and send it along with a certificate saying that her public key belongs to

```

INVITE sip: Alice.Smith@domain.com SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:bob@192.0.100.2>
Content-Length: 197
Content-Type: application/pkcs7-mime; smime-type=signed-data;
    name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m;
    handling=required

567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjhH
HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jh7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75

```

Figure 11.7: Message body signed using S/MIME

`sip: Alice.Smith@domain.com`. The recipient would be fooled by this certificate, since it does not have any means to check the validity of this certificate.

Self-signed certificates need a so-called *leap of faith*. That is, the first time a user exchanges SIP traffic with a second user the first user assumes that they are not under attack. When both users exchange traffic between them at a later time they assume that the keys used the first time are the valid ones. This mechanism is also used by other protocols such as SSH (specified in the Internet-Draft “SSH Protocol Architecture” [318]).

In many scenarios the leap of faith provides good security properties. Two users can exchange their self-signed certificates using a cable between their user agents, or they can establish an audio session to read aloud their public keys to each other and check whether they are the same as those received in the self-signed certificate. There are many ways to obtain another user’s public key in a secure enough manner. Users just need to choose the one they prefer in each situation.

11.5 Authenticated Identity Body

We have just seen how to protect message bodies using S/MIME. In addition to message bodies, S/MIME can be used to protect header fields as well. The idea is to place a copy of the header fields to be protected in a body and to protect it using S/MIME. The Authenticated Identity Body (AIB, specified in RFC 3893 [249]) is a good example of how to use S/MIME to integrity-protect header fields carrying identity information such as `From`, `To`, and `Contact`.

The header fields that the user wants to protect are placed in a body whose `Content-Type` is `message/sipfrag`. This type of body carries fragments of SIP messages as opposed to whole SIP messages (which are carried in `message/sip` bodies).

```

INVITE sip:Alice.Smith@domain.com SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:bob@192.0.100.2>
Content-Type: multipart/signed;
    protocol="application/pkcs7-signature";
    micalg=sha1; boundary="34573255067boundary"
Content-Length: 709

--34573255067boundary
Content-Type: application/sdp
Content-Disposition: session

v=0
o=Alice 2790844676 2867892807 IN IP4 192.0.0.1
s=Let's talk about swimming techniques
c=IN IP4 192.0.0.1
t=0 0
m=audio 20000 RTP/AVP 0
a=sendrecv
m=video 20002 RTP/AVP 31
a=sendrecv

--34573255067boundary
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
    handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jh77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
--34573255067boundary--

```

Figure 11.8: Message body signed using CMS-detached signatures

Figure 11.9 shows an INVITE request whose `From` and `Contact` header fields are integrity-protected. The INVITE carries three body parts: the usual SDP session description, a `message/sipfrag` body part with the header fields that we want to protect, and a digital signature over the previous body part.

The `message/sipfrag` body part carries the `Call-ID`, the `Cseq`, and the `Date` header fields to prevent cut-and-paste attacks. Otherwise, an attacker could cut the AIB with its signature and paste it into a different SIP message. The recipient of such a message would believe that it comes from Bob.

Proxies can also add AIB to requests and responses (as described in RFC 4474 [251]). When a proxy inserts (and signs) an AIB body, in addition to providing integrity protection for some header fields it is asserting that the information in those header fields is true. For example, Bob's outbound proxy uses digest to authenticate Bob and then inserts the AIB in Figure 11.9, confirming that Bob generated the request. Other proxies that trust Bob's outbound proxy but do not share any credentials with Bob will know that this request was generated by a user that is known by at least one proxy in their circle of trust. That is usually enough for a proxy to agree to route the request further.

11.6 IPsec

IPsec (whose architecture is specified in RFC 2401 [200]) provides confidentiality and integrity protection at the network layer. Nodes that want to exchange secure IPsec-protected traffic between them set up a so-called *security association*. A security association is identified by the addresses of the nodes and by its SPI (Security Parameters Index), and it contains the security parameters (e.g., keys and algorithms) that the nodes use to protect their traffic. IKE (Internet Key Exchange, specified in RFC 2409 [164]) is the key management protocol that is typically used to set up security associations.

11.6.1 ESP and AH

IPsec provides two protocols to protect data, namely ESP (Encapsulating Security Payload) and AH (Authentication Header). ESP provides integrity and (optionally) confidentiality while AH provides integrity only. The difference between the integrity provided by ESP and AH is that ESP only protects the contents of the IP packet (excluding the IP header) while AH protects the IP header as well. Figures 11.10 and 11.11 illustrate this difference. From now on, we focus on ESP because it is more widespread than AH and is the protocol used in the IMS.

ESP adds a header and a trailer to each IP packet, as shown in Figure 11.10 (some parts of the ESP trailer are encrypted and integrity-protected, while other parts are not; Figure 11.12 clarifies this point further). The ESP header contains the SPI, the sequence number of the packet, and the initialization vector for the encryption algorithm. The ESP trailer contains optional padding in case it is required by the encryption algorithm and data related to authentication (i.e., integrity protection) of the data. Figure 11.12 shows the format of an ESP-protected packet in more detail.

11.6.2 Tunnel and Transport Modes

ESP has two modes of operation: transport mode and tunnel mode. Transport mode is normally used between endpoints, while tunnel mode is typically used between security gateways to create virtual private networks.

```

INVITE sip:Alice.Smith@domain.com SIP/2.0
Via: SIP/2.0/UDP ws1.domain2.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Bob <sip:Bob.Brown@domain2.com>;tag=9hx34576s1
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Date: Thu, 9 Oct 2003 10:21:03 GMT
Contact: <sip:bob@192.0.100.2>
Content-Type: multipart/mixed; boundary=unique-boundary-1
Content-Length: 1066

--unique-boundary-1
Content-Type: application/sdp

v=0
o=bob 2890844526 2890844526 IN IP4 ws1.domain2.com
s=-
c=IN IP4 192.0.100.2
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
--unique-boundary-1
Content-Type: multipart/signed; boundary=boundary2
    protocol="application/pkcs7-signature"; micalg=sha1;
Content-Length: 696

--boundary2
Content-Type: message/sipfrag
Content-Disposition: aib; handling=optional

From: Bob <sip:Bob.Brown@domain2.com>
To: Alice <sip:Alice.Smith@domain.com>
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:bob@192.0.100.2>
Date: Thu, 9 Oct 2003 10:21:03 GMT
--boundary2
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
    handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
--boundary2--
--unique-boundary-1--

```

Figure 11.9: INVITE carrying an AIB body

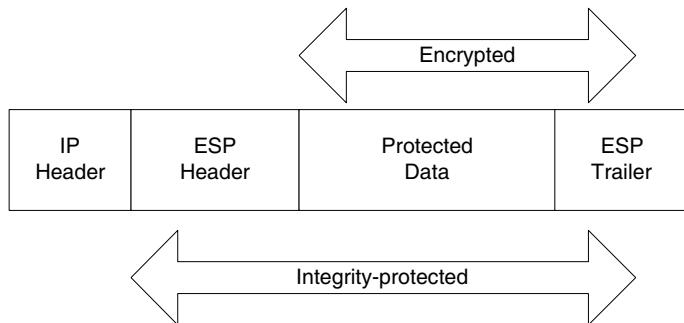


Figure 11.10: IP packet protected by ESP

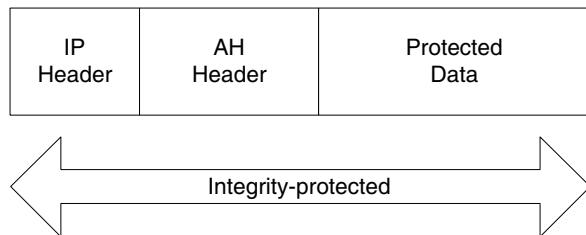


Figure 11.11: IP packet protected by AH

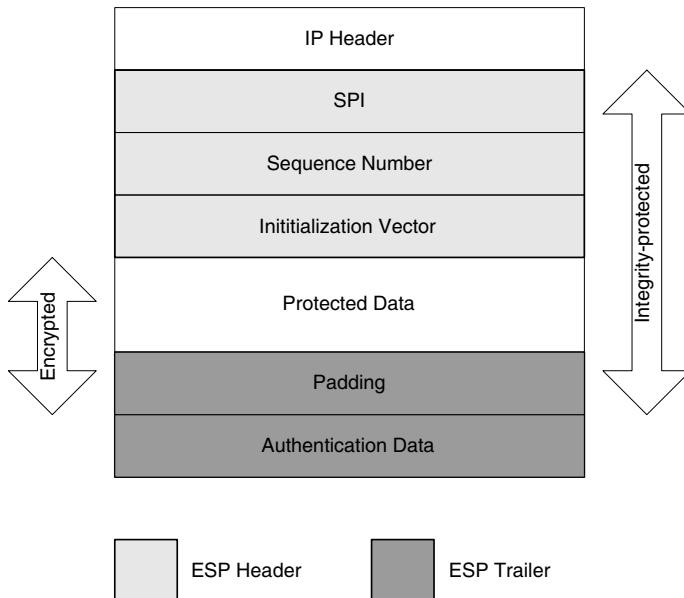


Figure 11.12: ESP header and trailer format

ESP in transport mode protects the payload of an IP packet. For example, two entities exchanging TCP traffic using ESP transport mode would protect the TCP headers and the actual contents carried by TCP, as shown in Figure 11.13.

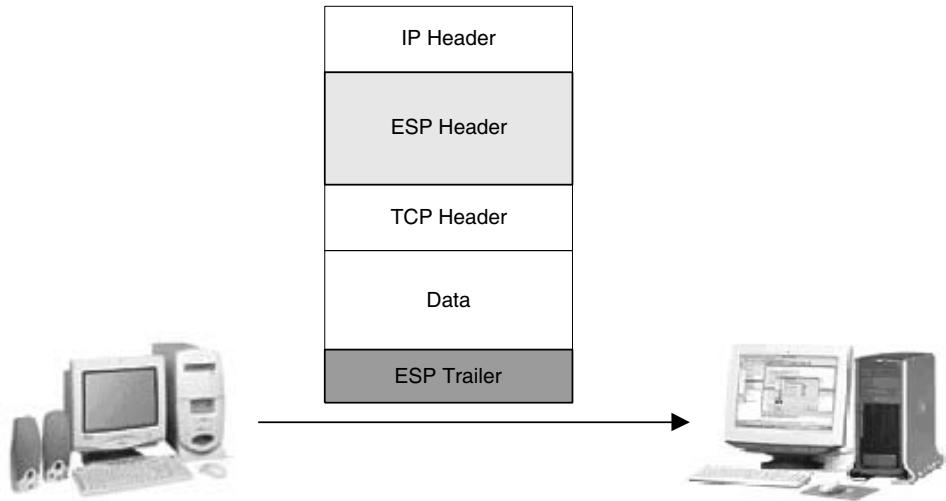


Figure 11.13: IPsec ESP in transport mode

ESP in tunnel mode protects an entire IP packet by encapsulating it into another IP packet. The outer IP packet carries the IP addresses of the security gateways while the inner IP packet remains untouched. Figure 11.14 shows two security gateways exchanging ESP tunnel-mode traffic between them. Note that, in this example, the traffic between the endpoints and the security gateway is not protected.

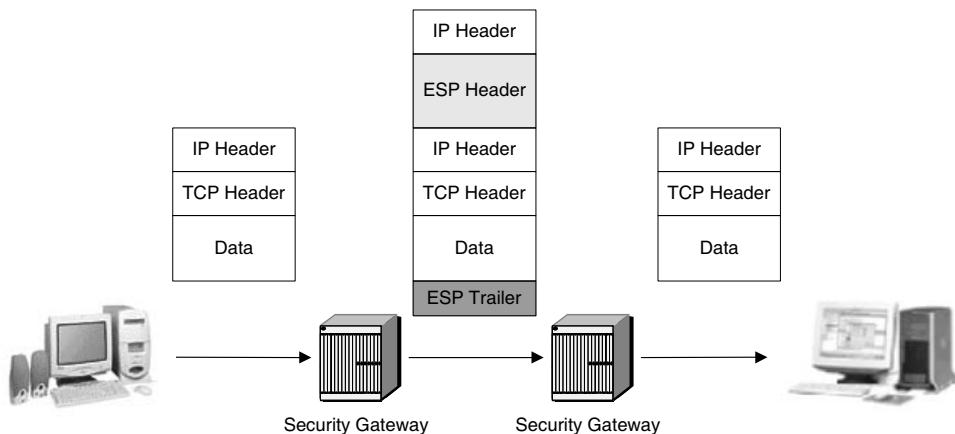


Figure 11.14: IPsec ESP in tunnel mode

11.6.3 Internet Key Exchange

IKE is the key management protocol that is typically used to set up IPsec security associations. It is based on the ISAKMP (Internet Security Association and Key Management Protocol, specified in RFC 2408 [215]) framework, which defines how to exchange information with a peer in a secure way to perform key exchanges. IKE defines several such key exchanges, and the IPsec DOI (Domain of Interpretation, specified in RFC 2407 [252]) document defines which attributes need to be negotiated in an IPsec security association.

The establishment of an IPsec security association consists of two steps. In the first step, peers establish a security association to run IKE on it. In the second step, peers use this security association to securely agree on the parameters of the IPsec security association that will carry the actual data traffic.

11.6.3.1 IKE Security Association Establishment

There are two modes to establish the IKE security association: the *main mode* and the *aggressive mode*. The main mode consists of six messages (i.e., three RTTs) and allows peers to hide their identities from potential attackers. The aggressive mode only uses three messages, does not provide identity protection, and offers less negotiation power.

11.6.3.2 IPsec Security Association Establishment

IPsec security associations are established using the so-called *quick mode*. The quick mode uses an IKE security association and consists of three messages. After this exchange, peers are ready to exchange traffic using the IPsec security association they just established.

11.7 Privacy

Users sometimes need to establish a session without disclosing personal information, such as their identity. For example, Bob wants to remain anonymous when he calls a television program to give his opinion about a politician. To do this, Bob could send an INVITE with a From header field set to “anonymous” instead of “Bob”. Nevertheless, proxies do not typically allow anonymous users to use their routing services. Bob needs to identify himself to his outbound proxy, but wants to remain anonymous to the rest of the network.

SIP User Agents can request different types of privacy from a proxy by using a Privacy header field (as described in RFC 3323 [247]). The proxy typically authenticates the user and then encrypts, removes, or simply changes the contents of the header fields with personal information about the user. The proxy can also insert an AIB (as described in Section 11.5), confirming that it has authenticated the user who wishes to remain anonymous. Proxies in closed environments can insert a P-Asserted-Identity header field (specified in RFC 3325 [194]) instead of an AIB.

For example, Bob can use the following From header field in his INVITE request and then use Digest to prove to his proxy that it was him who generated the INVITE request:

```
From: "Anonymous" <sip:anonymous@anonymous.invalid>
;tag=9hx34576sl
```

11.8 Encrypting Media Streams

In addition to securing SIP messages, users are often interested in securing the media that they exchange. For example, they may want to have confidential conversations or to be sure that no one is modifying the contents of their instant messages.

In a single session there can be several media streams with different security requirements. One media stream may have to be encrypted while another may only need to be integrity-protected. Therefore, the natural place to introduce media security is in the session description, rather than in the SIP header fields (which would apply to the whole session).

The key management extensions for SDP (specified in RFC 4567 [78]) are encoded as SDP attributes (`a=` lines). An additional attribute is added to the description of a media stream that carries the name of the key management protocol to be used and the data needed by that protocol. It is possible, although not common, to offer multiple key management protocols to the answerer, who will choose one. Figure 11.15 shows an example of a media stream that uses the MIKEY key management protocol. Note that the transport protocol RTP/SAVP (Real-Time Transport Protocol/Secure Audio and Video Profile) is not the common RTP/AVP (Real-Time Transport Protocol/Audio and Video Profile) we have seen so far in session descriptions. RTP/SAVP identifies the SRTP (Secure RTP) protocol, which is described in Section 16.2.4.

```
m=audio 20000 RTP/SAVP 0
a=key-mgmt:mikey <mikey-specific-data-in-binary-format>
```

Figure 11.15: Key management information in SDP

11.8.1 MIKEY

The key management protocol that appears in the session description in Figure 11.15 is called MIKEY (Multimedia Internet KEYing, specified in RFC 3830 [76]). MIKEY is a binary stand-alone key management protocol that uses two messages to establish keys to encrypt and/or integrity-protect a media stream. Since the offer/answer model (specified in RFC 3264 [283]) also uses two messages it is natural to embed MIKEY into the offer/answer model. That is, the offer carries the first MIKEY message in a “`a=key-mgmt:mikey <data>`” attribute and the answer carries the second MIKEY message in a similar attribute. Since MIKEY is a binary protocol and SDP is a textual format, MIKEY data is base64-encoded (as described in RFC 3548 [196]) before placing it into the SDP attribute.

MIKEY supports three methods of establishing keys between peers: using a pre-shared key, using public key encryption, and performing a Diffie–Hellman key exchange. In the three methods, peers establish a generation key that is used together with random nonces to generate the keys to encrypt or integrity-protect the traffic.

When peers have a shared secret (i.e., a pre-shared key), all the offerer needs to do is encrypt the generation key (along with other protocol parameters) with the pre-shared key and send it to the answerer. In the public key method the offerer encrypts the generation key (along with other protocol parameters) with a symmetric key (referred to as the envelope key) which is encrypted with the public key of the answerer. The Diffie–Hellman method consists of performing a Diffie–Hellman exchange to come up with the generation key.

Chapter 12

Security in the IMS

IMS security is divided into access security (specified in 3GPP TS 33.203 [28]) and network security (specified in 3GPP TS 33.210 [29]). Access security (which we describe in Section 12.1) includes authentication of users and the network, and protection of the traffic between the IMS terminal and the network. Network security (which we describe in Section 12.2) deals with traffic protection between network nodes, which may belong to the same or to different operators.

The IMS started originally supporting IPsec for both access and network security (we described IPsec in Section 11.6). Later, support for TLS was added to both access and network (we described TLS in Section 11.3). In addition, HTTP Digest Access Authentication and the HTTP Digest Access Authentication using Authentication and Key Agreement (AKA) are also supported (see Section 11.1). Early deployments of IMS used a simplified customized security solution which leveraged authentication at the GPRS level (specified in the Technical Report 3GPP TR 33.978 [20]). Finally, a variation of the early IMS security solution has been customized for the fixed IMS access in the so-called NASS-IMS bundled authentication. We expect new security mechanisms to be added in later IMS releases. The following sections address all of these security aspects.

12.1 Access Security

A user accessing the IMS first needs to be authenticated and then authorized to use IMS before they can use any IMS services. The authentication and authorization may generally lead to the establishment of IPsec security associations between the IMS terminal and the P-CSCF, a TLS connection between them, or it may lead to a link between the specific IP-CAN and the IMS. This process is piggybacked to the current IMS registration process. The S-CSCF, armed with the authentication vectors downloaded from the HSS (Home Subscriber Server), authenticates and authorizes the user. The S-CSCF delegates the role of establishing the access security association to/from the IMS terminal to the P-CSCF. This security association can either be an IPsec connection, a TLS connection, or leveraged from the IP-CAN. During the authentication process the user also authenticates the network to make sure that they are not speaking to a forged network.

12.1.1 Authentication and Authorization

IMS support several authentication mechanisms. The actual mechanism used is determined by the commonality among the supported mechanism by the network and by the IMS terminal. In general, the authentication mechanism is determined by the presence of a security module or smart card in the IMS terminal, such as a Universal Integrated Circuit Card (UICC), and the linkage of the IMS network to the IP-CAN.

12.1.1.1 HTTP Digest Access Authentication

HTTP Digest Access Authentication is the simplest form of authentication in SIP. We have already described it in Section 11.1. The mechanism is part of the core SIP specification (RFC 3261 [286]) and it is mandatory to be implemented in clients and servers. HTTP Digest Access Authentication is a straight-forward adaptation of the same mechanism used for web authentication, and originally specified in RFC 2617 [145] for HTTP. The mechanism merely requires a username and a password. The username is the Private User Identity. The password is a shared secret stored at the HSS and known by the user. The password is never sent in the clear, so it cannot be eavesdropped. In HTTP Digest Access Authentication the server always authenticates the client; the client can also authenticate the server.

HTTP Digest Access Authentication has some security limitations and, thus, is only allowed to be used to access the IMS via access networks that have not been defined by 3GPP. In particular, if a user provides his username and password to some friends, they can access their IMS services (and presumably be charged for them) and can impersonate the original user. In addition, HTTP Digest Access Authentication does not generate session keys, which are mandated for establishing an IPsec connection. So, it is not possible to use HTTP Digest Access Authentication in conjunction with IPsec. However, it is possible to use HTTP Digest Access Authentication and establish a TLS connection. Owing to these limitations, HTTP Digest Access Authentication is typically the least preferred authentication mechanism in IMS and, in particular, is not allowed for cellular IP-CANs (e.g., GPRS), where UICC-based mechanisms are required. If HTTP Digest Access Authentication is used, it is used over a TLS connection. Section 12.1.3 discusses the establishment of the TLS connection.

When using HTTP Digest Access Authentication, mutual authentication between a user and the network in the IMS is based on a username and long-term shared secret that is known to both the user and the network. This is called the password. The username and password combination can be stored in non-volatile memory in the terminal or they might need to be entered manually when the IMS terminal is started. The username and password combination is also stored in the HSS. To achieve mutual authentication the ISIM and the HSS have to show to each other that they know the secret key. However, the terminal that contains the ISIM speaks SIP, but the HSS does not. To resolve this issue the S-CSCF assigned to the user takes the role of the authenticator. Effectively, the HSS delegates this role to the S-CSCF.

The S-CSCF uses the Diameter protocol to obtain SIP Digest authentication vectors from the HSS to challenge the user agent. These authentication vectors contain data that allows the S-CSCF challenge the user agent and calculate the answer expected from the user agent to the challenge. If the user agent answers differently, the S-CSCF considers the authentication failed.

Figure 12.1 shows the diagram for HTTP Digest Access Authentication in IMS. The IMS terminal sends an initial REGISTER request (1) that contains an Authorization header field that includes, among other data, the username, which is set to the Private User Identity,

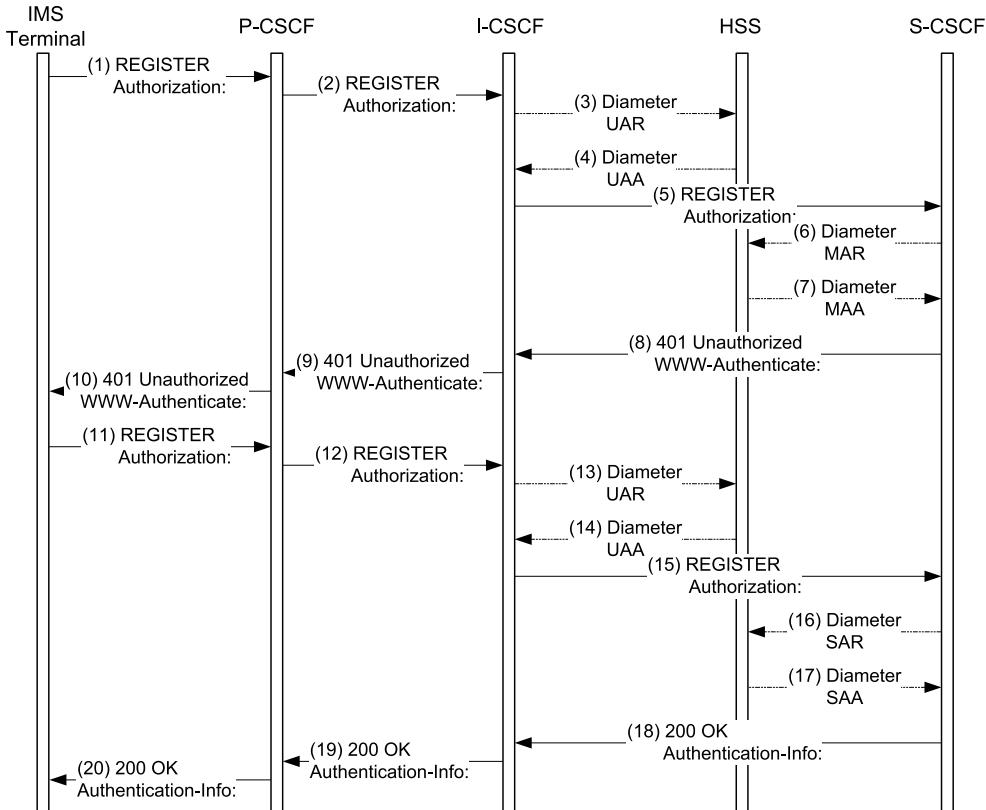


Figure 12.1: HTTP Digest Access Authentication in IMS

and realm of authentication, which is set to the home network domain name. Figure 12.2 shows an example of this header field.

```
Authorization: Digest
    username="Alice.Smith@domain.com",
    realm="domain.com",
    nonce="",
    uri="sip:domain.com",
    qop="auth",
    response=""
```

Figure 12.2: Authorization header field

An I-CSCF receives the request (2) and uses the realm and username to query the HSS (3) for routing information. The HSS returns (4) the address of a previously allocated S-CSCF, if there are any, to this user. The I-CSCF forwards the REGISTER request (5) to the S-CSCF, which sends the data included in the Authorization header field in various Attribute-Value Pairs (AVP) included in the Diameter MAR command (6) towards the HSS. The HSS then provides in the Diameter MAA command (7) the additional authentication data

that allows the S-CSCF to create a challenge. With that information, the S-CSCF creates a 401 (Unauthorized) response that contains a `WWW-Authenticate` response similar to that represented in Figure 12.8. This header field includes, among other data, a nonce, which is used to create the response, a realm, and algorithm set to MD5, denoting the HTTP Digest authentication. An example of this `WWW-Authenticate` header field is shown in Figure 12.3.

```
WWW-Authenticate: Digest
    realm="domain.com",
    domain="domain.com",
    nonce="CjPk9mRqNuT25eRkajM09uT19nM09uT19nMz50X25PZz==",
    qop="auth",
    algorithm=MD5
```

Figure 12.3: `WWW-Authenticate` header field

Eventually the IMS terminal receives the 401 (Unauthorized) response (10). The terminal computes the contents of the `WWW-Authenticate` header field and, in combination with the password, generates a response. This response is included in the `Authorization` header field of a new `REGISTER` request (11). Figure 12.4 shows an example of this `Authorization` header field.

```
Authorization: Digest
    username="Alice.Smith@domain.com",
    realm="domain.com",
    nonce="CjPk9mRqNuT25eRkajM09uT19nM09uT19nMz50X25PZz==",
    uri="sip:domain.com",
    qop="auth",
    nc=00000001,
    cnonce="0a4f113b",
    response="6629fae49393a05397450978507c4ef1",
```

Figure 12.4: `Authorization` header field

The P-CSCF receives this `REGISTER` request (11). Since there is no security associated or IP association established towards this IMS terminal, the P-CSCF adds an additional `integrity-protected` parameter set to the value “`ip-asoc-pending`” in the `Authorization` header field, prior to forwarding the `REGISTER` request (12) to an I-CSCF in the home network.

When the S-CSCF receives this `REGISTER` request (15), it determines that this is not an initial registration, due to the value of the `integrity-protection` parameter in the `Authorization` header field. Then the S-CSCF computes the response and compares it with the response included in the `Authorization` header field. If the locally computed response is equal to that included in the `Authorization` header field, then the user knows the password, therefore it is authenticated. The S-CSCF then downloads the user profile (16, 17) from the HSS, and then creates a 200 (OK) response (18) that contains an `Authentication-Info` that includes a next nonce, a response digest, which the client can then use to authenticate the server, and other data, according to the HTTP Digest Access Authentication mechanism specified in RFC 2617 [145]. Figure 12.5 shows an example of this `Authentication-Info` header field.

```

Authentication-Info:
    nextnonce="23dsFg9gSDhhSdh064SDfuNndnwUBashnRhBusbBefyrtt",
    qop="auth",
    cnonce="0a4f113b",
    nc=00000001,
    rsp-auth="4928314f023498a345b245fc24805411",

```

Figure 12.5: `Authentication-Info` header field

Eventually the IMS terminal receives this 200 (OK) response (20) and can locally compute a response for the server authentication, and verify whether it matches the response contained in the `rsp-auth` parameter of the `Authentication-Info` header field, in which case the network is also authenticated.

12.1.1.2 HTTP Digest Access Authentication using AKA

If a UICC is present in the terminal, then the IMS can use the authentication mechanism known as *HTTP Digest Access Authentication using Authentication Key Agreement* (also known as simply HTTP Digest AKA, which is specified in RFC 3310 [221]). This is the typical case of cellular access, where the UICC is required for accessing the network for circuit-switched and packet-switched services. This mechanism reuses the regular HTTP Digest Access Authentication built in SIP, but now, rather than using a regular username and a password, data derived from the UICC is used to construct the username and the password.

Let us go back to the UICC. A UICC contains one or more logical applications, as depicted in Figure 12.6. Each application stores a few configurations and parameters related to a particular usage. One of these applications is the ISIM (IP-Multimedia Services Identity Module). Other possible applications are the SIM (Subscriber Identity Module) and the USIM (UMTS Subscriber Identity Module). We described the UICC in detail as well as its applications and the parameters stored in each application in Section 3.6.

Cellular 3GPP networks allow access to the IMS when the UICC contains an ISIM or a USIM, although ISIM is preferred since it is tailored to the IMS. However, access with a USIM application is allowed in cases where the user has not updated their smart card to a more modern UICC that contains an ISIM. Owing to the weak security functions, access to IMS with a SIM application in the UICC is not allowed. Section 12.1.1.2.1 describes IMS authentication and authorization with an ISIM, whereas Section 12.1.1.2.2 describes the same procedures when the UICC contains a USIM.

The identification parameters that are stored in the ISIM in the 3GPP IMS are stored in the IMS terminal (pre-provisioned) or in the R-UIM (Removable User Identity Module) in the 3GPP2 IMS. These parameters are the same in both networks, as are the security functions. The storage can differ since 3GPP2 allows the IMS terminal or the R-UIM to store the identification parameters, but other than that there is no substantial difference. Section 12.1.1.2.1, which describes authentication and authorization with an ISIM, is also applicable to 3GPP2 networks.

12.1.1.2.1 HTTP Digest Access Authentication with AKA: UICC Contains an ISIM

This section describes the authentication and authorization procedures that take place between the IMS terminal and the network when the terminal is equipped with an ISIM

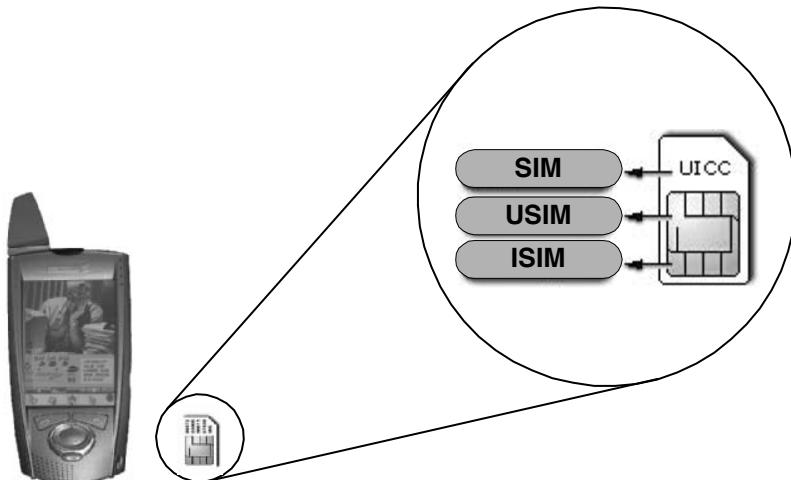


Figure 12.6: SIM, USIM, and ISIM in the UICC of 3GPP IMS terminals

application in the UICC and the authentication mechanism is HTTP Digest Access Authentication using AKA. This mechanism typically leads to the establishment of two IPsec security associations between the P-CSCF and the IMS Terminal

Mutual authentication between a user and the network in the IMS is based on a long-term shared secret between the ISIM in the terminal and the HSS in the network. Every ISIM contains a secret key. The secret key of every particular ISIM is also stored in its home HSS. Like with HTTP Digest Access Authentication, the HSS and the ISIM have to show to each other that they know the secret key. The S-CSCF assigned to the user takes the role of the authenticator, delegated from the HSS.

The S-CSCF uses the Diameter protocol to obtain authentication vectors from the HSS and to challenge the user agent. These authentication vectors contain a challenge and the answer expected from the user agent to that challenge. If the user agent answers differently, the S-CSCF considers the authentication to have failed. Let us see how the S-CSCF maps these challenges into a REGISTER transaction using HTTP Digest AKA.

The first thing an IMS terminal does when it logs onto an IMS network is to send a REGISTER request to its home network, as shown in Figure 12.7. The I-CSCF handling the REGISTER assigns, following the criteria obtained from the HSS in the Diameter exchange of messages (3) and (4), an S-CSCF for the user, that is tasked with authenticating and authorizing the user. To do so the S-CSCF downloads a number of authentication vectors from the HSS (7). Each vector contains a random challenge (RAND), a network authentication token (AUTN), the expected answer from the IMS terminal (XRES), a session key for integrity check (IK), and a session key for encryption (CK). The HSS creates the AUTN using the secret key that it shares with the ISIM and a sequence number (SQN) that is kept in sync between the ISIM and the HSS. Each authentication vector can be used to authenticate the ISIM only once. The S-CSCF downloads several vectors to avoid contacting the HSS every time it needs to authenticate the user again.

The S-CSCF uses the first authentication vector to build a digest challenge for the ISIM (as specified in RFC 3310 [221]). The S-CSCF builds a 401 (Unauthorized) response (8)

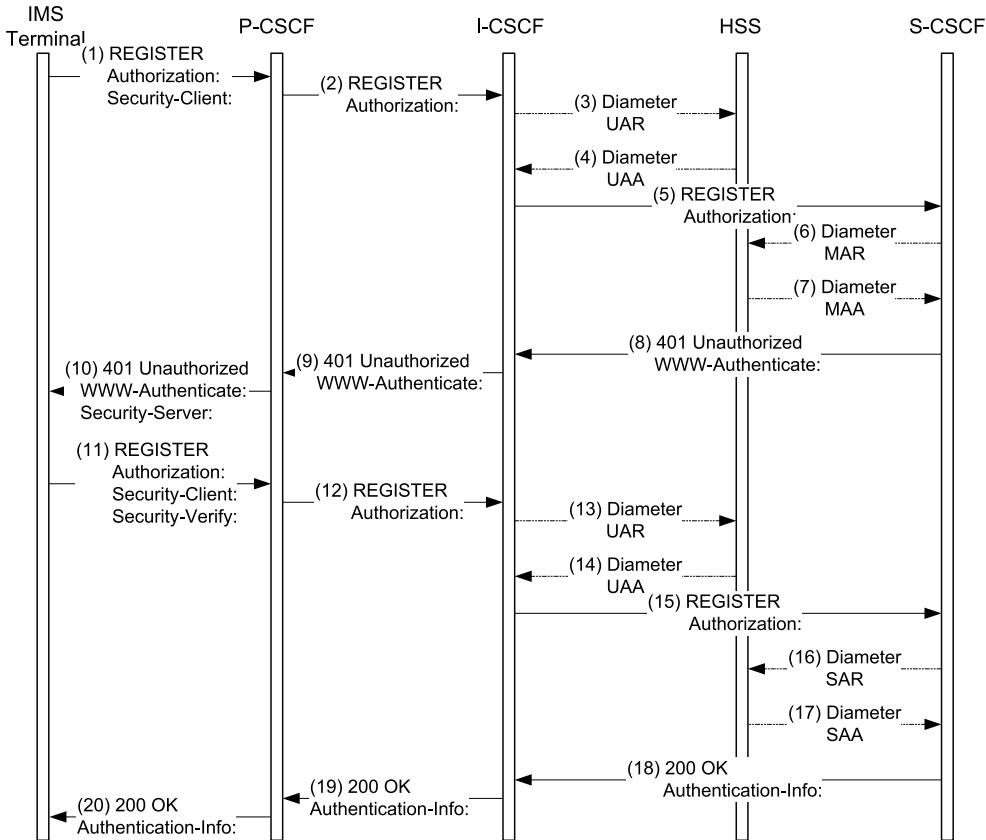


Figure 12.7: Authentication in initial REGISTER transaction

that includes a WWW-Authenticate header field (the digest operation is described in Section 11.1). The value of the nonce includes base64 encoding of the RAND and the AUTN. The value of the algorithm parameter is set to AKAv1-MD5. Figure 12.8 shows the contents of the WWW-Authenticate header field.

```
WWW-Authenticate: Digest
    realm="domain.com",
    nonce="CjPk9mRqNuT25eRkajM09uT19nM09uT19nMz50X25PZz==",
    qop="auth,auth-int",
    algorithm=AKAv1-MD5
```

Figure 12.8: WWW-Authenticate header field

When the terminal receives the 401 (Unauthorized) response (10) it deduces the RAND and the AUTN from the nonce. The ISIM verifies the AUTN using the SQN and its secret key. If the verification of the AUTN is successful, the terminal considers the network to be authenticated. In this case the ISIM uses its secret key and the received RAND to calculate the CK and IK keys and generate a response value (RES), which is returned to the S-CSCF in

the Authorization header field of a new REGISTER request (11). Figure 12.9 shows the contents of this header field.

```
Authorization: Digest
    username="Alice.Smith@domain.com",
    realm="domain.com",
    nonce="CjPk9mRqNuT25eRkajM09uT19nM09uT19nMz50X25PZz==",
    uri="sip:domain.com",
    qop="auth-int",
    nc=00000001,
    cnonce="0a4f113b",
    response="6629fae49393a05397450978507c4ef1",
```

Figure 12.9: Authorization header field

When the S-CSCF receives the REGISTER (15) it compares the RES value received with the expected value XRES in the authentication vector. If they match, the S-CSCF considers that the user is authenticated and returns a 200 (OK) response (18).

12.1.1.2.2 HTTP Digest Access Authentication with AKA: UICC Contains a USIM

An IMS terminal equipped with a UICC that contains a USIM but not an ISIM can still use the IMS. Obviously, the USIM does not contain the Private and Public User Identities or the long-term secret needed to authenticate the user by the IMS network. Still, the USIM contains an International Mobile Subscriber Identity (IMSI), which is equivalent to the Private User Identity in circuit- and packet-switched networks. The IMS terminal builds a temporary Private User Identity, a temporary Public User Identity, and a home network domain URI upon the contents of the IMSI. The procedure is described in detail in Section 5.5.2. The USIM also contains a long-term secret, typically used for authenticating in the circuit- and packet-switched networks. When the USIM is used to access an IMS network, both the network and the terminal use the long-term secret stored in the USIM and the HSS for authentication purposes.

In most cases the home operator would not like to disclose either the IMSI or the Private User Identity outside the home network. However, we have said that the temporary Private and Public User Identities are derived from the IMSI and, as we described in Section 5.5.2, they are visible in SIP messages. Therefore, the home operator has the ability to bar any Public User Identity, such as the temporary one, from being used in SIP messages other than the REGISTER request and its response. The IMS terminal can use any of the Public User Identities allocated to the user, as they are transferred to the terminal in the P-Associated-URI header field of the 200 (OK) response to the REGISTER. If an IMS terminal initiates a session with a barred Public User Identity, the S-CSCF will reject the session establishment.

12.1.2 IPsec Security Association Establishment

When the HTTP Digest Access Authentication using AKA is used for mutual authentication, the consequence of a successful authentication is the establishment of two IPsec security associations between the P-CSCF and the IMS terminal. Having two security associations, instead of one, allows terminals and P-CSCFs using UDP to receive the response to a

request (port number in the `Via` header field of the request) on a different port than the one they use to send the request (source port of the IP packet carrying the request). Some implementors believe that implementations following this behavior are more efficient than implementations using a single port, which is why 3GPP standardized a multi-port solution. On the other hand, terminals and P-CSCFs using TCP between them send responses on the same TCP connection (i.e., using the same ports) as they received the request. Figures 12.10 and 12.11 illustrate the use of ports in UDP and TCP, respectively. In both cases, one security association is established from the terminal's client-protected port to the P-CSCF's server-protected port and the other goes from the P-CSCF's client-protected port to the terminal's server-protected port. Both security associations support traffic in both directions.

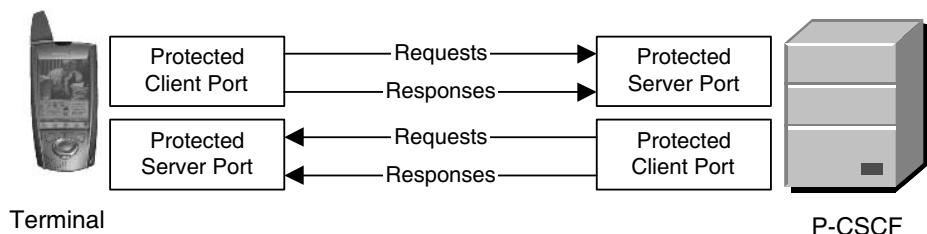


Figure 12.10: Use of ports and security associations with UDP

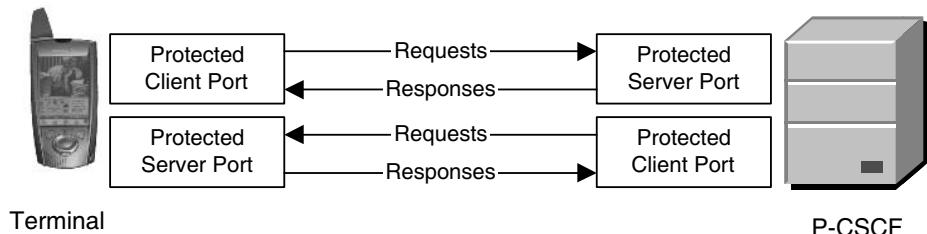


Figure 12.11: Use of ports and security associations with TCP

The P-CSCF and the terminal need to agree on a set of parameters to establish the two IPsec security associations between them (as specified in RFC 3329 [79]). The P-CSCF obtains the integrity and encryption keys (IK and CK) in a 401 (Unauthorized) response from the S-CSCF (which obtained them in an authentication vector from the HSS). The P-CSCF removes both keys from the response before relaying it to the IMS terminal. The P-CSCF and the IMS terminal use the same two REGISTER transactions (shown in Figure 12.7) that are used for authentication to negotiate the rest of the IPsec parameters.

The terminal adds a `Security-Client` header field to the REGISTER (1) (Figure 12.7) as shown in Figure 12.12. This header field contains the mechanisms (`ipsec-3gpp`), the authentication algorithm (`hmac-sha-1-96`), and the encryption algorithm (`aes-cbc`) the terminal supports as well as the SPIs (identifiers for the security associations) and port numbers that it will use.

```
Security-Client: ipsec-3gpp; alg=hmac-sha-1-96; ealg=aes-cbc;
    spi-c=23456789; spi-s=12345678;
    port-c=2468; port-s=1357
```

Figure 12.12: Security-Client header field

The P-CSCF adds a **Security-Server** header field to the 401 (Unauthorized) response (10), as shown in Figure 12.13. This header field contains the mechanism (ipsec-3gpp), authentication algorithm (hmac-sha-1-96), and encryption algorithm (aea-cbc) the P-CSCF supports as well as the SPIs and port numbers that it will use. In addition to this the P-CSCF indicates its preferred security mechanism q values. Our **Security-Server** header field has a single mechanism, but, when there are more, mechanisms with higher q values are preferred.

```
Security-Server: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96; ealg=aes-cbc;
    spi-c=98765432; spi-s=87654321;
    port-c=8642; port-s=7531
```

Figure 12.13: Security-Server header field

```
Security-Verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96; ealg=aes-cbc;
    spi-c=98765432; spi-s=87654321;
    port-c=8642; port-s=7531
```

Figure 12.14: Security-Verify header field

The security associations are ready to be used as soon as the terminal receives the **Security-Server** header field (10). So, the terminal sends a REGISTER (11) request over one of the just established security associations. The terminal includes a **Security-Verify** header field in this REGISTER, as shown in Figure 12.14, mirroring the contents of the **Security-Server** header field received previously. In this way the server will be sure that no man-in-the-middle modified the list of security mechanisms that it sent to the client. At this point in time, the two IPsec security associations are set up and being used.

In the event that there are several security mechanism supported by either the P-CSCF or the IMS terminal, all of them will be listed in the **Security-*** header fields, together with their associated metadata. This brings a new problem: an attacker could remove the strongest security mechanisms from the **Security-Server** list to force the terminal to use weaker security. With the addition of the **Security-Verify** header field an attacker that had modified the **Security-Server** list would need to break the security mechanism chosen in real time to modify the **Security-Verify** header field as well. Otherwise, the P-CSCF would notice the attack and abort the registration. This way of selecting a security mechanism is secure as long as the weakest mechanism in the list cannot be broken in real time.

12.1.3 TLS Connection Establishment

While IPsec is the initially defined security access mechanism for IMS, the P-CSCF and the IMS terminal may also support the TLS mechanism when using access networks that have not been defined by 3GPP. TLS is used together with HTTP Digest Access Authentication.

If TLS is supported, the token `tls` is listed in the `Security-*` header fields as one of the security mechanisms, i.e., at the same level as `ipsec-3gpp`. In general, all of the security parameters that can be negotiated inside the TLS connection are negotiated inside the TLS connection. This includes whether encryption and integrity protection are used, the security algorithms used for them, etc.

TLS, in general, works together with certificates. Both the client and the server can offer certificates to their counterparts. In IMS, the P-CSCF is provisioned with a certificate that allows the IMS terminal to authenticate the P-CSCF. However, the IMS terminal does not generally possess a certificate that can offer to the P-CSCF. User authentication towards the network is performed with the HTTP Digest Access Authentication mechanism that we described in Section 12.1.1.1.

The flow for TLS establishment in conjunction to HTTP Digest Access Authentication is essentially the one we described in Figure 12.7. In particular, SIP messages also use the security agreement mechanism that leads to the usage of `Security-*` header fields. Now these headers declare support for TLS. The IMS terminal establishes a TLS connection prior to sending the REGISTER request (11), so that this REGISTER request (11) and its response (20) are already protected with the TLS connection. A difference is that when the P-CSCF is going to forward the REGISTER request (12), it adds an `integrity-protected` parameter set to the value “`tls-pending`” to the Authorization header field. This informs the S-CSCF of the security association that is being set up. If the user is authenticated, then the S-CSCF generates a 200 (OK) response (18). The reception of this REGISTER request (19) at the P-CSCF moves the state of the pending TLS connection to fully established.

12.1.4 IP-CAN Linked Authentication

IMS offers two additional security mechanisms that reuse authentication previously performed with the IP-CAN and leverages that authentication at the IMS level. These mechanisms have some limitations and particular scenarios of applicability. The *Early IMS Security Solution* reuses the authentication at the GPRS level for IMS. The *NASS-IMS bundled authentication* is applicable for fixed access and reuses the authentication performed with the Network Attachment Subsystem (NASS), which is the IP-CAN for fixed access, as an IMS authentication. Both mechanisms are similar in nature. The following sections explore these two mechanisms.

12.1.4.1 Early IMS Security Solution

The Early IMS Security Solution, documented in 3GPP TR 33.978 [20], contains a number of simplifications and assumptions taken by early IMS implementations when IMS provides access over the GPRS packet network. The solution is only applicable to the GPRS access. The Early IMS Security Solution provides a fragile solution with little security for accessing the IMS. In particular, the solution is incompatible with IPsec. Owing to this, it can be considered as an implementation milestone in the race of any of the full security solutions, and should be avoided in deployments that take security seriously.

The Early IMS Security Solution creates a linkage or binding between the IP address of the IMS Terminal and the Public and Private User Identities. The GGSN plays a central role in the solution, because the GGSN allocates an IP address to the IMS terminal, once the user has been authenticated with its IMSI, which plays an equivalent role to the Private User Identity in GPRS. The GGSN provides the IP address, IMSI, and MSISDN (equivalent to the

Public User Identity) to the HSS. The HSS is provisioned with a binding of IMSI/MSISDN to Private/Public User Identities. The HSS then stores the IP address allocated to the IMS Terminal, so that any SIP signaling originated in that IP address device is authenticated and linked to the Private/Public User Identities bound to the IMSI/MSISDN of the user.

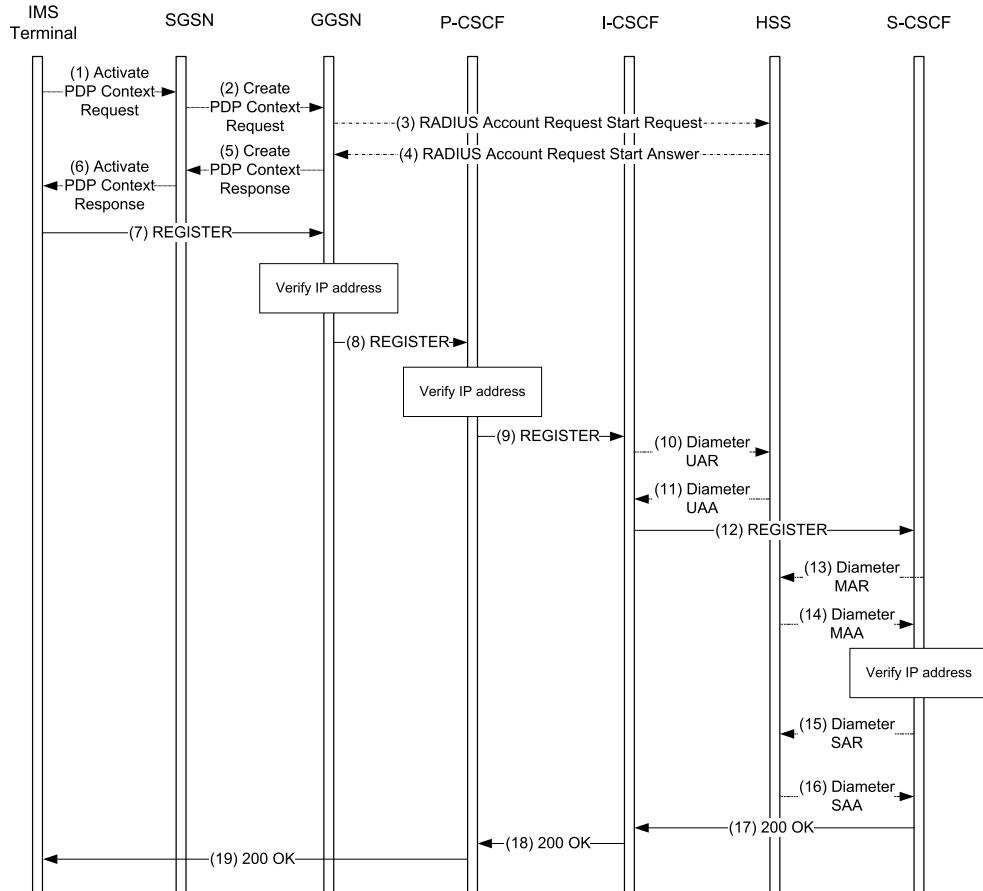


Figure 12.15: Early IMS Security Solution

We describe the Early IMS Security Solution with the help of Figure 12.15. An IMS terminal first needs to get connectivity from the GPRS network. It first sends a GPRS “Activate PDP Context Request” (1) to its SGSN, which sends a GPRS “Create PDP Context Request” (2) to the appropriate GGSN. The GGSN then allocates an IPv4 address or an IPv6 stateless auto-configuration prefix to the IMS terminal. After that, the GGSN send a RADIUS “Account Request Start Request” message to the HSS, indicating the IMSI, MSISDN, and IPv4 address or IPv6 stateless auto-configuration prefix allocated to the terminal. The HSS acknowledges the request (4). Then the GGSN and SGSN acknowledge (5, 6) their respective messages. At this point in time, the IMS terminal is properly configured with an IPv4 address or IPv6. The GGSN monitors all traffic that the IMS terminal generates to enforce that the source IP address is the IPv4 address or contains the IPv6 prefix that the GGSN allocated to the terminal.

Then the IMS terminal generates a SIP REGISTER request, similar to any of the initial REGISTER requests we have previously seen, but in this case, the IMS terminal does not include an `Authorization` header field nor a `Security-Client` or a `Security-Verify` header fields. The rest of the header fields of this REGISTER request are set as per regular registration procedures, with the only note that in those header field where the IMS terminal needs to insert the Public User Identity or home domain name, the terminal uses the IMSI-derived Public User Identity and home domain names (see Section 5.5.2 for details of the IMSI derivation of the Public User Identity). Figure 12.16 shows an example of this REGISTER request.

```
REGISTER sip:home1.net SIP/2.0
Via: SIP/2.0/UDP 192.0.2.37;comp=sigcomp;branch=z9hG4bK9h9ab
Max-Forwards: 70
From: <sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org>;tag=s82n
To: <sip:2483235551234@ims.mnc323.mcc248.3gppnetwork.org>
Contact: <sip:192.0.2.37:5060;comp=sigcomp>
          ;expires=600000
Call-ID: 23fi57lju
Cseq: 1 REGISTER
Supported: path
Content-Length: 0
```

Figure 12.16: REGISTER request (7)

Eventually the IMS terminal sends this REGISTER request (7) towards the P-CSCF. However, the REGISTER request is first received at the GGSN, which verifies the source IP address of the IP packet to be correct, i.e., the IPv4 address or containing the IPv6 prefix previously allocated by the GGSN to the IMS terminal. The GGSN only inspects the IP packet, not the inner contents (i.e., the SIP data). The GGSN forwards the IP packet containing the REGISTER request (8) to the P-CSCF.

On reception of the REGISTER request (8), the P-CSCF first verifies that the source IP address of the received IP packet matches the IP address indicated in the `Via` header field. If these two IP addresses differ, the P-CSCF adds a `received` parameter to the `Via` header field, containing the “seen” source IP address from which the IP packet was received, prior to forwarding the REGISTER request (9) to the I-CSCF. The idea being that if there is a Network Address Translator located between the IMS terminal and the P-CSCF, the source IP address of the IP packet is going to be different from the IP address that the terminal sees, which is included in the `Via` header field. The source IP address is needed for routing responses back towards the IMS terminal. Figure 12.17 shows an example of `Via` header that the P-CSCF includes in the REGISTER request (9).

```
Via: SIP/2.0/UDP 192.0.2.37; comp=sigcomp;
      branch=z9hG4bK9h9ab; received=10.5.3.7
```

Figure 12.17: Via header field in REGISTER request (9)

The I-CSCF receives a REGISTER request (9) that does not contain an `Authorization` header field. This creates a problem for the I-CSCF, because it needs to access the

Private User Identity for querying the HSS. In normal cases the Private User Identity would be explicitly included in the `username` directive of the Authorization header field. Fortunately, there is enough information in the SIP REGISTER request to derive the Private User Identity. Since the Public and Private User Identities are derived from the IMSI, the only difference between them is that the Public User Identity is a SIP URI, so it starts with the string “`sip:`” whereas the Private User Identity is not a SIP URI, so it is not prepended by that string. Since the Public User Identity is included in the `From` and `To` header fields, the I-CSCF can derive the Private User Identity and provide it to the HSS in the Diameter UAR command (10).

The HSS receives the Diameter UAR command (10), and it applies regular procedures, so, it replies with a Diameter UAA command (11) that contains the S-CSCF address, if previously allocated to that user. The I-CSCF forwards the REGISTER request (12) to the S-CSCF. The S-CSCF derives the Private User Identity in an identical way as the I-CSCF did and generates a Diameter MAR command (13) that contains the Private and Public User Identities. The HSS returns a Diameter MAA command that, in addition to regular data, contains the IPv4 address or IPv6 prefix allocated to this user, i.e., the one received in the RADIUS request (3). The S-CSCF then compares the IPv4 address or IPv6 prefix received from the HSS with the contents of the `received` parameter in the topmost `Via` header field, or with the value of the `Via` header field if there is no `received` parameter. If there is a match, the user is authenticated; otherwise, the authentication is rejected with a SIP 403 (Forbidden) response.

Assuming that the user is authenticated, the S-CSCF then downloads the user profile with the Diameter SAR and SAA commands (15, 16), and generates a SIP 200 (OK) response to the registration (17, 18, 19). This is a regular 200 (OK) response for REGISTER, which, among other data, includes the available Public User Identities in the `P-Associated-URI` header field.

12.1.4.2 NASS-IMS Bundled Authentication

The NASS-IMS bundled authentication mechanism, (specified in 3GPP TS 33.203 [28]), is only applicable to fixed access, and can be considered similar in nature to the early IMS security solution, in the sense that it is also a mechanism to authenticate the user at the lower layer for leveraging such authentication at the IMS layer. In the case of fixed access, the IP-CAN is composed of the NASS.

The NASS-IMS bundled authentication has sometimes been called “line authentication”, owing to its similarities with traditional physical telephone lines, where local exchanges authenticate the line but not the actual equipment that is connected at the other side of the line. In NASS-IMS bundled authentication, the “logical” line is authenticated, but not the user or the IMS terminal. This unveils some of the limitations of the NASS-IMS bundled authentication: users cannot roam to another access point (e.g., another building or house); anyone with physical access to the fixed line (e.g., Ethernet socket or home WLAN) can connect their IMS terminals and use IMS services at the cost of the line’s owner, something that also occurs in the traditional circuit-switched telephone world.

The NASS-IMS bundled authentication is only used in the absence of a UICC card in the terminal. Like the early IMS security solution, the NASS-IMS bundled authentication mechanism is based on associating an IP address of the IMS terminal with the IMS Public and Private User Identities, so that IMS traffic originated or destined to that IP address is authenticated.

We described the IMS-NASS bundled authentication mechanism with the help of Figure 12.18. In addition to the typical IMS nodes, the Connectivity Session and Repository Location Function (CLF) is also represented. Out of all functional entities that compose the NASS, only the CLF is involved in the IMS-NASS bundled authentication mechanism. The CLF stores a binding between the IP address allocated to the terminal and the line identifier, which is an identifier of the physical Digital Subscriber Line (DSL) used to provide the IP connectivity.

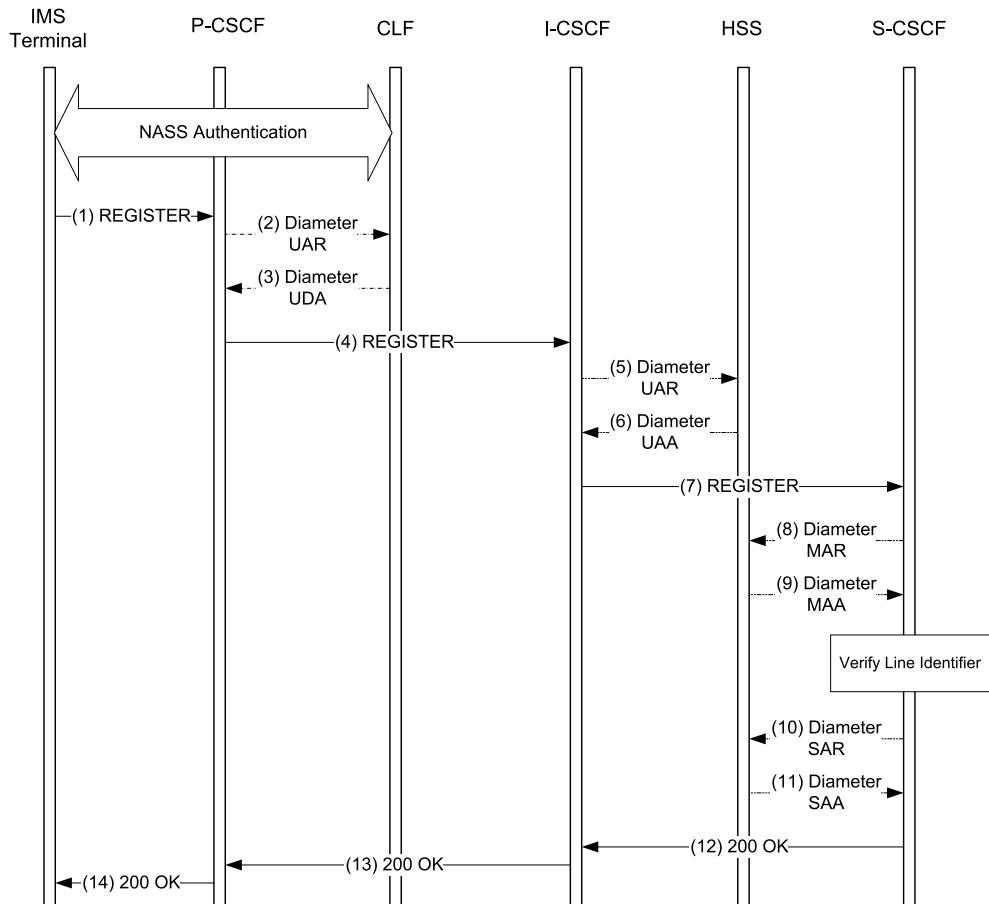


Figure 12.18: NASS-IMS bundled authentication

According to Figure 12.18 the IMS terminal obtains IP connectivity (including an IP address) from the NASS, which also contains the mentioned CLF. Then the IMS terminal sends a REGISTER request (1), which is similar to that used with any of the other security mechanisms, with the only difference that it does not include Security-Client or Security-Verify header fields, and it is optional to include an Authorization header field. The P-CSCF receives this REGISTER request (1) and sends a Diameter User-Data-Request (UDR) message (3) requesting Location Information pertaining to the user's IP address. The CLF answers with a Diameter User-Data-Answer (UDA)

message (4) providing the line identifier bound to the user's IP address. The P-CSCF then builds a new P-Access-Network-Info header field that includes a `ds1-location` parameter whose value is set to the line identifier. In order to distinguish between this P-Access-Network-Info header field and an existing one generated by the IMS terminal, the P-CSCF also includes a `network-provided` parameter. Then the P-CSCF adds this P-Access-Network-Info header field to the outgoing REGISTER request (4) and sends it to the I-CSCF.

The I-CSCF performs regular procedures, such as querying the HSS for routing information towards the allocated S-CSCF. Then it forwards the REGISTER request (7) to the S-CSCF. The S-CSCF then sends a Diameter MAR message (8) to the HSS, which is similar to a regular Diameter MAR message, except that this message might not contain a Private User Identity, e.g., because an Authorization header field was not included in the REGISTER request (7). The HSS contains a list of one or more fixed access lines which the user can use in conjunction with the IMS-NASS bundled authentication. So, the HSS includes this list of line identifiers together with other regular data in a Diameter MAA message (9) and sends it to the S-CSCF. The S-CSCF then verifies whether any of the line identifiers received from the HSS matches that received in the `ds1-location` parameter of the P-Access-Network-Info header field of the REGISTER request (7). If there is a match, then it means that the user is using any of the fixed access lines allocated to IMS-NASS bundled authentication, and the line is considered authenticated, so is the user.

The S-CSCF also downloads the user profile with the Diameter SAR/SAA pair (10, 11), and generates a SIP 200 (OK) response (12) to the REGISTER request (7). This 200 (OK) response (12) is forwarded back to the user (13, 14).

Any other SIP request that the IMS terminal generates, with the exception of ACK and CANCEL requests, are tagged at the P-CSCF with the `ds1-location` parameter in the P-Access-Network-Info header field, allowing the S-CSCF to verify the origin of the request.

12.2 Network Security

Network security deals with securing traffic between different security domains, where a security domain is a network that is managed by a single administrative authority. For example, sessions where the P-CSCF and the S-CSCF are in different networks involve traffic exchanges between different security domains.

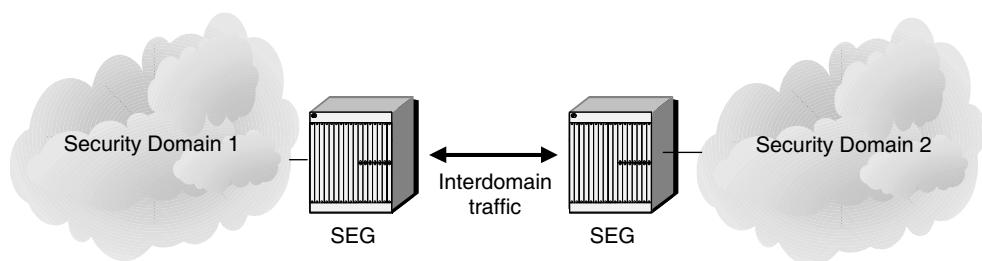


Figure 12.19: Inter-domain traffic through two security gateways

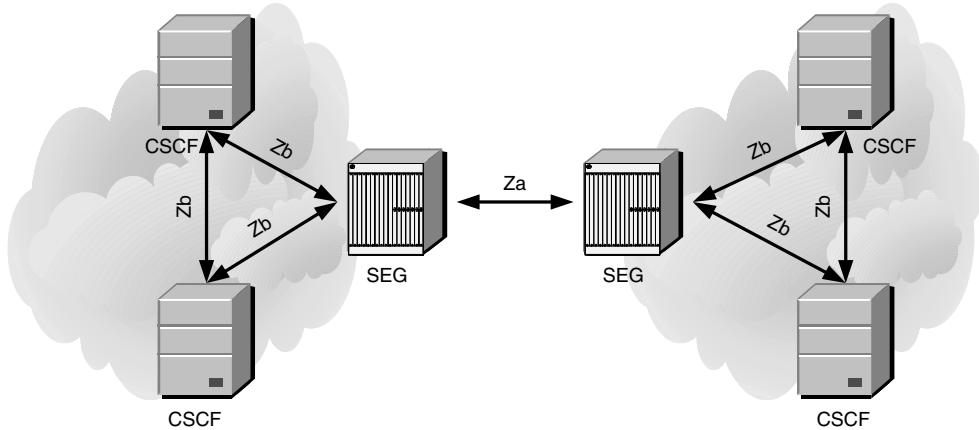


Figure 12.20: *Za* and *Zb* interfaces

All of the traffic entering or leaving a security domain traverses a SEG (Security Gateway). Consequently, traffic sent from one domain to another domain traverses two SEGs, as shown in Figure 12.19.

Traffic between SEGs is protected using IPsec ESP (specified in RFC 2406 [199] and RFC 4303 [198]) running in tunnel mode. The security associations between SEGs are established and maintained using IKEv1 (Internet Key Exchange version 1, specified in RFC 2409 [164]) or IKEv2 (specified in RFC 4306 [197]).

Within a security domain, network entities exchange traffic with the SEGs of the domain using IPsec. Network entities within a domain also use IPsec to exchange traffic between them. In this way, from a security standpoint SEGs are treated as any other network entity within the domain. Figure 12.20 illustrates this point. The interface between regular network entities and between network entities and SEGs is the same: the *Zb* interface. The interface between SEGs from different domains is called *Za*.

Authentication, integrity protection, and encryption are mandatory in the *Za* interface. This offers the inter-domain IMS traffic the maximum degree of protection.

The *Zb* interface is designed to protect the IMS signaling plane. As the interface only carries intra-operator traffic, it is up to the operator to decide whether to deploy the interface and, in the case where it is deployed, which security functions to include (integrity protection is mandatory if the *Zb* interface is implemented, but encryption is optional).

12.2.1 TLS Usage for Network Security

TLS is a built-in feature of SIP. SIP proxy servers need to implement TLS to be compliant with the specification. User Agents, such as IMS terminals and certain Application Servers, may implement TLS if they wish. Therefore, it is natural to bring TLS wherever possible into the network as well. TLS can provide integrity protection and confidentiality to the traffic sent through the TLS connection.

Any SIP element (such as CSCFs, BGCF, IBCF, HSS, etc.) can establish a TLS connection to any other node in the same network or in another network. TLS connection

establishment requires that a `sips` URI is published in DNS, with an appropriate NAPTR and SRV records indicating the TLS connection.

If the TLS connection is established between two nodes belonging to the same network, then the TLS connection can replace the `Za` interface. If the TLS connection is established between two nodes of different networks, the IP traffic is still routed through Security Gateways, which is further protected at the IP level over the `Za` interface.

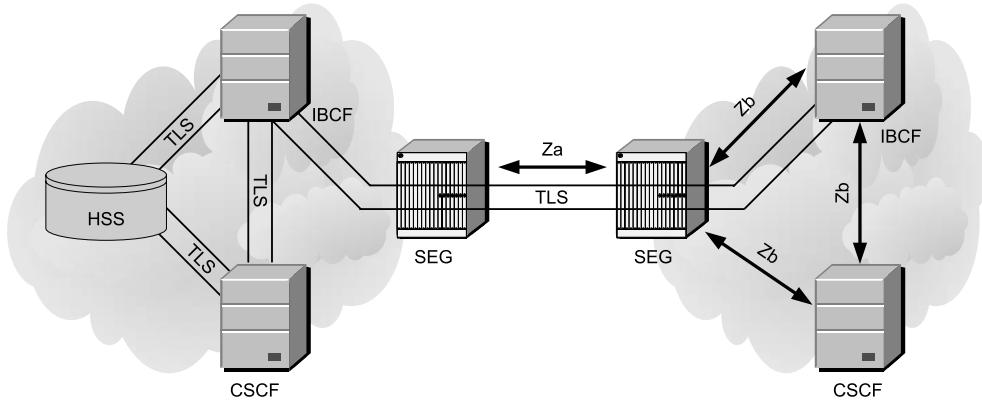


Figure 12.21: Usage of TLS in IMS

Consider the TLS usage in Figure 12.21. One of the networks has protected all of the internal interfaces with TLS. Also, the two networks have assigned one node, an IBCF in the figure, that sets up a TLS connection to another IBCF in the remote network. This TLS connection is still routed via the Security Gateways, which establish the IPsec connection. However, the TLS connection brings additional knowledge to each node at the edge of the TLS connection. First, each node can mutually authenticate the remote node using certificates. Second, once networks are authenticated, the IBCF can be provided with a list of trusted networks. If the TLS connection is established towards a trusted network, then the IBCF will not remove P-Asserted-Identity, P-Asserted-Service or any other header that requires trust between two networks.

In the absence of TLS, the `Za` interface is able to provide integrity protection and confidentiality, but IMS nodes cannot authenticate the other network, they cannot evaluate whether the other network is trusted or not (due to lack of authentication), which becomes a problem for determining the actions on certain SIP headers.

Chapter 13

Emergency Calls on the Internet

13.1 Introduction

This chapter is devoted to the complex topic of IP-based emergency calls. What makes IP-based emergency calls a complex issue? First of all, an emergency call needs to be routed to the closest Public Safety Answering Point (PSAP). This requires the call to be routed to the PSAP based on the caller's location, which in turns requires the caller's location to be determined roughly. Furthermore, it also requires a mechanism to translate the location into the real URI of the PSAP. All of these issues make IP-based emergency calls complex in comparison with regular IP-based calls using SIP.

There are three different types of emergency communications, of which this chapter covers just the first.

Citizen-to-authority communications. This type of communication refers to users dialling an emergency number, such as 112 in Europe or 911 in the USA. Usually these are referred to as *emergency calls*.

Authority-to-citizen communications. This typically refers to national emergency centers broadcasting alerting information related to emergency situations. For example, in the case of a rough weather condition, such as a hurricane or a tsunami, an alert can be issued to the population providing instructions for their safety. In some cases, this type of emergency communication may require warning messages or instructions to be launched to all citizens or a group of them who are located in a determined geographical area, e.g., where a disaster has taken place.

Authority-to-authority communications. This refers to the type of communication that takes place between two authorities during an emergency situation. This might be the communication that takes place between a national emergency coordination center and the police, fire brigade, or hospitals. Typically, this type of communication has higher priority than the rest of the calls, and it might pre-empt existing calls.

When users want to make an emergency call, they simply dial the local emergency number (e.g., 112, 911), and the call is connected to the closest emergency center, which will dispatch the requested emergency service (e.g., an ambulance). This process is simple from the user's point of view, because the user just needs to dial the local emergency number. However, the process can be split into the following building blocks.

First, the terminal needs to determine its location, or if this is not possible, some network entity must do it. This is required for two reasons. On one hand, the emergency call needs to be routed to the closest PSAP. A PSAP is a local emergency coordination center which is able to dispatch the appropriate emergency service (fire brigade, ambulance, police, etc.) to the requested location. A PSAP has a geographical area of responsibility. The size of this area varies from country to country. In some cases, a PSAP covers a small geographical area, such as a county or a city, whereas in other cases it covers a larger area, such as a state or a whole country. In any case, the idea is to connect the user with the closest PSAP and to dispatch the requested emergency personnel to the user's location. On the other hand, the location information is required at the PSAP to dispatch the emergency personnel to the user's location. Usually, the user indicates the geographical location to the PSAP agent, but if a location determination mechanism is available, then it might be more accurate or help in dispatching the emergency personnel. Location determination is further described in Section 13.2.

Second, the terminal needs to understand that the user is making an emergency call and needs to include such information in the signaling, so that exceptional procedures for emergency calls are followed. It is not necessary that the local emergency number is present in the signaling, but an indication that "this is an emergency call". The same is also true for GSM emergency calls, where there is an indication of the emergency call, but not the dialed number. Sometimes, there are different numbers for each of the emergency services, so, it might happen that the signaling contains an indication of the specific service that the user requested (e.g., police, or ambulance). The identification of emergency calls is analyzed in greater detail in Section 13.3.

Third, there is a need to find the URI of the PSAP that is able to answer the call, which is the PSAP responsible for the current user's location. As a consequence the emergency calls are routed based on the caller's location rather than on the callee's number. Either the terminal itself or a proxies in the network can determine the URI of the closest PSAP to the current location, but in any case, this determination requires a database to be queried with the user's location as input in order to obtain the routing address of the closest PSAP. Section 13.4 further analyzes the procedures to determine the closest PSAP.

Last, the emergency call is issued. If location information is available to the terminal, then the location is attached to the INVITE request that establishes the session. If not, a proxy need to determine the terminal's location to obtain the routing information to the PSAP. The INVITE request is routed to the closest PSAP where an agent answers. The agent at the PSAP can dispatch the requested service to the user's location. Issuing an emergency call is further described in Section 13.5.

The remaining sections of the chapter analyze the protocols and procedures that are available for each of these building blocks in a greater level of detail.

13.2 Location Acquisition

Location determination and acquisition is an important component of emergency calls. The terminal's location is required for routing the call to the PSAP that handles the emergency services in the area where the user is located. In addition, the PSAP can use that location information to dispatch the emergency service to that location. This section describes different mechanisms for location determination and then for transporting such location to the terminal.

Location can be expressed in either geodetic or civic format. All location determination protocols support both formats of location information. Geodetic location information can be expressed as a point, as a polygon, or as a shape of various formats.

Nearly all location determination protocols are able to convey it as a value or as a reference. A value contains the actual geodetic or civic location information. A reference is merely a pointer, such as an HTTP URI, to a server that stores the actual location. The receiver of the reference can invoke the URI to obtain the actual location, if required.

There are several mechanisms to determine the location of the user. Some mechanisms require the terminal to obtain the location, others require the network to locate it. Some mechanism are designed for fixed and wireless environments, others for the cellular space. In general, location determination can be classified into four different groups of mechanisms:

- (i) the user enters its location information;
- (ii) the access network provides access to a “wire database” with location information;
- (iii) terminal-measured location information;
- (iv) network-measured location information.

The following sections analyze the different location determination mechanisms.

13.2.1 *Manual Configuration*

The simplest mechanism for providing location information consists of manually configuring the terminal with either the geodetic or the civic location information. When an emergency call takes place, the terminal can send the pre-configured location information along with the INVITE request.

For obvious reasons, manual configuration is only applicable to fixed terminals. The location supplied is accurate as it is the configured information. However, it is prone to errors. Assume that we have a user whose fixed terminal is manually configured with location information. Assume that the user relocates to a different district or city. The terminal should be reconfigured with the new location information. If the user forgets to reconfigure it, emergency calls might be routed to the wrong PSAP. Therefore, we can think of manual configuration as the last resort for location determination.

13.2.2 *Location Acquired from DHCP*

We described DHCP in Section 5.1, where we indicated that IMS terminals can use DHCP to request configuration parameters, such as its IP address or the address of an outbound proxy (e.g., a P-CSCF in IMS). Another application of DHCP allows a terminal to request its civic or geospatial location information from a DHCP server. Essentially, the terminal includes in a DHCP request the list of parameters it is interested in receiving. So, the DHCP request includes, among other parameters, the DHCP option for Civic Addresses Configuration Information (specified in RFC4776 [297]) or the DHCP option for Coordinate-based Location Configuration Information (specified in RFC 3825 [254]). The former requests location information as a civic address, i.e., street, number, postal code, city, country, etc. The latter requests the location information as a combination of latitude, longitude, and altitude. In any case, both DHCP options are applicable to either DHCP for IPv4 (RFC 2131 [123]) and DHCP for IPv6 (RFC 3315 [124]).

This mechanism puts the burden of determining the location configuration on to a DHCP server. So, for either of these options to work correctly, the DHCP server has to gain access to the location information of the terminal that requests it. In fixed environments, such as enterprises or consumer ADSL connections, this is done in a database that matches the termination of a physical line with its civic or geospatial address. The DHCP server has access to such a database, sometimes it is even built into the server. When a new fixed line is deployed (e.g., an Ethernet switch port, the location of a WLAN access point, or an ADSL line), the civic or geospatial location information where the line terminates is added to entry of the line in the database. Then, when the DHCP server receives a DHCP request indicating either of the two options, the DHCP server queries the database with the line identifier, and obtains the requested location, which is sent in the DHCP response along with the rest of requested information (e.g., IP address, DNS server, SIP outbound proxy server).

The DHCP mechanism, although it could be applicable to both fixed and mobile networks, is typically used in stationary environments, for example, fixed networks. This also includes fixed WLAN access points.

13.2.3 *Location Acquired from Layer 2 Protocols*

There are a number of proprietary Layer 2 protocols that allow elements connected to a network to discover each other and discover how they are connected. The industry knows the burden of dealing with proprietary protocols, so in year 2005, the Link Layer Discovery Protocol (LLDP), specified in IEEE 802.1AB [170], emerged as the standard in 802-type LANs for the discovery of connected elements. LLDP is used for learning the physical topology information for network management purposes and for advertising the functionalities provided by each network element. For example, LLDP messages provide information about chassis and port identification, system name, system capabilities, system description, etc.

In 2006, the Telecommunications Industry Association (TIA) extended LLDP when it created Link Layer Discovery Protocol–Media Endpoint Discovery (LLDP-MED), specified in ANSI/TIA-1057 [74]. LLDP-MED simplifies the deployment of VoIP terminals in 802 LANs. LLDP-MED adds new messages to provide information about power over Ethernet, network policy configuration (e.g., the virtual LAN identification), media endpoint location information, and inventory management. For the purpose of emergency calls, our interest lies on the media endpoint location information that LLDP-MED is able to supply.

LLDP-MED is able to provide the location information of a terminal in three different formats. Two of these formats are defined by the IETF for use with the DHCP protocol; the third one is specified by the National Emergency Number Association (NENA). The three location information data formats that LLDP-MED supports are:

- coordinate-based location configuration information, as specified in RFC 3825 [254];
- civic address location configuration information, as specified in RFC 4776 [297];
- emergency Location Identification Number, specified by NENA TID 07-503 [218].

Owing to the nature of 802 LANs, this mechanism is applicable to stationary systems, for example, Ethernet LANs or networks terminated in managed WLAN access points.

13.2.4 Location Acquired from Application Layer Protocols

There are a number of protocols that are at the application level (layer 7) and are able to deliver the location of a device. The assumption is that a server is able to determine the location of the device. This might be the case when the server contains the database that maps either Ethernet ports or WLAN access points to the location of those ports and access points; or it might be the case when the server is able to determine the location of a mobile device through triangulation of the cells.

One of the protocols for retrieving location information at the application layer is HTTP Enabled Location Delivery (HELD). HELD assumes a server located in the access network that has access to the terminal location information. The server implements an HTTP server and the extensions specified by HELD in the Internet-Draft “HTTP Enabled Location Delivery (HELD)” [83].

HELD allows a terminal to retrieve its actual location information. This is known as location-by-value, and provides the terminal with the literal location information. HELD also allows a terminal to retrieve a pointer to its location information. This is known as location-by-reference and provides the terminal with one or more URIs that contain the actual terminal location.

HELD assumes that the client is configured with the URI of the HELD server that is able to determine the client’s location. HELD defines three messages to support the retrieval of location. These messages are, in fact, XML documents that are conveyed in HTTP requests or responses. The three HELD messages are known as *Location Request*, *Location Response*, and *error* messages.

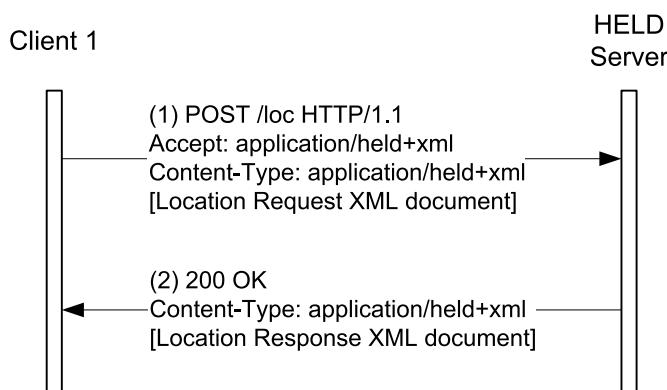


Figure 13.1: HELD operation with HTTP POST

The general operation of HELD is described with the aid of Figure 13.1. A client configured with the URI of the HELD server sends an HTTP POST request (1). This request indicates, among other information, the XML document types that the client understands. HELD documents are identified with the content type “application/held+xml”, which is inserted into the *Accept* header field. The POST request also contains a Location Request XML document, which is identified as a HELD document in the *Content-Type* header field. The Location Request document can be as simple as the example of Figure 13.2, which merely indicates that the client is interested in receiving both the civic and geodetic location information. The server retrieves the client’s location information, for example, by examining

the Ethernet port the request was received from. Then, it composes a Location Response XML document and attaches it to the 200 (OK) response (2). Figure 13.3 shows an example of a Location Response message. The example includes the location in both geodetic and civic location format. In fact, the Location Response message embeds a Presence Location Object, which we describe later in Section 19.12 when we analyze presence information documents in more detail.

```
<?xml version="1.0" encoding="UTF-8"?>
<locationRequest xmlns="urn:ietf:params:xml:ns:geopriv:held">
  <locationType>
    geodetic
    civic
  </locationType>
</locationRequest>
```

Figure 13.2: HELD Location Request XML message

A second mechanism to obtain the location is also presented in Figure 13.4. A second client wants to retrieve its location information. This second client sends an HTTP GET request (1) to its configured HELD server. Unlike the POST request (1) of Figure 13.1, this GET request (1) does not contain a body, so the semantics are merely “please send me my current location information”. Like the POST request (1) of Figure 13.1, this GET request (1) contains an Accept header field indicated the support for the HELD document format. Then the HELD server replies with a 200 (OK) response (2). Since the client did not have an opportunity to express its preference about the location format and other parameters, it is totally up to the HELD server to decide which format or formats to include in the HELD document.

13.2.5 Location Acquired from a GPS

GPS is the most sophisticated location acquisition mechanism. With the mass market advent of GPS systems, the price of GPS chips is decreasing, with the result that high-end mobile devices now contain a built-in GPS receiver. In addition, stand-alone GPS receivers can interface with mobile devices, so that the mobile device can read the current position from the GPS receiver.

The accuracy of a GPS receiver is of the order of tens of meters with 95% of confidence, which is more than enough for the purpose of emergency calls. This makes GPS the preferred mechanism for location information. However, there are some drawbacks. A GPS receiver requires a clear sky for receiving at least three GPS satellites in order to calculate the position. This precludes the usage of GPS receivers in buildings, such as offices, blocks of apartments, and any other kind of indoor facilities. It is also known that GPS has trouble in urban areas, especially where there are high buildings.

13.2.6 Wireless Triangulation

Wireless networks are also able to provide a wireless triangulation mechanism for supplying the terminal’s location. Triangulation is a process by which the location of a terminal is determined by measuring the radial distance, direction, strength, angle of arrival, or time of arrival of the received signal from three different points.

```
<?xml version="1.0" encoding="UTF-8"?>
<locationResponse xmlns="urn:ietf:params:xml:ns:geopriv:held">
    <presence xmlns="urn:ietf:params:xml:ns:pidf:geopriv10"
        xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
        entity="pres:someone@example.com">
        <tuple id="sdoihhs29">
            <status>
                <gp:geopriv>
                    <gp:location-info>
                        <gs:Circle
                            xmlns:gs="http://www.opengis.net/pidflo/1.0"
                            xmlns:gml="http://www.opengis.net/gml"
                            srsName="urn:ogc:def:crs:EPSG::4326">
                            <gml:pos>60.102937 22.320921</gml:pos>
                            <gs:radius uom="urn:ogc:def:uom:EPSG::9001">30
                            </gs:radius>
                        </gs:Circle>
                        <ca:civicAddress
                            xmlns:ca="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
                            xml:lang="FI">
                            <ca:country>FI</ca:country>
                            <ca:A3>Helsinki</ca:A3>
                            <ca:A6>Mannerheimintie</ca:A6>
                            <ca:HNO>14</ca:HNO>
                            <ca:NAM>Example Corporation</ca:NAM>
                            <ca:PC>00100</ca:PC>
                        </ca:civicAddress>
                    </gp:location-info>
                    <gp:usage-rules>
                        <gp:retransmission-allowed>false</gp:retransmission-allowed>
                        <gp:retention-expiry>
                            2007-05-25T12:35:02+10:00
                        </gp:retention-expiry>
                    </gp:usage-rules>
                </gp:geopriv>
            </status>
            <timestamp>2007-11-25T33:29:02+03:00</timestamp>
        </tuple>
    </presence>
</locationResponse>
```

Figure 13.3: Location Response XML message

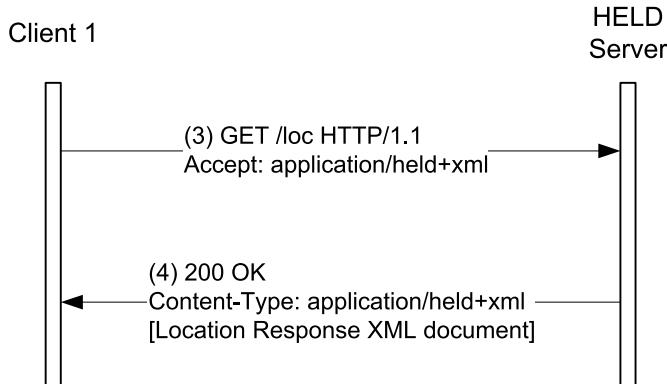


Figure 13.4: HELD operation with HTTP GET

13.3 Identifying Emergency Calls

An emergency call is typically triggered when the user dials a well-known number for emergency calls (e.g., 112, 911). In circuit-switched GSM emergency calls, the signaling associated with an emergency call does not actually contain the dialed number, but, instead, it turns a bit to indicate that “this is an emergency call”. This triggers the special procedures for routing the call based on location rather than the dialed number.

In the Internet there is also a need for indicating that the user is making an emergency call. For example, SIP proxies may need to apply special procedures to emergency calls, such as routing based on the location rather than on the dialed number. Proxies may be able to give high priority to emergency calls and bypass internal queues that, otherwise, may delay the completion of the call.

In the Internet, when a user makes an emergency call (e.g., by dialling 112, 911, or other local emergency number), the SIP User Agent uses a well-known SOS Uniform Resource Name (URN) to identify the call as an emergency call. This URN is specified in RFC 5031 [299]. The generic SOS URN is

`urn:service:sos`

The SIP User Agent inserts this SOS URN in the *Request-URI* of a SIP INVITE request.

In many countries, there are specific numbers that route calls to the police, ambulance, fire brigade, etc. RFC 5031 [299] provides other more specific URNs, including: `urn:service:sos.fire` to reach a fire service; `urn:service:sos.police` to reach a police or law enforcement force; or `urn:service:sos.physician` to reach a physician referral service.

The availability of SOS URNs allows terminal manufacturers to deploy universal devices that implement a “red button” for emergencies, independently of the country where the device is operating.

In most cases phones do not implement a red button to trigger an emergency call. Typically phones are configured with the local emergency number (either in the local configuration or available in the Universal Integrated Circuit Card (UICC)), so that when the user dials 112 or other similar emergency number, the phone creates an INVITE request whose *Request-URI* is effectively a SOS URN.

13.4 Locating the Closest PSAP

When a terminal is about to place an emergency call, it needs to learn the real SIP URI or TEL URL of the closest PSAP to the user's location. If the terminal is unable to find the SIP URI or TEL URL of its closest PSAP, a SIP proxy can assist it, in which case, the search of the SIP URI or TEL URL of the closest PSAP is delegated to the proxy. In any case, either the terminal or the proxy need to find out the PSAP URI. The IETF has developed extensions to HTTP to solve this piece of the puzzle. These extensions are specified in the Internet-Draft "LoST: A Location-to-Service Translation Protocol" [163].

LoST allows a client to request the URI that corresponds to a given location and a given service. In the case of emergency calls, the service is clearly emergency calls, and the URI is the PSAP URI. However, LoST can be used in any other context where a location and service pair needs to be translated into a URI.

The basic operation in LoST takes a location (either in geodetic or civic location information) and a service as input, and generates a URI as an output. This is the URI of the service in such location, so for the emergency call service this is the URI of the PSAP that serves that location. Furthermore, LoST is also able to convey the geographical boundary that is under the URI responsibility. Thus, a terminal can correlate its position with the PSAP boundary, and if the terminal moves outside that PSAP boundary, the terminal can do a new query to find the local PSAP for the new location.

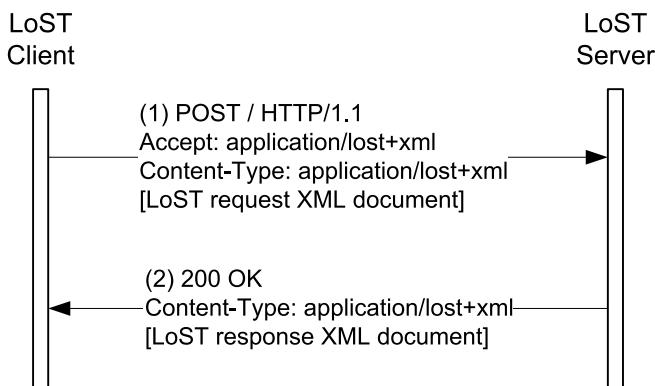


Figure 13.5: LoST operation

Like HELD, LoST reuses HTTP requests and responses to convey XML documents that operate as queries and responses. Actually, LoST uses POST requests and its responses (typically 200 (OK)) to transport the LoST queries and responses. The general operation of LoST is depicted in Figure 13.5. LoST queries and responses are, in fact, XML documents that are identified with the MIME type application/lost+xml. LoST defines the following pairs of queries and responses.

- <findService> and <findServiceResponse>. These are the main request and response in LoST. A client sends a <findService> request containing the service of its interest along with its civic or geodetic location and receives a <findServiceResponse> message that contains the URLs that map to such location and service. Applied to emergency calls, a client sends to its LoST server a

<findService> request containing its location and receives the URL of the closest PSAP.

- <getServiceBoundary> and <getServiceBoundaryResponse>. Any URL that has been mapped to a service has a boundary where the URL is applicable. For example, in emergency calls, a PSAP (which is identified by its URL) might have responsibility over a large city. LoST can convey the boundary where the PSAP URI is valid. However, a LoST server might not always send the boundary of a service in a response, but, instead, it can send a reference to it. The <getServiceBoundary> request allows a client to de-reference a service boundary, thus allowing the client to determine the boundary of URL associated to a service.
- <listServices> and <listServicesResponse>. We have indicated that LoST is a generic protocol that can be used not only in the context of emergency services, but also with any other service where a location should be mapped to a URL (e.g., a pizza delivery service). Clients send a <listServices> request to a LoST server to find out the services that the server understands.
- <listServicesByLocation> and <listServicesByLocationResponse>. This request and response pair is used to find out the available services in a given location.

```
<?xml version="1.0" encoding="UTF-8"?>
<findService
    xmlns="urn:ietf:params:xml:ns:lost1"
    xmlns:p2="http://www.opengis.net/gml"
    serviceBoundary="value"
    recursive="true">

    <location id="39a9e987e3b1" profile="geodetic-2d">
        <p2:Point id="point1" srsName="urn:ogc:def:crs:EPSG::4326">
            <p2:pos>43.6 -79.3</p2:pos>
        </p2:Point>
    </location>
    <service>urn:service:sos.police</service>

</findService>
```

Figure 13.6: A <findService> query in LoST

Let us take a look at an example of the core LoST query and response. A client generates an HTTP GET request towards its LoST server. The HTTP request includes a <findService> XML document, such as that shown in Figure 13.6. The location element contains the geodetic coordinates of interest, and the service element contains the service of interest, in this case, the police emergency services.

Then, the LoST server generates a <findServiceResponse> XML document, such as that included in Figure 13.7, and attaches it to a 200 (OK) response. The response contains the civic address corresponding to the geodetic coordinates included in the request, in addition to the SIP URI or TEL URL of the police emergency service and the boundary where such URI is applicable, which corresponds to a city.

```

<?xml version="1.0" encoding="UTF-8"?>
<findServiceResponse xmlns="urn:ietf:params:xml:ns:lost1">
<mapping
    expires="2008-01-09T23:42:12Z"
    lastUpdated="2006-11-01T01:00:00Z"
    source="ghotam.city.example.com"
    sourceId="1283b39a0b0ed329207">
    <displayName xml:lang="en">
        Gotham Police Department
    </displayName>
    <service>urn:service:sos.police</service>
    <serviceBoundary
        profile="urn:ietf:params:lost:location-profile:basic-civic">
        <civicAddress
            xmlns="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr">
            <country>USA</country>
            <A1>Ghotam State</A1>
            <A3>Gotham</A3>
            <PC>23456</PC>
        </civicAddress>
    </serviceBoundary>
    <uri>sip:police@gotham.city.example.com</uri>
    <serviceNumber>110</serviceNumber>
</mapping>
<path>
    <via source="ghotam.city.example.com"/>
</path>
<locationUsed id="2a0395b6677c27829e87354"/>
</findServiceResponse>

```

Figure 13.7: A `<findServiceResponse>` query in LoST

13.5 Issuing the Emergency Call

Let us take a look now at the process of building the INVITE request that the terminal places as an emergency call. In other words, this section tries to answer the question: what happens when the user dials the emergency number, such as 112 or 911?

To answer this question, we need to look at the different architectural models. There are several different cases, depending on the knowledge the terminal has about its location.

- (i) The terminal is able to acquire its location information. The terminal recognizes the emergency call and knows its location. The resolution of the PSAP URI can be done at either the terminal or at a proxy. The acquired location information can be a value or a reference. In the latter, either the terminal or an entity in the network may need to de-reference it.
- (ii) The terminal is not able to acquire its location information. The terminal may or may not recognize that the user is placing an emergency call. In any case, a proxy in the

network has to recognize the emergency call, provide the user's location, and determine the PSAP URI.

Let us analyze these two cases in more detail.

13.5.1 The Terminal Acquires its Location

Figure 13.8 presents an example of delivering an emergency call when the terminal is able to acquire its location information and resolve the PSAP URI. According to Figure 13.8, a user dials a string of numbers or a Emergency Service URN that is identified as an emergency call (1). The terminal takes a snapshot of its location information (2), in this example, acquired from a built-in GPS receiver. Note that any other form of location acquisition is also possible, for example, location acquired from DHCP, from a HELD server, etc.

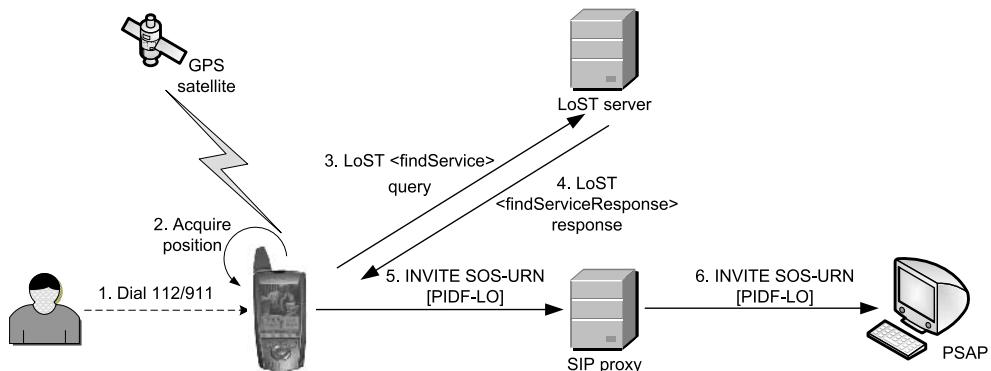


Figure 13.8: Issuing an emergency call: the terminal supplies its location information and resolves the PSAP URI

Then, the terminal must find the URI of the PSAP responsible for the current user's location. This is done by issuing a LoST `<findService>` query to a LoST server. The query (3) contains the current location and the emergency services URN. The LoST server answers with a `<findServiceResponse>` message (4) that contains the SIP URI or TEL URL of the PSAP allocated to that location, in addition to the boundary of that PSAP. It must be noted that the terminal can execute steps 3 and 4 at the time of issuing an emergency call or at any previous time, for example, when the terminal connects to the network. In the latter, as long as the terminal does not move outside the service boundary and the cached PSAP URI is not expired the terminal does not need to execute these steps.

Once the terminal has resolved the PSAP URI, then it creates an INVITE request (5). An example of this INVITE request is presented in Figure 13.9. The terminal sets the *Request-URI* and the *To* header field to the police SOS service URN. The *From* header field is populated with Alice's SIP URI. The PSAP URI is included in a *Route* header, so that the proxy that receives the request need not resolve the SOS service URN. The *Contact* header field contains a GRUU. We described GRUUs in Section 4.19.

Then, the INVITE request (5) contains two bodies: an SDP body and a Presence Information Data Format Location Object (PIDF-LO). These are presented in Figure 13.10. The PIDF-LO was initially designed as an extension to presence information, and as such,

```

INVITE urn:service:sos.police SIP/2.0
Via: SIP/2.0/TCP ws1.example.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
To: <urn:service:sos.police>
From: Alice <sip:Alice.Smith@example.com>
Route: <sip:police@gotham.city.example.com>
Call-ID: 6328776298220188511@ws1.example.com
Cseq: 1 INVITE
Contact: <sip:alice@ws1.example.com>;
    pub-gruu="sip:alice@example.com;gr=urn:uuid:
        a56d41bf-e0c9-4c80-abd3-82c40b973806";
    temp-gruu="sip:tgruu.2098dakds2dammksdf9hko@example.com;gr";
    +sip.instance=<urn:uuid:a56d41bf-e0c9-4c80-abd3-82c40b973806>;
    expires=3600
Allow: INVITE, ACK, CANCEL, BYE, MESSAGE, REFER
Supported: geolocation
Geolocation: <cid:alice23982@ws1.example.com>
    ;inserted-by=alice@ws1.example.com ;recipient=both
Content-Type: multipart/mixed; boundary=30a89j4m23
Content-Length: 1458

```

Figure 13.9: INVITE request (5)

we describe the PIDF-LO in greater detail in Section 19.12. For the time being, it is enough to know that the PIDF-LO is an XML document that contains geographical location information. In the example, the positioning information is contained in the gm:pos XML element.

Coming back to the INVITE request (5) in Figure 13.9, since the request has to include two bodies, namely SDP and PIDF-LO, they are wrapped in a multipart MIME container, which is shown in Figure 13.10. In addition, the INVITE request (5) contains a Geolocation header field, which is specified in the Internet-Draft “Location Conveyance for SIP” [253]. The Geolocation header field contains a pointer to the PIDF-LO document that follows in the message, in addition to an indication of who inserted the PIDF-LO document and who is supposed to use it. A Supported header field with the value set to the geolocation option-tag indicates that the terminal supports location conveyance.

Once the SIP proxy receives the INVITE request (5), it detects that the message tries to setup an emergency call, owing to the presence of the SOS service URN in the *Request-URI*. The proxy applies the procedures for loose routing described in the Internet-Draft “Applying Loose Routing to SIP User Agents” [278]. These procedures try to distinguish between a name (such as a URN), an address, and a route towards that address, and it does this by distinguishing retargeting from routing operations. This materializes in that the proxy retains the value of the SOS service URN that is included in the *Request-URI* and it also retains the value of the existing Route header field whose value points to the PSAP. So, in this case, the SIP proxy retains the message, including its multipart MIME body, mostly unmodified, and sends it (6) to the PSAP address. Eventually, an agent in the PSAP answers the call, inspects the attached location information, and dispatches the requested emergency service to that location.

```
--30a89j4m23
Content-Type: application/sdp
Content-Disposition: session
Content-Length: 209
Content-ID: <alice122@ws1.example.com>

v=0
o=alice 2890844526 2890844526 IN IP4 ws1.example.com
s=-
c=IN IP4 192.0.100.2
t=0 0
m=audio 20000 RTP/AVP 97 96
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event
--30a89j4m23
Content-Type: application/pidf+xml
Content-ID: <alice23982@ws1.example.com>

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
           xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
           entity="pres:alice@atlanta.example.com">
  <tuple id="349wsu82">
    <tstamp>2008-01-03T09:17:00Z</tstamp>
    <status>
      <gp:geopriv>
        <gp:location-info>
          <gml:location>
            <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
              <gml:pos>49.40 -123.26</gml:pos>
            </gml:Point>
          </gml:location>
        </gp:location-info>
        <gp:usage-rules>
          <gp:retransmission-allowed>no</gp:retransmission-allowed>
          <gp:retention-expiry>
            2008-01-03T09:18:00Z
          </gp:retention-expiry>
        </gp:usage-rules>
        <gp:method>DHCP</gp:method>
        <gp:provided-by>ws1.example.com</gp:provided-by>
      </gp:geopriv>
    </status>
  </tuple>
</presence>
--30a89j4m23--
```

Figure 13.10: Multipart body of the INVITE request (5)

It is worth noting that this model where the terminal obtains its location information from a GPS receiver and then resolves the PSAP URI demands the most from the terminal and the least from SIP proxies.

13.5.2 The Terminal Does Not Have its Own Location

In this model the terminal is not able to acquire its own location. Therefore, the functions of location acquisition and PSAP resolution are delegated to a SIP proxy. The location information is known by the access network. Since the proxy needs to find the terminal location, this model assumes that the SIP proxy is located in the access network.

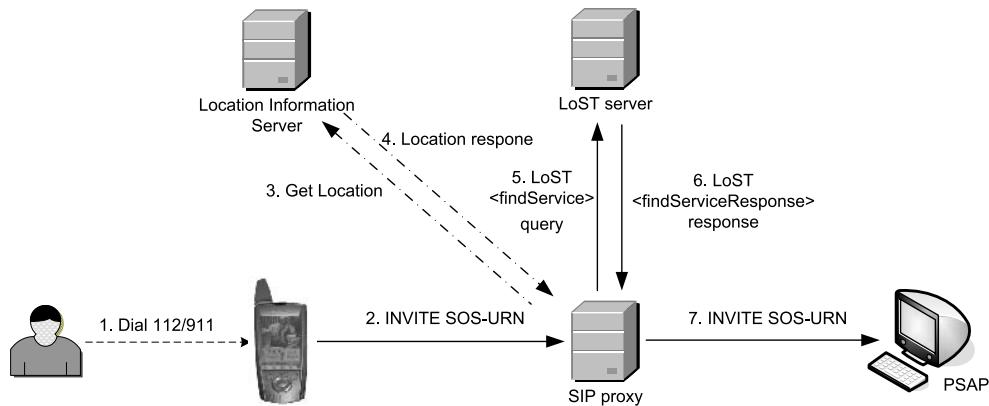


Figure 13.11: Issuing an emergency call: the terminal is unable to provide its location information

The sequence of events in this model is presented in Figure 13.11. First, the user dials an emergency number, such as 112, 911, etc. Assuming that the terminal is able to detect the dial string as an emergency call, the terminal builds an INVITE request (2), represented in Figure 13.12. In this INVITE request, the *Request-URI* and the To header field are set to the SOS service URN. The From header field is set to the user's URI. In this model, there is no Route header pointing to the PSAP URI, since the terminal is not able to obtain its location and, consequently, is not able to find the PSAP URI. Still the Contact header field contains a GRUU. In this model, though, since there is no location object, the only body is an SDP body, so there is no need for multipart MIME bodies or a Geolocation header field. The terminal sends this INVITE request (2) to a SIP proxy in the access network.

Upon receipt of the INVITE request (2), the SIP proxy in the access network analyzes the *Request-URI* and detects the call setup as an emergency call, owing to the presence of the SOS service URN. Since the INVITE request (2) does not contain a Route header, then the SIP proxy needs to first acquire the terminal's location. This requires the SIP proxy to contact (3) a location information server, which is dependent on the network itself. The location information server can be a HELD server (see Section 13.2.4), a wireless triangulation system, a wired database of location information, or any other kind of server that is aware of the terminal's location. Eventually, the SIP proxy receives the terminal's location information (4). Then the SIP proxy execute a LoST *<findService>* query (5) that returns the PSAP URI corresponding to such a location in a *findServiceResponse*

```

INVITE urn:service:sos.police SIP/2.0
Via: SIP/2.0/TCP ws1.example.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
To: <urn:service:sos.police>
From: Alice <sip:Alice.Smith@example.com>
Call-ID: 6328776298220188511@ws1.example.com
Cseq: 1 INVITE
Contact: <sip:alice@ws1.example.com>;
    pub-gruu="sip:alice@example.com;gr=urn:uuid:
        a56d41bf-e0c9-4c80-abd3-82c40b973806";
    temp-gruu="sip.tgruu.2098dakds2dammksdf9hko@example.com;gr";
    +sip.instance=<urn:uuid:a56d41bf-e0c9-4c80-abd3-82c40b973806>";
    expires=3600
Allow: INVITE, ACK, CANCEL, BYE, MESSAGE, REFER
Content-Type: application/sdp
Content-Length: 209

v=0
o=alice 2890844526 2890844526 IN IP4 ws1.example.com
s=-
c=IN IP4 192.0.100.2
t=0 0
m=audio 20000 RTP/AVP 97 96
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event

```

Figure 13.12: INVITE request (2)

response (6). The SIP proxy then builds the INVITE request (7) to be sent to the PSAP. The proxy adds a Route header field that contains the PSAP URI learned with LoST, and then the proxy applies the UA loose routing procedures: it retains the values of the *Request-URI* and the added Route header. The proxy also adds a Geolocation header field that contains a *reference* (a pointer), rather than a value, to the user's location (this is a result of the fact that pure proxies cannot add bodies).

Eventually the PSAP receives the INVITE request (7). The PSAP can de-reference the location reference included in the Geolocation header field (not shown in Figure 13.11) or an agent might request the location verbally from the user. Eventually, the emergency service is dispatched to that location.

There is a variation of this model. Assume that the terminal does not support emergency calls at all, and the user dials, e.g., 112. In this case, the terminal cannot populate the *Request-URI* with the SOS service URN, because it does not know the user is trying to place an emergency call. The terminal populates the *Request-URI* with a SIP URI that contains the dialed number, a phone context (such as the local domain name), and a parameter indicating that this is a dialed string, as per procedures of RFC 4967 [266]. An example of this dialed string that goes into the *Request-URI* is

```
sip:112;phone-context=example.com;user="dialstring"
```

If a SIP proxy receives an INVITE request with such Request-URI and the proxy is able to detect the dialed numbers as an emergency call, the proxy translates the *Request-URI* into an appropriate SOS service URN, and then the result is an INVITE request that is very similar to that shown in Figure 13.12, so the proxy applies the procedures described at the beginning of this section.

Chapter 14

Emergency Calls in the IMS

Owing to national or international regulatory requirements, most of the cellular networks in the world have to fulfill a set of requirements related to the establishment of emergency calls. The IMS is not an exception, as it is a part of a cellular network.

The requirements for the support of emergency calls vary from country to country. For instance, in many European countries it is required that the user be able to make a call to the emergency center when the IMS terminal is not even provided with a Universal Integrated Circuit Card (UICC). In other words, the network has to be able to route an emergency call to the emergency center even when the user is not authenticated. In other countries the network has to provide accurate geographical location information of the user making an emergency call.

IMS Releases 5 and 6 did not provide architectural support for delivering emergency calls over IMS. Full support for IMS emergency call is provided starting at 3GPP IMS Release 7. Release 5 and 6 IMS terminals must issue emergency calls over the circuit-switched domain. A Release 5/6 P-CSCF that receives an INVITE request for setting up an emergency call will just reject it by replying with a 380 (Alternative Service) response, indicating that the emergency call should be placed over the circuit-switched domain.

Now, let us focus on the support for emergency calls in IMS, as specified in 3GPP TS 23.167 [41]. We have just analyzed in the Chapter 13 the procedures for routing and delivering emergency calls over the Internet. We have seen in Section 13.2 the complexity of obtaining the user's location information and finding the PSAP that is responsible for this location. This chapter describes the architectural support for providing emergency calls over IMS.

14.1 Architecture for Supporting Emergency Calls in IMS

3GPP IMS has developed additional components to the IMS architecture in order to support emergency calls over IP. Before we take a look at the new nodes, let us briefly overview the features that IMS provides in the field of emergency calls.

IMS supports emergency calls issued by IMS terminals that do not recognize the call itself as an emergency call. This can happen when the user is roaming to a country where the local emergency number is not that common (for example, different than 112 or 911), or perhaps because the IMS terminal itself does not provide support for IMS emergency calls at all.

The IMS emergency services architecture also support IMS terminals that are not able to provide their location information, as well as those which can provide it. If the IMS terminal does not supply location information, the IMS network will try to locate it. If it is not possible, then the IMS network can route the emergency call to a default configured PSAP.

IMS requires the IMS terminals to be registered prior to making an emergency call. This is because for regulatory reasons in some countries, the PSAP that answers an emergency call must be able to call back at a later time. The only way the PSAP will be able to call a user back is if the user is registered with a Public User Identity that is bound to a contact address. This registration might be a regular IMS registration or a special one for the purpose of emergency calling. We describe the emergency registrations in IMS in Section 14.3.

IMS considers that most of the current deployed PSAPs are located in a circuit-switched network, such as the PSTN and, thus, it allows routing of emergency calls via a PSTN/CS gateway to PSAPs located in circuit-switched networks. However, IMS also considers that PSAPs will gradually migrate to IP-based communications, so it also supports SIP-based PSAPs.

Now, let us take a look at the architecture for delivering emergency calls over IMS. Figure 14.1 presents this architecture. The figure introduces new nodes with respect to the general IMS architecture that we described in Section 3.4. Let us analyze the role of the different nodes involved in emergency calls, starting from the IMS terminal.

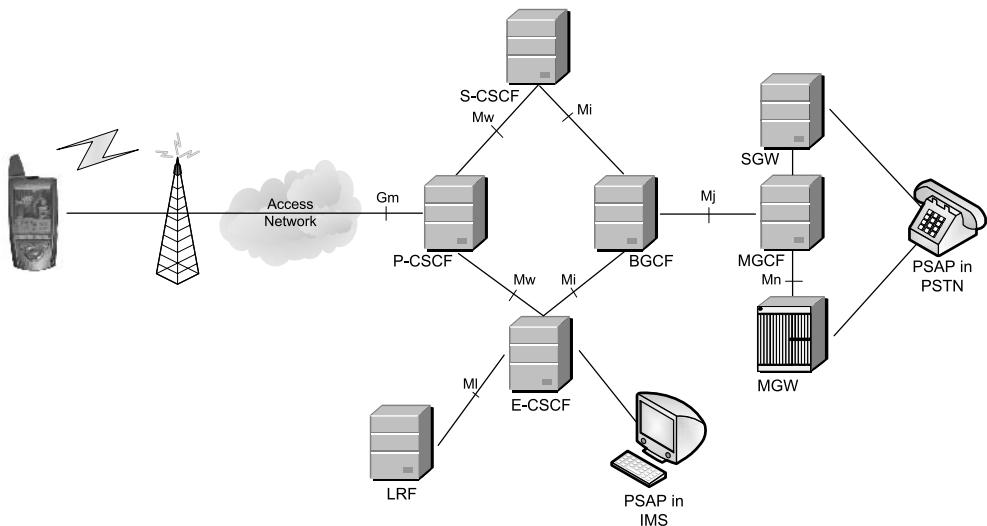


Figure 14.1: IMS architecture for supporting emergency calls

An emergency call is always initiated by the IMS terminal, typically as a result of the user dialing an emergency number (e.g., 112, 911), but sometimes, e.g., because a user presses a “red emergency button” in the IMS terminal.

If the IMS terminal is able to detect the emergency call, it follows the procedures for emergency calls that we describe in this chapter. Otherwise, it treats the call as a regular one. Depending on the network support and the roaming circumstances, this kind of regular call to a local emergency number may or may not succeed.

If the IMS terminal is not roaming on a visited network and is already registered, it can place an emergency call right away. Otherwise, the IMS terminal must perform a special

registration solely for the purpose of making emergency calls. The emergency registration provides the user with an emergency Public User Identity and an associated *tel URL*. This is needed for a potential emergency call-back, i.e., if the PSAP needs to call back the user.

If the IMS terminal is able to acquire its location information, then it includes it in the INVITE request for an emergency call. Eventually, the IMS terminal sends the INVITE request to the P-CSCF.

The P-CSCF detects emergency calls, due to dial string analysis or the presence of the SOS service URN in the *Request-URI*. If the IMS terminal is using an emergency Public User Identity, the P-CSCF does not assert such an identity.

The P-CSCF can contact the IP Connectivity Access Network (IP-CAN) to retrieve the IMS terminal location information, especially in the case of fixed broadband access, where the P-CSCF is able to retrieve line identifiers and location of that line. This is further described in Section 14.6. The P-CSCF also selects an Emergency CSCF (E-CSCF) that is located in the same network as the P-CSCF, and then forwards the INVITE request to it.

This new entity E-CSCF is used for the sole purpose of routing user-originated emergency calls. This means that if a PSAP calls back the user, that call will not traverse the E-CSCF. The E-CSCF can contact a Location Retrieval Function (LRF) to retrieve the IMS terminal location, if it is insufficient or not present in the INVITE Request. The LRF is an existing function in cellular networks, commonly used for location determination of classical circuit-switched terminals. The E-CSCF can also request the LRF to verify the location information supplied by the IMS terminal. The E-CSCF also contacts the LRF to obtain routing information and PSAP determination. The protocol that runs on the *Ml* interface, defined between the E-CSCF and the LRF, has not been specified at the time of writing.

If the routing information obtained from the LRF indicates that the PSAP is located in the IMS, the E-CSCF merely forwards the INVITE request to that PSAP. If the PSAP is identified with a telephone number, indicating that the PSAP is located in the circuit-switched domain, the E-CSCF applies procedures for breaking the call to the PSTN, i.e., forwards the call to the BGCF and PSTN/CS gateway. Policy rules in the E-CSCF can indicate that routing of emergency calls should take place via other nodes (e.g., an IBCF).

For non-roaming users, both the P-CSCF and the E-CSCF are located in the home network. For roaming users both the P-CSCF and the E-CSCF are always located in the visited network. This allows roaming users to dial visited emergency numbers that are valid only in the visited network, and have the emergency call answered by a close PSAP.

The LRF handles all of the IMS terminal location aspects. It is able to provide interim location information, initial location information, and updated location information. The LRF may interact or integrate existing functions in cellular networks, such as a Routing Determination Function (RDF) or a Gateway Mobile Location Center (GMLC). This evolved LRF that supports IMS emergency calls, which is based on the existing LRF in cellular networks for non-IMS purposes, is standardized in 3GPP TS 23.271 [14].

S-CSCFs have a small role in emergency calls. A S-CSCF is not contacted in user-originated emergency calls, but only when a PSAP is calling the user back. The S-CSCF does not require additional support for routing call-back emergency calls.

If a PSAP is located in a circuit-switched network, user-originated emergency calls traverse a BGCF and a PSTN/CS gateway. These are the regular BGCF and PSTN/CS gateways. Support for emergency calls falls into the category of mapping location information to the proper ISUP parameters.

14.2 Establishing an Emergency Call in IMS

Figure 14.2 shows the call flow of a typical IMS emergency call. In this example, we are assuming that the IMS terminal is not roaming (i.e., it is attached to its home network) and is able to detect the emergency call and supply its location information.

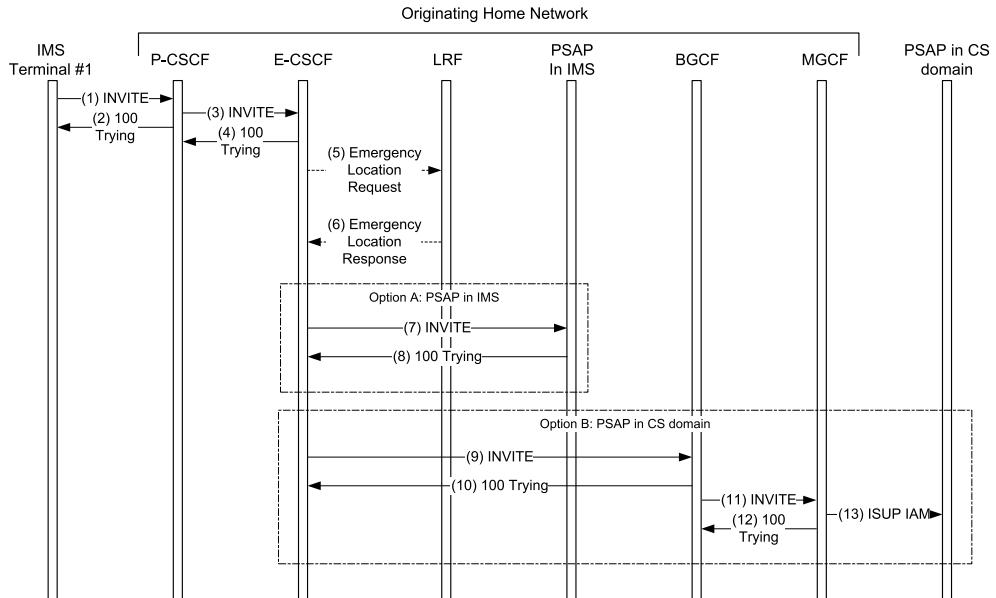


Figure 14.2: Emergency call in a non-roaming case

On issuing an emergency call, the UE creates an INVITE request (1) whose *Request-URI* and To header field contain the SOS service URN. The From header field is set to a Public User Identity in the format of a SIP URI or TEL URL. If the IMS terminal is able to acquire the identification of the cell in the radio network or the identity of a WLAN access node, then the IMS terminal includes it in the P-Access-Network-Info header field.

If the IMS terminal is able to acquire its location information it includes it in the INVITE request. If the acquired location information is a reference (e.g., a URI), then it is included in a Geolocation header field. If the IMS terminal acquires its geographical location information as a value, then it is included in a PIDF-LO body.

The IMS terminal also adds its public GRUU to the Contact header field. All in all, the INVITE request for emergency calls created under this case is very similar to that which we previously described for the Internet case, examples of which are provided in Figures 13.9 and 13.10.

Eventually, the IMS terminal sends the INVITE request (1) to its regular P-CSCF, which examines the *Request-URI*, classifies the call as an emergency call, and applies the emergency call procedures. The P-CSCF retains the SOS service URN in the *Request-URI*. In fixed broadband access, the P-CSCF contacts the NASS and retrieves the line identifier associated to the user, which is inserted in a P-Asserted-Network-Identity header field. Then, the P-CSCF selects an E-CSCF that will further route the call, so the P-CSCF adds a Route header field that contains the URI of that E-CSCF. The P-CSCF then does

regular processing to this INVITE request, except for not enforcing the Service-Route header field, not removing the P-Preferred-Identity header field, and not inserting a P-Asserted-Identity header field. Then, the P-CSCF forwards that INVITE request (3) to the E-CSCF.

The E-CSCF receives the INVITE request (3), which already contains location information. The E-CSCF analyzes the contents of the P-Asserted-Network-ID header field together with the location information to find out the address of the LRF. It then sends an Emergency Location Request (5) to the LRF to find out the address of the PSAP serving the location. The Emergency Location Request (5) contains the type of requested service (e.g., general emergency, police, ambulance, etc.), the IMS terminal's IP address, the Public User Identity, the access type, cell or line identifier, and supplied location information. With all that information, the LRF acts as a translation to service function and returns the PSAP URI in a Emergency Location Response (6) back to the E-CSCF. The LRF keeps state for the duration of the emergency call. This allows the PSAP to request an update of the IMS terminal location at any time.

The E-CSCF then has two routing options, depending on the results of the Emergency Location Response (6). If the PSAP URI is a SIP URI, then it indicates a PSAP connected to the IMS. In this case the E-CSCF removes its own URI from the Route header field, adds the PSAP URI to this Route header field, and preserves the *Request-URI* and the multipart body of the INVITE request that contains SDP and a PIDF-LO. Then, it forwards this INVITE request (7) to the PSAP in the IMS.

However, if the PSAP URI is a TEL URL, then it indicates that the PSAP is located in the PSTN. In this case, the E-CSCF uses the existing BGCF and PSTN/CS gateway to route the call. Therefore, the E-CSCF creates an INVITE request (9) that the BGCF can handle. This implies that the E-CSCF adds the BGCF URI as the topmost value in the Route header field and the TEL URL to the *Request-URI*. The E-CSCF preserves the location information included in the P-Access-Network-Info header field and the PIDF-LO object included in the body and sends the INVITE request (9) to the BGCF.

The BGCF applies regular routing procedures based on B-number analysis, selects a PSTN/CS gateway to break to a circuit-switched network, and forwards the INVITE request (11) to the selected MGCF.

The MGCF is responsible for translating the INVITE request (11) into the appropriate ISUP signaling (13). On doing so, the MGCF can use the received location information from the P-Asserted-Network-Info header field or the PIDF-LO and translate it into the appropriate ISUP parameters. Eventually, the call setup is received in a PSAP located in the CS domain.

A variation of the above call flow occurs when the IMS terminal is not able to provide its location information. In this case, when the E-CSCF sends an Emergency Location Request (5) to the LRF, in addition to the PSAP URI, the E-CSCF also request the IMS terminal location. The E-CSCF receives the IMS terminal location information and the PSAP URI in an Emergency Location Response (6). The E-CSCF, though, does not currently provide a means to convey that network-supplied location information further onwards.

14.3 IMS Registration for Emergency Calls

If an IMS terminal is willing to make an emergency call but the user is not registered to the IMS, or if it is registered but the IMS terminal is roaming, then it has to perform a special registration for the purpose of emergency calls. Once this registration for emergency calls

is completed, the user is provided with a temporary emergency Public User Identity and an associated telephone number that allows a PSAP to call back the user, if necessary.

Some types of IP-CAN allow some sort of emergency access, perhaps with no credentials. Prior to trying an IMS registration for emergency calls, the IMS terminal will try to gain an emergency access to the IP-CAN.

So, once the IMS terminal has received emergency access from its IP-CAN, then it builds an Emergency Public User Identity. The Emergency Public User Identity is built upon one of the existing Public User Identities, perhaps one that is stored in the ISIM application of the UICC, or perhaps one that is derived from a USIM application in the UICC. The Emergency Public User Identity is built by pre-pending the string “sos.” to the domain of the SIP URI of a Public User Identity. For example, for a user whose Public User Identity is `sip:alice@home1.net`, the corresponding Emergency Public User Identity is `sip:alice@sos.home1.net`.

The process for creating this emergency registration is now described with the help of the flow depicted in Figure 14.3. The IMS terminal creates a REGISTER request (1) with the From and To header fields set to the Emergency Public User Identity. The rest of the header fields are set as per regular registrations. The IMS terminal sends this REGISTER request (1) to its P-CSCF, which merely appends the `integrity-protected` parameter set to the value `no` to the Authorization header field, and forwards the request (2) to an I-CSCF in the home network, which further forwards it (6) to the allocated S-CSCF.

The S-CSCF examines the Private User Identity included in the Authorization header field, as well as the `integrity-protected` parameter, and determines that it needs to authenticate the user, so it generates a 401 Unauthorized response (8) that contains a challenge and forwards it back (9, 10) to the IMS terminal. This is very similar to regular registrations.

The IMS terminal examines the challenge contained in the `WWW-Authenticate` header field and creates an appropriate response and adds it to the Authorization header field of a new REGISTER request (11), which is forwarded (12) to the P-CSCF, now using the created security association. So, the P-CSCF appends an `integrity-protected` parameter set to the value `yes` to the Authorization header field of the REGISTER request (12) and forwards it to the I-CSCF, which forwards it (15) to the S-CSCF.

The S-CSCF verifies that the response to the challenge included in the Authorization header field is correct, an the integrity of the request. The S-CSCF binds the Emergency Public User Identity with the contents of the Path and Contact header fields. This allows to route a potential call-back from the PSAP to the user. Then, the S-CSCF creates a 200 (OK) response (18) that unlike regular registrations, does not contain a `Service-Route` header field and does not contain a temporary GRUU. The 200 (OK) response is further forwarded (19, 20) to the IMS terminal. At this time the IMS terminal is ready for creating an emergency call.

14.4 Call Back from the PSAP to a User

When a PSAP answers a call, the user’s identity is presented to the PSAP as either a regular Public User Identity or an Emergency Public User Identity. The latter is derived from the former.

If the communication established between a user and a PSAP finishes (for whatever reason), and the PSAP wants to call the user back, it has the caller’s Public User Identity, either a regular identity or an emergency one. So, the PSAP places a call to that identity by sending an INVITE request to it. Since the identity is allocated by the home network, the

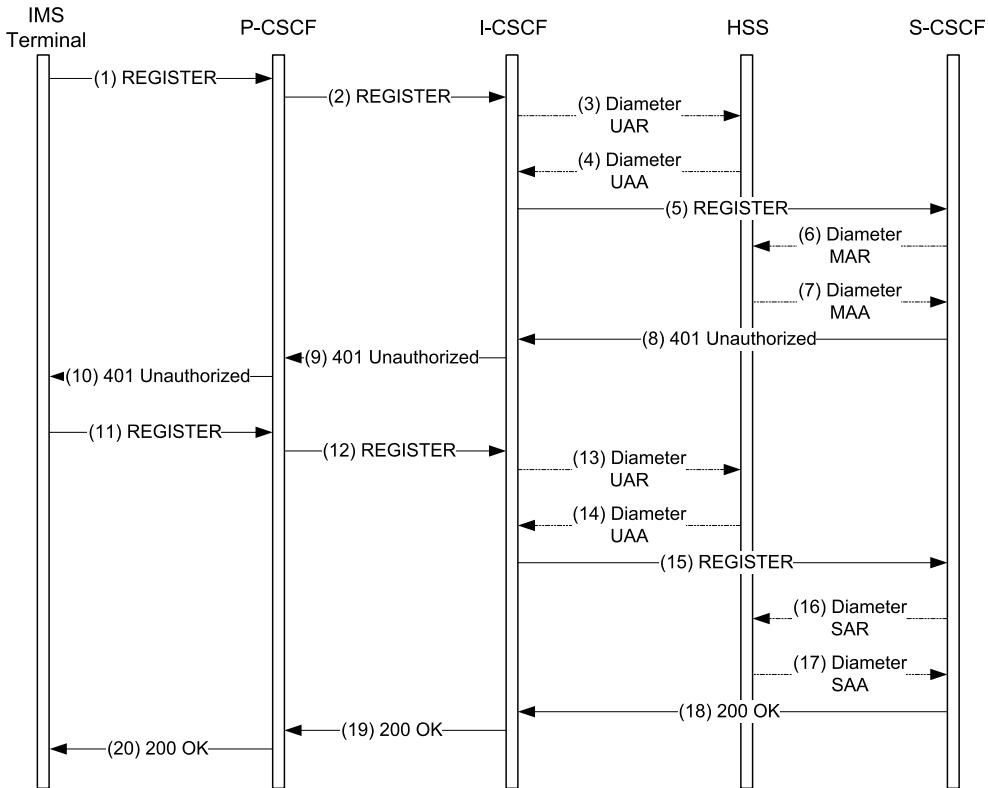


Figure 14.3: Emergency registration

INVITE request is routed to the home network and to the S-CSCF allocated to the user, in a very similar way as any other regular request is routed. Then, the S-CSCF further forwards the INVITE request to the IMS terminal via the P-CSCF, as it does with other regular INVITE requests.

14.5 Anonymous Calls

In circuit-switched networks it is technically possible to make anonymous emergency calls, i.e., calls from a mobile phone that do not contain a UICC. Therefore, the phone does not have a permanent phone number allocated to it. Although technically possible, some countries or networks operators do not allow this type of anonymous emergency calls, typically due to misuse of the service.

Similarly, it is also technically possible to make anonymous emergency calls over the IMS. This implies that the user does not have to have a Public User Identity allocated by an operator. However, before making an anonymous emergency call the IMS terminal needs to gain access from an IP-CAN. This might be anonymous access to the IP-CAN or might be a regular IP-CAN access. In any case, the IMS terminal has to gain an IP address, permission to access the network, and discover its P-CSCF. Then the IMS terminal can create an anonymous INVITE request, i.e., one whose From header field is set to any URI

in the @anonymous.invalid domain name. The rest of the INVITE request is similar to an authenticated emergency INVITE request, except for the P-Preferred-Identity header field, that now contains a SIP URI with an equipment identifier. The INVITE request is routed in the same way as any other emergency call.

14.6 Emergency Calls in Fixed Broadband Accesses

In fixed broadband access, there are additional procedures for placing emergency calls. For example, the IMS terminal may also use DHCP to obtain its location information. We described location information acquisition from DHCP in Section 13.2.2. Also, the IMS terminal can be considered attached to the home network, so roaming-specific cases are not applicable.

Fixed broadband networks also support emergency calls made from IMS terminals that are not able to supply their location information. Supporting these cases require an entity, such as the P-CSCF, to contact an LRF to request the location information. The LRF is part of the NASS, so it has access to detailed location information. The P-CSCF implements a new interface to the LRF, known as *Mq* for the purpose of emergency calls, which actually corresponds to the *e2* interface in fixed networks and it is based on the Diameter protocol. The details of the architecture are presented in Figure 14.4.

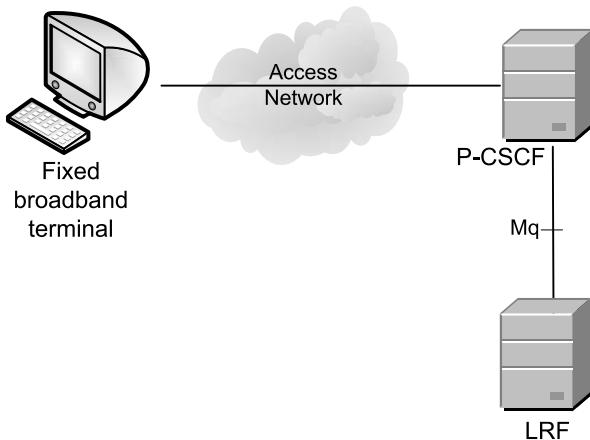


Figure 14.4: Additional interface in fixed broadband access

According to this additional interface, if the P-CSCF in a fixed broadband network receives an INVITE request for an emergency call that does not contain location information, the P-CSCF provides the line identifier to the LRF in order to obtain the location information of the IMS terminal. The line identifier is the same identifier that is used in the NASS-IMS bundled authentication in fixed access (see Section 12.1.4.2). Then the emergency call procedure continues as it does for any other emergency call.

Part III

The Media Plane in the IMS

Part III deals with the IMS media plane. We tackle several media types, although our focus is set on audio and video, especially in Chapter 15, where we describe how to represent audio and video signals in digital format using audio and video codecs. Chapter 16 describes how to transport encoded media in an IP network.

We follow a similar approach that we chose for Part II: we first introduce the technology and then explain how it is used in the IMS.

Chapter 15

Media Encoding

Media encoding is not an Internet technology. Hence, it is different from most of the technologies described so far. While many of the technologies we discussed earlier were originally developed for the Internet and later adapted to work in the IMS, media-encoding technologies were developed for other environments. For example, speech coding was developed with circuit-switched telephony in mind.

Owing to this we do not follow the same structure as in previous chapters, where we described how each technology is used on both the Internet and in the IMS. In this chapter we describe how speech-encoding technologies evolved from the simple codecs used in the fixed PSTN to the advanced codecs used in today's cellular networks. In addition, we look at video-encoding technologies.

15.1 Speech Encoding

Audio encoding consists of converting an analog audio signal into a digital signal. This digital signal is transmitted over the network and decoded at the receiver's side, as shown in Figure 15.1. If the audio signal consists of human speech, the process just described is referred to as speech encoding. We focus on speech encoding in this section.

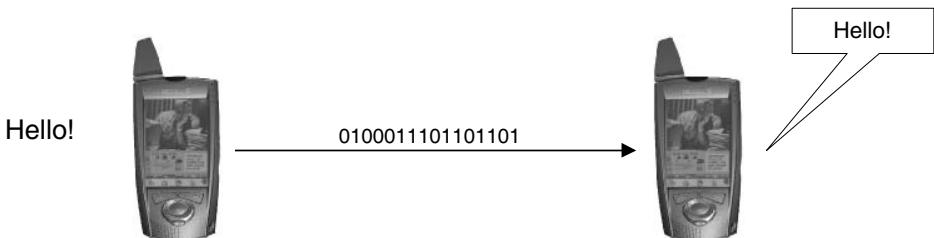


Figure 15.1: Digital transport of analog signals

The algorithm used to encode and decode the digital signal is referred to as a codec. Two important characteristics of any codec are its speech quality and its bandwidth. In general, codecs with higher bandwidth can achieve better quality. However, modern speech codecs achieve good speech quality with bandwidths below 10 kbit/s.

15.1.1 Pulse Code Modulation

The G.711 (ITU-T Recommendation G.711 [177]) codec, better known as PCM (Pulse Code Modulation), is the codec used in the fixed PSTN. In addition, nearly all of the SIP phones on the Internet support this codec.

PCM uses a sampling rate of 8000 Hz. That is, PCM takes 8000 samples per second and encodes each of them separately. This way, each sample contains 0.125 ms of speech, as shown in Figure 15.2.

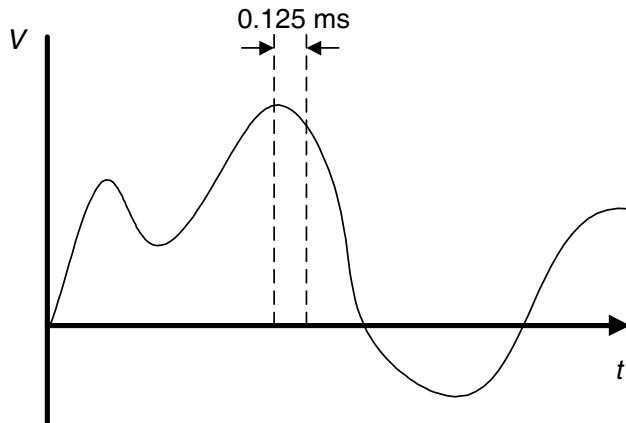


Figure 15.2: Sampling of an analog signal

For each sample it is necessary to encode the amplitude of the audio wave at that point. Still, even if the amplitude of the audio wave varies within the sampling interval (as we can see in Figure 15.2), a single value of the amplitude is provided per sample. Approximating the value of the amplitude in an interval by a single number creates some distortions, which could be reduced by making the sampling interval smaller. The sampling interval chosen by PCM (0.125 ms) provides enough quality to encode human speech.

Compact discs (CDs) storing music use a PCM codec at 44.1 kHz. This sampling frequency yields a sampling interval that is small enough to encode music with high fidelity, but requires a much larger bandwidth than the PCM codec used for telephony which we are describing.

Digital codecs such as PCM work with discrete values. So, PCM needs to provide a discrete value for the amplitude of the audio wave for each sample. PCM encodes this value using 8 bits, which give 256 different values. The mapping between the measured value of the amplitude and these 256 values is done by dividing the amplitude range into intervals and assigning a value to each interval, as shown in Figure 15.3. This process is known as quantization.

In quantization, the narrower the intervals the greater the fidelity and the higher the bandwidth (there are more values to encode). So, quantization offers the same tradeoff as sampling. In any case the distortion caused by quantization can be reduced if we know how our audio input looks. In our case we know how the audio input looks because human speech has been studied extensively.

PCM assigns narrower quantization intervals to amplitude values that are more common in human speech. In this way we obtain a higher quality in most cases and lower quality only

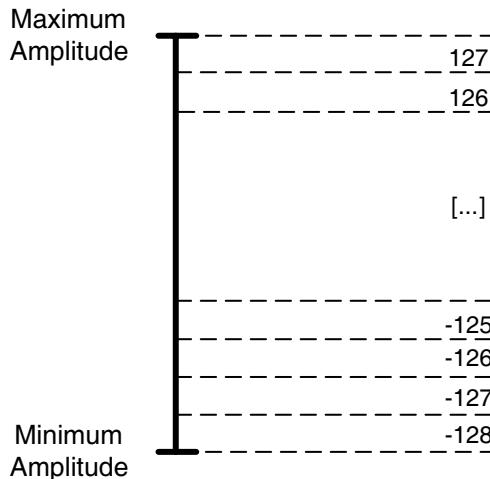


Figure 15.3: Quantization of an amplitude range

in corner cases (e.g., people screaming on the phone). PCM comes in two flavours called A-law and μ -law, depending on how the quantization is performed (both use logarithmic scales). A-law is used in Europe and μ -law is used in the USA. Other codecs, such as G.726 (ITU-T Recommendation G.726 [178]), use adaptive quantization. That is, the quantization intervals are made larger or smaller depending on the characteristics of the previous samples (sometimes, even later samples are taken into consideration).

The PCM codec we have just described produces 8000 8-bit samples per second. This yields a total bandwidth of 64 kbit/s. Now, let us look at how other codecs achieve an acceptable voice quality using a lower bandwidth than PCM.

15.1.2 Linear Prediction

Modern codecs can encode human speech using bandwidths well below the 64 kbit/s of PCM. One of the tools that allows us to do this is linear prediction. Codecs using linear prediction take advantage of the correlation between different voice samples to encode speech more efficiently. Linear prediction works on audio frames rather than on individual samples. An audio frame consists of a number of consecutive samples; a typical choice is to use 20 ms frames.

Human speech is highly correlated. That is, a particular voice frame can be expressed fairly accurately as a linear combination of previous frames. This property is used by linear prediction codecs. These codecs send the linear coefficients that describe the frame at hand; the decoder reconstructs the voice frame using these coefficients and the voice frames that it previously decoded. We refer to these coefficients as Linear Predictive Coding (LPC) coefficients.

However, LPC coefficients only describe a voice frame approximately. The difference between the original signal and the signal described by the LPC coefficients is the residual signal, as shown in Figure 15.4.

This residual signal does not show any correlation between nearby samples (also called short-term correlation) because the LPC coefficients remove this correlation. Nevertheless,

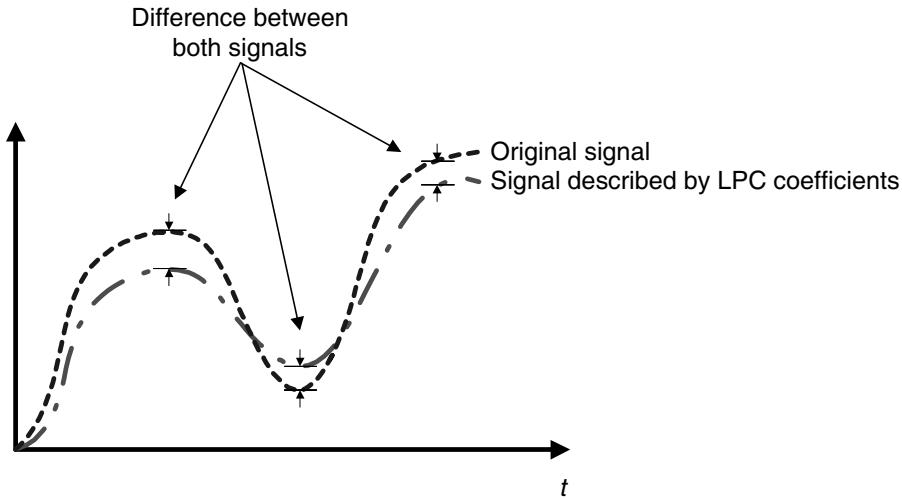


Figure 15.4: Residual signal after calculating the LPC coefficients

there is still a long-term correlation in this residual signal, caused mainly by the pitch of the voice. It is possible to apply linear prediction again, this time to this residual signal, to remove this long-term correlation. The parameters that approximate this residual signal are called Long-Term Predictors (LTPs).

If we add the signals described by the LPC coefficients and by the LTPs, we have a signal which is fairly close to the original signal. The difference between the original signal and this signal is another residual signal. This residual signal is typically encoded by the codec (different codecs use different mechanisms to encode this signal).

When the coder finishes the process that we have just described, it sends the LPC coefficients, the LTPs, and the encoded residual signal to the decoder. This information enables the decoder to produce a signal that is close enough to the original. Figure 15.5 shows the different contributors to the decoded signal (we do not show the original signal in this figure).

15.1.3 GSM-FR

Let us look at how a real codec uses the linear prediction mechanisms we have just described to encode human speech. We have chosen the GSM-FR (GSM Full Rate, specified in ETSI GSM 06.10 [129]) codec, a codec that has been implemented in millions of GSM terminals around the world. GSM-FR encodes speech at 13 kbit/s.

GSM-FR uses 20 ms frames to calculate eight LPC coefficients: two 6-bit coefficients, two 5-bit coefficients, two 4-bit coefficients, and two 3-bit coefficients. This yields a total size of 36 bits.

The residual 20 ms signal (the original signal minus the signal described by the LPC coefficients) is divided into four 5 ms subframes in order to calculate the LTPs. GSM-FR uses two LTP parameters to approximate this residual signal: the 7-bit LTP lag (the lag value indicates the period of the pitch) and the 2-bit LTP gain.

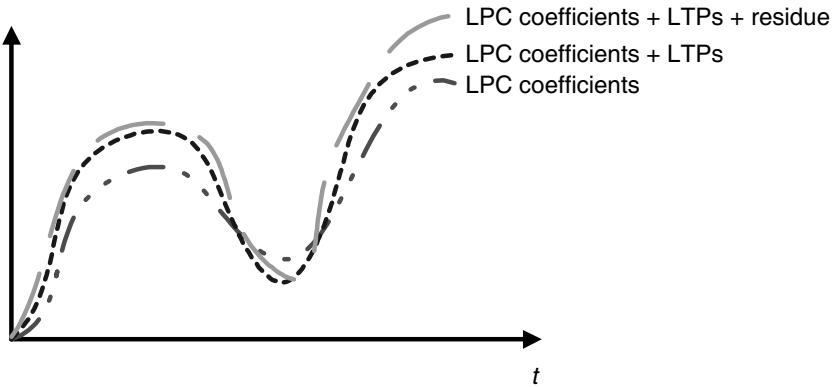


Figure 15.5: Contributors to the decoded signal

The next step consists of encoding the residual signal remaining after having calculated the LTPs. This signal is encoded using a procedure called RPE (Regular Pulse Excitation).

The residual signal after having calculated the LTPs consists of a 5 ms frame that contains 40 samples (each sample is 0.125 ms long). Instead of encoding all 40 samples, GSM-FR only encodes one-third of them in order to save bandwidth. The codec defines the following four sequences of samples:

- (1) sample 0, sample 3, sample 6, ..., sample 36;
- (2) sample 1, sample 4, sample 7, ..., sample 37;
- (3) sample 2, sample 5, sample 8, ..., sample 38;
- (4) sample 3, sample 6, sample 9, ..., sample 39.

The codec chooses the sequence that best approximates the residual signal (sequence selection is encoded using 2 bits, since there are four sequences) and encodes the samples of the chosen sequence using 3 bits per sample plus 6 bits to encode the maximum energy of all the samples in the sequence. Figure 15.6 shows how each parameter contributes to the total final bandwidth of 13 kbit/s.

15.1.4 AMR

Let us now describe the AMR (Adaptive Multi-Rate) speech codec (defined in 3GPP TS 26.090 [8]), which is the mandatory speech codec for 3GPP IMS terminals. That is, every 3GPP IMS terminal supports AMR and may optionally support other speech codecs, such as those described in previous sections.

15.1.4.1 AMR Modes

AMR consists of eight different codecs, each with a different bandwidth. Their bandwidths are 12.2, 10.2, 7.95, 7.40, 6.70, 5.90, 5.15, and 4.75 kbit/s. These codecs are referred to as AMR modes. The 12.2, 7.40, and 6.70 kbit/s AMR modes are also known as GSM-EFR (Enhanced Full Rate), TDMA-EFR, and PDC-EFR, respectively.

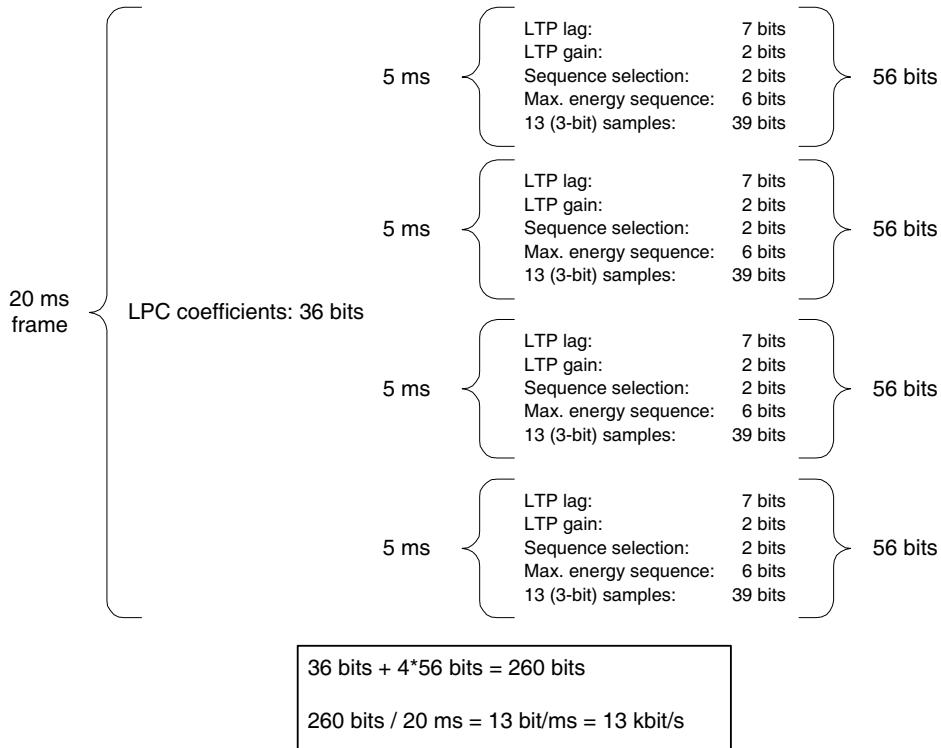


Figure 15.6: Contributors to the total bandwidth of GSM-FR

The AMR modes were initially designed to be used in GSM networks that provide a fixed bit rate for circuit-switched voice calls. This rate is split into channel encoding and data encoding. When the signal-to-noise ratio is low (poor radio quality) the terminals use low-bandwidth modes (e.g., 4.75 kbit/s). When the signal-to-noise ratio is high (good radio quality) the terminals use high-bandwidth modes (e.g., 12.2 kbit/s).

The AMR codec itself is capable of switching mode on a frame-by-frame basis. That is, one 20 ms speech frame may be encoded using an AMR mode and the next speech frame may be encoded using a different mode. Still, some circuit-switched transports impose tougher limitations on how often mode switching can be performed.

3G networks using WCDMA access do not use AMR modes in the same way as GSM networks. WCDMA uses fast power control and does not perform mode adaptation at the channel-encoding level. WCDMA networks use low-bandwidth modes to gain capacity when many users make voice calls at the same time.

When AMR is transported over RTP/UDP/IP (see Section 16.2.2) the overhead introduced by the RTP, UDP, and IP headers is fairly large. So, unless header compression is used the network does not gain much extra capacity by using the AMR low-bandwidth modes instead of high-bandwidth modes. Still, all of the modes need to be supported in order to make it easier to interoperate with circuit-switched networks.

15.1.4.2 LPC Coefficients Calculation

Let us now see how AMR encodes the speech. AMR uses linear prediction; therefore it uses LPC coefficients to approximate speech signals. These LPC coefficients are calculated differently in the 12.2 kbit/s mode from other modes. However, speech is divided into 20 ms frames in all modes. Each of these frames contains 160 0.125 ms samples.

In the 12.2 kbit/s mode, linear prediction is performed twice on a 30 ms window that contains the last 10 ms of the previous speech frame and the whole 20 ms of the speech frame at hand, as shown in Figure 15.7. Linear prediction is performed twice over the same window, but using different functions. In the remaining modes, linear prediction is performed only once over the 30 ms window.

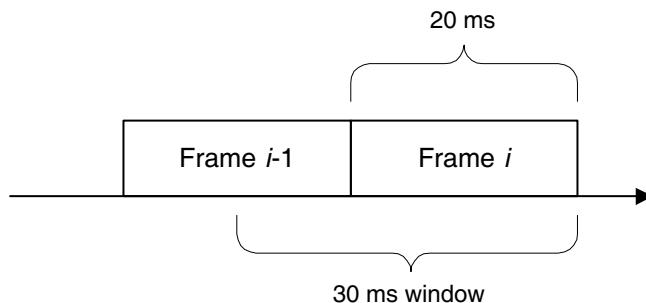


Figure 15.7: Window selection to perform linear prediction

Once the LPC coefficients are calculated they are transformed into LSPs (Line Spectral Pair). LSPs are the root of two polynomials that are formed using LPCs. Encoding LSPs instead of LPCs allows AMR to achieve a more compressed representation of the signal.

15.1.4.3 Codebooks

AMR uses two codebooks to encode the residual signal that we obtain after calculating the LSPs. The codebooks contain *excitation vectors* that are used by the decoder to synthesize a signal that approximates the residual signal being encoded. This way of encoding the residual signal is referred to as CELP (Code Excited Linear Prediction). AMR uses an adaptive codebook and a fixed codebook.

The contents of the adaptive codebook vary as subsequent speech frames are encoded. The adaptive codebook is used to encode the pitch of the speech. The parameters for the adaptive codebook are the *lag* and the *gain*.

The fixed codebook is implemented using an algebraic codebook. That is, an algebraic code is used to generate excitation vectors, also known as *innovation vectors*.

15.1.4.4 Adaptive Codebook

AMR uses both open-loop and closed-loop analysis to estimate the pitch of the speech. Open-loop analysis is performed first and provides near-optimal lag values. Closed-loop analysis is performed around lag values that were obtained previously and provides the final lag values to be encoded. Performing open-loop analysis before close-loop analysis reduces the difficulty of searching for the lag values.

The 12.2, 10.2, 7.95, 7.40, 6.70, and 5.90 kbit/s AMR modes perform open-loop pitch analysis on 10 ms speech windows, which gives two estimates of the pitch lag for each 20 ms frame. The 5.15 and 4.75 kbit/s AMR modes perform open-loop pitch analysis on 20 ms speech windows, which gives a single estimate of the pitch lag for each 20 ms frame.

Once the estimates of the pitch lag are obtained, AMR performs closed-loop pitch analysis around these estimates. This process results in the final value for the pitch lag. Then, AMR calculates the adaptive codebook gain value.

15.1.4.5 Fixed Codebook

The residual signal after calculating the parameters of the adaptive codebook (lag and gain) is approximated using a fixed algebraic codebook. This codebook contains a set of fixed pulses. The coder performs a search in this codebook to decide which pulse approximates the residual signal best.

15.1.4.6 Gains

The adaptive codebook gain was calculated after calculating the pitch lag. Let us now see how AMR calculates the fixed codebook gain.

Given the algebraic code, it is possible to predict the fixed codebook gain value. AMR takes advantage of this and only sends a correction factor to the decoder. The decoder obtains the real value for the fixed codebook gain using the predicted value and the correction factor.

The 12.2 and 7.95 kbit/s AMR modes encode the adaptive codebook gain and the correction factor separately, using 4 and 5 bits, respectively. Both values are provided for 5 ms speech windows.

The remaining modes encode the adaptive codebook gain and the correction factor jointly. The 10.2, 7.40, and 6.70 kbit/s modes use 7 bits per 5 ms speech window. The 5.90 and 5.15 kbit/s modes use 6 bits per 5 ms window. The 4.75 kbit/s mode uses 8 bits per 10 ms window.

Figures 15.8–15.15 show the different contributors to the final bandwidth of all the AMR modes.

15.1.5 AMR-WB

The AMR-WB (AMR-WideBand) codec (defined in 3GPP TS 26.190 [22]) encodes voice using 16 000 samples per second, as opposed to the 8000 samples per second used by AMR. This higher sampling frequency allows AMR-WB to encode a wider range of frequencies. Consequently, AMR-WB encodes speech with higher quality than the codecs we described earlier.

AMR-WB consists of a family of codecs which encode audio using the following bandwidths: 23.85, 23.05, 19.85, 18.25, 15.85, 14.25, 12.65, 8.85, and 6.60 kbit/s. AMR-WB is the mandatory codec for 3GPP IMS terminals that provide wideband services.

15.1.6 SMV

3GPP2 IMS terminals support the EVRC (Enhanced Variable Rate Codec) and the SMV (Selectable Mode Vocoder) codecs. First- and second-generation CDMA terminals already implement EVRC, while SMV is the preferred speech codec for CDMA2000® access. Let us now describe how SMV works.

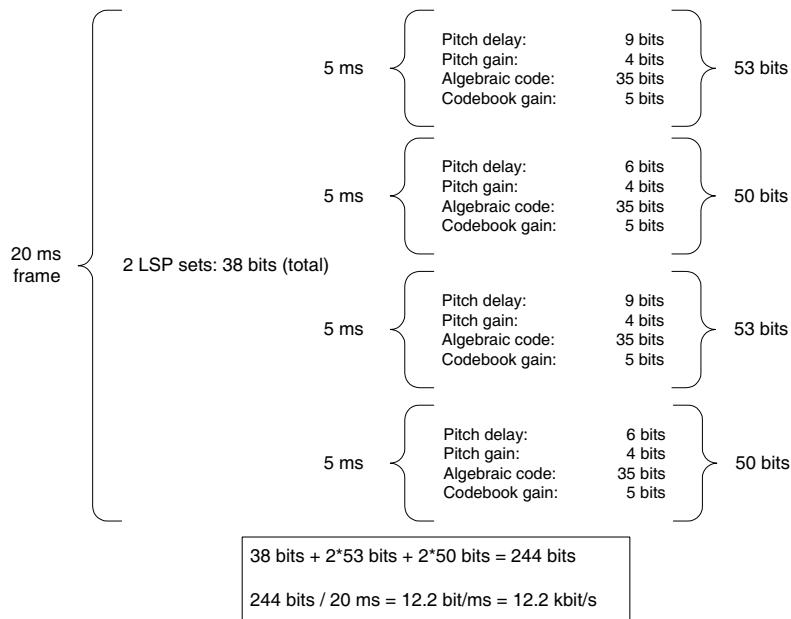


Figure 15.8: Contributors to the total bandwidth of 12.2 kbit/s AMR

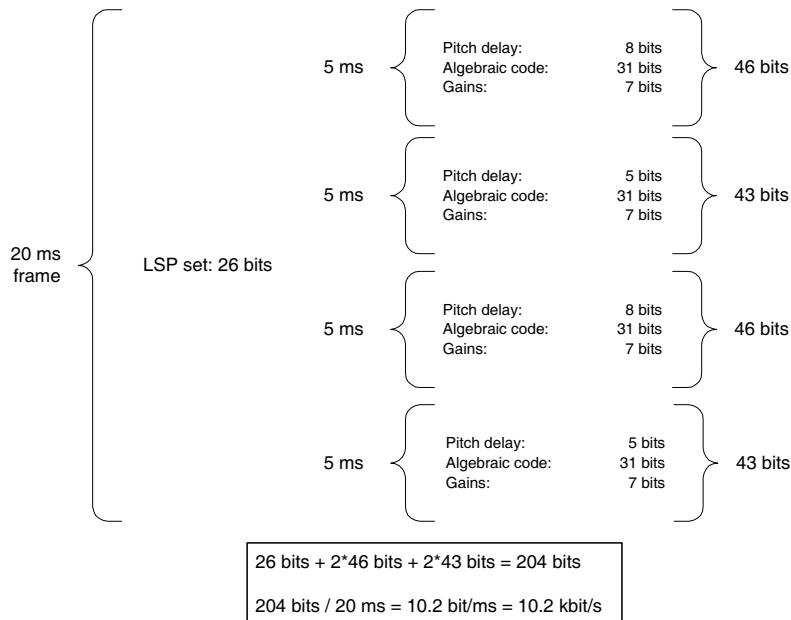


Figure 15.9: Contributors to the total bandwidth of 10.2 kbit/s AMR

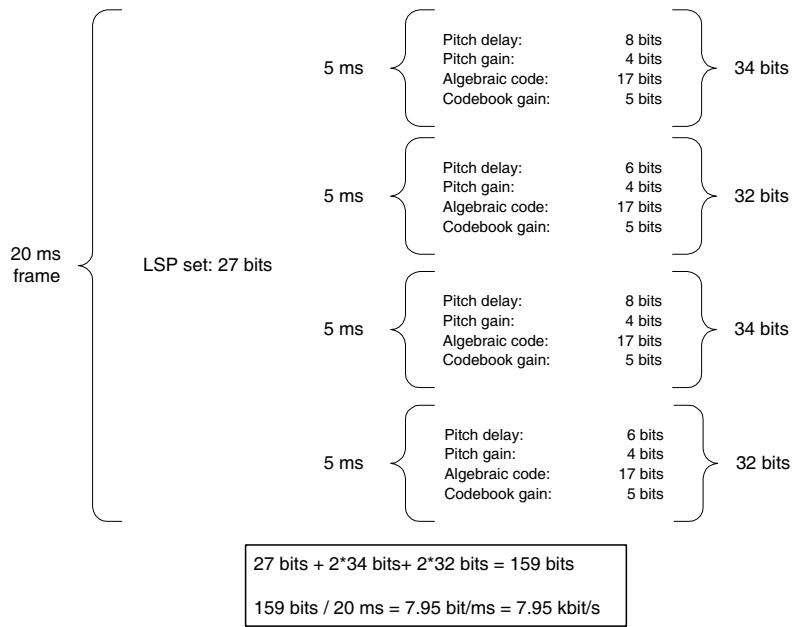


Figure 15.10: Contributors to the total bandwidth of 7.95 kbit/s AMR

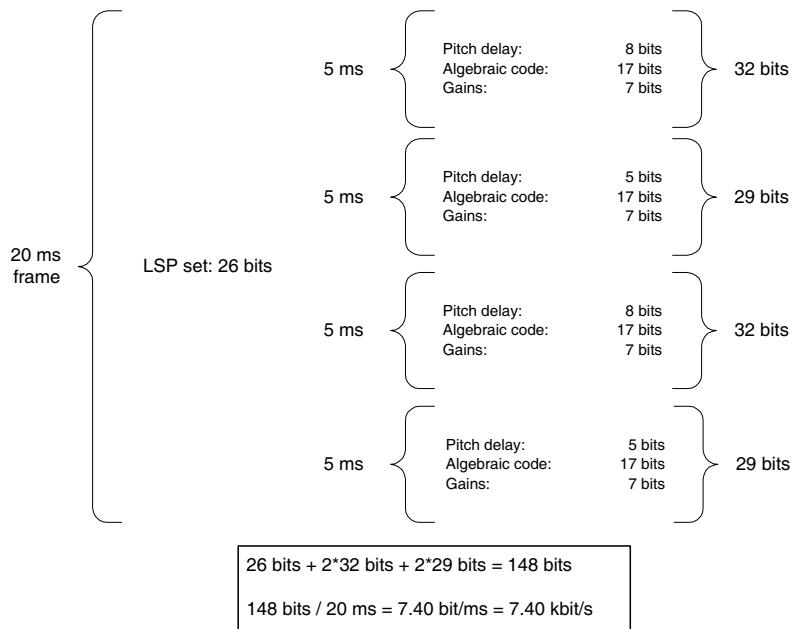


Figure 15.11: Contributors to the total bandwidth of 7.40 kbit/s AMR

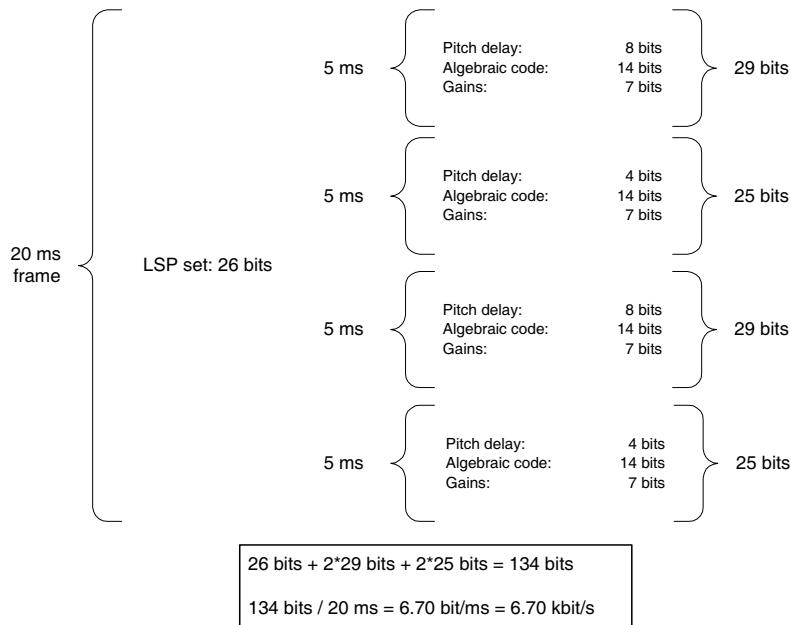


Figure 15.12: Contributors to the total bandwidth of 6.70 kbit/s AMR

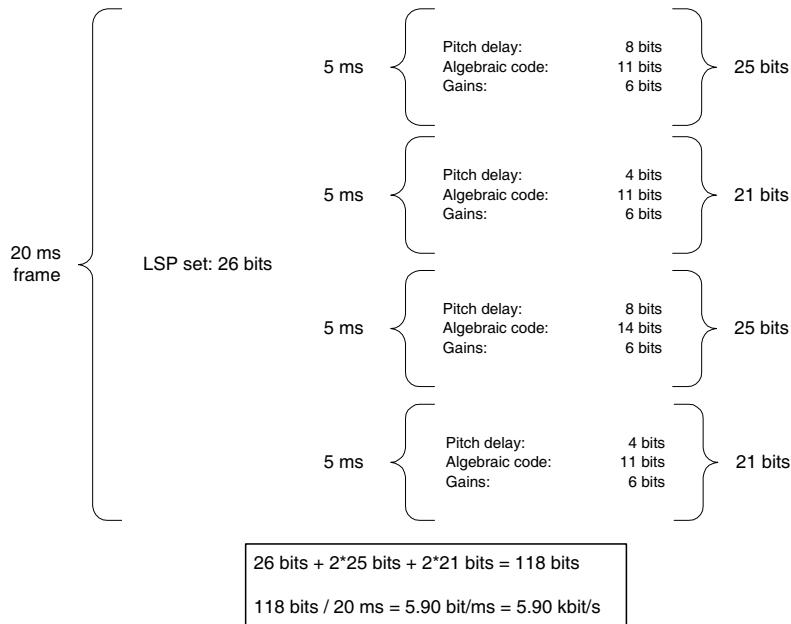


Figure 15.13: Contributors to the total bandwidth of 5.90 kbit/s AMR

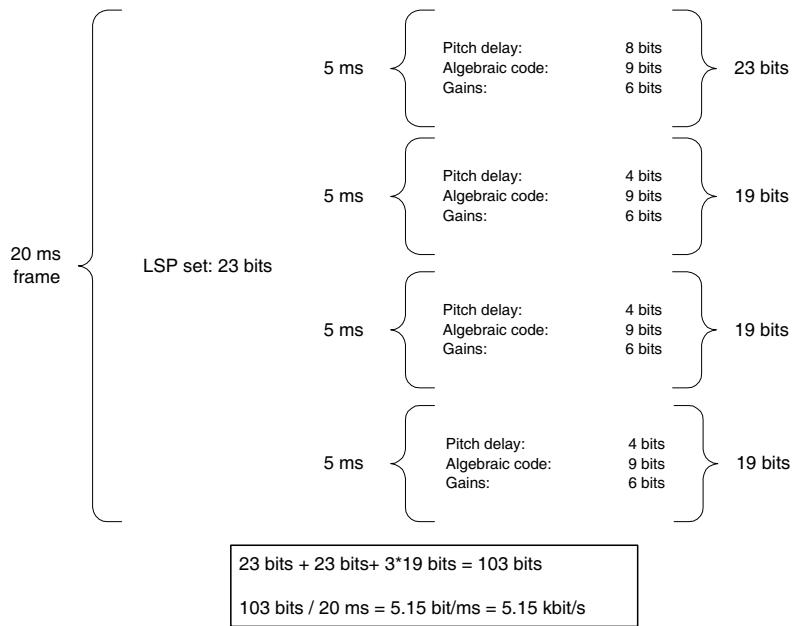


Figure 15.14: Contributors to the total bandwidth of 5.15 kbit/s AMR

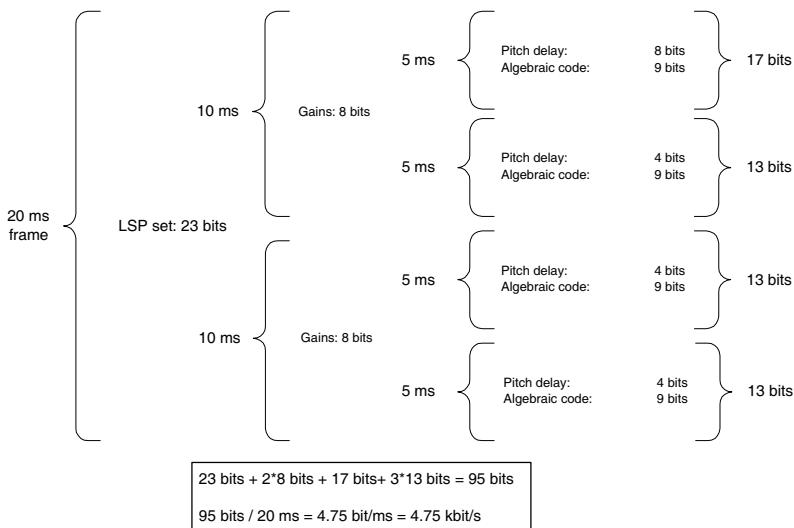


Figure 15.15: Contributors to the total bandwidth of 4.75 kbit/s AMR

SMV is a family of four codecs which are called Rate 1, Rate 1/2, Rate 1/4, and Rate 1/8. All of them work on 20 ms audio frames. We can find four modes in SMV: Mode 0 (premium mode), Mode 1 (standard mode), Mode 2 (economy mode), and Mode 3 (capacity-saving mode).

Every 20 ms audio frame is processed by one of the four codecs. The codec to be used is chosen based on the mode and on the characteristics of the audio frame at hand.

The Rate 1 and Rate 1/2 codecs use CELP to encode the audio frames (we saw in Section 15.1.4.3 that AMR also uses CELP). These two codecs use two codebooks, one adaptive and one fixed, and their respective gains to encode the audio signal.

The Rate 1/4 and Rate 1/8 codecs use random number generators to generate the LPC excitation. The Rate 1/4 codec provides a gain factor per 2 ms subframe while the Rate 1/8 codec provides a single gain factor per 20 ms frame. Table 15.1 shows how many bits each of the codecs use to encode a 20 ms frame and the resulting final bandwidth of the codec.

Table 15.1: SMV bandwidths

Codec	Bits per 20 ms frame	Resulting bandwidth (kbit/s)
Rate 1	171	8.55
Rate 1/2	80	4
Rate 1/4	40	2
Rate 1/8	16	0.8

15.2 Video Encoding

The building block of video encoding is image encoding. Given a set of encoded still images taken with a short interval between them, we can play them to create a video. We just need to show these images one after another quickly enough so that the human eye perceives this succession of images as a moving image.

Still image encoders divide the image to be encoded into colored dots called *pixels* (combination of the words “picture” and “element”). For a display working at its maximum resolution a pixel corresponds to the smallest element of the display which can be assigned a color. The resolution of a picture is given by the number of pixels used (horizontally and vertically) to encode it; a typical value is 640×480 .

Figure 15.16 shows a black-and-white picture encoded using a resolution of 10×10 pixels. Each of the pixels of this picture takes one of the following two values: black or white. So, we need one bit to encode the color of each pixel. Given that the picture has 10×10 pixels the size of the encoded picture will be 100 bits. Following the procedure we have just described, to encode a 640×480 picture with a color depth of 8 bits would yield 300 kilobytes.

If we want to have 25 frames per second (a value used by many television systems) using 300-kilobyte pictures we need a bandwidth of over 60 megabit/s. This is a very high figure that needs to be reduced somehow. There are a number of video compression techniques that make it possible to encode video using much lower bandwidth while still achieving acceptable quality. We describe the techniques used by H.263 (the mandatory video codec in the 3GPP IMS) in Section 15.2.2. Still, before dealing with H.263 we will mention other video codecs that are commonly used in different environments in Section 15.2.1.

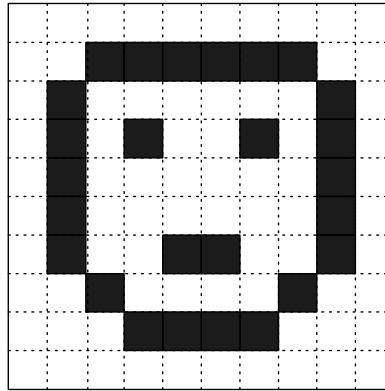


Figure 15.16: 10×10 black-and-white picture

15.2.1 Common Video Codecs

In this section we list a set of common video codecs and explain their relation with well-known video storage formats, such as DVD. In particular, we deal with the MPEG (Motion Picture Experts Group) and the H.261 video standards.

The MPEG video standards are used for both media storage and for videoconferencing. MPEG-1 (specified in ISO/IEC 11172 [171]) was developed to encode audio and video at rates of about 1.5 megabit/s. MPEG-1 defines an elementary stream as an audio or a video stream encoded following certain procedures. A videoconference with a single audio channel would consist of two elementary streams: audio and video. MPEG-1 also defines an encapsulation format that contains elementary streams and information on how to play these streams together (e.g., time stamps to perform lip synchronization).

Two user agents can exchange elementary streams and use external means (e.g., RTCP timestamps, described in Section 16.2.3) to do things such as media synchronization or exchange a single stream (using the encapsulation format) with all of the information needed to play all of the elementary streams in it (as described in [167]). Streams following the encapsulation format are referred to as system streams.

MPEG-2 (specified in ISO/IEC 13818 [172]) was developed to encode audio and video at rates higher than MPEG-1, providing higher quality as well. The VCD (Video CD) format is based on MPEG-1, while the SVCD (Super Video CD) format is based on MPEG-2. The format used in DVDs is based on MPEG-2.

MPEG-4 (specified in ISO/IEC 14496-1 [173]) uses audio-visual objects to describe audio-visual scenes, and it uses low bit rates. The DivX and XviD formats are based on MPEG-4.

MPEG defines three layers to encode audio: the higher the layer the higher the audio codec complexity. The third layer of MPEG is the well-known MP3 format, which is used to store music in different devices.

Another common video-encoding format is H.261 (ITU-T Recommendation H.261 [181]). H.261 is the video codec used by the H.320 (ITU-T Recommendation H.320 [184]) video-teleconferencing framework. This codec was designed to be used over ISDN lines and, therefore, produces bandwidths that are multiples of 64 kbit/s (which is the bandwidth provided by an ISDN channel), ranging from 64 to 1984 kbit/s.

15.2.2 H.263

The video codec H.263 (ITU-T Recommendation H.263 [183]) was developed for the H.324 (ITU-T Recommendation H.324 [190]) multimedia framework. H.263 was designed for low-bit-rate communications.

15.2.3 Image Encoding

H.263 supports five formats to encode images: sub-QCIF (Quarter Common Intermediate Format), QCIF, CIF, 4CIF, and 16CIF. All of these formats encode the color of the pixels using a luminance component and two chrominance components, but support different resolutions.

Images can be encoded using different color spaces. A common color space is RGB (red, green, blue), where the color of a pixel is represented by its red, green, and blue components. The color space used in H.263 consists of providing the luminance component and two chrominance components.

The luminance component is the black-and-white component of the image. That is, how dark (or light) each pixel is. The chrominance components are color difference components. They provide the color of a pixel in relation to the colors red and blue.

The image formats used by H.263 take advantage of the fact that the human eye is more sensitive to luminance information than to chrominance information and, therefore, only provide half-horizontal and half-vertical resolutions for chrominance. Table 15.2 shows the resolutions used for both values in each format.

Table 15.2: Luminance and chrominance resolution in CIF formats

Format	Luminance resolution	Chrominance resolution
Sub-QCIF	128×96	64×48
QCIF	176×144	88×72
CIF	352×288	176×144
4CIF	704×576	352×288
16CIF	1408×1152	704×576

As a result of providing less chrominance information than luminance information, blocks of 2×2 pixels share the same chrominance information. Figure 15.17 illustrates this point.

15.2.4 Temporal Correlation

H.263 supports two modes: *inter-coding* and *intra-coding*. In inter-coding mode the codec takes advantage of the temporal redundancy of the pictures. That is, it encodes a picture referencing those pictures that were encoded previously. In intra-coding mode, no references to other pictures are used to encode a particular picture.

Interpicture prediction is implemented in H.263 using motion estimation and compensation. The encoder finds corresponding blocks of pixels between frames and represents their relationship using motion vectors. These motion vectors are approximated using those motion vectors used in previous frames. The difference between this approximation (estimation) and the real motion vector is encoded separately (compensation).

L1	L2	L3	L4
C1	C1	C2	C2
L5	L6	L7	L8
C1	C1	C2	C2
L9	L10	L11	L12
C3	C3	C4	C4
L13	L14	L15	L16
C3	C3	C4	C4

Figure 15.17: Luminance and chrominance values for a block of pixels

15.2.5 Spatial Correlation

H.263 uses the DCT (Discrete Cosine Transform) to exploit spatial redundancy in the images and to achieve a compact representation of the coefficients that describe them. These coefficients are entropy-coded before being transmitted.

Entropy coding consists of performing lossless compression on the DCT coefficients. That is, using a reversible compression algorithm similar to those used to compress files in any personal computer.

15.3 Text Encoding

Text is already digital information, so there is no need to perform the analog-to-digital and digital-to-analog conversions that are necessary for audio and video. There are two types of text communications: instant messages and real-time text.

Instant messages convey a whole message, such as “How are you?”. That is, the sender types the message, edits it if necessary, and sends it. This way the receiver only receives the final version of the message. Chapter 21 looks at several ways of transporting instant messages.

Real-time text consists of transferring keystrokes instead of text. If the sender writes something and then deletes it to write something else, the receiver will see how these changes are performed. That is, the receiver receives letters and commands (e.g., carriage returns or delete characters) one by one as they are typed. The most common format for real-time text is T.140 (ITU-T Recommendation T.140 [182]).

15.4 Mandatory Codecs in the IMS

It would be desirable for all IMS terminals to support a common codec so that they could communicate directly with each other without the need for transcoders. Nevertheless, 3GPP and 3GPP2 could not agree on a common mandatory codec for their terminals. So, audio and video sessions between 3GPP and 3GPP2 IMS terminals usually involve transcoders. Still, both 3GPP and 3GPP2 specify which codecs must be supported by their IMS terminals.

All 3GPP IMS terminals support the AMR speech codec and the H.263 video codec (as specified in 3GPP TS 26.235 [48]). 3GPP IMS terminals providing wideband services support the AMR-WB audio codec and those terminals providing real-time text conversational services support T.140 (ITU-T Recommendation T.140 [182]).

Chapter 16

Media Transport

There are two types of media when it comes to media transport: media that tolerates a certain degree of packet loss and media that does not. Examples of the first type are audio and video and examples of the second type are web pages and instant messages. If a few of the packets transporting an audio stream get lost the recipient may notice a decrease in the quality of the sound, but will probably be able to understand anyway. On the other hand, an instant message under packet loss may change from “I will *not* come tomorrow” into “I will come tomorrow”.

The transport protocol for a particular media is chosen based on the type of media. Traditionally, TCP has been used to transport media reliably and UDP to transport media unreliably. Nevertheless, UDP is not suitable for transporting large amounts of data traffic because it lacks congestion control mechanisms (i.e., congestion control would need to be implemented at the application layer, but it is seldom implemented at all). To address this issue the IETF is developing the Datagram Congestion Control Protocol (DCCP).

16.1 Reliable Media Transport

There are two transport protocols that provide reliable delivery of user data: TCP and SCTP (Stream Control Transmission Protocol, specified in RFC 2960 [308]). TCP delivers a stream of bytes, while SCTP delivers messages.

TCP works best when the recipient of the data does not need to wait until all of the data have arrived in order to start processing them. An Internet browser is a good example of such a case. When the user requests a web page using HTTP the browser receives the contents of that web page over a TCP connection (or several TCP connections). The browser displays the information received so far to the user at every moment; it starts to draw pictures and display text. In this way the user does not need to wait until the whole web page is downloaded to start reading it.

On the other hand, when the application is interested in receiving all of the data at once, SCTP is a better choice. SCTP delivers messages rather than a stream of bytes to the application. A message may, for instance, contain an entire instant message (e.g., “Hello, Bob!”) or a file transferred between two users. Furthermore, SCTP provides features that are not present in TCP, such as better protection against DoS attacks, multi-homing, and multiple streams per SCTP association (connections are referred to as *associations* in SCTP).

Although SCTP provides some advantages over TCP for some applications, it still is not widely used. At present, TCP is by far the most widespread reliable transport protocol.

Many protocols establish TCP connections to carry control messages and user data. This is the case of HTTP, SMTP, and Telnet, to name a few. In these protocols the client sends a request to a server over a TCP connection and the server sends back the data requested by the client over the same connection. Clients establish this type of connection to well-known ports that identify the protocol in use. For instance, a web browser (which contains an HTTP client) sends requests to servers on port 80, which is the well-known HTTP port.

Still, SIP can be used to establish TCP connections or SCTP associations without using well-known ports. A session description describes the type of data to be transported and the protocol to be used. In this way, reliable connections are established in the same way as unreliable connections, such as audio and video streams. Figure 16.1 shows an SDP session description with a TCP connection that will transport images using T.38 (T.38 is the format used to send faxes). The *passive* direction attribute indicates that the recipient of this session description will be the *active* part and, so, will initiate the TCP connection.

```
v=0
o=Alice 2790844676 2867892807 IN IP4 192.0.0.1
s=I will send you a fax
c=IN IP4 192.0.0.1
t=0 0
m=image 20000 TCP t38
a=direction:passive
```

Figure 16.1: A TCP connection in SDP

16.2 Unreliable Media Transport

UDP is the transport protocol used to send media unreliably. Senders send UDP packets and hope that the recipient receives enough of them to be able to understand the message (e.g., to understand a voice message). Since lost packets are not retransmitted, applications introduce redundancy in the data to be transferred in order to be able to tolerate a certain level of packet loss. Some audio codecs, for instance, spread the information about every talk burst over several packets.

Although UDP is used widely it has a big disadvantage when it comes to transporting large amounts of data: it does not provide any congestion control mechanism. UDP senders do not slow down sending data even when the network is severely congested, making the congestion even worse.

The increase in the number of applications using UDP to send media, especially audio and video, made the IETF realize that we need an unreliable transport protocol that includes congestion control. One of the proposals was to create an SCTP extension for unreliable delivery. Nevertheless, the TCP-like congestion control mechanisms of SCTP do not suit some types of multimedia traffic. DCCP (specified in RFC 4340 [204]) supports different types of congestion control and has been designed with multimedia traffic in mind.

16.2.1 DCCP

DCCP is an unreliable transport protocol that provides connection establishment and termination as well as negotiation of congestion control algorithms.

DCCP connections are established with a three-way handshake, during which the characteristics of the connection are negotiated. In particular, DCCP peers negotiate the congestion control algorithm to be used. Congestion control algorithms are identified by their CCIDs (Congestion Control Identifiers). At present, there are three CCIDs to choose from: sender-based congestion control, TCP-like congestion control, and TFRC (TCP-Friendly Rate Control) congestion control.

TFRC (specified in RFC 3448 [159]) is especially suitable for multimedia traffic, because it avoids abrupt changes in the data rate while sharing the available bandwidth with TCP flows fairly. Nevertheless, DCCP is still a fairly new protocol and, so, is not widely used at present.

16.2.2 RTP

RTP (Real-time Transport Protocol, specified in RFC 3550 [301]) allows real-time media, such as audio and video, to be transported over unreliable transports, such as UDP and DCCP. It is always used in conjunction with RTCP (RTP Control Protocol), which provides quality-of-service statistics and information to perform inter-media synchronization.

The main purpose of RTP is to allow receivers to play out media at the proper pace, given that IP networks do not keep the timing relationship of the data being transported: that is, IP networks introduce jitter. If we send two IP packets to the same destination separated by 10 ms, nothing ensures that the second packet arrives at the destination exactly 10 ms after the first. The second packet may arrive right after the first, much later, or even before. Consequently, receivers cannot rely on the arrival times of the packets to recover the timing relationship of the media. They use RTP timestamps for this purpose.

Receivers place incoming RTP packets in a buffer according to their timestamps and start playing them. If a packet with a particular timestamp needs to be played and still has not arrived, the receiver uses interpolation techniques to fill the gap (e.g., in the case of audio it may play the last audio packet for a longer time). If this packet is received afterwards, it is simply discarded. Figure 16.2 shows a few packets in a buffer. The packet with timestamp 40 has not arrived yet. If it does not arrive by the time it needs to be played, it will be discarded when it eventually arrives.

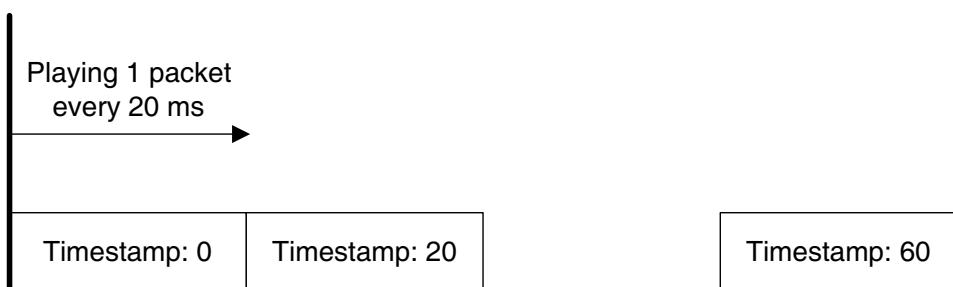


Figure 16.2: Receiver's buffer

The receiver in Figure 16.2 needs to make an important decision: when to start playing media to the user. If it starts as soon as the packet with timestamp 0 is received, it runs the risk that the packet with timestamp 20 might not arrive in time to be played (this would decrease

the quality of the media being played out). On the other hand, if the receiver waits many seconds before playing the media, the delay would be so big that the users would be unable to maintain a normal conversation, and it would have to implement a much larger buffer.

Different implementations use different parameters to decide the lengths of their buffers: long buffers cause long delays but are good quality while short buffers cause short delays but are poor quality. Let us use an example to discover what would be a good value for the buffer of a particular receiver. The graph in Figure 16.3 shows the delay experienced by packets sent from a sender to a receiver. In this example, most of the packets experience a delay of around 50 ms, some experience smaller delays, and a few experience much larger delays (lost packets have an infinite delay). A good tradeoff for the receiver would be to start playing packets 100 ms after they are sent. In this way, only a few packets that appear in the tail of the distribution (which have a too long delay) would be discarded when they arrive.

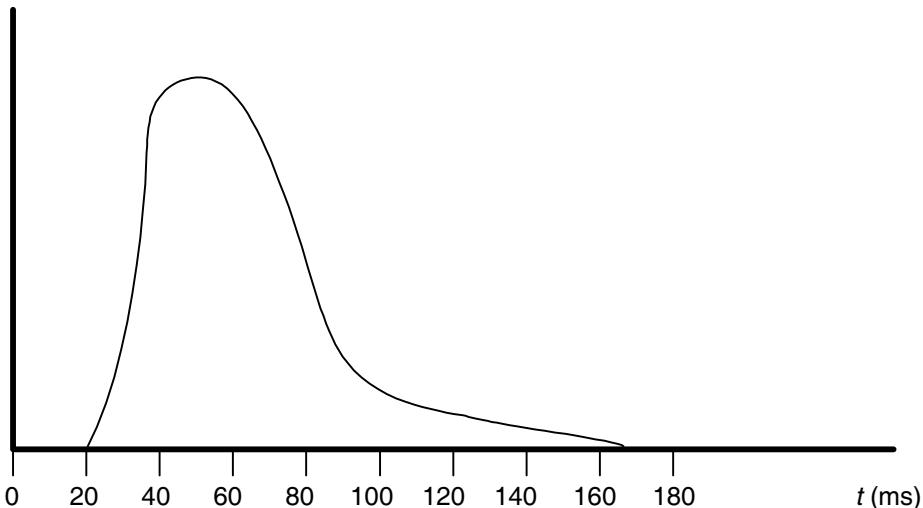


Figure 16.3: Packet arrival distribution

In addition to timestamps, RTP packets carry sequence numbers. Recipients use them to figure out how many packets are lost in the network during a transmission. If the network drops too many packets at a particular time, peers may decide to use a different codec, one that provides better quality under heavy packet loss (i.e., a codec with more redundancy).

RTP packets also carry binary sender identifiers and the payload type. Binary sender identifiers are used in conferences to identify the current speaker, and the payload type identifies the encoding and transport format of the data carried in the RTP packet. Payload types are numeric values that identify a particular codec and are typically negotiated (or simply assigned) using a session description protocol. There are two kinds of payload types: static and dynamic.

Static payload types are numbers that always correspond to the same codec. For example, payload type 0 corresponds to the G.711 μ -law audio codec (RFC 3551 [300] defines some audio and video static payload types) and payload type 31 corresponds to the video codec H.261.

Dynamic payload types are typically negotiated using the offer/answer model. They identify a particular codec within a particular session. The SDP session description in Figure 16.4 contains an audio stream that can be encoded using G.711 μ -law (static payload type 0) audio codec or using a 16-bit linear stereo codec sampled at 16 kHz (dynamic payload type 98, specified in the `a=rtpmap` line). The recipient of this audio stream will receive RTP packets whose payload types will be 0 or 98 and, based on this session description, will decode them.

```
v=0
o=Alice 2790844676 2867892807 IN IP4 192.0.0.1
s=Let's talk
c=IN IP4 192.0.0.1
t=0 0
m=audio 20000 RTP/AVP 0 98
a=rtpmap:98 L16/16000/2
```

Figure 16.4: SDP with static and dynamic payload types

Figure 16.5 shows all of the protocol headers of an RTP packet carrying an audio sample. The transport protocol used in this example is UDP.



Figure 16.5: Protocol headers of an RTP packet

16.2.3 RTCP

RTCP is a protocol that is always used together with RTP. It provides quality-of-service statistics, information to perform inter-media synchronization, and mappings between RTP binary sender identifiers and human-readable names. RTCP messages are sent by both RTP senders and RTP receivers.

To develop quality-of-service statistics, RTP senders report (using RTCP) the number of RTP packets they have sent to the network and RTP receivers report the number of packets they have received. In this way it is easy to deduce the packet loss rate that the session is experiencing.

RTP senders use RTCP to provide a mapping between the timestamps of their media streams and a wall-clock. In this way, receivers can synchronize the play-out of different media streams. References to a wall-clock are needed because clock frequencies used for the timestamps of different media streams may be different and the initial value of the timestamps is random. So, by only inspecting the timestamps of two media streams, it is impossible to determine which samples should be played at the same time. A common example of this type of inter-media synchronization is lip sync; that is, audio–video synchronization. Figure 16.6 shows how RTCP mappings help the receiver perform lip sync.

RTCP messages also provide mappings between RTP binary sender identifiers and human-readable names. This is particularly useful in conferences where the media from

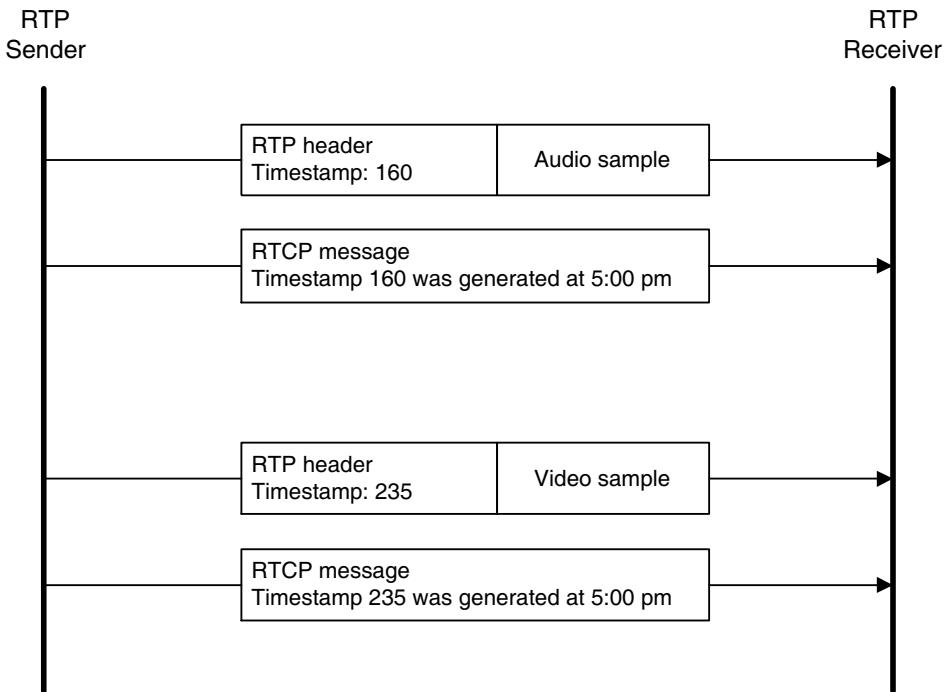


Figure 16.6: Lip synch using RTCP

all the participants is received at the same transport address. In this way the users can be informed about who is speaking at every moment.

When SDP is used, RTP packets are normally sent to a port with an even number and RTCP messages are sent to the consecutive odd port. For instance, a user agent that generates the session description in Figure 16.4 will receive RTP packets carrying audio samples on UDP port 20000 and RTCP packets related to the audio stream on UDP port 20001.

16.2.4 SRTP

SRTP (specified in RFC 3711 [84]) provides confidentiality, message authentication, and replay protection to RTP and RTCP traffic. Figure 16.7 shows which portions of an RTP packet are authenticated and which are encrypted.

Peers using SRTP to exchange media use a key management protocol (as described in Section 11.8) to come up with a master key, which is used to generate session keys. Session keys are typically refreshed periodically so that attackers do not have access to large amounts of traffic encrypted under the same key.

16.3 Media Transport in the IMS

The IMS uses RTP over UDP to transport media unreliable. DCCP may be used in the future, but at present it is not mature or widespread enough for the IMS.

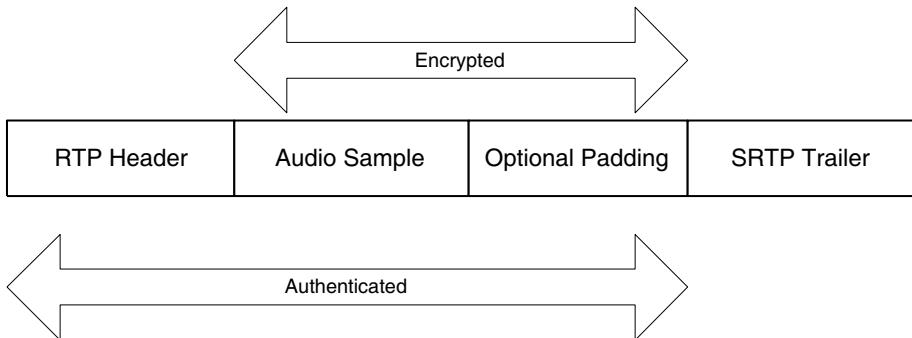


Figure 16.7: SRTP protection

Regarding security the IMS does not provide any kind of security at the media level. It is assumed that the traffic on the radio access is encrypted at lower layers and that the core IMS network is trustworthy. So, SRTP is not supported.

When it comes to reliable transport protocols the natural choice would be to use TCP. It has been a stable protocol for many years and is supported virtually everywhere. In addition to this there are TCP extensions specifically designed to make its flow control mechanisms more suitable for radio links (e.g., RFC 3522 [211] defines the Eifel detection algorithm).

Part IV

Building Services with the IMS

Now that we have introduced all of the technologies used in the IMS (see Parts II and III), we are ready to describe how we can use those technologies to achieve the main goal of IMS: providing services.

In previous sections we looked at how the IMS provides some services (e.g., multimedia sessions). Other services do not need extra standardization and can be provided with all of the tools we have described (e.g., a voicemail service). In addition, there are services that require extra standardization. It is these services that Part IV focuses on.

We have chosen some of the most significant services that will be initially provided by the IMS: presence, instant messaging, and Push-to-talk. Still, the reader should keep in mind that this is not, by any means, an exhaustive list of IMS services. Many other new and innovative services will be developed in the future using the IMS infrastructure. For instance, 3GPP and the IETF are currently working on standards for conferencing. However, we have decided not to include them in this part because they are still at an early stage of development.

Chapter 17

Service Configuration on the Internet

Many services require a mechanism for allowing users to manage their service configuration. For example, a presence server requires presentities (users) to authorize which watchers can see their presence information. A Push-to-talk over Cellular (PoC) service requires users to create and manage groups. Likewise a conference may require users to configure a dial-out or dial-in list of participants, their privileges (who can speak or who can send or receive which media type), and so on. All of these use cases share many commonalities: a user has to perform non-real-time operations on a server to manipulate one or more documents that configure or personalize their instance of the service.

Usually the user creates a configuration document locally in their terminal and then uploads it to a server. Sometimes, the user just needs to make a small change to an existing document, so it is not worth uploading the complete document. Instead, it is desireable to have the capability to update part of the document. In some other cases the user changes their usual terminal and uses a different one, so they may first need to download a fresh copy of the current configuration document, make some changes, and upload it (either complete or a part of it) to the server.

Typically, configuration documents are highly structured. Owing to this, the trend nowadays is to use the Extensible Markup Language (XML) (specified by the World Wide Web Consortium in the W3C recommendation XML 1.1 [93]), for formatting documents that personalize the instance of the service.

So, we know that configuration documents are effectively XML documents, but how are they sent to and received from the server that stores them? The XML Configuration Access Protocol (XCAP) solves this vacuum.

17.1 The XML Configuration Access Protocol (XCAP)

The problem of document management can be depicted in its most simplified way as in Figure 17.1. A user creates an XML document in his computer and wants to upload it to a server. The server will use the document at a later time to produce a personalized instance of the service.

The problem to be solved requires designing a protocol that allows such an upload procedure. HTTP (specified in RFC 2616 [144]) provides a good baseline, since it provides

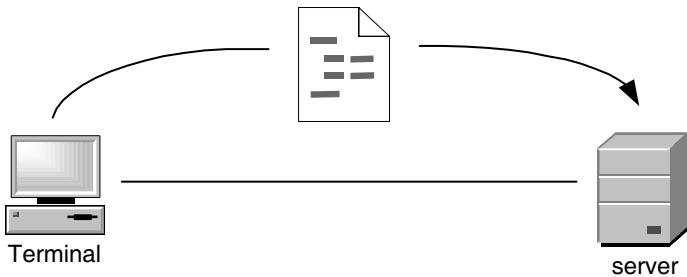


Figure 17.1: Overview of Document Management

the POST and PUT methods for transferring files from the client to the server. It also contains a GET method for downloading a document from the server. However, there are additional requirements that prevent the usage of HTTP. For example, on many occasions users just need to modify a small piece of an existing XML document. For instance, when a user wants to add a “buddy” to his list of friends in the presence service: the user just wants to edit a few bytes of a potentially large document. It is not worth uploading the whole XML document again, since the document is mostly unchanged, just the modified content. HTTP alone does not offer that functionality.

The IETF decided then to develop a set of conventions and rules for using HTTP to upload and download complete or portions of XML documents to and from a server. This resulted in the creation of XCAP (specified in RFC 4825 [277]). XCAP is, therefore, not a new protocol, but a set of conventions for using HTTP for managing remotely stored XML documents. Figure 17.2 shows a schematic representation of the protocol stack used by XCAP.

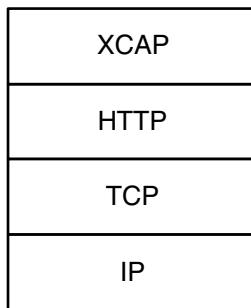


Figure 17.2: The XCAP protocol stack

XCAP provides a client with the means to read, write, and modify XML application configuration data remotely stored on a server. This includes, for example, modifying the list of users in a presence list, authorization policies (e.g., a list of authorized watchers) or the list of participants in a conference. XCAP does not control the user interface (e.g., the graphical representation of the list), rather, it focuses on the data structure.

XCAP defines conventions that map XML documents and their components (e.g., elements, attributes) to HTTP URIs. Therefore, XCAP provides a unique way to represent an XML document or a component with an HTTP URI. It also defines the rules that govern how

modification of one resource affects another. This means that XCAP allows users to modify an XML document, but still the resulting document has to be compliant with the original XML schema. In addition, XCAP also defines the basic authorization policies associated with access to resources.

XCAP implements a set of operations that are mapped to regular HTTP 1.1 methods. The operations also set some HTTP headers to a particular value. XCAP provides the client with the operations that can manipulate the whole XML document, an arbitrary XML element of the document, or an arbitrary XML attribute of an element. For each of these levels, XCAP provides the means for creating, deleting, modifying, replacing, and fetching. So combining all together, XCAP provides the means to create, delete, modify, replace or fetch a complete XML document, an element, or an attribute.

Figure 17.3 shows an example of an XCAP request in which Alice is sending an HTTP PUT request to create a new presence list. We provide details of how to manipulate presence lists in Chapter 19. For the time being, just notice that XCAP is merely HTTP and XML with some additional rules. We are going to describe the basics of XML in Section 17.2.

```
PUT /pr-lists/users/sip:alice@example.com/friends.xml HTTP/1.1
Host: xcap.example.com
Content-Type: application/resource-lists+xml
Content-Length: 460

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists
    xmlns="urn:ietf:params:xml:ns:resource-lists">
    <list name="family" uri="sip:family@example.com">
        <entry name="Bob" uri="sip:bob@home1.com">
            <display-name>Bob</display-name>
        </entry>
        <entry name="Cynthia" uri="sip:cynthia@example.com">
            <display-name>Cynthia</display-name>
        </entry>
    </list>
</resource-lists>
```

Figure 17.3: Example of an XCAP operation

XCAP defines two functional elements: an XCAP client and an XCAP server. They are depicted in Figure 17.4. An XCAP client is an HTTP 1.1 compliant client that supports the rules and conventions specified by XCAP; it sends HTTP requests and receives HTTP responses. An XCAP server is an HTTP 1.1 compliant server that also supports the rules and conventions specified by XCAP; it receives HTTP requests and generates HTTP responses.

17.1.1 XCAP Application Usage

As explained in the previous section, XCAP is a generic protocol that can be used for a number of purposes related to application data configuration of XML documents stored in a server. We have also seen that there are several applications that utilize XCAP. For example, in the presence service, presentities use XCAP to control whether watchers can see all

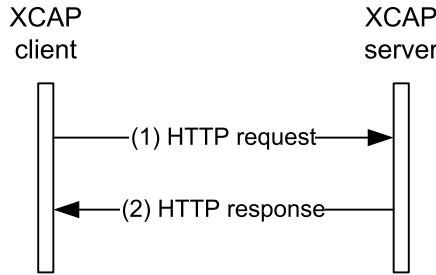


Figure 17.4: XCAP functional elements

or part of the presence information (watcher authorization is described in greater detail in Section 19.14). In a centralized conference service, the creator of a dial-out conference can use XCAP to configure the list of participants of the conference. In general, whenever there is a need to configure configuration application data, such as a list remotely stored in XML format, XCAP is a good protocol to fulfill these needs.

Owing to this versatility XCAP introduces the concept of an *application usage*. An application usage defines how a particular application uses XCAP to interact with the XCAP server. For instance, each of the previously mentioned applications of XCAP (presence list management, authorization policies, and conference list management) constitutes an application usage on its own. Each application usage is identified by an AUID (Application Unique ID) that uniquely identifies the application usage. There are two types of AUIDs: standard (i.e., application usages standardized in the IETF) and vendor-proprietary (i.e., private application usages).

The IETF has defined a number of XCAP application usages related to the presence service.

XCAP application usage for resource lists. Specified in RFC 4826 [273], provides the means to manipulate resource lists that are typically used as presence lists.

XCAP application usage for presence authorization. Defined in RFC 5025 [276], allows a client to specify presence authorization rules, i.e., the set of rules that grants certain watchers the permissions to access certain subsets of the presentity's presence information.

XCAP application usage for manipulating presence documents. Specified in RFC 4827 [175], allows users to perform *hard state manipulation* of presence data, i.e., set the presence status by configuration (i.e., using XCAP), as opposed to the *soft state manipulation* performed through the more common presence publication, i.e., using the SIP PUBLISH method.

17.2 An Overview of XML

XML documents constitute a very important aspect for configuring applications. Usually, the application configuration data are stored in XML format, therefore, XML documents (or portions of them) are transported between the client and the server. While there is extensive literature on XML, we provide a high-level overview that allows readers to better understand the protocol used for manipulating XML documents.

XML documents are text-based documents and, thus, are human-readable. XML documents contain a structured representation of data, but the document itself does not do anything. Therefore, an XML document merely provides the means to represent structured data.

The XML document in Figure 17.5 contains the representation of the data pertaining to the second edition of this book, which we assume is stored in a file named `ims.xml`. Let us use this example to further illustrate a few concepts. The first line of the document is the XML declaration that defines the XML version and the encoding used in the document. Then, the data that constitute the tree follow through the following lines. Each node in the tree is called an *XML element*. XML elements start with the name of the element enclosed in angle brackets, e.g. `<book>`, and terminates with a closing tag that contains a slash '/' and the name of the element, e.g. `</book>`.

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="urn:org:miguel:book"
      isbn="0470018186" edition="2">
  <title>The 3G IP Multimedia Subsystem (IMS)</title>
  <subtitle>Merging the Internet and the cellular worlds</subtitle>
  <author id="1">Gonzalo Camarillo</author>
  <author id="2">Miguel A. Garcia-Martin</author>
  <publisher>John Wiley and Sons, Ltd</publisher>
</book>
```

Figure 17.5: Example of an XML document: `ims.xml`

XML elements can contain other child elements, processing instructions, namespace declarations, comments, and text nodes. In the example in Figure 17.5, the `book` element contains the `title`, `subtitle`, and so on, child elements.

XML elements usually contain a text node that represents a *value*. In the example in Figure 17.5, the value of the `title` element is “The 3G IP Multimedia Subsystem (IMS)”. XML elements can be also empty, in which case, a compact notation can indicate the beginning and end tags of the empty element by including a slash '/' at the end of the element name. For example, `<test/>` is an empty element.

Last, XML elements can contain *attributes* that further characterize the element, typically by defining its metadata. In Figure 17.5, the `book` element contains three attributes: `xmlns`, `isbn`, and `edition`. Unlike elements, attributes cannot be empty and can only appear once in a given element.

XML is extensible, so new elements and attributes can be added whenever they are needed. This is, perhaps, the most important property of XML.

An XML document is said to be *well formed* when it meets the basic constraints for all XML documents. For example, each open tag has a closing tag (except for the compact notation of empty elements); the names of the attributes are unique; elements are properly nested; attributes values must be quoted, etc.

Sometimes XML documents need to be drafted according to a predefined structure that provides additional constraints of the data. In the example of Figure 17.5, there should not be two `title` elements, but there can be several `author` elements. All of these constraints, which are specific to the type of data that the XML document represents, are typically defined in separate additional documents, for example, in Document Type Definition (DTD)

documents, XML schema documents or RelaxNG documents. An XML document is said to be *valid* when it meets the constraints defined in the DTD, XML schema, RelaxNG, or any other document that defines the constraints.

17.2.1 XML Namespaces

When XML designers create XML documents, they need to give names to the XML elements and attributes of the document. It is easy to foresee that sooner or later there will be conflicts in the names given to XML elements and attributes, e.g., two designers could create the book element in different documents, each one with a different structure and syntax. Furthermore, if these two book elements were part of the same XML document as sibling nodes, then it would not be possible to determine which is which.

To solve this problem, XML introduces the concept of *namespaces*, which are specified in the W3C Recommendation “Namespaces in XML 1.1 (Second Edition)” [92]. A namespace is an abstract space where names pertaining to XML elements and attributes belong to. By making namespaces globally unique, it is possible to differentiate, e.g., two book elements that belong to two different XML schemas. So, the namespace provides the XML document with a context where the document makes sense.

Namespaces are identified by International Resource Identifiers (IRI), specified in RFC 3987 [125]. In the case of IETF documents, IRIs are usually URNs, specified in RFC 3986 [86], and they are registered under the tree urn:ietf:params:xml:ns. A URN is a persistent name that identifies a resource independently of its location. An XML document must indicate at least one namespace where the elements, attributes, etc. belong, but it is also possible, and it is actually very common, to create an XML document that contains elements and attributes belonging to different namespaces. Usually, a namespace declaration is included in the root element of the document, indicating all of the namespaces used in the document. The declaration is made in `xmlns` attributes that modify the root element. The namespace declaration typically contains the default namespace and additional namespaces used throughout the document. The default namespace applies to any unprefixed element. Additional namespaces are mapped to a prefix, so it is easier to distinguish those elements that belong to other non-default namespaces.

Let us illustrate the concept with the example of Figure 17.6, which is an extension of our XML document for representing books. The document now defines two namespaces with the “`xmlns`” attributes of the book element. The default namespace is “urn:org:miguel:book”. Any element that is not prepended by a prefix belongs to that namespace. A second namespace named “urn:org:miguel:classification” is declared. This second namespace is mapped to the prefix “`c1`”, so any XML element (and attributes of it) prefixed with “`c1`” belongs to that namespace. So, the example shows the elements `class`, `technical`, `internet`, `telecom`, and `wireless` that belong to the “urn:org:miguel:classification” namespace. We can also see that some of these elements are, in fact, empty XML elements, since the element tag serves as both opening and closing tag.

17.3 HTTP URIs that Identify XCAP Resources

We have discussed earlier that XCAP is used to manage remotely stored XML documents. XCAP is able to manage XML documents, XML elements, XML element values, and XML attributes. Each of them is considered a *resource* and it is identified with a URI. Actually,

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="urn:org:miguel:book"
      xmlns:cl="urn:org:miguel:classification"
      isbn="0470018186" edition="2">
  <title>The 3G IP Multimedia Subsystem</title>
  <subtitle>Merging the Internet and the cellular worlds</subtitle>
  <author id="1">Gonzalo Camarillo</author>
  <author id="2">Miguel A. Garcia-Martin</author>
  <publisher>John Wiley and Sons, Ltd</publisher>
  <cl:class>
    <cl:technical true="yes"/>
    <cl:internet true="yes"/>
    <cl:telecom true="yes"/>
    <cl:wireless true="yes"/>
  </cl:class>
</book>
```

Figure 17.6: Various namespaces in an XML document

since XCAP is not a new protocol, but a set of conventions to use HTTP for managing XML documents, there are no XCAP URIs, but rather, HTTP URIs that represent XCAP resources. Let us take a closer look at the mechanism that XCAP utilizes to identify resources.

Figure 17.7 shows a schematic representation of an HTTP URI that represents an XCAP resource, while Figure 17.8 shows an example of it. Actually, the example in Figure 17.8 represents the `id` attribute whose value is “2” of the `<author>` element that we saw in Figure 17.5.

```
XCAP resource = XCAP root
  + document selector
  [+ node selector separator]
  [+ node selector]
  [+ query]
```

Figure 17.7: Construction of HTTP URIs that represent XCAP resources

`http://xcap.example.com/root/bibliography/users/sip:miguel@example.com/ims.xml/~/book/author[@id="2"]`

Figure 17.8: An HTTP URI that represents an XCAP resource

The HTTP URI represented in Figure 17.8 starts with the HTTP URI scheme, “`http://`”, followed by the hostname of the server, “`xcap.example.com`”, and XCAP root locator, “`/root`”. All three form the *XCAP root URI* that identifies the hierarchy where XCAP services are located in the HTTP server. So, in the example, the XCAP root URI indicates an HTTP protocol operation on a server called “`xcap.example.com`” on a (potentially virtual) directory called “`/root`”.

The XCAP root URI is followed by a *document selector*, which starts with the application usage to which the document pertains, in the example, the “`bibliography`”

XCAP application usage. Then, there are one of two possible subtrees, which identify either the “*global*” subtree or the “*users*” subtree, and are identified with the literals *global* or *users*, respectively. The *global* subtree contains all of the documents where data is set by all users. The “*users*” subtree contains documents that apply to a particular user. The name of the user follows the *users* subtree, so, the example in Figure 17.8 refers to user “`sip:miguel@example.com`”, so the URI is referring to a document owned by that user. Then the actual XML document follows, in the example, “`ims.xml`”. To summarize: Figure 17.8 is selecting an XML document named “`ims.xml`” of the “*bibliography*” application usage; the “`ims.xml`” document is located in the user directory “`sip:miguel@example.com`”.

The *document selector* is followed by double tilde characters “`~~`”, which are called the *node selector separators* and are used to separate the *document selector* from the *node selector* which follows. The *node selector* identifies an XML element, attribute, etc. within the selected document. In the example in Figure 17.8 the *node selector* points to the `book` element and its child `author` element whose attribute “`id`” has the value “`2`”. This is denoted as “`/book/author[@id='2']`”.

Last, an optional *query* component can be concatenated to the HTTP URI that represents an XCAP resource. This is used, for example, when XML namespaces have to be included in the URI. In most cases, the *query* component is not needed because the default namespace for the application usage is assumed.

Usually HTTP URIs that represent XCAP resources are included in the *Request-URI* of HTTP requests. The encoding rules of the *Request-URI* field in an HTTP request does not allow the presence of some characters, such as square brackets. Owing to this, special characters are percent-encoded in *Request-URIs*, so a ‘[’ character is encoded as a ‘%5b’, a ‘]’ character is encoded as ‘%5d’, and the double quotes ‘”’ as ‘%22’. Therefore, when the example of an HTTP URI that we saw in Figure 17.8 is included in a *Request-URI*, it yields the percent-encoded HTTP URI of Figure 17.9.

```
http://xcap.example.com/root/bibliography/users/sip:miguel@example.com
/ims.xml/~~/book/author%5b@id=%222%22%5d
```

Figure 17.9: A percent-encoded HTTP URI representing an XCAP resource

17.4 XCAP Operations

As we indicated earlier, XCAP allows an XCAP client to do different types of operations in a remote XML document. XCAP operations are mapped to HTTP requests, whose *Request-URI* values indicate the resource (document, element, attribute) that are to be created, deleted, replaced, modified, or fetched. Let us take a deeper look at these operations and how they are mapped to HTTP 1.1 requests.

17.4.1 Create or Replace Operations

To create a new XML document in a remote server or to replace an existing one, an XCAP client invokes the HTTP PUT method. The *Request-URI* in the HTTP PUT request identifies the XML document that is to be created (if it previously did not exist) or replaced (if it existed). The XML document to be created or replaced is attached to the request as a MIME

body, and it is identified by a Content-Type header field with the value defined by the application usage. Figure 17.10 shows an example of an XCAP operation that creates or replaces an “ims.xml” document.

```
PUT /root/bibliography/users/sip:miguel@example.com/ims.xml HTTP/1.1
Host: xcap.example.com
Content-Type: application/bibliography+xml

<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="urn:org:miguel:book"
      isbn="0470018186" edition="2">
  <title>The 3G IP Multimedia Subsystem</title>
  <subtitle>Merging the Internet and the cellular worlds</subtitle>
  <author id="1">Gonzalo Camarillo</author>
  <author id="2">Miguel A. Garcia-Martin</author>
  <publisher>John Wiley and Sons, Ltd</publisher>
</book>
```

Figure 17.10: Creating or replacing a complete XML document

If the XCAP client wishes to create or replace a single element or attribute of an existing XML document, the operation is similar: the XCAP client invokes the HTTP PUT method with a *Request-URI* that points to the XML element or attribute that is to be created or modified. The Content-Type header field is set to either “application/xcap-el+xml” or “application/xcap-att+xml”, depending on whether the operation affects an element or an attribute, respectively. The PUT request then contains an XML body that includes the element or attribute to be created or replaced. For example, Figure 17.11 inserts a `keywords` element as a child element of `book`, in the “ims.xml” document.

```
PUT /root/bibliography/users/sip:miguel@example.com/ims.xml/
    ~/book/keywords HTTP/1.1
Host: xcap.example.com
Content-Type: application/xcap-el+xml

<keywords>
  <keyword>IMS</keyword>
  <keyword>multimedia</keyword>
  <keyword>3G</keyword>
</keywords>
```

Figure 17.11: Adding an element to an XML document

Since both create and replace operations are mapped onto the same HTTP request, there must be a way to differentiate them. There is, and it is very simple: if the node selector does not match an existing element, then it is a create operation; if the node selector selects an existing element within the document, then it is a replace operation.

17.4.2 Delete Operations

To delete an XML document, element, or attribute in a remote server, an XCAP client creates a DELETE HTTP request. The *Request-URI* uniquely selects the document, element, or attribute to be deleted. Upon reception of the DELETE method, the server deletes the document or attribute. If it is an element, it deletes the element with all of its attributes and child elements. The resulting document must still be in conformance with the application usage, i.e., it must be compliant with the XML Schema that determines which elements and attributes are mandatory and optional and further constraints, otherwise, the server will not honor the delete request. Figure 17.12 shows an example of a delete operation that removes the “ims.xml” document from the server.

```
DELETE /root/bibliography/users/sip:miguel@example.com/ims.xml HTTP/1.1
Host: xcap.example.com
```

Figure 17.12: Deleting an XML document

17.4.3 Fetching Operations

There are many cases in which an XCAP client needs to fetch an existing document, element, or attribute, that is remotely stored. For example, if a user is using a new terminal, he most likely does not have any of the XML documents that he created earlier with a different terminal. In this case, before he manipulates his data, he must first fetch existing documents from the server.

To fetch a document, element, or attribute, the client creates an HTTP GET request. This requires that the XCAP client has knowledge of the name of the XML document. The *Request-URI* selects the document, element, or attribute to be fetched. The server will reply in a 200 (OK) response that contains a MIME body. If a full XML document was requested, the response contains the document. If an element or attribute was requested, the response contains the requested element or attribute, properly identified with a Content-Type header.

17.5 Entity Tags and Conditional Operations

The concept of *entity tags* is not new to XCAP, since it is inherited from HTTP, so we need to go back to HTTP for a proper description. An *entity tag* is an opaque string of characters that is associated to the contents of a resource, for example, a web page. If the resource (e.g., the web page) changes its contents, e.g., due to an update, then the HTTP server creates a new entity tag for such resource¹⁵. So, we can consider an entity tag as a sort of fingerprint of a resource.

Entity tags are transported in ETag header fields that are part of HTTP responses. Figure 17.13 depicts a couple of HTTP requests where the corresponding responses contain an ETag header field. A client invokes the HTTP GET method to request a given resource (1). The HTTP server answers with a 200 (OK) response (2) that contains the requested content, as a MIME body, and an ETag header field whose value identifies the served version of the content. When the content changes, perhaps due to some external action, the server assigns a new entity tag. If, later, the client (or any other client) requests the same resource (3),

¹⁵Creation of entity tags is optional for HTTP servers, however, it is mandatory for XCAP servers

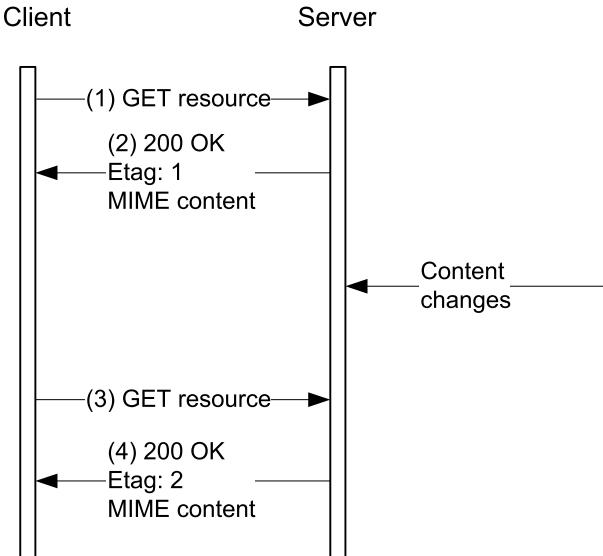


Figure 17.13: Entity tags in HTTP

the server returns the new value of the entity tag in the ETag header field of the 200 (OK) response (4), along with the MIME body of the content.

Entity tags are key to perform conditional HTTP requests, and by extension, conditional XCAP operations. Let us take a look at the need for conditional HTTP requests with the help of Figure 17.14. Assume the client depicted in Figure 17.13, which has issued the GET requests depicted in Figure 17.13. The client has stored a cached copy of the content identified by the *Request-URI* of the last GET request. The entity tag associated to that version of the content is “2”, since this was the value returned in the ETag header of the last 200 (OK) response. Now the clients wants to retrieve the content identified by the same *Request-URI*, if it has been updated with respect to the cached version. Otherwise, it would be a waste of bandwidth to download the same content. So, according to Figure 17.14, the client sends a GET request (1) indicating the URI of the resource. The GET request also includes an *If-None-Match* header field that contains the entity tag of the cached copy of the content. If the resource has not been modified with respect to that value of the entity tag, the server responds with a 304 (Not Modified) response, which does not include the content. This allows the client to periodically poll the server for changes of a given resource, without wasting much bandwidth in unnecessary unchanged content.

Now, let us assume that the client wants to modify the contents of the resource, assuming that the client has cached the latest version of the content. This is implemented with a conditional PUT request (3) that includes an *If-Match* header field containing the entity tag of the cached version of the content. The PUT request also contains a MIME body with the new desired content. The server receives the request, and compares the value of the entity tag in the *If-Match* header field with that of the stored version. If they match, then it means both the server and the client are synchronized, and the PUT request is executed, meaning that the MIME body included in the PUT request replaces the existing content. The server

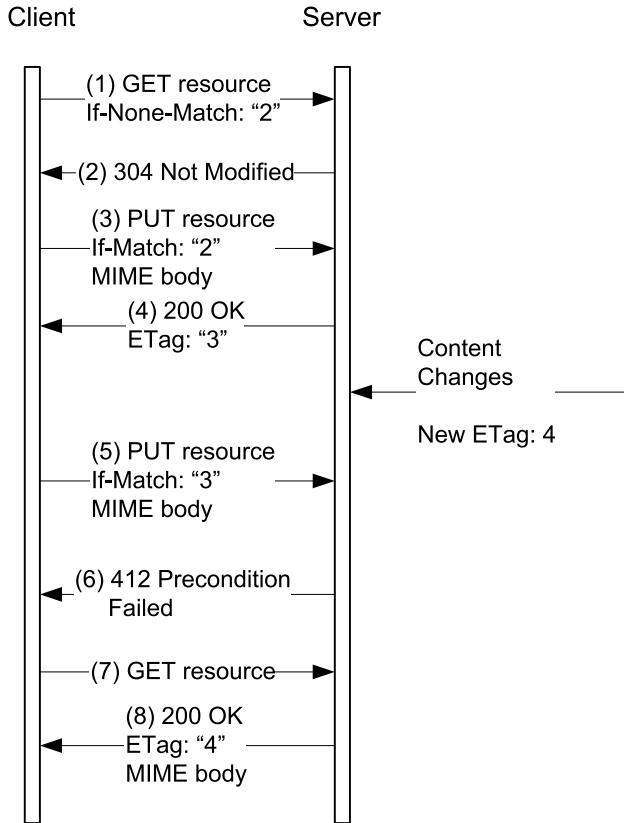


Figure 17.14: HTTP conditional requests

answers with a 200 (OK) response that includes an ETag header field that contains the newly allocated entity tag for the new version of the content, “3”.

Let us assume now that the content changes due to some external interaction, perhaps due to another client PUT request, or some manually updated data. The server allocates a new entity tag for that version of the content, “4”. If our client now wants to update the content, thinking that the latest known entity tag of the content is “3”, then it will send a PUT request (5) that contains the If-Match header field set to “3” along with a MIME body of the new desired content. However, “3” is not the latest entity tag of the content, so the server does not execute the conditional PUT request. Rather, it answers with a 412 (Precondition Failed) response. At this point in time, the only thing that the client can do to synchronize the content with that of the server, is to issue an unconditional GET request (7) to retrieve the contents identified by the resource. The server answers with a 200 (OK) response that contains an ETag header field with the value of the entity tag, “4”, along with a MIME body that contains the latest version of the content.

All of the HTTP conditional requests are directly applicable to XCAP, as conditional operations, since XCAP uses HTTP as an underlying protocol. XCAP conditional operations are very useful. For example, before adding a friend to a presence list, the client should make sure that they have the latest version of the presence list, otherwise, an unconditional

insertion of a new friend could produce undesired results. On the other hand, conditional fetch operations make sure that the content is only downloaded from the server if it has changed with respect to the version stored in the client, avoiding an unnecessary waste of bandwidth.

17.6 Subscriptions to Changes in XML Documents

We indicated earlier that any authorized user can modify a given XML document from any XCAP client. Therefore, it is possible for an XCAP client to retrieve an XML document, e.g., a list of friends in a presence list, and store it into memory. Later, another XCAP client might retrieve the same document, modify the list by adding or deleting friends to it, and upload the changes to the XCAP server. The first XCAP client will not be notified of the changes made by the second XCAP client. This creates a problem in typical situations where an XML document is accessed from different devices, such as a computer and a mobile device.

A possible solution to overcome this problem consists of periodically performing a conditional fetch operation on the XML document. This effectively minimizes the risk of getting stalled with an outdated XML document. The maximum time a client could have an outdated XML document is equal to the period of the polling interval. While polling is a solution, it does not offer the accuracy required by many applications, unless the polling interval is kept short, in which case the XCAP overhead traffic might become non-negligible. Furthermore, typically a user might be managing dozens of XML documents, so it is certainly not efficient to poll each document periodically to find out if it has changed.

Another more accurate solution is offered by the combination of two specifications: The XCAP Diff Event Package, specified in the Internet-Draft “An XCAP Diff Event Package” [312] and the XCAP Diff Format, specified in the Internet-Draft “An XCAP Document Format for Indicating A Change in XCAP Resources” [291]. These two specifications jointly provide a subscription/notification mechanism for getting one or more XML documents synchronized with those stored in an XCAP server. The former provides the general behavior for an event package; the latter specifies an XCAP-Diff XML document format that is able to express the changes occurred in XML documents. The mechanism, therefore, uses SIP and the SIP-event notification framework to keep XCAP documents synchronized. Since the mechanism is based on subscriptions, whenever there is a change in an XML document, the server sends a notification to all subscribers indicating the details of the change.

The XCAP-Diff mechanism allows the terminal to subscribe not only to changes to the whole XML document, but also to changes in a particular element or attribute of an XML document. Furthermore, the subscriber can issue a subscription for a collection of XML documents, elements, attributes, even those pertaining to different XML documents. So, a single subscription can manage the synchronization of several documents, elements, or attributes pertaining to XML documents stored in the server. This wide range of different subscriptions, varying from an element or attribute, passing through a whole document, and ending in all readable documents stored in the server, provides the required flexibility to meet all of the possible deployment requirements.

The subscription to the `xcap-diff` event package requires some means to indicate the list of XML documents, elements, attributes, etc., of the subscription. This list of XML resources is contained in another XML document called a *resource list*. The resource list contains a collection of URLs that represent XCAP resources or collections of XML documents for which subscription to the changes in them provokes a notification to the subscriber.

This resource list is included as a MIME body in the SUBSCRIBE request that creates the subscription.

The SUBSCRIBE request sets an Event header field to the name of the event package: `xcap-diff`. The Event header value is also accompanied by a `diff-processing` parameter that indicates to the server how it should express the differences between two versions of the document. The value in this parameter determines the actual content and format of future notifications. There are three possible values of the `diff-processing` parameter: “`no-patching`”, “`xcap-patching`”, and “`aggregate`”.

The value “`no-patching`” in the `diff-processing` parameter tells the server that the client is interested in being informed when a document, element, attribute, etc., has changed. If the notification describes a change in the value of an XML element or attribute, the new value of the changed element or attribute is included in the notification. However, if the subscription is made towards an XML document, and something in that document changes, the new document is not sent in the notification. Instead, only the new entity tag of that changed XML document is included in the notification. If the XCAP client wishes to retrieve the XML document, it would need to invoke an XCAP retrieve operation to receive the new version of the document.

The value “`xcap-patching`” in the `diff-processing` parameter, which is the default value, indicates to the server that the client is interested in receiving the actual changes in XML documents, elements, attributes, etc., along with the values of the entity tags (ETag) of those resources. The value “`xcap-patching`” provides a sequence of all changes that the resource has suffered from the time it was last notified until the current time, offering the possibility to re-create all intermediate values that the resource had until the current time.

Finally, the value “`aggregate`” in the `diff-processing` parameter makes the server to provide the initial and final values of the change in an XML document, element, or attribute, but not the intermediate values that the resource had in between initial and final value. Figure 17.15 shows an example of an Event header field used to subscribe to changes of XML documents.

```
Event: xcap-diff; diff-processing=xcap-patching
```

Figure 17.15: Event header field in a SUBSCRIBE request

Figure 17.16 depicts the operation of the XCAP Diff event package and its format. An XCAP/SIP client creates a regular SIP subscription to the `xcap-diff` event package by sending a SIP SUBSCRIBE request (1) to the XCAP/SIP server. This SUBSCRIBE request (1) contains a resource list that describes the XML documents, elements, attributes, etc., for which changes in them should be notified, as indicated earlier. An example of this SUBSCRIBE request (1) is shown in Figure 17.17. The *Request-URI* is set to the SIP URI of the server. The body of the SUBSCRIBE request contains a resource list that provides the URIs of each of the resources for which a subscription to changes is wanted. In the example of Figure 17.17 there are three URIs, corresponding to all of the documents pertaining to the `pidf-manipulation` XCAP application usage, the index of documents of the `resource-lists` XCAP application usage, and an element of the `rls-services` XCAP application usage.

The XCAP/SIP server receives the SUBSCRIBE request (1), generates and 200 (OK) response (2), and then examines the body included in the SUBSCRIBE request (1). For each requested resource to be monitored, it first verifies the requested documents for which the

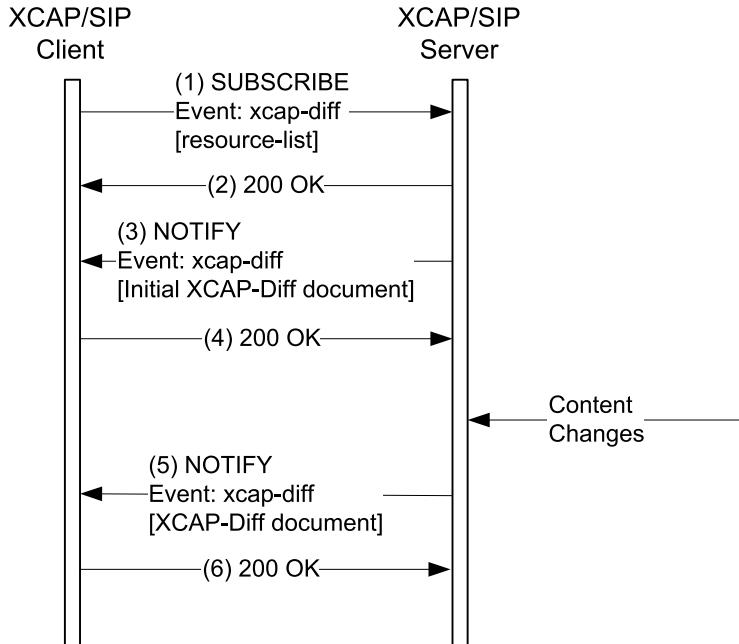


Figure 17.16: Subscription to changes in XML documents

user has read permission (here we have simplified the authentication mechanism). Then the XCAP/SIP server creates an initial XCAP-Diff document that contains the references to the subscribed resources (documents, elements, or attributes) including all of the data required to create an HTTP URI for retrieving them, along with the current entity tags. Then the server includes this XCAP-Diff document in a NOTIFY request (3). Figure 17.18 shows an example of an initial XCAP-Diff document included in the NOTIFY request (3) of Figure 17.16, which includes data about a single document in the pidf-manipulation XCAP application usage, the requested document in the resource-lists XCAP application usage, and the element `sip:colleagues@example.com` in the rls-services application usage. For each reference to an XML document, element, or attribute included in the body of the NOTIFY request, the value of the entity tag is included. When the terminal receives the NOTIFY request (3), it can verify whether it has a cached copy of the same version of documents, owing to the presence of the entity tag of each document. If it does not have that version of the documents, then the terminal does an unconditional fetch operation on the URI of the document to retrieve that base version.

If at a later time, any of the documents change, perhaps because another authorized user makes a change in the document, the server sends a NOTIFY request (5) that contains another XCAP-Diff document. The XCAP-Diff document lists the affected document in the `<document>` element, both with the old version (identified with the `previous-etag` entity tag attribute) and the new version (identified with the `new-etag` entity tag attribute). For example, assume that the `index` document of the pidf-manipulation application usage has changed. The NOTIFY request (5) contains an XCAP-Diff document that lists the changed document along with its new entity-tag. This is illustrated in Figure 17.19.

```

SUBSCRIBE sip:xcap.example.com SIP/2.0
Via: SIP/2.0/UDP pc55.example.com;branch=z9hG4bKn9s66
From: Alice <sip:alice@example.com>;tag=d9sjopo
To: Bob <sip:xcap.example.com>
Call-ID: 29d9s@pc55.example.com
CSeq: 392 SUBSCRIBE
Max-Forwards: 70
Expires: 3600
Event: xcap-diff; diff-processing=aggregate
Accept: application/xcap-diff+xml
Contact: <sip:alice@pc55.example.com>
Content-Type: application/resource-lists+xml
Content-Length: 376

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
  <list>
    <entry uri="pidf-manipulation/" />
    <entry uri="resource-lists/users/sip:alice@example.com/index" />
    <entry uri="rls-services/users/sip:alice@example.com/index" />
    ~~/*/service%5b@uri='sip:colleagues@example.com'%5d"/>
  </list>
</resource-lists>

```

Figure 17.17: SUBSCRIBE request (1)

17.7 XML Patch Operations

There are many occasions when the initial version of an XML document evolves, owing to modifications in the document. For example, if a user is adding one more friend to a presence list, most likely he is modifying an existing XML document to add a new XML element. In those cases, it is beneficial to develop a language for expressing the changes that a document suffered from one version to another one. XML patch operations, specified in the Internet-Draft “An Extensible Markup Language (XML) Patch Operations Framework Utilizing XML Path Language (XPath) Selectors” [311] provides exactly that functionality. The specification defines the format of XML Patch Operations documents, which as one can expect, are XML documents that offer a set of conventions to express changes incurred in other XML documents. The recipient of an XML patch operations document can apply those “patches” to an existing document and obtain, as a result, a new version of the XML document.

The patching process is graphically illustrated in Figure 17.20. An existing XML document is identified with an entity tag “1”. An XML patch operations document describes the operations that need to be applied to the XML document with entity tag “1”. When those operations are executed, the result is another version of the XML document, which receives a different entity tag (“2” in the example).

There are three patch operations: add, replace, and remove. Each of them is represented in an `<add>`, `<replace>`, or `<remove>` elements, respectively, in an XML patch operations document. Each patch operation can be applied to either an element, attribute, namespace,

```

<?xml version="1.0" encoding="UTF-8"?>
<xcap-diff xmlns="urn:ietf:params:xml:ns:xcap-diff"
            xcap-root="http://xcap.example.com/">

<document new-etag="3ba8spp03"
          sel="pidf-manipulation/users/sip:alice@example.com/index"/>

<document new-etag="39dnsa21"
          sel="resource-lists/users/sip:alice@example.com/index"/>

<xd:element sel="rls-services/users/sip:alice@example.com/index"
             ~~~/*service%5b@uri='sip:colleagues@example.com'%5d"
             xmlns:xd="urn:ietf:params:xml:ns:xcap-diff"
             xmlns="urn:ietf:params:xml:ns:rls-services"
             xmlns:rl="urn:ietf:params:xml:ns:resource-lists">
    <service uri="sip:colleagues@example.com">
        <list name="marketing">
            <rl:entry uri="sip:amber@example.com"/>
            <rl:entry uri="sip:leo@example.com"/>
            <rl:entry uri="sip:john@example.com"/>
        </list>
        <packages>
            <package>presence</package>
        </packages>
    </service>
</xd:element>
</xcap-diff>
```

Figure 17.18: Initial XCAP-Diff document included in a NOTIFY request (3)

```

<?xml version="1.0" encoding="UTF-8"?>
<xcap-diff xmlns="urn:ietf:params:xml:ns:xcap-diff"
            xcap-root="http://xcap.example.com/">

<document previous-etag="3ba8spp03"
            new-etag="b8s8b91FG"
            sel="pidf-manipulation/users/sip:alice@example.com/index"/>

</xcap-diff>
```

Figure 17.19: Subsequent XCAP-Diff document included in a NOTIFY request (5)

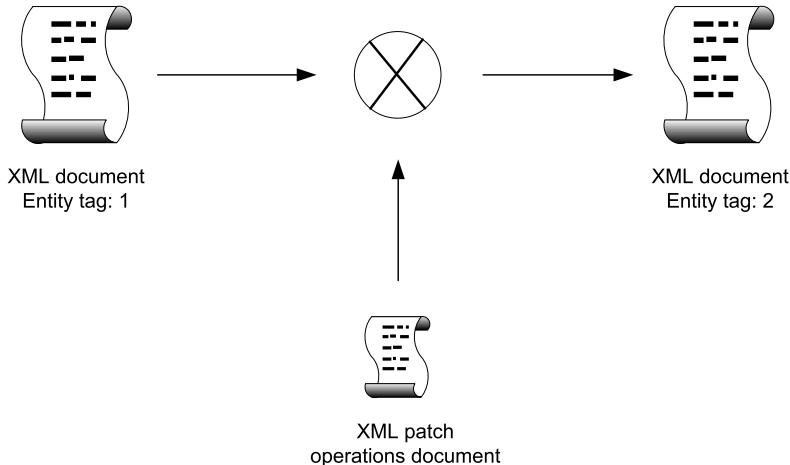


Figure 17.20: XML patching operations

comment, text, or processing instruction. A `sel` attribute indicates the *selector*, i.e., the target of the change. The process is intuitive: the value of the `sel` attribute traverses the hierarchy of the XML document until it reaches the target.

Let us assume our favorite XML document, “ims.xml”, included in Figure 17.5. Let us assume that we want to change the title of the book so that it includes the IMS acronym in it. We can apply the patches of the XML patch operations document contained in Figure 17.21 to replace the `<title>` element with a new one. The XML patch operations document contains a single operation: replace. The `sel` attribute describes the hierarchy of the document: first, go the book element, then to the `title` element, and finally, select the whole text (the value) in it.

```

<?xml version="1.0" encoding="UTF-8"?>
<p:diff xmlns="urn:org:miguel:book"
          xmlns:p="urn:ietf:params:xml:ns:diff">

    <p:replace sel="book/title/text()">
        The 3G IP Multimedia Subsystem (IMS)
    </p:replace>
</p:diff>

```

Figure 17.21: XML patch operations document

There is an immediate advantage of XML patch operations over XCAP: XML patch operations allows a series of multiple operations that can affect different elements or attributes of the XML document to be pipelined. However, XCAP allows a single operation on a single object (element, attribute, or document). The consequence is that XML patch operations can express complex sequences of operations, while if the same set of operations were to be performed with XCAP, it would require an XCAP request per operation.

Chapter 18

Service Configuration in the IMS

Chapter 17 provided a description of the protocols at the user's disposal for configuring services on the Internet. We saw that the service configuration architecture assumes an XML document stored on a server. The client retrieves a copy of the XML document, makes changes to it, and sends the delta back to the server.

In IMS, the architecture for service configuration architecture is developed around the XML Document Management (XDM) architecture created by the Open Mobile Alliance (OMA) in the XDM [244] set of specifications. XDM allows a user to modify XML documents stored in remote network servers. It also allows the local copy of those XML documents in the IMS terminal to be synchronized with the copy stored in network servers, so that if the user makes changes to one XML document from a given IMS terminal, other terminals are updated with the latest changes. Last, the XDM architecture also provides limited support for searches of information stored in these XML documents.

18.1 XDM architecture

Let us describe the XDM architecture with the help of Figure 18.1. The XDM architecture assumes a terminal that implements the role of an XCAP client, and one or more servers, called XDM servers, that implement the role of XCAP servers. XML documents are stored in any of the servers and kept synchronized with the copy in the client. When a user wants to change a configuration setting in a service, such as adding a friend to a presence list, the user changes the local copy of the XML document in the client and sends the change to the server, which applies it, and stores an updated version of the XML configuration document.

The XDM architecture introduces the following concepts.

XDM client (XDMC). This is an XCAP client running in the IMS terminal. The XDMC implements the core XDM features and some application-specific features.

Aggregation proxy. This is an HTTP proxy that is configured as an HTTP reverse proxy. The Aggregation Proxy authenticates the XDMC towards an XDMS. It may also route XCAP requests towards an XDMS or towards an Aggregation Proxy located in remote network. It also routes search requests towards the Search proxy. When receiving responses, it can compress the body of the response.

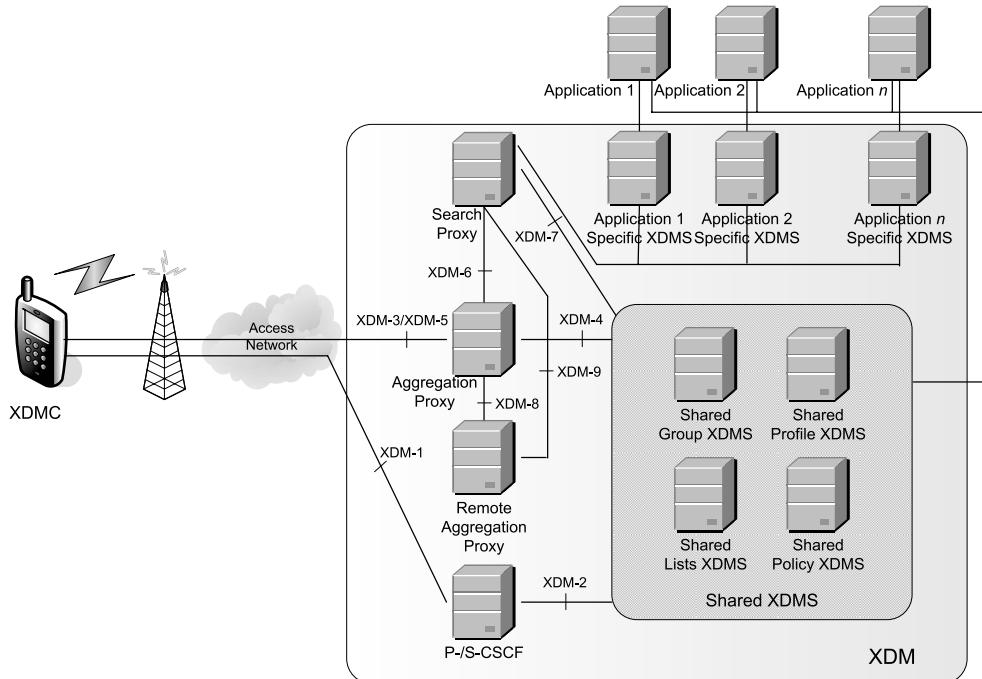


Figure 18.1: XDM architecture

Search proxy. This is an HTTP proxy that processes search requests received from XDMCs towards XDMSs or remote Aggregation Proxies. On received responses, the Search proxy combines the results of the responses received from XDMSs and remote Aggregation Proxies before forwarding them to the XDMC.

Shared XDM servers. XDM servers (XMDSS) that are used by several applications. Shared XDMSs are effectively XCAP servers. There are specialized shared XDMSs: shared list XDMS, shared group XDMS, shared profile XDMS, and shared policy XDMS.

XDMSs. This is an application-specific server for service configuration purposes. Effectively, XDMSs are XCAP servers that serve a single application.

Most of the interfaces of the XDM architecture are implemented with XCAP. This is the case of the interfaces *XDM-3*, *XDM-4*, *XDM-8*. However, interfaces *XDM-1* and *XDM-2* are used for subscription to changes in XML documents, based on the XCAP event package for SIP. Consequently, *XDM-1* and *XDM-2* are SIP interfaces. Interfaces *XDM-5*, *XDM-6*, *XDM-7*, and *XDM-9* implement the Limited XQuery over HTTP protocol. Unnamed interfaces in Figure 18.1 are defined by their respective applications, but they typically consist of XCAP, SIP for subscriptions to XCAP event packages, and Limited XQuery over HTTP.

The XDM architecture considers different applications that can be customers of the XDM service (or enabler, as it is called by the OMA). For example, the presence service is a customer of XDM, because the list of watchers, the authorization policies, etc., are all stored in XML documents managed by XDM. Other services that use XDM include Push-to-talk

over Cellular and PSTN/ISDN simulation services. Owing to this diversity of customers of XDM, the XDM architecture considers the existence of different XDM servers, each one perhaps specialized in serving a given application. An XDMS is a logical representation of the service configuration aspects of a service or application. In real products, it is expected that XDM servers are integrated into the specific server (e.g., presence server, PoC server, etc.).

Similarly, XDMCs are not really new entities, but just the logical representation of an XDM client in an IMS terminal.

18.2 Downloading an XML Document, Attribute, or Element

When an XDMC wants to receive an XML document, or a part of it, e.g., an attribute or element, it invokes the HTTP GET request in XCAP. The basic HTTP GET operation according to the XDM architecture is shown with the help of Figure 18.2. The XDMC sends an HTTP GET request (1) to the Aggregation Proxy. It is assumed that the address of the Aggregation Proxy is pre-provisioned in the IMS terminal. An example of this HTTP GET request (1) is shown in Figure 18.3. The *Request-URI* is set to the part of the XCAP URI that is identified in HTTP, thus, excluding the host name. Of relevance in the HTTP GET request is the inclusion of an X-3GPP-Intended-Identity header field that is populated with the identity that the user wants to present to the Aggregation Proxy. The HTTP X-3GPP-Intended-Identity header field is specified in 3GPP TS 24.109 [9] and contains a SIP Public User Identity of the user. This Public User Identity is also present as a username in the XCAP URI of the GET request.

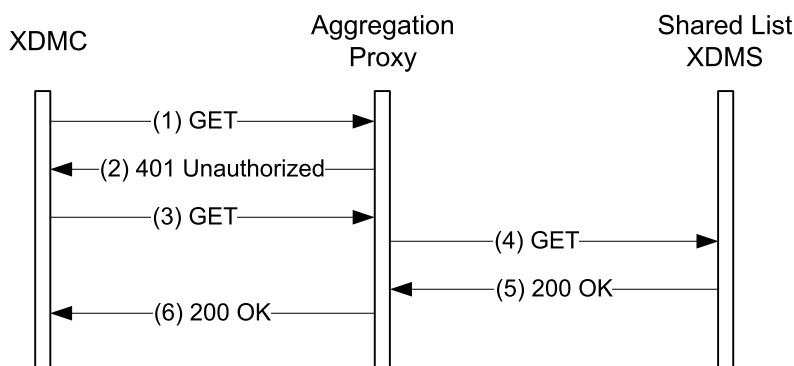


Figure 18.2: XDM: basic GET operation

```

GET /resource-lists/users/sip:alice@home1.com/index HTTP/1.1
Host: xdm.home1.com
User-Agent: XDM-client/OMA2.0
Date: Thu, 04 Mar 2008 22:10:45 GMT
X-3GPP-Intended-Identity: "sip:alice@home1.com"
Accept-Encoding: gzip
  
```

Figure 18.3: HTTP GET request (1)

The Aggregation Proxy first needs to authenticate the user. There are several approaches to authenticate a user. One of them consist of using the Generic Authentication Architecture (GAA, specified in 3GPP TS 33.222 [15]). Essentially, GAA uses HTTP over TLS (specified in RFC 4346 [121]), for security purposes, and then either a certificate in the client or a share password. So, if GAA is used, all of the authentication takes place at the TLS level and not at the HTTP level.

However, Figure 18.2 shows another authentication mechanism based on HTTP Digest Authentication (specified in RFC 2617 [145]). The Aggregation Proxy receives an HTTP GET request (1) and issues a 401 (Unauthorized) response (2) that contains a challenge in the WWW-Authenticate header field. As a minimum, the challenge contains the realm where the user will be authenticated, and a nonce, which is a random value that prevents against replay attacks. An example of this 401 (Unauthorized) response is shown in Figure 18.4.

```
HTTP/1.1 401 Unauthorized
Server: XDM-proxy/OMA2.0
Date: Thu, 04 Mar 2008 22:10:46 GMT
WWW-Authenticate: Digest realm="home1.com", qop=auth-int,
    nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093"
Content-Length: 0
```

Figure 18.4: HTTP 401 (Unauthorized) response (2)

The IMS terminal receives the 401 (Unauthorized) response (2), extracts the challenge, computes the response to that challenge (considering the username, realm, password, and nonce), and generates a second HTTP GET request (3) that contains that response to the challenge in the Authorization header field. An example of this HTTP GET request (3) is shown in Figure 18.5.

```
GET /resource-lists/users/sip:alice@home1.com/index HTTP/1.1
Host: xdm.home1.com
User-Agent: XDM-client/OMA2.0
Date: Thu, 04 Mar 2008 22:10:49 GMT
X-3GPP-Intended-Identity: "sip:alice@home1.com"
Authorization: Digest realm="home1.com", qop=auth-int,
    nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
    username="sip:alice@home1.com", nc=00000001,
    uri="/resource-lists/users/sip:alice@home1.com/index",
    response="2c8ee200cec7f6e966c932a9242554e4",
    cnonce="3bb96fe6e98a02ccb4555609c1b1bb56"
```

Figure 18.5: HTTP GET request (3)

The Aggregation Proxy, upon receiving of this new HTTP GET request (3) that contains the credentials, first evaluates the response to the presented challenge and computes if it matches the response provided by the XDMC. If the response to the challenge is correct, then it removes the Authorization header field and continues processing the request. The Aggregation Proxy adds its own Via header field, and preserves the X-3GPP-Intended-Identity header field. Then it inspects the *Request-URI*, determines,

based on the Application Unique ID (AUID), the actual server that is responsible for serving the request, and forwards the HTTP GET request (4) to that server. In the example, the AUID is set to `resource-list`; the Shared List XDMS is responsible for that application, thus is the recipient of the HTTP GET request (4). An example of the forwarded request is shown in Figure 18.6.

```
GET /resource-lists/users/sip:alice@home1.com/index HTTP/1.1
Host: xdm.home1.com
Via: HTTP/1.1 proxy.home1.com
User-Agent: XDM-client/OMA2.0
Date: Thu, 04 Mar 2008 22:10:50 GMT
X-3GPP-Intended-Identity: "sip:alice@home1.com"
```

Figure 18.6: HTTP GET request (4)

The Shared List XDMS then provides the requested XML document, element, or attribute in a 200 (OK) response (5), and forwards the response back to the Aggregation Proxy. An example of this response is shown in Figure 18.7.

```
HTTP/1.1 200 OK
Server: XDM-serv/OMA2.0
Date: Thu, 04 Mar 2008 22:10:50 GMT
Etag: "asd92d092"
Content-Type: application/resource-lists+xml
Content-Length: 238

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
    <list name="family">
        <entry uri="sip:daddy.joe@home1.com"/>
        <entry uri="sip:mom.berta@home1.com"/>
    </list>
</resource-lists>
```

Figure 18.7: 200 (OK) response (5)

The Aggregation Proxy then compresses the body of the 200 (OK) response, i.e., the XML document, according to any of the supported algorithms specified by the XDMC in the HTTP GET request (3). It also adds an `Authentication-Info` header field that contains the next nonce that the XDMC can use in a future HTTP request (so, future requests can directly include the response to a challenge and avoid one round trip). Then it forwards it back to the XDMC. An example of this response is shown in Figure 18.8.

18.3 Directory Retrieval

Sometimes users may need to retrieve all of their XML configuration documents that pertain to one or more application usages. This might be the case, for example, if the user is configuring a new IMS terminal. The XDM architecture offers a mechanism to do perform

```

HTTP/1.1 200 OK
Server: XDM-serv/OMA2.0
Via: HTTP/1.1 proxy.home1.com
Date: Thu, 04 Mar 2008 22:10:51 GMT
Etag: "asd92d092"
Authentication-Info: nextnonce="2a219bcb2ad25ae32cbe280c249f"
Content-Encoding: gzip
Content-Type: application/resource-lists+xml
Content-Length: 159

[binary compressed data]

```

Figure 18.8: 200 (OK) response (6)

this type of operation. The mechanism is briefly described in Figure 18.9, which for the sake of brevity shows only two shared XDMSSs.

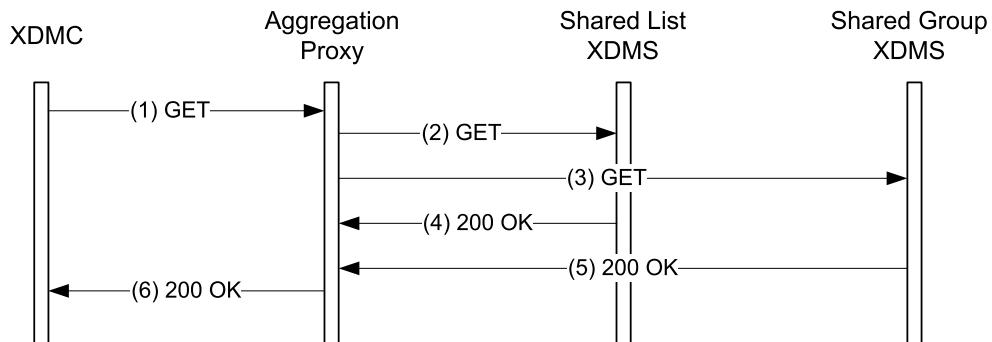


Figure 18.9: Directory retrieval with XDM

According to Figure 18.9, when the XDMC wants to obtain a directory retrieval for one or more applications, the XDMC sends an HTTP GET request (1) to its Aggregation Proxy. Figure 18.9 assumes that the user is in possession of the next nonce value, obtained in a previous HTTP operation, so the XDMC can compose a response to a challenge to authenticate towards the Aggregation Proxy. An example of this HTTP request (1) is shown in Figure 18.10.

What makes the HTTP GET request (1) in Figure 18.10 different, so that it is interpreted as a directory retrieval operation? The *Request-URI*, which partially contains the URI of the resource, is the key. The *Request-URI* contains a special AUID called “org.openmobilealliance.xcap-directory”, which indicates the directory retrieval operation. In addition, the *Request-URI* refers to a special document called “directory.xml” which is available under the user’s tree. The OMA defines the structure of the XML document “directory.xml” in the XDM specification [245].

The Aggregation Proxy, once it has verified the identity of the user, forwards the GET request (2, 3) to each of the XDMSSs. Then, each XDMSS returns a 200 (OK) response (4, 5) that contains a “directory.xml” document that lists all of the XML documents belonging to that user and stored in that server for that application usage. An example of one of the

```

GET /org.openmobilealliance.xcap-directory/users/sip:alice@home1.com
/directory.xml HTTP/1.1
Host: xdm.home1.com
User-Agent: XDM-client/OMA2.0
Date: Thu, 05 Mar 2008 22:10:49 GMT
X-3GPP-Intended-Identity: "sip:alice@home1.com"
Authorization: Digest realm="home1.com", qop=auth-int,
nonce="102b98b7102dd2f0e8b11d0f600bfb0c789",
username="sip:alice@home1.com", nc=00000002,
uri="/org.openmobilealliance.xcap-directory/users
/sip:alice@home1.com/directory.xml",
response="3a59fff43373a95592464948595c2e45",
cnonce="abc123e6e98a02251455360951b1a900"

```

Figure 18.10: Directory retrieval: HTTP GET request (1)

200 (OK) response (5) is shown in Figure 18.11. The “directory.xml” is identified with a MIME type of vnd.oma.xcap-directory+xml.

```

HTTP/1.1 200 OK
Server: XDM-serv/OMA2.0
Date: Thu, 05 Mar 2008 22:10:52 GMT
Etag: "9sda09s"
Content-Type: application/vnd.oma.xcap-directory+xml
Content-Length: 432

<?xml version="1.0" encoding="UTF-8"?>
<xcap-directory xmlns="urn:oma:xml:xdm:xcap-directory" >
  <folder auid="groups">
    <entry uri="http://xdm.home1.com/org.openmobilealliance.groups
               /users/sip:alice@home1.com/family" etag="2098ds" />
    <entry uri="http://xdm.home1.com/org.openmobilealliance.groups
               /users/sip:alice@home1.com/co-workers" etag="39s09s" />
  </folder>
</xcap-directory>

```

Figure 18.11: Directory retrieval: 200 (OK) response (5)

The Aggregation Proxy waits sufficient time to receive all of the responses and then combines all the received “directory.xml” documents into a single one, which is included in a 200 (OK) response (6) to the XDMC (see Figure 18.12). Each `folder` element contains an `auid` attribute that indicates the application usage to which the included data pertains. This allows the XDMC to distinguish the application usages for which the user has XML documents, and the list of documents of each application usage.

XDM not only allows a user to retrieve a directory of all of their documents stored in all servers, but also allows all user documents pertaining to a given XCAP application usage to be retrieved. This is done with a smart trick: XCAP allows a single element to be requested that is part of an XML document; the XDMC requests the `folder` element of the requested

```

HTTP/1.1 200 OK
Server: XDM-serv/OMA2.0
Date: Thu, 05 Mar 2008 22:10:55 GMT
Etag: "x98w2k920"
Authentication-Info: nextnonce="3a2154334ce3409fd24e26872024"
Content-Type: application/vnd.oma.xcap-directory+xml
Content-Length: 599

<?xml version="1.0" encoding="UTF-8"?>
<xcap-directory xmlns="urn:oma:xml:xdm:xcap-directory" >
  <folder auid="groups">
    <entry uri="http://xdm.home1.com/org.openmobilealliance.groups
      /users/sip:alice@home1.com/family" etag="2098ds" />
    <entry uri="http://xdm.home1.com/org.openmobilealliance.groups
      /users/sip:alice@home1.com/co-workers" etag="39s09s" />
  </folder>
  <folder auid="resource-lists">
    <entry uri="http://xdm.home1.com/resource-lists/users
      /sip:alice@home1.com/index" etag="0s982k" />
  </folder>
</xcap-directory>

```

Figure 18.12: Directory retrieval: Aggregated 200 (OK) response (6)

application usage which is included in the overall `directory.xml` document. The flow is shown in Figure 18.13.

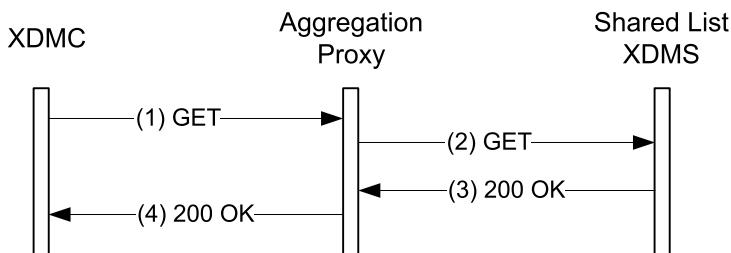


Figure 18.13: Directory retrieval with XDM for a given XCAP application usage

The XDMC sends an HTTP GET (1) whose *Request-URI* includes the pointer to the folder element whose *auid* attribute matches the requested application usage in the `directory.xml` document. In order to request a fraction of the `directory.xml` document, the *Request-URI* also contains the special AUID

`org.openmobilealliance.xcap-directory`

as well as the special file “`directory.xml`”. Figure 18.14 shows an example of this HTTP GET request (1). The notation

`/xcap-directory/folder%5b@auid=%22resource-lists%22%5d`

is an escaped representation of

```
/xcap-directory/folder[@auid="resource-lists"]
```

which is the XCAP mechanism to indicate the `folder` element whose `auid` attribute is set to the value `resource-lists`.

```
GET /org.openmobilealliance.xcap-directory/users/sip:alice@home1.com
/directory.xml/~~/xcap-directory
/folder%5b@auid=%22resource-lists%22%5d HTTP/1.1
Host: xdm.home1.com
User-Agent: XDM-client/OMA2.0
Date: Thu, 18 Mar 2008 00:14:33 GMT
X-3GPP-Intended-Identity: "sip:alice@home1.com"
Authorization: Digest realm="home1.com", qop=auth-int,
nonce="a126802384820945208bp98c0808e0f8021",
username="sip:alice@home1.com", nc=00000003,
uri="/org.openmobilealliance.xcap-directory
/users/sip:alice@home1.com/directory.xml
/~~/xcap-directory
/folder%5b@auid=%22resource-lists%22%5d",
response="2143cc17309a58b37459b3745b38bb39",
cnonce="c2948330a8530498d43309835d942531"
```

Figure 18.14: Directory retrieval of a specific application usage: HTTP GET request (1)

Upon receiving this HTTP GET request (1), the Aggregation Proxy verifies the credentials included in the request, and forwards the request (2) to the appropriate server. The address of this server depends on the application usage. In this case, the Shared List XDMS is responsible for the `resource-list` application usage. The Shared List XDMS builds an XCAP document that contains the `request` element. This is a special XCAP MIME type `application/xcap-el+xml` that is used to denote an element of an XML document, as opposed to the complete XML document. Figure 18.15 shows an example of this 200 (OK) response (3). The Aggregation Proxy merely forwards this response to the XDMC (4).

18.4 Data Search with XDM

XDM provides a limited search function, allowing XDMCs to search for data remotely stored in XDMSs. The search sequence flow is shown in Figure 18.16. In general, search requests originated in an XDMC are sent to an Aggregation Proxy, which forwards it to an appropriate Search Proxy, which further forwards it to the appropriate XDMS. As usually, the architecture defines functional elements that, in real implementations, can be combined in single boxes. So, it is appropriate to combine the Aggregation Proxy and the Search Proxy into the same physical equipment. Then, the actual search takes place in the XDMS, which returns the results of the search operation in an HTTP response.

When an XDMC wishes to search some data in one or more remotely stored XML document, the XDMC creates an HTTP POST request (1) that contains all of the details required to perform the search. This POST request encodes some details of the search operation in the *Request-URI*, but the complete detailed parameters of the search are included

```

HTTP/1.1 200 OK
Server: XDM-serv/OMA2.0
Date: Thu, 18 Mar 2008 00:14:35 GMT
Etag: "209d29d"
Content-Type: application/xcap-el+xml
Content-Length: 428

<?xml version="1.0" encoding="UTF-8"?>
<xcap-directory xmlns="urn:oma:xml:xdm:xcap-directory" >
  <folder aid="groups">
    <entry uri="http://xdm.home1.com/org.openmobilealliance.groups
               /users/sip:alice@home1.com/family" etag="9203" />
    <entry uri="http://xdm.home1.com/org.openmobilealliance.groups
               /users/sip:alice@home1.com/co-workers" etag="0293" />
  </folder>
</xcap-directory>

```

Figure 18.15: Directory retrieval of a specific application usage: 200 (OK) response (3)

in a special XML document called *the Search XML document*, which is included as a body of the POST request. The search function reuses the XQuery 1.0 specification [88]. XQuery provides mechanisms to extract and manipulate data from XML documents. In the context of XDM Search operations, XQuery expressions are included in Search XML documents which are transported in HTTP POST requests.

One of the parameters that characterize the search operation consists of the list of input documents where the search takes place. The document or documents where the search takes place can be composed of either all documents of all users pertaining to a given XCAP application usage, all documents pertaining to a given application usage and a given user, or a particular document. For example, let *AUID* be the AUID, *sip:alice@home1.com* a Public User Identity, and *mylist.xml* an XML document. If the search takes place over all documents stored under the “users” directory of the *AUID*, the document list is encoded as “[*AUID*]/users/”; if the search is restricted to Alice’s documents for the *AUID* application, then the search is encoded as “[*AUID*]/users/*sip:alice@home1.com*/”; finally, if the search is restricted to a single user’s document, then the list of documents is encoded with the following pattern: “[*AUID*]/users/*sip:alice@home1.com*/*mylist.xml*”.

So, where is this document list encoded in HTTP? It is encoded in two different places: in the Search XML document that is included in the HTTP POST request (1), and in the target parameter of the *Request-URI* of the same HTTP POST request (1). The reason for encoding the request in two places is twofold: on the one side, the Search XML document contains very detailed and extensive information of the search parameter, more than just the input documents, so, this justifies the presence of search documents in the Search XML body; on the other side, the Search Proxy requires knowledge of the AUID and the user’s name for routing the HTTP POST request (1) to the appropriate XDMS. Making this routing information accessible, e.g., in a parameter of the *Request-URI* makes things easy for the Search Proxy that, otherwise, would need to open the body and parse the Search XML document to route the HTTP POST request (1).

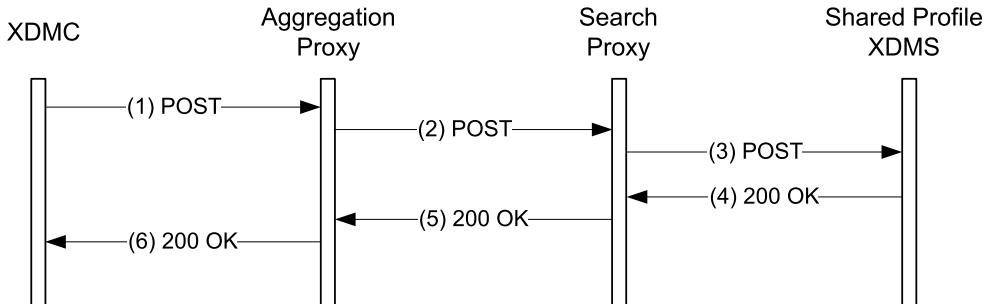


Figure 18.16: Searching data with XDM

Let us continue the description of the search request included in the HTTP POST request (1) with the help of the HTTP POST request example of Figure 18.17. According to this figure, the *Request-URI* contains a special minted AUID set to the string “org.openmobilealliance.search”. This allows the Aggregation Proxy to identify this request as a search request and forward it to the Search Proxy. In addition, the *Request-URI* contains a *target* parameter whose main purpose is to aid the Search Proxy in routing the HTTP POST request to the appropriate XDMS. A second optional domain restricts the search query to either the home domain, any other particular domain, or all remote domains. The *domain* parameter defaults to the “home” value.

In order to construct a search query, the XDMC creates an HTTP POST request addressed, in the *Request-URI*, to a specially constructed URI that identifies the search operation and the main search parameters. In Figure 18.17 the *Request-URI* is set to

```
/org.openmobilealliance.search?
  target=org.openmobilealliance.user-profile/users/
```

The first part of the *Request-URI*, set to “/org.openmobilealliance.search”, merely indicates that this is a search request, and it is later used by the Aggregation Proxy to route the request to a Search Proxy. The *target* parameter indicates the collection of documents where the search takes place, including the AUID, which is set to the *user-profile* AUID defined by the OMA. The *target* parameter is used by the Search Proxy to route the request to the appropriate XDMS.

The core parameters of the search are included in the Search XML document, which is identified with a MIME type of “application/vnd.oma-search+xml”. Let us take a closer look at the Search document with the help of the example in Figure 18.17. A Search XML document contains a root element **search-set** element that contains a single **search** element, which contains either a **request** or **response** element. The example of Figure 18.17 shows a **request** element, this is a request for data. Non-error responses contain a **response** element instead. The **request** element contains a **query** element that contains a limited or simplified XQuery expression according to the W3C XQuery 1.0 specification [88]. The XQuery expression is enclosed in the CDATA section of the Search XML document.

The XQuery expression begins with a preamble indicating the XQuery version number, and it defines a default namespace. Then it contains a simple “for” iteration loop that selects the collection of documents that serve as input to the search. In the example in Figure 18.17

```

POST /org.openmobilealliance.search?
    target=org.openmobilealliance.user-profile/users/ HTTP/1.1
Host: xdm.home1.com
User-Agent: XDM-client/OMA2.0
Date: Thu, 21 Mar 2008 23:35:23 GMT
X-3GPP-Intended-Identity: "sip:alice@home1.com"
Authorization: Digest realm="home1.com", qop=auth-int,
                nonce="a126802384820945208bp98c0808e0f8021",
                username="sip:alice@home1.com", nc=00000003,
                uri="/org.openmobilealliance.search?",
                target=org.openmobilealliance.user-profile/users/",
                response="390982098203984e0a84d0c410792ee0",
                cnonce="1a893b098385c674eef845098cc50a86"
Accept-Encoding: gzip
Content-Type: application/vnd.oma-search+xml
Content-Length: 647

<?xml version="1.0" encoding="UTF-8"?>
<search-set xmlns="urn:oma:xml:xdm:search">
    <search id="12098123">
        <request>
            <query>
                <! [CDATA [
                    xquery version "1.0";
                    declare default element
                        namespace "urn:oma:xml:xdm:user-profile";

                    for $x in
                        collection("org.openmobilealliance.user-profile/users/")
                            /user-profiles/user-profile
                        where ($x/profession/title="Engineer") and
                            ($x/address/country="ES")
                    return
                        <user-profile>{$x/@uri}{$x/display-name}</user-profile>
                ]]>
            </query>
        </request>
    </search>
</search-set>

```

Figure 18.17: Search operation: HTTP POST request (1)

```

HTTP/1.1 200 OK
Server: XDM-serv/OMA2.0
Date: Thu, 21 Mar 2008 23:35:30 GMT
Etag: "32092"
Content-Type: application/vnd.oma-search+xml
Content-Length: 537

<?xml version="1.0" encoding="UTF-8"?>
<search-set xmlns="urn:oma:xml:xdm:search"
             xmlns:up="urn:oma:xml:xdm:user-profile">

    <search id="12098123">
        <response>
            <up:user-profile uri="sip:miguel@example.com">
                <up:display-name>Miguel Garcia</up:display-name>
            </up:user-profile>
            <up:user-profile uri="sip:gonzalo@example.com">
                <up:display-name>Gonzalo Camarillo</up:display-name>
            </up:user-profile>
        </response>
    </search>
</search-set>

```

Figure 18.18: Search operation: HTTP 200 (OK) response (4)

the value selects all of the documents for all users pertaining to the User Profile application usage stored under the “users” tree in the Shared Profile XDMS, and for each document, the element `user-profile`, which is a child of the `user-profiles` element, is selected. Note that this information is also present in the `target` parameter of the *Request-URI* of the HTTP POST request, however, there it is only used for routing purposes.

The Search XML document then includes a “where” clause that indicates the condition that, in case it is evaluated as true, selects the data. In the example, the condition is set to those entries in the document where the data contains a `profession` element that includes a `title` element set to “Engineer” and the `address` element contains a `country` element set to “ES”. So, in human parlance, the Search document of Figure 18.17 is searching for a list of Spanish engineers across all of the user’s documents of the User Profile application usage. Last, the “return” clause specifies what should be returned in the response. In the example, the URI and display name of each match is returned.

Let us continue with the description of the sequence flow of Figure 18.16. The XDMC sends the HTTP POST request (1) to its Aggregation Proxy, which first verifies the credentials included in the `Authorization` header, and then inspect the *Request-URI*. Since the *Request-URI* contains an indication of a search request, the Aggregation proxy forwards the HTTP POST request (2) to a Search proxy. The Search Proxy inspects the `target` and `domain` parameters of the *Request-URI* for routing the request. This `target` parameter contains the application usage for the request, so the Search Proxy is able to determine, upon looking into an internal table, which is the XDMS responsible for that application usage. In the example in Figure 18.17, the User Profile application usage is indicated, so the Search

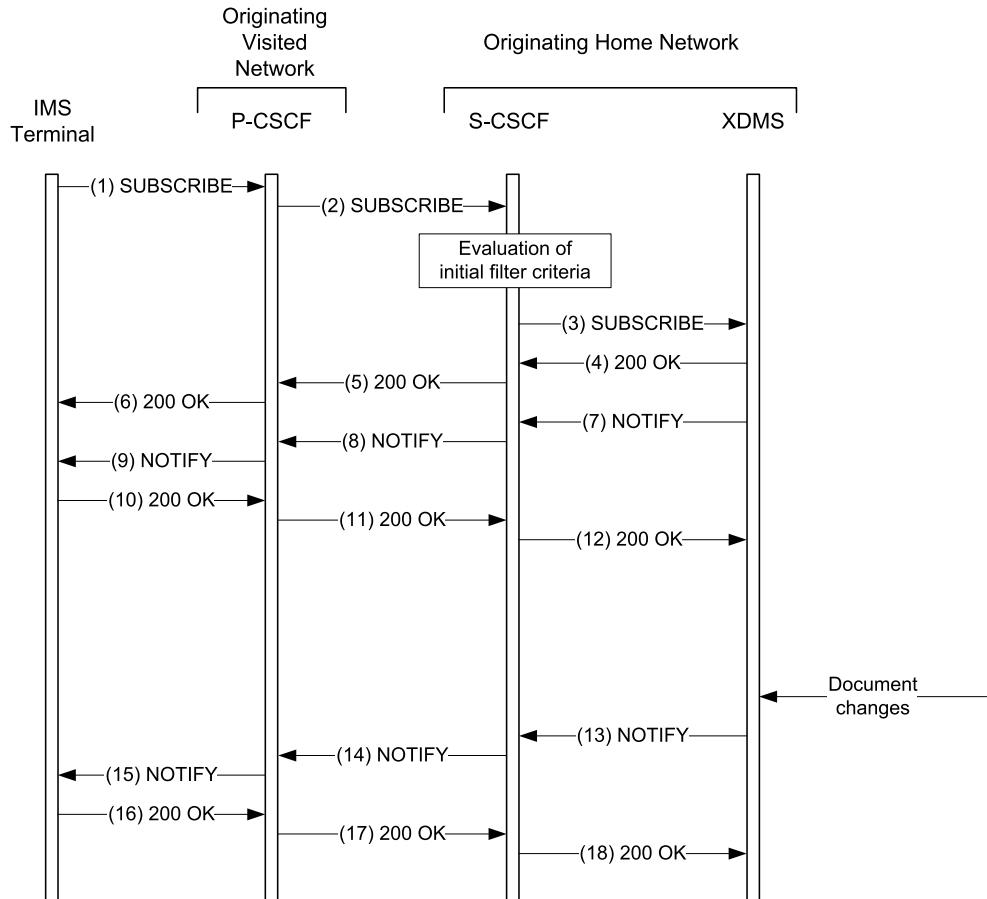


Figure 18.19: Subscription to changes in XML documents

Proxy forwards the HTTP POST request (3) to the Shared Profile XDMS. If the *Request-URI* had included a *domain* parameter set to a value different than the default value “home”, the Search Proxy would have forwarded the HTTP POST request to a remote Aggregation Proxy. The Search Proxy can also forward the request to different servers, including different servers located in different domains.

When the Shared Profile XDMS (or any other appropriate XDMS) receives the HTTP POST request (3), it inspects the Search XML document to establish the parameters of the search operation and the presentation of the results. The XDMS generates a 200 (OK) response (4) that contains a Search XML document. The Search XML document contains a *response* element that contains the list of matches formatted according to the *return* clause of the XQuery request. Figure 18.18 shows an example of the data returned by the Shared Profile XDMS.

The HTTP 200 (OK) response (4) is routed back to the Search Proxy. If the request was forwarded to several servers, the Search Proxy would collect all of the data from the different servers and aggregate it into a single response. Then, the Search Proxy forwards the HTTP

200 (OK) response (5) to the Aggregation Proxy. The Aggregation Proxy can compress the payload, if supported by the XDMC, but in any case, it forwards the response (6) to the XDMC. The XDMC now parses the data and presents it to the user.

18.5 Subscribing to Changes in XML Documents

XDM also provides a subscription/notification mechanism to changes in XML documents stored in any of the XDMSSs. The mechanism is a straight application of the XCAP-Diff event package that we described in Section 17.6.

Consider the example in Figure 18.19. The SUBSCRIBE request (1) contains the list of documents to which subscriptions are required, as well as an indication of the desired “diff processing” model (see Section 17.6 for a detailed description of the “diff processing” model). An initial XCAP-Diff document is included in the NOTIFY request (7). When a document changes, a new XCAP-Diff document contains the list of changes a second NOTIFY request (13).

Chapter 19

The Presence Service on the Internet

Presence is one of those basic services that, day by day, is becoming omnipresent. On the one hand, the presence service is able to provide an extensive customized amount of information about a given user to a set of users. On the other hand, third-party services are able to read and understand presence information, so that the service provided to the user is modified (actually, we should say customized) according to the user's needs and preferences expressed in the presence information.

19.1 Overview of the Presence Service

Presence is the service that allows a user to be informed about the reachability, availability, and willingness of communication with another user. The presence service is able to indicate whether other users are online or not, and if they are online, whether they are idle or busy (e.g., attending a meeting or engaged in a phone call). In addition, the presence service allows users to give details of their communication means and capabilities (e.g., whether they have audio, video, instant messaging, etc., capabilities and in which terminal those capabilities are present).

The presence framework defines various roles, as shown in Figure 19.1. The person who is providing presence information to the presence service is called a *presence entity*, or for short, a *presentity*. In Figure 19.1 Alice plays the role of a presentity. The presentity is supplying *presence information* (i.e., the set of attributes that characterize the properties of a presentity, such as status, capabilities, communication address, etc.). A given presentity has several devices known as *Presence User Agents* (PUAs) which provide information about her presence.

Figure 19.1 shows three PUAs: an IMS terminal, a laptop, and a desktop computer. Each has a piece of information about Alice, the presentity. The laptop knows whether Alice is logged on or not, as does the desktop computer. The IMS terminal knows Alice's registration status and whether she is engaged in any type of communication. They can have even richer presence information, such as what time Alice will be back from lunch, whether Alice is available for videoconferences, or whether she only wants to receive voice calls right now. All of the PUAs send their pieces of information to a *Presence Agent* (PA). The PA gathers all of the information received and obtains a complete picture of Alice's presence.

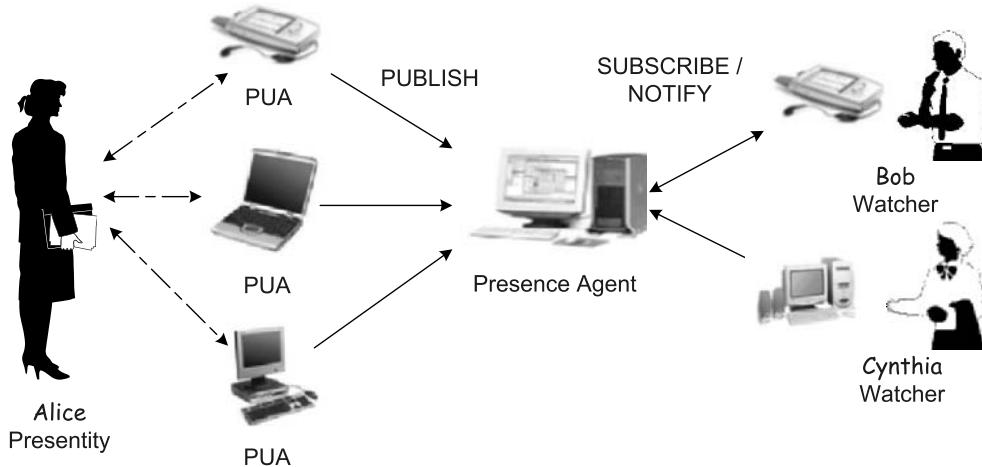


Figure 19.1: SIP presence architecture

A Presence Agent can be an integral part of a *Presence Server* (PS). A PS is a functional entity that acts as either a PA or as a proxy server for SUBSCRIBE requests. The term Presence Server is often used to refer to a PA. While both are quite similar in functionality, PSs also act as proxy servers for SUBSCRIBE request, while PAs do not.

Figure 19.1 also shows two *watchers*: Bob and Cynthia. A watcher is an entity that requests (from the PA) presence information about a presentity. There are several types of watchers. A fetcher is a watcher that retrieves the current presentity's presence information from the PA. A subscribed watcher asks to be notified about future changes in the presentity's presence information, so that the subscribed watcher has an accurate updated view of the presentity's presence information.

Typically, applications combine the watcher and presentity functionalities in a single piece of software, thus hiding the functional distinction of presence publication from presence information acquisition by the end-user. However, since both functions are different and governed by different procedures we treat them separately.

The presence service is a particular application built on top of the SIP event notification framework (we described the SIP event notification framework in Section 4.15). The framework allows a watcher to subscribe to or fetch (using a SIP SUBSCRIBE transaction) the presentity's presence information. The subscription state is kept in the presentity's PA, which acts as a notifier (according to the SIP event notification framework). The PA notifies (using SIP NOTIFY transactions) all of the subscribed watchers when a change has occurred in the presentity's presence information.

All SUBSCRIBE/NOTIFY transactions contain a SIP Event header field that identifies the actual event the subscription or notification is related to. RFC 3856 [268] defines the “presence” event package identified by the value *presence* in the Event header field of SUBSCRIBE and NOTIFY requests.

NOTIFY requests usually contain a MIME body that indicates the presentity's presence information. This body is an XML document formatted according to certain rules. Since the presentity's presence information is carried in an XML document, which is highly extensible, then it is easy to extend the presence information with all sorts of imaginable bells and

whistles. The fact that presence information is not carried directly in SIP, but in XML documents that are transported in SIP, provides a clear separation between the transport protocol layer (SIP) and the application protocol layer (XML documents containing presence information).

19.1.1 The pres URI

Traditionally, Internet technologies have used URIs to identify resources that can be accessed with a protocol (e.g., *sip*, *http*, and *ftp* URIs) or are associated to functionality (e.g., *tel* and *mailto* URIs). Presence defines (in RFC 3859 [248]) a *pres* URI for identifying a presentity or a watcher. It must be noted that the *pres* URI is protocol-agnostic: therefore, there is no information indicated in the URI on how to access the resource. However, when SIP is used to access presence resources it is recommended to use *sip* or *sips* URIs as they are protocol-specific.

The syntax of the *pres* URI is

```
PRES-URI      = "pres:" [ to ] [ headers ]
to            = mailbox
headers       = "?" header *( "&" header )
header        = hname "=" hvalue
hname         = *uric
hvalue        = *uric
```

An example of a *pres* URI is

```
pres:alice@example.com
```

A *pres* URI can replace a *sip* URI in any header field where a *sip* URI can be present, such as From, To, Route, Record-Route header fields and the Request-URI. It might be used in exceptional cases when a gateway from SIP to another protocol is involved. In typical scenarios only *sip* and *sips* URIs are used.

19.2 The Presence Life Cycle

As if it were a product, the presentity's supplied presence information has a life cycle. During its life cycle the presence information suffers a number of transformations, from its creation phase to its shipping and handling, storage, and the final delivery phase to consumers (watchers, in the case of presence).

Figure 19.2 shows a schematic representation of the first part of the life cycle of the presence information. A presentity (on the left-hand side of the figure), has some presence information to publish. The actual presence information varies slightly depending on which PUA the presentity is using. There can be several PUAs supplying different presence information, such as a computer, a mobile phone, and a fixed phone. For example, the presentity might be away from the keyboard of the computer, but engaged in a call on her mobile phone, so these details are reflected in their presence information.

At some point in time each of these PUAs will send a SIP PUBLISH request containing their view of the presentity's presence information in a presence document. This is the presence publication process, which is described in Section 19.4. The presence document, received at the PA, is fed into the merging process. The merging process, governed by

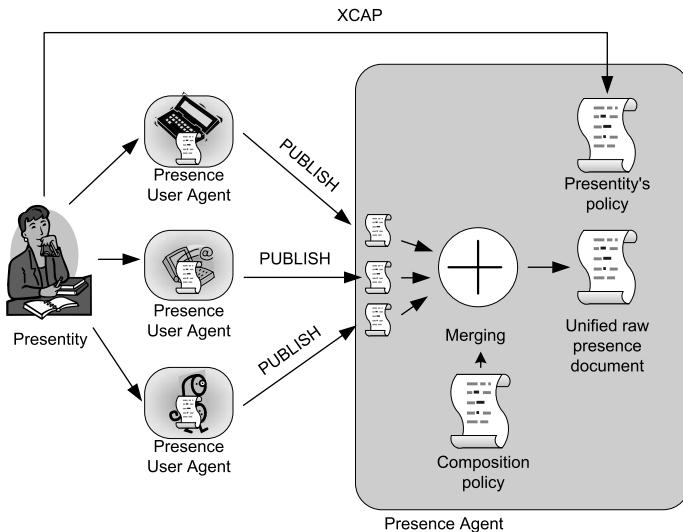


Figure 19.2: SIP presence life cycle (part 1)

a composition policy, allows the three presence documents to be merged into a unified raw version of the presentity's presence information. In addition, the presentity uploads a presentity's policy document (typically using XCAP, see Chapter 17). The presentity's policy document provides additional privacy settings that the PA will apply before serving the presentity's information to authorized watcher. For example, the policy document can indicate that certain watchers will not receive location information, while others will.

The second part of the presence life cycle is depicted in Figure 19.3. A watcher¹⁶ subscribes to a presentity's presence information by sending a SUBSCRIBE request to the presentity's URI. This SUBSCRIBE request can optionally contain a filter to limit the information that the watcher is interested in on the presentity. The PA receives the SUBSCRIBE request, authenticates the watcher, and extracts the watcher's identity and the filter (if present). Then the PA takes the presentity's unified raw presence document, the presentity's privacy policy document, and the watcher's identity, and applies the privacy policy to the unified presence document. The result is a potential presence document that is tailored to the watcher. This document is still potential because it has to suffer further transformations.

Then the PA takes the potential presence document and applies the watcher's filter that was received in the SUBSCRIBER request. This basically eliminates any extra information that the watcher is not interested in receiving. For example, a watcher may just be interested in receiving updates when the user changes his basic status information (e.g., online, offline), but not when the geographical coordinates change or his activities are updated. We describe the event notification filtering in Section 19.16.3.

Once the PA has a filtered presence document, there are two options: if full notifications are used, or if this is the first document of a partial notification, a full presence document is created (not shown in the figure); if partial notifications are used, and a full document has

¹⁶Note that the figure depicts a single watcher subscribed to the presentity's presence information. Typically several watchers will be subscribed to the same presentity's presence information.

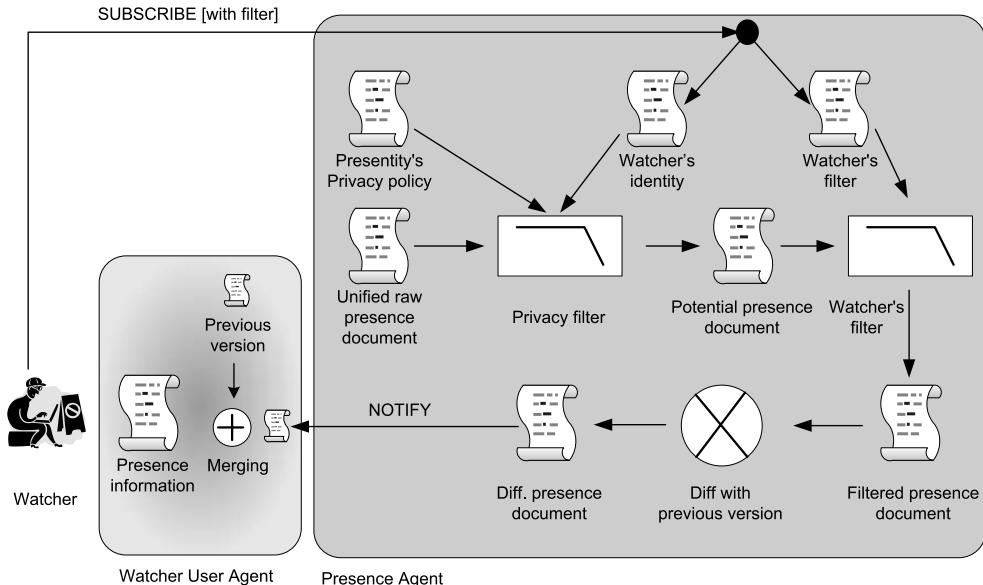


Figure 19.3: SIP presence life cycle (part 2)

been sent previously , the PA creates a differenced presence document with respect to the last previously sent copy, in which case only the changes are transmitted, as opposed to full presence documents. Partial notifications are further described in Section 19.16.1.

The PA then creates a NOTIFY request that contains the presence document and sends it to the watcher. This is the notification process, which is described in Section 19.3. If it is a differenced version, then the watcher uses the previously stored version and the received version with the changes, merges them, and gets the complete presence information document that is eventually used to display to the watcher. If it is a full presence document (not shown in the figure), all of the data are already contained in the document, so the watcher user agent merely displays the contents to the user.

19.3 Presence Subscriptions and Notifications

The interface defined between a watcher and a PA allows a watcher to subscribe to the presence information of a presentity. Presence subscription is implemented with a SIP SUBSCRIBE transaction. The subscription can be a simple fetch operation, whereby the watcher just wants to obtain the current presence information of a presentity, but does not want to be informed about future changes to such an information. Likewise the SUBSCRIBE request can install a subscription that lasts for a period of time (negotiated in the Expires header field). In this case the watcher obtains updates of the presentity's presence information whenever that information changes. If a watcher wants to keep the subscription active they need to renew it prior to its expiration.

Figure 19.4 shows an example flow. A watcher sends a SUBSCRIBE request (1) to the PA, the request including an Event header field set to presence, indicating the subscription to the presence information of a particular presentity. The PA authenticates and authorizes the watcher and answers with a 200 (OK) response (2), followed by a NOTIFY request (3),

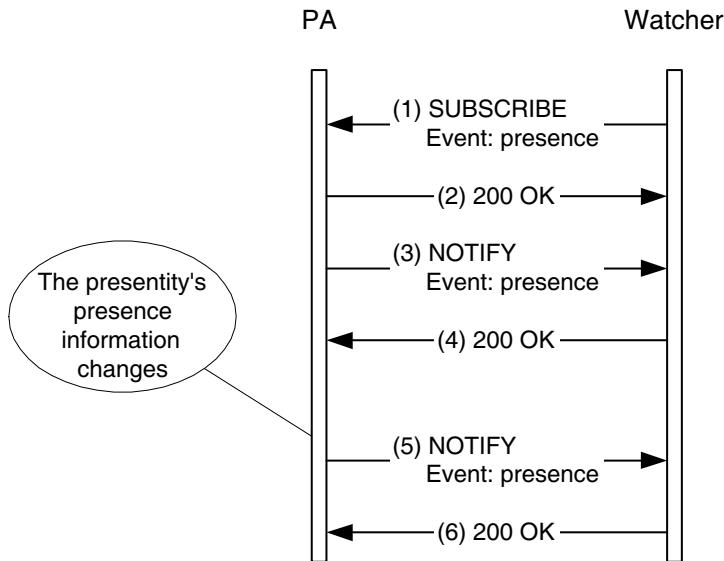


Figure 19.4: Subscription and notification of presence information

which, at this stage, may or may not contain an XML document that describes the presentity’s presence information. In case the NOTIFY contains a presence information document, then it is actually an XML document that is formatted according to rules of the Presence Information Data Format (PIDF), which we describe later in Section 19.5. In some cases, such as when the watcher has not yet been authorized, this NOTIFY request (3) just conveys the status of the subscription, in a `Subscription-State` header field, but it does not convey any PIDF document describing the presentity’s presence information.

The watcher acknowledges the reception of the NOTIFY request (3) by sending a 200 (OK) response (4) back to the PA. When the watcher is eventually authorized to obtain the presentity’s presence information, or whenever the presentity’s presence information changes, the PA sends to the watcher a new NOTIFY request (5) that includes a presence document (PIDF). The watcher replies with a 200 (OK) response (6).

Figure 19.5 shows an example of a SUBSCRIBE request that a watcher, such as Alice, sends as a subscription to Bob’s presence information. The `Request-URI` field is set to the presentity’s URI, i.e., Bob’s URI. Since the `Expires` header field is set to a non-zero value, then it is a subscription operation that will expire at some point in time. Alice suggested the subscription be installed for one hour. The PA will set the time in an `Expires` header field in the 200 (OK) response to this SUBSCRIBE request.

The SUBSCRIBE request also contains an `Accept` header field that lists the MIME types that watcher is able to understand for this particular subscription. This determines the type of MIME bodies that the PA will include later in NOTIFY requests. In the example of Figure 19.5, the watcher supports the PIDF format.

Figure 19.6 shows an example of a NOTIFY request that carries presence information in a PIDF document. A `Subscription-State` header field indicates the status of the subscription and the expiration timer.

```
SUBSCRIBE sip:bob@example.com SIP/2.0
Via: SIP/2.0/UDP pc.example.com;branch=z9hG4bKn9s66
From: Alice <sip:alice@example.com>;tag=d9sjopo
To: Bob <sip:bob@example.com>
Call-ID: b90dfn@pc.example.com
CSeq: 1 SUBSCRIBE
Max-Forwards: 70
Expires: 3600
Event: presence
Accept: application/pidf+xml
Contact: <sip:alice@pc.example.com>
Content-Length: 0
```

Figure 19.5: SUBSCRIBE request (1)

```
NOTIFY sip:alice@pc.example.com SIP/2.0
Via: SIP/2.0/UDP ps.example.com;branch=z9hG4bK72187
From: Bob <sip:bob@example.com>;tag=ns9s9d
To: Alice <sip:alice@example.com>;tag=d9sjopo
Call-ID: b90dfn@pc.example.com
CSeq: 8 NOTIFY
Subscription-State: active; expires=3000
Max-Forwards: 70
Contact: <sip:ps.example.com>
Event: presence
Content-Type: application/pidf+xml
Content-Length: 320

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
           entity="pres:alice@example.com">

    <tuple id="a3lkjdaf">
        <status>
            <basic>open</basic>
        </status>
        <contact priority="1.0">sip:alice@example.com</contact>
    </tuple>
</presence>
```

Figure 19.6: NOTIFY request (5)

19.4 Presence Publication

Section 19.3 provided an overview of the mechanisms that watchers have at their disposal for subscribing to someone else's presence information. We saw that this subscription typically terminates in a PA that collects all of the presence information of one or more users. However, we still have not described how presentities make their presence information available to the PA.

An obvious mechanism to use in this interface is the REGISTER method. REGISTER transactions provide the current location (IP address, not to be confused with geographical location) of the user. Therefore, when users are not registered the PA sets their presence to "offline" and when they are registered the PA sets their presence to "online". On the other hand, the semantics of the REGISTER method are very clear: REGISTER binds an Address-Of-Record (public identity) with a contact address. Therefore, it does not seem appropriate to overload these semantics for the purpose of presence publication. Consequently, we need another mechanism that allows PUAs to upload presence information (e.g., PIDF/RPID documents) to a PA.

The IETF defined the SIP PUBLISH method in RFC 3903 [219]. The purpose of a PUBLISH request is to publish the event state used within the framework for SIP-specific event notification (RFC 3265 [264]). Thus, the PUBLISH method is not only used for presence publication, it is generic enough to be used to publish any state associated with an event package. However, we focus in this section on presence publication.

Figure 19.7 shows a typical flow used to publish presence information: the PUA sends a PUBLISH request (1) that contains a PIDF document to the PA. We describe PIDF documents in Section 19.5. This PUBLISH request is represented in Figure 19.8. The Content-Type header field is set to the value application/pidf+xml, which identifies the payload as a PIDF document.

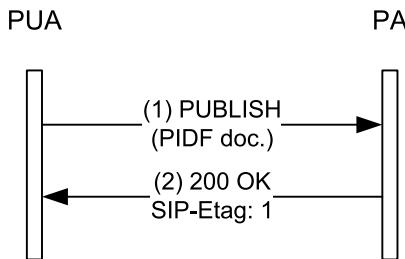


Figure 19.7: Publication of presence information

The PA acknowledges the reception of the PUBLISH request by sending a 200 (OK) response (2). The 200 (OK) response contains a SIP-Etag header field that can be used for providing a version number of the stored document. This is used in partial publications, which we will describe later in Section 19.16.2. The 200 (OK) response does not contain a body.

19.5 Presence Information Data Format (PIDF)

The PIDF is a protocol-agnostic XML document that is designed to carry the semantics of presence information between two presence entities. The PIDF is specified in

```

PUBLISH sip:alice@example.com SIP/2.0
Via: SIP/2.0/UDP pc.example.com;branch=z9hG4bKn9s9d
To: Alice <sip:alice@example.com>
From: Alice <sip:alice@example.com>;tag=429j2
Call-ID: 092us2309isdd@pc.example.com
CSeq: 2 PUBLISH
Max-Forwards: 70
Expires: 3600
Event: presence
Content-Type: application/pidf+xml
Content-Length: 320

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
           entity="pres:alice@example.com">

  <tuple id="a3lkjdaf">
    <status>
      <basic>open</basic>
    </status>
    <contact priority="1.0">sip:alice@example.com</contact>
  </tuple>
</presence>

```

Figure 19.8: PUBLISH request (1)

RFC 3863 [309]. The PIDF constitutes a common profile for presence so that various protocols, not only SIP, can use it to transport presence information.

The PIDF is designed with a minimalist approach (i.e., it includes a minimal set of features to fulfill the basic requirements). This minimal approach guarantees the reusability of the PIDF with different protocols. On the other hand, the PIDF is highly extensible, so it is possible to extend the format whenever there is a need to cross beyond the minimal model. Some extensions are being designed aimed at providing a more accurate view of the presence of a presentity.

The PIDF encodes the presence information in an XML document that can be transported, like any other MIME document, in presence publications (PUBLISH transactions) and presence notifications (NOTIFY transactions) operations. The PIDF defines a new MIME media type `application/pidf+xml` to indicate the type of application and encoding.

19.5.1 Contents of the PIDF

A PIDF document contains the presence information of a presentity. This information consists of a number of elements, each one referred to as a tuple. Each tuple includes the presentity's status (open or closed, meaning online or offline, respectively), an optional contact element that provides a contact URI, an optional note, an optional timestamp, and possibly other element extensions.

It should be noted that the PIDF only defines the *open* status and the *closed* status, which for most applications is not enough. The PIDF lets extensions define other statuses such as “at home”, “on the phone”, “away”, etc.

Figure 19.9 shows an example of the PIDF of the presentity identified as `pres:alice@example.com`. Her only tuple reveals that she is online for communications. She is providing a contact in the form of a TEL URI [295], and a note indicating that this is her cellular phone.

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
           entity="pres:alice@example.com">

  <tuple id="qoica32">
    <status>
      <basic>open</basic>
    </status>
    <contact priority="0.9">URI}tel:+1555876543</contact>
    <note>My cell phone</note>
  </tuple>
</presence>
```

Figure 19.9: Example of the PIDF

19.6 The Presence Data Model for SIP

Since the PIDF is protocol-agnostic, it does not go deep enough to identify what are the pieces of information represented in it. Certainly the PIDF represents presence information as a series of tuples, but it does not clearly indicate what a tuple is suppose to model, nor does it indicate how to map tuples to the various protocol elements available in SIP.

RFC 4479, “A Data Model for Presence” [271], tries to cover this vacuum by providing a model that maps tuples to SIP communication systems. The model is centered around three different aspects of a presentity.

Service. A communications service, such as instant messaging or voice over IP, is a system for interaction between users that provides certain modalities or content.

Device. A communications device is a physical component that a user interacts with in order to initiate or receive communications. Examples are a phone, PDA, or PC.

Person. The end-user, and for purposes of presence, is characterized by states, such as “busy” or “sad”, which have an impact on their ability and willingness to communicate.

Figure 19.10 illustrates the presence data model. The model considers that a presence entity, or *presentity*, is characterized by four different data components: the presentity URI, the person, the service, and the device, with each (except for the presentity URI) containing some data associated to the person, service, or device. The presence data model stresses the importance of the presence data reported, not of the data component that reported it. As an example, a mobile phone (a device) might be reporting that the user (the person data

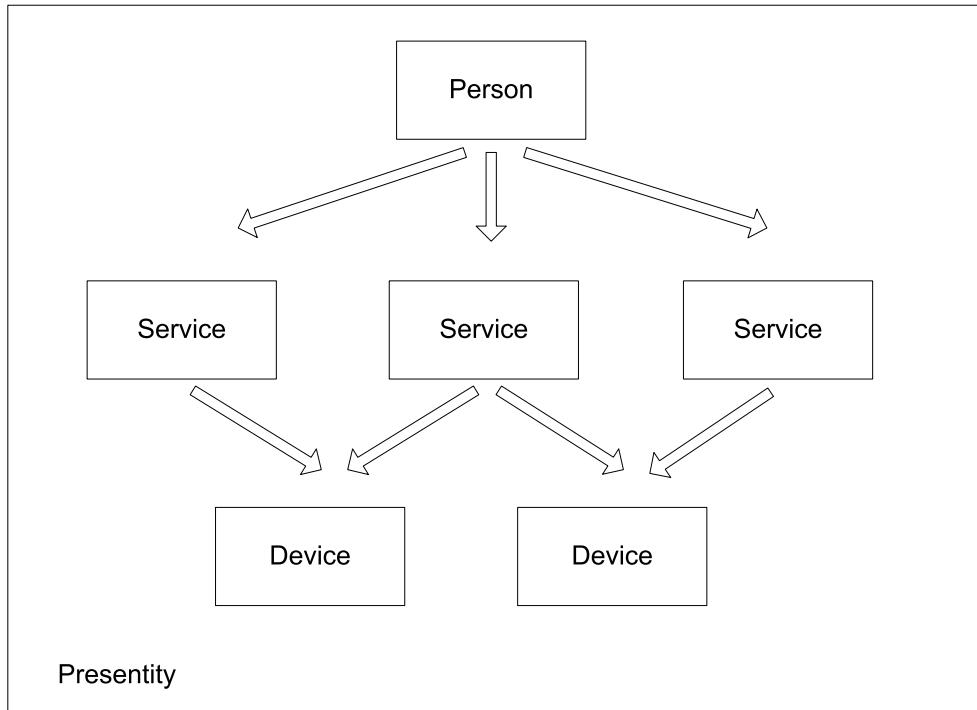


Figure 19.10: The SIP presence data model

component) is busy, independently of the fact that it is a mobile phone (a device) reporting that information.

The SIP presence data model considers that each presentity might have one or more presentity URIs. A common case is a presentity whose presence information is represented with a *pres* URI (see Section 19.1.1), a *sip* URI, and a *sips* URI.

The person data component provides information about the user himself. Two different aspects are considered: characteristics of the user and their status. Characteristics refer to static user data that does not change often with time, such as birthday or height. Status refers to dynamic information about the user, such as the user's activities (the user is on the phone, in a meeting, etc.), or the mood (the user is sad, happy, etc.).

The SIP presence model allows only one person data component per presentity, although it allows the person component to refer to something that behaves as a person but it is not exactly a person, for example, a group of assistants in a call center or an animal.

Presentities access services, and the willingness of presentities to communicate with some services is modeled with the service data component. A service can include: videotelephony, push-to-talk, instant messaging, etc. Like the person data component, the service data component can be described in terms of characteristics (static service data) and status (dynamic service data). Characteristics of the service include, for example, the SIP methods supported by the service, or other capabilities that represent the service (e.g., audio, video). Status includes, for example, whether the user is willing to communicate with that service or not. Services can also describe a URI that can be invoked to reach the service.

The last data component is the device data component. Devices model the physical platform in which services execute. Mobile phones, personal computers, and personal digital assistants are all examples of devices. Like services and persons, devices are described in terms of characteristics and status. Characteristics include the display size, number of colors, etc. Status includes the remaining battery load and the geographical location of the device. Devices are identified by a device ID, which is a Uniform Resource Name (URN) [216] that temporarily uniquely identifies the device. The device ID could be an International Mobile Equipment Identity (IMEI), an Electronic Serial Number (ESN), or a Medium Access Control (MAC) address.

19.7 Mapping the SIP Presence Data Model to the PIDF

Since the PIDF was created earlier than the SIP presence data model, and since the purpose of PIDF is to become the common minimum denominator across different presence systems, there is a need to map the SIP presence data model to the existing PIDF. The idea is to reuse the PIDF in its current state where possible, and extend it when required.

Mapping of the SIP presence data model to the PIDF is achieved by reusing the existing XML elements in the PIDF, with clarified semantics according to the SIP presence data model, and by extending the PIDF with new elements to accommodate the new data components.

We describe the mapping of the SIP presence data model to the PIDF with the help of Figure 19.11. The presence root element contains an `entity` attribute that, in the case of the SIP presence data model, is the `presence` URI.

The `tuple` element of the PIDF is used to describe the service data component of the SIP presence. A child `status` element is used to describe the characteristics or dynamic information of the service. The `contact` element of the PIDF contains the service URI, a URI that indicates how the user can be contacted for that particular service. The static service information is new extension elements that become children of the `tuple` element.

A new `person` element, a sibling of `tuple`, is created to contain the person data component. A new child `status` element carries the dynamic person information, whereas new extension elements carry the static information.

Like `person`, a new `device` element, which appears as a child of `presence`, is created to convey the device data component. The `device` element also contains a `status` element that contains dynamic device information, and a number of extensions that contain the static device information. A `deviceID` element, child of `device`, contains the device ID.

In each of the mentioned `status` elements, a number of new child elements are created to contain the actual dynamic information. However, those are not represented in Figure 19.11 for the sake of clarity.

19.8 Rich PIDF

We have just described (Section 19.5) how the PIDF document defines a minimalist model to describe the presence information of a `presence`. In commercial systems this minimalist model does not give enough detailed information. For instance, Alice might be interested in informing her watchers that she is online but not willing to accept any form of communication because she is driving. Unfortunately, the PIDF alone does not provide us with the semantics to express such information.

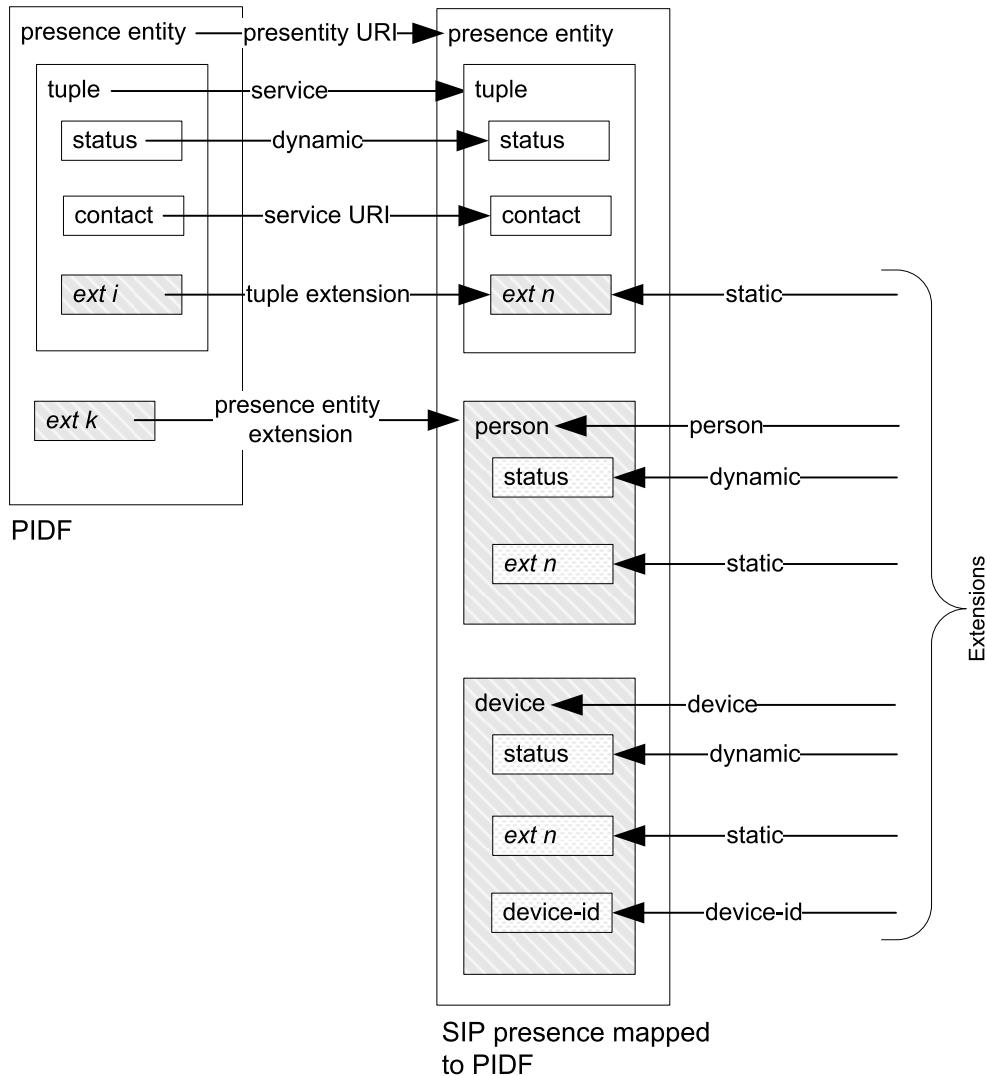


Figure 19.11: The SIP presence data model mapped to the PIDF

The Rich Presence Information Data Format (RPID) is an extension to the PIDF that allows a presentity such as Alice to express detailed and rich presence information to her watchers. Like the PIDF, RPID is encoded in XML. RPID is backward compatible with the PIDF. If watchers do not understand the RPID extension, they can at least obtain the minimal information from the PIDF document. The RPID extension is specified in RFC 4480 [302].

19.8.1 Contents of the RPID

A presentity such as Alice can set her rich presence information by manually operating on the appropriate setting of her presence software. However, RPID allows an automaton that has

access to the presentity's presence information to set such information up automatically. For instance, a calendar application can automatically set the presentity's presence information to "online – in a meeting" when the presentity's agenda indicates so. A SIP phone can automatically update the presentity's presence information to indicate that the presentity is engaged in a call when the presentity answers the phone.

Let us take a look at the type of information that the RPID is able to carry. The RPID extensions are applicable to person, services (tuple), and devices according to the presence data model.

RPID defines an **activities** element. It contains one or more **activity** elements that indicate the activity the presentity is currently undertaking. The specification allows the **activity** element to express that the presentity is on the phone, away, has a calendar appointment, is having breakfast, lunch, dinner, a meal, is not at work due to a national or local holiday, at work, in a meeting, steering a vehicle, in transit, traveling, on vacation, sleeping, just busy, in a performance, playing, presentation, watching TV, on a permanent absence, or performing some unknown activity. The list of values is expandable, so future extensions can add new values when needed.

A **class** element allows the presentity to group similar person elements, devices, or services that belong to the same class. The presentity allocates the class to a tuple. The PA can use this information to filter tuples according to the class.

A **deviceID** element contains an identifier of the device that provides a particular service. It allows us to differentiate different devices that are contributing to the same presence information. The device identifier is a URN [216] that identifies the device. So, for example, it could be an IMEI, an ESN, or a MAC address.

The **mood** status element is able to indicate the mood of a person (e.g., sad, happy, afraid, confused, impressed, offended, etc.).

The RPID includes two elements that contain information related to the place where the person is located. On the one side, **place-is** status elements contain properties of the place, e.g., a noisy environment, dark, quiet, too bright, uncomfortable, or inappropriate. **place-type** status elements allows our presentity, Alice, to indicate the type of place she is currently in. The possible initial values (usually extensible) are home, office, library, theater, hotel, restaurant, school, industrial, quiet, noisy, public, street, public area, aircraft, ship, bus, train, airport, station, mall, outdoors, bar, club, cafe, classroom, convention center, cycle, hospital, prison, underway, or unknown.

The RPID also includes a **privacy** element that indicates the type of media that the presentity will be able to safely receive with privacy, e.g., without third parties being able to intercept. The possible values include: audio, video, or text, indicating, respectively, that the presentity can receive audio, video, or text without others intercepting that type of media.

The **relationship** element in the RPID indicates the type of relationship the presentity has with an alternative contact. The possible values can indicate that an alternative contact is part of her family, friend, an associate, assistant, supervisor, self, or unknown.

The **service-class** element indicates whether the service is an electronic, postal, or delivery service, or describes in-person communications.

The **sphere** element in the RPID indicates the current state and role the presentity plays. Possible values are home, work, or unknown. This is useful information that allows the presentity to set visibility rules when she is playing a certain role. For instance, a member of the family may have access to additional information, such as a home webcam URI, when the presentity sets the **sphere** to home, while co-workers will not have access to this information.

The `status-icon` element contains a pointer (e.g., an HTTP URI) to an icon representing the person or service.

The `time-offset` element is able to express the offset in minutes from UTC at the user's current location.

The `user-input` element allows us to express human user input or the usage state of the service or device. It can contain either the value “active” or “idle”, including an optional `last-input` attribute that indicates the origin time of such a state.

Figure 19.12 shows an example of the rich presence information that Alice provides to her watchers. The presence information is encoded according to the PIDF with the extensions defined by the presence data model and RPID. Alice is providing her presence information from a PC. The device is providing the idle state since a point in time. Alice indicates that she is in the away state, at home, in a quiet environment for receiving communications, and is in a happy mood.

19.9 CIPID

The Contact Information in Presence Information Data Format (CIPID) is an extension to the PIDF that provides additional information about a presentity or a tuple, such as references to her business card, home page, map, sound, display name, and icon. Typically these are extensions to the `person` element in the presence data model, although the `tuple` element can also be extended with CIPID information in some cases, such as when the information describes a service referring to another person (e.g., when the person has a given relationship different than “self”). The CIPID is specified in RFC 4482 [296].

CIPID adds new `card`, `display-name`, `homepage`, `icon`, `map`, and `sound` elements to the `person` or `tuple` elements in the PIDF.

The `card` element contains a URI that points to a business card stored in LDIF (LDAP Data Interchange Format, specified in RFC 2849 [156]) or vCard (specified in RFC 2426 [118]) format.

The `display-name` element adds a name to the tuple or presentity that the presentity suggests to the watcher's user interface to display.

The `homepage` element contains a URI that points to the web home page of the presentity. The `icon` element contains a URI that points to an image of the presentity.

The `map` element contains a URI that points to a tuple's or presentity's map. It could be a GIF or PNG file, but also a GIS document.

The `sound` element contains a URI that points to a tuple's or presentity's sound. The format of such a file is not standardized, but it is recommended to support MP3.

Figure 19.13 shows an example of Alice's PIDF document that includes CIPID information in the `person` element. Alice is publishing pointers to her business card, home page, icon, map, and sound.

19.10 Timed Presence Extension to the PIDF

We have seen that the PIDF together with the RPID provides the current status of a presentity. However, they cannot provide information about past or future actions that the presentity had taken or will take. For instance, a presentity may start a meeting in the next half an hour. If a presentity publishes this information, watchers may decide to postpone interaction with the presentity until that meeting is over. The Timed Presence extension is specified in

```

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
    xmlns:r="urn:ietf:params:xml:ns:pidf:rpid"
    entity="pres:alice@example.com">

    <tuple id="3bfua">
        <status>
            <basic>open</basic>
        </status>
        <dm:deviceID>urn:device:001349038B74</dm:deviceID>
        <r:service-class><r:electronic/></r:service-class>
        <r:status-icon>
            http://www.example.com/alice/icon.jpg
        </r:status-icon>
        <contact priority="0.8">
            sip:alice@pc.example.com
        </contact>
        <timestamp>2005-06-05T07:52:14Z</timestamp>
    </tuple>

    <dm:device id="vjsa43">
        <r:user-input idle-threshold="300"
            last-input="2005-06-03T00:23:21">idle</r:user-input>
        <dm:deviceID>urn:device:001349038B74</dm:deviceID>
        <dm:note>PC</dm:note>
    </dm:device>

    <dm:person id="alice">
        <r:activities><r:away/></r:activities>
        <r:mood><r:happy/></r:mood>
        <r:place-is>
            <r:audio>
                <r:quiet/>
            </r:audio>
            <r:video>
                <r:quiet/>
            </r:video>
        </r:place-is>
        <r:place-type>
            <r:home/>
        </r:place-type>
        <r:privacy>
            <r:audio/>
            <r:video/>
            <r:text/>
        </r:privacy>
        <r:sphere>
            <r:home/>
        </r:sphere>
    </dm:person>
    <note>I have some visitors at home</note>
</presence>

```

Figure 19.12: Example of the RPID

```

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
           xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
           xmlns:ci="urn:ietf:params:xml:ns:pidf:cipid"
           entity="pres:alice@example.com">

    <tuple id="39dsaq">
        <status>
            <basic>open</basic>
        </status>
        <contact>sip:alice@ws3.example.com</contact>
        <timestamp>2005-06-12T18:53:02Z</timestamp>
    </tuple>

    <dm:person id="dpoia1">
        <ci:card>http://example.com/alice/card.vcd</ci:card>
        <ci:homepage>http://example.com/alice</ci:homepage>
        <ci:icon>http://example.com/alice/icon.gif</ci:icon>
        <ci:map>http://example.com/alice/map.png</ci:map>
        <ci:sound>http://example.com/alice/mysound.mp3</ci:sound>
        <dm:timestamp>2005-06-12T18:53:02Z</timestamp>
    </dm:person>
</presence>

```

Figure 19.13: Example of the CIPID

RFC 4481 [298] and allows a presentity to express what they are going to be doing in the immediate future or actions that took place in the near past.

A new `timed-status` element that contains information about the starting time of the event is added to the PIDF tuple element, or the data model person or device elements. The starting time of the event is encoded in a `from` attribute, whereas an optional `until` attribute indicates the time when the event will stop.

Figure 19.14 shows an example of the timed status extension. Alice is publishing that she will be offline from 13:00 to 15:00. Let us imagine that it is 12:45 when a watcher gains access to this information. The watcher wants to interact (through a call or instant messaging) with Alice, but the interaction may take more than 15 minutes. Since Alice will be offline in 15 minutes, perhaps due to a scheduled meeting, the watcher is able to delay the interaction with Alice until 15:00 when she will most likely be online again.

19.11 Presence Capabilities

We have seen in previous sections how presentities can express their presence status including online status, contact address, device capabilities, etc. When we described the basic operation of SIP we explored the Caller Preferences and User Agent Capabilities extension to SIP (see Section 4.12). This extension is concerned with the registration and session establishment processes in SIP. It seems natural to mimic that extension for the presence publication and watcher notification procedures, so that presentities could indicate in their presence information the same features that they would otherwise express in a registration.

```

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
           xmlns:ts="urn:ietf:params:xml:ns:pidf:timed-status"
           entity="pres:alice@example.com">

    <tuple id="qoica32">
        <status>
            <basic>open</basic>
        </status>
        <ts:timed-status from="2004-02-15T13:00:00.000+02:00"
                         until="2004-02-15T15:00:00.000+02:00">
            <basic>closed</basic>
        </ts:timed-status>
        <contact>sip:alice@example.com</contact>
    </tuple>
</presence>
```

Figure 19.14: Example of the *timed status* extension

Any watcher receiving a notification about the presentity's presence status could also receive the information about the presentity's supported features. This has the advantage that, in case a watcher wants to initiate any form of communication with a presentity, the watcher knows in advance the capabilities supported at the remote end. For instance, if Alice knows that Bob is using an application (service) that does not support text but does support audio and video, she may want to initiate an audiovisual session, rather than sending an instant message.

This is exactly what the presence capabilities extension does. The Internet-Draft “SIP User Agent Capability Extension to the Presence Information Data Format (PIDF)” [208] provides an extension to the PIDF that maps the caller preferences features (defined in RFC 3840 [288]) to new XML elements that are part of a PIDF document. These new elements express whether the presentity is reachable on a mobile or fixed device, whether it supports audio or video capabilities, the list of supported SIP methods and SIP event packages, etc.

In addition to all of the features defined in RFC 3840 [288], the Presence Capabilities allow us to express a few other features, such as “type”, “message”, and “language” that are not defined in RFC 3840 [288].

Presence capabilities are subdivided into service capabilities and device capabilities. Service capabilities characterize features related to services, for example, whether the service supports audio, video, messaging, full duplex operation, etc. Device capabilities are features that characterize a physical device, for example, whether a device is mobile or fixed. As the reader may expect, the presence capabilities are built on top of the presence data model and, as such, expand the tuple and device elements with service capabilities and device capabilities, respectively.

So presence capabilities define two new elements that act as containers of service and device capabilities: these are the `servcaps` and `devcaps` elements, respectively. Each of these elements contains a collection of new elements representing a feature that characterizes

the service or the device. Let us take a more detailed look at the service and device capabilities.

19.11.1 Service Capabilities

Service capabilities are enclosed in a `servcaps` XML element. The `servcaps` element has to be a child of the `tuple` XML element defined in the PIDF, since `tuple` elements are meant to describe services, according to the presence data model. A `servcaps` XML element can contain a number of `audio`, `application`, `data`, `control`, `video`, `text`, `message`, `type`, `automata`, `class`, `duplex`, `description`, `event-packages`, `priority`, `methods`, `extensions`, `schemes`, `actor`, `isfocus`, and `languages` elements.

The `audio`, `application`, `data`, `control`, `video`, `text`, and `message` elements are boolean indications (true or false) of the support of the service for audio, application, data, control, video, text, or message streams, respectively. Note that these elements refer to the type of media streams supported by the service, not by the device. So if there are two services (e.g., two applications) running on the same device, and one supports only audio media streams, it will indicate this, even when the device supports other capabilities (e.g., video).

The `type` element indicates possible MIME types that the service is able to accept. These MIME types are typically indicated in a `Content-Type` header field in SIP.

The `class` element indicates whether the service is used for business communications or personal communications.

The `duplex` element indicates whether a communications service can simultaneously send and receive media (value of `full`), alternate between sending and receiving media (value of `half`), can only receive media (value of `receive-only`), or only send media (value of `send-only`).

The `description` element contains a textual description of the service. An `xml:lang` attribute indicates the language of the textual description. It is possible to include several descriptions in different languages.

The `event-packages` element contains a list of SIP event packages supported by the service. As in the types of supported media streams, the `event-packages` elements refer to the SIP event packages supported by the service (`tuple` XML element) under `description`, not all of the SIP event packages supported by the union of all of the applications running in the device.

The `priority` element indicates the call priorities that the service is able to handle. Priorities are expressed as integers, and the application can indicate ranges of supported and non-supported priorities.

The `methods` element indicates the list of supported (and, if available, the non-supported too) SIP methods in the service. Like the rest of the service capabilities, this element refers to the list of supported methods in the service (`tuple` element in the PIDF) where it appears, and it does not refer to the list of supported methods by the union of all of the applications or services running in the device.

The list of supported and non-supported SIP extensions that the service implements is indicated in the `extensions` element. The list of supported SIP extensions corresponds are those option-tags that can appear in `Supported` or `Require` SIP header fields.

The `schemes` element indicates the list of URI schemes that the service is able to handle (e.g., `sip`, `sips`, `tel`, `http`, etc.).

The `actor` element allows the presentity to indicate the type of entity residing behind the service, for example, if it is the principal associated with the service, an attendant of substitute

of the principal, a message taker (such as a voice mail system), or some person or automata that can provide further information about the principal.

The `isfocus` element indicates that the service is a centralized conference server.

The `languages` element indicates the ability of the service to display human languages.

19.11.2 Device Capabilities

Device capabilities are enclosed in a `devcaps` XML element. The `devcaps` element has to be a child of the `device` XML element defined in the presence data model. A `devcaps` XML element can contain `mobility`, `priority`, and `description` elements.

The `mobility` element merely indicates whether the device is `fixed` or `mobile`.

The `priority` reflects the priorities of the call that the device is willing to handle. It can contain a range of values that are accepted by the device.

The `description` element indicates a textual description of the device.

19.11.3 An Example of the Presence Capabilities Document

Figures 19.15 and 19.16 show an example of a PIDF document extended with the presence data model and the presence capabilities (the example has been split into two parts for presentation purposes).

The presentity, Alice, is indicating her presence capabilities related to both the service (i.e., the `servcaps` element under the `tuple` element), and the device capabilities (i.e., the `devcaps` element that extends the `device` element defined in the presence data model). Alice describes the capabilities that her service supports: audio, video, text, and full duplex capabilities. She then describes the list of SIP methods, event packages, SIP extensions, and URI schemes that her service is able to handle. In the devices capabilities section she indicates that she is using a mobile device.

19.12 Geographical Location in Presence

We indicated earlier that the PIDF is highly extensible, and we have already seen a few extensions that add additional information to the PIDF. An interesting extension is the so-called *location object*, specified in RFC 4119 [250] and extended in RFC 5139 [310], and commonly known as PIDF Location Object (PIDF-LO). The PIDF-LO allows geographical or civic location to be added to the PIDF. This opens up a large number of applications, such as location-based services and emergency calls. The idea is very simple: when a PUA publishes presence information, it also publishes the current location of such PUA, either as a geodetic location information or as a civic location information. Duly authorized watchers receive the presentity's presence information along with the location information, and can use appropriately the location information, for example, visualize it in real time in a map, or provide an instant message with the closest Italian restaurants in the neighborhood. The possibilities are endless.

PIDF location objects can indicate Global Positioning System (GPS) coordinates or a civic address location. Therefore, there are two sub-formats of the geographical location object. Implementations are mandated to implement the GPS coordinates sub-format and can optionally implement the civic address location sub-format. If they implement the civic address location sub-format, the revised civic PIDF-LO, specified in RFC 5139 [310], is used.

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:cap="urn:ietf:params:xml:ns:pidf:caps"
    xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
    entity="pres:alice@example.com">

    <tuple id="3bfua">
        <status>
            <basic>open</basic>
        </status>
        <cap:servcaps>
            <cap:audio>true</cap:audio>
            <cap:video>true</cap:video>
            <cap:message>true</cap:message>
            <cap:duplex>
                <cap:supported>
                    <cap:full/>
                </cap:supported>
            </cap:duplex>
            <cap:description xml:lang="en">
                General service
            </cap:description>
            <cap:methods>
                <cap:supported>
                    <cap:INVITE/>
                    <cap:ACK/>
                    <cap:CANCEL/>
                    <cap:BYE/>
                    <cap:MESSAGE/>
                    <cap:PRACK/>
                    <cap:UPDATE/>
                </cap:supported>
            </cap:methods>
            <cap:event-packages>
                <cap:supported>
                    <cap:reg/>
                </cap:supported>
            </cap:event-packages>
            <cap:extensions>
                <cap:supported>
                    <cap:100rel/>
                    <cap:precondition/>
                </cap:supported>
            </cap:extensions>
        </cap:servcaps>
    </tuple>
</presence>
```

Figure 19.15: Example of the presence capabilities extension (part 1)

```

<cap:schemes>
  <cap:supported>
    <cap:s>sip</cap:s>
    <cap:s>URI}sips</cap:s>
  </cap:supported>
</cap:schemes>
</cap:servcaps>
<contact>sip:alice@example.com</contact>
</tuple>

<dm:device id="92n2kljnd2">
  <cap:devcaps>
    <cap:mobility>
      <cap:supported>
        <cap:mobile/>
      </cap:supported>
    </cap:mobility>
  </cap:devcaps>
  <dm:deviceID>urn:device:039fa209</dm:deviceID>
</dm:device>
</presence>

```

Figure 19.16: Example of the presence capabilities extension (part 2)

A geographical location object starts with a `geopriv` element followed by a child `location-info` element, which are placed as children elements to the `status` element, a child of the `tuple` element in PIDF. The `location-info` element has a child element that depends on the chosen sub-format. If the location object is expressed in GPS coordinates, the `location-info` contains a child `location` element further children elements that encode the GPS coordinates. These GPS coordinates are encoded according to the Geography Markup Language [225] specified by the Open Geospatial Consortium.

Figure 19.17 shows an example of a PIDF-LO that embeds a geographical location object expressed in GPS coordinates. The GPS coordinates are expressed with elements belonging to the GML namespace and are prepended with a `gml` prefix.

In addition to the `location-info` element, the `location` element also contains a `usage-rules` element that contains the privacy policy related to the location object. In the example of Figure 19.17 the policy indicates, in a `retransmission-allowed` element, that the recipient of the PIDF is authorized to re-distribute the location information to third parties such as watchers. A `retention-policy` indicates the absolute time when the recipient should discard the location information, perhaps because it might be obsolete.

An example of a PIDF-LO containing a civic address location is shown in Figure 19.18. The civic location is expressed in a number of elements that belong to the `civicLoc` namespace and are prepended by the `c1` prefix. The example in Figure 19.18 indicates the civic location: Linnoitustie 6, 4th floor, 02600 Espoo, Finland. The privacy policy rules that determine the privacy of the civic address location are encoded in the `usage-rules` element.

```

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
  xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
  xmlns:gml="urn:opengis:specification:gml:schema-xsd:feature:v3.0"
  entity="pres:alice@example.com">

  <tuple id="a3lkjdaf">
    <status>
      <gp:geopriv>
        <gp:location-info>
          <gml:location>
            <gml:Point gml:id="point1" srsName="epsg:4326">
              <gml:coordinates>37:46:30N 122:25:10W</gml:coordinates>
            </gml:Point>
          </gml:location>
        </gp:location-info>
      <gp:usage-rules>
        <gp:retransmission-allowed>yes</gp:retransmission-allowed>
        <gp:retention-expiry>
          2007-11-06T20:25:29Z
        </gp:retention-expiry>
      </gp:usage-rules>
    </gp:geopriv>
  </status>
  <contact priority="1.0">sip:alice@example.com</contact>
  <timestamp>2007-11-05T20:25:29Z</timestamp>
  </tuple>
</presence>
```

Figure 19.17: Example of a location object expressed with GPS coordinates

19.13 Watcher Information

In the previous sections we addressed two main problems that the presence service solves: how presentities can publish their presence information and how watchers can be updated when changes in such presence information occur. There is a further problem that we have not yet addressed: how can a presentity such as Alice (or any other authorized observer) be informed of the watchers who are subscribed to her presence information. This information is needed to authorize such watchers and provide them with the adequate amount of presence information. In addition to the presentity, it is also possible that other services or authorized observers receive notifications of the watchers of a particular presentity.

In order to solve this problem the IETF has created a *Watcher info* event template-package defined in RFC 3857 [270]. Event template-packages are event packages that can be applied to any other event package. The watcher info event template-package provides the subscriber with information about who is watching the subscribed resource. If the watcher info event template-package is applied to presence, the value of the Event header field is set to presence.winfo.

```

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
  xmlns:gp="urn:ietf:params:xml:ns:pidf:geopriv10"
  xmlns:cl="urn:ietf:params:xml:ns:pidf:geopriv10:civicLoc"
  entity="pres:alice@example.com">

  <tuple id="a3lkjdaf">
    <status>
      <gp:geopriv>
        <gp:location-info>
          <cl:civicAddress>
            <cl:country>FI</cl:country>
            <cl:A1>Espoo</cl:A1>
            <cl:A6>Linnoitustie</cl:A6>
            <cl:HNO>6</cl:HNO>
            <cl:FLR>4</cl:FLR>
            <cl:PC>02600</cl:PC>
          </cl:civicAddress>
        </gp:location-info>
      <gp:usage-rules>
        <gp:retransmission-allowed>yes</gp:retransmission-allowed>
        <gp:retention-expiry>
          2007-11-06T20:25:29Z
        </gp:retention-expiry>
      </gp:usage-rules>
    </gp:geopriv>
    </status>
    <contact priority="1.0">sip:alice@example.com</contact>
    <timestamp>2007-11-05T20:25:29Z</timestamp>
  </tuple>
</presence>

```

Figure 19.18: Example of a PIDF-LO containing civic address location

Let us see how watcher info works with the help of Figure 19.19. A subscriber, which typically also acts as a PUA, sends a SUBSCRIBE request (1) to their PA with an Event header field set to presence.winfo. The PA authenticates and authorizes the subscription and answers with a 200 (OK) response (2). The PA also sends a NOTIFY request (3) to indicate the status of the subscription. This NOTIFY can also contain an XML document containing the list of watchers of the presence information of the presentity. The PA will keep the subscriber updated, using NOTIFY requests (5), about changes in the list of watchers of presence information. That is, it will inform Alice every time a new watcher subscribes or unsubscribes to the presentity's presence information.

Being informed about watchers is the motivation behind creating watcher info functionality, but it is not the only one. Perhaps the main motivation is related to presence authorization. When a watcher subscribes to a presentity's presence information the presentity needs to

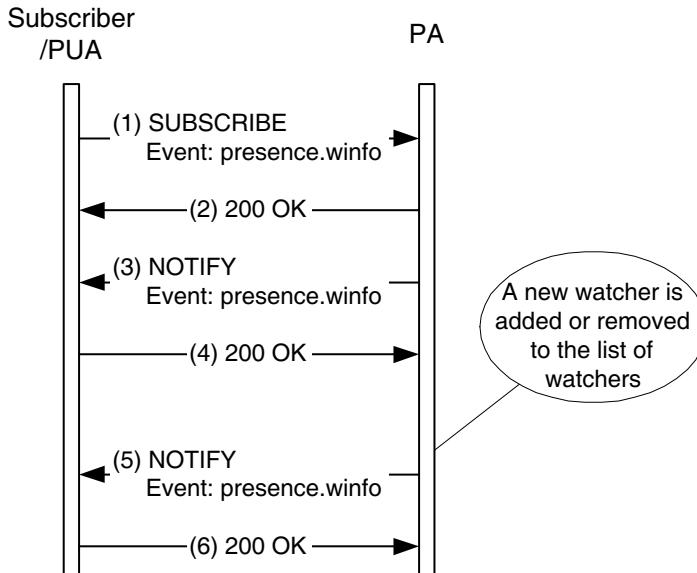


Figure 19.19: A subscriber PUA compiles the list of her watchers

authorize the subscription. In order for the presentity to be able to authorize watcher subscriptions the presentity must be aware of who is requesting to watch the presentity's presence information and what is the subscription status of each of these watchers. The watcher info event package provides the means to transport this information.

The watcher info event package is encoded in XML. The package defines a new MIME type application/watcherinfo+xml. When a SIP request or response contains a Content-Type header field set to application/watcherinfo+xml, it is indicating that the body is an XML document that contains watcher information.

Figure 19.20 shows an example of a watcher info XML document, where the presentity is Alice and the watcher is Bob. Bob has a subscription to Alice's presence, but it is in the pending state, and is just waiting for Alice to authorize the subscription. Watcher subscription authorization is done by means other than SIP, for example, with XCAP (see Chapter 17). Section 19.14 further discusses the authorization of watchers.

```
<?xml version="1.0"?>
<watcherinfo xmlns="urn:ietf:params:xml:ns:watcherinfo"
              version="0" state="full">
  <watcher-list resource="sip:alice@example.com"
                package="presence">
    <watcher id="342sd2" event="subscribe" status="pending">
      sip:bob@example.com</watcher>
    </watcher-list>
  </watcherinfo>
```

Figure 19.20: Example of a watcher info XML document

19.14 Watcher Authorization: Presence Authorization Rules

Presentities are informed that new watchers are trying to subscribe to their presence information via the `Watcher info` event package applied to presence, as we just described in Section 19.13. However, the `Watcher info` event package does not provide a mechanism for the presentity to convey authorization of watchers to the PA. Furthermore, it does not provide a mechanism for the presentity to indicate to the PA which parts of the presence information should be provided to that watcher.

The solution to address these problems comes in the format of presence authorization rules, specified in the RFC 5025 [276]. Presence authorization rules are effectively drafted in *presence authorization policy* documents. The remainder of this section describes the structure of such documents and how to convey them to the PA.

A presence authorization policy document is an XML document that the presentity creates to indicate to the PA which watchers are authorized to obtain the presentity's presence information, and which parts of that extensive long information they should receive. The document contains a set of rules that the PA can evaluate for decided with presence information is supplied to which watchers. The presentity sends presence authorization rules documents to the PA by using XCAP. We described XCAP in Chapter 17. The presentity can upload a presence authorization policy prior or after getting a notification (via the `watcher-info` event package).

19.14.1 Common Policy

Presence authorization policy documents are built upon a more generic policy documents called *common policy* documents, specified in RFC 4745 [304]. Common policy documents are not restricted to presence, so they can be used by any kind of service where there is a need to indicate a policy. In fact, common policy documents are meant to be extensible by other services, such as presence or location information. So, effectively, a presence authorization policy document is based on a common policy document and is extended with presence specific rules.

Let us start by taking a look at the structure of a common policy document first and then we look at the presence authorization policy extensions.

A common policy document is an XML document that contains the description of zero or more rules, collectively known as the *rule set*. The rule set determines the level of detail of information that is offered to a watcher. Each rule is structured into three parts: conditions, actions, and transformations.

The “condition” part of a rule determines whether the actions and transformations are executed or not, so the condition can be seen as the “if” statement of a conditional sentence. A condition contains a number of expressions. Each of them results in a TRUE or FALSE evaluation. If all of the expressions evaluate to TRUE, the condition is valid and applies to the requested information, so the server then applies the actions and transformations to the requested information. Conditions can be set, for example, on the identity or the domain of the watcher, the time of the day, or some specific detail of the presentity's presence information.

Actions and transformations are usually called “permissions” collectively. A permission can be seen as the “then” in a conditional statement. Permissions determine what a PA needs to do before supplying the requested information to the watcher.

The “actions” part in a rule indicates to the PA which action must be taken if the condition is fulfilled. Applied to presence, actions indicate whether the PA should accept the subscription or not.

The “transformations” part in a rule provide the PA with the means to express privacy filters. Once a condition is fulfilled and the actions are executed, the PA takes the input data, applies the transformations, obtains some output data, and offers that data to the watcher. Applied to presence, transformations can indicate, for example, which level of detail is provided to the watcher, or which XML elements in the PIDF are provided.

While the common policy defines the general structure of policy documents, its collections of rules, and the three main components of a rule (conditions, actions, and transformations), it does not go into detail in specifying what those conditions and transformations are. These are left for the specific usages of the common policy. In the case of presence, the specific usage of common policy documents is presence authorization policy documents.

19.14.2 Presence Authorization Policy Documents

Now that the concept of common policy has been described, let us take a look at presence authorization policy documents to illustrate the whole idea of authorization in the context of the presence service.

Figure 19.21 shows an example of a presence authorization policy document, which is built upon the common policy and the presence authorization rules. In this example, there is a single rule in the rule set. The rule contains the three parts: conditions, actions, and transformations. The rule can be interpreted in human parlance as follows: If the user is authenticated and their identity is either `sip:charlie@example.com` or `+1-972-555-232323`, then they are allowed to subscribe to the presence information of the presentity. The watcher receives the following parts of the presentity’s presence information:

- any device information contained in the `device` data component;
- the SIP and TEL URIs of the presentity;
- the whole the `person` data component;
- the `activities` element;
- the `mood` element;
- the `status-icon` element.

How is this achieved? In Figure 19.21 it can be seen that the document begins with a `ruleset` element that contains zero or more rules. In the example, this rule set contains a single rule, identified by the element `rule`. The rule contains `actions`, `conditions`, and `transformations` elements. The `actions` element contains a single expression, `identity`, which evaluates to TRUE if any of its child elements evaluates to true. In this case, the `identity` element contains two `one` elements with the identities of watchers where this rule applies. The `one` element is used to match a single identity, whereas a `many` element would have been used to match a collection of identities, for example, those belonging to a given domain.

Continuing with the example of Figure 19.21, then we find the `actions` element that defines the actions part. The `actions` element contains a single `sub-handling` element

```

<?xml version="1.0" encoding="UTF-8"?>
<cp:ruleset xmlns="urn:ietf:params:xml:ns:pres-rules"
  xmlns:pr="urn:ietf:params:xml:ns:pres-rules"
  xmlns:cp="urn:ietf:params:xml:ns:common-policy">

  <cp:rule id="1">
    <cp:conditions>
      <cp:identity>
        <cp:one id="sip:charlie@example.com"/>
        <cp:one id="URI}tel:+1-972-555-232323"/>
      </cp:identity>
    </cp:conditions>

    <cp:actions>
      <pr:sub-handling>allow</pr:sub-handling>
    </cp:actions>

    <cp:transformations>
      <pr:provide-devices>
        <pr:all-devices/>
      </pr:provide-devices>
      <pr:provide-services>
        <pr:service-uri-scheme>sip</pr:service-uri-scheme>
        <pr:service-uri-scheme>URI}tel</pr:service-uri-scheme>
      </pr:provide-services>
      <pr:provide-persons>
        <pr:all-persons/>
      </pr:provide-persons>
      <pr:provide-activities>true</pr:provide-activities>
      <pr:provide-mood>true</pr:provide-user-input>
      <pr:provide-status-icon>true</pr:provide-status-icon>
    </cp:transformations>
  </cp:rule>
</cp:ruleset>

```

Figure 19.21: Example of a presence authorization policy document

(for subscription handling) that indicates what the PA should do with the subscription in case the conditions evaluate to TRUE. The following are possible values of the `sub-handling` element:

block. The PA must reject the subscription

confirm. The PA must set the subscription to the pending state and await for further instructions from the presentity.

polite-block. The PA must set the subscription to the active state and it must provide a presence document that indicates that the presentity is unavailable.

allow. The PA must set the subscription to the active state

After the actions, the transformations follow, indicated by the `transformations` element. The transformations provide an indication to the PA for how to manipulate the raw presentity's presence information before it is supplied to the watcher, so transformations constitute a powerful privacy filter controlled by the presentity. There are two groups of transformations: one that grants access to the person, device, and service data of the presence information; another that grants access to specific elements of the presence information. In the former group, we find the `provide-devices`, `provide-services`, and `provide-persons` elements. In the latter group, the `provide-activities`, `provide-mood`, and `provide-status-icon`. Other elements that are not included in the example of Figure 19.21, but that could have indicated other transformations, include: `provide-class`, `provide-deviceID`, `provide-place-is`, `provide-place-type`, `provide-privacy`, `provide-relationship`, `provide-sphere`, `provide-time-offset`, `provide-user-input`, and `provide-note`. Furthermore, it is also possible to define new additional attributes either by extensions or by making use of the built-in `provide-unknown-attribute`. This allows an attribute to be selected by its namespace and name. There is also a `provide-all-attributes` that selects all of the attributes in the presentity's presence information.

19.14.3 Uploading Presence Authorization Policy Documents to the Presence Agent

When a presentity needs to upload or modify an existing presence authorization policy document, it uses XCAP to send it to the PA. We described XCAP in detail in Chapter 17. We saw that XCAP defines application usages for the applications that require to use XCAP, and that each one is identified by an AUID. In the case of presence authorization rules, the application usage for XCAP is also specified in the RFC 5025 [276]. XCAP provides versatility to the presentity to submit a complete presence authorization policy document, such as that we saw in Figure 19.21, as well as manipulate XML elements and attributes. So, if a document already exists, and the presentity wants to change the value of the `sub-handling` element to authorize the subscription, then the XCAP request can be fairly short.

19.14.4 Watcher Authorization: Complete Example

Figure 19.22 shows a complete sequence of events related to the authorization of watchers. The PUA first subscribes to its own watcher information in steps 1 to 4. A NOTIFY request (3) provides the presentity with information regarding all of their watchers and the authorization status of their subscriptions. At some point in time later, a new watcher tries to subscribe to the presentity's presence information. A SUBSCRIBE request (5) is received at the presentity's PA. The PA examines the current presence authorization policy document for that presentity, and determines that there are no rules where the condition matches the identity of the watcher. So, the PA first notifies the watcher (step 7) that the subscription is set to the "pending" state until further instructions are received from the

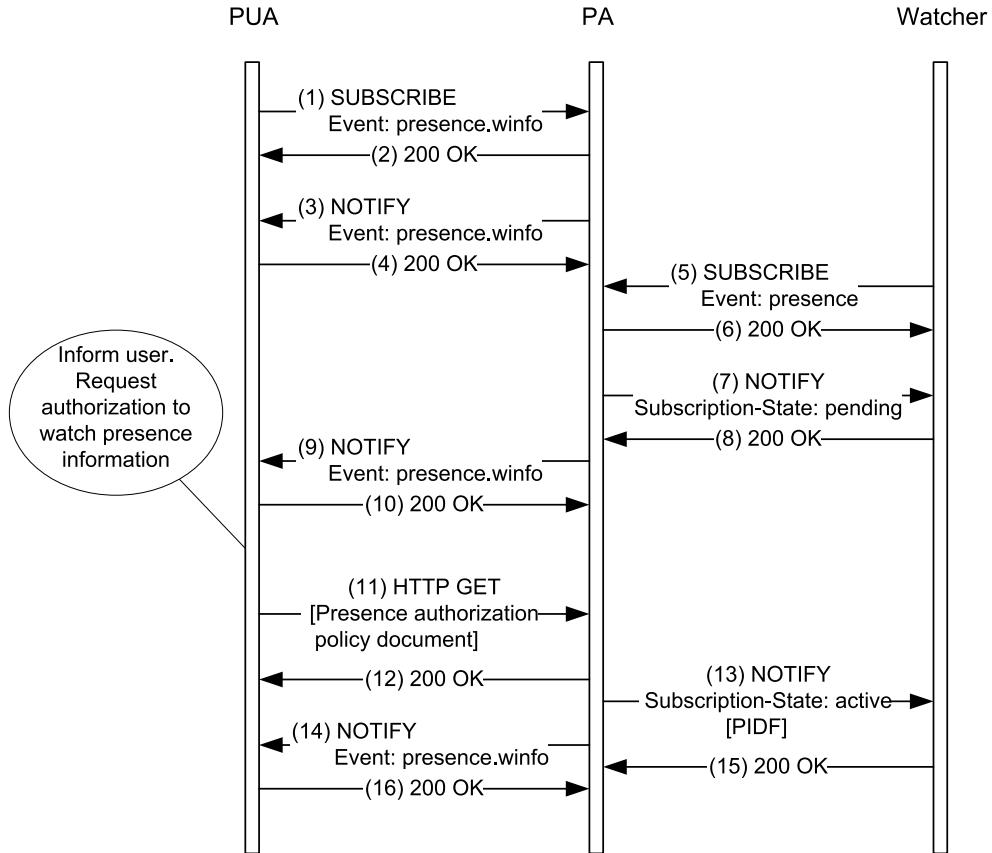


Figure 19.22: Watcher authorization: complete flow

presence. The PA also notifies the presentity (step 9) about this new subscription, indicating the “pending” status. The PUA eventually informs the user and asks permission to access the presence information. If the user grants the access, the PA performs an XCAP operation (step 11) to modify the presence authorization policy document. The modification consists of adding a new rule or modifying an existing rule by adding a new identity. When the PA receives the updated presence authorization policy document, it changes the subscription state of the watcher, evaluates any possible transformations, and provides a PIDF document that contains the authorized information in a NOTIFY request (13), which also contains a `Subscription-State` header field set to “active” to indicate the new subscription state. The PA also notifies the presentity in step 14, via the `watcher_info` event package, indicating the new watcher and its subscription state.

19.15 URI-list Services and Resource Lists

We saw in Section 19.3 that every time a watcher wants to subscribe to the presence information of a presentity the watcher needs to exchange a SUBSCRIBE transaction and

a NOTIFY transaction with the presentity's PA, just to set up the subscription. If the PA challenges the watcher the number of SIP transactions may be even larger. As a minimum the watcher always exchanges four SIP messages with the presentity's PA for each presentity the watcher is subscribed to.

For example, Alice wants to know the presence status of her friends Bob, David, and Peter. She acts as a watcher and sends a SUBSCRIBE request to each of their PAs (1), (3), (5), as shown in Figure 19.23. Later, she receives a NOTIFY request from each of them (7), (9), (11). If the watcher is subscribed to, say, the presence information of 100 presentities, the watcher's presence application needs to process a minimum of 400 SIP messages (perhaps more) every time the watcher initializes the presence application, and this is just to set up the subscriptions. Typically, NOTIFY requests do not contain the presentity's presence information at this stage. Obviously, this mechanism does not scale well, particularly in wireless environments: the number of messages required to set up the subscriptions should be independent of the number of presentities in the watcher's list.

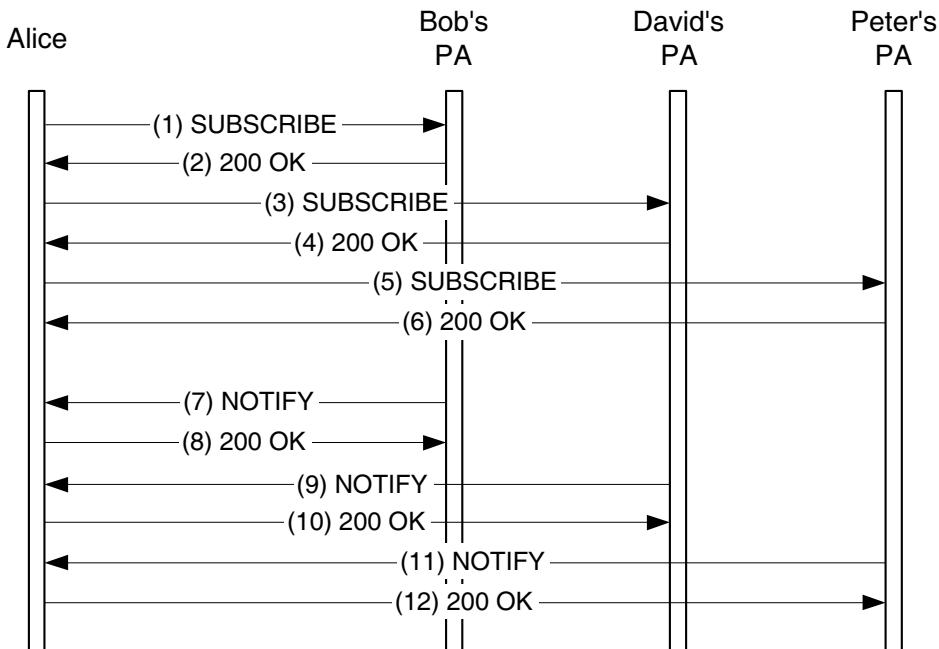


Figure 19.23: Long presence list without a URI-list service

In order to solve this problem the IETF has created the concept of *resource lists*. A resource list is a list of SIP URIs that is stored in a new functional entity called the *Resource List Server* (RLS), which is sometimes known as a *URI-list service* for SUBSCRIBE requests. The list is addressed with its own SIP URI. When the concept of resource lists is applied to the presence service it is also called a *presence list*. A presence list contains a list of all of the presentities a watcher is subscribed to.

A SIP URI-list service receives a request from a watcher and forwards it to multiple PUAs. SIP URI-list services used for subscriptions (described in RFC 4662 [265]) also

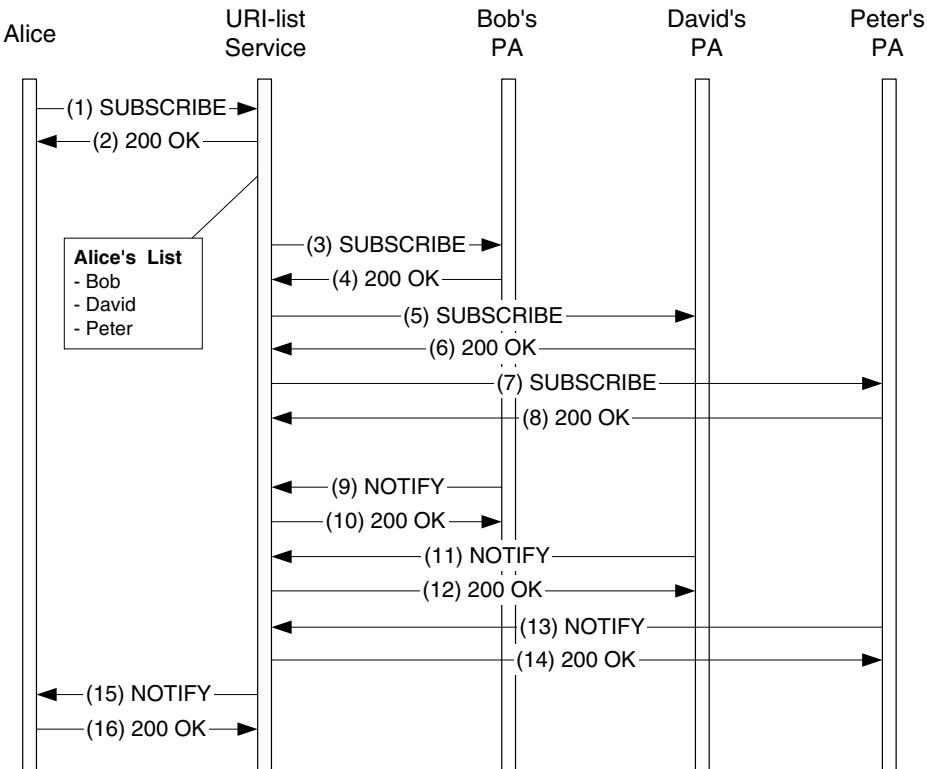


Figure 19.24: Resource list through a URI-list service

summarize the presence information received in NOTIFY requests. Figure 19.24 shows how this type of URI-list service works.

Instead of sending a SUBSCRIBE request to every user in her presence list, Alice sends a single SUBSCRIBE request (1) addressed to her presence list. The request is received by the SIP URI-list service, or Resource List Server. Alice has previously provided the URI-list service, using an out-of-band configuration mechanism (typically XCAP), with her presence list (which is a list of the presentity's URIs). The URI-list service sends a SUBSCRIBE request to every URI in the list (3), (5), (7). Later, when the URI-list service receives the NOTIFY requests from them (9), (11), (13), it aggregates the presence information and sends a single NOTIFY request (15) to Alice. This mechanism saves a considerable amount of bandwidth on Alice's access network; moreover, it requires much less processing power in the watcher.

The RLS mechanism defines a new MIME type `application/rlmi+xml`. This document contains a list of subscribed resources (presentities) that points to each of the PIDF documents of the presentities. As a consequence, NOTIFY requests contain a multipart/related document that comprises a collection of other embedded XML documents.

The use of presence lists rather than individual subscriptions has the side effect of not storing the presence list locally in the presence application (e.g., the computer or the IMS terminal), but in a network node. When Alice uses her presence application from a different terminal than usual, she can gain access to her presence list, since it is stored in the network. In her new terminal she just needs to configure the URI of her own presence list to gain access to it, rather than configuring all of the URIs of the presentities one by one. This allows seamless terminal roaming. For instance, it allows Alice to become a watcher when she is logged on to a PC, such as an Internet kiosk.

19.16 Presence Optimizations

We have discussed the presence service in this chapter, and we have seen several examples of presence publication, subscription, and notification. Presence publication and presence notification operations contain a PIDF/RPID XML document. PIDF/RPID documents are naturally large because they are rich in information. A watcher who is subscribed to a number of presentities may receive one of these XML documents every time the presentity's presence information changes. When presence information reaches a small device that has constraints in memory, processing capabilities, battery lifetime, and available bandwidth, the device may be overwhelmed by the large amount of information and might not be able to acquire or process it in real time.

3GPP engineers, aware of the constraints of cellular devices and networks, have worked with IETF engineers, the experts in protocol design, in order to find solutions to optimize the amount of presence information transmitted to presence clients. Obviously, there has to be a compromise between the amount of information sent, the frequency of the notifications, and the bandwidth used to send that information. Sending less information in presence documents may lead to users not receiving a good experience with presence systems used from wireless terminals. Sending presence information less periodically will lead to an inaccurate presence view of the presentities. It is important that the user receives accurate and rich presence information while the bandwidth used is reduced.

We have described the use of Resource List Servers, a mechanism that solves not only bandwidth problems but also terminal-roaming problems. In Section 4.15.1.1 we described a mechanism to throttle and control the rate of notifications. This mechanism is not only applicable to presence, but to any event package controlled by the event notification framework. Apart from these mechanisms, there are other presence-specific mechanisms that are intended to reduce the amount of presence information transmitted to watchers: partial notifications, which we describe in Section 19.16.1, and event filtering, which we describe in Section 19.16.3 .

19.16.1 Partial Notification of Presence Information

A partial notification is a notification that may carry a subset of the presentity's presence information. The partial notification mechanism defines a new XML body that is able to transport a partial or full state. The XML body is quite similar in structure to the PIDF, with the addition of a few new elements that indicate the version number, whether the document contains a full or a partial state, and whether a tuple has been completely removed from the presentity's presence information. The new XML body is identified with a MIME type application/pidf-partial+xml.

The mechanism is illustrated in Figure 19.25. A watcher subscribes to a presentity's presence information. The SUBSCRIBE request (1) contains an Accept header field indicating support for both the PIDF document and the partial PIDF document. A weight or preference is also indicated through the q parameter. The PA installs the subscription and sends a first NOTIFY request (3) that contains an XML document containing the full presence state, although encoded according to the rules of the partial notification. Therefore, the Content-Type header field of this NOTIFY is set to application/pidf-partial+xml. Later, when there is a need to update the watcher with new information the PA sends a new NOTIFY request (5), but in this case it contains only the changes (additions, modifications, or deletions) with respect to the full-state document.

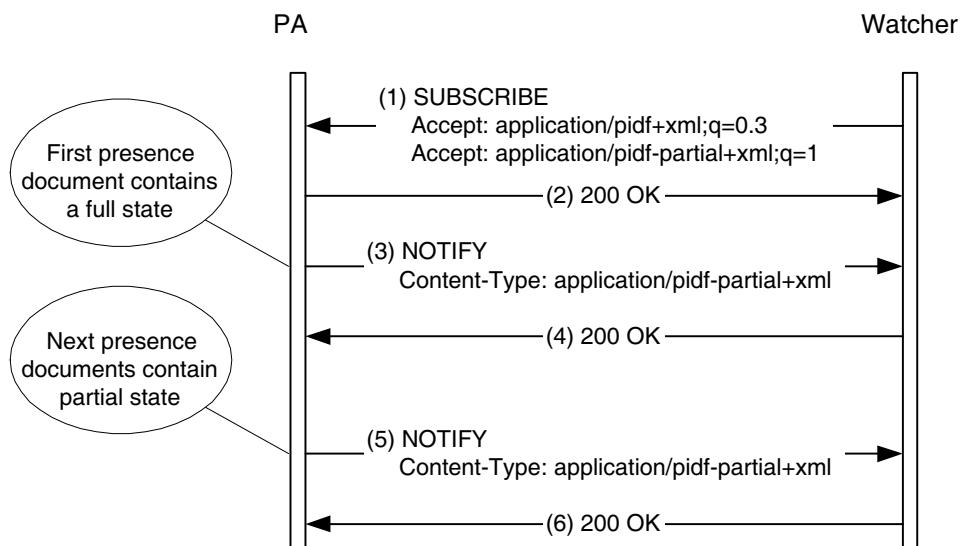


Figure 19.25: Partial notification

19.16.2 Partial Presence Publication

In a general case, when a presentity is publishing their presence information, the PUA sends a PUBLISH request that contains a PIDF document, as we described in Section 19.4. PIDF documents, when accompanied by extensions (e.g., presence data model, RPID, CIPID, presence capabilities, etc.) can have a considerable extension. However, in most cases, the amount of information that changes in a presentity is minimal. For example, the status is likely to change often, but not the rest of the information. So, it seems reasonable to have a mechanism where a presentity can publish only the changes of the presence information that have taken place with respect to an earlier publication. This is achieved by partial presence publication, specified in the Internet-Draft “Publication of Partial Presence Information” [223]. Partial publication uses a format very similar to PIDF documents, called partial PIDF documents, and specified in the Internet-Draft “Presence Information Data format (PIDF) Extension for Partial Presence” [209].

Partial PIDF reuses the XML patch operations framework that we described earlier in Section 17.7 in the context of XCAP. In this case, XML patch operations are applied to PIDF documents and transported with SIP. However, the principles of the XML patch operations apply equally.

Figure 19.26 describes the partial publication mechanism. A PUA first does a full publication by including a full PIDF document in a PUBLISH request (1). This is represented in Figure 19.27. The PUBLISH request (1) contains a partial PIDF document, identified by the value `application/pidf-diff+xml` in the Content-Type header field. It is worth noting that since this is an initial publication, the presence document contains full state, indicated by the root `<pidf-full>` element.

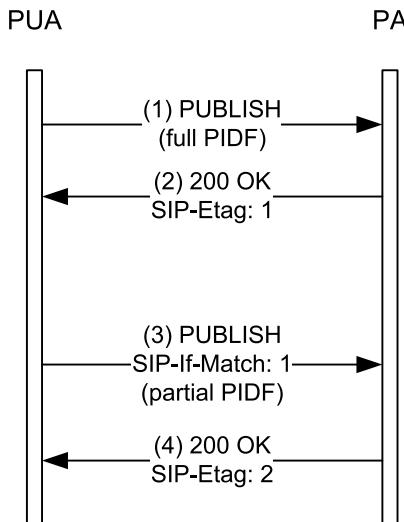


Figure 19.26: Publication of partial presence information

The PA receives the initial PIDF document, assigns an entity tag to that document, and returns it in a SIP-Etag of a 200 (OK) response (2) to the PUA. Later, at some point in time, the PUA changes its presence information. It creates a partial PIDF document and includes it into a SIP PUBLISH request (3) that contains a SIP-If-Match header with the value of the entity tag previously learnt. An example of that PUBLISH request (3) is shown in Figure 19.28. Since this PUBLISH request (3) contains a subsequent state, the root element of the partial PIDF document is set to `<pidf-diff>`. The partial PIDF document indicates that the user is no longer available, by replacing the value of the `<status>` element to `close` and removing the RPID activities.

The PA receives the request. If the entity tag contained in the SIP-If-Match header matches the entity tag of the PIDF document stored at the server, then it inspects the partial PIDF document and performs the changes. Otherwise, the PA would answer with a 412 (Conditional Request Failed) response and would need to perform a full publication.

19.16.3 Event Notification Filtering

In daily usage a PA may deliver a lot of presence-related information about a presentity to one or more watchers. It may happen that watchers are not interested in all of the information,

```
PUBLISH sip:alice@example.com SIP/2.0
Via: SIP/2.0/UDP pc.example.com;branch=z9hG4bKn9s9d
To: Alice <sip:alice@example.com>
From: Alice <sip:alice@example.com>;tag=429j2
Call-ID: 092us2309isdd@pc.example.com
CSeq: 1 PUBLISH
Max-Forwards: 70
Expires: 3600
Event: presence
Content-Type: application/pidf-diff+xml
Content-Length: 759

<?xml version="1.0" encoding="UTF-8"?>
<p:pidf-full xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:p="urn:ietf:params:xml:ns:pidf-diff"
    xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
    xmlns:r="urn:ietf:params:xml:ns:pidf:rpid"
    entity="pres:alice@example.com">

    <tuple id="a3lkjdaf">
        <status>
            <basic>open</basic>
        </status>
        <dm:deviceID>urn:device:001349038B74</dm:deviceID>
        <contact priority="1.0">sip:alice@example.com</contact>
    </tuple>

    <dm:person id="4d4">
        <r:activities>
            <r:busy/>
            <r:on-the-phone/>
        </r:activities>
        <r:mood>
            <r:happy/>
        </r:mood>
    </dm:person>
</p:pidf-full>
```

Figure 19.27: PUBLISH request (1) with initial state

```

PUBLISH sip:alice@example.com SIP/2.0
Via: SIP/2.0/UDP pc.example.com;branch=z9hG4bKn9s9d
To: Alice <sip:alice@example.com>
From: Alice <sip:alice@example.com>;tag=429j2
Call-ID: 092us2309isdd@pc.example.com
CSeq: 2 PUBLISH
Max-Forwards: 70
Expires: 3600
Event: presence
SIP-If-Match: 1
Content-Type: application/pidf-diff+xml
Content-Length: 477

<?xml version="1.0" encoding="UTF-8"?>
<p:pidf-diff xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:p="urn:ietf:params:xml:ns:pidf-diff"
    xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
    xmlns:r="urn:ietf:params:xml:ns:pidf:rpid"
    entity="pres:alice@example.com">

    <p:replace sel="*/tuple[@id='a3lkjdaf']/status/basic/text()">
        <close></p:replace>

        <p:remove sel="*/dm:person[@id='4d4']" ws="both"/>
    </p:pidf-full>

```

Figure 19.28: PUBLISH request (3) with subsequent state

but just a subset of it. For instance, Alice may be interested to know when Bob is online or offline, but she may not be interested in knowing his detailed status (i.e., whether he is in a meeting, out to lunch or speaking on the phone). In another scenario Alice may just be interested in knowing details of Bob's instant messaging capabilities, but she may not be interested in the communication address of phones, email, web pages, etc.

If a watcher is only interested in a subset of the presence information available for a particular presentity and, especially, if the watcher has a narrow bandwidth connectivity channel, then it seems a waste of bandwidth, processing power, and even battery to send the whole presentity's presence information to the watcher. Therefore, it seems natural for a watcher to be able to specify a set of rules that filter the presence information to just those pieces that are of interest to the watcher.

Event notification filtering allows watchers to specify a filter that is applicable to the notification of events they are subscribed to. Filters can be applicable to elements of the XML presence information, attributes, extensions, transitions between two states, etc.

To apply event notification filtering, the watcher includes a new XML body that specifies the filter to be applied to that subscription. The filter is included as a body of the SUBSCRIBE request. The local policy of the presentity's PA overrides the filter, so the filter just gives information to the PA on those parts of the presence information that the watcher is interested in.

Chapter 20

The Presence Service in the IMS

Chapter 19 gave an overview of the presence service on the Internet, as defined by the IETF. This chapter focuses on the use of the presence service in the IMS. We explore the IMS architecture that supports the presence service and the applicability of presence to the IMS.

3GPP has defined, in 3GPP TS 24.141 [51], a presence service that runs over IMS, but mostly, 3GPP is just maintaining the specification, not actively progressing it. The presence service in IMS has since moved to OMA. OMA considers the presence service as an *enabler*, i.e., the set of specifications that enables a service. The main specification of the OMA Presence SIMPLE enabler is the OMA Presence SIMPLE specification [246]. This chapter describes this OMA Presence SIMPLE enabler.

20.1 The Foundation of Services

When we described the presence service on the Internet we unveiled a few of the powerful and rich possibilities that the presence service can offer to both end-users and other services.

On the one hand, end-users benefit from the presence service since they decide what information related to presence they want to provide to a list of authorized watchers. Presentities can decide the information they want to publish, such as communication address, capabilities of the terminals, or availability to establish a communication. Watchers receive that information in real time and decide how and when to interact with the presentity. All of these features enrich the end-user communication experience.

On the other hand, presence information is not only available to end-users but also to other services. These other services can benefit from the presence information supplied. For instance, an answering machine server is interested in knowing when users are online to send them an instant message announcing that they have pending voicemails stored in the server. A video server can benefit by adapting the bandwidth of the streaming video to the characteristics of the network where the presentity's device is connected. For all of these reasons we refer to the presence service as the foundation for service provision, as depicted in Figure 20.1.

20.2 Presence Architecture in the IMS

Figure 20.2 depicts the presence IMS architecture in conjunction with the Service configuration offered by the XDM architecture. The IMS terminal plays the role of both a watcher

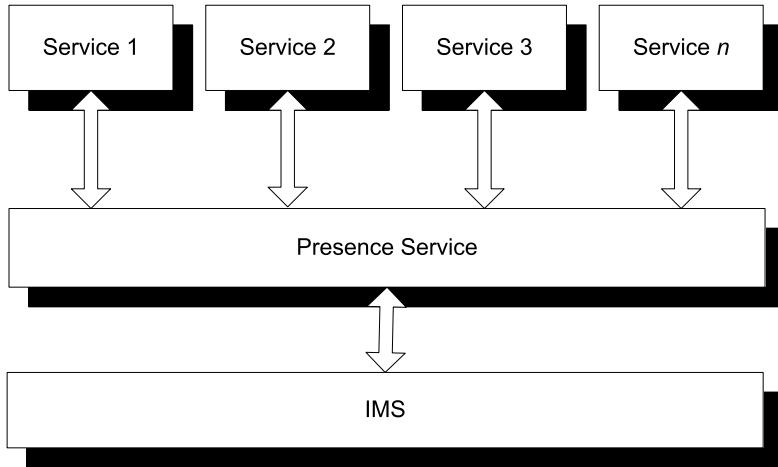


Figure 20.1: Presence service: the foundation of all the services

and a PUA (Presence User Agent), in addition to an XDMC). Central to the architecture is the function of the Presence Server (PS), which acts as a Presence Agent (PA) and is located in the home network. The PS is complemented with a Content Server, a Presence XDMS, and with a Presence Content XDMS. All of these are Application Servers from the IMS architecture point of view.

Typically presence requires the management of lists. A Resource List Server (RLS) provides the required functionality to host different lists. The RLS is complemented with a RLS XDMS and a Shared XDMS. All of these are Application Servers from the IMS architecture point of view.

Finally, Application Servers can also act as watchers of presence information. Typically a watcher in the presence service is implemented in connection to some other service that needs to keep track of the user's presence.

There are a number of interfaces in the Presence service, of which a non-comprehensive view is provided in Figure 20.2. However, most of the interfaces are realizations of the generic *ISC* interface, which is based on SIP, and the *Ut* interface, which is based on XCAP. The PS can also implement the *Sh* interface, which is based on Diameter. SIP interfaces are used for real-time presence data whereas XCAP interfaces are used for configuration.

20.3 Presence Publication

When the IMS presence application is launched, e.g., in an IMS terminal, the application publishes the current presentity's presence information. Figure 20.3 shows the flow and, as we can see, there is not much difference from the mechanism we explained in Section 19.4. The IMS terminal sends a PUBLISH request (1) that contains an Event header set to presence and includes a Presence Information Data Format (PIDF) document that describes the presence information. The S-CSCF receives the request (2) and evaluates the initial filter criteria for the presentity. One of the initial filter criteria indicates that PUBLISH requests containing an Event header set to presence ought to be forwarded to the PS where the presentity's presence information is stored. So, the S-CSCF forwards the PUBLISH

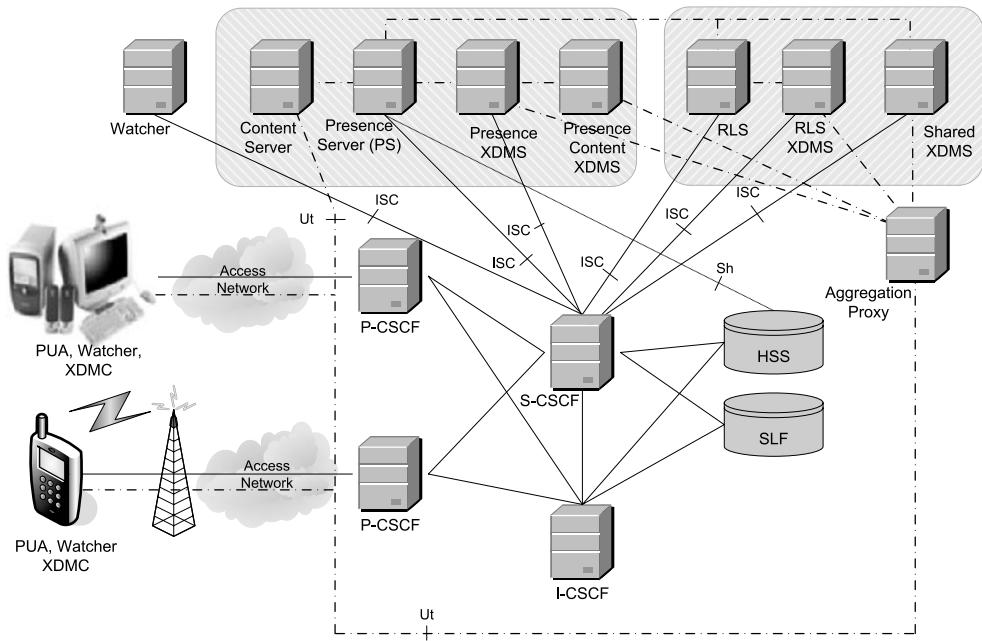


Figure 20.2: SIP-based presence architecture in the IMS

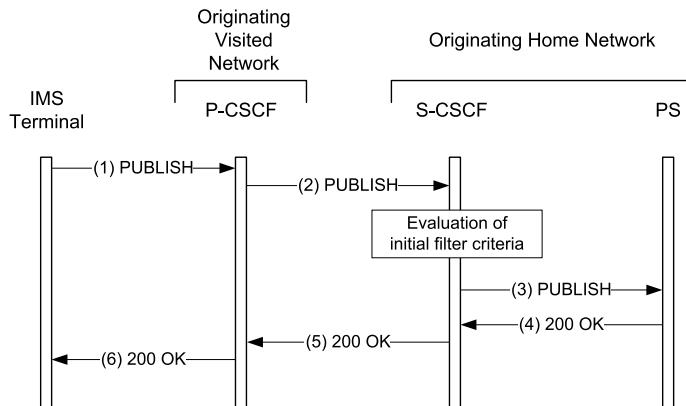


Figure 20.3: The IMS terminal publishing presence information

request (3) to that Application Server. The PS authorizes the publication and sends a 200 (OK) response (4).

Apart from the IMS terminal, any other network entity can act as the source of presence information and publish the user's presence information. These entities are called *Presence Network Agents (PNAs)* and the publish presence information on behalf of the user in a similar manner as the PUA would: using the SIP PUBLISH method that contains a PIDF document. Any network entity can act as a PNA: A Mobile Switching Center (MSC) server, a Gateway GPRS Support Node (GGSN), a Serving GPRS Support Node (SGSN), a Home Location

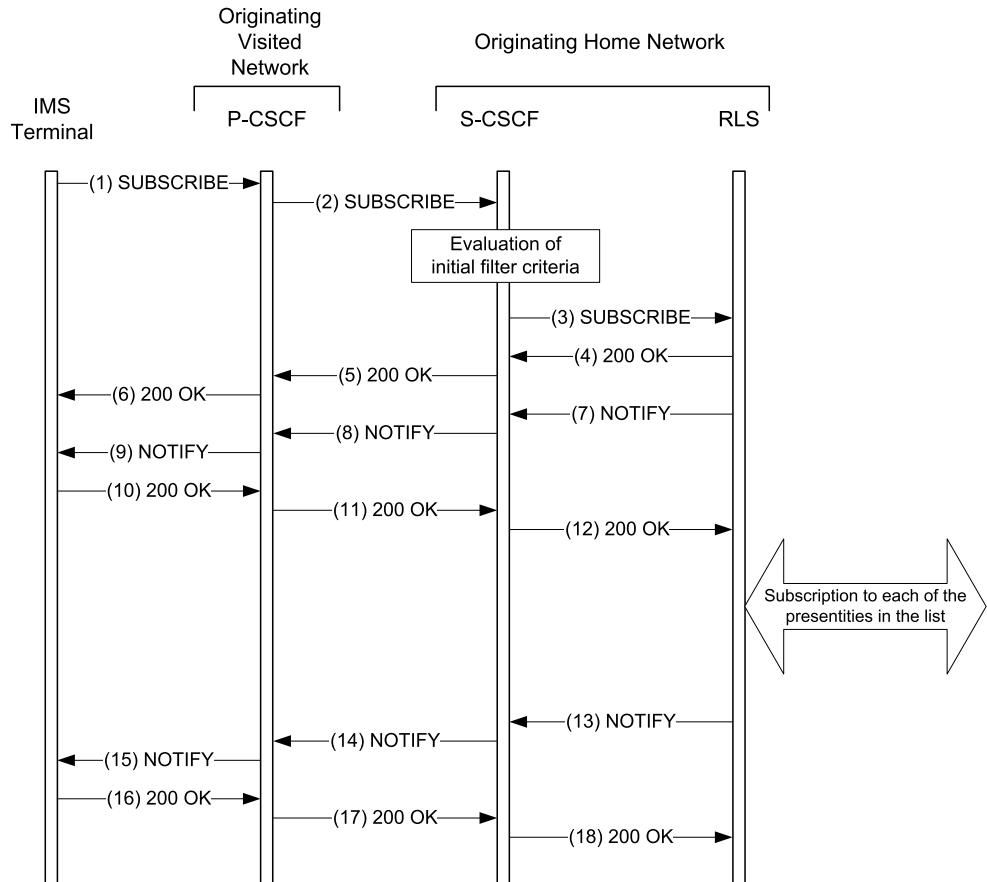


Figure 20.4: Watcher subscription to her own list

Register (HLR) or Home Subscriber Server (HSS), a S-CSCF, a Gateway Mobile Location Center (GMCL), etc.

20.4 Watcher Subscription

A user such as Alice is operating an IMS terminal that can act as a watcher. When she starts the presence application, her IMS terminal subscribes to the presence information of her presentities.

Although IMS allows watchers to subscribe individually to each of their presentities, as mentioned in Chapter 19, typically watchers subscribe to one or a few own presence lists, which are hosted in an RLS in their home network. This performs a very efficient batched subscription in a single shot, saving enormous amount of time to complete the subscriptions and saves quite a lot of bandwidth.

The flow is illustrated in Figure 20.4, which assumes that the RLS is provisioned with a list of URIs of presentities to which Alice eventually wants to subscribe. The watcher application residing in the IMS terminal sends a SUBSCRIBE request (1) addressed to her

list (e.g., `sip:alice-list@home1.net`). The SUBSCRIBE request contains a `Require` header field set to `eventlist` to indicate that the subscription is addressed to a list rather than a single presentity. An `Event` header field set to `presence` indicates that the subscription is for the presence event package. The SUBSCRIBE request (2) is received at the S-CSCF, which evaluates the initial filter criteria. One of those criteria indicates that SUBSCRIBE requests whose `Event` header field is set to `presence` and `Require` header field is set to `eventlist`, such as the received one, ought to be forwarded (3) to an Application Server that happens to be an RLS. The RLS, after verifying the identity of the subscriber and authorizing the subscription, sends a 200 (OK) response (4). The RLS also sends an initial NOTIFY request (7) that does not contain any presence information at this stage. The RLS subscribes one by one to all of the presentities listed in the resource list and, when enough information has been received, generates another NOTIFY request (13) that includes a presence document comprising the aggregated presence information of the other presentities.

Figure 20.5 shows the RLS subscribing to one of the presentities contained in the resource list upon the arrival of a subscription to a list. The process is repeated for each of the URIs included in the list. The RLS sends a SUBSCRIBE request (1) addressed to a presentity in the list. The request contains an `Event` header field set to `presence`. The request is forwarded via the S-CSCF in the RLS home network (2) to the I-CSCF in the presentity's network. The I-CSCF queries the HSS using the Diameter protocol, (3), (4) in Figure 20.5, to locate the S-CSCF allocated to the presentity and forwards the SUBSCRIBE request (5) to the allocated S-CSCF. The S-CSCF evaluates the initial filter criteria where there is a criterion indicating that the request ought to be forwarded to the presentity's PS (6). After sending the 200 (OK) response (7) the PS sends a NOTIFY request (11) that contains the presentity's presence information. In the example in Figure 20.4 neither the presentity's S-CSCF nor the presentity's I-CSCF record the route. Therefore, the PS sends the NOTIFY request (11) directly to the next node that recorded the route: the S-CSCF in the RLS network. The lack of this S-CSCF record route avoids the S-CSCF becoming locked when the presentity is not registered. So, if the presentity registers later there will be an S-CSCF allocation that may lead to a different S-CSCF.

20.5 Watcher Information and Authorization of Watchers

As described in Section 19.13, watcher information allows presentities to be informed of who is watching their presence information and what is the state of those watcher subscriptions. Typically, when the presence application is started in the IMS terminal, presentities subscribe to their own watcher information state, so that they can later receive notifications of watcher subscriptions when they occur.

The high-level watcher information flow is presented in Figure 20.6. For the sake of clarity, the flow assumes that the PS and the Presence XDMS are co-located. The IMS terminal of the presentity subscribes to its own watcher information by sending a SUBSCRIBE request (1) addressed in the *Request-URI* to the user's own Public User Identity. The `Event` header field is set to the value `presence.winfo`. The PS receives the SUBSCRIBE request (3), acknowledges with a 200 (OK) response (4), and generates an immediate NOTIFY request (7) as with any subscription.

At some point later in time, a watcher wants to subscribe to this user's presence information, so they send a SUBSCRIBE request (13) that is received in the PS. Since the PS does not have authorization from the presentity for this new watcher, the watcher subscription is set to the "pending" state. The PS answers the SUBSCRIBE request (13) with

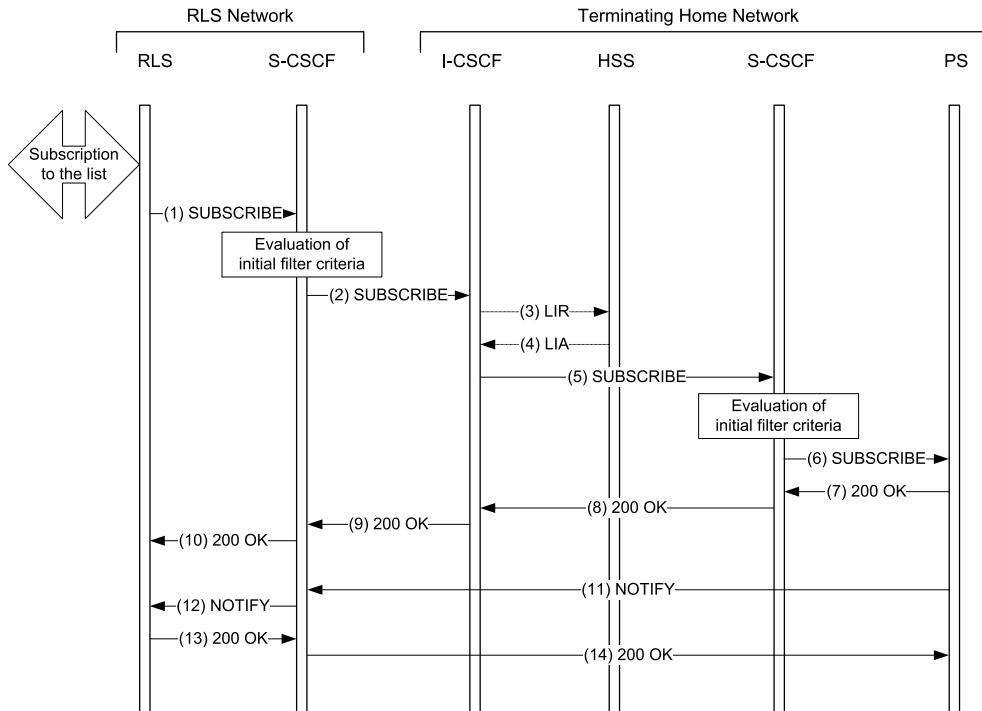


Figure 20.5: The RLS subscribes to a presentity

a 202 (Accepted) response (14) and immediately sends a NOTIFY request (15) that contains a *Subscription-State* header field set to *pending*. Then it also generates a NOTIFY request (17) to the presentity that includes a body describing the watcher information details, such as the watcher's public user identity and the subscription state, which is in *pending* state. The presence application in the IMS terminal can bring the pending authorization of a watcher to the attention of the user.

So, how can the presentity authorize this watcher subscription? The presentity uses XCAP and the *Presence authorization rules* XCAP application usage to provide that authorization. We discussed Presence authorization rules in greater detail in Section 19.14 in the context of the Internet. In IMS, the presence authorization rules are stored in the Presence XDMS. When the presentity wants to authorize a new watcher, it executes an XCAP operation invoking the HTTP PUT method (23) with the XCAP AUID set to *pres-rules*. This HTTP PUT request (23) includes an XML document that adds or modifies the entry pertaining to the identity of the new watcher and their subscription state is set to *active* state. The same XML document can also indicate which information of the presentity's PIDF is provided to this new watcher.

The Aggregation Proxy receives the HTTP PUT request (23) and forwards it (24) to the Presence XDMS that communicates the new authorization to the PS. The PS now generates a NOTIFY request (27) to the watcher to indicate that its subscription has been set to *active* state. This NOTIFY request (27) can also contain a PIDF document that contains the presentity's presence information authorized by the presentity to be received by this particular watcher.

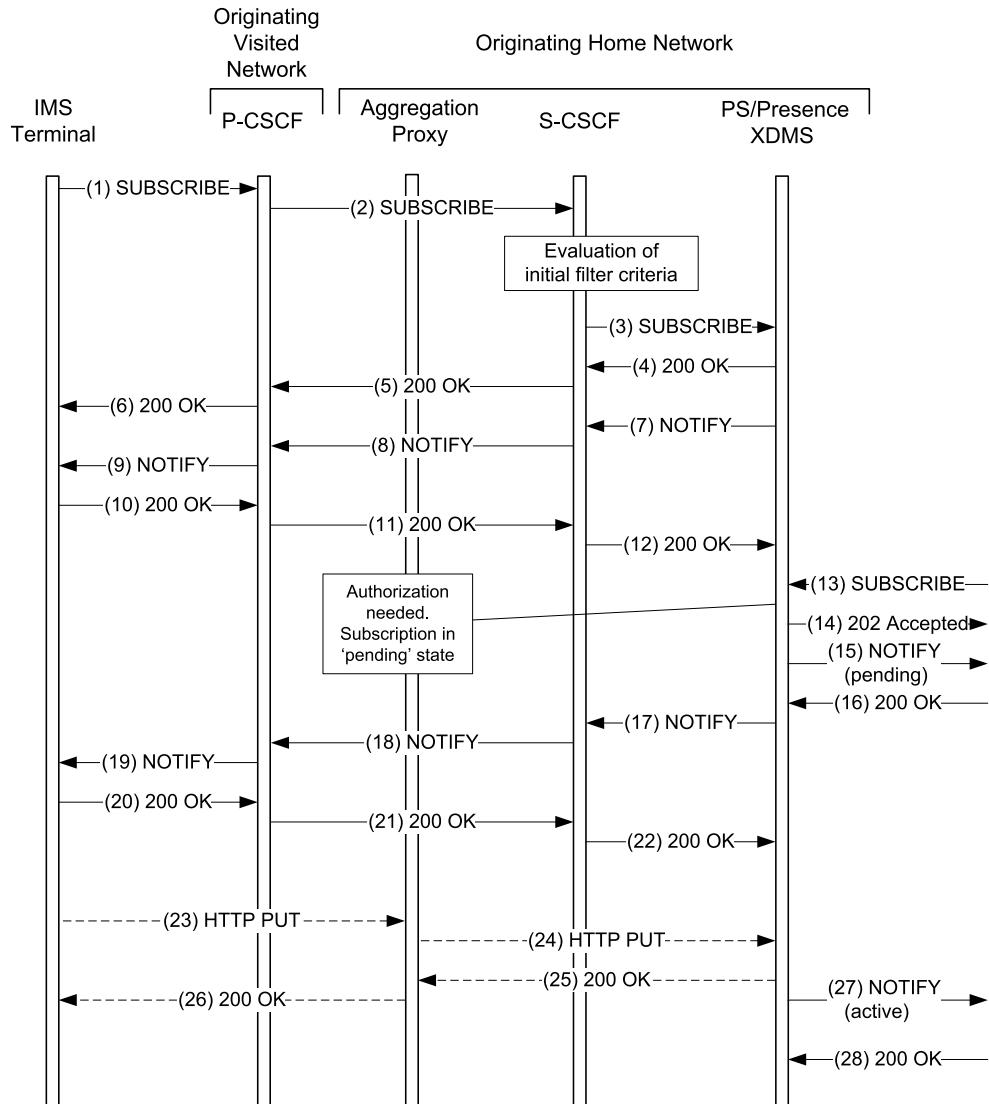


Figure 20.6: Subscription to own watcher information

20.6 Presence Optimizations

In particular, when the IMS terminal is wireless, there is a strong requirement to minimize the bandwidth used over the air interface. This has lead to the development of a few extensions that try to minimize the number of bits transferred over the air interface. Some of these extensions are applicable to any SIP subscription notification event package, while others are specific to the presence event package. We have discussed some of these optimizations in previous sections. Others are briefly mentioned here.

Section 4.15.1.1 discussed the event throttling mechanism, which allows the rate of notifications in any event package to be limited.

In certain scenarios it is possible to suppress the body of notifications, if the event state has not changed with respect to a previous notification. Sometimes it is even possible to suppress the complete NOTIFY request. Conditional event notification was described in Section 4.15.1.2.

Section 19.16.3 described the event notification filtering mechanism, which lets the watcher indicate the specific components of the PIDF it is really interested in receiving. Then, the PS removes those non-wanted components in the PIDF, saving unneeded information from being transmitted.

Partial notifications of presence information, which we already described in Section 19.16.1, allows the PS to include only a delta of the PIDF with respect an earlier version, saving lots of bandwidth when there are small changes in the PIDF. Its corresponding partial publication of presence information, described in Section 19.16.2, applies the same techniques to the publication of presence information.

Section 4.16 described SigComp (Signaling Compression), a mechanism that allows us to compress SIP and SDP (and make it compressed binary). SigComp is implemented in the P-CSCF and the IMS terminal. While the general SigComp mechanism is applicable to compress the SIP of SUBSCRIBE, NOTIFY, and PUBLISH requests, it will not have substantial effect in XML bodies such as the PIDF. To address this problem, a Presence-specific dictionary for SigComp, specified in RFC 5112 [149], is able to boost compression of PIDF bodies as well.

In Section 4.17 we described the content indirection mechanism. This mechanism allows an endpoint to include a reference to content (e.g., files) which is stored in a server. The reference is transmitted to a remote endpoint, which may download it, if it is in the user's interest. The benefit is that the downloading typically uses HTTP, therefore, downloading does not involve SIP servers, which are relieved from treating typically large pieces of data. Within the presence server in IMS, a user may store files in a content server, using HTTP, and manage those files through the content XDMS, using XCAP.

The Presence service in IMS is intending to support a mechanism to suppress presence notifications during a the duration of a subscription. The exact mechanism details are still under design at the time of this writing.

20.7 OMA Extensions to PIDF

The Open Mobile Alliance (OMA) has defined a small number of additional elements that complete the standard PIDF for mobile purposes. Consider the PIDF example in Figure 20.7, which contains all of these new OMA-defined elements.

OMA defines a *willingness* element, which can be used to indicated the user's willingness to receive incoming communications for the specific application and device. The element can take the values *open* or *closed*. This element is part of the *service* component of the presence data model.

A *session-participation* element indicates whether the user has at least one active session of a particular service. The element can take the values *open* or *closed* and is included in the *service* component of the presence data model.

The *registration-state* element indicates the presentity's registration state pertaining to a particular service. The element can take the values *active* or *terminated* and is included in the *service* component of the presence data model.

```

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
           xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
           xmlns:op="urn:oma:xml:prs:pidf:oma-pres"
           entity="pres:alice@example.com">

    <tuple id="3bfua">
        <status>
            <basic>open</basic>
        </status>
        <op:willingness>
            <op:basic>open</op:basic>
        </op:willingness>
        <op:session-participation>
            <op:basic>closed</op:basic>
        </op:session-participation>
        <op:registration-state>active</op:registration-state>
        <op:barring-state>terminated</op:barring-state>
        <op:service-description>
            <op:service-id>
                org.openmobilealliance:PoC-session
            </op:service-id>
            <op:version>1.0</op:version>
        </op:service-description>
        <contact priority="0.8">
            sip:alice@pc.example.com
        </contact>
        <timestampl>2005-06-05T07:52:14Z</timestampl>
    </tuple>

    <dm:device id="vjsa43">
        <op:network-availability>
            <op:network id="IMS">
                <op:active/>
            </op:network>
        </op:network-availability>
        <dm:deviceID>urn:device:001349038B74</dm:deviceID>
        <dm:note>PC</dm:note>
    </dm:device>

    <dm:person>
        <op:override-willingness>open</op:override-willingness>
    </dm:person>
</presence>
```

Figure 20.7: Example of a PIDF extended with OMA information

The **barring-state** element indicates whether the user has activated the incoming barring communication service of a particular service. The element can take the values **active** or **terminated** and is included in the *service* component of the presence data model.

A **service-description** element is used to indicate OMA-specific services. The element can have two children elements: **service-id** and **version**. The **service-id** element identifies the service and can take a number of values defined by OMA; the **version** element identifies the version of the service. The **service-description** element is included in the *service* component of the presence data model.

A **network-availability** element is used to indicate the type of network the user's device is connected to. The **network-availability** element contains a **network** element that includes an **id** attribute set to the type of connected network. This **id** attribute can take the value **IMS** to indicate an IMS application network, but it also can take any IP-CAN access type with the same value as the **P-Access-Network-Info** takes. The **network** element contains a child element set to either **active** or **terminated**, whose actual semantics depend on the type of network. The **network-availability** element is included in the *device* component of the presence data model.

A **presence** entity can use the **override-willingness** element to provide a generic willingness for all applications. As a consequence, the presence of the **override-willingness** takes precedence over the application and device specific **willingness**. The **override-willingness** element can take the values **open** or **closed**, and it is included in the *person* component of the presence data model.

Chapter 21

Instant Messaging on the Internet

Instant messaging is one of today's most popular services. Many youngsters (and not-so-young people) use the service to keep in touch with their relatives, friends, co-workers, etc. Millions of instant messages are sent every day. So, it will come as no surprise that such a popular service is already supported in the IMS.

Instant messaging is the service that allows a user to send some content to another user in near-real time. Because of the real-time characteristics of instant messages the content is typically not stored in network nodes, as often happens with other services such as email.

The content in an instant message is typically a text message, but can be an HTML page, a picture, a file containing a song, a video clip, or any generic file.

The instant messaging service combines perfectly with the presence service, since presence allows a user to be informed when other users become available (e.g., connect to the network). Then, users can send instant messages to their friends and start some sort of messaging conversation.

21.1 The *im* URI

Like presence, mail, or AAA functions, an instant messaging service can be identified by an *im* URI. Like the *pres* URI the *im* URI does not define the protocol used to access an instant message resource. So, whenever SIP is the protocol used to send the instant message it is recommended to use *sip* or *sips* URIs.

The syntax of the *im* URI is

```
IM-URI      = "im:" [ to ] [ headers ]
to          = mailbox
headers    = "?" header *( "&" header )
header     = hname "=" hvalue
hname      = *urlc
hvalue     = *urlc
```

An example of an *im* URI is

```
im:alice@example.com
```

21.2 Modes of Instant Messages

There are two modes of operation of the instant message service, depending on whether they are stand-alone instant messages or part of a session of instant messages.

We refer to a *pager-mode* instant message as one that is sent as a stand-alone message, not having any relation with previous or future instant messages. This mode of instant messaging is referred to as “pager mode” because the model resembles the way a two-way pager works. The model is also similar to the SMS (Short Message Service) in cellular networks.

We refer to a *session-based* instant message as one that is sent as part of an existing session, typically established with a SIP INVITE request.

Both models have different requirements and constraints; hence, their implementation is different. The following sections describe the implementation of both models.

21.3 Pager-mode Instant Messaging

The IETF has created an extension to SIP that allows a SIP UA to send an instant message to another UA. The extension consists of a new SIP method named *MESSAGE*. The SIP MESSAGE method, which is specified in RFC 3428 [115], is able to transport any kind of payload in the body of the message, formatted with an appropriate MIME type.

Figure 21.1 illustrates the mode of operation. A UAC sends a MESSAGE request (1) to a proxy. The detailed contents of the request are shown in Figure 21.2. The proxy forwards the MESSAGE request (2) like any other SIP request, even when the proxy does not support or understand the SIP MESSAGE method. Eventually, the UAS will receive it and answer with a 200 (OK) response (3) that is forwarded (4) to the UAC.

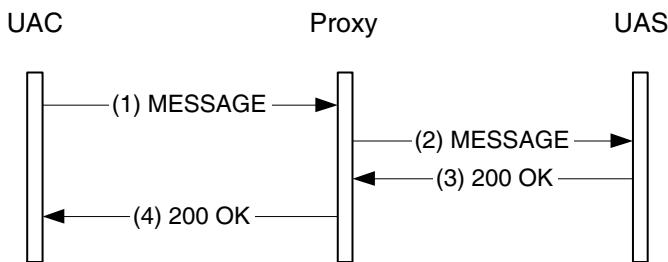


Figure 21.1: Pager-mode instant message with a MESSAGE request

Like OPTIONS or REGISTER, MESSAGE requests do not create a SIP dialog. They are stand-alone requests. However, they can be sent as part of an existing dialog (e.g., created by a SIP INVITE request).

21.3.1 Congestion Control with MESSAGE

One of the problems with SIP derives from the fact that any proxy can change the transport protocol from TCP to UDP, SCTP, or other transport protocols and vice versa. UDP is notorious for not offering any congestion control, whereas TCP and SCTP do offer congestion control. If a UA is sending a large instant message over a transport protocol that does not offer congestion control, the network proxies can become congested and stop processing other SIP requests such as INVITE, SUBSCRIBE, etc. Even if a UA sends a large SIP MESSAGE

```

MESSAGE sip:bob@example.org SIP/2.0
Via: SIP/2.0/TCP alicepc.example.com;branch=z9hG4bK776sgd43d
Max-Forwards: 70
From: Alice <sip:alice@example.com>;tag=48912
To: Bob <sip:bob@example.org>
Call-ID: a3d3hdj5ws223ns6lk8djds
Cseq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 31

```

Hi, what is going on today?

Figure 21.2: (1) MESSAGE

over a transport protocol that implements end-to-end congestion control (e.g., TCP, SCTP), the next proxy can switch to UDP and congestion can occur.

At the time of writing, SIP does not offer a mechanism for a UA to indicate that all proxies in the path must use a transport protocol that implements end-to-end congestion control. Consequently, a limit has been placed on the SIP MESSAGE method such that MESSAGE requests cannot exceed the MTU (Maximum Transmission Unit) minus 200 bytes. If the MTU is not known to the UAC this limit is 1300 bytes.

A solution to sending SIP MESSAGE requests with large bodies is to use the content indirection mechanism. The UAC uses HTTP, or any other protocol that runs over a congestion-controlled transport protocol, to store the body of the SIP request in a server. Then, the UAC inserts a link to the URI where the payload is stored, instead of sending the whole body embedded in the SIP MESSAGE request. When the UAS receives the SIP request, it uses HTTP or the appropriate protocol to download the body. We described the content indirection mechanism in more detail in Section 4.17.

Another solution to getting around the size limit problem with MESSAGE is to use the session-based instant message mode rather than pager mode. Let us take a look at session-based instant messaging.

21.4 Session-based Instant Messaging

The session-based instant message mode uses the SIP INVITE method to establish a session where the media plane is not audio or video, but an exchange of instant messages.

When a UAC establishes a session to send and receive audio or video, the media is sent via the Real-Time Transport Protocol (RTP, specified in RFC 3550 [301]). However, when the UAC establishes a session to send and receive instant messages the actual media (the collection of instant messages) are sent over the Message Session Relay Protocol (MSRP, specified in RFC 4975 [114]).

MSRP is a simple text-based protocol whose main characteristic is that it runs over transport protocols that offer congestion control, such as TCP (RFC 793 [257]), SCTP (RFC 2960 [308]), and TLS over TCP (RFC 2246 [120]). Explicitly, MSRP does not run over UDP (RFC 768 [255]) or any other transport protocol that does not offer end-to-end congestion control. Because of this, the main characteristic of MSRP is not imposing a restriction on the size of an instant message.

Another characteristic of MSRP is that it runs on the media plane. Therefore, MSRP messages do not traverse SIP proxies. This is an advantage, since SIP proxies are not disturbed with proxying large instant messages.

MSRP supports instant messages to traverse zero, one, or two MSRP relays. MSRP relays play an important role when one of the endpoints is located behind a NAT (Network Address Translator). They are also helpful for network administrators when configuring firewall traversal. In addition, MSRP relays can provide logging and statistical usage. For historical reasons, the behavior of MSRP relays is specified in a separate document, RFC 4976 [193].

21.4.1 The MSRP and MSRPS URLs

MSRP defines two new URLs to address MSRP resources: *msrp* and *msrps*. Both URLs are similar in concept, but the *msrps* URL indicates a requirement for a secure TLS connection over TCP. They have the following structure:

```
"msrp://" [userinfo "@"]
hostport ["/" session-id] ";" transport
"msrps://" [userinfo "@"]
hostport ["/" session-id] ";" transport
```

where *userinfo* and *hostport* are specified in RFC 2396 [85], and *session-id* and *transport* are strings of characters defined by MSRP.

An example of an MSRP URL is

```
msrp://alice@pc.example.com/ds1kj2nd;tcp
```

21.4.2 MSRP Overview

As is the case in SIP, MSRP is a text-based protocol whose messages are either requests or responses. However, unlike SIP, not every MSRP request is answered by an MSRP response. Like SIP, MSRP defines methods that indicate the semantics of a request. There are currently three methods defined in MSRP.

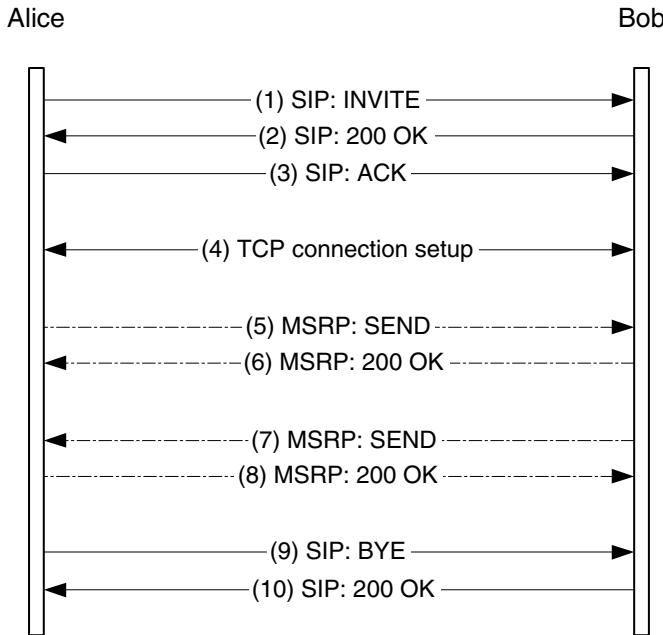
- SEND: sends an instant message of any arbitrary length from one endpoint to another.
- REPORT: an endpoint or a relay provides message delivery notifications.
- AUTH: used by endpoints to authenticate to relays.

Like SIP, MSRP requests and responses contain a number of headers that describe, for example, the message identification, the path to the receiver or to the sender, or the content type.

MSRP SEND requests typically carry a body encoded according to MIME-encoding rules. As a minimum, all of the MSRP implementations support the `message/cpim` MIME-type format (specified in RFC 3862 [203]). Typically, MSRP implementations support other MIME-type formats, such as `text/plain` or `text/html`.

The MSRP standards define new extensions to SDP (Session Description Protocol, specified in RFC 2327 [160]) to be able to express the media type `message` and associated attributes.

Figure 21.3 shows a high-level flow of the interleaving that takes place between SIP and MSRP when a session is established. The endpoint that originates the session, Alice, sends a SIP INVITE request (1) that contains an SDP offer indicating the message media type and

**Figure 21.3:** Establishment of a session of instant messages

```

INVITE sip:Bob.Brown@example.org SIP/2.0
Via: SIP/2.0/UDP ws1.example.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Alice <sip:Alice.Smith@example.com>;tag=hx34576s1
To: Bob <sip:Bob.Brown@example.org>
Call-ID: 2098adkj20
Cseq: 22 INVITE
Contact: <sip:alice@192.0.100.2>
Content-Type: application/sdp
Content-Length: 220

v=0
o=alice 2890844526 2890844526 IN IP4 ws1.example.com
s=-
c=IN IP4 ws1.example.com
t=0 0
m=message 8231 msrp/tcp *
a=accept-types:message/cpim text/plain text/html
a=path:msrp://ws1.example.com:8231/9s9cpl;tcp
  
```

Figure 21.4: INVITE request (1)

the support for MSRP. An example of this INVITE request is shown in Figure 21.4. We do not show any potential SIP proxy server that is in the path for the sake of clarity.

The MSRP address of the originator of the session is indicated in the MSRP URL contained in the `a=path` line in the SDP of the INVITE request (1). Bob answers with a 200 OK response (2) that also contains his MSRP URL in the `a=path` line of the SDP. Figure 21.5 shows the complete SIP response. The session is acknowledged with the SIP ACK request (3).

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ws1.example.com:5060;branch=z9hG4bK74gh5
From: Alice <sip:Alice.Smith@example.com>;tag=hx34576s1
To: Bob <sip:Bob.Brown@example.org>;tag=ba03s
Call-ID: 6328776298220188511@192.0.100.2
Cseq: 1 INVITE
Contact: <sip:bob@bob.example.org>
Content-Type: application/sdp
Content-Length: 215

v=0
o=bob 2890844528 2890844528 IN IP4 bob.example.org
s=-
c=IN IP4 bob.example.org
t=0 0
m=message 7283 msrp/tcp *
a=accept-types:message/cpim text/plain text/html
a=path:msrp://bob.example.org:7283/d9s9a;tcp
```

Figure 21.5: 200 (OK) response (2)

Once the session is established, the SDP offerer (Alice, in our example), establishes a TCP connection (4) to the answerer's MSRP URL (Bob's MSRP URL) and sends an immediate SEND request (5) that may or may not have a payload. Figure 21.6 shows an MSRP SEND request (5) that includes an actual instant message.

```
MSRP 230cmqj SEND
To-Path: msrp://bob.example.org:7283/d9s9a;tcp
From-Path: msrp://ws1.example.com:8231/9s9cpl;tcp
Message-ID: 309203
Byte-Range: 1-20/20
Content-Type: text/plain

This is Alice typing
-----230cmqj$
```

Figure 21.6: An instant message carried in an MSRP SEND request (5)

Then Bob replies with an MSRP 200 response (6) to confirm that the message has been received. Figure 21.7 shows the 200 OK response.

Any of the endpoints is able to close the session at any time. When an endpoint wants to close the session it sends a SIP BYE request (9), as it would do with any other session it wanted to close.

```

MSRP 230cmqj 200 OK
To-Path: msrp://ws1.example.com:8231/9s9cp1;tcp
From-Path: msrp://bob.example.org:7283/d9s9a;tcp
Message-ID: 309203
Byte-Range: 1-20/20
-----230cmqj$
```

Figure 21.7: MSRP 200 response (6)

21.4.3 Extensions to SDP due to MSRP

MSRP extends SDP by adding new forms of encoding existing lines (for example, the `m=` line), and by adding new attributes (`a=` lines) to describe MSRP sessions.

Figure 21.8 shows an example of the SDP generated by an MSRP endpoint. The first five lines (`v=`, `o=`, `s=`, `c=`, and `t=`) are created as in regular SDP. In particular, in spite of the presence of an MSRP URL later, the `c=` lines contain the same hostname or IP address that later will be encoded in the MSRP URL.

```

v=0
o=bob 2890844528 2890844528 IN IP4 bob.example.org
s=-
c=IN IP4 bob.example.org
t=0 0
m=message 7283 msrp/tcp *
a=accept-types:message/cpim
a=accept-wrapped-types:text/plain text/html *
a=path:msrp://bob.example.org:7283/d9s9a;tcp
a=max-size=8000
```

Figure 21.8: SDP describing an MSRP media

The `m=` line contains a `message` media type to indicate the type of media. The port number is set to the real port number that, otherwise, also appears encoded later in the MSRP URL. The transport protocol is set to `msrp/tcp`, or `msrp/tls/tcp`, as required. The last item in the `m=` line would indicate the format list of the media (e.g., the codec in audio or video media types). MSRP sets this item to an asterisk “*”.

The list of supported MIME body types in MSRP SEND messages is encoded in a new `a=accept-types` line. MSRP mandates to support, at least, `message/cpim` (for messages encoded according to RFC 3862 [203]) and `text/plain` (for pure text messages). Although it is common to support other types, such as `text/html`, `image/jpeg`, etc. An asterisk “*” at the end of the list indicates that the list of supported MIME body types is not comprehensive (perhaps due to the large number of supported types), and that the peer endpoint may attempt to send instant messages that include other types.

The `a=accept-wrapped-types` describes a list of MIME body types that are only accepted when they are wrapped in another MIME body type. The encoding is the same as the `a=accept-types` line. This allows, for example, a gateway to force messages to be wrapped in the type identified in the `a=accept-types`, and still describe general types that are accepted inside the wrapper (in the `a=accept-wrapped-types` line). For example, the

SDP in Figure 21.8 indicates that the MSRP endpoint accepts MSRP messages that include a Message/CPIM body which, in turn, includes a plain text or HTML body (or any other non-declared type, because of the presence of an asterisk in the `a=accept-wrapped-types` line).

The new `a=path` line indicates the MSRP URL of the endpoint that is creating the SDP. If the endpoint requires a relay to operate, then the line contains two entries, one describing the endpoint and the other describing the relay. Note that there is some redundancy in the information, since the `a=path` contains the MSRP URL, and the `c=` and `m=` lines contain the hostname (or IP address) and the port number, respectively. MSRP prefers to use URLs to identify endpoints, since they are a very rich means of describing the connection. The `c=` and `m=` lines are populated in a traditional way to allow backward compatibility with some nodes that they require to access the IP address and port number of the endpoint (the P-CSCF is an example of this).

The new `a=max-size` line indicates the maximum size of the message that the endpoint wishes to receive. The value is indicated in octets and provides a hint to the receiver of the SDP.

21.4.4 MSRP Core Functionality

MSRP is able to transport instant messages between two endpoints, perhaps through some relays in the path. Instant messages are, in this context, any MIME-encoded body of any arbitrary length, for instance, a pure text message, a large video file, or an image. Essentially, anything that can be MIME-encoded can be transported in MSRP.

Because of the fact that some instant messages can be large in nature (think of a large video file), and because of the users' requirement to be able to share a single TCP connection to simultaneously send a large instant message without disturbing other existing text conversations, MSRP provides a chunking and rechunking mechanism. In the case where a message is chunked, the endpoint splits the payload into several chunks and sends each chunk in a SEND request. Both endpoints and relays are able to split the contents of an instant message into a number of chunks, of no longer than 2048 octets. The receiver endpoint can glue all of the received chunks to compose the original instant message.

In addition, MSRP assumes that messages might be interruptible, meaning that a user may decide to abort the transmission of a message prior to the completion of its transmission.

In a different scenario, a user might want to send a message whose length is not known at the time the message transmission starts. This might be the case, for example, for a video file that is being created at the time the instant message transmission starts.

In order to support all of these requirements, MSRP uses a boundary-based framing mechanism. SEND requests include, right after the body, a boundary string that identifies the end of the SEND request. In addition, the boundary string also contains a flag indicating whether the request contains a complete instant message, a chunk, or an aborted message.

Figure 21.9 shows an example of a simple MSRP message. The first line contains the protocol name MSRP followed by unique boundary string "bnsk1s" and the method SEND. The unique boundary string identifies the end of the message, as we will see below.

The SEND request in Figure 21.9 also includes From-Path and To-Path headers that contain the full MSRP URLs of the sender and the receiver, respectively, for this particular session. Potential relays (serving either the sender or the recipient) would also be listed in the From-Path or To-Path headers.

```

MSRP bnsk1s SEND
To-Path: msrp://alice.example.com:4423/xodj2;tcp
From-Path: msrp://bob.example.org:15000/vnskq;tcp
Message-ID: 003293
Byte-Range: 1-22/22
Content-Type: text/plain

Hi, how are you today?
-----bnsk1s$
```

Figure 21.9: MSRP SEND request

Next in the SEND request we find the Message-ID header, which contains a unique message identifier within the session. If a message is split into several chunks, each of the SEND requests has the same Message-ID value. Finally, responses also have the same Message-ID value as the corresponding request.

The example in Figure 21.9 also shows a Byte-Range header that indicates the range of bytes of the body transmitted, and the total number of bytes of the complete message (if known). In our example the Byte-Range header indicates that messages ranging from 1 to 22 of a total of 22 bytes are being sent (hence, it is a complete message). Byte-Range headers are also present in 200 OK responses to the SEND request for the purpose of acknowledging the range of bytes received at the receiving endpoint.

A Content-Type header indicates the MIME type that is encoded in the body data. In our example plain text is sent.

Then an empty line separates the MSRP header from the actual body data. After the body data, seven dashes “-” plus the boundary string declared in the request line follow. In the example shown in Figure 21.9 the boundary string is “bnsk1s”. The receiver of a SEND request just needs to search for seven dashes “-” and the boundary string to find the end of the body. The last character after the boundary string is a flag that can have any of the following values.

- \$: the body data contains a complete message or the last chunk of a message.
- +: the body data contains a chunk that does not complete the message.
- #: the body contains an aborted message; the endpoint will not send the remaining bytes or chunks of the message.

Figure 21.10 shows two SEND requests, each containing a chunk that makes a complete message. The boundary strings are different in each message, but the Message-ID value is the same. The Byte-Range header indicates the number of bytes sent in each chunk and its position with respect to the complete message. The flag at the end of the first message indicates that more chunks will be sent in other SEND requests. In the last message the flag indicates that it is the last chunk of a message.

21.4.5 Status and Reports

MSRP introduces the concept of status and reports in instant messaging. On the one hand, an MSRP endpoint might be interested in the *transaction status*, which refers to the status of

```

MSRP ea1dof SEND
To-Path: msrp://alice.example.com:4423/xodj2;tcp
From-Path: msrp://bob.example.org:15000/vnskq;tcp
Message-ID: 459874
Byte-Range: 1-11/22
Content-Type: text/plain

Hi, how are
-----ea1dof+

MSRP ea1eeo SEND
To-Path: msrp://alice.example.com:4423/xodj2;tcp
From-Path: msrp://bob.example.org:15000/vnskq;tcp
Message-ID: 459874
Byte-Range: 12-22/22
Content-Type: text/plain

you today?
-----ea1eeo$
```

Figure 21.10: MSRP SEND request

the delivery of an instant message to a next hop (a relay or an endpoint). On the other hand, an MSRP endpoint might be interested in receiving the status of the delivery of the instant message at the other end, in what is called the *request status*.

The transaction status is typically sent in MSRP responses, although on some occasions it can be carried in REPORT requests. In contrast, the request status is always carried in REPORT requests.

The sender of the SEND request governs the type of status the user is interested in receiving. MSRP provides the sender with two headers that control the request of status indications: the Success-Report and Failure-Report headers.

Success-Report can take the values “yes” or “no”. A value of “yes” means that the receiver will generate a REPORT request when the last chunk of the message or a complete message is received. A value of “no” means that the recipient will not generate a report of the successful reception of the message. The default value, in the absent of a Success-Report header, is “no”.

Failure-Report can take the values “yes”, “no”, or “partial”. A value of “yes” means that the receiver will generate an error response if the transaction fails. In some cases (e.g., a gateway) it can generate a 200 OK response and then, later, when a response from the other system is received, it can generate a REPORT request. A value of “no” means that the recipient will not generate any response, not even a successful response. A value of “partial” indicates that the recipient will not generate 200 OK responses, but it will generate other failure responses. In the absence of a Failure-Report header, the default value of “yes” is assumed.

With all of these values in place, all of the different scenarios can be accommodated. For example, a system administrator who wants to send an instant message to all users indicating that the system is shutting down is probably not interested in receiving failure reports, so they

would set the `Failure-Report` header to “no”. In another example, an online securities trading system will most likely set the `Success-Report` header to “yes”. However, in a public Internet chat system, where performance is important, the `Success-Report` might be set to “no”.

The request of success reports is illustrated in Figure 21.11. Alice sends an MSRP SEND request (1) whose `Success-Report` header is set to “yes”. We assume that the request contains a complete message. The 200 OK response (2) constitutes a transaction report and it is generated by Alice’s relay, which relays the request (3) to Bob’s relay. Eventually Bob receives the SEND request (5). Then Bob honors the `Success-Report` header and generates a REPORT request (7).

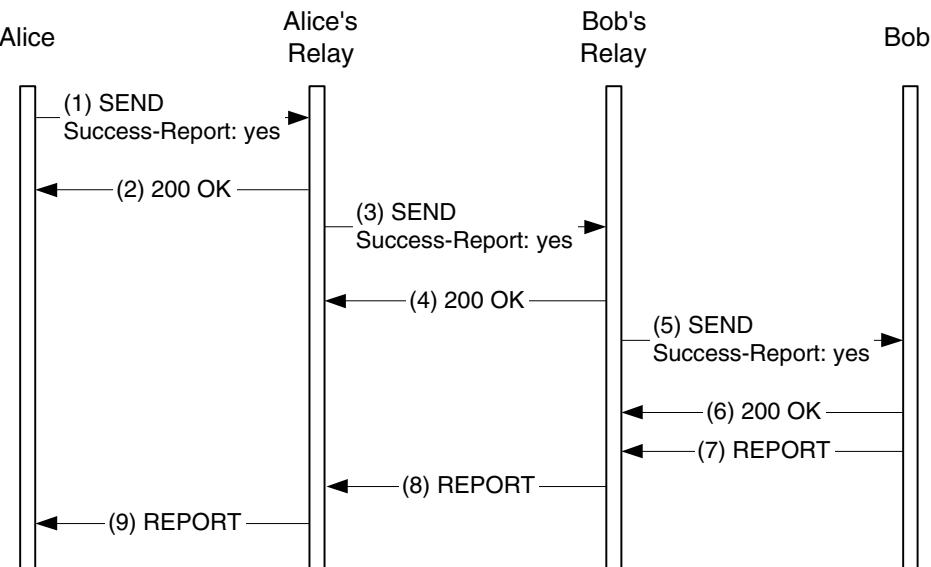


Figure 21.11: Request for a success report

Let us take a look at the MSRP REPORT request (7). Figure 21.12 shows an example. REPORT requests keep the value of the `Message-ID` header of the SEND request the report belongs to. The `Byte-Range` header indicates the range of bytes to which the report applies.

```

MSRP 439dscd REPORT
To-Path: msrp://ws1.example.com:8231/9s9cp1;tcp
From-Path: msrp://bob.example.org:7283/d9s9a;tcp
Message-ID: 309203
Byte-Range: 1-22/22
Status: 000 200 OK
-----439dscd$
  
```

Figure 21.12: REPORT request (7)

A new `Status` header contains the status of the request. The first three digits in the value indicate the namespace of the status. The namespace indicates the context of the rest of the

information present in the value of the `Status` header. Only namespace “000” is standardized at the time of writing and it indicates that the rest of the values in the header correspond to the status code of a transaction response code. In our example the rest of the `Status` header value is populated with a “200 OK”, indicating that the `REPORT` request has the same meaning as a “200 OK” response, i.e., the `SEND` request has been successfully received.

Figure 21.13 shows an example where Bob receives a `SEND` request (4) with the `Failure-Report` header set to “yes”. If the transaction does not exist at Bob’s endpoint, he generates a 481 “Session does not exist” response (6) that is received at Bob’s relay. Since Bob’s relay had already acknowledged the `SEND` request (3) with a 200 OK response (4), and now it has received further information that the request was not successfully received by Bob, the relay generates a `REPORT` request (7) that contains a `Status` request set to the value “000 481 Session does not exist” (the 000 indicates the namespace of the MSRP responses, and the rest indicates the actual response received by the relay). When Alice receives the `REPORT` request (8) she can determine that the request failed due to the value of the `Status` header.

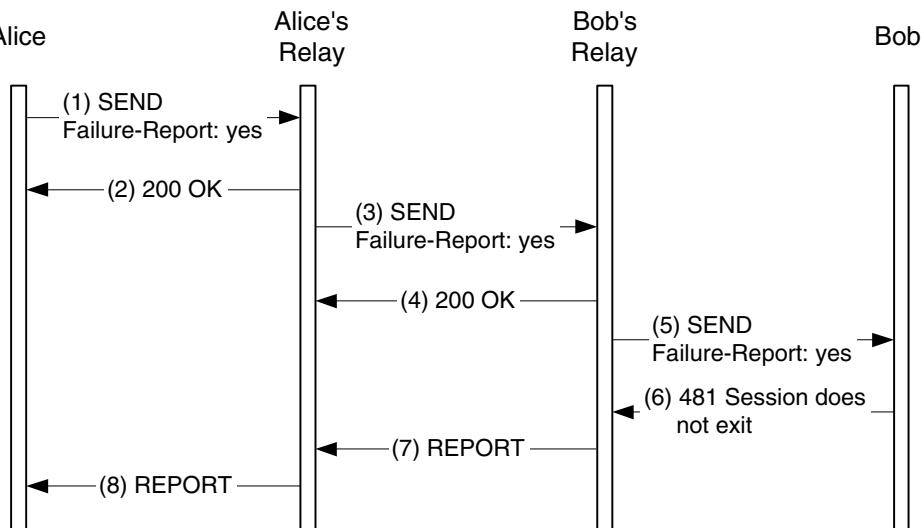


Figure 21.13: Request for a failure report

21.4.6 MSRP Relays

Although Figures 21.11 and 21.13 have intuitively indicated the use of relays in MSRP, we have not given a formal description. MSRP relays are specified in RFC 4976 [193].

An MSRP relay is a specialized node in transiting MSRP messages between two other MSRP nodes (endpoints or other relays). MSRP relays, which are located in the media plane, must not be confused with SIP proxies, which are located in the signaling plane. MSRP does not offer a mechanism for an endpoint to discover its relay, so it is assumed that each endpoint is provisioned with its MSRP relay, in a similar way as HTTP proxies are configured in HTML browsers.

When an endpoint wants to make use of an MSRP relay, it first opens a TLS connection towards its relay, authenticates (by sending an AUTH request), and if authentication is successful the relay provides the endpoint with an MSRPS URL that the endpoint can use for its MSRP sessions.

Figure 21.14 shows the detailed flow of signals. First, the endpoint opens the TLS connection (1) towards the relay. After that, the endpoint sends an AUTH request (2), which is answered by the relay with a 401 response (3) that contains the name of a realm where a username and password combination should be valid. The endpoint then builds a new AUTH request (4) that contains a valid username and password combination in that realm. If they are correct, the MSRP relay answers with a 200 OK response (5). At any time the endpoint can initiate an MSRP session by first sending a SIP INVITE request (6) that contains an SDP offer that declares both the MSRP relay URL and the MSRP endpoint URL.

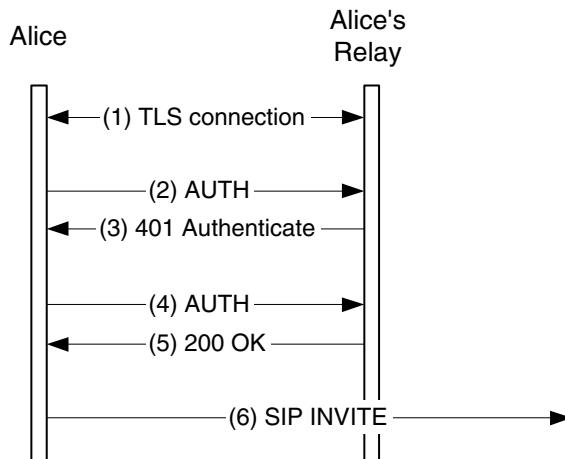


Figure 21.14: Authentication to an MSRP relay

Let us take a look at the authentication and authorization details in MSRP. Figure 21.15 shows an example of an AUTH request that Alice sends to her MSRP relay. The To-Path and From-Path contain the relay and Alice’s MSRPS URL, respectively. It is assumed that the relay URL has either been provisioned to Alice’s endpoint or has been learnt somehow (MSRP does not specify how to learn the relay MSRPS URL).

```

MSRP w39sn AUTH
To-Path: msrps://alice@relay.example.com:3233;tcp
From-Path: msrps://alice.example.com:9892/98cjs;tcp
Message-ID: 020391
-----w39sn$
  
```

Figure 21.15: AUTH request (2)

The MSRP relay answers with a 401 (Authenticate) response (3), as shown in Figure 21.16. The response includes a WWW-Authenticate header containing the realm where the username and password should be valid. The WWW-Authenticate header is imported

from HTTP authentication, and it is specified in RFC 2617 [145]. Unlike SIP, where the only authentication mechanism imported from HTTP is Digest, MSRP only imports the Basic authentication mechanism from HTTP. Basic authentication is simpler than Digest, but usernames and passwords are sent in the clear. While that is a major drawback for SIP, it is not for MSRP, since the endpoint-to-relay connection is protected and encrypted with TLS. Thus, even though Basic authentication sends usernames and password openly within a channel, the channel is encrypted, and eavesdroppers will not be able to guess them.

```
MSRP w39sn 401 Authenticate
From-Path: msrps://alice@relay.example.com:3233;tcp
To-Path: msrps://alice.example.com:9892/98cjs;tcp
Message-ID: 020391
WWW-Authenticate: Basic realm="relay.example.com"
-----w39sn$
```

Figure 21.16: 401 (Authenticate) response (3)

Then the endpoint creates the response and sends a new AUTH request (4) to the relay. The Authorization header, also imported from RFC 2617 [145] contains a Base64 encoded string (specified in RFC 3548 [196]) that contains the username and the password. Figure 21.17 shows an example of this request.

```
MSRP p2pe3 AUTH
To-Path: msrps://alice@relay.example.com:3233;tcp
From-Path: msrps://alice.example.com:9892/98cjs;tcp
Message-ID: 929195
Authorization: Basic bWlndWVsLmdhcmNpYTp3cm90ZSB0aGlzIGNoYXB0ZXI=
-----p2pe3$
```

Figure 21.17: AUTH request (4)

When the MSRP relay receives this new AUTH request (3), it decodes the Authorization header value and verifies that the username and password combination is valid in the administrative realm. If everything is correct, the MSRP relay creates a 200 (OK) response (5). This response contains a Use-Path header that contains one or more MSRPS URLs of the relay or relays that the endpoint has to use for this session. The MSRPS URLs returned are “session” URLs (i.e., they contain a `session-id` path, and they are valid only for one unique session). Figure 21.18 shows an example of the 200 (OK) response (5).

```
MSRP p2pe3 200 OK
From-Path: msrps://alice@relay.example.com:3233;tcp
To-Path: msrps://alice.example.com:9892/98cjs;tcp
Message-ID: 929195
Use-Path: msrps://relay.example.com:3233/uwdudqd3s;tcp
-----p2pe3$
```

Figure 21.18: 200 (OK) response (5)

Then, at any time, the endpoint can create an INVITE request (6) to establish the MSRP session. This INVITE request is very similar to the first shown in Figure 21.4. The

differences are subtle: the `m=` line indicates the usage of TLS; and the `a=path` line in the SDP now contains two or more MSRPS URLs, the first pertaining to the relay (learnt from the `User-Path` header) and the other belonging to the endpoint. Figure 21.19 shows an example of this INVITE request (6).¹⁷

```

INVITE sip:Bob.Brown@example.org SIP/2.0
Via: SIP/2.0/UDP ws1.example.com:5060;branch=z9hG4bK74g3d
Max-Forwards: 70
From: Alice <sip:Alice.Smith@example.com>;tag=329s8a
To: Bob <sip:Bob.Brown@example.org>
Call-ID: 438fw34kjaljs
Cseq: 56 INVITE
Contact: <sip:alice@192.0.100.2>
Content-Type: application/sdp
Content-Length: 274

v=0
o=alice 2890844526 2890844526 IN IP4 ws1.example.com
s=-
c=IN IP4 ws1.example.com
t=0 0
m=message 8231 msrp/tls/tcp *
a=accept-types:message/cpim text/plain text/html
a=path:msrps://relay.example.com:3233/uwduqd3s;tcp
    msrps://alice.example.com:9892/98cjs;tcp

```

Figure 21.19: INVITE request (6)

Let us take a look at an end-to-end example. Figure 21.20 depicts the flow of information in an end-to-end MSRP session. We have omitted potential SIP proxies for the sake of clarity, and we have assumed that both Alice and Bob are using MSRP relays, and both have been authenticated by their respective relay.

The flow begins when Alice sends a SIP INVITE request (1) that is similar to the one described in Figure 21.19. The SDP in the INVITE request (1) contains the MSRPS URLs of both Alice and her relay. Bob receives the INVITE request (1) and replies with a 200 OK response (2) that also contains Bob's MSRPS URL and that for his relay. At this point, both Alice and Bob are able to send an MSRP SEND request that contains some content.

An example of such an MSRP SEND request is shown in Figure 21.21. The `To-Path` header includes the two MSRP relays and Bob's endpoint.

When Alice's relay receives the SEND request (3), it verifies that the session (in Alice's MSRP relay URL) is correct and bound to the TLS connection from where the request was received. Alice's MSRP relay answers (typically with a 200 OK response) and then tries to forward the request. Hence, the 200 OK merely indicates the successful reception of the SEND request at the next hop rather than the reception of the request at the remote endpoint.

Here we can see the usefulness of REPORT requests, either in success or failure circumstances. In successful cases, Bob's endpoint will send a REPORT request, after

¹⁷Note that the URLs contained in the `a=path` line should be listed in a single line, separated by just a single blank space. However, for presentation purposes, we show them separated by a carriage return and a few spaces.

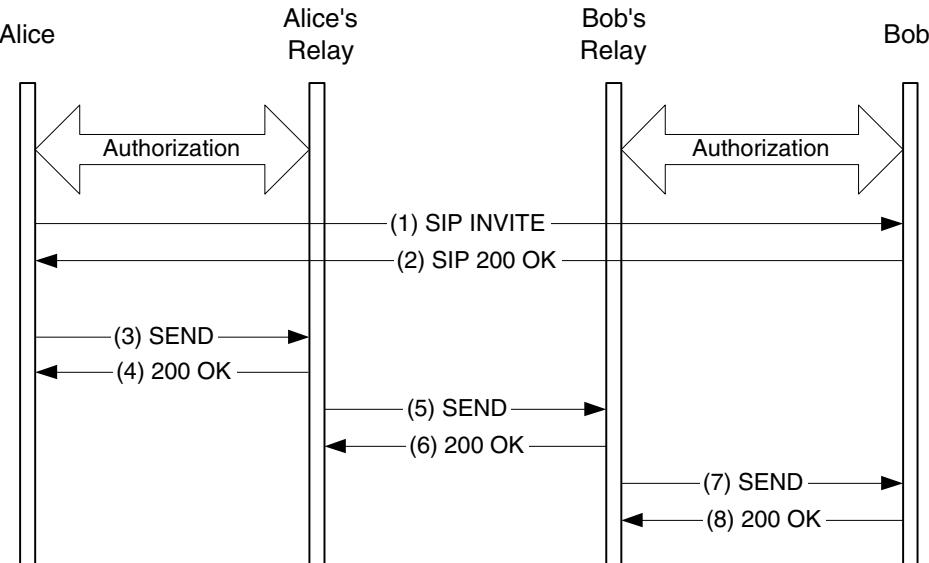


Figure 21.20: End-to-end session establishment with MSRP relays

```

MSRP 230cmqj SEND
To-Path: msrps://relay.example.com:3233/uwduqd3s;tcp
          msrps://otherrelay.example.org:23153/b8s8d;tcp
          msrp://bob.example.org:7283/d9s9a;tcp
From-Path: msrp://ws1.example.com:8231/9s9cpl;tcp
Message-ID: 193254
Byte-Range: 1-20/20
Content-Type: text/plain

This is Alice typing
-----230cmqj$
  
```

Figure 21.21: MSRP SEND request (3)

receiving the SEND request (7), that will be forwarded to Alice. In failure circumstances a relay that detects an error generates a REPORT request to the sender.

21.5 The “isComposing” Indication

Creating an instant message is an operation that involves human beings and, because of this, creating an instant message can take from a few seconds to an undetermined length of time. Sometimes it is useful to know that the other party is creating an instant message, so that the user can wait some time for that instant message to arrive.

Internet instant messaging systems have developed the *isComposing indication*. As the name suggests, this is short piece of information that indicates to the remote party that there is a current ongoing process of composing an instant message. Typically the terminal, computer,

etc., considers activity in the keyboard, mouse, keypad, or similar, to detect when the user is composing an instant message, and then, signals this information to the remote end.

A new type of isComposing XML document is specified in RFC 3394 [294]. The document is identified with the content type of `application/im-iscomposing+xml`. This XML document can be carried either in SIP MESSAGE requests (pager mode) or MSRP SEND requests (session mode), whatever is applicable at the time.

Consider the example of an isComposing XML document presented in Figure 21.22. The state element contains the main piece of information, i.e., whether the user is composing (`active`) or not (`idle`). Additional elements can be present, such as `lastactive`, which indicates the date and time at which the user was last active, `contenttype`, which indicates the type of content the user is composing, or `refresh`, which indicates the time interval at which updates will be sent.

```
<?xml version="1.0" encoding="UTF-8"?>
<isComposing xmlns="urn:ietf:params:xml:ns:im-iscomposing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <state>idle</state>
    <lastactive>2009-04-27T08:21:00Z</lastactive>
    <contenttype>audio</contenttype>
    <refresh>120</refresh>
</isComposing>
```

Figure 21.22: isComposing XML document

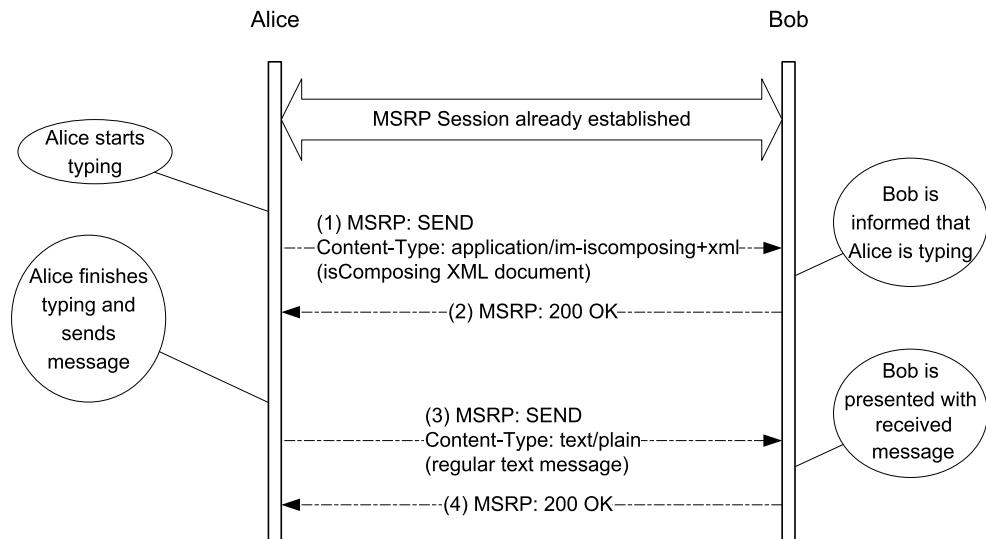


Figure 21.23: Sending isComposing indication with MSRP

Figure 21.23 describes the typical operation of the *isComposing indication*. Assume that Alice and Bob are already engaged in a messaging session. When Alice starts typing, her terminal sends an MSRP SEND request (1) that contains an isComposing XML body. Bob's

terminal receives the request and renders the information to Bob. A few seconds later, Alice finishes the composition of the instant message and sends it to Bob in an MSRP SEND request (3). Then Bob's terminal renders the received message to Bob.

21.6 Messaging Multiple Parties

There are many cases when user wants to send an instant message to more than a single recipient. There is always a possibility for the sender to send independent messages to each of the recipients, however, not only is this solution inefficient, it also precludes other parties to reply to all of the recipients. Therefore, the idea of a multi-party conference of instant messages is lost.

To address this problem, the notion of *MESSAGE URI-list services* and *chat rooms* are brought into the picture. A MESSAGE URI-list service is a server that receives one SIP MESSAGE request and sends multiple MESSAGE requests, with the same payload as the receive MESSAGE, to a number of recipients. A chat room is the counterpart for session-based messages.

21.6.1 MESSAGE URI-List Services

In order to send a page-mode instant message to multiple recipients, the terminal needs the support of a *MESSAGE URI-list Service*, which is a network server that is able to receive a single MESSAGE request and send out multiple copies of the message, one addressed to each recipient. The MESSAGE URI-list service is specified in the Internet-Draft “Multiple-Recipient MESSAGE Requests in SIP” [153]. The operation is shown in Figure 21.24, where Alice sends a MESSAGE request (1) to the URI-list Service, which replies with a 202 (Accepted) response (2). Then the URI-list service further distributes a copy of the MESSAGE request to Bob, Carol, and Dean (3, 4, and 5), respectively.

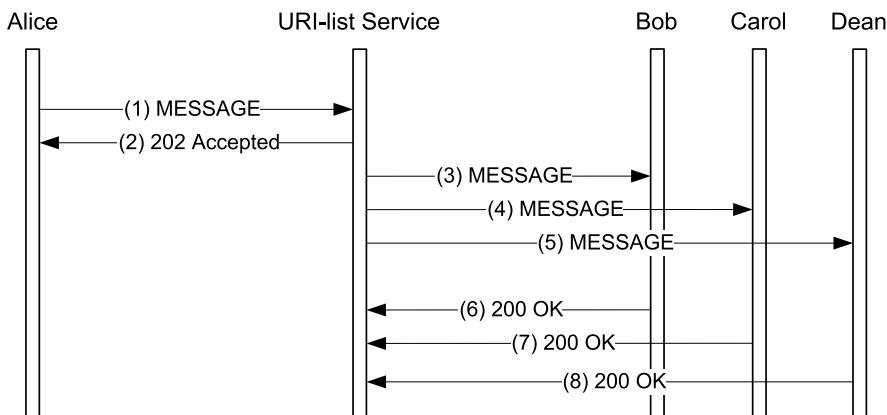


Figure 21.24: Multi-party messaging with a MESSAGE URI-list Service

So, where is the list of recipients conveyed? The list of recipient is a *resource list* XML document (specified in RFC 4826 [273]). We have already described resource lists in the context of XCAP (see Sections 17.1.1 and 17.6) and in the context of watcher subscription in

```

MESSAGE sip:uri-list@example.org SIP/2.0
Via: SIP/2.0/TCP alicepc.example.com;branch=z9hG4bK776sgd43d
Max-Forwards: 70
From: Alice <sip:alice@example.com>;tag=48912
To: URI-list service <sip:uri-list@example.org>
Call-ID: a3d3hdj5ws223ns6lk8djds
Cseq: 5 MESSAGE
Require: recipient-list-message
Content-Type: multipart/mixed;boundary="2bnasd9"
Content-Length: 552

--2bnasd9
Content-Type: text/plain

Hello folks!!!

--2bnasd9
Content-Type: application/resource-lists+xml
Content-Disposition: recipient-list

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists"
    xmlns:cp="urn:ietf:params:xml:ns:copycontrol">
    <list>
        <entry uri="sip:bob@example.com" cp:copyControl="to" />
        <entry uri="sip:carol@example.net" cp:copyControl="cc"/>
        <entry uri="sip:dean@example.com" cp:copyControl="bcc"/>
    </list>
</resource-lists>
--2bnasd9--

```

Figure 21.25: MESSAGE request (1) containing a URI list

the presence service (see Section 20.4). Now, in the context of instant messaging, resource lists contain the list of recipients of the MESSAGE request.

Still, there are two possibilities for conveying the resource list to the URI-list Service. One is that Alice attaches the resource list (the list of recipients) to the MESSAGE request (1) that she sends to the URI-list Service. This is something like an *ad-hoc* type of operation, and requires a multipart MIME body to be sent as a payload of the MESSAGE request (1). The multipart MIME body contains two bodies, one is the regular instant message payload (e.g., a text message), and the other is a resource list.

The other possibility consists of using XCAP to store the resource list in the URI-list server prior to sending the MESSAGE request (1). This is useful when the distribution list can be reused in the future for further messages. It does not require the multipart MIME body as payload of the MESSAGE request (1) because the request contains only the regular instant message payload (e.g., the text message).

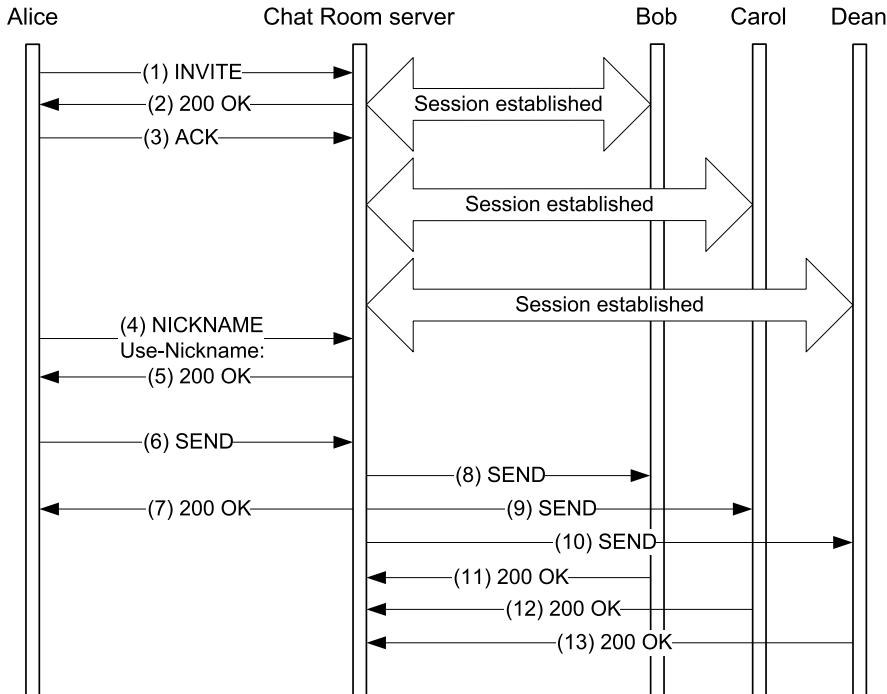


Figure 21.26: Session-based multi-party messaging: chat rooms

URIs in a resource list can also carry `copyControl` attributes (specified in the Internet Draft “XML Format Extension for Representing Copy Control Attributes in Resource Lists” [152]). A `copyControl` attribute describes why a message was delivered to a particular URI. It performs the same function as the `To`, `Cc`, and `Bcc` header fields in email.

Consider now Figure 21.25, which shows the MESSAGE request (1) that Alice sends to her URI-list server. This MESSAGE request contains a `Require` header field that includes the option-tag `recipient-list-message`, to indicate that the URI-list server must be able to understand these extensions prior to processing the message. The request also contains a multipart MIME body that, in turn, contains a plain text message and a resource list. The resource list indicates the list of recipients, each one qualified with a different `copyControl` attribute.

The resource list then creates a similar MESSAGE request (3, 4, 5) to each of the recipients. In fact, this outbound MESSAGE request (3, 4, 5) also contains a resource list that indicates the list of visible recipients (i.e., those qualified with a `to` or `cc` `copyControl` attribute), with the only difference that the `Content-Disposition` of these lists is set to `recipient-list-history` rather than `recipient-list`.

21.6.2 Chat Rooms

Chat rooms are considered as a type of session-based multi-party messaging. Typically, chat rooms are allocated with a distinct SIP URI. Users join the chat room (i.e., they establish a session of instant messages), and then they can safely exchange instant messages with the

```

INVITE sip:chatroom33@example.org SIP/2.0
Via: SIP/2.0/UDP ws1.example.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Alice <sip:Alice.Smith@example.com>;tag=hx34576sl
To: <sip:chatroom33@example.org>
Call-ID: 2098adkj20
Cseq: 22 INVITE
Contact: <sip:alice@192.0.100.2>
Content-Type: application/sdp
Content-Length: 259

v=0
o=alice 2890844526 2890844526 IN IP4 ws1.example.com
s=-
c=IN
IP4 ws1.example.com
t=0 0
m=message 8231 msrp/tcp *
a=accept-types:message/cpim text/plain text/html
a=path:msrp://ws1.example.com:8231/9s9cpl;tcp
a=chatroom:nickname private-messages

```

Figure 21.27: INVITE request (1) to join a chat room

```

MSRP b8ws9d NICKNAME
To-Path: msrps://chat.example.com:34012/bujsd;tcp
From-Path: msrp://ws1.example.com:8231/9s9cpl;tcp
Message-ID: 32092
Use-Nickname: "Alice in Wonderland"
-----b8ws9d$

```

Figure 21.28: MSRP NICKNAME request (4)

rest of users that are logged into the same chat room. Effectively, the chat room server is acting as a conference server where the media happens to be MSRP.

The chatroom functionality, which is specified in the Internet-Draft “Multi-party Instant Message (IM) Sessions Using MSRP” [222], provides support for nickname allocation to users and private messages (messages that are distributed through the chat room but to a limited distribution list).

Consider the flow of Figure 21.26, where Alice sends an INVITE request (1) to join the chat room. This INVITE request, which is shown in Figure 21.27, describes a messaging media with MSRP and it also contains a new `chatroom` attribute in the SDP offer that indicates support for nicknames and private messaging.

The chat room server replies with a 200 (OK) response (2) that contains an SDP answer, which also includes a `chatroom` attribute indicating the chat room capabilities. Then Alice can set a nickname that identifies her in that chat room. This is done with a new MSRP NICKNAME method (4). A new `Use-Nickname` header contains the suggested nickname for Alice within that chat room. Figure 21.28 shows an example of this NICKNAME request.

```

MSRP yhuh9sd SEND
To-Path: msrps://chat.example.com:34012/bujsd;tcp
From-Path: msrp://ws1.example.com:8231/9s9cpl;tcp
Message-ID: 99s9s2
Byte-Range: 1-*/*
Content-Type: message/cpim

To: <sip:chatroom33@example.com;transport=tcp>
From: <sip:Alice.Smith@example.com>
DateTime: 2008-04-27T12:53:31-03:00
Content-Type: text/plain

Hello guys, how are you today?
-----yhuh9sd$
```

Figure 21.29: Regular instant message in the chat room

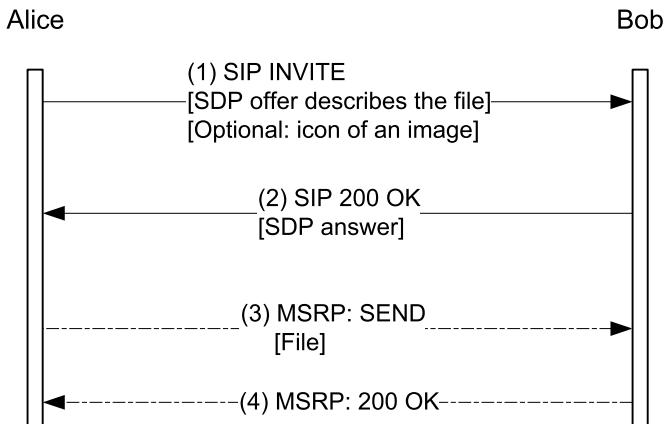


Figure 21.30: File transfer with SIP and MSRP

If that nickname is available in the chat room, the chatroom server replies with a 200 (OK) response.

Then Alice is ready for sending or receiving instant messages in the chat room. All messages sent via the chatroom server are sent in a MSRP SEND request, and always contain a Message/CPIM wrapper that encloses the actual message. The Message/CPIM wrapper is of utmost importance, because it is used to differentiate between regular instant messages, which are distributed to all of the participants in the chat room, and private messages, which have a limited distribution to a subset of the participants. Figure 21.29 shows an example of an instant message that Alice sends to all of the participants in the chat room. This is seen in the To header field of the Message/CPIM wrapper, which contains the URI of the chat room. If instead this header field had contained one or more URIs pertaining to participants, then the message would have been considered a private message and the chat room server would have distributed only to those participants.

```

INVITE sip:Bob.Brown@example.org SIP/2.0
Via: SIP/2.0/UDP ws1.example.com:5060;branch=z9hG4bK74gh5
Max-Forwards: 70
From: Alice <sip:Alice.Smith@example.com>;tag=hx34576s1
To: Bob <sip:Bob.Brown@example.org>
Call-ID: 2098adkj20
Cseq: 22 INVITE
Contact: <sip:alice@192.0.100.2>
Content-Type: multipart/related; type="application/sdp" ;
    boundary="boundaryfiletransfer"
Content-Length: 1836

--boundaryfiletransfer
Content-Type: application/sdp
Content-Length: 571

v=0
o=alice 2890844526 2890844526 IN IP4 alicepc.example.com
s=
c=IN IP4 alicepc.example.com
t=0 0
m=message 7834 TCP/MSRP *
i=My picture
a=sendonly
a=accept-types:message/cpim
a=accept-wrapped-types:*
a=path:msrp://ws1.example.com:7834/ba98d;tcp
a=file-selector:name:"Picture.jpg" type:image/jpeg
size:4092 hash:sha-1:
72:24:5F:E8:65:3D:DA:F3:71:36:2F:86:D4:71:91:3E:E4:A2:CE:2E
a=file-transfer-id:Q6LMoGymJdh0IKIgD6wD0jkcfgva4xvE
a=file-disposition:render
a=file-date:creation:"Sun, 27 May 2008 14:21:31 +0300"
a=file-icon:cid:id2@ws1.example.com

--boundaryfiletransfer
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <id2@ws1.example.com>
Content-Length: 1024
Content-Disposition: icon

[file icon]
--boundaryfiletransfer--

```

Figure 21.31: INVITE request (1) for a file transfer

Typically participants in a chat room create a subscription to the *conference event package*, in order to learn the list of participants in the chat room, their identities, nicknames, etc. We describe the conference event package in further detail in Chapter 23.

21.7 File Transfer

Most of the instant messaging clients used in the Internet provide a mechanism to facilitate the exchange of files between two users who are engaged in a session of instant messages. This is a so-called file transfer operation. SIP also allows similar file transfer operations.

SIP file transfer operations are modeled with the description of the file in SDP, but the actual transfer of the file takes place with MSRP (remember that MSRP is able to transfer any content of any arbitrary length, including files). File transfer, which is specified in the Internet-Draft “SDP Offer/Answer Mechanism to Enable File Transfer” [155], offers two modes of operation: push and pull. In push mode, a user suggests sending a file to the second user. In pull mode, a user requests the other user to send her a file.

Figure 21.30 shows a typical flow for a file transfer operation. Alice selects a file, constructs a description of the file in an SDP offer, and sends an INVITE request (1) that contains this SDP offer describing the file. This INVITE request (1) can also be a re-INVITE, if a SIP session of any media (MSRP, audio, video, etc.) was already established between both parties. If the file that Alice wants to send Bob is an image, then she can also attach a small icon to the INVITE request, although then the SDP and the icon constitute a multipart MIME type.

Figure 21.31 shows an example of an INVITE request that contains a multipart MIME body. One of the enclosed bodies is an SDP offer that describes a file to be sent; the other contains an icon of the file. The SDP contains a `sendonly` attribute, indicating a push operation (so Alice will send a file to Bob). A new `file-selector` attribute describes the file, including its name, type, size, and hash. Other new SDP attributes complete the file description.

Bob’s terminal receives this INVITE request (1) and alerts Bob with the file data that Alice wants to send, including the icon. Then Bob can decide, based on the title of the image, size, date of creation, etc., whether he wants to receive file. In case he accepts, he sends a 200 (OK) response that also contains SDP accepted the MSRP session. Then Alice can send the file in a single interruptible MSRP SEND request (3), or as a sequence of chunked MSRP SEND requests (each one containing a chunk of the file).

Chapter 22

The Instant Messaging Service in the IMS

Chapter 21 described the basic components of the instant messaging service. We learnt that there are two modes of operation: pager mode and session-based mode. In this chapter we analyze how these two modes are applied to the IMS. We explore the basic call flows and present examples of services that can be enriched with instant message capabilities.

22.1 Pager-mode Instant Messaging in the IMS

The pager-mode instant messaging service was introduced with the first phase of IMS that came as part of Release 5 of the 3GPP specifications. 3GPP TS 23.228 [43] already contained requirements for Application Servers (ASes) and S-CSCFs to be able to send textual information to an IMS terminal. 3GPP TS 24.229 [37] introduces support for the MESSAGE method extension. The specification mandates IMS terminals to implement the MESSAGE method (specified in RFC 3428 [115]) and to allow implementation to be an optional feature in S-CSCFs and ASes (e.g., if required by the service). Obviously, pager-mode instant messages are subject to the constraints (e.g., message size, etc.) that we described in Section 21.3.

The main purpose of a pager-mode instant message is to allow the S-CSCF or ASes to send short instant messages to the IMS terminal. Since the MESSAGE method is already implemented in IMS terminals, users are able to send pager-mode instant messages to other IMS users. The flow is simple, as depicted in Figure 22.1.

An example of a service provided with the SIP MESSAGE method is shown in Figure 22.2. An AS is the controller of a voicemail system. The AS is interested to know when the user successfully logs on to the IMS, so that the AS can inform the user that there are pending voicemails to be retrieved. The service is implemented as follows: the user registers with the IMS as usual, (1)–(20) in Figure 22.2. When the registration is complete the S-CSCF evaluates the initial filter criteria. One of the initial filter criteria indicates that the S-CSCF should perform a third-party registration with a particular AS. The S-CSCF then sends a third-party REGISTER request (21) to the indicated AS. The purpose of the third-party REGISTER request is not to register the user with the AS, but instead to indicate to the AS that the user has just registered with the S-CSCF. Upon receipt of the REGISTER

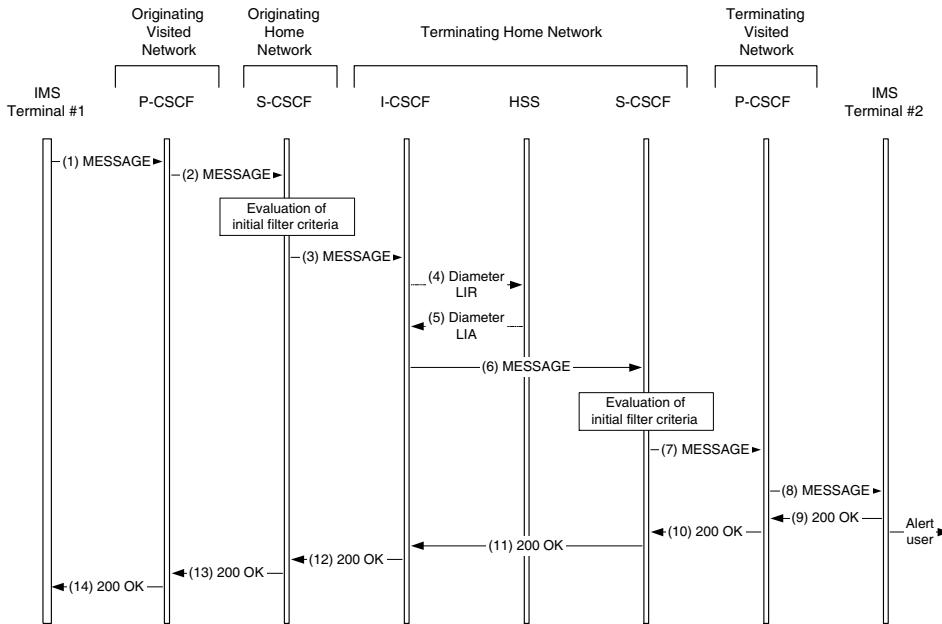


Figure 22.1: Pager-mode instant messaging in the IMS

request (21) the AS generates a MESSAGE request (23) that contains some informative text, maybe a link to a website that holds the transcription of the pending messages to retrieve or any other information that the service designer considers appropriate. The MESSAGE request is forwarded via the S-CSCF and P-CSCF as any other SIP message.

22.2 Pager-mode Instant Messaging to Multiple Recipients

IMS also allows IMS terminals to send MESSAGE request to multiple recipients. This is done in cooperation with the help of a List Server, which takes the role of a URI-list server. In fact, the technology is a straight-forward application of MESSAGE URI-list services that we described in Section 21.6.1.

The UE is able to send the list of recipients attached to the MESSAGE request, as a resource list, along with the instant message. The MESSAGE request is addressed to the Public Service Identity of the list AS that executes the multiple messaging service.

IMS terminals can also indicate when the user is composing a message, by making use of the `isComposing` feature for instant message. We described the `isComposing` feature in Section 21.5.

22.3 Session-based Instant Messaging in the IMS

The session-based instant messaging service was introduced in Release 6 of the 3GPP specifications. The detailed protocol specification is described in 3GPP TS 24.247 [45]. Session-based instant messaging was not included in Release 5 because at the time 3GPP closed

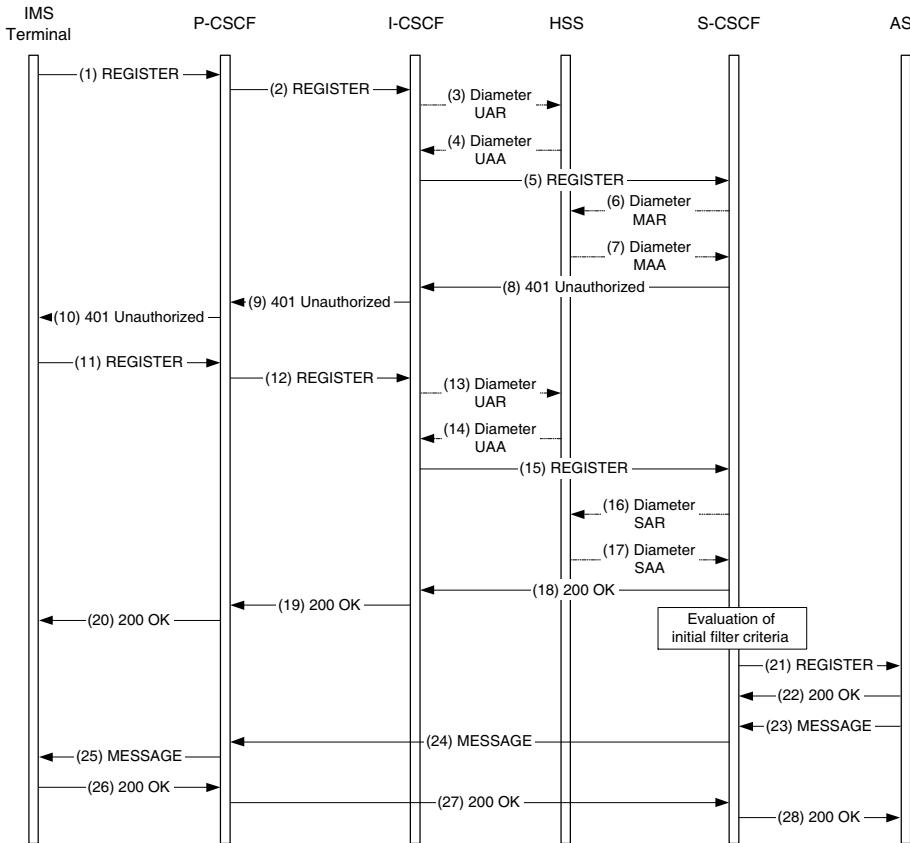


Figure 22.2: Example of a service provided with pager-mode instant messages

Release 5 the IETF had just started work on session-based instant messaging. Therefore, the functionality of session-based instant messaging was postponed until Release 6.

We described in Section 21.4 how to establish a session of instant messages with an INVITE request that contains provisions in SDP for the instant message media. The Message Session Relay Protocol (MSRP) is the actual protocol used to transport the messages.

In the IMS, MSRP is implemented in the IMS terminals. In addition, the MRFP may also implement MSRP. The reason behind this is that there are two different scenarios for establishing a session of instant messages. In the first scenario, which we show in Figure 22.3, an IMS terminal establishes a session toward another endpoint. SIP messages traverse regular IMS nodes (P-CSCFs, S-CSCFs, perhaps ASes, etc.). MSRP is then sent end-to-end. The only difference from a basic session setup consists of the absence of the precondition extension requirement in the INVITE request, if session-based messaging is the only media stream declared in SDP. This is the reason for not having 183 (Session Progress) responses, PRACK, or UPDATE requests in the flow.

In the second scenario the MRFC and MRFP are intermediaries in the network. This might be a result of operator constraints, such as the ability to generate charging events

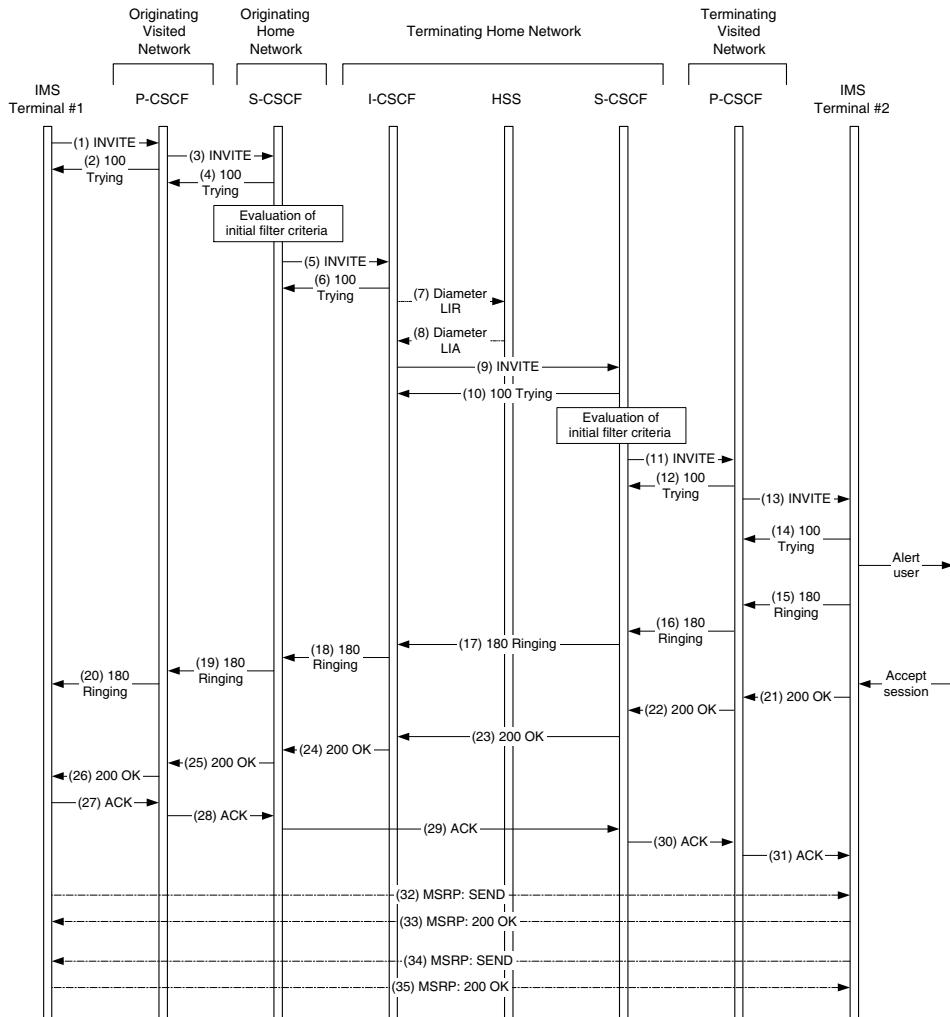


Figure 22.3: Session-based instant messages: end-to-end MSRP session

depending on the size of the message or on any additional contents in MSRP SEND messages. Another reason might be that the MRF is acting as a multiparty conference unit for instant messages (also known as chat rooms). Figure 22.4 shows the flow for a multiparty conference. For the sake of simplicity we assume that both users who join the conference belong to the same network operator, although they may have allocated different S-CSCFs and P-CSCFs. In the figure a first user sends an INVITE request (1) that traverses their allocated P-CSCF and S-CSCF. Their S-CSCF forwards the INVITE request (5) to the MRFC. The MRFC, which controls the MRF by means of the H.248 protocol (7), creates a new termination for the user. Then the user sends an empty MSRP SEND request (14), i.e., where there is no body. This allows the MRF to bind the TCP connection to the MSRP session.

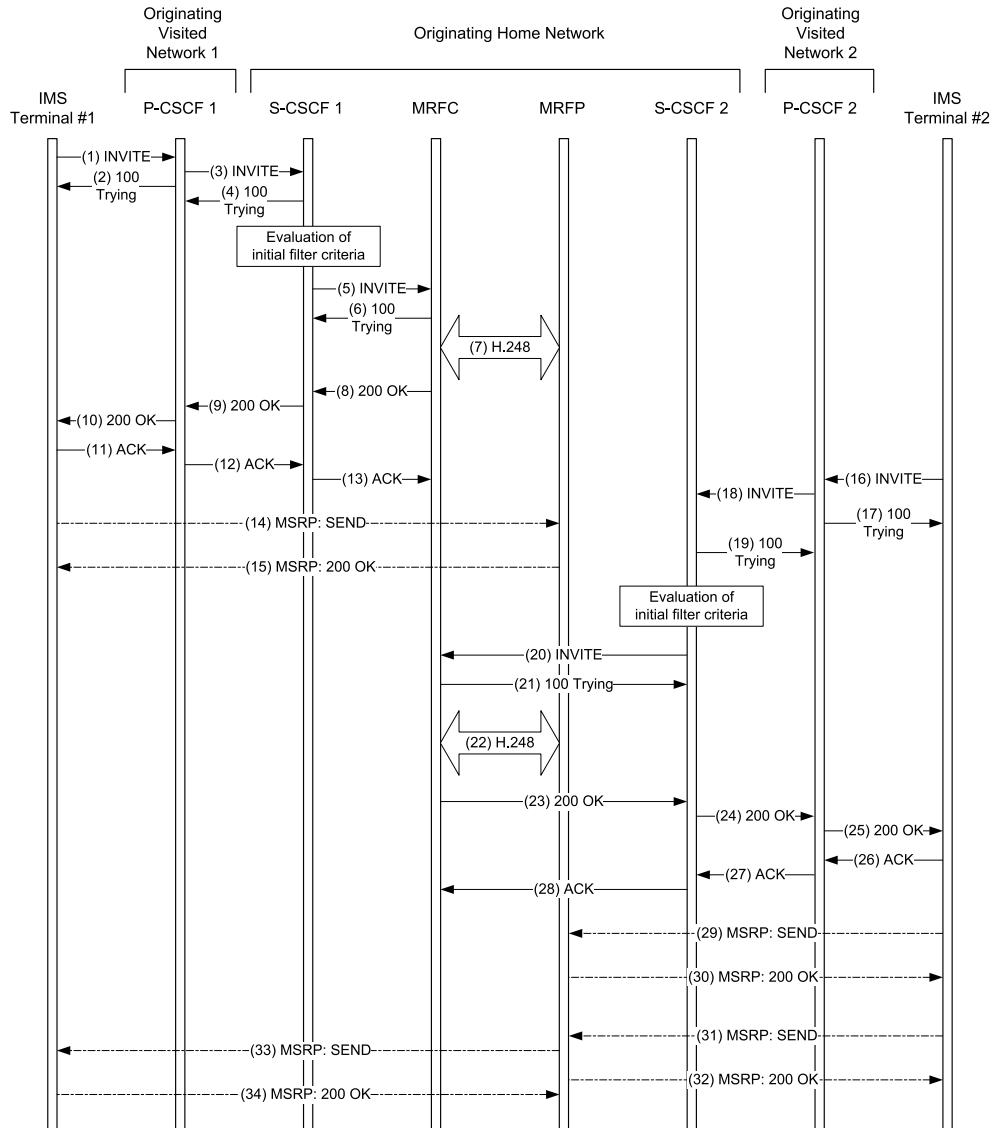


Figure 22.4: A multi-party session-based conference (chat server)

Later, a second user joins the conference and establishes another session with the MRFC. Once the SIP session is established, the user also sends an empty MSRP SEND request (29) that allows the MRFP to bind the TCP connection to the MSRP session. At any time any of the users can send an instant message that is transported over an MSRP SEND request: for instance, when the second user sends an MSRP SEND request (31) the MRFP sends a copy of it (33) to the remaining participants of the conference.

22.4 File Transfer

IMS also supports the transfer of files with SIP and MSRP. We have already described this technology in Section 21.7: this service is a straight-forward application of the file transfer mechanism that we described earlier.

Chapter 23

Conferencing on the Internet

Conferencing involves communication among several users. Multimedia conferencing, including audio, video, instant messaging, whiteboard sharing, and file transfer, is a popular service on the Internet and in enterprises. Chat rooms where users exchange instant messages are an example of a conference service on the Internet. The collaboration tools used in most enterprises are also examples of conferences.

Thus, conferences are not limited to traditional unmoderated audio or video conferences. They can include all types of media and can be moderated by using floor control mechanisms.

Conferencing is an important area for enterprises with employees working in different countries. A conference system including collaboration tools can save much money and time by reducing the need for face-to-face meetings where attendees need to travel great distances.

However, we are still far from having conference systems that can replace face-to-face meetings completely. That is why there is much ongoing research in areas such as telepresence and virtual reality. The goal is to make virtual interactions as close to real ones as possible.

23.1 Conferencing Standardization at the IETF

In the past, working groups such as MMUSIC did some work on conferencing (e.g., SDP was designed with multiparty sessions in mind). Lately, the working groups that have been active in this area have been SIPPING and XCON. In fact, implementers sometimes find it confusing to have similar specifications in the same area coming from two different working groups. Knowing the history behind conferencing standardization at the IETF will help readers understand how the specifications coming from both working groups relate among them.

Initially, the SIPPING working group developed a set of specifications that described how to provide conferencing services using SIP. Coming from the SIPPING working group, these specifications were, unsurprisingly, very much focused on SIP. Pieces needed to build a complete conference service such as floor control and conference management mechanisms (beyond the simple ones SIP provides) were out of the scope of this work.

The XCON working group was chartered to work on generalizing the work done in SIPPING so that different signaling protocols (not only SIP) could be used and to specify those missing pieces needed to build a complete conference system. The charter was limited to centralized conferences where clients connect to a central server following a star topology.

Conferences using different topologies such as full-meshed and cascaded conferences were left out of scope.

The results of the work of these two working groups include two conferencing frameworks: the SIPPING conferencing framework and the XCON conferencing framework. We discuss both of them, their differences, and how they relate to each other.

23.2 The SIPPING Conferencing Framework

The SIPPING conferencing framework (specified in RFC 4353 [272]) describes three conferencing models: loosely coupled, fully distributed, and tightly coupled. In the loosely-coupled conferencing model, shown in Figure 23.1, media streams are multicast. Conference participants join the multicast group of the conference using, for example, IGMP (Internet Group Management Protocol, specified in RFC 3376 [95]) in order to receive media. Conference participants do not typically have any signaling relationship between them. Still, they can use SIP to invite new participants into the conference. A SIP INVITE request sent to a new participant would contain (in its body) all information needed to join the multicast group.

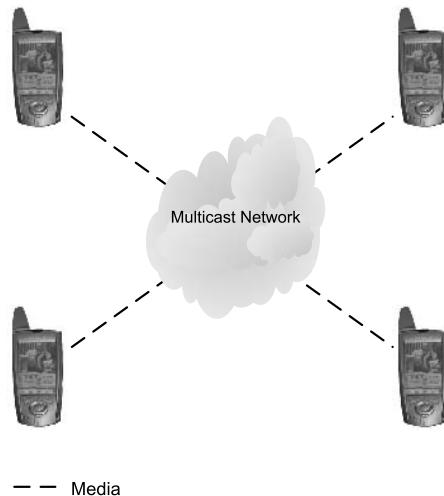


Figure 23.1: The loosely-coupled conference model

In the fully-distributed conferencing model, shown in Figure 23.2, each participant has a signaling relationship with all of the other participants in the conference. Each participant sends media to all of the other participants.

In the tightly-coupled conferencing model, shown in Figure 23.3, each participant has a signaling relationship with a central conference server. The central conference server mixes the media received from different participants and distributes it to all of them.

Of course, the three conferencing models just described are not the only models that can be implemented with SIP. Many other variants are possible. For example, when the central conference server in a tightly-coupled conference is distributed among several SIP nodes, the resulting model is typically referred to as the cascaded conferencing model. In any case, the SIPPING conferencing framework focuses on the tightly-coupled conferencing model; the rest of the models are considered to be out of scope of our work.

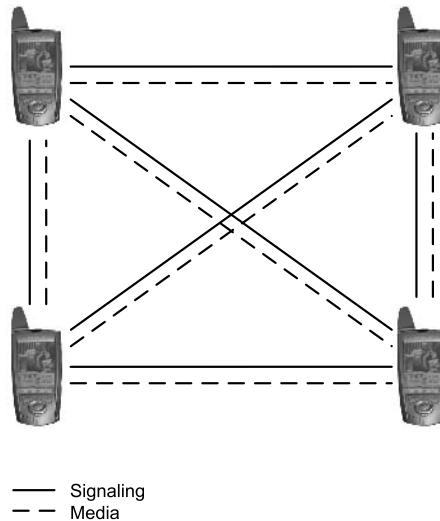


Figure 23.2: The fully-distributed conference model

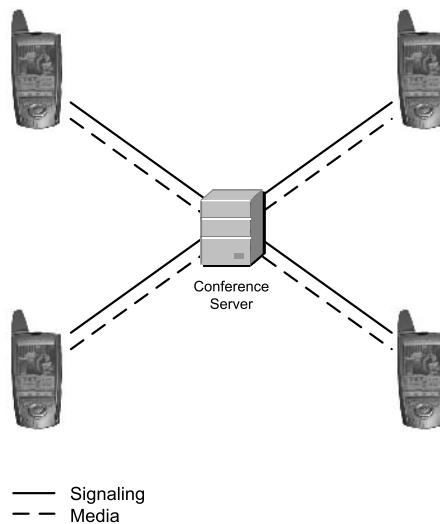


Figure 23.3: The tightly-coupled conference model

23.2.1 Signaling Architecture

Figure 23.4 shows the signaling architecture proposed by the SIPPING conferencing framework. The conference server consists of several logical functions: the conference policy, the conference policy server, and the focus, which includes the conference notification service.

The conference policy is the set of rules that define a conference. The conference policy includes information about the participants of the conference, the time and date when the conference will take place, the media streams the conference has, etc. Participants manipulate

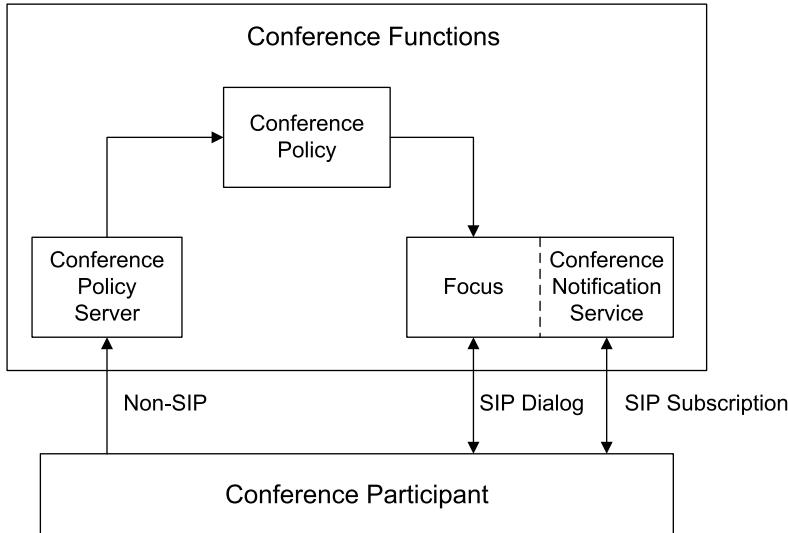


Figure 23.4: Signaling architecture in the SIPPING framework

the conference policy (e.g., to add a video stream to an audio-only conference) through the conference policy server. The protocol between participants and the conference policy server is left unspecified.

The focus interacts with the conference participants using SIP. It acts as a user agent towards all of the participants. The focus includes the conference notification service, which provides participants with information about the conference using the SIP event package for the conference state (specified in RFC 4575 [289]). This event package defines an XML-based format to convey conference-related information. Figure 23.5 shows an example of a document that uses this format. This document, which is mostly self-explanatory, describes a conference and provides information about two of its participants: Bob and Alice. Bob was kicked out from the conference because he experienced bad voice quality and Alice was brought in into the conference by Mike. Note that even though the number of participants in the conference is 33 (see the `<user-count>` element), the document only provides detailed information about two of them (Bob and Alice). Conferencing servers can omit information about certain users for policy reasons.

The XML document in Figure 23.5 is already fairly long, even though it only carries information about two users. A document describing a large conference with many users would be much longer. In principle, every time a small change occurs in the conference (e.g., one user leaves the conference), the conference notifications service would need to send a new large XML document that would very similar to the last one it sent (e.g., the only difference would be in the elements related to the user that left). This would result in a non-efficient bandwidth use.

In order to avoid this situation, the SIP event package for conference state implements a mechanism for partial notifications. The “state” attribute indicates whether an element carries full or partial information. In addition, the “state” attribute can also indicate that an element

```
<?xml version="1.0" encoding="UTF-8"?>
<conference-info
    xmlns="urn:ietf:params:xml:ns:conference-info"
    entity="URI}sips:conf233@example.com"
    state="full" version="1">
    <!-- CONFERENCE INFO -->
    <conference-description>
        <subject>Agenda: This month's goals</subject>
        <service-uris>
            <entry>
                <uri>http://sharepoint/salesgroup/</uri>
                <purpose>web-page</purpose>
            </entry>
        </service-uris>
    </conference-description>
    <!-- CONFERENCE STATE -->
    <conference-state>
        <user-count>33</user-count>
    </conference-state>
    <!-- USERS -->
    <users>
        <user entity="sip:bob@example.com" state="full">
            <display-text>Bob Hoskins</display-text>
        <!-- ENDPOINTS -->
        <endpoint entity="sip:bob@pc33.example.com">
            <display-text>Bob's Laptop</display-text>
            <status>disconnected</status>
            <disconnection-method>departed</disconnection-method>
            <disconnection-info>
                <when>2005-03-04T20:00:00Z</when>
                <reason>bad voice quality</reason>
                <by>sip:mike@example.com</by>
            </disconnection-info>
        <!-- MEDIA -->
        <media id="1">
            <display-text>main audio</display-text>
            <type>audio</type>
            <label>34567</label>
            <src-id>432424</src-id>
            <status>sendrecv</status>
        </media>
    </endpoint>
</user>
```

Figure 23.5: Example of an XML-based conference description (part 1)

```

<!-- USER -->
<user entity="sip:alice@example.com" state="full">
<display-text>Alice</display-text>
<!-- ENDPOINTS -->
<endpoint entity="sip:4kfk4j392jsu@example.com;grid=433kj4j3u">
<status>connected</status>
<joining-method>dialed-out</joining-method>
<joining-info>
<when>2005-03-04T20:00:00Z</when>
<by>sip:mike@example.com</by>
</joining-info>
<!-- MEDIA -->
<media id="1">
<display-text>main audio</display-text>
<type>audio</type>
<label>34567</label>
<src-id>534232</src-id>
<status>sendrecv</status>
</media>
</endpoint>
</user>
</users>
</conference-info>

```

Figure 23.6: Example of an XML-based conference description (part 2)

has been deleted. Accordingly, the “state” attribute can take on the following values: full, partial, or deleted. An element with a “state” attribute with a value of partial carries only the information that has changed since the previous document was sent to the participant.

If a parent element has a “state” of full, all of its child elements should also have a “state” of full. On the other hand, if a parent element has a “state” of partial, its child elements can have any “state”. The default value for the “state” attribute is full. The only elements that can carry a “state” attribute are `<conference-info>`, `<users>`, `<user>`, `<endpoint>`, `<sidebars-by-val>`, and `<sidebars-by-ref>`. In Figure 23.5, all of the “state” attributes have a value of full.

23.2.2 Media Architecture

The SIPPING conferencing framework describes the following media plane realizations: centralized server, endpoint server, media server component, distributed mixing, and cascaded mixers.

In the centralized-server model, a central server handles both signaling and media, as shown in Figure 23.3. In the endpoint-server model, one of the endpoints behaves as the central server in the centralized-server model, as shown in Figure 23.7. The endpoint-server model is typically the result of a two-party call between two endpoints that transitions into an *ad-hoc* conference. This is the case when the users involved in the original two-party call decide to bring in one or more additional users into the call at some point.

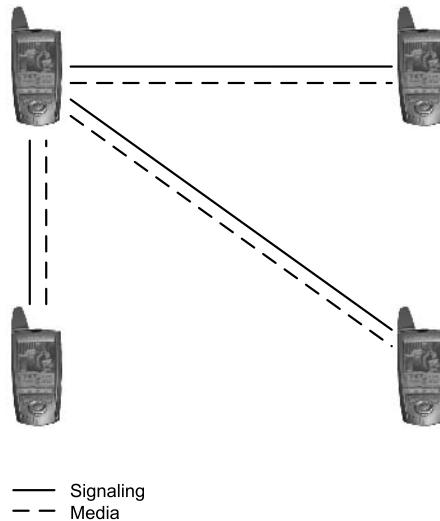


Figure 23.7: The endpoint-server model

The endpoint-server model works well when the endpoint performing the mixing does not have processing, bandwidth, or battery constraints. Conferences between endpoints with those constraints are better handled by a central server.

In the media-server-component model, the central server of the centralized-server model is divided into two servers: an application server and a mixing server. The application server interacts with the conference participants but does not have mixing capabilities. The mixing server performs the actual media mixing.

The interface between the application server and the mixing server is based on SIP. The application server can use SIP mechanisms such as third-party call control (specified in RFC 3725 [282]) to instruct the mixing server how to mix the conference's media streams.

The SIPPING conferencing framework does not talk about distributed conference servers that use a protocol other than SIP (e.g., H.248 [189]) between the server handling SIP signaling (i.e., hosting the focus) and the server performing the mixing. However, this model can be considered a special case of the centralized-server model in which the internal structure of the server is distributed.

In the distributed-mixing model, the central server of the centralized-server model handles signaling but not media. The central server does not have any media mixing capabilities; instead, it instructs users to exchange media among them. In this model, the conference server is, effectively, a third-party call controller (as specified in RFC 3725 [282]). Figure 23.8 shows how, in this model, media can be exchanged using unicast or multicast.

In the cascaded-servers model, the mixing functionality is distributed among several physical mixers. The central server handling the signaling of the conference coordinates all of the mixers so that all users receive the conference's media correctly.

23.3 The XCON Conferencing Framework

As discussed earlier, the XCON working group was chartered to work on generalizing the work on conferencing performed on SIPPING, which was specific to SIP. The XCON

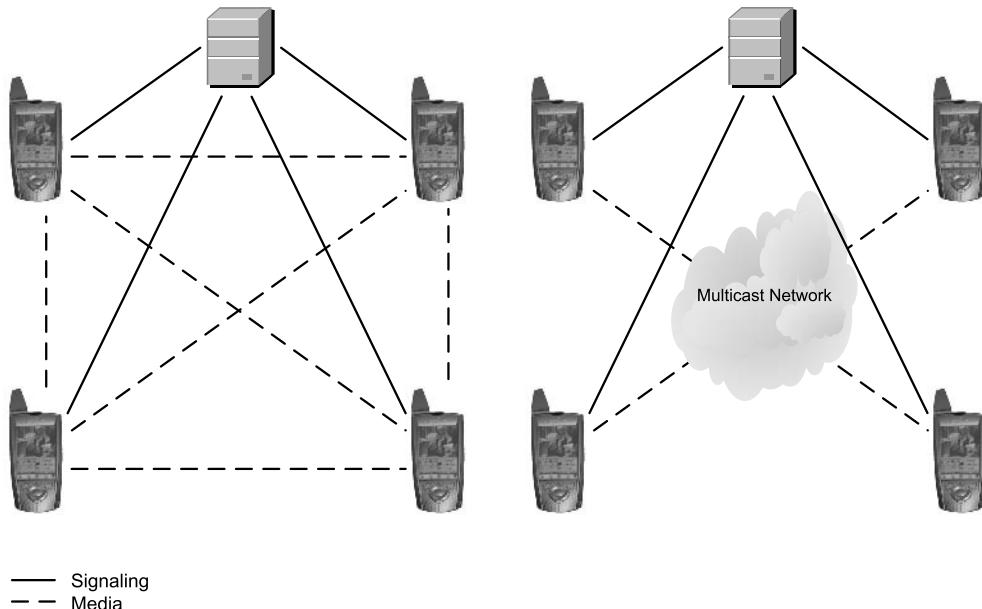


Figure 23.8: The distributed-mixing model

framework (specified in the Internet-Draft “A Framework and Data Model for Centralized Conferencing” [82]) defines the conferencing architecture shown in Figure 23.9. This figure shows a conference system able to host several conferences. That is why the figure shows more than one conference object.

23.3.1 Conference Objects

A conference object contains all of the information related to a given conference. It is the same concept as the conference policy in the SIPPING conferencing framework (see Figure 23.4) with a different name.

Figure 23.5 shows an example of the XML-based format to describe conference policies developed by the SIPPING working group (which is specified in RFC 4575 [289]). The XCON working group extended this format so that it can be used to describe more general conferences (i.e., not only SIP-based conferences) and to provide more information about a given conference (e.g., floor-control-related information was missing from the original format and was added by the XCON working group). The resulting format is referred to as the XCON data model (which is specified in the Internet-Draft “Conference Information Data Model for Centralized Conferencing (XCON)” [224]).

The improvements in the XCON data model, with respect to the original format defined by the SIPPING working group, include the ability to carry different types of URIs and the inclusion of information that relates to floor control, conference scheduling, and media controls (e.g., a control to mute a media stream).

In order to create a conference, it is necessary to create its conference object. The initial values for the variables of a conference object are typically taken from a conference blueprint. A conference blueprint is a template to create conference objects. For example, a conference

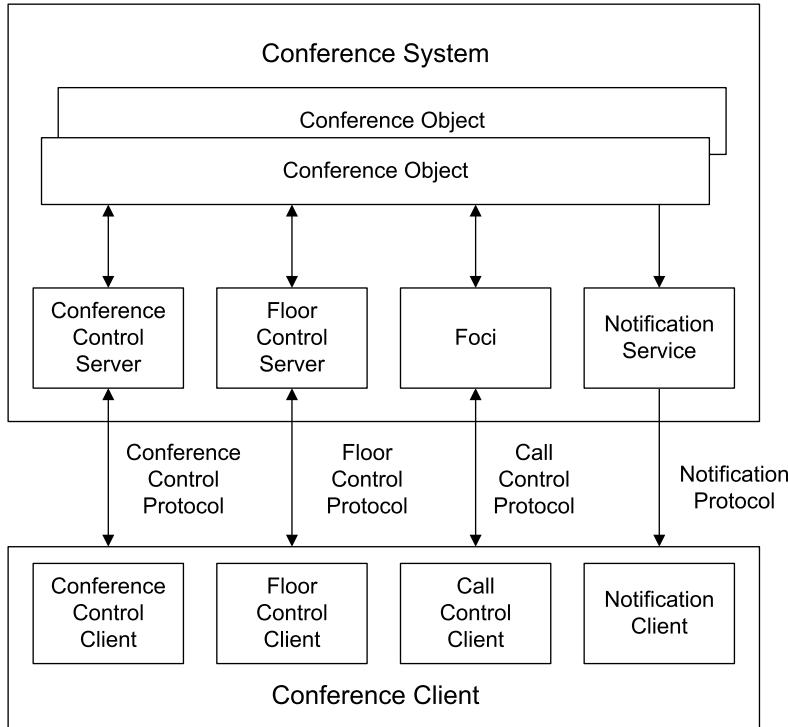


Figure 23.9: XCON architecture

system may have a conference blueprint with the typical values to create an audio-only conference.

23.3.2 *Conference Control Server*

Users can manipulate conference objects and, thus, the properties of any conference, using a conference control protocol. Such a protocol runs between the participant's conference control client and the conference control server.

The XCON working group is chartered to develop a conference control protocol. We expect this working group to specify such a protocol in the future. One of the main decisions concerning this protocol is whether it should follow a semantic approach or a syntactic approach.

A semantically-oriented protocol would have primitives to perform conference-related operations such as create a conference, add a user to a conference, and remove a media stream from a conference. Such primitives would have an effect on a conference object (which is described by an XML document). For example, the creation of a conference would create a new conference object. The addition of a user would add a new `<user>` element to the XML document describing the conference object.

A syntactically-oriented protocol would have primitives to operate directly at the XML level. For example, in order to add a user to a conference, the protocol would directly instruct

the conference control server to add a <user> element to the XML document describing the conference object.

Both approaches have advantages and disadvantages. A syntactically-oriented protocol may initially be more complex since it would need to provide general XML manipulation mechanisms. On the other hand, it would not need to be extended in order to manipulate new data model elements that may be defined in the future. A semantically-oriented protocol may initially be simpler and, in general, more efficient but would need to be extended in order to perform new operations. Specifying policies (e.g., only the moderator can add new user into the conference) seems to be easier if the semantic approach is followed.

The XCON working group started working on an XCAP-based protocol that followed a syntactic approach. However, that protocol was abandoned and, at present, it seems that the conference control protocol to be developed by XCON will follow a semantic approach.

23.3.3 *Foci and Notification Service*

As in the SIPPING conferencing framework, an XCON focus has a signaling relationship with the user agents in the conference. However, in the XCON framework, a conference can have multiple foci; each one handling a different protocol (e.g., SIP and H.323).

In the SIPPING framework, both the focus and the notification service used SIP and, thus, were part of the same logical entity. The XCON framework separates them into two different logical entities because they can use different protocols.

As discussed earlier, the XCON data model extends the XML-based format used by the SIPPING notification service (which is specified in RFC 4575 [289]). The XCON notification service needs to be able to use this extended format (i.e., the XCON data model) in its notifications. An extension to the SIP event package for conference state (also specified in RFC 4575 [289]) has been defined so that the event package can carry information in the format specified in the XCON data model (this extension is specified in the Internet-Draft “Conference Event Package Data Format Extension for Centralized Conferencing (XCON)” [113]).

23.3.4 *Floor Control Server*

Floor control is used to manage the access to a shared resource. Examples of resources in a conferencing environment are a shared whiteboard, a video stream, and a voice stream. The user that has the floor corresponding to a resource at a given moment is allowed to access the resource. For example, the user that has the floor corresponding to a shared whiteboard, is allowed to draw on the whiteboard.

It is important to note the difference between not being allowed to do something and actually being kept from doing it. Let us think of a face-to-face conference where all participants have their own microphone. The conference’s chair will indicate which participant can speak (e.g., to ask a question) at a given time. However, the chair does not need to manage access to the microphones. If the participants are polite enough, they will only talk into their microphones when they are told to by the chair.

However, if participants start talking when it is not their turn, the chair may have to disable all of the microphones except the one of the participant that has the floor at any given time.

Therefore, the fact that a conference uses floor control does not imply that floor-control-related decisions are enforced in any way. They may or may not be enforced, depending on the environment.

In XCON, the enforcement of floor-control-related decisions is outside the scope of floor control. That is, the floor control server uses a floor control protocol to communicate with its clients. However, if the floor control protocol wants to enforce its decisions, it will use a different protocol. For example, the floor control server could use H.248 to instruct the conference's mixer to ignore incoming media from participants that do not hold the floor.

A conference can have multiple floors. Each of them can control the access to a different resource within the conference. A floor control server can have different policies regarding multiple floors. For example, in a video conference, the floor control server can grant the video floor to the participant holding the audio floor (i.e., video follows the speaker). However, if the current speaker does not have a camera, the floor control server can grant the video floor to the conference's chair.

A floor control server receives floor requests from different participants and needs to decide which floor request to grant. A floor control server can implement an automatic algorithm to make this decision (e.g., first come, first served) or contact a conference chair so that he or she makes it.

When a floor control server receives multiple floor requests for the same floor, it does not need to grant or deny all floor requests immediately. It can put them in a queue and grant them one by one at a later point. In this way, participants requesting the floor do not need to keep sending floor requests until one is granted.

The XCON WG has specified a floor control protocol: BFCP (Binary Floor Control Protocol, specified in RFC 4582 [106]), which is discussed in the following section.

23.4 The Binary Floor Control Protocol (BFCP)

BFCP (specified in RFC 4582 [106]) was designed to be a simple protocol. The goal was to provide enough functionality for well-defined scenarios.

The scenarios which BFCP had to be able to cover included those with low-bandwidth links. BFCP needed to be able to meet the delay requirements of applications such as Push-to-talk, where users often use low-bandwidth radio links and the time from when a user requests to speak (i.e., requests a floor) until the user is allowed to do so (i.e., the floor is granted) should be fairly short. Because of these design constraints, BFCP was designed to use a binary encoding.

Figure 23.10 shows the BFCP architecture. There is a central floor control server that communicates with floor participants and floor chairs. Floor participants request floors from the floor control server, which can contact floor chairs in order to decide whether or not to grant those floor requests.

Of course, a floor participant can also act as a floor chair. These roles are defined on a transaction-by-transaction basis. In one transaction a client can act as a floor participant and in the next transaction the same client can act as a floor chair.

Floor participants and floor chairs can request to be informed about the status of a particular floor or a particular floor request. The floor control server sends notifications every time the status of the floor or the floor request changes.

BFCP supports third-party floor requests. That is, a floor participant can request a floor for another participant. This functionality is useful for distributed conferencing clients that implement floor control functionality and media handling in different devices.

A floor participant can request more than one floor in an atomic operation. Such a floor request will only be granted when all of the floors requested can be granted to the participant.

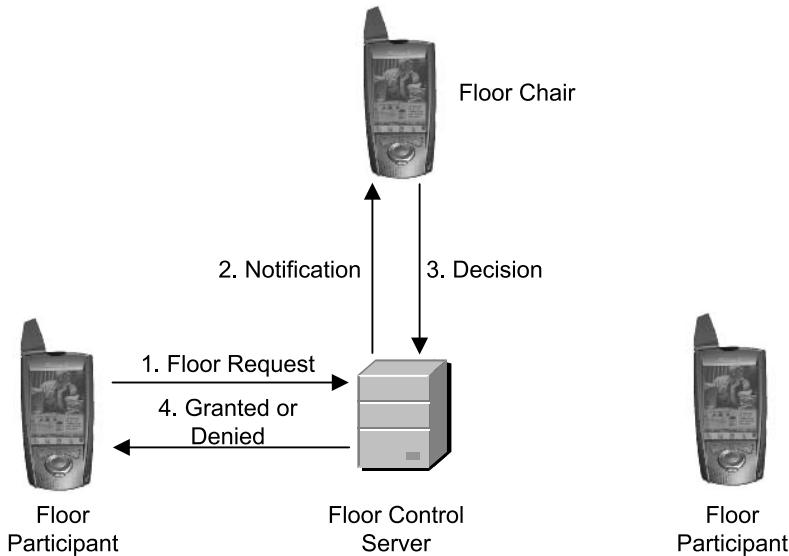


Figure 23.10: BFCP architecture

23.4.1 Contacting the Floor Control Server

In order to contact a conference's floor control server, the conference participants (which will act as BFCP clients when they contact the floor control server) need to obtain the floor control server's address. Once a BFCP client (the term BFCP client refers to a floor participant or a floor chair that communicates with a BFCP floor control server) obtains the floor control server's address, the client establishes a TCP connection with the server in order to be able to exchange BFCP messages between them.

There are two ways for clients to establish a connection with a floor control server: inside and outside the context of an offer/answer exchange.

23.4.1.1 Inside an Offer/Answer Exchange

Connections within the context of an offer/answer exchange are established in the same manner as any other media stream. The client and the server perform an offer/answer exchange using SIP where the exchange the parameters needed to establish the connection (e.g., IP addresses and port numbers). In addition, the floor control server's SDP description (it can be the offer or the answer in the offer/answer exchange) also contains parameters related to BFCP. These parameters are the conference ID and the floor IDs to be used by the client, the role each endpoint will perform (client or floor control server), and the relation between floors and media streams (the SDP format to establish BFCP connections is specified in RFC 4583 [99]).

Figure 23.11 shows an example of an SDP session description generated by a floor control server (only “m” and “a” lines are shown for simplicity reasons). The “setup” and “connection” attributes (specified in RFC 4145 [319]) relate to the establishment of the TCP connection. The “fingerprint” attribute (specified in RFC 4572 [206]) relates to the establishment of TLS on top of the TCP connection in order to provide integrity protection and confidentiality.

```

m=application 50000 TCP/TLS/BFCP *
a=setup:passive
a=connection:new
a=fingerprint:SHA-1 \
    4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
a=floorctrl:s-only
a=confid:4321
a=userid:1234
a=floorid:1 m-stream:10
a=floorid:2 m-stream:11
m=audio 50002 RTP/AVP 0
a=label:10
m=video 50004 RTP/AVP 31
a=label:11

```

Figure 23.11: Floor control server's SDP session description

The “floorctrl” attribute indicates that the entity generating this session description can only act as a floor control server, and not as a client. Although in a typical centralized conference it is always quite clear which entity is the floor control server and which one is the client, negotiating which entity acts as floor control server can be useful in other scenarios. For example, two endpoint establishing a floor-controlled shared whiteboarding session between them need to decide which endpoint acts as the floor control server.

The “confid” and the “userid” attributes provide the client with the conference ID and the client’s BFCP user ID respectively. The client will use these values in the BFCP messages it sends to the floor control server.

The “floorid” attributes associate floors with media stream. In this case, the client will need to request floor “1” in order to use the audio stream (whose label is “10”) and floor “2” in order to use the video stream (whose label is “11”).

23.4.1.2 Outside an Offer/Answer Exchange

A BFCP client can also establish a connection with a floor control server without using an offer/answer exchange (how to do it is specified in RFC 5018 [100]). In order to establish the connection, the client needs to obtain the same data as when an offer/answer exchange is used (i.e., the server’s IP address and port number, the conference and user ID, floor IDs and their relationship with resources such as media streams, etc.). Instead of getting all of these data in a session description from the floor control server, the client typically obtains all of the data it needs using the conference event package. The XCON data model describes how to encode all of these data in an XML document.

Once the client obtains, via the conference event package, all of the data it needs, it establishes a TCP connection to the floor control server. The client uses the conference ID and the user ID obtained through the conference event package in its BFCP messages.

23.4.2 BFCP Message Flow

Figure 23.12 shows a typical BFCP message flow with three entities: a BFCP client, the floor control server, and a floor chair.

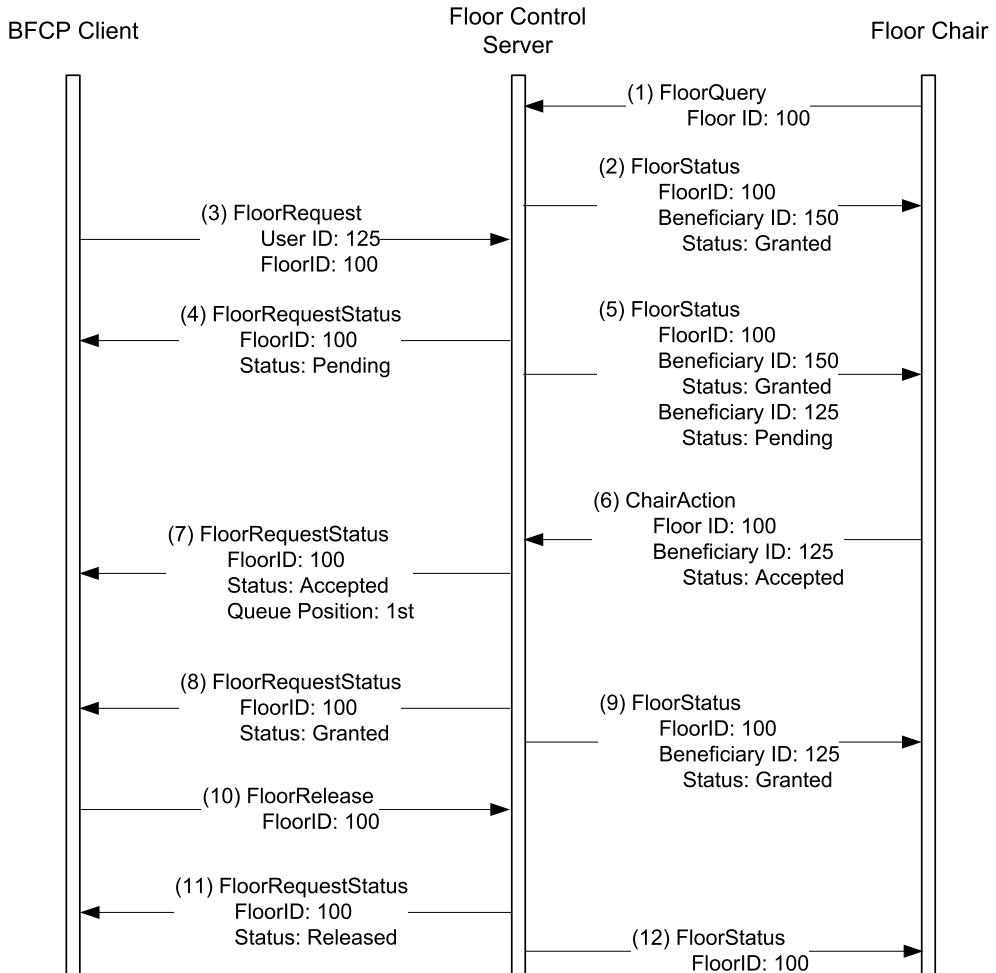


Figure 23.12: BFCP message flow

The floor chair in Figure 23.12 is responsible for the floor whose ID is 100. The floor chair joins the ongoing conference by requesting information about floor 100 by sending a *FloorQuery* message to the floor control server. The floor control server informs the floor chair that floor 100 is currently granted to the user whose ID is 150 (another floor chair was handling floor 100 before the floor chair in the figure joined the conference).

User 125 requests floor 100 by sending a *FloorRequest* message to the floor control server. The floor control server informs the BFCP client about the status of its floor request using *FloorRequestStatus* messages.

When the floor chair is notified about the new floor request by user 125, the floor chair accepts the floor request by sending a *ChairAction* message to the floor control server. In our example, the floor control server is configured to place accepted floor requests in a queue. In this case, the floor request is placed in the first position of the queue. At a later point, when

the current holder of the floor (i.e., user 150) releases the floor, the floor control server grants the floor request in the first position of the queue. That is, the floor request by user 125.

Note that, in this conference, the floor chair simply accepts or rejects floor requests. The floor control server takes care of performing queue management and granting floor requests when they reach the top of the queue. BFCP also allows floor chairs to be more in control of how floor requests are handled. In a different conference, the floor control server may be configured to let floor chairs perform queue management and decide when to grant particular floor requests.

When user 125 is finished using the resource associated to floor 100, the user releases the floor by sending a `FloorRelease` message. The floor control server informs the floor chair that there are no floor requests for floor 100 with a `FloorStatus` message.

23.4.3 BFCP Primitives

BFCP defines a set of primitives or messages. Table 23.1 lists all of the messages defined so far. The table also shows the direction in which each message can be sent. “P” denotes floor participant, “S” denotes floor control server, and “Ch” denotes floor chair.

Table 23.1: BFCP Primitives

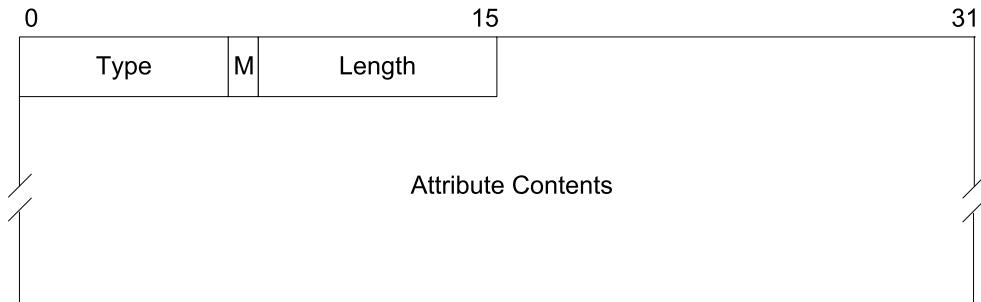
Primitive	Direction	
FloorRequest	P → S	
FloorRelease	P → S	
FloorRequestQuery	P → S	Ch → S
FloorRequestStatus	P ← S	Ch ← S
UserQuery	P → S	Ch → S
UserStatus	P ← S	Ch ← S
FloorQuery	P → S	Ch → S
FloorStatus	P ← S	Ch ← S
ChairAction		Ch → S
ChairActionAck		Ch ← S
Hello	P → S	Ch → S
HelloAck	P ← S	Ch ← S
Error	P ← S	Ch ← S

23.4.4 BFCP Encoding

BFCP messages use a TLV (Type-Length-Value)-based binary encoding. A BFCP message consists of a common header followed by attributes. Figure 23.13 shows the format of the common header. The payload length field contains the length of all attributes following the common header.

Figure 23.14 shows the format of BFCP attributes. Following a TLV-based format, every attribute carries its type (i.e., which attribute it is), its length, and the actual contents of the attribute. In addition, attributes carry an “M” bit, which indicates whether or not it is

0	15	31
Ver	Reserved	Primitive
Payload Length		
Conference ID		
Transaction ID		User ID

Figure 23.13: BFCP common header format**Figure 23.14:** BFCP attribute format

0	15	31
0 0 0 0 0 1 0	M	0 0 0 0 0 1 0 0
Floor ID		

Figure 23.15: BFCP Floor ID attribute

mandatory for the receiver of the BFCP message to understand this attribute. The “M” bit is useful for extension attributes where even if the receiver does not understand the extension attribute, it can still process the message.

Figure 23.15 shows an example of a BFCP attribute: the Floor ID attribute. Its attribute type is 0000010 and, since the value of Floor IDs is always a 16-bit number (i.e., 2 bytes), the attribute’s length is always 4 bytes (binary 100).

Chapter 24

Conferencing in the IMS

In Chapter 23, we introduced the basic technologies and architectures developed by the IETF in the conferencing area. In this chapter, we discuss how those technologies are used in the IMS to provide a conferencing service. This chapter is fairly brief because applying the technologies described in Chapter 23 to the IMS architecture is relatively straight-forward.

24.1 The IMS Conferencing Service

The IMS conferencing service (specified in 3GPP TS 24.147 [32]) is based on the SIPPING conferencing framework (specified in RFC 4353 [272]). Of the specifications produced within the XCON working group, the IMS conferencing service only uses BFCP (specified in RFC 4582 [106]). In the future, as the work in the XCON working group progresses, the IMS conferencing service may use the XCON framework and the conference control protocol developed by the XCON working group. However, at present, they are not yet used in the context of this IMS service.

The IMS conferencing services is based on the tightly-coupled conference model described in Figure 23.3. In the IMS, the conference server is distributed into two logical entities: one handling signaling and the other handling media, as shown in Figure 24.1. The former corresponds to a combination of an AS and an MRFC; the latter corresponds to an MRFP.

PSTN interworking is also part of the IMS conferencing service. Users on the PSTN can participate in IMS conferences through an MGCF, which acts as a conference participant and talks SIP to the AS/MRFC part of the conference server.

24.1.1 Creating and Joining a Conference

A client creates a conference at a server by sending an INVITE request (1) to the server's conference factory URI (defined in RFC 4579 [195]), as shown in Figure 24.2. The server responds with a 200 (OK) response (2) that carries the new conference URI in its `Contact` header field. Users joining the conference send INVITE requests (4) to this conference URI. Therefore, the conference factory URI is only used at conference creation in order to obtain a conference URI, which is allocated by the conference server. Once a conference is created, it is identified by its conference URI.

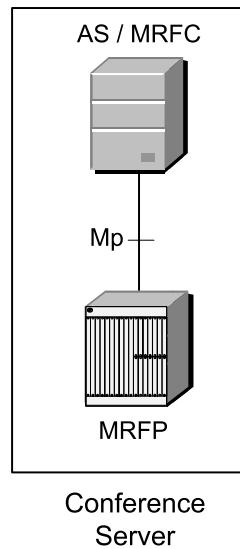


Figure 24.1: The IMS conference service architecture

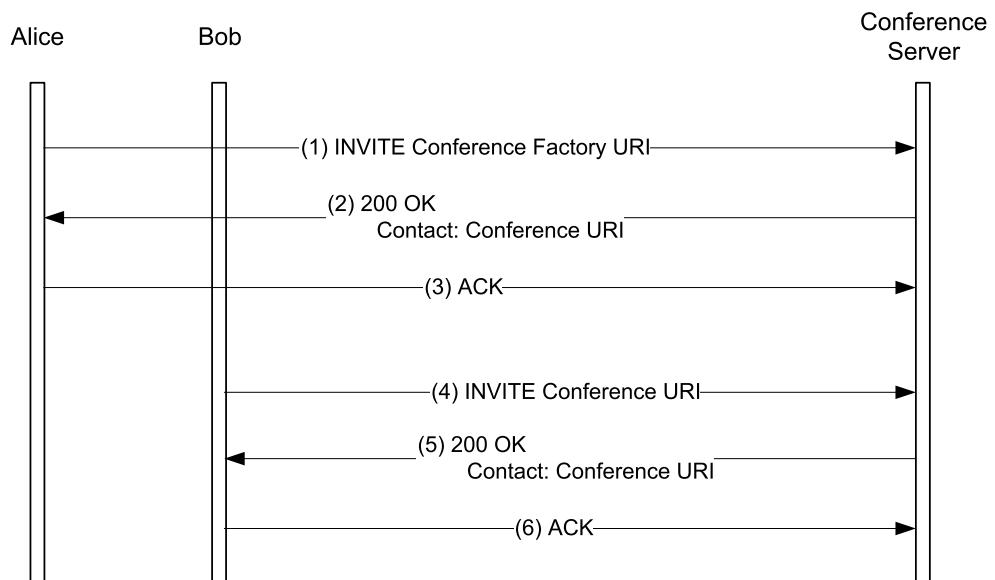


Figure 24.2: Conference creation using a conference factory URI

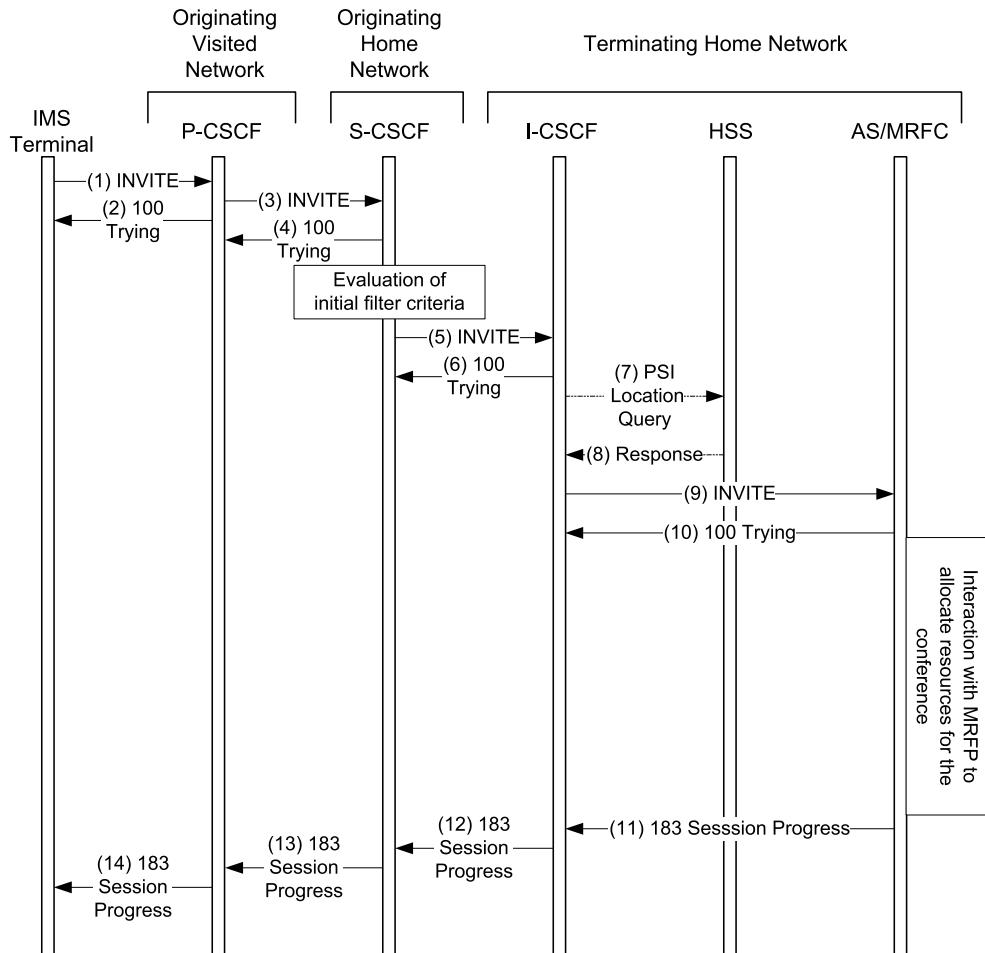


Figure 24.3: Joining a conference identified by a PSI

In Figure 24.2, Bob joins the conference by sending an INVITE request (4) to the conference URI. Bob can obtain such a URI in several ways. For example, Alice can send it to him in an email or tell him on the phone. Alternatively, Alice can send a REFER request to Bob so that Bob sends an INVITE to the conference URI. Alice can also send a REFER request to the conference server so that it sends an INVITE to Bob.

Clients can also request the creation of a conference with a particular conference URI. To do this, a client sends its initial INVITE request to the conference URI it wants to be allocated for the conference (the host part of the URI needs to route to the conference server). On receiving such an INVITE request, the conference server checks if its policy allows it to allocate such a conference URI (e.g., the URI may not be available because it is in use by another conference) and, if it does, the conference server returns the same conference URI in the contact header field of a 200 (OK) response.

Conference URIs are PSIs (see Section 3.5.4) and are routed as such. Figure 24.3 shows an IMS terminal sending an INVITE request to a conference URI that is hosted at a different

domain. The I-CSCF of the terminating domain consults the HSS (7) in order to resolve the PSI (i.e., the conference URI). The HSS provides the I-CSCF (8) with the address of the AS/MRFC corresponding to the PSI and the I-CSCF relays the INVITE request to that AS/MRFC (9). Before returning a 183 (Session Progress) response (11), the AS/MRFC allocates resources at an MRFP.

24.1.2 Other Actions

Conference participants can use the SIP conference event package (specified in RFC 4575 [289]) to obtain information about the conference, as described in Section 23.2.1. Conferences implementing floor control use BFCP (specified in RFC 4582 [106]).

24.2 Relation with the Work in TISPAN and OMA

The IMS conference service described in this chapter has been specified by 3GPP. In addition, there is also work by both TISPAN and the OMA in this area.

As will be discussed in Chapter 26, TISPAN has specified the TISPAN simulation services, which later contributed to 3GPP under the name of Multimedia Telephony Services. These services are telephony services similar to the PSTN/ISDN supplementary services but implemented using the IMS infrastructure and its protocols (e.g., SIP). One of these simulation services is the CONF service, which is discussed in Section 26.4. The CONF service is specified in ETSI TS 183 005 [135] and 3GPP TS 24.605 [66], which are largely based on the IMS service described in this chapter (which is mainly based on 3GPP TS 24.147 [32]).

The OMA has also specified services in the conferencing area. The OMA PoC (Push-to-talk over Cellular) service, which we discuss in Chapter 25, is a conferencing service that uses a particular floor control policy. OMA SIMPLE IM [242] and CPM (Converged IP Messaging) [240] mostly focus on messaging-based services.

Chapter 25

Push-to-talk over Cellular

Push-to-talk over Cellular (PoC) was the first IMS-based service deployed by several mobile operators because it does not require the deployment of new radio technologies. PoC can run on top of low-bandwidth and high-delay links, which are inappropriate for running other types of services, such as voice calls.

PoC is a walkie-talkie type of service. Users press (and hold) a button when they want to say something, but they do not start speaking until their terminal tells them to do so (usually by beeping). At this point, users say whatever they want to say and signal the end of their speech by releasing the button.

Unlike regular voice calls, which are full-duplex, PoC is a half-duplex service; that is, only one user can speak at a time.

PoC sessions can have more than two participants. At a given time, one user speaks and the rest listen (as in the two-party case). A simple way of understanding a multiparty PoC is a group of friends going to the movies. One at a time, they take turns to tell the rest which movie they want to watch, at which point they can make the final choice (usually after some extra rounds of discussions).

Even though Push-to-talk was originally designed to be a voice-only service, it currently supports different media types such as streamed video and instant messages.

25.1 PoC Standardization

When the definition of the PoC service started there were several incompatible PoC specifications. Many were not based on the IMS, but consisted of proprietary solutions implemented by a single vendor. As a result these PoC solutions generally could not interoperate with equipment from other vendors.

Many operators willing to provide PoC services felt uncomfortable with the situation just described and asked a few vendors for a standard solution based on the IMS. As a consequence, a group of vendors teamed up to develop an open PoC industry standard. These vendors were Ericsson, Motorola, Nokia, and Siemens. The result of this collaboration was a set of publicly available PoC specifications.

It was clear that a widely-accepted PoC standard which took into account the requirements of most of the industry was needed. The industry standard was a good starting point, but there was still a long way to go before having a fully-featured PoC service. This situation prompted the OMA (Open Mobile Alliance) to create the PoC working group to start working

on the OMA PoC service. (For a description of OMA, its structure, and the different types of recommendations it produces, see Section 2.6.)

OMA decided to base its PoC service on the IMS. So, the consortium that developed the PoC industry standard provided OMA with their PoC specifications, which were also based on the IMS. These specifications were taken as the starting point for the OMA PoC standard.

At the same time the IETF started working on some building blocks that were missing in SIP and in the conferencing architecture to be able to provide a fully-featured PoC service. These building blocks were needed by the OMA for its PoC service.

Section 25.2 covers the building blocks developed by the IETF that are relevant to PoC. Section 25.3 describes the PoC service as standardized by the OMA.

As you have probably noticed, we have not introduced a chapter called “PoC on the Internet” as we have done with other services covered in this book. Instead, we describe the relevant IETF specifications in Section 25.2. We have chosen to do so because the IETF has not defined a PoC framework. The IETF has a conferencing framework and, from the IETF perspective, PoC is just a conference. A conference that uses a set of extensions (e.g., conference establishment using request-contained lists) and a particular floor control policy, but a conference nevertheless.

25.2 IETF Work Relevant to PoC

Given that a PoC session is, at the end of the day, a conference, all of the work developed in the IETF on conferencing is very relevant to PoC. As described in Chapter 23, there are two main IETF Working Groups (WGs) involved in this conferencing work: SIPPING and XCON.

The SIPPING WG has developed a set of extensions to establish conferences using SIP. However, conferencing-related issues that do not have to do with SIP (e.g., floor control) are outside the scope of the SIPPING WG. These issues are typically handled by the XCON WG, which focuses on centralized conferences.

Chapter 23 discusses the work on general conferencing developed by these two WGs. This work includes BFCP (specified RFC 4582 [106]), which is a floor control protocol specifically designed so that its messages are small enough to be used with BFCP in low-bandwidth radio environments such as those in which PoC is expected to work.

In addition to the work on general conferencing just discussed, the IETF has developed a set of extensions to SIP that are referred to as URI-list services (see Section 25.2.1). The IETF developed these extensions after noticing that some of the OMA PoC requirements related to multiparty sessions could not be met by existing IETF technology.

The IETF has also developed two SIP extensions that only apply to the OMA PoC service: an event package to discover the settings of a PoC terminal (specified in RFC 4354 [148]) and a set of SIP header fields (specified in RFC 4964 [73] and the Internet-Draft “Requesting Answering Modes for SIP” [315]).

25.2.1 *URI-list Services*

Some services involve multiple very similar transactions. For example, a user may need to send a page-mode instant message to a number of friends telling them at what time they should meet in the movie theater. The user would send one message to each friend; however, all of the messages would have the same contents (e.g., “Let’s meet at seven.”). If the user of our example sits on a low-bandwidth access, sending all of those messages can take a while.

URI-list services were designed for this type of situation (their framework is specified in the Internet-Draft “Requirements and Framework for Session Initiation Protocol (SIP) Uniform Resource Identifier (URI)-List Services” [107]).

Servers providing a URI-list service perform a similar transaction to all of the members (identified by URIs) of a list provided by the user agent invoking the service. In Section 21.6.1, we introduced a URI-list service for MESSAGE requests. URI-list services can be applied to other requests as well. The idea is always the same. The URI-list service receives a request with a URI-list and sends similar requests to the URIs on the list.

In addition to the URI-list service for MESSAGE (specified in the Internet-Draft “Multiple-Recipient MESSAGE Requests in SIP” [153]), there are URI-list services defined for methods such as INVITE (specified in the Internet-Draft “Conference Establishment Using Request-Contained Lists in SIP” [102]) and SUBSCRIBE (specified in the Internet-Draft “Subscriptions to Request-Contained Resource Lists in SIP” [108]). The INVITE URI-list service can be used to establish a conference with multiple participants and the SUBSCRIBE URI-list service can be used to subscribe to the presence information of several users.

25.2.1.1 Multiple REFER

An extension that may seem like a URI-service but is not is the multiple-REFER extension (specified in the Internet-Draft “Referring to Multiple Resources in SIP” [105]). The difference between this extension and the URI-list services is that, while a URI-list service replicates the same transaction towards a set of users, the recipient of a multiple REFER executes the transaction identified by the Refer-To header field (which does not need to be a REFER transaction). For example, a user may send a REFER to a conference focus so that the focus INVITES a number of new users into the conference, as shown in Figure 25.1.

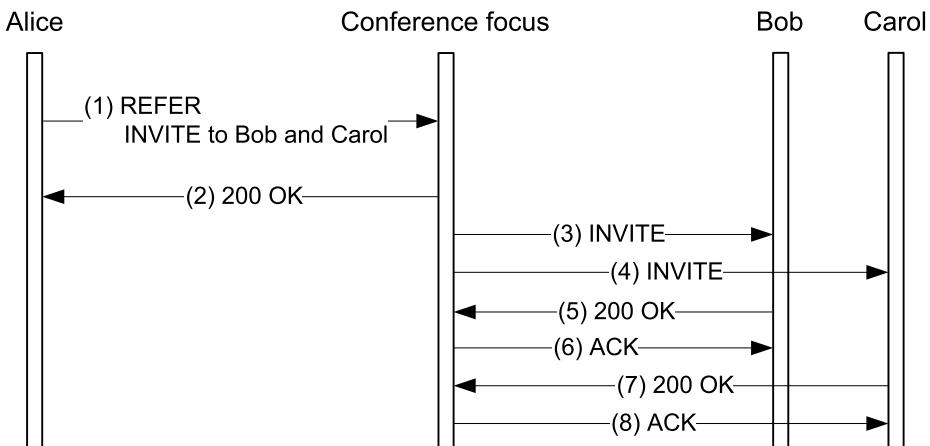


Figure 25.1: Multiple REFER

Since multiple REFERS are typically used within the context of an application, the user agent generating it generally has an application-specific means to discover the result of the transactions initiated by the server (in our example, the INVITE transactions initiated by the conference focus). In the conferencing example, the user may use the conference

event package to discover which users were brought into the conference successfully. Consequently, multiple REFERs are normally combined with an extension (specified in RFC 4488 [207]) that eliminates the implicit subscription that is usually linked to REFER. Once again, the subscription to the results of the transactions initiated by the server is eliminated by that extension. That is why Figure 25.1 does not show any NOTIFY requests from the conference focus to Alice (see Section 4.18 for a discussion of the REFER method and its implicit subscription).

Both URI-list services and multiple-REFER are used by PoC. URI-list services are used to establish multiparty PoC sessions (INVITE URI-list service) and to send multiple-recipient page-mode instant messages (MESSAGE URI-list service). Multiple-REFER is used to invite multiple users to a PoC session.

25.2.1.2 URI-list Format

A user agent using a URI-list service or generating a multiple REFER needs to include a list of URIs in its request. The default format for these lists is supposed to be service-specific, but effectively, all of the URI-list services defined so far and multiple-REFER use the same URI-list format. This format is based on XML and is a simplified version (e.g., hierarchical lists are not allowed) of the general format for representing resource lists (specified in RFC 4826 [273]).

Figure 25.2 shows an example of an INVITE request that carries two body parts: an SDP session description and a URI list in the XML-based format just described.

URIs in a URI lists can also carry copyControl attributes (specified in the Internet Draft “XML Format Extension for Representing Copy Control Attributes in Resource Lists” [152]). A copyControl describes why a message was delivered to a particular URI. It performs the same function as the To: and Cc: header fields in email.

25.2.1.3 Consent-based Communications

As we have already stated, URI-list services allow user agents using low-bandwidth accesses to request the generation of a potentially large number of transactions towards a set of URIs. While this type of service is a great tool for implementing services such as PoC, servers providing URI-list services could be used as traffic amplifiers to launch DoS attacks.

An attacker would just need to generate a single request with a URI list containing the URIs of the victims. The attacker would send this request to a URI-list service. The URI-list service would then flood the members of the list with undesired traffic.

This type of attack is similar to a form of email SPAM, where the attacker places the email address of the victim on a distribution list. The victim keeps receiving the messages sent to the distribution list but has no means to unsubscribe from the list.

In order to avoid this type of attack in SIP, URI-list services need to obtain permission from the recipients before sending them any traffic. Effectively, they implement a form of consent-based communication (as specified in the Internet-Draft “A Framework for Consent-Based Communications in SIP” [279]) where entities need to agree to communicate before the actual communication takes place.

OMA has not yet studied how to apply this consent framework to PoC. Therefore, at this point, PoC does not use this framework.

```

INVITE sip:conf-fact@example.com SIP/2.0
Via: SIP/2.0/TCP client.chicago.example.com
;branch=z9hG4bKhjhs8ass83
Max-Forwards: 70
To: Conf Factory <sip:conf-fact@example.com>
From: Carol <sip:carol@chicago.example.com>;tag=32331
Call-ID: d432fa84b4c76e66710
Cseq: 1 INVITE
Contact: <sip:carol@client.chicago.example.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER,
      SUBSCRIBE, NOTIFY
Allow-Events: dialog
Accept: application/sdp, message/sipfrag
Require: recipient-list-invite
Content-Type: multipart/mixed;boundary="boundary1"
Content-Length: 690

--boundary1
Content-Type: application/sdp
v=0
o=carol 2890844526 2890842807 IN IP4 chicago.example.com
s=-
c=IN IP4 192.0.2.1
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 20002 RTP/AVP 31
a=rtpmap:31 H261/90000

--boundary1
Content-Type: application/resource-lists+xml
Content-Disposition: recipient-list

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <list>
    <entry uri="sip:bill@example.com" />
    <entry uri="sip:joe@example.org" />
    <entry uri="sip:ted@example.net" />
  </list>
</resource-lists>
--boundary1--

```

Figure 25.2: INVITE with two body parts

25.2.2 Event Package for PoC Settings

In the PoC service, situations in which the network needs to be informed about the settings of a PoC terminal occur. For example, if a PoC terminal is in *don't disturb* mode (i.e., the terminal will reject all incoming session invitations) the network can reject directly any session invitation for the terminal. Sending the invitation to the terminal, only to have it rejected, would consume radio resources unnecessarily.

A terminal keeps its home PoC server updated on the terminal's settings by sending PUBLISH requests (whose bodies follow the format specified in RFC 4354 [148]).

25.2.3 SIP Header Fields

The PoC server defines the concept of answer mode, which can be set to automatic or manual. A terminal in automatic answer mode accepts session invitations automatically, without any user intervention.

The Answer-Mode header field (specified in the Internet-Draft “Requesting Answering Modes for SIP” [315]) carries information related to the answer mode. The Answer-Mode header field can be inserted into an INVITE request to request a particular answer mode from the callee. In addition, the callee can insert this header field in a response to indicate which answer mode was actually applied.

The P-Answer-State header field (specified in RFC 4964 [73]) can be included in a response to an INVITE to indicate which entity (the user agent server or an intermediary) generated the response. The use of both header fields is further described in the following sections.

25.3 Architecture

In this section we look at the architecture of the PoC service (as specified in the Candidate Enabler Release Package for PoC Version 2.0 [243]). Figure 25.3 indicates the nodes involved in PoC and the interfaces between them.

The User Equipment contains six logical elements: the DM (Device Management) client, the presence source, the watcher, the PoC client, the UE (User Equipment) PoC Box, and the XDMC (XML Document Management Client). The DM client uses the OMA Device Management Protocol (as specified in the OMA Device Management Enabler [241]) to communicate with the DM server over the DM-1 interface.

The presence source and the watcher use SIP (as specified in the OMA Presence Simple Enabler [242]) to communicate with the SIP/IP core over the PRS-1 and PRS-2 interfaces, respectively. The PoC client uses SIP to communicate with the SIP/IP Core over the POC-1 interface, and RTP, MSRP, and MBCP (Media Burst Control Protocol) to communicate with the PoC server over the POC-3 interface. MBCP is a floor control protocol based on RTCP that is used to signal which user is allowed to send media at a given time.

The UE PoC Box uses SIP to communicate with the SIP/IP Core over the POC-9 interface, and RTP, MSRP, and MBCP to communicate with the PoC server over the POC-10 interface. The UE PoC Box (and the NW PoC Box) can store data that can be retrieved by the user at a later point.

The XDMC (as specified in the OMA XML Document Management Candidate Enabler [244]) uses SIP to communicate with the SIP/IP Core over the XDM-1 interface and XCAP to communicate with the Aggregation Proxy over the XDM-3 interface. The XDM-3

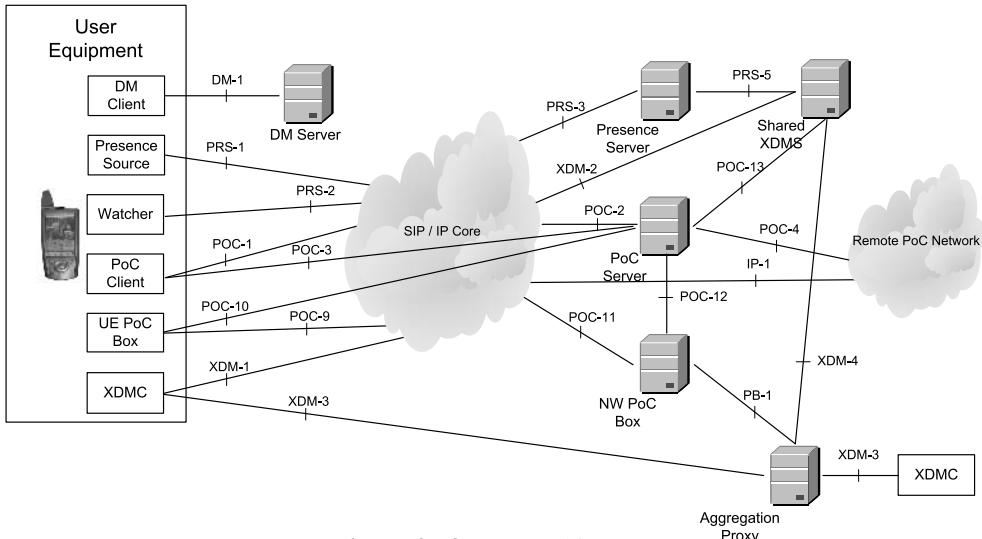


Figure 25.3: PoC architecture

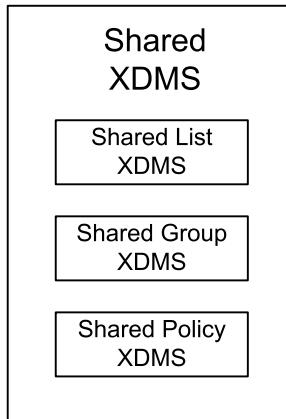


Figure 25.4: Structure of the Shared XDMS

interface is used to perform document management (e.g., set up a URI list with the user's golf buddies) and the XDM-2 interface is used to subscribe to changes in documents that are stored in the network.

The Aggregation Proxy acts as a single point for the XDMC to contact the network. The Aggregation Proxy performs user authentication and routes the XCAP messages from the XDMC (received over the XDM-3 interface) to the appropriate XDMS. The Aggregation proxy uses XCAP to communicate with the NW PoC Box over the PB-1 interface and with the Shared XDMS over the XDM-4 interface.

The NW PoC Box uses SIP to communicate with the SIP/IP Core over the POC-11 interface, and RTP, MSRP, and MBCP to communicate with the PoC server over the POC-12 interface.

The Shared XDMS uses XCAP to communicate with the Aggregation Proxy over the XDM-4 interface and with the PoC Server over the POC-13 interface. Since the documents managed by the Shared XDMS may be shared with other services, the Shared XDMS has additional interfaces towards those services. For example, the interface between the Shared XDMS and the Presence Server is referred to as PRS-5 and is based on XCAP.

The PoC Server uses SIP to communicate with the SIP/IP Core over the POC-2 interface. In addition, it uses RTP and MBCP to communicate with the PoC Client over the POC-3 interface and with other PoC networks over the POC-4 interface. Furthermore, the PoC server uses XCAP to communicate with the Shared XDMS over the POC-13 interface.

The SIP/IP Core can be realized in different ways. Nevertheless, we expect that it will usually be realized by using the IMS. Consequently, the SIP/IP Core cloud would correspond to the IMS (as described in 3GPP TR 23.979 [6]).

Table 25.1 shows the protocols used in the different interfaces.

Table 25.1: PoC interfaces

Interface	Protocol
POC-1	SIP
POC-2	SIP
POC-3	RTP / MSRP / MBCP
POC-4	RTP / MSRP / MBCP
POC-9	SIP
POC-10	RTP / MSRP / MBCP
POC-11	SIP
POC-13	XCAP
XDM-1	SIP
XDM-2	SIP
XDM-3	XCAP
XDM-4	XCAP
PRS-1	SIP
PRS-2	SIP
PRS-5	XCAP
IP-1	SIP
DM-1	OMA Device Management Protocol
PB-1	XCAP

25.4 Registration

In order to use the PoC service, a terminal needs to register with the PoC service. When the terminal performs IMS registration, it adds the `+g.poc.talkburst` and `+g.poc.groupad` feature tags to the `Contact` header field of the `REGISTER` request. On receiving these feature tags, the S-CSCF can perform a third-party registration towards the PoC server of the domain (i.e., the user's home PoC server).

The `+g.poc.talkburst` feature tag indicates that the terminal can handle PoC sessions. The `+g.poc.groupad` feature tag indicates that the terminal can handle group advertisement (see Section 25.8).

25.5 PoC Server Roles

A PoC server within a session can perform two roles: Controlling PoC Function or Participating PoC Function. A given PoC server in a given PoC session will be performing one or both roles. However, only one PoC server in a session performs the Controlling PoC Function. This server may or may not perform the Participating PoC Function as well. The rest of the PoC servers, assuming that there are several PoC servers involved in the session, will only perform the Participating PoC Function.

The PoC server performing the Controlling PoC Function is usually referred to as the controlling PoC server. Similarly, the PoC servers performing the Participating PoC Function are usually referred to as participating PoC servers.

Figure 25.5 shows a PoC session with one controlling and four participating PoC servers. Each of the PoC servers is in a different domain. In order to simplify this and other figures in this chapter, we do not show all of the elements involved in the session. That is, we do not show the IMS nodes between the different PoC entities.

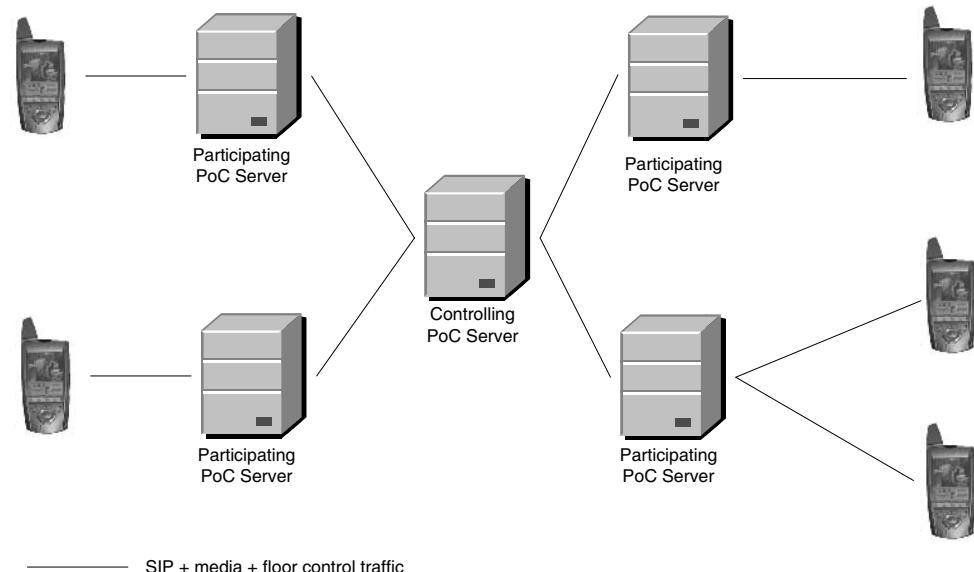


Figure 25.5: PoC session with a central controlling PoC server

The controlling PoC server provides centralized PoC session handling. This includes media distribution, centralized floor control, and policy enforcement for participation in group sessions. The participating PoC server exchanges SIP signaling with the client and with the controlling PoC server and, optionally, relays media and floor control messages between them. When a participating PoC server chooses not to be on the media path, clients exchange media and floor control traffic directly with the controlling PoC server.

Figure 25.6 shows another PoC session where the controlling PoC server is co-located with a participating PoC server. That is, the same PoC server performs both roles at the same time.

The process of determining which of the PoC servers involved in a session acts as the controlling PoC server depends on the type of the session. Section 25.6 describes the different

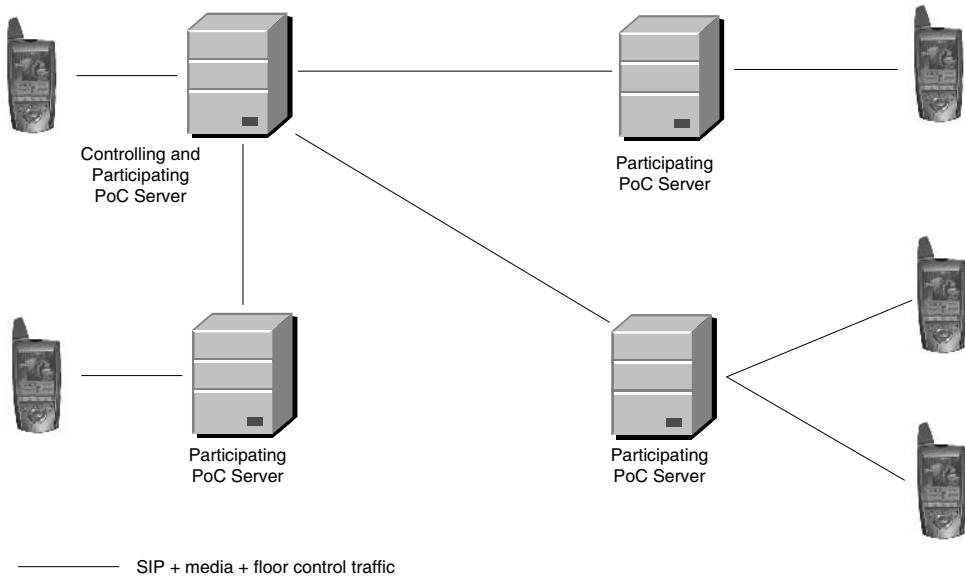


Figure 25.6: Controlling and participating PoC server

PoC session types and discusses the procedures to determine the controlling PoC server in each.

25.6 PoC Session Types

PoC defines the following session types or communication modes.

One-to-one. A PoC session between two users.

Ad-hoc PoC Group. A user selects a set of users in an *ad-hoc* fashion (e.g., picking them from the terminal's address book) and invites all of them into a multiparty PoC session.

Pre-arranged PoC Group. Similar to the *ad-hoc* PoC group, the pre-arranged PoC group also consists of a multiparty PoC session. Nevertheless, the users participating in the session are selected beforehand, not in an *ad-hoc* manner when it is established. That is, a pre-arranged PoC group includes a predefined set of users (e.g., the user's golf buddies).

Chat PoC Group. Chat PoC groups are also multiparty PoC sessions. However, when a user joins a chat PoC group, no invitations are sent to other users. Conversely, when a user joins a pre-arranged PoC group, all of the users that belong to that PoC group are invited to the PoC session.

These PoC session types are classified into two forms of PoC sessions: one-to-one and one-to-many. One-to-many PoC sessions include *ad-hoc*, pre-arranged, and chat PoC groups.

Now, let us look at the signaling involved in the establishment of the different session types. In addition, we discuss which PoC server is selected as the controlling PoC server

in each session type. However, before looking at these issues, we would like to provide an important clarification.

PoC defines two session establishment types: using *on-demand* signaling and using a *pre-established session*. Both session establishment types are discussed in Section 25.9. In the following message flows, we discuss session establishment procedures using only *on-demand* signaling. The use of a pre-established session is an optimization, which is described in Section 25.9.

25.6.1 One-to-one PoC Sessions

In one-to-one PoC sessions, the controlling PoC server is the inviting user's PoC server. That is, this PoC server is, at the same time, the participating PoC server of the inviting user and the controlling PoC server for the session.

Figure 25.7 shows the message flow for one-to-one PoC session establishment. In this message flow, we have included the SIP/IP Core in order to illustrate how filter criteria are used to route requests to the PoC servers. In subsequent message flows, we do not show the SIP/IP Core for the sake of clarity.

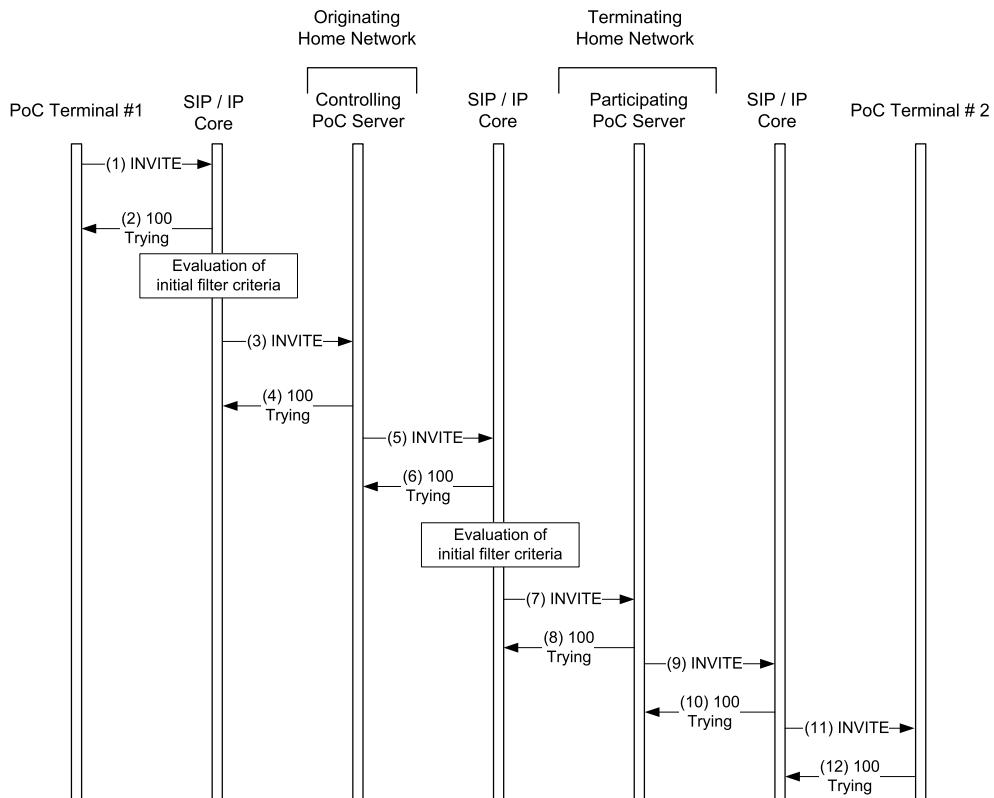


Figure 25.7: One-to-one PoC session establishment

The PoC terminal generates an INVITE (1) which is addressed to its home PoC server. The INVITE contains two body parts: an SDP session description and a URI list. The URI list contains the URI of the callee.

On receiving the INVITE (1), the originating user's S-CSCF, as usual, evaluates the initial filter criteria for the user and finally forwards the INVITE request on to the PoC server.

When the PoC server receives the INVITE request (3), it forwards it (5) towards the home domain of the callee.

The S-CSCF of the terminating user receives the INVITE (5) and evaluates the initial filter criteria for the terminating user. According to the filter criteria, an incoming INVITE request with the +g.poc.talkburst feature tag should be routed to the PoC server of the domain.

When the PoC server receives the INVITE (7), it generates a new INVITE (9) that will be routed by the SIP/IP Core towards the terminating user.

25.6.2 Ad-hoc PoC Group

In *ad-hoc* PoC group sessions, the controlling PoC server is the PoC server of the inviting user. That is, this PoC server is, at the same time, the inviting user's participating PoC server and the controlling PoC server for the session.

Figure 25.8 shows the message flow for an *ad-hoc* PoC session establishment. The message flow is the same as that for one-to-one PoC sessions. The only difference between both cases is that, in the one-to-one case, the URI list contains the address of a single callee whereas in the *ad-hoc* PoC group case the URI list contains the URIs of all of the callees.

On receiving INVITE (1), the controlling PoC server generates an INVITE towards each of the URIs in the URI list. This results in two INVITES: (3) and (4). These INVITES contain a single body part: an SDP session description. The participating PoC servers route these INVITES towards the terminating terminals.

25.6.3 Pre-arranged PoC Group

In pre-arranged PoC group sessions, the controlling PoC server is the PoC server hosting the pre-arranged PoC group. That is, the controlling PoC server is the PoC server of the domain that owns the URI that identifies the pre-arranged PoC group.

Figure 25.9 shows the message flow for pre-arranged PoC group session establishment. In this example, the inviting user's participating PoC server is not the controlling PoC server because the pre-arranged PoC group is hosted in another domain.

The INVITE (1) generated by the inviting terminal does not carry a URI list because the members of the pre-arranged PoC group have been previously set up in the network. The Request-URI of this INVITE (1) identifies the pre-arranged PoC group at the controlling PoC server.

The inviting user's PoC server behaves as a participating PoC server and, thus, relays the INVITE (3) to the controlling PoC server. On receiving this INVITE (3), the controlling PoC server invites all of the members of the pre-arranged PoC group.

Some pre-arranged PoC groups use a special media distribution policy whereby a user (called the distinguished participant) can talk to the whole group and listen to the answers from each individual user (called ordinary participants). However, the rest of the users (the ordinary participants) cannot talk or listen to each other. They only talk and listen to the

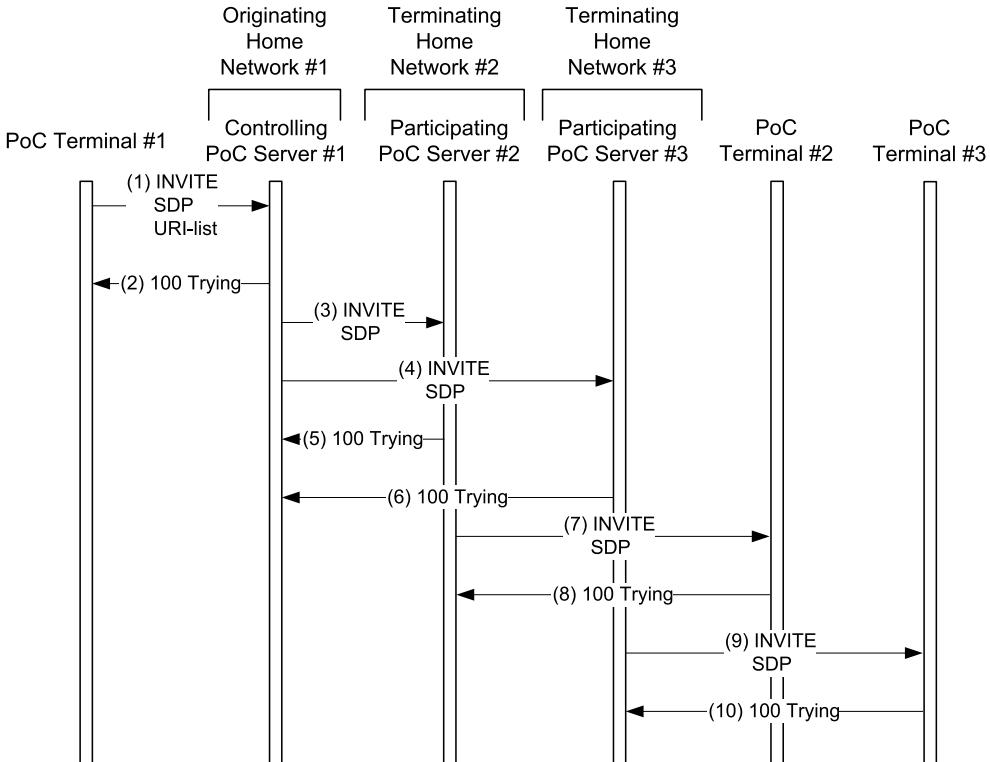


Figure 25.8: Ad-hoc PoC group session establishment

distinguished participant. When this form of media distribution is used in a pre-arranged PoC group, the session is referred to as a one-to-many-to-one PoC session.

There are scenarios where one-to-many-to-one PoC sessions are useful. For example, a taxi dispatcher needs to inform all of the drivers about customers waiting for a taxi, but the individual drivers answer only to the dispatcher. Any given driver does not hear the answers from the other drivers to the dispatcher. The terms of PoC Dispatcher and PoC Fleet Member are defined to cover the scenario just described.

The PoC Dispatcher of a pre-arranged PoC group can also send media to a subset of the PoC Fleet Members. In order to indicate which PoC Fleet Members should be receiving a given media burst, the PoC Dispatcher inserts a URI list in its INVITE request with the intended receivers.

25.6.4 Chat PoC Group

In Chat PoC group sessions, the controlling PoC server is the PoC server hosting the chat PoC group. That is, the controlling PoC server is the PoC server of the domain that owns the URI that identifies the chat PoC group. Chat PoC groups can be open or restricted. A restricted chat PoC group can only be joined only by users that are members of the chat PoC group.

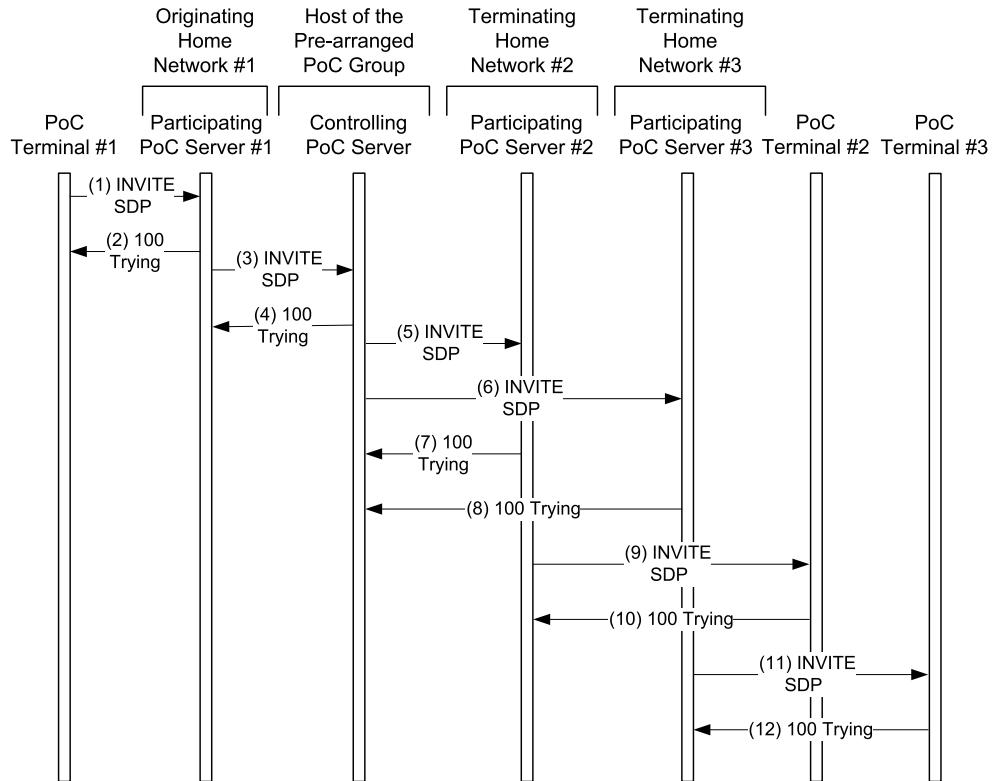


Figure 25.9: Pre-arranged PoC group session establishment

Figure 25.10 shows the message flow for chat PoC group session establishment. In this example, the participating PoC server of the inviting user is not the controlling PoC server because the chat PoC group is hosted in another domain.

The INVITE (1) generated by the inviting terminal does not carry a URI list because joining a chat room does not trigger any invitations to other users. The Request-URI of this INVITE (1) identifies the chat PoC group at the controlling PoC server.

The PoC server of the inviting user behaves as a participating PoC server and, thus, relays the INVITE (3) to the controlling PoC server. On receiving this INVITE (3), the controlling PoC server returns a 200 (OK) response (5) accepting the user into the chat PoC group session. Note that in this case, as opposed to what happens in *ad-hoc* and pre-arranged PoC group sessions, the controlling PoC server does not invite any users on receiving an INVITE request.

25.7 Adding Users to a PoC Session

New users can be added to an ongoing PoC session in two ways: the new user sends an INVITE request to the URI of the session or the controlling PoC server sends an INVITE request to the new user.

Participants in a PoC session can have the controlling PoC server send an INVITE request to the new user by sending a REFER request to the controlling PoC server. Figure 25.11

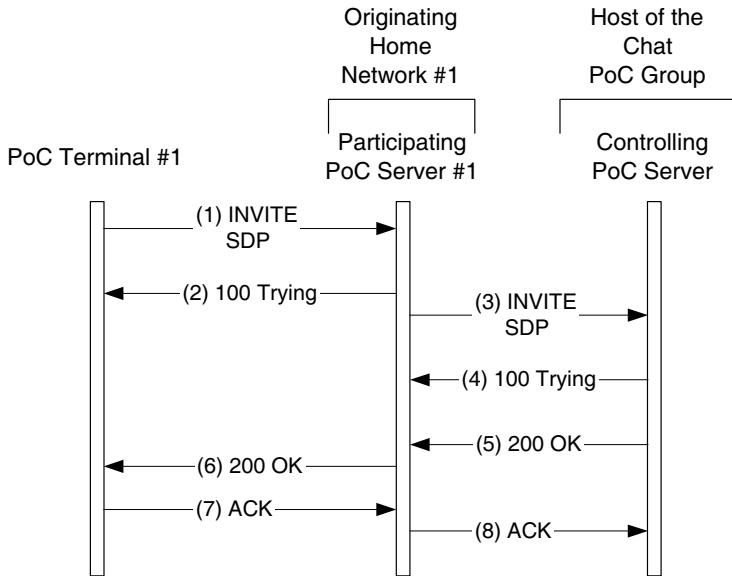


Figure 25.10: Chat PoC group session establishment

shows how a controlling PoC server receives a multiple REFER and, as a consequence, invites two new users to an ongoing PoC session.

Different session types have different authorization policies regarding which users can be added to an ongoing PoC session. In one-to-one and *ad-hoc* PoC sessions, new users can only be added to an ongoing session if they are invited by a participant. That is, a random user cannot send an INVITE request to the URI of an ongoing PoC session and have it accepted. The only users that the controlling PoC server of an ongoing one-to-one or *ad-hoc* session accepts an incoming INVITE request from are users that leave the session for some reason and later rejoin.

Participation in pre-arranged PoC group sessions is limited to members of the pre-arranged PoC group. That is, the controlling PoC server will not allow any other users to join the PoC session.

Participation in chat PoC group sessions may be limited to a set of users or open to everyone. Users can join a chat PoC group session by sending an INVITE to the URI of the chat PoC group, as described in Figure 25.10, or by accepting an invitation from the controlling PoC server.

25.8 Group Advertisements

In previous sections, we have seen that in order to join a pre-arranged and a chat PoC group session, a user needs to know its URI. Users can obtain URIs identifying PoC groups in many ways, such as in an email, in a phone conversation, from a piece of paper, or in a face-to-face meeting. In addition, PoC defines a means for PoC users to exchange these URIs: users can exchange URIs of PoC groups using *group advertisements*.

A group advertisement is a MESSAGE request that carries an XML body that contains the URI of the PoC group and, optionally, information about it (e.g., the topics its members

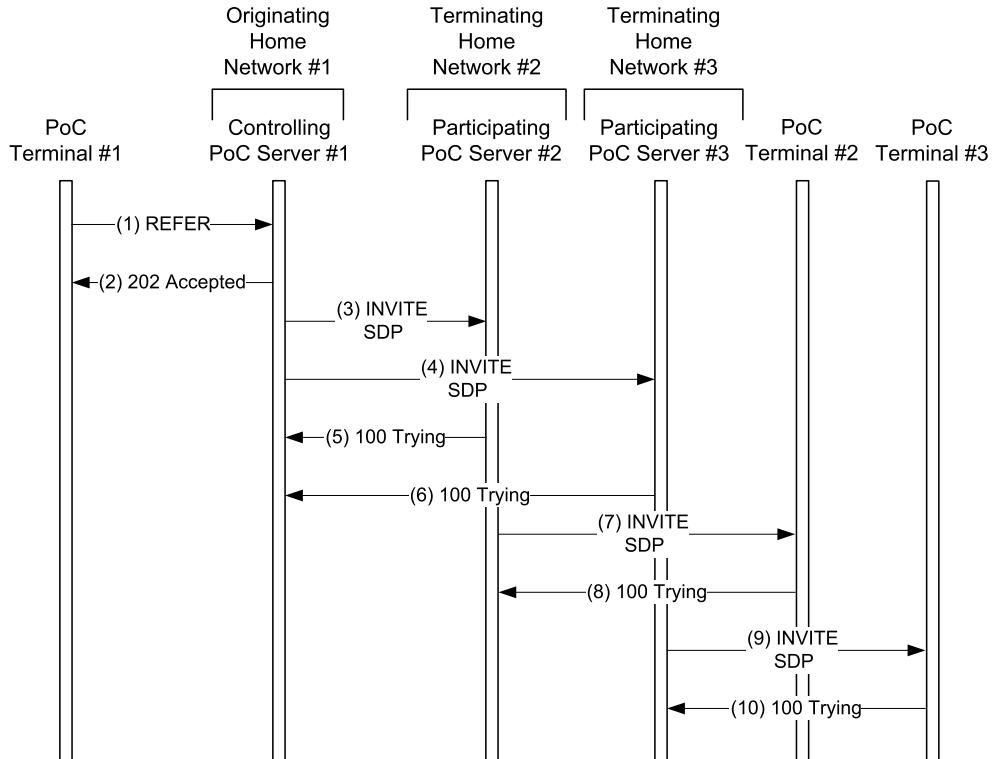


Figure 25.11: Bringing new users into a PoC session

usually talk about). In addition, the MESSAGE request carries a `+g.poc.groupad` feature tag in an Accept-Contact header field.

25.9 Session Establishment Types

All of the message flows we have discussed so far use so-called *on-demand* signaling. Nevertheless, PoC defines two session establishment types: using *on-demand* signaling and using a *pre-established session*.

When on-demand signaling is used, terminals generate INVITE requests to create new PoC sessions or join chat PoC groups. On the terminating side, the terminating user's PoC server relays incoming INVITE requests to the user. Figures 25.7–25.10 are examples of on-demand signaling.

The use of pre-established sessions results in an optimization that allows a more rapid session establishment at the terminating side. The terminal using the pre-established session establishes a session (using an INVITE request) with its home PoC server, typically right after registration. As in a regular session establishment, the terminal and its home PoC server negotiate all of the parameters needed to exchange media and floor control protocol messages. Nevertheless, they do not exchange media immediately.

At a later point, when the home PoC server receives an INVITE for the user, there is no need to use any SIP signaling towards the terminal. The session that was established

previously is used to deliver media and floor control messages to the terminal. The floor control protocol carries all of the information the terminal needs about the incoming INVITE received by the PoC server.

It is also possible to use a pre-established session at the originating side. However, the pre-established session does not eliminate the need for SIP signaling at the originating side. It only replaces the typical INVITE transaction used to establish a new session with a REFER transaction that instructs the PoC server to generate an INVITE.

Figure 25.12 shows a message flow where both the originating and the terminating sides use pre-established sessions. The terminals set up their pre-established sessions using INVITE transactions (1) and (4), respectively. At a later point, the originating terminal sends a REFER request (7) to its home PoC server. This REFER request (7) prompts the home PoC server to invite the terminating user to a one-to-one PoC session. The INVITE request (11) generated by the originating home PoC server arrives at the terminating PoC server, which generates a final response (12) without contacting the terminating terminal. The originating terminal is informed about the result of the invitation in a NOTIFY request (14).

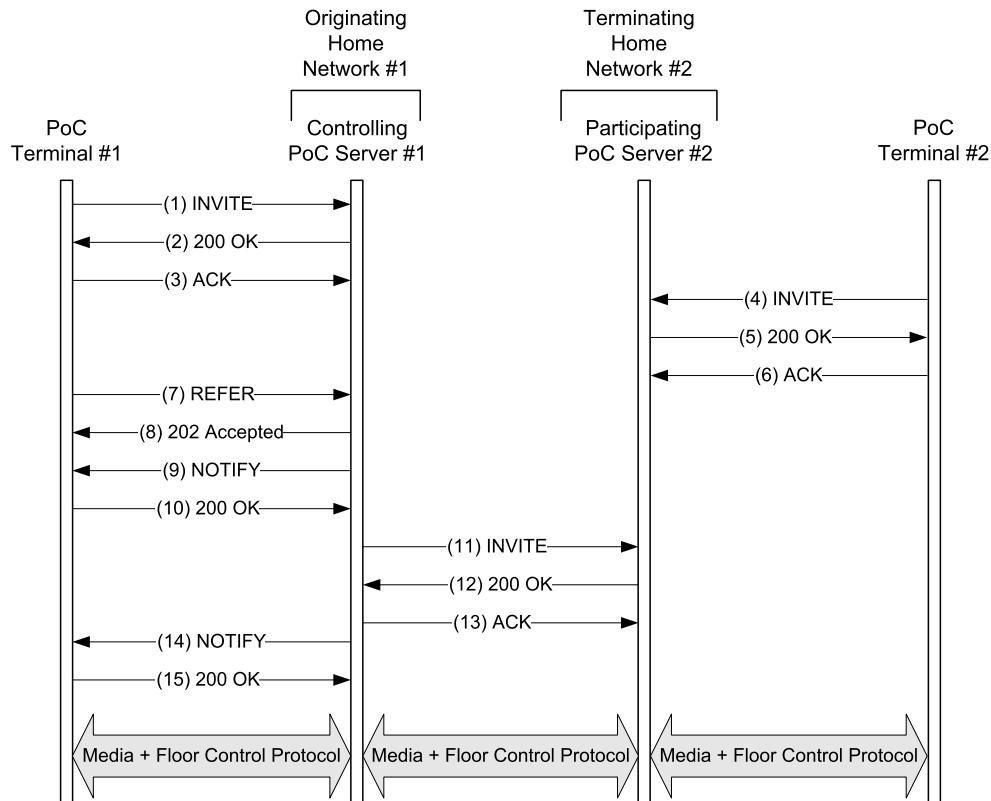


Figure 25.12: Pre-established session

Note that when REFER is used, the user agent server needs to generate a NOTIFY request immediately after accepting the REFER request (see Section 4.18). That is why the PoC server sends the first NOTIFY request (9).

It is possible to eliminate all of these NOTIFY requests by using an extension to REFER (specified in RFC 4488 [207]) that eliminates the implicit subscription that is usually linked to REFER. When terminals apply this extension to a REFER request, they use the floor control protocol to discover when the terminating user answers.

25.10 Answer Modes

PoC defines two answer modes: manual and automatic. The manual answer mode is the answer mode used in traditional telephones. When the terminal receives a PoC session invitation, the terminal alerts the user (usually by ringing). At that point, the user decides to accept or to reject the invitation.

Figure 25.13 shows the message flow for the manual answer mode. Note that PoC terminals do not use reliable provisional responses (the use of reliable provisional responses between PoC servers is optional). That is why there is no PRACK transaction after the 180 (Ringing) (2) response from the terminal.

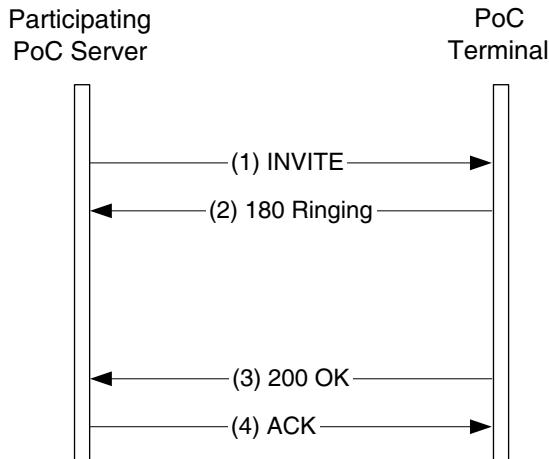


Figure 25.13: Manual answer mode

In the automatic answer mode, the user sets up the terminal to accept PoC sessions automatically. When the terminal receives a PoC session invitation, the terminal accepts it right away and starts playing the incoming media related to that session. This means that a terminal in automatic answer mode can start displaying incoming media (e.g., playing media through its speaker) at any point without any user intervention. This behavior is very similar to that of walkie-talkies.

Of course, a user can configure its terminal to use the automatic answer mode only for PoC session invitations coming from particular users.

Figure 25.14 shows the message flow for the automatic answer mode. The terminal responds directly with a 200 (OK) (2) final response.

PoC callers can request terminals at the callee side to apply a particular answer mode by using the `Answer-Mode` SIP header field (specified in the Internet-Draft “Requesting Answering Modes for SIP” [315]). In other cases, the caller may request a privileged answer mode, sometimes also called *manual answer override*, by setting the `Priv-Answer-Mode`

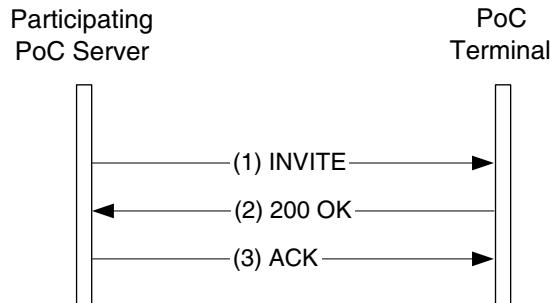


Figure 25.14: Automatic answer mode

SIP header field (also specified in the same Internet-Draft “Requesting Answering Modes for SIP” [315]) to an automatic or manual value. When an authorized PoC caller requests an automatic privileged answer mode, the callee’s terminal will answer automatically even if it is configured in manual answer mode. A situation where an automatic privileged answer mode may be useful is an emergency when the user has to be alerted about something urgently.

25.11 Right-to-send-media Indication Types

A controlling PoC server sending an INVITE request to a participating PoC server expects, following standard SIP procedures, to receive a response. If this response contains a session description, the controlling PoC server can start sending media using the media parameters just received.

In a regular SIP session, such a response is generated by the user agent server, and usually consists of a 183 (Session Progress) or a 200 (OK) response. Nevertheless, in PoC, the entity generating such a response is not always the terminal (i.e., the user agent server). The participating PoC server can act as a B2BUA and answer on behalf of the terminal.

The situation where a controlling PoC server receives a response that was originally generated by the terminating terminal is referred to as a *confirmed* answer state. The situation where a controlling PoC server receives a response that was generated by the participating PoC server without contacting the terminating terminal is referred to as an *unconfirmed* answer state.

Figure 25.15 shows a message flow with a confirmed answer state. When the participating PoC server receives a 200 (OK) response (5) from the terminal, it relays it to the controlling PoC server.

Figure 25.16 shows a message flow with an unconfirmed answer state. The participating PoC server generates a 183 (Session Progress) response (3) before even contacting the terminal. This response carries a P-Answer-State header field (which is specified in RFC 4964 [73]) with the value unconfirmed.

The reason why a participating PoC server may want to provide use of the unconfirmed answer state is that the participating PoC server may be pretty sure that the terminal will accept the session anyway in a short period of time.

There are two situations where a participating PoC server can be pretty sure that the terminal will accept a session: when an automatic privileged answer mode is used or when the terminal is configured in automatic answer mode. A terminal informs its participating

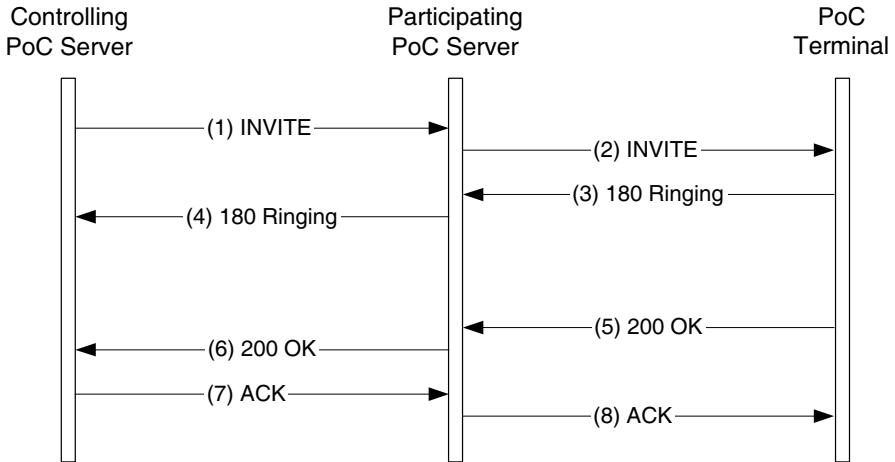


Figure 25.15: Confirmed answer state

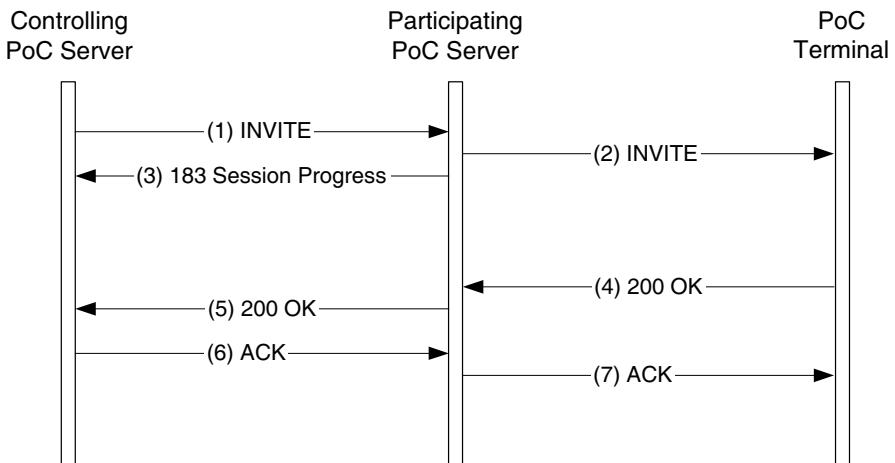


Figure 25.16: Unconfirmed answer state

PoC server of the terminal's current answer mode by sending PUBLISH requests (whose bodies follow the format specified in RFC 4354 [148]).

On receiving a response indicating an unconfirmed answer state, the controlling PoC server can provide the inviting PoC terminal with an unconfirmed right-to-send-media indication in a 200 (OK) response with a P-Answer-State header field with the value `unconfirmed`. Unconfirmed right-to-send-media indications are an optimization that allows callers to start speaking a little earlier than using traditional confirmed indications.

Note that when a PoC server generates an unconfirmed right-to-send-media indication, it needs to be ready to buffer incoming media in case the terminating terminal takes a little

longer than expected to accept the session. If a PoC server does not support media buffering, it cannot generate unconfirmed right-to-send-media indications.

25.12 Participant Information

A user taking part in a PoC group session may want to be informed about who is participating in the session. The user may want to know who accepted the invitation and who rejected it.

Users receive this type of information by having their terminals subscribe to the conference state event package (specified in RFC 4575 [289]). The controlling PoC server provides information about the participants in a PoC session by sending NOTIFY requests with XML-encoded bodies.

25.13 Barring and Instant Personal Alerts

A user who does not want to receive PoC session invitations can use *PoC session barring*. The terminal informs its home PoC server about this setting using a PUBLISH request (see Section 25.2.2). A PoC server does not relay any PoC session invitations to a terminal using PoC session barring.

When a user invites another user who is using PoC session barring to a PoC session, the inviting user will receive an error response. In this case, the inviting user can send an *Instant Personal Alert* to the user using PoC session barring. An Instant Personal Alert requests its receiver to call back (at a better time) the sender of the Instant Personal Alert. Instant Personal Alerts are implemented using MESSAGE requests.

Still, a user may not want to even receive Instant Personal Alerts. In this case, the user would use the *Instant Personal Alert Barring* feature. The terminal informs its home PoC server about this setting using a PUBLISH request (see Section 25.2.2). A PoC server does not relay any Instant Personal Alert to a terminal using Instant Personal Alert barring.

25.14 Full Duplex Call Follow On

The Full Duplex Call Follow on feature allows users involved in a PoC to establish a full duplex voice call between them if needed. The way the new full duplex voice call is established is outside the scope of PoC. Establishment procedures vary depending on the type of call (e.g., a circuit-switched call or a VoIP call).

25.15 The User Plane

The previous sections described what is known as the PoC control plane, which deals with the establishment of PoC sessions. In this section, we look at the PoC user plane. The PoC user plane includes the POC-3, POC-4, and POC-10 interfaces in Figure 25.3. These interfaces are based on RTP (see Section 16.2.2), RTCP (see Section 16.2.3), and MSRP (see Section 21.4).

The PoC's user plane supports two types of media: continuous and discrete. Continuous media (e.g., voice or video) is transported using RTP. Discrete media (e.g., instant messages) is transported using MSRP. When continuous media is used, RTP transports encoded-media and RTCP provides quality feedback.

In addition to media transport and quality feedback, the PoC user plane also includes floor control. In PoC terminology, floor control is referred to as Media Burst Control and, consequently, a floor control protocol to be used in PoC is referred to as MBCP.

At this point, the only floor control protocol or MBCP supported by PoC is an RTCP-based protocol. However, PoC supports the negotiation of the MBCP to be used for a particular session. So, in the future it may be possible to use other MBCPs such as BFCP (which is specified in RFC 4582 [106]).

PoC defines the following four QoE (Quality of Experience) profiles: Basic, Premium, Professional, and Official Government Use. A user's access network performs media resource management based on the QoE of the user's subscription.

25.15.1 Media Encoding

The default audio codecs used in PoC are the same as those defined for the IMS (see Section 15.4). Consequently, 3GPP mandates that PoC clients support, as a minimum, AMR narrowband and that PoC servers support, also as a minimum, both AMR narrowband and wideband. 3GPP2 mandates that both PoC clients and servers support, as a minimum, EVRC (Enhanced Variable Rate Codec).

25.15.2 Media Burst Control Protocol

As we mentioned earlier, the only MBCP for PoC defined so far is based on RTCP. In this section, we describe this RTCP-based floor control protocol.

25.15.2.1 Message Encoding

This MBCP is an extension to RTCP that uses RTCP APP messages. RTCP APP messages are application-specific RTCP messages. That is, RTCP APP messages defined by a particular application carry an application identifier and a message subtype, which identifies the application-specific message type.

For example, the RTCP APP message used to request a floor would carry the application identifier for OMA PoC version 2 and the message subtype corresponding to the “MBCP Media Burst Request”. In our descriptions, we refer to such a message simply as a MBCP Media Burst Request. However, the reader should keep in mind that the message is encoded as a RTCP APP message with a particular message subtype.

Of course, different message subtypes have different structures. For example, a MBCP Media Burst Request message will not have the same application-specific fields as a MBCP Media Burst Granted message.

Figure 25.17 shows the structure of an RTCP APP message. The message contains the following fields.

Ver. The current version is 2.

P. Padding.

Message Subtype. Padding.

Packet Type (APP). Identifies this message as an RTCP APP message.

Length. Length of the message.

SSRC/CSRC. Identifies the originator of this message.

Application Name. The 4-byte identifier used for OMA PoC version 2 is “PoC1” (this name comes from OMA PoC version 1).

Application-dependent Data. The structure of this field depends on the message subtype.

0		15		31
Ver	P	Message Subtype	Packet Type (APP)	Length
SSRC/CSRC				
Application Name				
Source Address				
Application-depedent Data				

Figure 25.17: RTCP APP message

25.15.2.2 Message Reliability

MBCP does not use the standard RTCP rules to send messages. Instead, it defines a set of timers that drive message retransmissions. When a PoC entity sends a MBCP message, it starts a timer. If this timer fires before an expected event occurs (e.g., a response for the message is received) the message is retransmitted.

25.15.2.3 Message Types

MBCP performs floor control in a PoC session. The PoC client that holds the floor has permission to send media. The following MBCP messages (i.e., RTCP APP message subtypes within the “PoC1” application) are defined.

MBCP Media Burst Request. A PoC client requests the floor from a PoC server.

MBCP Media Burst Granted. A PoC server grants the floor to a PoC client.

MBCP Media Burst Deny. A PoC server denies the floor to a PoC client.

MBCP Media Burst Release. A PoC client is oversending media and releases the floor.

MBCP Media Burst Idle. A PoC server informs the PoC clients in a session that no client holds the floor at that point.

MBCP Media Burst Taken. A PoC server informs the PoC clients in a session that one of the clients holds the floor at that point.

MBCP Media Burst Revoke. A PoC server revokes the floor from a PoC client.

MBCP Media Burst Acknowledgement. A PoC client acknowledges receipt of a MBCP message from a PoC server. At this point, only the MBCP Connect and MBCP Disconnect messages require the client to acknowledge their reception.

In addition, the following MBCP messages are used between PoC clients and servers using pre-established sessions.

MBCP Disconnect. A PoC server terminates a PoC session that was established using a pre-established session.

MBCP Connect. A PoC server sets up a PoC session that is being established using a pre-established session.

Some PoC servers support queuing. That is, in addition to granting or denying a media burst request right away, the server can place the request in a queue. PoC clients and servers which support queuing use the following MBCP message.

MBCP Media Burst Request Queue Status Response. APoC server informs a PoC client that a floor request has been queued.

25.15.2.4 Message Flow

Figure 25.18 shows the message flow of a PoC session between a PoC client using on-demand signaling and a PoC client using a pre-established session. The terminating PoC client is in automatic answer mode and its participating PoC server uses a confirmed right-to-send-media indication. In addition to providing MBCP message flows, this example illustrates the interactions between SIP and MBCP.

The originating PoC client generates an INVITE request (1) to establish a one-to-one PoC session. On receiving this INVITE request (1), the participating PoC server of this PoC client becomes the controlling PoC server of the session. In addition, the PoC server considers the INVITE request as an implicit floor request. That is, the PoC server acts as if the PoC client had sent a MBCP Media Burst Request message.

This type of INVITE request is considered as an implicit floor request to reduce the PoC session establishment time. Otherwise, the PoC client would need to issue a MBCP Media Burst Request right after receiving the 200 (OK) response for the INVITE request.

On receiving an INVITE request (3), the participating PoC server of the terminating PoC client responds with a 200 (OK) response (4). Since this PoC server has a pre-established session with the PoC client, the PoC server informs the client using MBCP. The PoC server sends a MBCP Connect message (6) that informs the PoC client about the new PoC session. The PoC client acknowledges receipt of the MBCP Connect message (6) with a MBCP Acknowledgement message (7).

Once the controlling PoC server receives the 200 (OK) response (4), it grants the floor to the originating PoC client by sending a MBCP Media Burst Granted message (10). The controlling PoC server also informs the terminating PoC client about the fact that the originating PoC client now holds the floor by sending a MBCP Media Burst Taken message (11).

When the originating PoC client is done sending media, it sends a MBCP Media Burst Release message (13) in order to release the floor. On receiving this message, the controlling PoC server informs the PoC clients in the session that the floor is idle by sending MBCP Media Burst Idle messages (14) and (15).

At a later point, the terminating PoC client requests the floor by sending a MBCP Media Burst Request message (17). The process followed by the controlling PoC server to grant the floor is the same as it followed before. That is, the controlling PoC server sends a MBCP

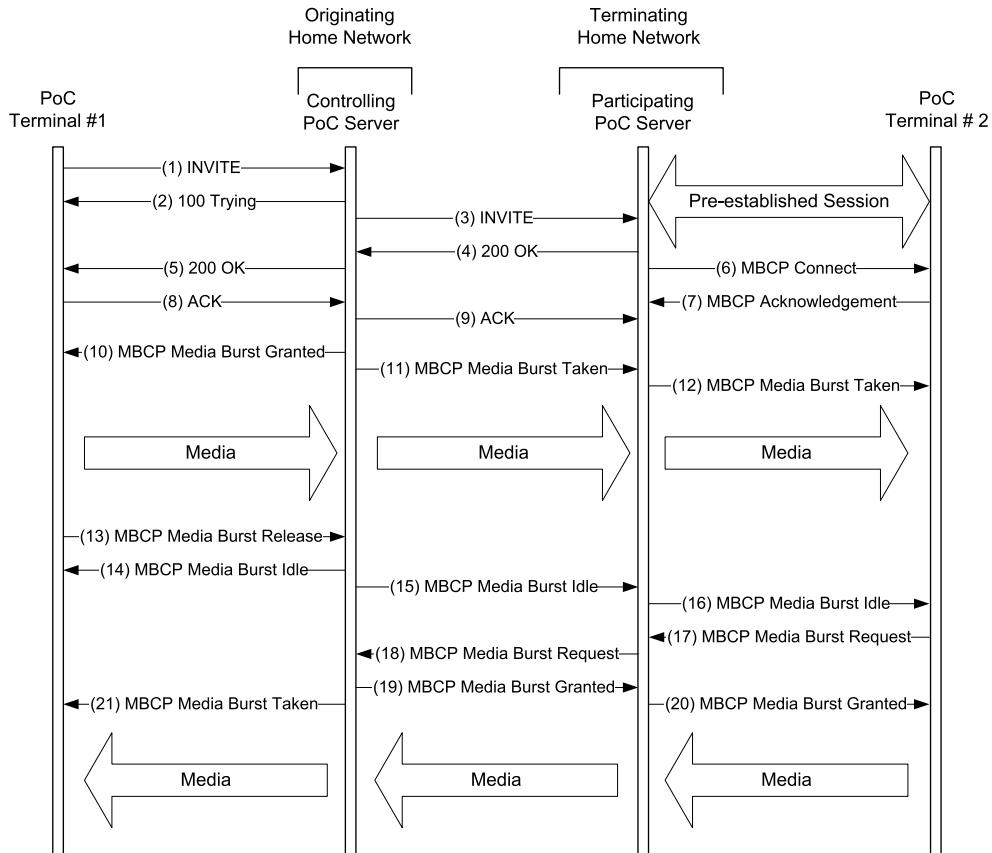


Figure 25.18: MBCP message flow

Media Burst Granted to the requester of the floor (19) and a MBCP Media Burst Taken (21) to the other PoC client. The only different is that this time this process was triggered by a MBCP Media Burst Request message, and previously it was triggered by an implicit floor request in an INVITE request.

25.16 Simultaneous PoC Sessions

A PoC client can be involved in several PoC sessions at the same time. However, a PoC client only receives media bursts for one of the sessions at a time. Otherwise, it could be confusing for the user. Figure 25.19 shows a PoC client, which is involved in two simultaneous PoC sessions, and its participating PoC server.

The participating PoC server receives two media bursts, each of them coming from a different PoC session. The participating PoC server chooses to drop the media burst that belongs to PoC Session 1 and to relay the media burst that belongs to PoC Session 2 to the PoC client.

Of course, the PoC client needs to be able to influence the decision by the PoC server of which media bursts should be relayed and which should be dropped. To influence this type

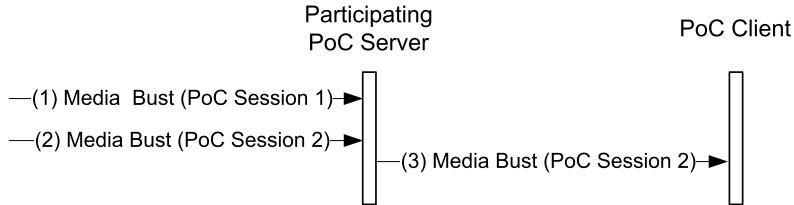


Figure 25.19: Simultaneous arrival of media bursts

of decision, PoC clients can mark a particular PoC session as the primary PoC session or lock themselves into a particular PoC session (session marking and locking are performed by using SDP parameters in an INVITE, re-INVITE, or UPDATE request). Let us have a look at how these actions by a PoC client determine the media forwarding policy at its participating PoC server.

A participating PoC server serving a PoC client has the concept of an active PoC session. At any given point, the participating PoC server only relays to the PoC client media received from the active session. If the PoC client is involved in only one PoC session, that session is the active one. Consequently, the PoC server relays media received from that PoC session to the PoC client.

When a PoC client is involved in simultaneous PoC sessions, any of the sessions may become the active one. The participating PoC server follows a set of rules to decide which session is active. The following is a summary of those rules.

- If at any point the PoC client is granted the floor in a PoC session (because the PoC client requested it previously), that session becomes the active one.
- If the PoC client locks itself in a PoC session, that session is the active one and media from other sessions is not relayed to the client.
- If the active session is the primary session, media from other sessions is not relayed to the client as long as media is received from the primary session.
- If the active session is the secondary session, media from other secondary sessions is not relayed to the client. However, if media from the primary session is received, the primary session becomes the active one (and, so, media from this session starts being relayed to the client).

25.17 Charging in PoC

The IMS PCC (Policy Control and Charging) architecture is shown in Figure 8.1. The PCC architecture supports both offline and online charging, as discussed in Section 8.2. Both offline and online charging can be applied to PoC (as specified in 3GPP TR 32.272 [25]). When offline charging is used, the PoC server communicates with the CDF over the *Rf* interface (see Figure 8.9). When online charging is used, the PoC server communicates with the online charging system over the *Ro* interface (see Figure 8.19). (The format of the CDRs used in PoC is specified in 3GPP TR 32.98 [57].)

Chapter 26

Multimedia Telephony Services: PSTN/ISDN Simulation Services

We described earlier that the core IMS enables multimedia services and traditional telephony services. In this section we focus on the traditional telephony services that, in the context of the PSTN and ISDN, are globally known as *PSTN/ISDN supplementary services*. Traditional supplementary services are: Call Forwarding, Call Hold/Resume, Connected Line Identification Presentation/Restriction, etc.

In IMS, Applications Servers (ASes) are also able to provide *PSTN/ISDN simulation services*. These are telephony services similar in nature to the *PSTN/ISDN supplementary services*, but different in the realization since SIP is the call control protocol and is quite different to the PSTN/ISDN protocols. PSTN/ISDN simulation services are a simulated version of the PSTN/ISDN supplementary services. However, the aim of the service remains.

A number of these PSTN/ISDN simulation services are specified. Most of them assume an AS that is able to provide the service, either to the originating or to the terminating party. It is perfectly valid to combine several of these services into a single AS, a decision that is left to the implementors.

The PSTN/ISDN simulation services were initially developed by ETSI TISPAN in the context of services for fixed access to IMS. Later, especially when TISPAN IMS merged with 3GPP IMS, the TISPAN-created PSTN/ISDN simulation services were slightly improved for mobile access devices and renamed as *IMS multimedia telephony services*. Through this chapter we used both terms, *IMS multimedia telephony services* and *PSTN/ISDN simulation services* in an interchangeable manner. As for documentation aspects, 3GPP combines all of these services in a single specification, 3GPP TS 24.173 [36], which is an umbrella that contains pointers to each of the specifications that describe one or more related services. However, for historical reasons, ETSI published the set of PSTN/ISDN simulation services in the ETSI namespace of specifications. Each service or group of related services is described as a separate specification. Later, those ETSI specifications were contributed to 3GPP, and published under the 3GPP namespace. In general, the reader who is interested in more information should be looking at 3GPP TS 24.173 [36].

PSTN/ISDN simulation services were developed with a clear focus on compatibility with existing PSTN/ISDN supplementary services. Although PSTN/ISDN simulation services were originally created in the context of fixed broadband access to IMS, they have been embraced and are nowadays part of the full IMS, including wireless devices.

Each PSTN/ISDN simulation service is based on the corresponding supplementary service in the PSTN/ISDN. However, in many cases the name of the service is changed in IMS, since in SIP the concept of a call can be broadened with messages, subscriptions, notifications, or multimedia sessions. Therefore, most of the PSTN/ISDN simulation services refer to *communication* rather than *calls*.

In Table 26.1 we take a brief look at each of these PSTN/ISDN simulation services. The O/T column indicates whether the service is provided to the originating side of a call (i.e., the caller) or the terminating side (i.e., the callee).

26.1 Providing Audible Announcements

Before we start looking into the different Multimedia Telephony Services, we ought to describe a common building block that is used by a few of these services: providing an audible announcement. In effect, this is inherited from the PSTN/ISDN services, where the only mechanism to provide feedback to the user was to provide an audible announcement. In IMS, one could use other mechanisms to provide feedback, such as sending an instant message, rendering a web page in the display of the phone, etc. However, the common minimum denominator, device-independent mechanism, is to provide an audible announcement. This section describes the different ways to provide an audible announcement. These feature can be used by most of the Multimedia Telephony Services that we describe in the following sections.

The procedures for audible announcements are specified in ETSI TS 183 028 [142] and its equivalent 3GPP TS 24.628 [60]. Audible announcements can be provided at the time a session is being established, during an established session, or at the time the session is being released. The procedures are slightly different in each case. Audible announcements can also be sent when a session attempt is rejected.

26.1.1 Announcement at the Time a Session is Being Established

In the case of an audible announcement that is provided at the time the session is being established there are four different ways to implement it.

- (i) If the announcement is to be rendered at the callee, then the `Call-Info` header field in the INVITE request carries the SIP or HTTP URI that identifies the source of the announcement. This makes the callee's terminal fetch the URL included in the header and render it to the user.
- (ii) If the announcement is to be rendered at the caller, then an `Alert-Info` header field is included in a 180 (Ringing) response. The header contains a SIP or HTTP URI that identifies the source of the announcement. The caller's terminal fetches the URL included in the header and renders it to the user.
- (iii) Using the Early Media mechanisms for the gateway model specified in RFC 3960 [111] in conjunction with the `P-Early-Media` header field (specified in RFC 5009 [128]).
- (iv) Using the Early Media mechanisms in an early dialog, as specified in RFC 3960 [111], in conjunction with the `P-Early-Media` header field (specified in RFC 5009 [128]). The overall effect consists of the caller having two early SIP dialogs at the time the session is being established. One dialog is established between the caller and the

Table 26.1: PSTN/ISDN simulation services in IMS

Abbreviation	PSTN/ISDN simulation service	PSTN/ISDN supplementary service	O/T
CDIV	Communication Diversion	Call Diversion	T
CFU	Communication Forwarding Unconditional	Call Forwarding Unconditional	T
CFB	Communication Forwarding on Busy user	Call Forwarding Busy	T
CFNR	Communication Forwarding on No Reply	Call Forwarding No Reply	T
CFNL	Communication Forwarding on Not Logged-in	—	T
CONF	Conference	Conference Calling	O/T
MWI	Message Waiting Indication	Message Waiting Indication	T
OIP	Originating Identification Presentation	Calling Line Identification Presentation	T
OIR	Originating Identification Restriction	Calling Line Identification Restriction	O
TIP	Terminating Identification Presentation	Connected Line Identification Presentation	O
TIR	Terminating Identification Restriction	Connected Line Identification Restriction	T
CW	Communication Waiting	Call Waiting	T
HOLD	Communication Hold	Call Hold	O/T
ACR	Anonymous Communication Rejection	Anonymous Call Rejection	T
CB	Communication Barring	Call Barring	O/T
ICB	Incoming Communication Barring	Incoming Call Barring	T
OCB	Outgoing Communication Barring	Outgoing Call Barring	O
AoC	Advice of Charge	Advice of Charge	O
CCBS	Completion of Communications to Busy Subscriber	Completion of Calls to Busy Subscriber	O
CCNR	Completion of Communications on No Reply	Completion of Calls on No Reply	O
MCID	Malicious Communication Identification	Malicious Call Identification	T
ECT	Explicit Communication Transfer	Explicit Call Transfer	O/T

callee; the other is established between the caller and the AS. The AS can then send unidirectional early media on the pre-session established with that early dialog.

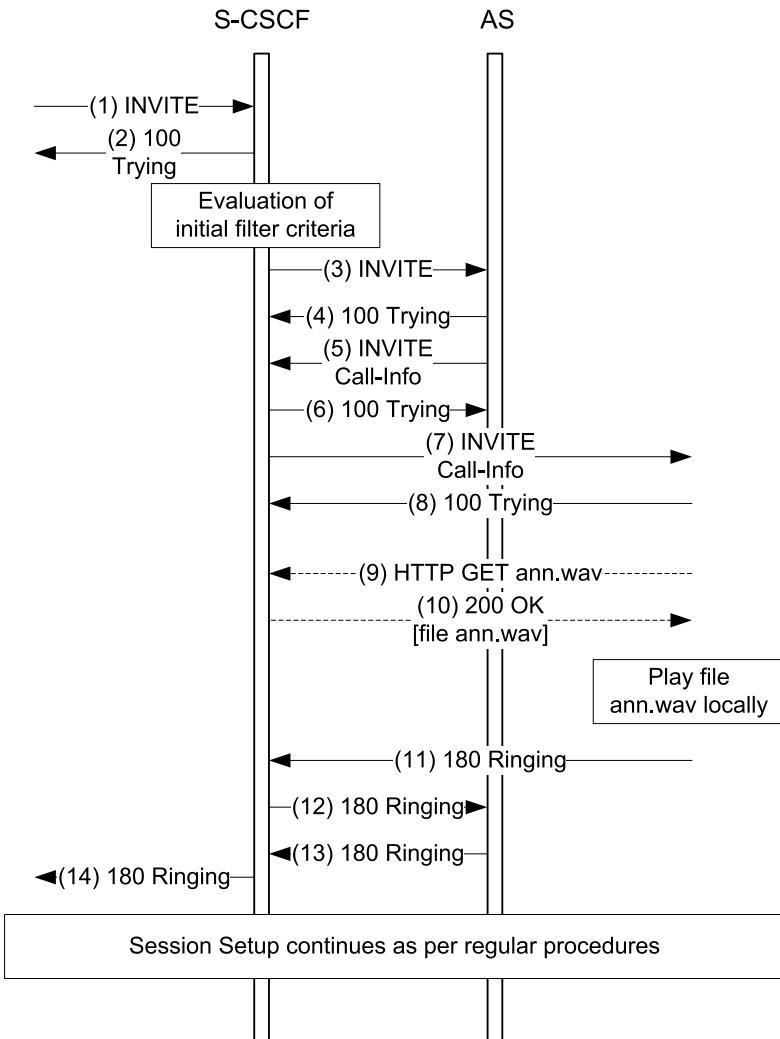


Figure 26.1: Announcement to the callee using the Call-Info header field

Figure 26.1 shows the realization of an announcement played to the callee by using the mechanism based on the `Call-Info` header field. The S-CSCF and AS can be located in either the caller's or the callee's network. The call flow starts when the S-CSCF receives an INVITE request (1) that, due to initial Filter Criteria evaluation, is forwarded (3) to an AS. The AS inserts a protoCall-Info header field that contains an HTTP link pointing to a file located in the AS itself. Then the AS performs regular routing and forwards back the request (5) to the S-CSCF, which further routes it (7) to its destination. The callee receives the INVITE request (7) and invokes the URL included in the `Call-Info` header field, sending

then an HTTP GET request (9) to retrieve the file, which is sent in a 200 (OK) response (10). Then the callee locally plays that file. The callee also sends a 180 (Ringing) response (11) which is forwarded back to the user. The session setup continues as per regular procedures.

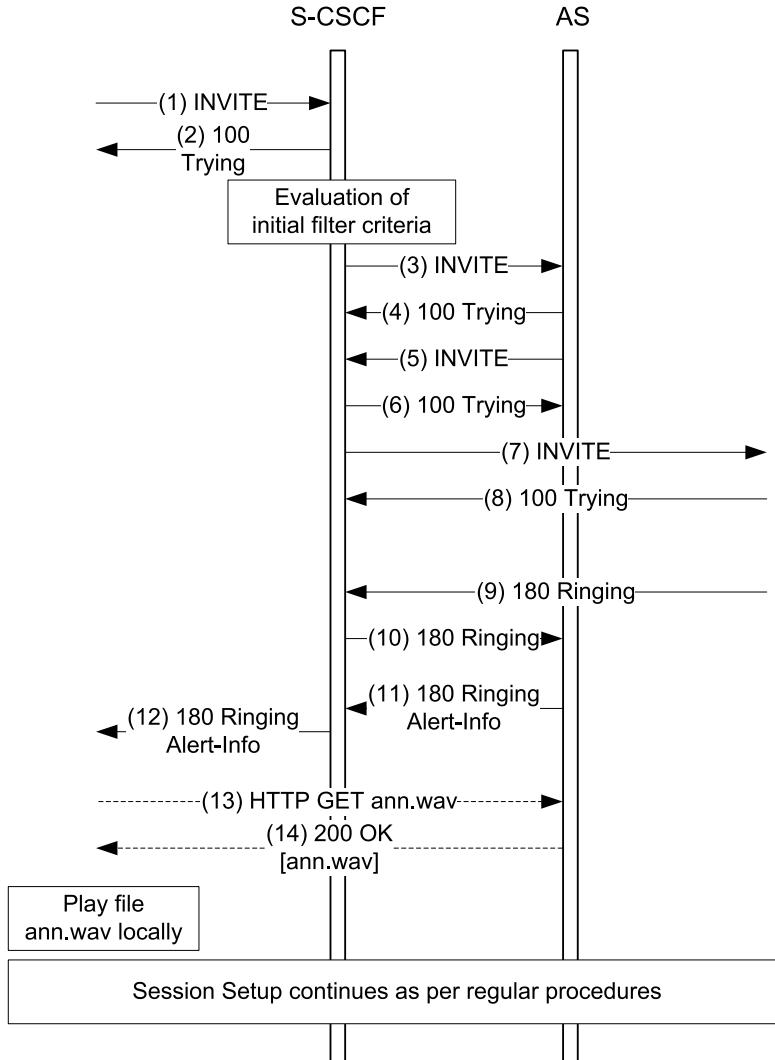


Figure 26.2: Announcement to the caller using the `Alert-Info` header field

Figure 26.2 shows the realization of an announcement played to the caller by using the mechanism based on the `Alert-Info` header field. The S-CSCF and AS can be located in either the caller's or the callee's network. The call flow starts when the S-CSCF receives an INVITE request (1) that, due to initial Filter Criteria evaluation, it is forwarded (3) to an AS. The AS performs regular routing and forwards back the request (5) to the S-CSCF, which further routes it (7) to its destination. At some point in time the callee sends a 180 (Ringing) response (9) that is received at the AS (10). The AS then adds an `Alert-Info` header field response (11) that is forwarded back to the S-CSCF (12). The S-CSCF then sends an HTTP GET request (13) to the AS to retrieve the file, which is sent in a 200 (OK) response (14) containing the file content '[ann.wav]'. A note indicates that the S-CSCF will 'Play file ann.wav locally'. A final note states that 'Session Setup continues as per regular procedures'.

that contains an HTTP link pointing to a file located on the AS itself and forward the 180 (Ringing) response (11) back to the S-CSCF and the caller (12). The caller then invokes the URI included in the Alert-Info header field of the 180 (Ringing) response; in this case, since it is an HTTP URI, it invokes an HTTP GET request (13) to retrieve the file, which is sent in a 200 (OK) response (14). The caller plays the announcement as a ringing tone until the session is setup (i.e., reception of a 200 (OK) response for an INVITE request).

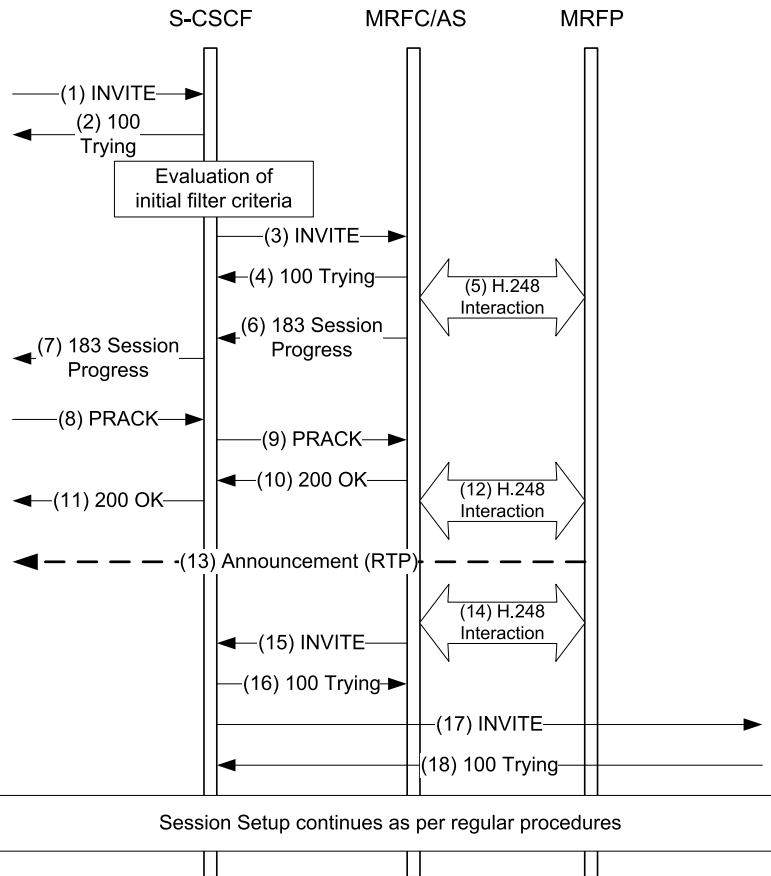


Figure 26.3: Announcement during the establishment of a session. Early media with early dialog

Figure 26.3 depicts the call flow applicable to one of these mechanisms to provide an announcement at the time the session is established. The flow, which uses the early media mechanisms, assumes an AS that can be serving either the caller or callee. The flow starts when a S-CSCF, either in the caller or callee network, receives an INVITE request (1) to establish a new session. Because of initial Filter Criteria evaluation, the S-CSCF routes the INVITE request (3) to an AS, which, together with the MRF, decides to play an announcement. The AS interacts with the MRFC to provide the announcement. The MRFC interacts (5) with the MRFP to provide the connection address for an early dialog. The MRFC then creates a reliable 183 (Session Progress) response (6) that contains SDP with

the connection parameters of the MRFP. This 183 response also contains a P-Early-Media header field indicating that authorized early media will be sent in sendonly mode. Since this 183 (Session Progress) response contains SDP, it is sent reliably to the caller, so, it requires to be acknowledged with a PRACK request (8, 9), which is responded with a corresponding 200 (OK) response (10, 11). At this point in time, the caller is able to receive audio from the MRFP. The MRFC then commands (12) the MRFP to start playing the announcement. The MRFP plays the announcement (13), and when it finishes, it interacts (14) with the MRFC to inform it. The MRFC interacts with the AS to continue processing the initial INVITE request, which is routed to its destination via the S-CSCF (15, 17). The session setup continues as per regular procedures.

26.1.2 Announcement During the Duration of the Session

In the case of an audible announcement that is provided at any time when the session has been already established, there are two mechanisms to implement it. Both of them require the presence of an AS in the signaling path acting as a B2BUA.

- The AS creates a re-INVITE request that contains a Call-Info header field pointing to the source of the announcement. This re-INVITE request can be sent to either the caller, the callee, or both.
- The AS reuses the existing audio media stream to play media on it. This might require a re-INVITE to change the connection address or the supported media types (codecs).

26.1.3 Announcement at the End of the Session

An AS that is already in the signaling path acting as a B2BUA may also send an audible announcement towards the end of the session. In this case the mechanism consists of delaying the release of the session and reusing either the established audio media stream, or creating an additional one, to play the announcement.

26.1.4 Announcement When a Session is Rejected

Audible announcements can be also provided when an Application Server is rejecting the establishment of a session. Assume that a caller sends an INVITE request that is received by an AS which decides to reject the call. The AS wants to provide an audible announcement to the caller, potentially indicating the reasons for the rejection. In this case, there are four mechanisms to implement the announcement.

- The AS includes an Error-Info header field in a 300, 400, 500, or 600 class response to the INVITE request. The Error-Info header field contains a URL that points to the source of the announcement.
- Using the Early Media mechanisms for the gateway model specified in RFC 3960 [111] in conjunction with the P-Early-Media header field (specified in RFC 5009 [128]) and the Reason header field containing a cause value.
- Using the Early Media mechanisms in an early dialog specified in RFC 3960 [111] in conjunction with the P-Early-Media header field (specified in RFC 5009 [128]) and the Reason header field containing a cause value.

- The AS first accepts the communication request, plays the in-band announcement over the established audio media stream, and then tears down the session.

26.2 Communication Diversion (CDIV) and Communication Forwarding

The Communication Diversion service allows a user to divert their communications to another user. The service is specified in ETSI TS 183 004 [134] and 3GPP TS 24.604 [65] and comprises a number of diversion services: Communication Forwarding Unconditional (CFU), Communication Forwarding on Busy user (CFB), Communication Forwarding on No Reply (CFNR), Communication Deflection (CD), Communication Forwarding on Subscriber Not Reachable (CFNRC), and Communication Forwarding on Not Logged-in (CFNL).

The service is implemented in an application server that, depending on the actual service, automatically diverts all session attempts to the given user, or after the SIP request has been sent to the user, but the user sent a busy response, did not answer, etc.

CDIV also implements the `History-Info` header field to record a forwarding operation. This allows both the caller and callee to learn that the call has suffered some forwarding en route. We described the `History-Info` in Section 5.9.1.

Let us take a look at the service flow with the help of Figure 26.4. The reader may have noticed that the figure is quite simplified and only nodes and signaling flows that are relevant for the implementation of the service are depicted. According to Figure 26.4, the S-CSCF that is responsible for the called party receives an INVITE request (1), addressed in the Request-URI to the called party, e.g., `sip:bob@home2.net`. The S-CSCF evaluates the initial Filter Criteria and, as a result, forwards the INVITE request (2) to an AS, which happens to execute the Communication Forwarding Unconditional service. If the AS is acting as a Back-To-Back User Agent, then it can terminate the SIP signaling and generate a 181 (Call is being forwarded) response back to the calling party. The B2BUA AS, as part of the service logic, replaces the Request-URI field with that of the diverted user, e.g., `sip:charlie@home3.net`, creates a new SIP dialog, and generates a new INVITE request addressed to the new target.

If the AS is acting as a proxy, the procedures are similar, except that the AS does not generate 181 responses, and continues to proxy the received INVITE request, therefore, keeping the `From`, `To`, `Call-ID`, etc. header fields intact.

The AS also records the retargeting when it adds a new value to the `History-Info` header field. Figure 26.5 shows an example of this `History-Info` header field. The first entry in the `History-Info` header field denotes the original Request-URI, while the second entry indicates the retargeted URI. The cause parameter, specified in RFC 4458 [192], which receives the code 302, has the purpose of indicating the type of communication diversion service. Table 26.2 contains the complete list of redirection codes and their mapping to communication diversion services.

Since the `History-Info` header field might create privacy concerns, a new value called `history` is added to the `Privacy` header field. A user who does not want the network to record the history information adds the `history` value to the `Privacy` header field.

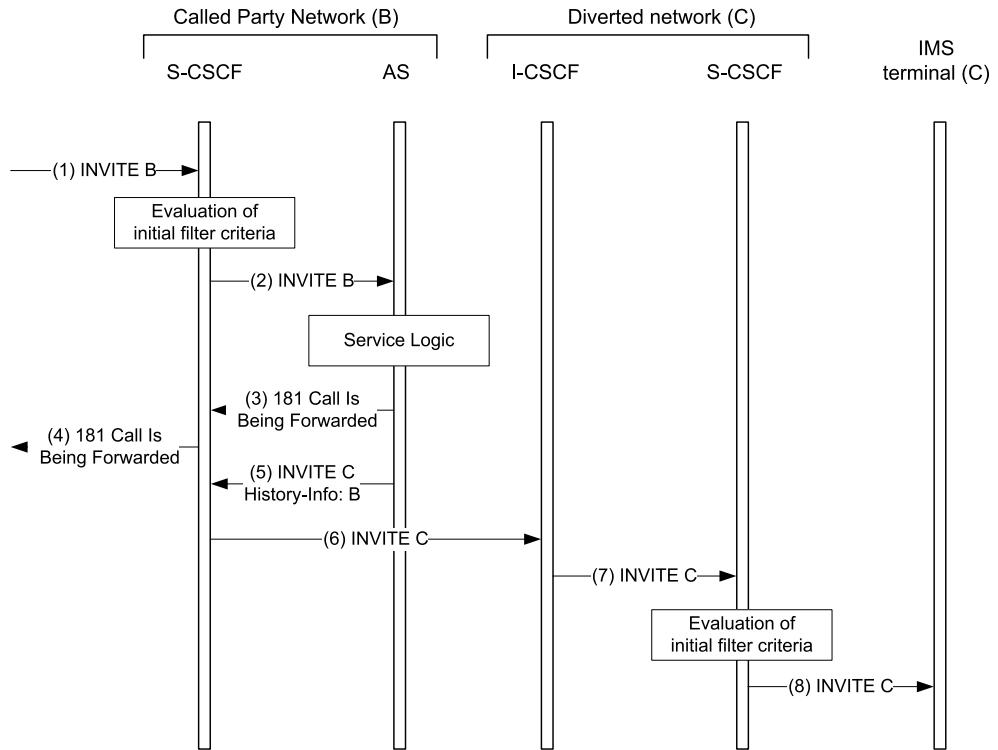


Figure 26.4: Communication Diversion: Communication Forwarding Unconditional

History-Info: <sip:bob@home2.net>;index=1,
 <sip:charlie@home3.net;Reason=SIP;cause=302>;index=1.1

Figure 26.5: The AS inserts a History-Info header field to INVITE request (5)

Table 26.2: Mapping between diversion conditions and the cause parameter code

Communication diversion condition	Code
Communication Forwarding Busy	486
Communication Forwarding on No Reply	408
Communication Forwarding Unconditional	302
Communication Forwarding on Not Logged-in	404
Communication Forwarding on Subscriber Not reachable	503
Communication Deflection (Immediate response)	480
Communication Deflection (During Alerting)	487

26.3 Communication Diversion Notification (CDIVN)

The Communication Diversion Notification (CDIVN) is not a service by itself, but a complement to CDIV. It allows users of the CDIV service to be informed when the service is actually executed, i.e., when a diversion is served to the user.

CDIVN is implemented on top of the SIP-event notification framework specified in RFC 3265 [264]. The service has defined a new event package, called `comm-div-info`, whose data format provides the required notification information.

When a UA wants to use the CDIVN service, it sends a SUBSCRIBE request addressed to the Public User Identity of the user. The Event header field is populated with the name of the event package: `comm-div-info`. This SUBSCRIBE request is routed, with the help of the initial Filter Criteria, to the AS that executes CDIV services. Then, whenever the CDIV AS executes a diversion to this user, it sends a notification with detailed information.

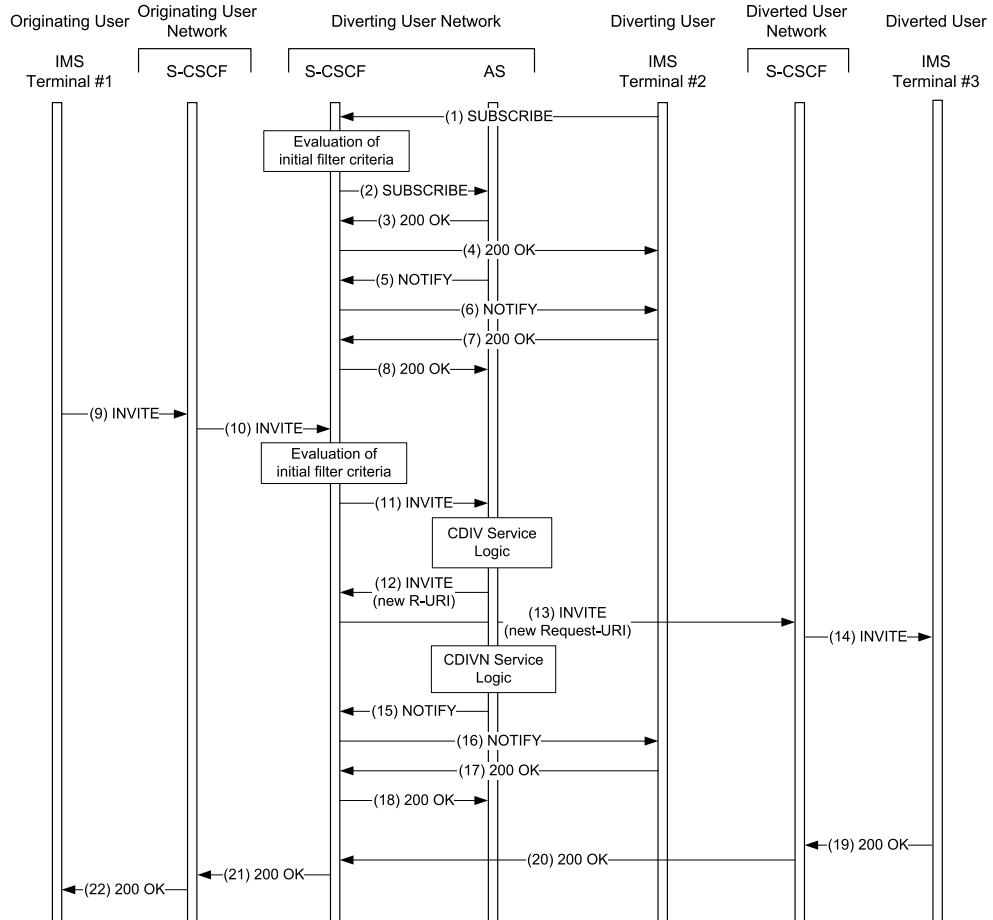


Figure 26.6: Communication Diversion Notification

Figure 26.6 depicts a complete flow including Communication Diversion and Communication Diversion Notification. There are three users and three networks involved. For the sake of simplicity, only relevant nodes and messages are shown.

The flow starts when the user who has activated the CDIV service subscribes to the complementary CDIVN service by sending a SUBSCRIBE request (1) addressed to himself, and containing the Event header field set to the `comm-div-info` event. His S-CSCF receives the request, evaluates the initial Filter Criteria, and forwards the SUBSCRIBE request (2)

to an AS, which is the same AS that executes CDIV and CDIVN. The AS replies with a 200 (OK) response (3, 4) and then a NOTIFY request (5, 6) that contains information about the subscription state.

At some point later in time, an the originating user starts a session addressed to the diverting user by sending an INVITE request (9) which traverses the originating user S-CSCF and is routed to the diverting user S-CSCF (10). This S-CSCF evaluates the initial Filter Criteria and as a result the INVITE request (11) is forwarded to the CDIV/CDIVN AS. This AS executes the CDIV service logic, replaces the Request-URI with the SIP URI of the diverted user, adds a History-Info header field, and forwards the INVITE request (12) back to the S-CSCF, honoring an existing Route header field. The diverting user's S-CSCF routes the INVITE request (13) to the diverted user network, arriving eventually to the S-CSCF allocated to the diverted user. This S-CSCF, once the request is routed via zero or more ASes, forwards the request (14) to the user. Coming back to the CDIV/CDIVN AS, it also executes the CDIVN service by sending a NOTIFY request (15), which is routed (16) via the diverting user's S-CSCF to the subscriber. The diverted user can inspect the body of the NOTIFY request to find out the details of the diversion (such as originating URI, diverted URI, time of diversion, etc.).

Sometimes the user cannot receive a notification of an executed diversion because the user is not registered to the network. Communication Forwarding on Subscriber Not Reachable (CFNRc) and Communication Forwarding on Not Logged-in (CFNL) are examples of services where the user is not registered at the time a diversion occurs. Therefore, it is not possible that the user receives a notification of the CDIVN service. In these cases, the event state information is cached and the notification is deferred until the user subscribes again. However, the notification is not deferred forever, but just a given amount of time (operator configurable timer). Should the user register to the IMS network and subscribe to the CDIVN service before that timer expires, the CDIVN AS will send him a notification including all of the past missed events.

26.4 Conference (CONF)

The Conference service provides a centralized conference server that is able to accept either multimedia participants or PSTN participants. CONF is specified in ETSI TS 183 005 [135] and 3GPP TS 24.605 [66], which are largely based on 3GPP TS 24.147 [32].

The service makes extensive usage of the conference event package, specified in RFC 4575 [289]. This allows participants in a conference to subscribe to the conference state and to receive notifications with the list and status of participants, types of media, topic of discussion, and other conference-related information.

The service also uses the SIP Replaces header field (specified in RFC 3891 [213]) in order to allow an *ad-hoc* conference to become centralized.

We have already discussed this service in Chapter 24. Please refer to that chapter for more detailed information.

26.5 Message Waiting Indication (MWI)

The MWI service allows a user to be informed about the reception of a multimedia voice mail in their multimedia voice mail server. The information can be delivered in real time, but at the time a message is left in the multimedia voice mail server, the user is not typically online. Therefore, users are notified of existing voice mails when they connect to the network. The

service is specified in ETSI TS 183 006 [139] and 3GPP TS 24.606[70]. They are a straight implementation of `message-summary` event package specified in RFC 3842 [212].

MWI defines the concept of an MWI AS. This is the AS that keeps track of the received voice or multimedia message. IMS terminals can subscribe to the `message-summary` event package. The MWI AS will provide notifications of received messages.

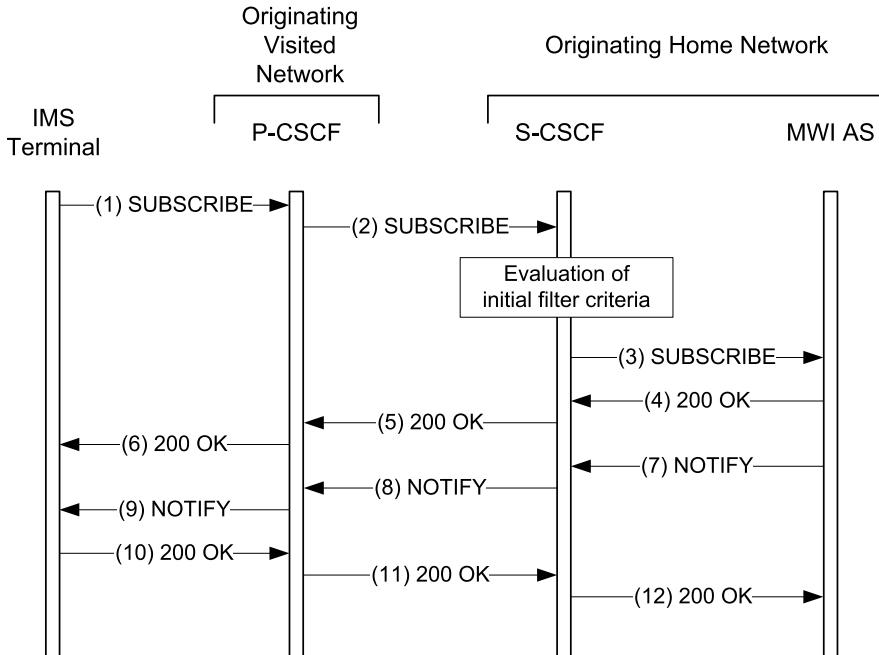


Figure 26.7: Subscription to the MWI service

Figure 26.7 depicts the signaling flow for a subscription to the MWI service. Typically this subscription might take place when the phone registers to the IMS, so if user has received voice or multimedia messages while he or she was not registered to the IMS, the user is notified as soon as possible. The IMS terminal sends a SUBSCRIBE request (1) that contains an Event header field set to `message-summary`. The request is forwarded to the S-CSCF via the P-CSCF (1, 2). The S-CSCF evaluates the initial Filter Criteria and concludes that it should forward the request (3) to the MWI AS, which acknowledges receipt of the SUBSCRIBE request with a 200 (OK) response (4). The MWI AS also creates a NOTIFY request (7) that contains a message summary document. This NOTIFY request eventually reaches the IMS terminal (9) which sends a 200 OK response (10).

The NOTIFY request (7, 8, 9) creates a simple-message-summary document, such as that included in Figure 26.8. The payload of the NOTIFY request contains a summary of the received messages for all of the different public user identities allocated to the user. Messages are classified by its type: voice messages, video messages, fax messages, etc. The digits indicate the number of new/old messages, with the number of urgent messages within brackets. A short summary of each message follows.

Once the IMS terminal receives the first NOTIFY request that contains a message summary document, it is not likely that further NOTIFY requests will be received, because

```
 NOTIFY sip:[1080::8:800:200C:417A]:5059;comp=sigcomp SIP/2.0
Via: SIP/2.0/UDP mwi.home1.net;branch=z9hG4bK332b23.1
Max-Forwards: 70
Route: <sip:scscf1.home1.net;lr>, <sip:pcscf1.home1.net;lr>
From: <sip:user1@home1.net>;tag=31415
To: <sip:user1@home1.net>;tag=151170
Call-ID: b89rjhnedlrfjfslsj40a222
CSeq: 43 NOTIFY
Subscription-State: active;expires=600000
Event: message-summary
Contact: <sip:mwi.home1.net>
Content-Type: application/simple-message-summary
Content-Length: 816

Messages-Waiting: yes
Message-Account: sip:user1@home1.net
Voice-Message: 4/1 (2/0)
Video-Message: 1/1 (0/0)
Fax-Message: 1/1 (0/1)

To: <user1_public1@home1.net>
From: <user2_public1@home1.net>
Subject: call me back!
Date: 19 Apr 2005 21:45:31 -0700
Priority: urgent
Message-ID: 27775334485@mwi.home1.net
Message-Context: voice-message

To: <user1_public2@home1.net>
From: <user2_public1@home1.net>
Subject: Where are you that late???
Date: 19 Apr 2005 23:45:31 -0700
Priority: urgent
Message-ID: 27775334485@mwi.home1.net
Message-Context: voice-message

To: <user1_public1@home1.net>
From: <user3_public1@home3.net>
Subject: Did you see that penalty!!!
Date: Tue, 19 Apr 2005 22:12:31 -0700
Priority: normal
Message-ID: 26775334485@mwi.home1.net
Message-Context: video-message
```

Figure 26.8: NOTIFY request (7) in the MWI service

it is expected that the user will receive incoming sessions directly. However, there is a combination of services that brings additional use cases. Assume a user who does not answer an incoming session, and the user has activated some of the Communication Forwarding services, e.g., Communication Forwarding on No Reply, Communication Forwarding on Busy user, etc., where the incoming session is diverted to a multimedia voice mail server. Then the user will receive an updated notification if its IMS terminal is still subscribed to the message-summary event package.

26.6 Originating Identification Presentation/Restriction (OIP, OIR)

The Originating Identity Presentation (OIP) service provides the callee with the possibility of receiving the trusted identity of the caller. On the other side, the Originating Identity Restriction (OIR) service provides the caller with the possibility of restricting the presentation of the caller's identity at the callee's terminal. It must be noted that OIP is a service offered to callees, whereas OIR is a service offered to callers. Both services are specified in ETSI TS 183 007 [140] and 3GPP TS 24.607 [71], although the service is mostly inherent to SIP.

OIP and OIR use the Privacy header field (specified in RFC 3323 [247]), the P-Asserted-Identity, and P-Preferred-Identity header fields (both specified in RFC 3325 [194]). The usage of these headers is further defined in 3GPP TS 24.229 [37]. These services also provide the user with the *Ut* interface (see Section 26.14) or other configuration interface to set their preferences.

The OIP service is enabled by the caller adding a P-Preferred-Identity header field that contains a registered identity for the caller. The P-CSCF verifies that the value of the header is correct and, based on the authenticated information of the IMS terminal, replaces the P-Preferred-Identity with a P-Asserted-Identity header field.

The OIR service provides two modes of operation: permanent and temporary. The former acts automatically on all SIP requests issued by the user, whereas the latter requires a specific action on a per call basis. The mode of operation is configured in the AS, typically using the *Ut* interface (see Section 26.14).

In OIR permanent mode, an AS inserts a Privacy header field set to "id" or "header", depending on the user's pre-arranged preferences. The value "header" in the Privacy header is the way the user can indicate to the AS to obfuscate or strip those headers that might otherwise reveal information about the user. This includes stripping *Via* and *Record-Route* header fields and obfuscating the *Contact* header field. The *From* header field is selected by the terminal, and it should not reveal information.

When a terminal selects the value "id" in the Privacy header field, it indicates to the AS to keep the P-Asserted-Identity header field available to entities belonging to the trusted domain (e.g., the home network and selected networks), but to remove that header field when the request leaves the trusted domain. Notice that the callee is never trusted, so when the Privacy header field is set to "id", the P-Asserted-Identity header field is never delivered to the callee.

In OIR temporary mode, the UE inserts a Privacy header field that contains, for example, the "id" value. In that case the callee's S-CSCF removes the P-Asserted-Identity header field, so that the callee does not receive it in the SIP request.

The service requires transitive trust between the different networks involved in a session. In the lack of trust between networks, the P-Asserted-Identity header field is lost.

A more sophisticated mechanism based on cryptographic assurance of the caller's identity is not supported at present.

26.7 Terminating Identification Presentation/Restriction (TIP, TIR)

The Terminating Identification Presentation (TIP) and Terminating Identification Restriction (TIR) services are counterparts to the OIP/OIR services. The TIP service provides the caller with the possibility of receiving the asserted identity of the callee, whereas the TIR service provides the callee with the possibility of restricting the presentation of their identity to the caller. It must be noted that TIP is a service offered to callers, whereas TIR is a service offered to callees. Both services are specified in ETSI TS 183 008 [141] and 3GPP TS 24.608 [52] and are based on a straight implementation of the Privacy and P-Asserted-Identity header fields, specified in RFC 3323 [247] and RFC 3325 [194], respectively.

One would think in principle that the services would not be very useful, because the caller is calling the callee, so the caller should know the identity of the callee in advance. Although the thought is correct, there might be scenarios including redirections, call diversions, etc., where the callee answering the call is not the one that the caller originally called. Hence, the TIP/TIR services become a bit more interesting.

Similarly to OIR, TIR provides two modes of operation: temporary and permanent. The mode of operation is configured in the AS, typically using the *Ut* interface (see Section 26.14).

In temporary TIR mode, the behavior of the AS depends on the default action if there is a lack of indication from the callee. If the callee does not include a Privacy header, then the default behavior applies. In case the default behavior is to restrict the callee's identity, the AS inserts a Privacy header set to "id". However, if the default behavior for temporary TIR mode is "not restricted", then the AS takes no action.

In permanent TIR mode, if the callee does not include a Privacy header field in a SIP response, an AS will include a Privacy field set to "id" to indicate that the identity of the callee should not be revealed to the caller.

In any case, the callee can always request the TIR service by inserting a Privacy header field set to "id" in any non-100 SIP response.

At the caller's side, regular procedures take place, typically at an AS, which, if the Privacy header field is set to "id", removes any P-Asserted-Identity header field present in any response.

The TIP service is fairly simple: if the callee does not want to hide their identity, then the caller receives the identity in the P-Asserted-Identity header field in responses.

26.8 Anonymous Communication Rejection (ACR) and Communication Barring (CB)

The following three similar services are collectively known as Communication Barring (CB) services.

- (a) The Incoming Communications Barring (ICB) allows a callee to instruct the network to reject incoming communications that fulfil certain conditions.
- (b) The Anonymous Communication Rejection (ACR) constitutes a particular case of ICB, the particular condition for rejection being that the incoming request is anonymous.

- (c) The Outgoing Communication Barring (OCB) provides a mechanism for the network to reject outgoing communications that fulfil certain conditions.

It should be noted that ICB and ACR are services offered to the callee, whereas OCB is a service offered to the caller. The three services, which are specified in ETSI TS 183 011 [133] and 3GPP TS 24.611 [64], are implemented in an AS.

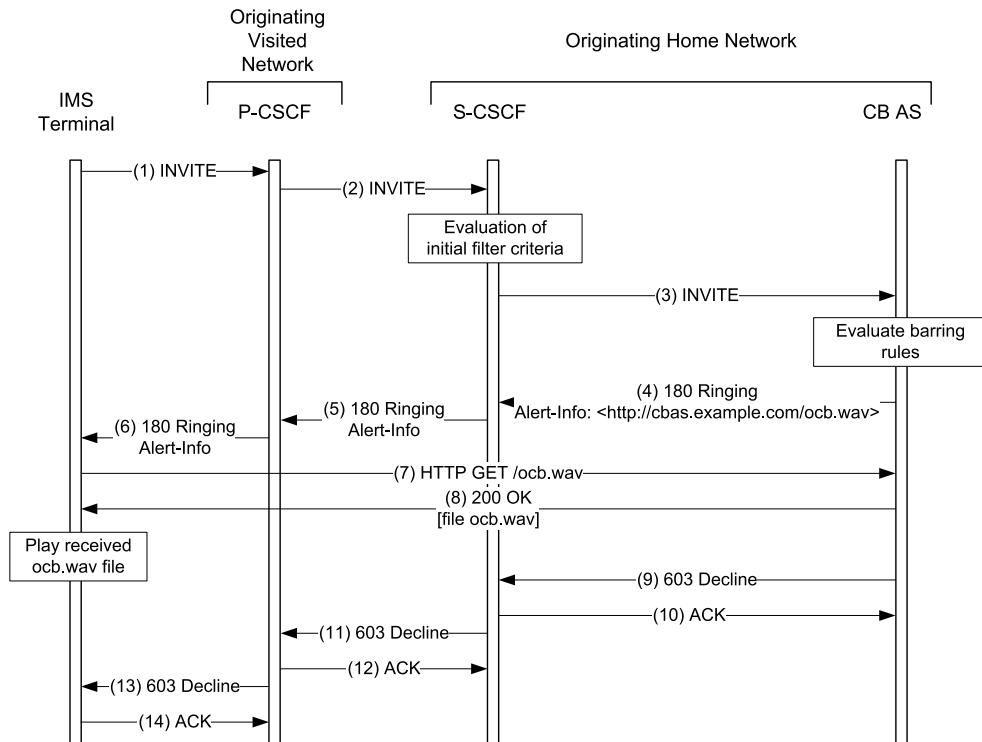


Figure 26.9: Outgoing Communication Barring (OCB)

An originating AS that executes the OCB service, rejects an outbound session attempt, generates a 603 (Decline) response, and sends it to the calling party. Prior to generating the 603 (Decline) response, it can also send an audible announcement to the calling party. In case the `Alert-Info` header field is used in a 180 (Ringing) response, the value of that header contains a SIP or HTTP URI that identifies the announcement resource. This is depicted in Figure 26.9, where the initial INVITE request (1, 2, 3) is routed, due to a criterion of the initial Filter Criteria, to the Communication Barring (CB) AS. The CB AS has a set of rules that dictate which outgoing sessions are barred. For example, these rules, which might be set by the company that pays for the sessions, may indicate that sessions to premium services are barred. If the session should be barred, the CB AS generates a 180 (Ringing) response (4, 5, 6) that contains an `Alert-Info` header field, whose value contains a URI. When the IMS terminal receives the 180 (Ringing) response, it contacts the server indicated in that URI, in this case, with an HTTP GET (7) request. The HTTP server sends the requested file in a 200 (OK) response (8). The IMS terminal then plays the downloaded file. At some point in

time the CB AS generates a 603 (Decline) response (9, 11, 13), which finalizes the session attempt.

Figure 26.10 shows an ICB service with an announcement signaled through the Alert-Info header field. The call flow is similar to the OCB service, with the difference that the ICB service is executed in the terminating network. Therefore, the announcement is played to the caller, which generally is not a user of the terminating network. Thus, it requires a proper policy (e.g., a proper firewall configuration) to allow access to the announcement server from networks other than the home network of the user.

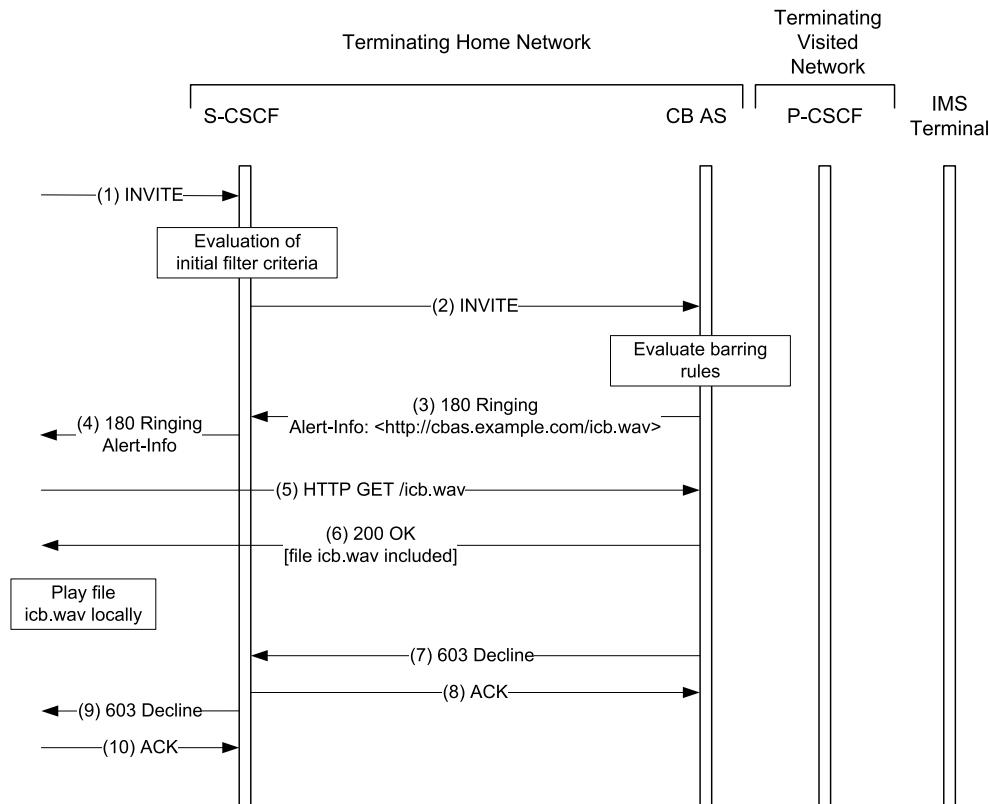


Figure 26.10: Incoming Communication Barring (ICB)

26.9 Advice of Charge (AoC)

The AoC service offers the served user with information about the approximate costs of the session. The service is specified in ETSI TS 183 047 [131] and 3GPP TS 24.647 [61], and only applies to sessions initiated with an INVITE request, leaving out, for example, subscriptions, stand-alone MESSAGE requests, etc.

Depending on at which point in time the AoC information is offered, there are several types of AoC information.

Advice of Charge at setup time (AoC-S). The charging information is supplied at the time the session is established. If the charging rates change during the session, the new rates are also supplied.

Advice of Charge during the communication (AoC-D). Cumulative charging information is supplied periodically during the time the session is active.

Advice of Charge at the end of the communication (AoC-E). The charging information is supplied when the session is ended.

The service is implemented in an AS that acts as a Back-to-Back User Agent (B2BUA). This AS receives the signaling information, collects and calculates the charging information, and supplies it back to the served user. The charging information is supplied in an XML document. This AoC XML document is transported in different SIP messages, e.g., INVITE, INFO, BYE, or any response to any of them. Figure 26.11 shows an example of one of these charging documents that contains AoC-S and AoC-D information.

```
<?xml version="1.0" encoding="UTF-8"?>
<aoc xmlns="http://uri.etsi.org/ngn/params/xml/simservs/aoc"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      >
  <aoc-s>
    <charged-items>
      <basic>
        <free-charge/>
      </basic>
    </charged-items>
  </aoc-s>
  <aoc-d>
    <charging-info>subtotal</charging-info>
    <recorded-charges>
      <recorded-currency-units>
        <currency-id>EUR</currency-id>
        <currency-amount>0.22</currency-amount>
      </recorded-currency-units>
    </recorded-charges>
    <billing-id>normal-charging</billing-id>
  </aoc-d>
</aoc>
```

Figure 26.11: Example of an AoC XML document

An associated MIME type of application/vnd.etsi.aoc+xml identifies an AoC XML document in a Content-Type header field of the message that carries the content. In addition, SIP messages carrying an AoC XML document set the Content-Type header field to the value `render` and the `handling` parameter is set to `optional`, in order to indicate that the user should be informed with the contents of the AoC XML document, although this information is not strictly necessary for the session handling.

Figure 26.12 shows a possible call flow for the AoC-S service. A user sends an INVITE request (1), which is received at their S-CSCF (2), and through the evaluation of the initial

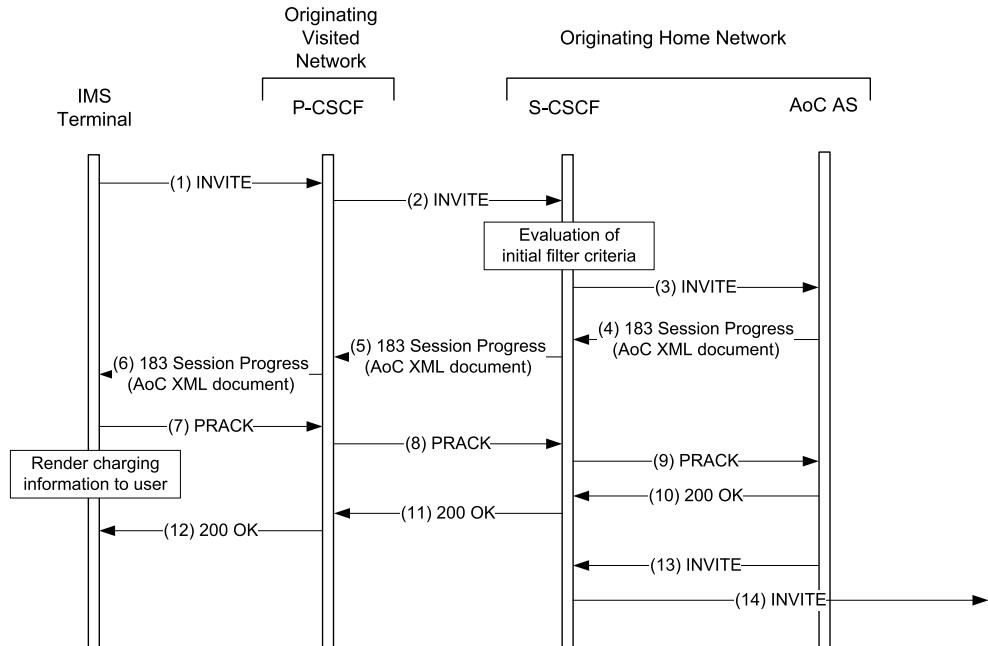


Figure 26.12: Advice of Charge at Set-Up time

Filter Criteria, the INVITE request (3) is forwarded to the AoC AS. The AoC AS is provisioned to provide AoC-S information, so, it analyzes the INVITE request and its SDP body, calculates the foreseen price, generates an AoC XML document, and adds it to a 183 (Session Progress) response (4), which is sent reliably back to the user (5, 6). The IMS terminal generates a PRACK request (7) and renders the received charging information to the user. The PRACK request is forwarded to the AoC AS (8, 9), and acknowledged with a 200 (OK) response (10, 11, 12). The AoC AS acts as a B2BUA and maintain different call legs between the caller and the callee, so it also generates an INVITE request (13) based on the contents of the received INVITE request (3), which is forwarded via the S-CSCF to its destination (4). The session setup completes as usual (not shown in the figure).

The call flow in Figure 26.12 can be applied when the caller supports reliable provisional responses, e.g., the INVITE request contains a `Supported` header field with the value `100rel` in it. Otherwise, the AoC AS cannot add an AoC XML body to the 183 (Session Progress) response, because it could be lost and the user would not receive the charging information. If the IMS terminal did not support provisional responses, the AoC AS would delay the inclusion of the AoC XML document until the 200 (OK) response to the INVITE request (which is always reliable, due to the following ACK request).

The call flow in Figure 26.12 depicts the service provided to the caller. The AoC-S service can also be provided at the callee side. In that case, the INVITE request would have been routed to the AoC AS, which acting as a B2BUA, would have added an AoC XML document to the request. Notice that, since the INVITE request already contains an SDP body, when a second body is added, it would have been wrapped in a multipart MIME wrapper.

If both users are subscribed to AoC-D to their respective network operators, when their AoC ASes have relevant information to send to their served user, they generate a new AoC

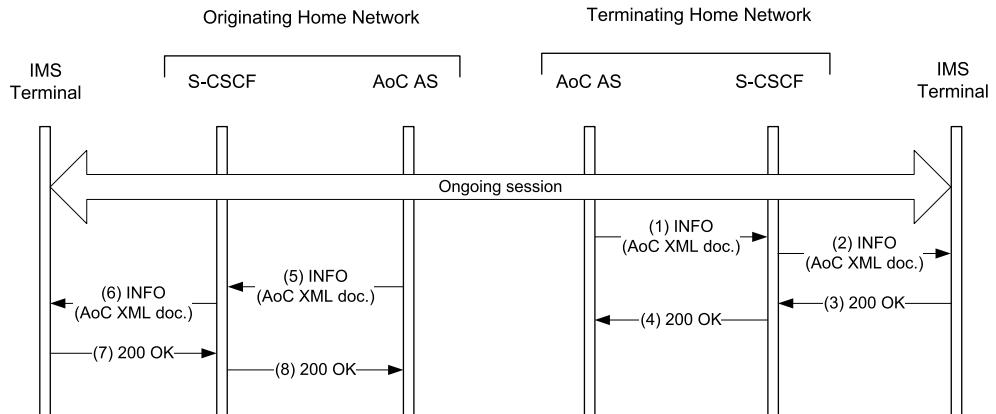


Figure 26.13: Advice of Charge during a communication

XML document and add it to an INFO request, as shown in Figure 26.13, flows 1 and 2 for the terminating side, and flows 5 and 6 for the originating side. The IMS terminals then render the received information to the user. Note that there is no synchronization between the originating and terminating network although the figure just shows both users being provided with the same AoC-D service.

Figure 26.14 depicts the flow for AoC-E, when both users receive the service from their respective networks. Either the BYE request (6, 7) or the 200 (OK) response (13, 14) to it includes an AoC XML document generated by the respective AoC AS. The document is then rendered to the user at the same time that the session is closed.

26.10 Completion of Communications to Busy Subscriber (CCBS) and Completion of Communications on No Reply (CCNR)

The CCBS and CCNR services, which at the time of this writing are still in draft status (work in progress), are being specified in ETSI TS 183 042 [130] and 3GPP TS 24.642 [62]. Both services have quite a few similarities, so, whenever possible, we describe both services at the same time.

The CCBS and CCNR services work as follows: a caller calls a callee who is busy (CCBS) or does not reply (CCBR). The caller would like to setup the session as soon as the callee is available, so the caller invokes the CCBS or CCNR service. An AS logs the signaling and inserts the call attempt into a queue management system. The AS then monitors the call status of the callee. When the callee becomes available again, the AS pops up the next call attempt in the queue and calls the caller (this process is called *recall*). When the caller accepts the recall, the AS will call the callee. If the callee answers this time, the AS will cross connect both legs of the call.

The CCBS/CCNR service provides a queue management system, so that if several callers are waiting for a callee to become available, the service is able to manage the queue and assign a virtual timeslot to the caller to place their call again, once the callee becomes available. The queue management system also allows the caller to delete or suspend a queued call.

The IETF has developed a dialog event package (specified in RFC 4235 [290]) that solves the problem of finding when a callee becomes available again. This is instantiated

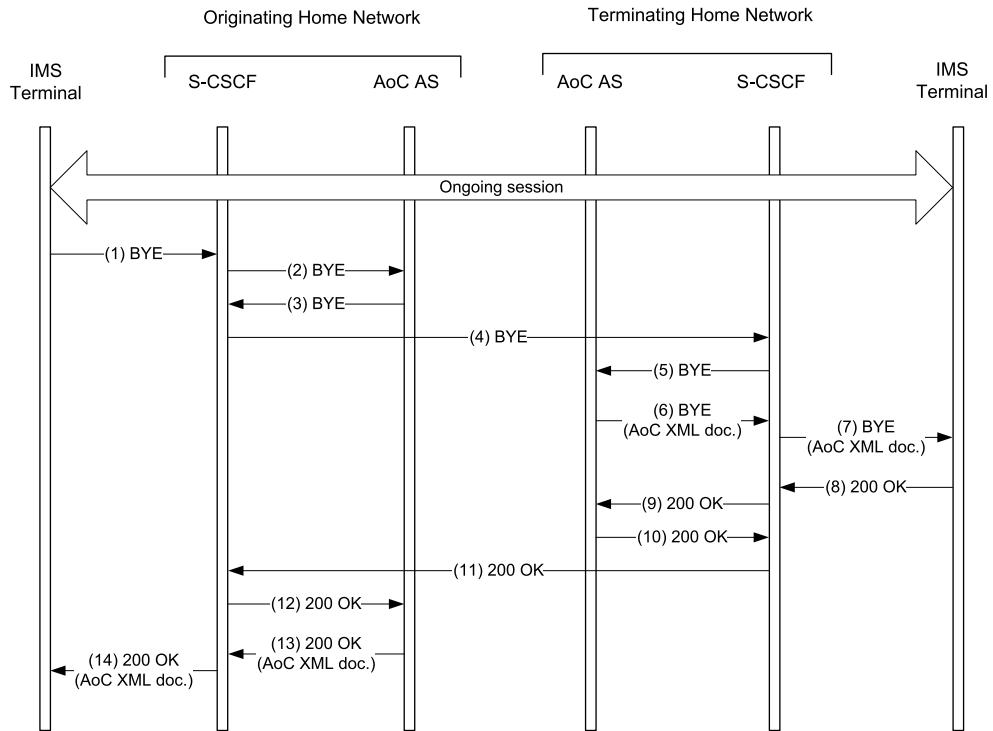


Figure 26.14: Advice of Charge at the end of a communication

by providing a subscription/notification service to a call (SIP dialog) on a terminal. However, the dialog event package is not able to manage queues, which is a requirement for the CCBS/CCNR services. The approach taken by these services consists of inserting an AS serving in the signaling path at the caller side, and another AS serving the callee see. These ASes subscribe to the dialog event package of the caller and callee to provide queue support.

Since the service is still being specified at the time of this writing, the actual details of the implementation of the service are not yet clear.

26.11 Malicious Communication Identification (MCID)

The MCID service allows a user to indicate their suspicions of a malicious communication (e.g., harassing, threatening, obscene, etc.). The network then records the identities of the caller and callee and the date and time of invocation in a log . The log is made available to the appropriate authorities (typically this log is not made available to the invoking user).

The MCID service (specified in ETSI TS 183 016 [138] and 3GPP TS 24.616 [69]) allows two modes of operation: permanent and temporary. In permanent MCID, all communications where the user is the callee are logged. In temporary mode, the callee selects which communications must be logged.

The service is implemented in an AS at the callee's home network. All SIP signaling destined for the user is forwarded to the MCID AS, independently of whether the service is operating in permanent or temporary mode. In permanent mode, all relevant SIP parameters

are inserted in the log. In temporary mode, the same parameters are first put into a temporary log. Only upon an indication from the user is the temporary log copied to the permanent log. In any case, the temporary log is always cleared some time after the communication ceases.

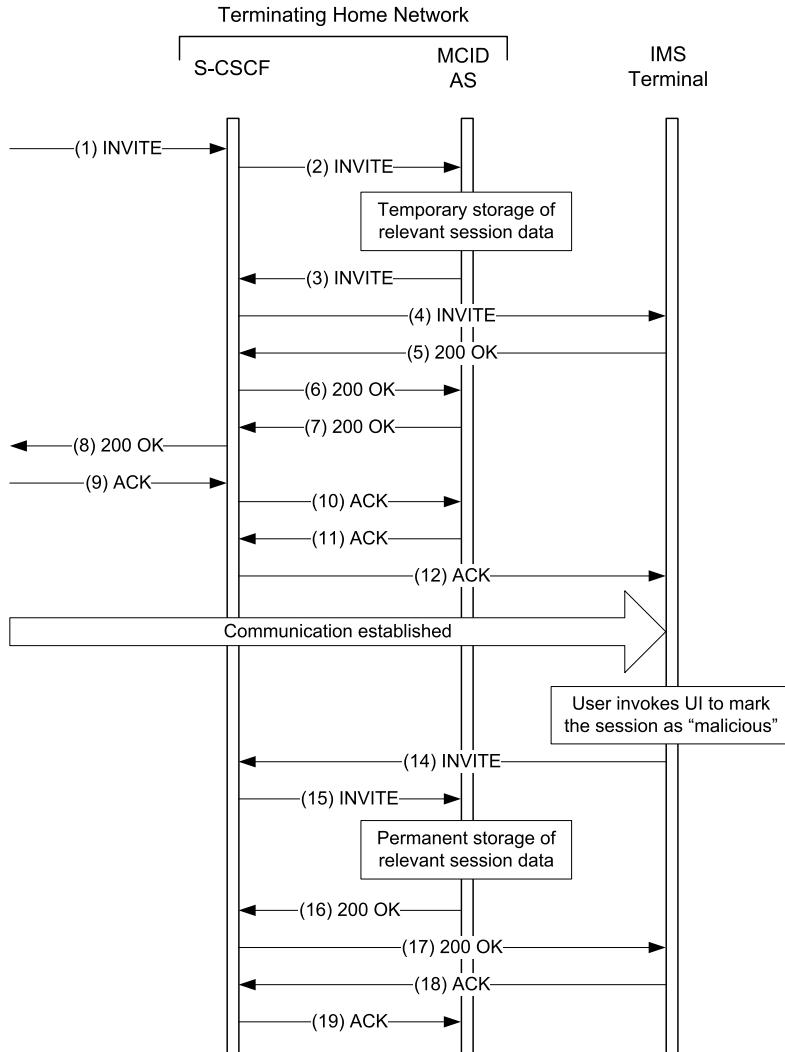


Figure 26.15: Malicious Communication Identification in temporary mode

Figure 26.15 depicts the call flow related to the temporary mode of MCID (only relevant nodes). The callee's S-CSCF receives an INVITE request (1) that, because of the initial Filter Criteria evaluation, is forwarded (2) to the MCID AS. The MCID registers the relevant data of the INVITE request into a temporary registry that will eventually be cleared some time after the call is finished. Then the MCID forwards the INVITE request (3) back to the S-CSCF, which routes it (4) to the callee (via at least a P-CSCF, not shown). The rest of the INVITE transaction is completed as any other transaction. Eventually, the two parties are engaged in a session. Then, the called party identifies the call as malicious and signals to the MCID AS

the situation. This is done with a re-INVITE request (14) addressed to the MCID AS. The MCID AS moves the call-related entry from the temporary log to the permanent one.

Each time an entry is created in the temporary or permanent log, the following data is recorded:

- (a) caller and callee's identities, retrieved from the P-Asserted-Identity header field and the Request-URI;
- (b) local time and date of the establishment of the session;
- (c) call diversion information retrieved from the History-Info header field;
- (d) Referred-By header field, if available;
- (e) unverified identities, namely From and To header fields;
- (f) caller contact information retrieved from the Contact header field.

Typically, in IMS originated sessions, the caller is authenticated by their home network, and their identity is conveyed in the P-Asserted-Identity header field. However, if the call is originated in the PSTN, the asserted identity does not need to be present in the signaling, although it is available upon request. If the MCID AS requires the caller's identity, then it sends an INFO request to the originator of the call, which presumably is an MGCF node, containing an MCID XML document. This document merely contains a request indication for MCID. The MGCF answers with a 200 (OK) responses and generates an IDR ISUP message, which is answered with an IDS ISUP message containing the network asserted identity of the caller. The MGCF copies that identity to a new XML MCID document and adds it to a new INFO request that is sent to the MCID AS. In this way, the MCID AS is able to log the caller's asserted identity.

26.12 Communication Hold (HOLD)

The HOLD service allows a user to suspend one or more media streams of a multimedia session and to resume them at a later time. The service can also include playing an announcement to the held user. HOLD is specified in ETSI TS 183 010 [132] and 3GPP TS 24.610 [63]. They are a straight implementation of the SDP offer/answer model specified in RFC 3264 [283] and already supported by 3GPP TS 24.229 [37].

Holding an ongoing session is achieved by sending a new SDP offer where each of the media streams to be held are marked as `sendonly` if they were previously bidirectional media streams. To resume the session, a new SDP offer is issued where each of the held media streams is marked with the default `sendrecv`.

The HOLD service also allows an AS to play some announcement or music to the held party. This is achieved by an AS that acts as a third-party call controller and replaces the existing session of one of the users with an AS-originated session that plays the announcement or music until the session is resumed. A call flow is shown in Figure 26.16, where the assumption is that a regular session is already established between two terminals. A user decides to put the remote party on hold, and their IMS terminal sends a re-INVITE request (1) where the media streams are set in “`sendonly`” mode by adding a `sendonly` attribute in SDP, to indicate that the terminal does not want to receive media. This INVITE request (1) traverses the P-CSCF (not shown in the figure), the S-CSCF, and eventually is

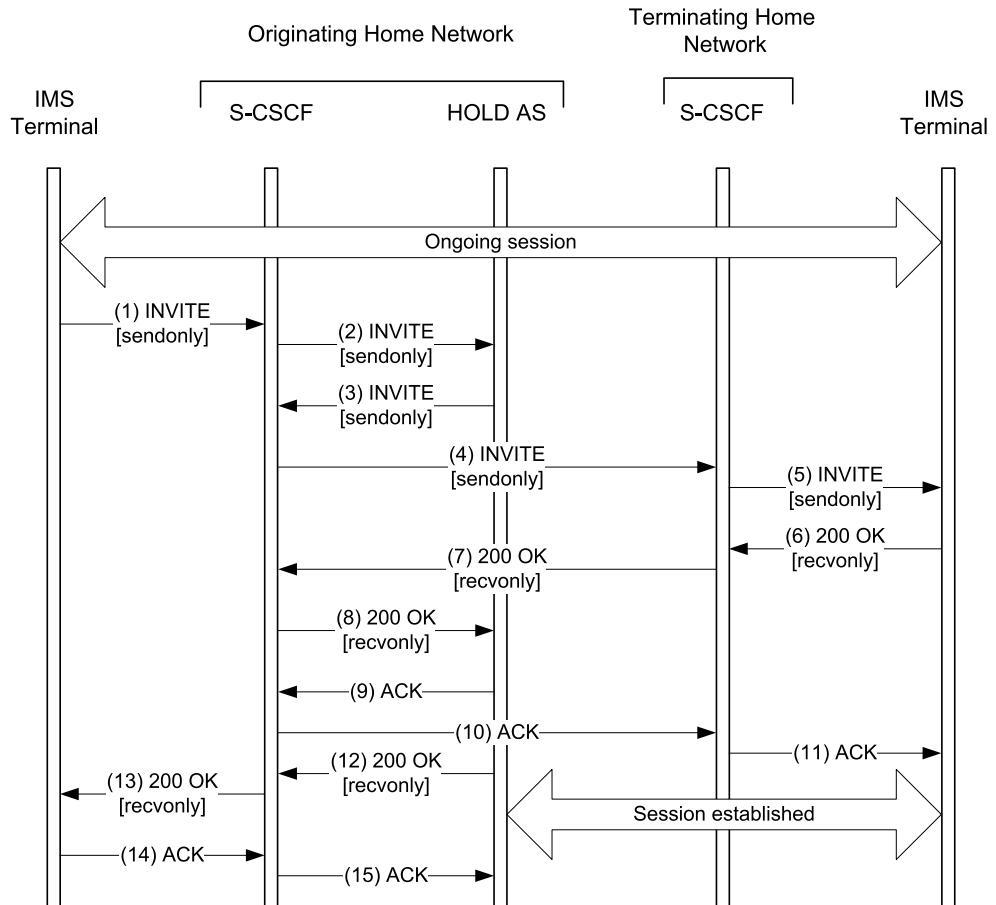


Figure 26.16: Music on HOLD service

received at the HOLD AS (2), which is acting as a B2BUA, so it separates the SIP leg from the other SIP leg towards the remote party. The HOLD AS creates another INVITE request (3), which also contains the `sendonly` attribute, since the AS will be sending music, but not receiving anything. This INVITE request (3) is sent back to the S-CSCF, for routing towards the remote party (4, 5). The remote IMS creates a 200 (OK) response (6) where the SDP includes `recvonly` attribute to indicate that it is receiving media. The rest of the re-INVITE transaction continues as per regular procedures. At this point in time there are two independent sessions, each established between each of the terminals and the HOLD AS. So, the HOLD AS is able to play music to one of the terminals.

Resuming the music on hold service consists of repeating the same flow depicted in Figure 26.16, but with the difference that the INVITE request (1 to 5) and its 200 (OK) response (6 to 8, 12, 13) contain SDP with the “`sendrecv`” direction.

26.13 Explicit Communication Transfer (ECT)

The ECT service allows a party who has an ongoing communication with a second party to transfer the communication to a third party, so that when the service is completed the first party has no ongoing communications and the second and third parties have an established session (or are in the process of establishing it).

The service, which is specified in ETSI TS 183 029 [136] and 3GPP TS 24.629 [67], is implemented with the help of an AS that controls the transfer of the communication. The service is largely based on the usage of the SIP REFER method (specified in RFC 3515 [306]), the SIP Replaces header field (specified in RFC 3891 [213]), and the SIP Referred-By header field (specified in RFC 3892 [307]).

ECT brings new terminology as detailed below.

Transferor. The party that initiates the transfer operation towards a third party and eventually removes itself from the current communication.

Transferee. The party of the initial communication who remains in the session once the transfer is completed.

Transfer target. The third party who eventually is engaged in a communication with the transferee.

There are two classes of ECT service.

Blind/Assured transfer. The transferor transfers the call to the transfer target without a previous communication.

Consultative transfer. The transferor puts the transferee on hold, initiates a communication with the transfer target, and eventually transfers the communication to the transfer target.

Figure 26.17 shows the sequence flow of the blind transfer mode of the ECT service. The figure shows the transferee (IMS terminal 1) and its ECT AS, the transferor (IMS terminal 2) and its ECT AS, and the transfer target (IMS terminal 3) and its ECT AS, but for the sake of simplicity, does not show all of the other CSCFs involved in the signaling. Let us assume that the transferee and the transferor are already engaged in a session. This session made their respective ECT ASes be part of the signaling path. The main problem of realization of the service consists of simulating the classical ECT charging model, which implies that in the ECT call setup, the transferee is charged for the call leg from the transferee to the transferor, while the transferor is charged for the call leg from the transferor to the transfer target. This classical charging model requires a mechanism where the transferor ECT AS is involved in the ECT call.

Assuming that the session is established, the transferor first puts the transferee on hold, for example, by invoking the HOLD service (see Section 26.12). Then the transferor sends a REFER request (1) that contains a Refer-To header field set to the transfer target URI. The transferor ECT AS receives the REFER request and obfuscates the transfer target URI so that the ECT session setup will visit this ECT AS. It also stores the original transfer target URI and the obfuscated URI for a temporary amount of time. The obfuscated URI inserted by the ECT AS resolves to itself, so that a future INVITE request will point to the transferor ECT AS. So, the transferor ECT AS sends the REFER request (2) with the obfuscated URI in the Refer-To header field, which eventually reaches the transferee ECT AS, which

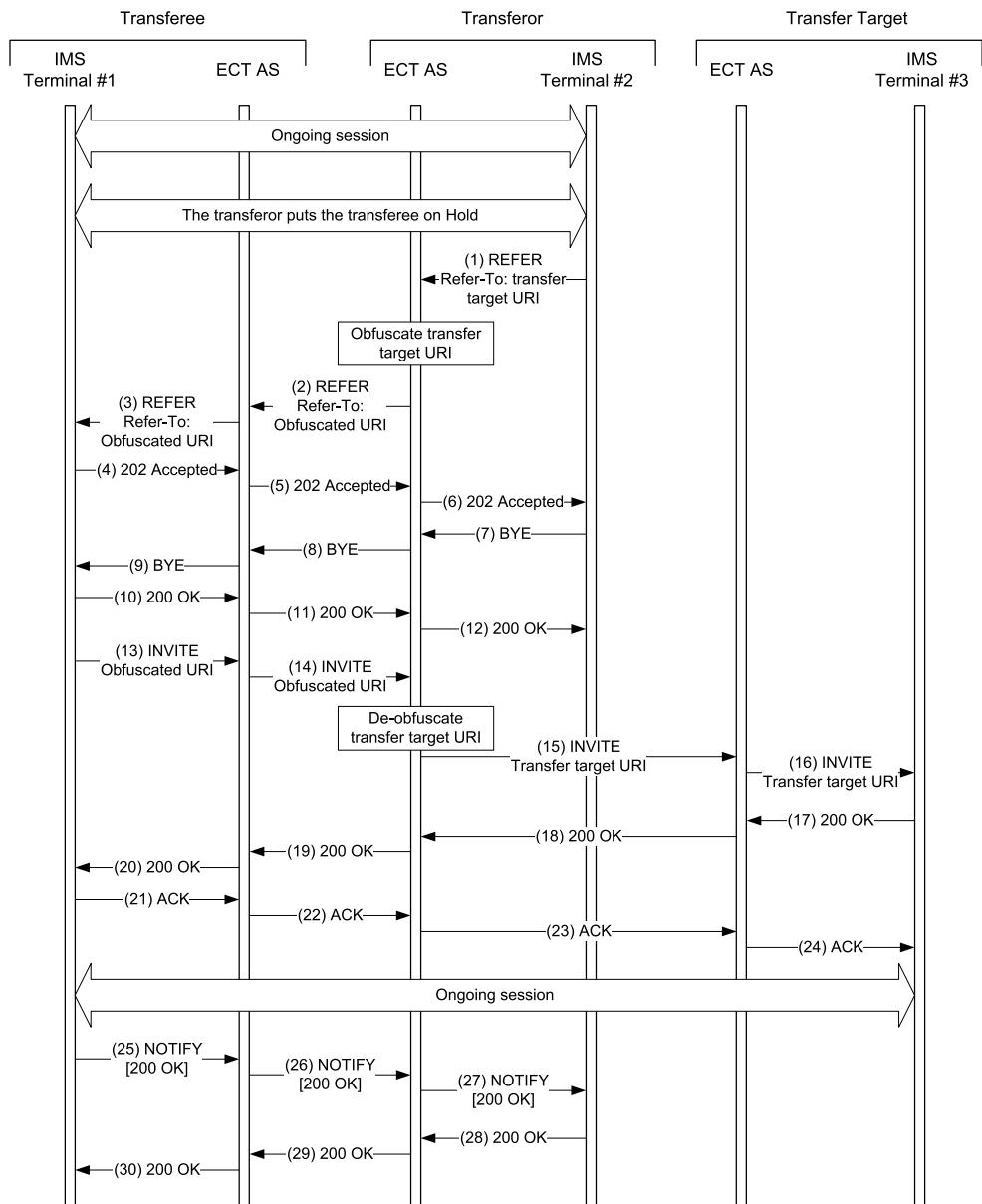


Figure 26.17: Explicit Communication Transfer: blind transfer

merely correlates the session and applies appropriate charging, and forwards the request to the transferee (3). The transferee acknowledges with a 202 (Accepted) response (4), according to the procedures of the REFER method.

In the meantime, the transferor tears down the session by sending a BYE request (7), all the way to the transferee (8, 9). On the other side, the transferee decides to accept the ECT service and sends an INVITE request (13) addressed (in the Request-URI) to the obfuscated URI included as previously received in the REFER request. Since this URI resolves to the transferor ECT AS, this AS will eventually receive the INVITE request (14). Then the transferor ECT AS replaces the obfuscated URI with the original transfer target URI and sends the INVITE request (15) to that target URI. This INVITE request will eventually be intercepted by the transfer target ECT AS, which forwards it (16) to the transfer target. The session setup procedure continues as in a regular call, and the session is eventually established between the transferee and the transfer target.

The transferee also sends a NOTIFY request (25) to the transferor. This is in accordance with the procedures of the REFER method, which creates an implicit subscription. The NOTIFY request (25) contains the result of the INVITE transaction (13 to 20), so it indicates that the transferee received a 200 (OK) response to the INVITE request.

Figure 26.18 depicts the first part of the flow that realizes the ECT service as a consultative transfer. The service assumes that two parties, the transferee (IMS terminal 1) and the transferor (IMS terminal 2) are engaged in a session. The session was set up via their respective ECT ASes. At some point in time, the transferor places the transferee on hold, by invoking the HOLD service (see Section 26.12). Then the transferee establishes a parallel session with the transfer target (IMS terminal 3), so that a consultation takes place. When the consultation is over, the transferor places the transfer target on HOLD and sends a REFER request (1) to the transferee. This REFER request (1) contains a Refer-To header field whose value is the transfer target URI. In addition to the transfer target URI, there are other encoded headers in the URI that, when encoded in the future INVITE request (7), will provide the transfer target with the information of the SIP dialog that such INVITE will have to replace. Therefore, these encoded header field are a Replaces header field that identifies the existing established dialog between the transferor and the transfer target, and a Require header field that requires the usage of the “Replaces” extension specified in RFC 3891 [213]. An example of the value of this Refer-To header field is shown in Figure 26.19.

As in the blind transfer mode, this REFER request (1) is received at the transferor ECT AS, which obfuscates the transfer target URI contained in the Refer-To header field. The new URI resolves to the transferor ECT AS. Then the transferor ECT AS forwards the REFER request (2, 3) to the transferee. The transferee now puts the transferor on HOLD and sends an INVITE request (7) to the obfuscated URI received in the Refer-To header field of REFER request (3). This INVITE request (7) also contains the encoded header fields in the URI, namely, a Replaces and a Require header field. The transferor ECT AS receives the INVITE request (8), restores the original transfer target URI, and sends the INVITE request (9, 10) to the correct transfer target. The session setup continues as per regular procedures. Eventually, a session is setup between the transferee and the transfer target.

Since the INVITE request (10) in Figure 26.18 contains a Replaces header field that lists the value of the initial SIP dialog that this second session is going to replace, the transfer target honors it, replacing the old SIP session with this new one. This further depicted in Figure 26.20. The transfer target sends a BYE request (19) to tear down the initial session,

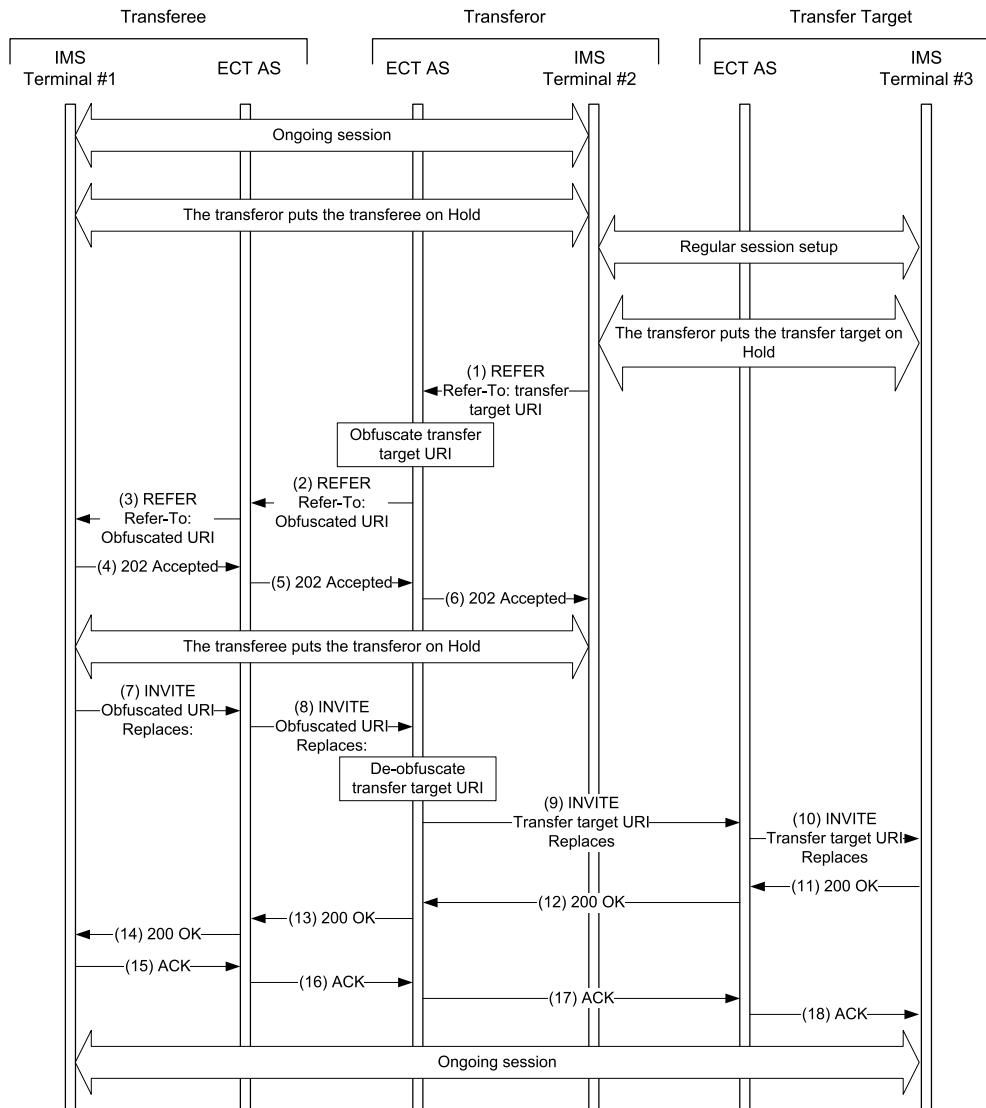


Figure 26.18: Explicit Communication Transfer: consultative transfer (part 1)

```
Refer-To: <sip:user1@home1.net?Replaces=29d8w2929%40bob.home2.net%3B
          to-tag%3D77B443%3Bfrom-tag%3D6472A6D&Require=replaces>
```

Figure 26.19: Refer-To header included in the REFER request (1)

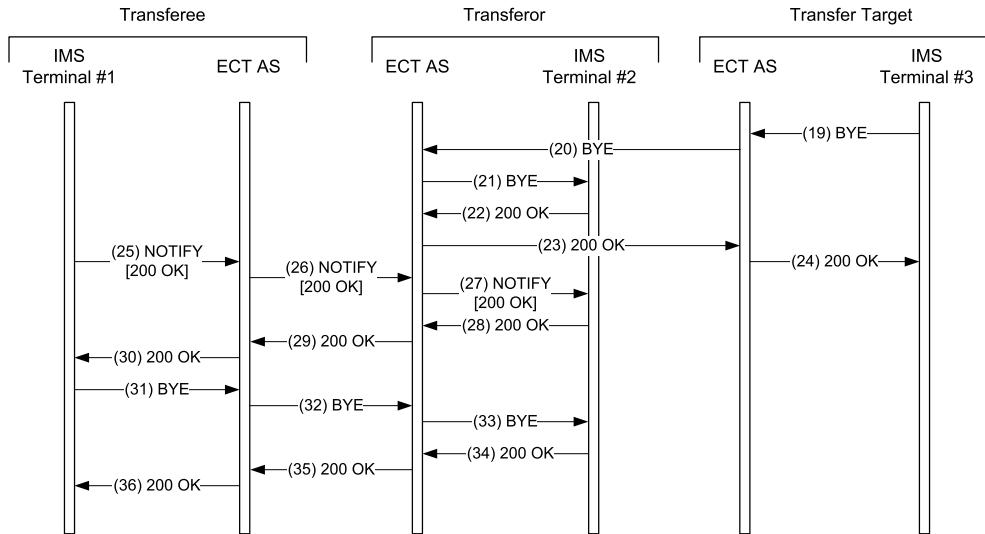


Figure 26.20: Explicit Communication Transfer: consultative transfer (part 2)

a BYE request that is forwarded (20, 21) to the transferor. The initial session, which was established between the transferor and the transfer target does no longer exist.

In addition, the transferee honors the implicit subscription created by the REFER request (3 in Figure 26.18) and sends a NOTIFY request (25) that contains the result of the INVITE transaction, therefore, it indicates that the INVITE request resulted with a 200 (OK) response. The NOTIFY request is further forwarded (26, 27) to the transferor. Last, but not least, the transferee tears down the session established between the transferee and the transferor, something that is done with a BYE request (31, 32, 33). The only remaining session at this time is established between the transferee and the transfer target.

26.14 User Settings in PSTN/ISDN Simulation Services

We have described earlier in this chapter how a user can invoke PSTN/ISDN simulation services that are similar but not the same as the PSTN/ISDN supplementary services. Some of these services require a previous configuration before operation can take place. For example, Communication Diversion services typically require the user to setup the URI to which incoming sessions will be forwarded; or Communication Barring, where the barring of incoming communications requires the user to define the identity of the caller for whom incoming sessions will be barred.

IMS provides users with the ability to activate, deactivate and configure or personalize most of their PSTN/ISDN simulation services. The *Ut* interface is used for this type of configuration and XCAP is the protocol running over that interface. Note that XCAP is just one of the mechanisms at our disposal to configure PSTN/ISDN simulation services in IMS. Other mechanisms can also be used, including providing a regular web page for the user to configure their services, or by using calls to special dialled strings. For example, in many countries, unconditional call forwarding is activated using the *21* dial string, call forwarding when busy is activated using *67*, and call forwarding on no answer with *61*.

However, this section is about the native way of configuring services in IMS, which uses XCAP. We have already described XCAP in Chapter 17. Now we have another application of XCAP for activating, deactivating, and configuring services.

User settings for PSTN/ISDN simulation services are defined in ETSI TS 183 023 [137] and 3GPP TS 24.623 [68]. The data is stored in a modular XML document. This document contains common parts, applicable to all of the services, and a collection of XML subdocument trees, each one representing a PSTN/ISDN simulation service. This approach allows us to store and then later manipulate the `simservs` XML document in a centralized server (e.g., an AS that provides all of the PSTN/ISDN simulation services), or in a distributed fashion (e.g., when the AS hosts a few PSTN/ISDN simulation services). The XML schema that defines the layout of the `simservs` XML document and its constraints is defined in ETSI TS 183 023 [137]. The subdocuments that are service-specific are defined in the specification that defines the service. This basically means that a `simservs` XML document can contain data pertaining to several simulation services at the same time.

ETSI defined a new XCAP application usage and allocated the AUID

`simservs.ngn.etsi.org`.

The new application usage defines the rules and conventions used in manipulating `simservs` XML documents. A `simservs` XML document has also an associated MIME type value of `application/simservs+xml`, which is visible in the `Content-Type` header field of HTTP requests and responses.

Users can then invoke XCAP operations (e.g., HTTP PUT, GET, DELETE) to activate and deactivate the service, and to configure the operational parameters of each service.

Chapter 27

Voice Call Continuity

This chapter is devoted to the Voice Call Continuity (VCC) feature in the IMS. VCC, which is specified in 3GPP TS 23.206 [27] and 24.206 [26], allows smooth transitions of voice calls executed over the IMS and Circuit-Switched (CS) calls and vice versa. The feature allows a double IMS/CS terminal to initiate or receive a voice call in either the IMS or CS domain and move it to the other domain during the duration of the call.

VCC considers voice calls exclusively. At the time of writing, 3GPP is standardizing an enhanced version of VCC that includes other types of media streams different than audio. This is known as the Multimedia Session Continuity (MMSC), which it is not considered in this chapter.

VCC fills an important gap in the transition of CS to IMS. Consider a user that is accessing the IMS over a narrow bandwidth packet data channel. For example, this narrow bandwidth channel can be a regular GSM cellular access over GPRS. The GPRS connectivity is used for signaling and, thus, for SIP signaling towards the IMS. Voice over IP (e.g., RTP packets) will not have the appropriate quality of service, mostly due to the delays and lack of bandwidth. However, CS connections are available, and working reasonably well. So, with the VCC feature enabled, the user can establish audio communications that use a regular CS bearer. The session is controlled with SIP in the IMS. Assume now that, at a later time, the user initiates another packet data IP-CAN access with higher bandwidth, perhaps Wireless LAN, or perhaps cellular High Speed Packet Access. He can then move the CS audio stream over the packet data IP-CAN, and perhaps enrich the session with video or some other data stream.

The goal of VCC is thus: maintain existing audio calls at any cost when the user moves between areas with different connectivity characteristics. This is achieved by handing the audio stream from CS to IMS or vice versa. The feature can be implemented transparently from the point of view of the user. However, the feature requires VCC-enabled dual-mode IMS/CS terminals and network support.

Because of the interaction between IMS and CS networks, this chapter assumes that the reader has basic knowledge of the GSM architecture and its evolution into 3G networks.

27.1 Overview of Voice Call Continuity

Figure 27.1 presents a high-level overview of the VCC architecture. The figure shows two VCC-enabled dual-mode IMS/CS terminals. Both can have access to packet-switched accesses, which leads to an IMS core network, and CS accesses, which leads to the CS core

network. A PSTN gateway composed of an MGCF and MGW interfaces the IMS domain with the CS domain. While Figure 27.1 shows two VCC-enabled terminals, the feature applies to each terminal in an independent way, so each terminal can use either of the CS or IMS domains in its own call leg without support from the remote terminal.

VCC is always implemented in the home network of the served user. VCC introduces the notion of *anchoring*, which consist of splitting the call into two call legs, and introducing a fixed point in the network where the two legs are anchored. This allows to hand one leg from one domain to the other without creating interferences with the remaining leg.

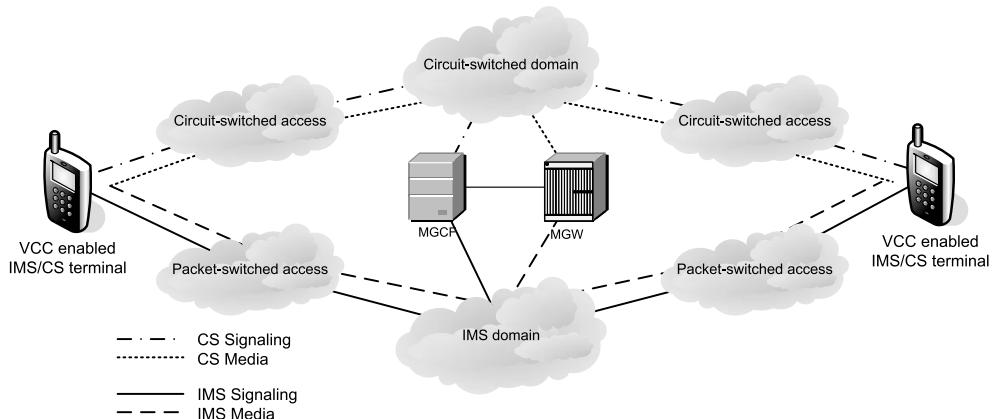


Figure 27.1: Overview of the VCC architecture

Figure 27.2 shows a possible usage of a call established between two VCC-enabled terminals. One terminal is using the IMS signaling and media to make the call; the other is using the CS signaling and media. At any point in time, either of the two terminals can transfer its own leg to the other domain without disrupting the communication.

Other more complex combinations are also possible. For example, a VCC-enabled terminal can initiate IMS signaling to establish a call that uses CS media. Then, later, the terminal can replace the CS audio stream with an RTP audio stream that uses the IMS.

VCC also introduces the notion of *routing numbers*, which are intermediate telephone numbers that are used to route the call appropriately via the nodes that provide the VCC function. The CS Domain Routing Number (CSRN) is a dynamic routing number that provides routing from the IMS to the CS domain. The IP Multimedia Routing Number (IMRN) is a routable number that points to the IMS from the CS point of view. Effectively, the IMRN is a Tel URL that is considered as a Public Service Identity (PSI) in the IMS.

VCC also introduces the notion of a VCC Domain Transfer Number (VDN), which is an E.164 number that the terminal can use to request a domain transfer from IMS to the CS domain. The VDN is not a number that resolves to any of the VCC nodes; it is merely a regular E.164 number whose semantics is “activate a domain transfer”. There is also a counterpart VCC Domain Transfer URI (VDI), which is a SIP URI that the UE can use to perform a domain transfer from the CS domain to the IMS domain. Like the IMRN, the VDI is also a PSI. The VDI has the semantics of “activate a domain transfer”. The VDI and VDN are assumed to be pre-provisioned to the terminal.

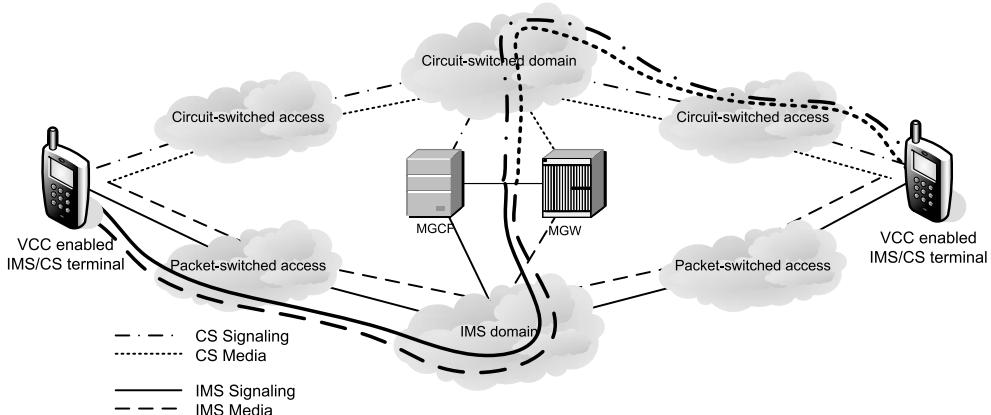


Figure 27.2: Example of signaling and media path with VCC-enabled terminals

27.2 VCC Architecture

The VCC feature introduces the concept of a *VCC-enabled terminal*, which is a terminal that has access to traditional CS voice and IMS applications. The VCC-enabled terminal implements domain selection policies for both originating calls and domain transfers. This policy is used when selecting a domain for originating calls. On the terminating side, it controls the user preferences for terminating calls. The VCC-enabled terminal stores the VDN and VDI to be used for domain transfer execution. In addition, a VCC-enabled terminal implements procedures for transferring a leg from circuit-switched to IMS or vice versa.

The VCC-enabled terminal is configured with an E.164 number, so that it is reachable from the CS domain, as well as a TEL URL and a SIP URI to be reachable from the IMS side. Thus, there is a correlation between these numbers in different domains.

VCC also introduces the concepts of the *VCC application*, which is a collection of functional entities that provide all of the required functionality for selecting the domain (CS or IMS) for terminating calls and to assist the VCC-enabled phone for transferring a leg from one domain to the other. The VCC application also provides the anchoring point to enabling transfer of domains during active calls.

Figure 27.3 shows the general architecture of the VCC application and the related interfaces. The VCC application is composed of four main functions: the Domain Transfer Function (DTF), the Domain Selection Function (DSF), the CS Adaptation Function (CSAF), and the service for Customized Applications for Mobile network Enhanced Logic (CAMEL service). The interfaces between these four functions are not standardized, so implementations either implement proprietary interfaces or combine several functions in the same node.

The Domain Transfer Function (DTF) is responsible for executing the transfer of an access leg from the CS domain to the IMS domain or vice versa. From the point of view of SIP and IMS, the DTF acts as an Application Server (AS) configured as a Back-to-Back User Agent (B2BUA) operating as a third-party call controller. In addition, the DTF maintains policies related to the transfer of domains. The DTF interfaces the S-CSCF via the *ISC* interface, the I-CSCF via the *Ma*, and the HSS via the *Sh* interface.

The Domain Selection Function (DSF) is involved in terminating calls. The DSF selects the domain over which the terminating leg of an incoming call will be delivered. To take this

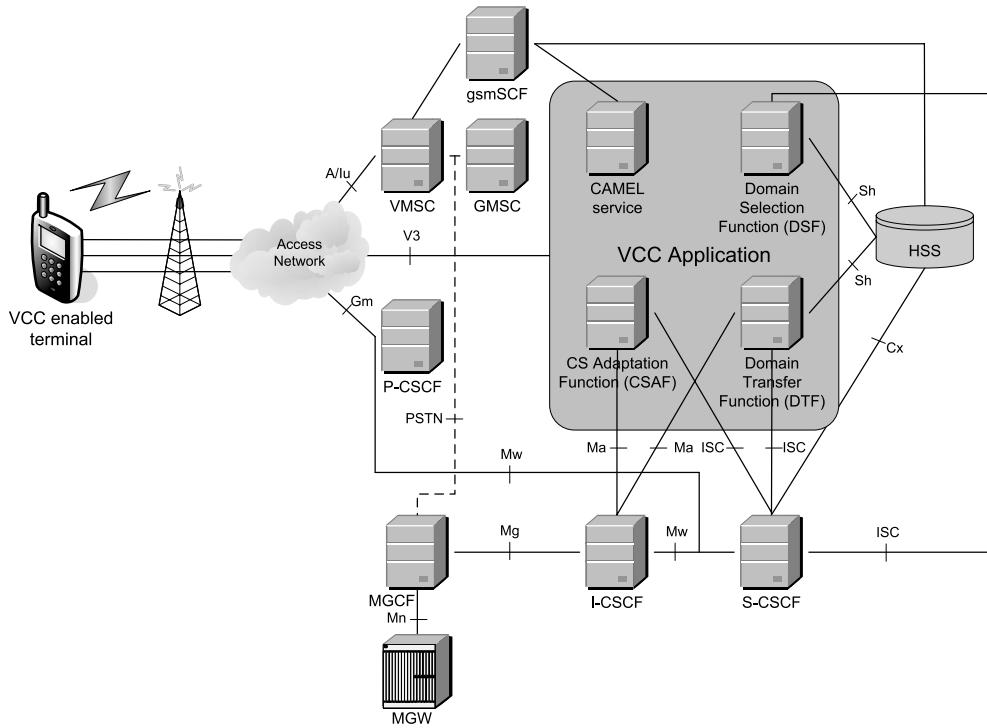


Figure 27.3: VCC architecture

decision, the DSF considers operator policy and user preferences. However, it is also helped by the user's registration status in the IMS and CS domains, so, for example, if a user is not registered to the IMS but it is to the CS domain, the call will be delivered over the CS domain. The DSF interfaces the S-CSCF via the ISC interface and the HSS via the Sh interface.

The CS Adaptation Function (CSAF) is responsible for allocating and de-allocating the IMRN for CS originated calls. The CSAF communicates call related data from CS to IMS (such as called an calling party numbers). The CSAF acts as a SIP proxy for CS originated calls and CS legs established for domain transfer to CS. The CSAF acts as a SIP User Agent towards the IMS. The CSAF interfaces the S-CSCF via the ISC interface and the I-CSCF via the Ma interface.

CAMEL is a set of mobile networks standards for providing services in CS networks. CAMEL is based on the Intelligent Networks (IN) standards. One of the nodes of the CAMEL architecture is the gsmSCF, which is able to monitor the status of a call and provide certain services. Well, in VCC, the CAMEL service function provides an alternative for routing calls using CAMEL services. The functions of the CAMEL service overlap those of the CSAF, and include allocating IMRN for routing CS calls to IMS, communicating call related data from CS to IMS (such as called an calling party numbers). However, it also provides unique functions, such as enforcing the CS redirection policy. The CAMEL service interfaces the CAMEL gsmSCF via an unspecified interface. This mostly indicates that the CAMEL service is essentially an extension to the existing gsmSCF.

In addition to the interfaces between the individual functions of the VCC application, a VCC-enabled terminal implements the *V3* interface towards the VCC application (or any of the individual functions). The *V3* interface is used for configuration settings and uses XCAP. We described XCAP in Chapter 17.

Other relevant nodes for the VCC service include a standard PSTN/CS gateway, further composed of an MGCF and MGW. In addition, a visited MSC has also a role when the access leg traverses the circuit-switched domain. In the IMS domain, a P-CSCF is also involved when the access leg uses the IMS.

27.3 Registration

There are no requirements for the VCC terminal with respect registrations. The IMS side of a VCC-enabled terminal registers to the IMS as per regular procedures (see Section 5.5). In addition, the CS side of the VCC-enabled terminal performs the regular attachment to the network.

The VCC application needs to become aware of the IMS registration of the VCC-enabled terminal. This is done by configuring an initial Filter Criterion that triggers a third-party registration towards the VCC application when the S-CSCF receives a regular REGISTER request from the VCC enabled terminal. In addition, the VCC application can subscribe to the *reg* event package at the S-CSCF, which keeps the VCC application informed whenever the VCC-enabled terminal re-registers or deregisters (see Section 5.6 for a description of the *reg* event state package). An alternative to notifications of the *reg* event package is to use the *Sh* interface to subscribe to the user's registration state at the HSS.

27.4 Call Origination and Anchoring

Prior to executing a domain transfer of an ongoing voice call, the call must have been setup (either in the IMS or the CS domain), and it has to be anchored, so that it can be transferred to the other domain at a later stage. This section presents call flows for the originating side indicating the involved entities in the call anchoring and revealing the details of it. The flows only show the relevant nodes involved in the call anchoring for the originating VCC-enabled terminal, so, for example, the originating P-CSCF is never shown, neither are any of the terminating nodes.

27.4.1 IMS Originated Call Leg

Figure 27.4 shows the call flow of an IMS originated call anchored in the network for enabling a potential domain transfer at a later stage. When the user initiates the call, the VCC-enabled terminal performs a domain selection that takes into consideration the access network availability and user preferences. In this example, the result of the domain selection leads to setting up the call via the IMS domain. Therefore, the VCC-enabled terminal sends a regular INVITE request (1) that traverses the P-CSCF (not shown in the figure) and other IMS nodes until it eventually reaches the S-CSCF allocated to the user in the originating home network. This S-CSCF evaluates the initial filter criteria and as a result of that evaluation, it forwards the INVITE request (3) to the DTF, as per regular procedures over the *ISC* interface. Of importance, the S-CSCF adds a *Route* header field whose value is a URI that resolves to the same S-CSCF and contains (e.g., in the user part of the URI) a unique identifier of the dialog.

The DTF receives the INVITE request (3), decides to anchor the call, and acts as a B2BUA AS, so it ends up the signaling towards the VCC terminal. On the other side, the DTF initiates a new originating call, so it creates an INVITE request (5) where it keeps the received Route header field value that contains the dialog identifier. The DTF can, however, modify the [From, To, Call-ID, and CSeq header fields, since they are new on this leg. Then the DTF sends it to the S-CSCF which is routed according to the originating procedures. This INVITE request (5) is received at the S-CSCF, which evaluates the filter criteria for providing originating services, continuing from the next trigger point. Eventually, the S-CSCF forwards the INVITE request (7) to its destination.

The session setup continues as per regular procedures, and eventually is setup. Note that the signaling is split into two call legs. There is a call leg established between the VCC terminal and the DTF (via the S-CSCF) and another call leg established between the DTF and the called party (via the S-CSCF as well). This anchoring and separation of call legs enables a future domain transfer.

27.4.2 CS Originated Call Leg using CAMEL Services

Assume now that a user is placing an audio call. Their VCC-enabled terminal determines, based on the available access technology and user preferences, that the CS domain should be used. This flow sequence is shown in Figure 27.5. The VCC-enabled terminals sends a SETUP message (1) to its VMSC. This message includes the list of supported codecs and the called party number. The VMSC receives the SETUP message (1) and evaluates the origination triggers, resulting in sending a CAMEL IDP message (2) to the gsmSCF.

The CAMEL IDP message (2) contains the calling and called party number, among other parameters. The gsmSCF determines whether the call should be feasible of the VCC service, and decides to anchor the call and route it via the IMS. The CAMEL service function allocates an IMRN to be used in the case of a domain transfer and provides it back to the gsmSCF. The IMRN is key to routing the call through the IMS and via the CSAF. Effectively, the IMRN is a PSI that is configured to be routed to the CSAF.

Continuing with the sequence flow, the CAMEL service interacts (3) with the CSAF (via yet another unspecified interface), so that the CSAF is aware of the call itself and the allocated IMRN. Then the CAMEL service in combination with the gsmSCF replies to the CAMEL IDP message (2) with a CAMEL CONNECT message (4) that includes a Destination Routing Address set to the allocated IMRN. The CAMEL CONNECT message (4) is received at the VMSC, which then routes the call to the received IMRN by performing regular CS routing.

The network is configured so that, from the CS domain, the IMRN is routed to a PSTN Gateway that interfaces the CS domain with the IMS domain. So, the VMSC sends an ISUP IAM message (5) that is routed to an MGCF. The MGCF interacts (6) with the Media Gateway via the H.248 protocol and then creates an INVITE request (7) addressed in the *Request-URI* and To header field to the IMRN. The INVITE request contains a Session Description Protocol whose audio stream points to a connection IP address in the MGW. Then the MGCF sends this INVITE request (7) to the I-CSCF, which applies regular routing procedures for PSI routing and eventually forwards it (9) to the CSAF.

The CSAF decides to anchor the call and acts together with the DTF with which it interacts (11) to provide the proper anchoring. The CSAF/DTF acts as a SIP B2BUA. The CSAF/DTF then retrieves, in particular, the called party number that was received in the CAMEL IDP message (2) and later forwarded due to interaction (3) to the CSAF. The CSAF/DTF creates an INVITE request (12) addressed in the *Request-URI* and To header

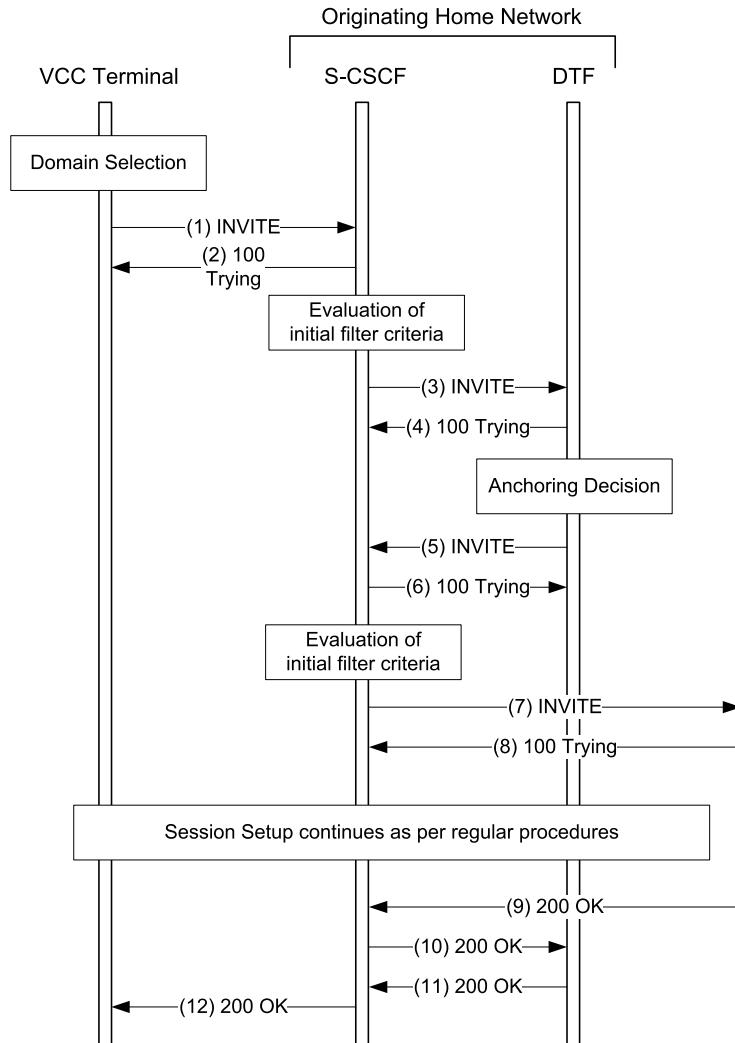


Figure 27.4: VCC anchoring: IMS originated call leg

field to the called party number. The Contact header of this INVITE request (12) is set to the SIP URI of the DTF.

The DTF then forwards the INVITE request (12) to the S-CSCF of the originating user, which applies regular routing procedures and forwards the INVITE request (14) to the next hop towards its destination. The rest of the session setup continues as per regular procedures. When the session is setup, there are two call legs: a CS call leg involving the VCC terminal, the VMSC plus the gsmSSF, the CAMEL service plus the gsmSCF, the CSAF, and the MGCF plus the MGW; and a IMS call leg involving the MGCF plus the MGW, the I-CSCF, S-CSCF, the CSAF, and the DTF. The IMS leg is anchored in the CSAF/DTF which is acting as a SIP B2BUA AS.

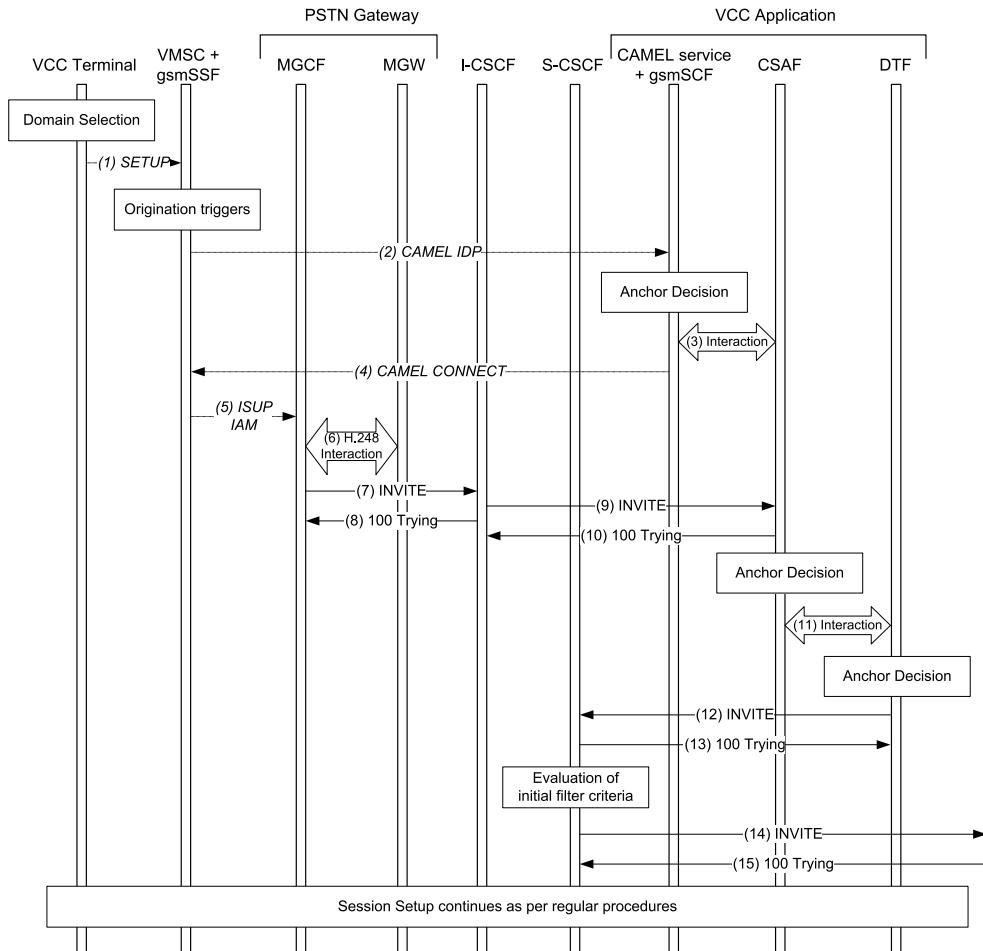


Figure 27.5: VCC anchoring: CS originated call leg using CAMEL services

27.4.3 CS Originated Call Leg using CAMEL and ISUP Call Diversion

The call flow shown in Figure 27.5 has a small variation, which is shown in Figure 27.6. According to Figure 27.6, when the CAMEL service in the VCC application receives the CAMEL IDP message (2), rather than contacting the CSAF that would provide the IMRN, the CAMEL service generates the IMRN. Then the CAMEL service replies to the CAMEL IDP message (2) with a CAMEL CONNECT (3) that contains the destination routing address set to the IMRN (as in the call flow of Figure 27.5) but now also contains the Original Called-Party ID, the redirecting Party ID, and some Redirection information. All of this extra information is conveyed in the ISUP IAM message (4).

The MGCF receives the ISUP IAM message (4) and creates an INVITE request (6) which is similar to that in the call flow of Figure 27.5, but now also includes a History-Info header field listing the additional redirection data, namely, the original called number and the redirecting number received in the ISUP IAM message (4).

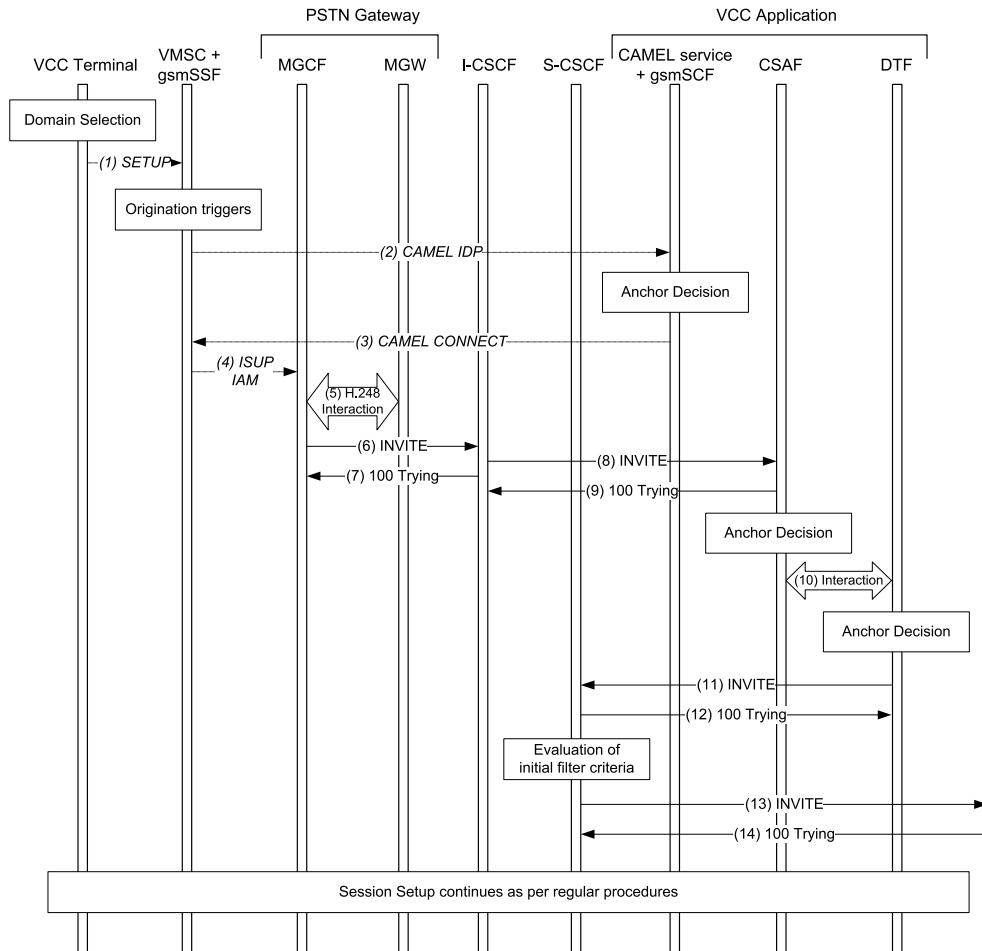


Figure 27.6: VCC anchoring: CS originated call leg using CAMEL services and ISUP call diversion

When the CSAF receives the INVITE request (8), it will extract the number included in the History-Info header and pass it to the DTF, so that the INVITE request (11) is addressed to that original destination number and does not include that entry in the History-Info header field.

27.5 Call Termination and Anchoring

Prior to executing a domain transfer on the terminating access leg, the call has to be anchored, either in the IMS or the CS domain. This section presents the VCC related terminating call flows indicating the involved entities in the call anchoring for call made to VCC-enabled terminals. As in the originating case, the flows only show the relevant nodes involved in the call anchoring, so, for example, the P-CSCF is never shown, neither are any of the terminating nodes.

27.5.1 IMS Terminated Call Leg

Figure 27.7 presents a terminating call flow when the call is anchored in the IMS. In the terminating home network, the S-CSCF allocated to the called party receives a regular INVITE request (1), whose destination address is included in the *Request-URI*, so there is no information related to VCC in this request. The S-CSCF evaluates the initial filter criteria, and as a result of this evaluation, the request is routed to the DTF function that is part of the VCC application. The S-CSCF, as per regular procedures, adds a *Route* header field that contains the SIP URI of the S-CSCF, so that the request can be routed back to the S-CSCF.

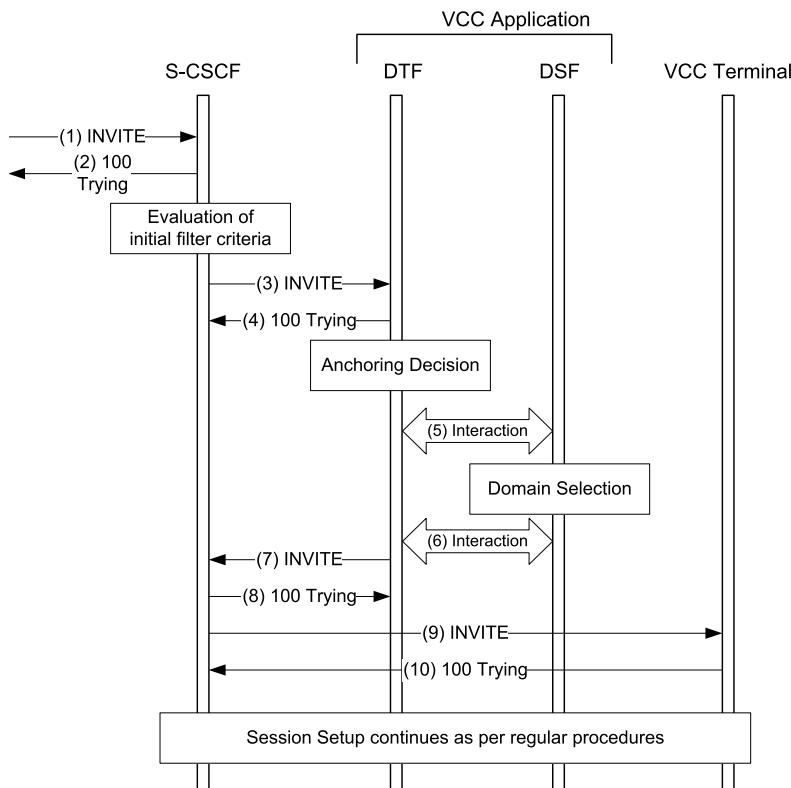


Figure 27.7: VCC anchoring: terminating call leg uses the IMS access leg

The DTF receives the INVITE request (3) and takes the decision of anchoring the call based on the local policy. Then the DTF interacts (5) with the DSF over a non-specified protocol. The DSF decides to use the IMS as the domain for the access network technology, based on local policy, user preferences, and registration state. The DSF then communicates that decision (6) back to the DTF. The DTF acts as a SIP B2BUA in order to anchor the call, so it re-creates a new INVITE request (7) and sends it back to the S-CSCF, honoring the *Route* header field value that was included in the INVITE request (3). The S-CSCF applies regular routing procedures to the INVITE request (7), such as replacing the destination address included in the *Request-URI* with the contact information learned during registration. Then the S-CSCF forwards the INVITE request (9) to the VCC terminal. The session setup proceeds as per regular procedures. At this point, the call is anchored in the VCC application.

27.5.2 CS Terminated Call Leg

Figure 27.8 presents the call flow of a call that arrives at the called party IMS network. The call is routed to the VCC application, which anchors the call and decides to route it via the CS access leg.

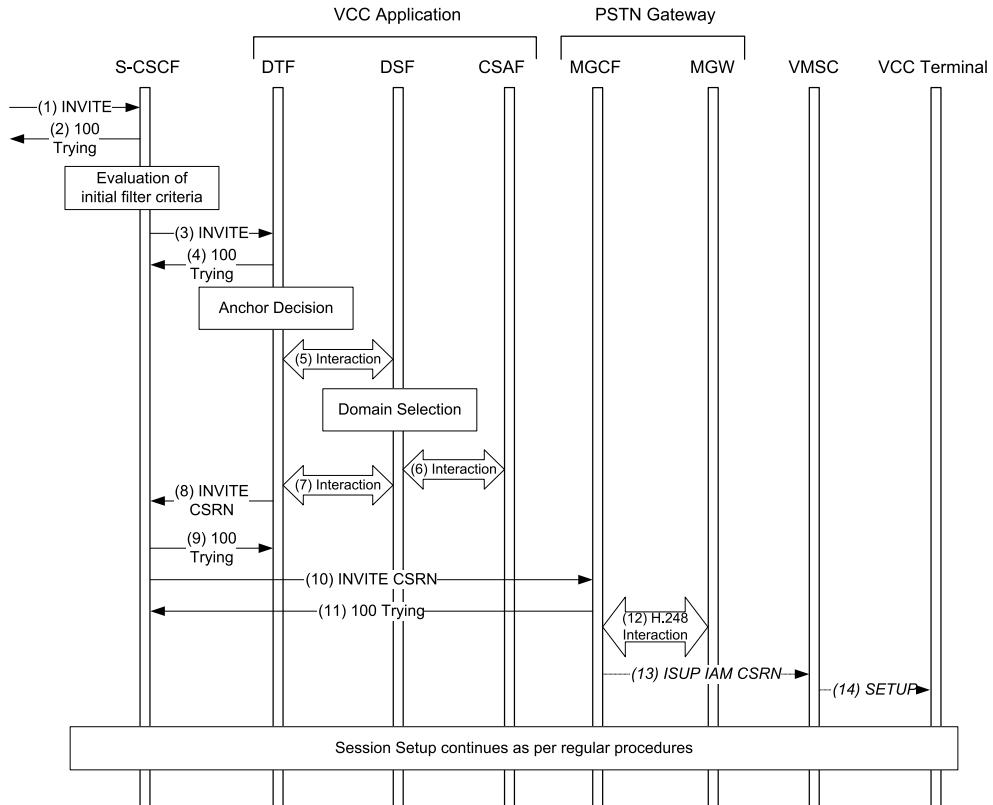


Figure 27.8: VCC anchoring: terminating call uses the CS access leg

At some point in time, the called party's S-CSCF receives a regular INVITE request (1). The S-CSCF evaluates the initial filter criteria and as a result of it, the S-CSCF sends INVITE request (3) to the DTF function of the VCC application, not before adding a Route header field that contains the S-CSCF own SIP URI, to be used at a later stage.

The DTF decides to anchor the call based on local policy. Then it interacts (5) with the DSF, which decides to select the CS domain as the access leg for the terminating call, based on local policy, user preferences, and registration status. The DSF also interacts (6) with the CSAF and they determine the CSRN, which is a dynamic routing number that is able to route the call to the VCC terminal. Then the DSF interacts (7) with the DTF. The DTF, acting as a SIP B2BUA re-creates a new INVITE request (8) addressed to the CSRN.

Honoring the Route header field of the INVITE request (3), the DTF sends the INVITE request (8) back to the S-CSCF, which applies regular routing procedures. Since the destination address included in the *Request-URI* is effectively a telephone number in the

circuit-switched domain, the INVITE request (10) is routed to an MGCF (perhaps via one or more BGCFs, not shown in the Figure 27.8).

The MGCF interacts (12) with the MGW and then sends an ISUP IAM message (13) where the called party is the CSRN. The VMSC maps the CSRN to the called party, so it sends a SETUP message (14) to the VCC terminal. This is a regular SETUP message, so it does not contain any information related to VCC. The session setup continues as per regular procedures. At this point in time, there is a circuit-switched bearer established between the VCC terminal and the MGW, and an IMS (RTP) bearer established between the MGW and the originating point. In addition, the call is anchored in the DTF, something that allows a potential domain transfer at a later stage.

27.5.3 CS Originated Call is Terminated in the IMS using CAMEL

Figure 27.9 presents the call flow of a call attempt that is received at the terminating home network over the CS domain. The VCC application determines that the call is delivered over the IMS domain, so it is routed to an PSTN/CS gateway that converts it to the IMS domain, where the VCC terminal is eventually contacted.

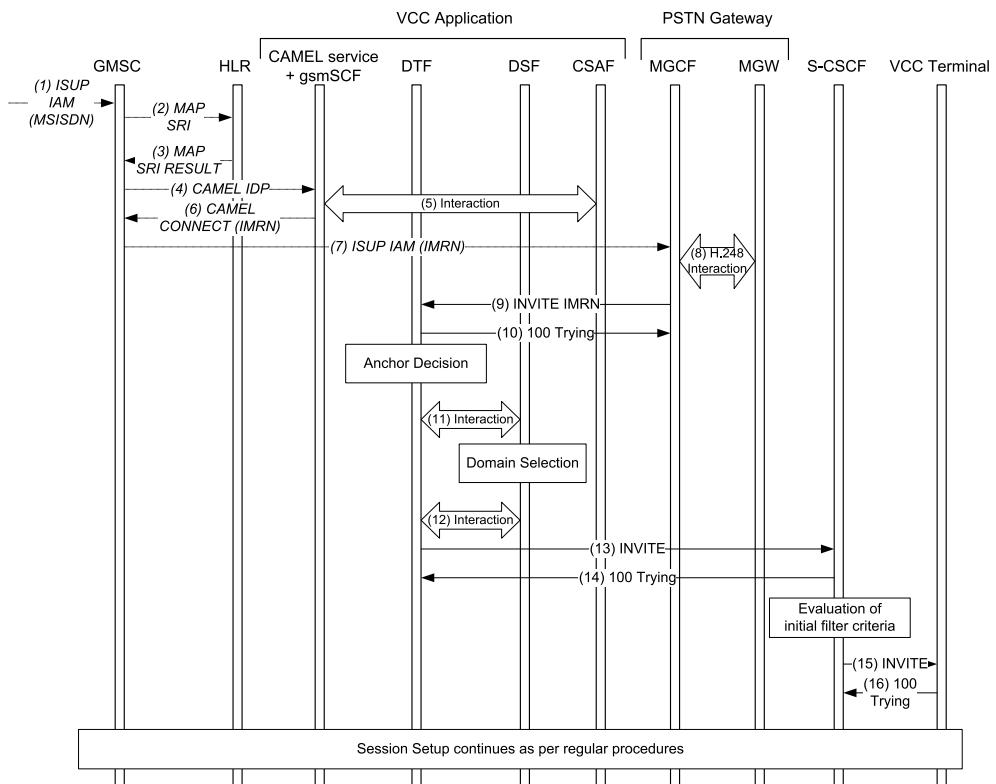


Figure 27.9: Call coming from the CS domain is terminated in the IMS; CAMEL is used

The call flow starts when a Gateway Mobile Switching Center (GMSC) receives and ISUP IAM message (1) addressed to the regular phone number (MSISDN) of the user.

The GMSC interrogates the Home Location Register (HLR) using the Mobile Application Part (MAP) protocol, to be precise, a Send Routing Information (SRI) message (2). The HLR returns a MAP SRI RESULT message (3) that contains the Terminating CAMEL Subscription Information (T-CSI). The T-CSI includes all of the required CAMEL information for providing the VCC service, including the address the of the gsmSCF allocated to the user.

The GMSC issues a CAMEL IDP message (4) that is received at the gsmSCF and the CAMEL service, which is part of the VCC application. Those interact with the CSAF, which provides the IMRN allocated to the user. The IMRN is returned in a CAMEL CONNECT message (6) back to the GMSC.

The GMSC then continues the call by sending an ISUP IAM message (7) now addressed to the IMRN received from the VCC application. Since the IMRN resolves to the IMS, regular CS routing procedures cause the ISUP IAM message (7) be received at an MGCF that interfaces to the IMS. The MGCF interacts with (8) the MGW to select an IP address and port number for media, and creates an INVITE request (9) addressed to the IMRN. The IMRN is effectively a PSI, so the MGCF sends the INVITE request (9) first to an I-CSCF (not shown in the figure), which routes it to the DTF.

The DTF decides to anchor the call and interacts with the DSF, which performs the domain selection, and decides to use the IMS for delivering the call. The DTF maps the IMRN with a SIP URI or TEL URL of the user, and then acting as a B2BUA, creates an INVITE request (13) addressed to that identity. The S-CSCF allocated to the user receives it and applies regular procedures, such as evaluating the initial filter criteria. It also applies regular routing procedures, such as replacing the *Request-URI* with the contact information of the user. Then the S-CSCF forwards the INVITE request (15) to the VCC terminal via a P-CSCF (not shown in the figure). The session is then completed as per regular procedures.

27.5.4 CS Originated Call is Terminated in the CS Domain

Figure 27.10 presents the call flow of a CS originated call that is routed to the IMS domain and, in particular, to the VCC application for anchoring, and eventually routed via the CS access domain to the VCC terminal.

According to Figure 27.10, a Gateway MSC receives an ISUP IAM message (1) address to the regular telephone number of the user. The GMSC queries the HRL with a MAP SRI message (2). The HLR determines the IMRN allocated to that user and returns it in a MAP SRI result message (3). Then the GMSC sends an ISUP IAM message (4) addressed to the IMRN. Since the IMRN is a telephone number in the IMS domain, the CS routes point to an MGCF that interfaces the IMS. The MGCF creates an INVITE request (6) addressed to the IMRN and sends it to the I-CSCF. The IMRN is a PSI, so the I-CSCF applies routing procedures for PSIs and forwards it to the CSAF. The CSAF interacts (8) with the DTF, which decides to anchor the call. Then the DTF interacts (9) with the DSF, which decides to select the CS domain for delivering the call, derives the CSRN, and communicates these decisions back to the DTF (10). The DTF then generates an INVITE request (11) addressed to the CSRN and sends it to a S-CSCF. Since the CSRN is a number exclusively allocated in a CS network, the S-CSCF forwards the INVITE request (13) to an MGCF via a BGCF (not shown in the figure). Note that this MGCF and its corresponding MGW need not necessarily be the same as those involved with messages 4 to 6. However, for the sake of simplicity, Figure 27.10 shows a single MGCF and a single MGW. After some interaction (15) with its MGW, the MGCF sends an ISUP IAM request (16) addressed to the CSRN. This is routed

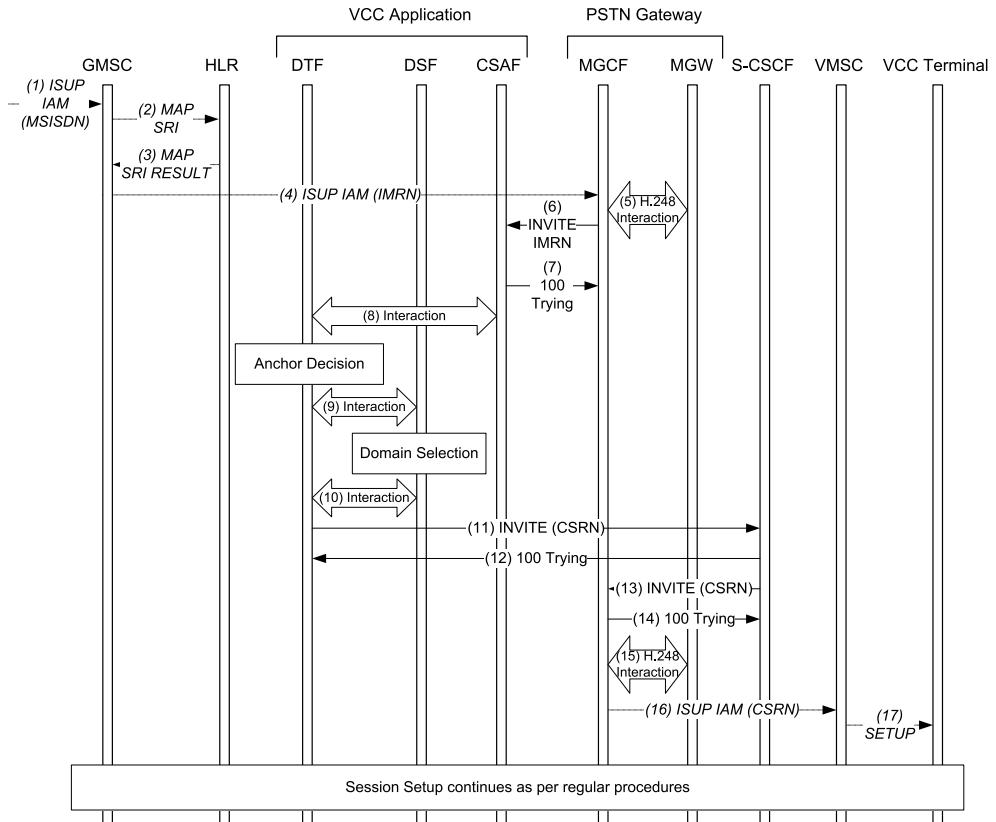


Figure 27.10: Call coming from the CS domain is routed to the VCC in the IMS and then terminated in the CS domain

to the visited MSC allocated to the VCC terminal, which sends the SETUP message (17) to establish the call. The reset of the session setup continues as per regular procedures.

Once the session is setup, there is a CS leg established in the core network until an MGCF is reached, the call then gateways to the IMS and is anchored in the DTF, and then it is routed back to the CS domain.

27.6 Domain Transfer

So far we have seen how audio calls are anchored and delivered via the CS or IMS domain, either in the originating or terminating side. Let us take a look now at the domain transfer scenarios. These take place when the real VCC service is invoked.

27.6.1 Transfer from the CS Domain to the IMS Domain

Figure 27.11 shows the call flow of a call where one of the users is initially anchored to the IMS. At some point in time during the call, the VCC terminal determines that a transfer of domain is needed. This might be caused due to a number of factors, for example, due to the

availability of a high bandwidth radio bearer, such as Wireless LAN. The VCC terminal sends an INVITE request (1) via the P-CSCF (not shown) to the S-CSCF. This INVITE request is addressed, in the *Request-URI* to the VDI, which is a PSI that resolves to the DTF. The S-CSCF evaluates the initial filter criteria and routes the INVITE request (3) to the DTF.

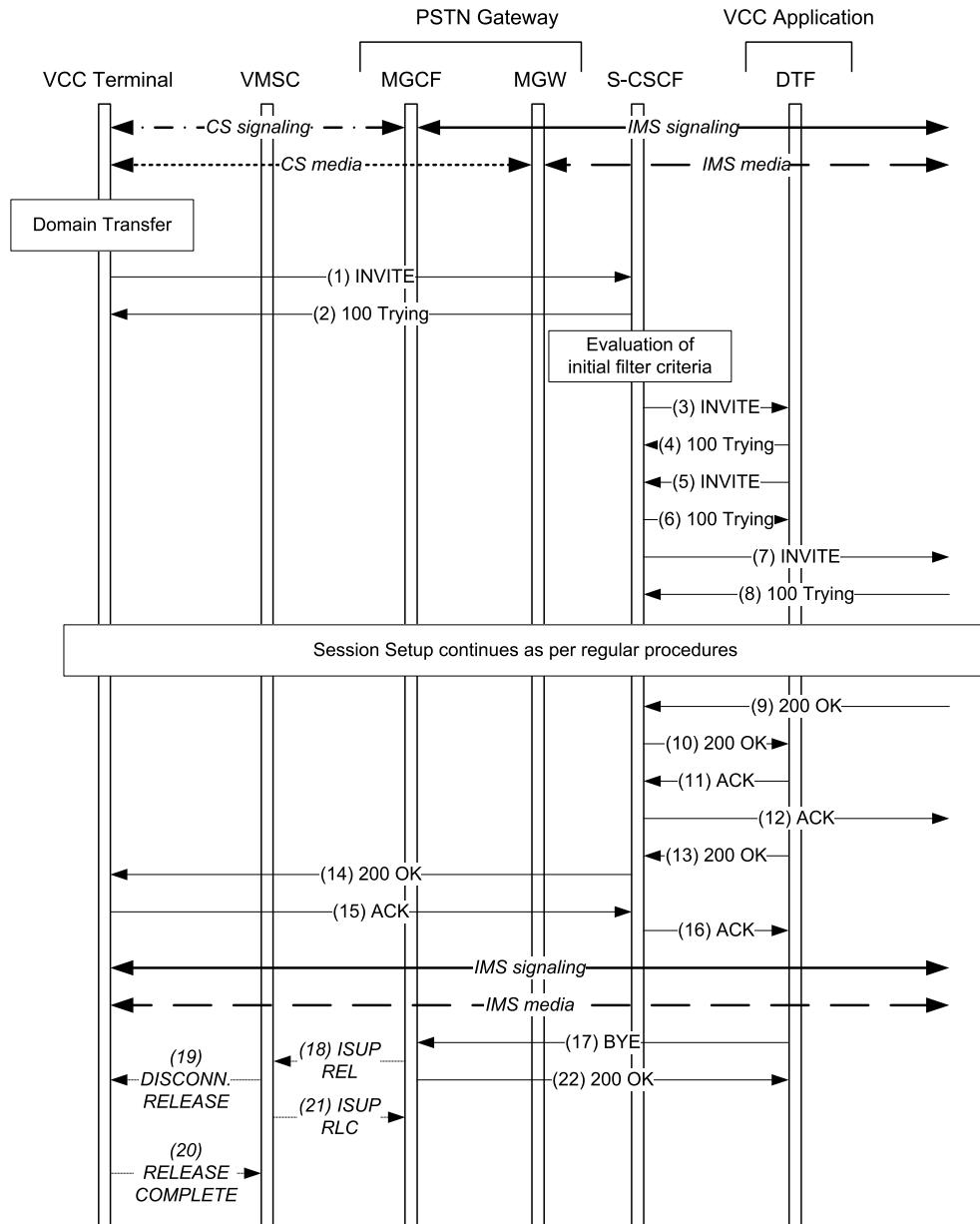


Figure 27.11: Domain transfer of an ongoing call from the CS to the IMS domain

The DTF is an AS acting as a B2BUA. On the other leg the DTF has an existing SIP dialog established with the remote party. The DTF then sends a re-INVITE request (5) towards the remote party. The idea is to inform the remote party of the change of access leg of the VCC terminal from the CS to the IMS domain. The S-CSCF receives the INVITE request (5) and routes it towards the remote party (7). The session completes as per regular procedures.

When the remote party sends its 200 OK response (9), it is propagated backwards until it reaches the VCC terminal, so, for example, the 200 OK response (14) is an answer to the INVITE request (1). When the VCC terminal receives the 200 OK response (14) the session is already setup in the IMS, so the VCC terminal can seamlessly switch to IMS as soon as the first RTP packets are received.

Then the DTF needs to tear down the initial existing CS call leg, so it sends a BYE request (17) on the initial SIP dialog. This BYE request is routed, following the routing rules for mid-dialog request, to the I-CSCF (not shown in the figure) and eventually to the MGCF. The MGCF generates an ISUP REL message (18) to the VMSC, which sends a DISCONNECT RELEASE message (19) to the VCC terminal. The VCC terminal then disconnects the CS access leg and retains the IMS access leg.

27.6.2 Transfer from the IMS Domain to the CS Domain

Figure 27.12 shows the flow associated with a domain transfer from the IMS to the CS domain. The figure assumes that there is ongoing audio call established over the IMS. The VCC terminal has its session anchored to the VCC application, thus allowing a domain transfer at any time.

At some point in time, perhaps due to a change in the access network conditions, the VCC terminal decides to transfer the domain of the call from the IMS to the CS domain, so it sends a SETUP message (1) address to the VDN, which is a regular E.164 number used for activating a domain transfer. The visited MSC receives the SETUP message (1) and sends a CAMEL IDP message (2) to the gsmSCF, with the purpose of asking for routing information. The gsmSCF interacts (4) with the CSAF to obtain an IMRN and then forwards the IMRN back to the IMRN in a CAMEL CONNECT message (4).

Once the VMSC receives the routing information, i.e., the IMRN, the VMSC proceeds with the call setup by sending a CALL PROCEEDING message (5) to the VCC terminal and an ISUP IAM message (5) addressed to that IMRN. Since the IMRN is a telephone number allocated to the IMS, regular CS routing routes the ISUP IAM message (5) eventually to an MGCF that interfaces the IMS. The MGCF first interacts (7) with the MGW to obtain, among other things, the IP address and port number of the MGW where RTP packets should be sent. The MGCF then creates an INVITE request that is addressed in the *Request-URI* to the IMRN, expressed now as a tel URL. Since the IMRN is a PSI, this INVITE request (8) is routed via an I-CSCF (not shown in the figure) to the CSAF. The CSAF recognizes the IMRN as an associated number to one of the existing ongoing calls anchored in the VCC application.

The CSAF then interacts (10) with the DTF, which is acting as a B2BUA, and which already has an established leg towards the destination due to anchoring when the session was initially established. The DTF then generates a re-INVITE request (11) on this existing SIP dialog towards the S-CSCF. The purpose of this re-INVITE is to modify the dialog information, if needed, according to the new access leg. So, for example, the *From* header can change reflecting now the tel URL of the VCC terminal, but still preserving the same tag

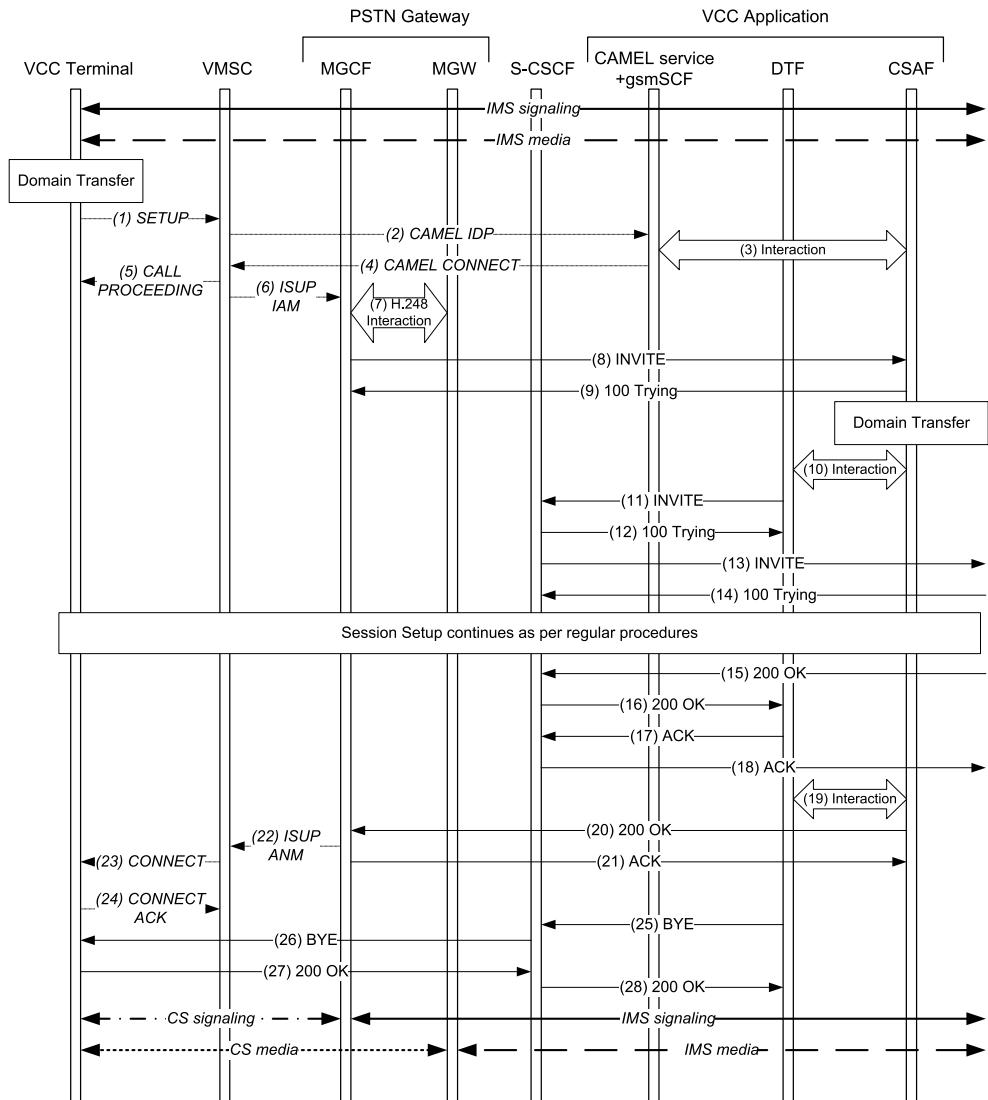


Figure 27.12: Domain transfer of an ongoing call from the IMS to the CS domain

value of the original SIP dialog. The S-CSCF further routes the INVITE request (13) to the destination. The re-INVITE proceeds as a regular session setup.

At some point in time, the S-CSCF receives a 200 OK response (15) from the remote party acknowledging the new parameters in the session. The S-CSCF forwards the 200 OK response (16) to the DTF, which generates an ACK request (18), which is forwarded (19) to the remote party. In addition, the DTF interacts (19) with the CSAF to indicate the success in the transaction.

The CSAF then generates a 200 OK response (20), which acknowledges the received INVITE request (8). The MGCF eventually receives this 200 OK response (20), generates an

ACK request (21) and an ISUP ANM message (22) to the VMSC. The VMSC then sends a CONNECT message (23) to the VCC terminal, which provides an acknowledgement to the VMSC with a CONNECT ACK (24) message.

At this point in time the MGCF is splitting the signaling session, since it has a CS access leg towards the VCC terminal and an IMS access leg towards the VCC application and the remote party. Similarly, the MGCF is spitting the media into two as well. So, the VCC terminal can start making usage of the CS access leg at any time.

The DTF also tears down the IMS access leg by sending a BYE request (25) on the original SIP dialog. This BYE request is received at the S-CSCF, which routes it back (26) to the VCC terminal via a P-CSCF (not shown in the figure). At this point in time, the IMS access leg has been torn down and only the CS access leg remains.

Appendix A

List of IMS-related Specifications

A.1 Introduction

Throughout this book we have seen many references to specifications that concern the IMS. Sometimes, it is not easy to discover which specifications relate to the IMS and which do not. We have compiled a list of the IMS-related specifications and provide this information to the reader as a guideline. We have tried to make the list as comprehensive as possible, but since new specifications are added, removed, replaced, or renumbered we encourage the reader to check the status of a specification to guarantee it is up to date.

In some cases the whole specification is not devoted to the IMS, just a part of it. The interested reader should analyze which parts are applicable to the IMS.

Finally, we have decided not to list the IETF specifications that are relevant for the IMS, since the list is too large. However, the 3GPP specifications contain references to the IETF documents that are relevant, giving the reader the option to verify the source document. IETF RFCs are available for downloading at the RFC editor web page:

<http://www.rfc-editor.org/rfcsearch.html>

A.2 3GPP Specifications

The home page for the 3GPP specifications is

<http://www.3gpp.org/specs/specs.htm>

If you are looking for the latest specification belonging to a 3GPP Release, the entry web page is

<http://www.3gpp.org/ftp/Specs/latest/>

An archive of all of the versions of a specification is available at

<http://www.3gpp.org/ftp/Specs/archive/>

A web page that lists all of the 3GPP specifications including their status is stored at

<http://www.3gpp.org/ftp/Specs/html-info/SpecReleaseMatrix.htm>

Table A.1 lists the IMS-related specifications. The *Spec#* column indicates the specification number. 3GPP specifications follow an “xx.yyy” pattern, where “xx” is the series number and “yyy” is the actual specification. The termination “8yy” is reserved for internal technical reports and “9yy” is for published technical reports.

The *Title* column in Table A.1 indicates the title of the specification. The *R#* column indicates the first 3GPP release to incorporate the particular specification. Unless otherwise noted, once a specification becomes part of a release it becomes part of subsequent releases automatically, even when there are no changes to that specification. Some specifications may take an informative note written in the *Observations* column.

Some specifications refer to Stages 1, 2, or 3. Stage 1 specifications contain requirement specifications. Stage 2 defines the architecture and high-level functional description. Stage 3 is the detailed description of the functional implementation.

A.3 ETSI NGN Specifications

Table A.2 lists some of the ETSI NGN specifications. ETSI NGN specifications related to IMS are based on 3GPP and OMA specifications. Sometimes an endorsement is produced, which is a delta of a 3GPP or OMA document.

ETSI NGN specifications are available for download at

<http://pda.etsi.org/pda/queryform.asp>

A.4 OMA Specifications

Table A.3 lists some of the OMA specifications related to IMS. OMA specifications are grouped into enabler releases and are available for download at

http://www.openmobilealliance.org/Technical/released_enablers.aspx

Table A.1: 3GPP IMS-related specifications

Spec#	Title	R#	Observations
21.905	Vocabulary for 3GPP Specifications	R5	
22.066	Support of Mobile Number Portability (MNP); Stage 1	R6	
22.101	Service aspects; Service principles	R5	
22.141	Presence service; Stage 1	R6	
22.228	Service requirements for the Internet Protocol (IP) multimedia core network subsystem; Stage 1	R5	
22.250	IP Multimedia Subsystem (IMS) Group Management; Stage 1	R6	
22.340	IP Multimedia Subsystem (IMS) messaging; Stage 1	R6	
22.800	IMS Subscription and access scenarios	R6	
23.002	Network Architecture	R5	
23.003	Numbering, Addressing and Identification	R5	
23.125	Overall high-level functionality and architecture impacts of flow-based charging; Stage 2	R6	
23.141	Presence service; Architecture and functional description; Stage 2	R6	
23.167	IP Multimedia Subsystem (IMS) emergency sessions	R7	
23.206	Voice Call Continuity (VCC) between Circuit Switched (CS) and IP Multimedia Subsystem (IMS); Stage 2	R7	
23.207	End-to-end Quality of Service (QoS) concept and architecture	R5	
23.218	IP Multimedia (IM) session handling; IM call model; Stage 2	R5	
23.221	Architectural requirements	R5	
23.228	P Multimedia Subsystem (IMS); Stage 2	R5	
23.271	Location Services (LCS); Functional description; Stage 2	R6	

Table A.1: Continued

Spec#	Title	R#	Observations
23.278	Customised Applications for Mobile network Enhanced Logic (CAMEL) – IP Multimedia System (IMS) interworking; Stage 2	R5	
23.279	Combining Circuit Switched (CS) and IP Multimedia Subsystem (IMS) services; Stage 2	R7	
23.864	Commonality and interoperability between IP Multimedia System (IMS) core networks	R6	
23.917	Dynamic policy control enhancements for end-to-end QoS, Feasibility study	R6	
23.979	3GPP enablers for Push-To-Talk over Cellular (PoC) services Stage 2	R6	
23.981	Interworking aspects and migration scenarios for IPv4-based IP Multimedia Subsystem (IMS) Implementations	R6	
24.141	Presence service using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3	R6	
24.147	Conferencing using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3	R6	
24.173	IMS multimedia telephony communication service and supplementary services; Stage 3	R7	
24.206	Voice call continuity between Circuit Switched (CS) and IP Multimedia Subsystem (IMS); Stage 3	R7	
24.228	Signaling flows for the IP multimedia call control based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3	R5	
24.229	Internet Protocol (IP) multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3	R5	
24.247	Messaging using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3	R6	
24.279	Combining Circuit Switched (CS) and IP Multimedia Subsystem (IMS) services; Stage 3	R7	
24.604	Communication Diversion (CDIV); Protocol specification	R8	

Table A.1: Continued

Spec#	Title	R#	Observations
24.605	Conference (CONF); Protocol specification	R8	
24.606	Message Waiting Indication (MWI); Protocol specification	R8	
24.607	Originating Identification Presentation (OIP) and Originating Identification Restriction (OIR); Protocol specification	R8	
24.608	Terminating Identification Presentation (TIP) and Terminating Identification Restriction (TIR); Protocol specification	R8	
24.610	Communication HOLD (HOLD); Protocol specification	R8	
24.611	Anonymous Communication Rejection (ACRACR) and Communication Barring (CB); Protocol specification	R8	
24.615	Communication Waiting (CW); Protocol specification	R8	
24.616	Malicious Communication Identification (MCID); Protocol specification	R8	
24.623	Extensible Markup Language (XML) Configuration Access Protocol (XCAP) over the Ut interface for Manipulating NGN PSTN/ISDN Simulation Services; Protocol specification	R8	
24.628	Common Basic Communication procedures; Protocol specification	R8	
24.629	Explicit Communication Transfer (ECT); Protocol specification	R8	
24.642	Completion of Communications to Busy Subscriber (CCBS) and Completion of Communications by No Reply (CCNR); Protocol specification	R8	
24.647	Advice Of Charge (AOC); Protocol specification	R8	
24.654	Closed User Group (CUG); Protocol specification	R8	
24.658	Closed User Group (CUG); Protocol specification	R8	
26.235	Packet-switched conversational multimedia applications; Default codecs	R5	
26.236	Packet-switched conversational multimedia applications; Transport protocols	R5	

Table A.1: Continued

Spec#	Title	R#	Observations
29.162	Interworking between the IM CN subsystem and IP networks	R6	
29.163	Interworking between the IP Multimedia (IM) Core Network (CN) subsystem and Circuit-Switched (CS) networks	R6	
29.207	Policy control over Go interface	R5	Deprecated
29.208	End-to-end Quality of Service (QoS) signaling flows	R5	
29.209	Policy control over Gq interface	R6	
29.210	Charging rule provisioning over Gx interface	R6	
29.211	Rx Interface and Rx/Gx signaling flows	R6	
29.212	Policy and charging control over Gx reference point	R7	
29.213	Policy and charging control signaling flows and Quality of Service (QoS) parameter mapping	R7	
29.214	Policy and charging control over Rx reference point	R7	
29.228	IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signaling flows and message contents	R5	
29.229	Cx and Dx interfaces based on the Diameter protocol; Protocol details	R5	
29.278	Customised Applications for Mobile network Enhanced Logic (CAMEL); CAMEL Application Part (CAP) specification for IP Multimedia Subsystems (IMS)	R5	
29.278	Customized Applications for Mobile network Enhanced Logic (CAMEL); CAMEL Application Part (CAP) specification for IP Multimedia Subsystems (IMS)	R5	
29.328	IP Multimedia Subsystem (IMS) Sh interface signaling flows and message content	R5	
29.332	Media Gateway Control Function (MGCF) - IM Media Gateway (IM-MGW); Mn interface	R6	
29.329	Sh interface based on the Diameter protocol	R5	
29.962	Signaling interworking between the 3GPP profile of the Session Initiation Protocol (SIP) and non-3GPP SIP usage	R6	

Table A.1: Continued

Spec#	Title	R#	Observations
31.103	Characteristics of the IP Multimedia Services Identity Module (ISIM) application	R5	
32.200	Telecommunication management; Charging management; Charging principles	R5	Only R5. Replaced by 32.240 in R6
32.225	Telecommunication management; Charging management; Charging data description for the IP Multimedia Subsystem (IMS)	R5	Only R5. Replaced by 32.260 in R6
32.240	Telecommunication management; Charging management; Charging Architecture and Principles	R6	
32.260	Telecommunication management; Charging management; IP Multimedia Subsystem (IMS) charging	R6	
32.421	Telecommunication management; Subscriber and equipment trace: Trace concepts and requirements	R6	
33.102	3G security; Security architecture	R5	
33.108	3G security; Handover interface for Lawful Interception (LI)	R5	
33.141	Presence service; Security	R6	
33.203	3G security; Access security for IP-based services	R5	
33.210	3G security; Network Domain Security (NDS); IP network layer security	R5	
33.978	Security aspects of early IP Multimedia Subsystem (IMS)	R6	

Table A.2: ETSI NGN specifications

Spec#	Title
TR 180 000	NGN Terminology
TR 180 001	NGN Release 1; Release definition
ES 282 001	NGN Functional Architecture Release 1
ES 282 002	PSTN/ISDN Emulation Sub-system (PES); Functional architecture
ES 282 003	Resource and Admission Control Sub-system (RACS); Functional Architecture
ES 282 004	NGN Functional Architecture; Network Attachment Sub-System (NASS)
TS 182 005	Organization of user data
TS 182 006	IP Multimedia Subsystem (IMS); Stage 2 description [3GPP TS 23.228 v7.2.0, modified]
ES 282 007	IP Multimedia Subsystem (IMS); Functional architecture
TS 182 008	Presence Service; Architecture and functional description; [Endorsement of 3GPP TS 23.141 and OMA-AD-Presence-SIMPLE-V1_0]
TS 182 009	NGN Architecture to support emergency communication from citizen to authority [Endorsed document 3GPP TS 23.167, Release 7]
ES 282 010	Charging [Endorsement of 3GPP TS 32.240 v6.3.0, 3GPP TS 32.260 v6.3.0, 3GPP TS 32.297 v6.1.0, 3GPP TS 32.298 v6.1.0 and 3GPP TS 32.299 v6.4.0 modified]
TS 182 011	XML Document Management; Architecture and functional description [OMA-AD-XDM-V1_0-20051006-C modified]
TS 182 012	IMS-based PSTN/ISDN Emulation Subsystem; Functional architecture
TR 182 017	Direct Communication Service; Architecture and functional description [Endorsement of OMA-AD-PoC-V1]
EN 383 001	Interworking between Session Initiation Protocol (SIP) and Bearer Independent Call Control (BICC) Protocol or ISDN User Part (ISUP) [ITU-T Recommendation Q.1912.5, modified] Interworking for SIP/SIP-T (BICC, ISUP)
ES 283 003	IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP) Stage 3 [3GPP TS 24.229 (Release 7), modified]

Table A.2: Continued

Spec#	Title
TS 183 004	PSTN/ISDN simulation services: Communication Diversion (CDIV); Protocol specification
TS 183 005	PSTN/ISDN simulation services: Conference (CONF); Protocol specification
TS 183 006	PSTN/ISDN simulation services; Message Waiting Indication (MWI); Protocol specification
TS 183 007	PSTN/ISDN simulation services: Originating Identification Presentation (OIP) and Originating Identification Restriction (OIR); Protocol specification
TS 183 008	PSTN/ISDN simulation services: Terminating Identification Presentation (TIP) and Terminating Identification Restriction (TIR); Protocol specification
TS 183 010	NGN Signaling Control Protocol; Communication Hold (HOLD); PSTN/ISDN simulation services
TS 183 011	PSTN/ISDN simulation services: Anonymous Communication Rejection (ACRACR) and Communication Barring (CB); Protocol specification
TR 183 013	Analysis of relevant 3GPP IMS specifications for use in TISPAN NGN Release 1 specifications
TS 183 016	PSTN/ISDN simulation services; Malicious Communication Identification (MCID); Protocol specification
TS 183 021	Endorsement of TS 29.162 Interworking between IM CN subsystem and IP networks
TS 183 023	PSTN/ISDN simulation services; Extensible Markup Language (XML) Configuration Access Protocol (XCAP) over the Ut interface for Manipulating NGN PSTN/ISDN Simulation Services
TS 183 028	NGN Signaling Control Protocol; Common Basic Communication procedures
ES 283 027	Interworking SIP-ISUP for TISPAN-IMS
TS 183 028	Common Basic Communication procedures; Protocol specification
TS 183 029	PSTN/ISDN Simulation Services: Explicit Communication Transfer (ECT); Protocol specification
ES 283 030	Presence Service Capability; Protocol Specification [3GPP TS 24.141 V7.0.0, modified and OMA-TS-Presence_SIMPLE-V1_0, modified]; Presence Capability support

Table A.2: Continued

Spec#	Title
ES 283 031	IP Multimedia: H.248 Profile for controlling Multimedia Resource Function Processors (MRFP) in the IP Multimedia System (IMS); Protocol specification
TS 183 033	IP Multimedia; Diameter based protocol for the interfaces between the Call Session Control Function and the User Profile Server Function/Subscription Locator Function; Signaling flows and protocol details [3GPP TS 29.228 V6.8.0 and 3GPP TS 29.229 V6.6.0, modified]
TS 183 038	PSTN/ISDN Simulation Services; Extensible Markup Language (XML) Document Management; Protocol Specification [Endorsement of OMA-TS-XDM_Core-V1_0-20051103-C and OMA-TS-XDM_Shared-V1_0-20051006-C]
TS 183 041	Messaging service using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3: Protocol specifications [Endorsement of 3GPP TS 24.247 Release 6]
TS 187 003	NGN Security; Security Architecture

Table A.3: OMA specifications

Enabler	Title
IP Multimedia Subsystem (IMS in OMA) V1.0	Utilization of IMS capabilities Architecture
OMA Presence Simple V2.0	Presence Simple Architecture Presence Simple Specification Presence Content XDM Specification Presence XDM Specification Resource List Server (RLS) XDM Specification OMA Management Object for SIMPLE Presence
OMA Push to talk over Cellular V2.0	Push to talk over Cellular (PoC) - Architecture OMA PoC Control Plane OMA PoC Document Management OMA PoC Endorsement of OMA IM TS PoC Interworking Service PoC Invocation Descriptor OMA PoC System Description PoC User Plane
OMA SIMPLE IM V1.0	Instant Messaging using SIMPLE Architecture IM XDM Specification OMA SIMPLE IM Charging Specification Instant Messaging using SIMPLE
OMA XML Document Management V2.0	XML Document Management Architecture XML Document Management Specification OMA Management Object for XML Document Management Shared Group XDM Specification Shared List XDM Specification Shared Policy XDM Specification Shared Profile XDM Specification

References

- [1] 3GPP. Customized Applications for Mobile network Enhanced Logic (CAMEL); CAMEL Application Part (CAP) specification for IP Multimedia Subsystems (IMS). TS 29.278, 3rd Generation Partnership Project (3GPP), December 2005.
- [2] 3GPP. Specification of the Subscriber Identity Module – Mobile Equipment (SIM-ME) interface. TS 51.011, 3rd Generation Partnership Project (3GPP), June 2005.
- [3] 3GPP. Customised Applications for Mobile network Enhanced Logic (CAMEL) Phase 4; Stage 2; IM CN Interworking. TS 23.278, 3rd Generation Partnership Project (3GPP), March 2006.
- [4] 3GPP. Interworking between the IM CN subsystem and IP networks. TS 29.162, 3rd Generation Partnership Project (3GPP), March 2006.
- [5] 3GPP. Speech codec speech processing functions; Adaptive Multi-Rate – Wideband (AMR-WB) speech codec; General description. TS 26.171, 3rd Generation Partnership Project (3GPP), October 2006.
- [6] 3GPP. 3GPP enablers for Open Mobile Alliance (OMA) Push-to-talk over Cellular (PoC) services; Stage 2. TR 23.979, 3rd Generation Partnership Project (3GPP), June 2007.
- [7] 3GPP. AMR speech Codec; General description. TS 26.071, 3rd Generation Partnership Project (3GPP), July 2007.
- [8] 3GPP. AMR speech Codec; Transcoding Functions. TS 26.090, 3rd Generation Partnership Project (3GPP), June 2007.
- [9] 3GPP. Bootstrapping interface (Ub) and network application function interface (Ua); Protocol details. TS 24.109, 3rd Generation Partnership Project (3GPP), December 2007.
- [10] 3GPP. Characteristics of the IP Multimedia Services Identity Module (ISIM) application. TS 31.103, 3rd Generation Partnership Project (3GPP), September 2007.
- [11] 3GPP. Combining Circuit Switched (CS) and IP Multimedia Subsystem (IMS) services; Stage 2. TS 23.279, 3rd Generation Partnership Project (3GPP), September 2007.
- [12] 3GPP. Combining Circuit Switched (CS) and IP Multimedia Subsystem (IMS) services; Stage 3. TS 24.279, 3rd Generation Partnership Project (3GPP), December 2007.
- [13] 3GPP. End-to-end Quality of Service (QoS) concept and architecture. TS 23.207, 3rd Generation Partnership Project (3GPP), June 2007.
- [14] 3GPP. Functional stage 2 description of Location Services (LCS). TS 23.271, 3rd Generation Partnership Project (3GPP), September 2007.
- [15] 3GPP. Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS). TS 33.222, 3rd Generation Partnership Project (3GPP), December 2007.

- [16] 3GPP. Interworking aspects and migration scenarios for IPv4-based IP Multimedia Subsystem (IMS) implementations. TR 23.981, 3rd Generation Partnership Project (3GPP), June 2007.
- [17] 3GPP. Network architecture. TS 23.002, 3rd Generation Partnership Project (3GPP), December 2007.
- [18] 3GPP. Open Service Access (OSA) Application Programming Interface (API); Part 1: Overview. TS 29.198-01, 3rd Generation Partnership Project (3GPP), March 2007.
- [19] 3GPP. Policy control over Gq interface. TS 29.209, 3rd Generation Partnership Project (3GPP), June 2007.
- [20] 3GPP. Security aspects of early IP Multimedia Subsystem (IMS). TR 33.978, 3rd Generation Partnership Project (3GPP), June 2007.
- [21] 3GPP. Specification of the Subscriber Identity Module – Mobile Equipment (SIM-ME) Interface. TS 11.11, 3rd Generation Partnership Project (3GPP), June 2007.
- [22] 3GPP. Speech codec speech processing functions; Adaptive Multi-Rate – Wideband (AMR-WB) speech codec; Transcoding functions. TS 26.190, 3rd Generation Partnership Project (3GPP), June 2007.
- [23] 3GPP. Telecommunication management; Charging management; Charging Data Record (CDR) transfer. TS 32.295, 3rd Generation Partnership Project (3GPP), June 2007.
- [24] 3GPP. Telecommunication management; Charging management; Online Charging System (OCS): Applications and interfaces. TS 32.296, 3rd Generation Partnership Project (3GPP), December 2007.
- [25] 3GPP. Telecommunication management; Charging management; Push-to-talk over Cellular (PoC) charging. TS 32.272, 3rd Generation Partnership Project (3GPP), June 2007.
- [26] 3GPP. Voice call continuity between Circuit Switched (CS) and IP Multimedia Subsystem (IMS); Stage 3. TS 24.206, 3rd Generation Partnership Project (3GPP), December 2007.
- [27] 3GPP. Voice Call Continuity (VCC) between Circuit Switched (CS) and IP Multimedia Subsystem (IMS); Stage 2. TS 23.206, 3rd Generation Partnership Project (3GPP), December 2007.
- [28] 3GPP. 3G security; Access security for IP-based services. TS 33.203, 3rd Generation Partnership Project (3GPP), March 2008.
- [29] 3GPP. 3G security; Network Domain Security (NDS); IP network layer security. TS 33.210, 3rd Generation Partnership Project (3GPP), March 2008.
- [30] 3GPP. Architectural requirements. TS 23.221, 3rd Generation Partnership Project (3GPP), March 2008.
- [31] 3GPP. Characteristics of the Universal Subscriber Identity Module (USIM) application. TS 31.102, 3rd Generation Partnership Project (3GPP), March 2008.
- [32] 3GPP. Conferencing using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3. TS 24.147, 3rd Generation Partnership Project (3GPP), March 2008.
- [33] 3GPP. Cx and Dx interfaces based on the Diameter protocol; Protocol details. TS 29.229, 3rd Generation Partnership Project (3GPP), March 2008.
- [34] 3GPP. General Packet Radio Service (GPRS); GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface. TS 29.060, 3rd Generation Partnership Project (3GPP), March 2008.
- [35] 3GPP. General Packet Radio Service (GPRS); Service description; Stage 2. TS 23.060, 3rd Generation Partnership Project (3GPP), March 2008.
- [36] 3GPP. IMS Multimedia telephony service and supplementary services; Stage 3. TS 24.173, 3rd Generation Partnership Project (3GPP), March 2008.

- [37] 3GPP. Internet Protocol (IP) multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3. TS 24.229, 3rd Generation Partnership Project (3GPP), March 2008.
- [38] 3GPP. Interworking between the IP Multimedia (IM) Core Network (CN) subsystem and Circuit Switched (CS) networks. TS 29.163, 3rd Generation Partnership Project (3GPP), March 2008.
- [39] 3GPP. IP Multimedia (IM) session handling; IM call model; Stage 2. TS 23.218, 3rd Generation Partnership Project (3GPP), March 2008.
- [40] 3GPP. IP Multimedia (IM) Subsystem Cx and Dx Interfaces; Signalling flows and message contents. TS 29.228, 3rd Generation Partnership Project (3GPP), March 2008.
- [41] 3GPP. IP Multimedia Subsystem (IMS) emergency sessions. TS 23.167, 3rd Generation Partnership Project (3GPP), March 2008.
- [42] 3GPP. IP Multimedia Subsystem (IMS) Sh interface; Signalling flows and message contents. TS 29.328, 3rd Generation Partnership Project (3GPP), March 2008.
- [43] 3GPP. IP Multimedia Subsystem (IMS); Stage 2. TS 23.228, 3rd Generation Partnership Project (3GPP), March 2008.
- [44] 3GPP. Media Gateway Control Function (MGCF) – IM Media Gateway (IM-MGW); Mn interface. TS 29.332, 3rd Generation Partnership Project (3GPP), March 2008.
- [45] 3GPP. Messaging service using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3. TS 24.247, 3rd Generation Partnership Project (3GPP), March 2008.
- [46] 3GPP. Mobile Application Part (MAP) specification. TS 29.002, 3rd Generation Partnership Project (3GPP), March 2008.
- [47] 3GPP. Mobile radio interface Layer 3 specification; Core network protocols; Stage 3. TS 24.008, 3rd Generation Partnership Project (3GPP), March 2008.
- [48] 3GPP. Packet switched conversational multimedia applications; Default codecs. TS 26.235, 3rd Generation Partnership Project (3GPP), March 2008.
- [49] 3GPP. Policy and charging control over Gx reference point. TS 29.212, 3rd Generation Partnership Project (3GPP), March 2008.
- [50] 3GPP. Policy and charging control over Rx reference point. TS 29.214, 3rd Generation Partnership Project (3GPP), March 2008.
- [51] 3GPP. Presence service using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3. TS 24.141, 3rd Generation Partnership Project (3GPP), March 2008.
- [52] 3GPP. PSTN/ISDN simulation services Terminating Identification Presentation (TIP) and Terminating Identification Restriction (TIR); Protocol specification. TS 24.608, 3rd Generation Partnership Project (3GPP), April 2008.
- [53] 3GPP. Service requirements for the Internet Protocol (IP) multimedia core network subsystem (IMS); Stage 1. TS 22.228, 3rd Generation Partnership Project (3GPP), March 2008.
- [54] 3GPP. Sh interface based on the Diameter protocol; Protocol details. TS 29.329, 3rd Generation Partnership Project (3GPP), March 2008.
- [55] 3GPP. Telecommunication management; Charging management; Charging architecture and principles. TS 32.240, 3rd Generation Partnership Project (3GPP), March 2008.
- [56] 3GPP. Telecommunication management; Charging management; Charging Data Record (CDR) file format and transfer. TS 32.297, 3rd Generation Partnership Project (3GPP), March 2008.
- [57] 3GPP. Telecommunication management; Charging management; Charging Data Record (CDR) parameter description. TS 32.298, 3rd Generation Partnership Project (3GPP), March 2008.

- [58] 3GPP. Telecommunication management; Charging management; Diameter charging applications. TS 32.299, 3rd Generation Partnership Project (3GPP), March 2008.
- [59] 3GPP. Telecommunication management; Charging management; IP Multimedia Subsystem (IMS) charging. TS 32.260, 3rd Generation Partnership Project (3GPP), March 2008.
- [60] 3GPP. TISPAN; Common Basic Communication procedures; Protocol specification. TS 24.628, 3rd Generation Partnership Project (3GPP), April 2008.
- [61] 3GPP. TISPAN; NGN IMS Supplementary Services; Advice Of Charge (AOC). TS 24.647, 3rd Generation Partnership Project (3GPP), April 2008.
- [62] 3GPP. TISPAN; NGN IMS Supplementary Services; Completion of Communications to Busy Subscriber (CCBS) and Completion of Communications by No Reply (CCNR). TS 24.642, 3rd Generation Partnership Project (3GPP), April 2008.
- [63] 3GPP. TISPAN; NGN Signalling Control Protocol; Communication HOLD (HOLD) PSTN/ISDN simulation services; Protocol specification. TS 24.610, 3rd Generation Partnership Project (3GPP), April 2008.
- [64] 3GPP. TISPAN; PSTN/ISDN simulation services: Anonymous Communication Rejection (ACR) and Communication Barring (CB); Protocol specification. TS 24.611, 3rd Generation Partnership Project (3GPP), April 2008.
- [65] 3GPP. TISPAN; PSTN/ISDN simulation services: Communication Diversion (CDIV); Protocol specification. TS 24.604, 3rd Generation Partnership Project (3GPP), April 2008.
- [66] 3GPP. TISPAN; PSTN/ISDN simulation services: Conference (CONF); Protocol specification. TS 24.605, 3rd Generation Partnership Project (3GPP), April 2008.
- [67] 3GPP. TISPAN; PSTN/ISDN simulation services: Explicit Communication Transfer (ECT); Protocol specification. TS 24.629, 3rd Generation Partnership Project (3GPP), April 2008.
- [68] 3GPP. TISPAN; PSTN/ISDN simulation services; Extensible Markup Language (XML) Configuration Access Protocol (XCAP) over the Ut interface for Manipulating NGN PSTN/ISDN Simulation Services. TS 24.623, 3rd Generation Partnership Project (3GPP), April 2008.
- [69] 3GPP. TISPAN; PSTN/ISDN simulation services; Malicious Communication Identification (MCID); Protocol specification. TS 24.616, 3rd Generation Partnership Project (3GPP), April 2008.
- [70] 3GPP. TISPAN; PSTN/ISDN simulation services; Message Waiting Indication (MWI); Protocol specification. TS 24.606, 3rd Generation Partnership Project (3GPP), April 2008.
- [71] 3GPP. TISPAN; PSTN/ISDN simulation services; Originating Identification Presentation (OIP) and Originating Identification Restriction (OIR); Protocol specification. TS 24.607, 3rd Generation Partnership Project (3GPP), April 2008.
- [72] B. Aboba and M. Beadles. The Network Access Identifier. RFC 2486, Internet Engineering Task Force, January 1999.
- [73] A. Allen, J. Holm, and T. Hallin. The P-Answer-State Header Extension to the Session Initiation Protocol for the Open Mobile Alliance Push to Talk over Cellular. RFC 4964, Internet Engineering Task Force, September 2007.
- [74] ANSI/TIA. Link Layer Discovery Protocol for Media Endpoint Devices. TIA Standard TIA-1057, Telecommunications Industry Association, April 2006.
- [75] M. Arango, A. Dugan, I. Elliott, C. Huitema, and S. Pickett. Media Gateway Control Protocol (MGCP) Version 1.0. RFC 2705, Internet Engineering Task Force, October 1999.
- [76] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman. MIKEY: Multimedia Internet KEYing. RFC 3830, Internet Engineering Task Force, August 2004.

- [77] J. Arkko, G. Kuijpers, H. Soliman, J. Loughney, and J. Wiljakka. Internet Protocol Version 6 (IPv6) for Some Second and Third Generation Cellular Hosts. RFC 3316, Internet Engineering Task Force, April 2003.
- [78] J. Arkko, F. Lindholm, M. Naslund, K. Norrman, and E. Carrara. Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP). RFC 4567, Internet Engineering Task Force, July 2006.
- [79] J. Arkko, V. Torvinen, G. Camarillo, A. Niemi, and T. Haukka. Security Mechanism Agreement for the Session Initiation Protocol (SIP). RFC 3329, Internet Engineering Task Force, January 2003.
- [80] F. Audet and C. Jennings. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787, Internet Engineering Task Force, January 2007.
- [81] M. Barnes. An Extension to the Session Initiation Protocol (SIP) for Request History Information. RFC 4244, Internet Engineering Task Force, November 2005.
- [82] M. Barnes, C. Boulton, and O. Levin. A Framework for Centralized Conferencing. Internet-Draft draft-ietf-xcon-framework-11, Internet Engineering Task Force, April 2008. Work in progress.
- [83] M. Barnes, J. Winterbottom, M. Thomson, and B. Stark. HTTP Enabled Location Delivery (HELD). Internet-Draft draft-ietf-geopriv-http-location-delivery-07, Internet Engineering Task Force, April 2008. Work in progress.
- [84] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The Secure Real-time Transport Protocol (SRTP). RFC 3711, Internet Engineering Task Force, March 2004.
- [85] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396, Internet Engineering Task Force, August 1998.
- [86] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, Internet Engineering Task Force, January 2005.
- [87] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC 2475, Internet Engineering Task Force, December 1998.
- [88] S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie, and J. Simeon. XQuery 1.0: An XML Queru Language. W3c Recommendation, World Wide Web Consortium, January 2007.
- [89] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633, Internet Engineering Task Force, June 1994.
- [90] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205, Internet Engineering Task Force, September 1997.
- [91] S. Bradner, P. Calhoun, H. Cuschieri, S. Dennett, G. Flynn, M. Lipford, and M. McPheters. 3GPP2-IETF Standardization Collaboration. RFC 3131, Internet Engineering Task Force, June 2001.
- [92] T. Bray, D. Hollander, A. Layman, and R. Tobin. Namespaces in XML 1.1 (Second Edition). W3c Recommendation, World Wide Web Consortium, August 2006.
- [93] T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Cowan. Extensible Markup Language (XML) 1.1 (Second Edition). W3c Recommendation, World Wide Web Consortium, August 2006.
- [94] E. Burger. A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages. RFC 4483, Internet Engineering Task Force, May 2006.

- [95] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet Group Management Protocol, Version 3. RFC 3376, Internet Engineering Task Force, October 2002.
- [96] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter Base Protocol. RFC 3588, Internet Engineering Task Force, September 2003.
- [97] G. Camarillo. *SIP Demystified*. McGraw-Hill, 2001.
- [98] G. Camarillo. Compressing the Session Initiation Protocol (SIP). RFC 3486, Internet Engineering Task Force, February 2003.
- [99] G. Camarillo. Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams. RFC 4583, Internet Engineering Task Force, November 2006.
- [100] G. Camarillo. Connection Establishment in the Binary Floor Control Protocol (BFCP). RFC 5018, Internet Engineering Task Force, September 2007.
- [101] G. Camarillo, G. Eriksson, J. Holler, and H. Schulzrinne. Grouping of Media Lines in the Session Description Protocol (SDP). RFC 3388, Internet Engineering Task Force, December 2002.
- [102] G. Camarillo and A. Johnston. Conference Establishment Using Request-Contained Lists in the Session Initiation Protocol (SIP). Internet-Draft draft-ietf-sip-uri-list-conferencing-02, Internet Engineering Task Force, November 2007. Work in progress.
- [103] G. Camarillo, W. Marshall, and J. Rosenberg. Integration of Resource Management and Session Initiation Protocol (SIP). RFC 3312, Internet Engineering Task Force, October 2002.
- [104] G. Camarillo and A. Monrad. Mapping of Media Streams to Resource Reservation Flows. RFC 3524, Internet Engineering Task Force, April 2003.
- [105] G. Camarillo, A. Niemi, M. Isomaki, M. García-Martín, and H. Khatabil. Referring to Multiple Resources in the Session Initiation Protocol (SIP). Internet-Draft draft-ietf-sip-multiple-refer-03, Internet Engineering Task Force, December 2007. Work in progress.
- [106] G. Camarillo, J. Ott, and K. Drage. The Binary Floor Control Protocol (BFCP). RFC 4582, Internet Engineering Task Force, November 2006.
- [107] G. Camarillo and A. Roach. Framework and Security Considerations for Session Initiation Protocol (SIP) Uniform Resource Identifier (URI)-List Services. Internet-Draft draft-ietf-sipping-uri-services-07, Internet Engineering Task Force, November 2007. Work in progress.
- [108] G. Camarillo, A. Roach, and O. Levin. Subscriptions to Request-Contained Resource Lists in the Session Initiation Protocol (SIP). Internet-Draft draft-ietf-sip-uri-list-subscribe-02, Internet Engineering Task Force, November 2007. Work in progress.
- [109] G. Camarillo, A. B. Roach, J. Peterson, and L. Ong. Integrated Services Digital Network (ISDN) User Part (ISUP) to Session Initiation Protocol (SIP) Mapping. RFC 3398, Internet Engineering Task Force, December 2002.
- [110] G. Camarillo, A. B. Roach, J. Peterson, and L. Ong. Mapping of Integrated Services Digital Network (ISDN) User Part (ISUP) Overlap Signalling to the Session Initiation Protocol (SIP). RFC 3578, Internet Engineering Task Force, August 2003.
- [111] G. Camarillo and H. Schulzrinne. Early Media and Ringing Tone Generation in the Session Initiation Protocol (SIP). RFC 3960, Internet Engineering Task Force, December 2004.
- [112] G. Camarillo, H. Schulzrinne, and R. Kantola. Evaluation of Transport Protocols for the Session Initiation Protocol. *IEEE Network*, 17(5), 2003.

- [113] G. Camarillo, S. Srinivasan, R. Even, and J. Urpalainen. Conference Event Package Data Format Extension for Centralized Conferencing (XCON). Internet-Draft draft-ietf-xcon-event-package-00, Internet Engineering Task Force, February 2008. Work in progress.
- [114] B. Campbell, R. Mahy, and C. Jennings. The Message Session Relay Protocol (MSRP). RFC 4975, Internet Engineering Task Force, September 2007.
- [115] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. Session Initiation Protocol (SIP) Extension for Instant Messaging. RFC 3428, Internet Engineering Task Force, December 2002.
- [116] K. Chan, R. Sahita, S. Hahn, and K. McCloghrie. Differentiated Services Quality of Service Policy Information Base. RFC 3317, Internet Engineering Task Force, March 2003.
- [117] B. Davie, A. Charny, J. C. R. Bennet, K. Benson, J. Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246, Internet Engineering Task Force, March 2002.
- [118] F. Dawson and T. Howes. vCard MIME Directory Profile. RFC 2426, Internet Engineering Task Force, September 1998.
- [119] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, Internet Engineering Task Force, December 1998.
- [120] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, Internet Engineering Task Force, January 1999.
- [121] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, Internet Engineering Task Force, April 2006.
- [122] K. Drage. A Session Initiation Protocol (SIP) Extension for the Identification of Services. Internet-Draft draft-drage-sipping-service-identification-01, Internet Engineering Task Force, July 2007. Work in progress.
- [123] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, Internet Engineering Task Force, March 1997.
- [124] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315, Internet Engineering Task Force, July 2003.
- [125] M. Duerst and M. Suignard. Internationalized Resource Identifiers (IRIs). RFC 3987, Internet Engineering Task Force, January 2005.
- [126] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. The COPS (Common Open Policy Service) Protocol. RFC 2748, Internet Engineering Task Force, January 2000.
- [127] D. Eastlake 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1). RFC 3174, Internet Engineering Task Force, September 2001.
- [128] R. Ejza. Private Header (P-Header) Extension to the Session Initiation Protocol (SIP) for Authorization of Early Media. RFC 5009, Internet Engineering Task Force, September 2007.
- [129] ETSI. Digital cellular telecommunications system (Phase 2+); Full rate speech; Transcoding (GSM 06.10 version 5.1.1). ETS 300 961, European Telecommunications Standards Institute, May 1998.
- [130] ETSI. Draft: Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); PSTN/ISDN Simulation Services; Completion of Communications to Busy Subscriber (CCBS); Completion of Communications by No Reply (CCNR); Protocol Specification. TS 183 042, European Telecommunications Standards Institute, May 2007.

- [131] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); NGN IMS Supplementary Services; Advice of Charge (AoC). TS 183 047, European Telecommunications Standards Institute, March 2007.
- [132] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); NGN Signalling Control Protocol; Communication Hold (HOLD); PSTN/ISDN simulation services; Protocol specification. TS 183 010, European Telecommunications Standards Institute, April 2007.
- [133] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); PSTN/ISDN Simulation Services; Anonymous Communication Rejection (ACR) and Communication Barring (CB); Protocol Specification. TS 183 011, European Telecommunications Standards Institute, March 2007.
- [134] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); PSTN/ISDN simulation services; Communication Diversion (CDIV); Protocol specification. TS 183 004, European Telecommunications Standards Institute, March 2007.
- [135] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); PSTN/ISDN Simulation Services; Conference (CONF); Protocol specification. TS 183 005, European Telecommunications Standards Institute, April 2007.
- [136] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); PSTN/ISDN simulation services; Explicit Communication Transfer (ECT); Protocol specification. TS 183 029, European Telecommunications Standards Institute, April 2007.
- [137] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); PSTN/ISDN simulation services; Extensible Markup Language (XML) Configuration Access Protocol (XCAP) over the Ut interface for Manipulating NGN PSTN/ISDN Simulation Services. TS 183 023, European Telecommunications Standards Institute, April 2007.
- [138] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); PSTN/ISDN simulation services; Malicious Call Identification (MCID); Protocol Specification. TS 183 016, European Telecommunications Standards Institute, July 2007.
- [139] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); PSTN/ISDN simulation services; Message Waiting Indication (MWI); Protocol Specification. TS 183 006, European Telecommunications Standards Institute, March 2007.
- [140] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); PSTN/ISDN simulation services; Originating Identification Presentation (OIP) and Originating Identification Restriction (OIR); Protocol Specification. TS 183 007, European Telecommunications Standards Institute, March 2007.
- [141] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); PSTN/ISDN simulation services; Terminating Identification Presentation (TIP) and Terminating Identification Restriction (TIR); Protocol Specification. TS 183 008, European Telecommunications Standards Institute, March 2007.
- [142] ETSI. Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN); Common Basic Communication procedures; Protocol specification. TS 183 028, European Telecommunications Standards Institute, January 2008.
- [143] P. Faltstrom. E.164 number and DNS. RFC 2916, Internet Engineering Task Force, September 2000.

- [144] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999.
- [145] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP Authentication: Basic and Digest Access Authentication. RFC 2617, Internet Engineering Task Force, June 1999.
- [146] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, Internet Engineering Task Force, November 1996.
- [147] J. Galvin, S. Murphy, S. Crocker, and N. Freed. Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted. RFC 1847, Internet Engineering Task Force, October 1995.
- [148] M. García-Martín. A Session Initiation Protocol (SIP) Event Package and Data Format for Various Settings in Support for the Push-to-Talk over Cellular (PoC) Service. RFC 4354, Internet Engineering Task Force, January 2006.
- [149] M. García-Martín. The Presence-Specific Static Dictionary for Signaling Compression (SigComp). RFC 5112, Internet Engineering Task Force, January 2008.
- [150] M. García-Martín, M. Belinchon, M. Pallares-Lopez, C. Canales-Valenzuela, and K. Tammi. Diameter Session Initiation Protocol (SIP) Application. RFC 4740, Internet Engineering Task Force, November 2006.
- [151] M. García-Martín, C. Bormann, J. Ott, R. Price, and A. B. Roach. The Session Initiation Protocol (SIP) and Session Description Protocol (SDP) Static Dictionary for Signaling Compression (SigComp). RFC 3485, Internet Engineering Task Force, February 2003.
- [152] M. García-Martín and G. Camarillo. Extensible Markup Language (XML) Format Extension for Representing Copy Control Attributes in Resource Lists. Internet-Draft draft-ietf-sipping-capacity-attribute-06, Internet Engineering Task Force, December 2007. Work in progress.
- [153] M. García-Martín and G. Camarillo. Multiple-Recipient MESSAGE Requests in the Session Initiation Protocol (SIP). Internet-Draft draft-ietf-sip-uri-list-message-03, Internet Engineering Task Force, December 2007. Work in progress.
- [154] M. García-Martín, E. Henrikson, and D. Mills. Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP). RFC 3455, Internet Engineering Task Force, January 2003.
- [155] M. García-Martín, M. Isomaki, G. Camarillo, S. Loreto, and P. Kyzivat. A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer. Internet-Draft draft-ietf-mmusic-file-transfer-mech-07, Internet Engineering Task Force, March 2008. Work in progress.
- [156] G. Good. The LDAP Data Interchange Format (LDIF) – Technical Specification. RFC 2849, Internet Engineering Task Force, June 2000.
- [157] D. Grossman. New Terminology and Clarifications for Diffserv. RFC 3260, Internet Engineering Task Force, April 2002.
- [158] H. Hakala, L. Mattila, J-P. Koskinen, M. Stura, and J. Loughney. Diameter Credit-Control Application. RFC 4006, Internet Engineering Task Force, August 2005.
- [159] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 3448, Internet Engineering Task Force, January 2003.
- [160] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327, Internet Engineering Task Force, April 1998.

- [161] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. RFC 2543, Internet Engineering Task Force, March 1999.
- [162] H. Hannu, J. Christoffersson, S. Forsgren, K.-C. Leung, Z. Liu, and R. Price. Signaling Compression (SigComp) – Extended Operations. RFC 3321, Internet Engineering Task Force, January 2003.
- [163] T. Hardie, A. Newton, H. Schulzrinne, and H. Tschofenig. LoST: A Location-to-Service Translation Protocol. Internet-Draft draft-ietf-ecrit-lost-09, Internet Engineering Task Force, March 2008. Work in progress.
- [164] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, Internet Engineering Task Force, November 1998.
- [165] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. RFC 2597, Internet Engineering Task Force, June 1999.
- [166] V. Hilt, G. Camarillo, and J. Rosenberg. A Framework for Session Initiation Protocol (SIP) Session Policies. Internet-Draft draft-ietf-sip-session-policy-framework-03, Internet Engineering Task Force, April 2008. Work in progress.
- [167] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar. RTP Payload Format for MPEG1/MPEG2 Video. RFC 2250, Internet Engineering Task Force, January 1998.
- [168] R. Housley. Cryptographic Message Syntax (CMS). RFC 3369, Internet Engineering Task Force, August 2002.
- [169] G. Huston and I. Leuca. OMA-IETF Standardization Collaboration. RFC 3975, Internet Engineering Task Force, January 2005.
- [170] IEEE. Station and Media Access Control Connectivity Discovery. Standard IEEE 802.1AB, Institute of Electrical and Electronics Engineers, May 2005.
- [171] ISO. Coding of moving pictures and associated audio for digital storage media up to about 1.5 Mbit/s. Standard ISO/IEC 11172, International Organization for Standardization, November 1993.
- [172] ISO. Generic coding of moving pictures and associated audio information. Standard ISO/IEC 13818, International Organization for Standardization, November 1994.
- [173] ISO. Information technology – Coding of audio-visual objects – Part 1: Systems. Standard ISO/IEC 14496-1, International Organization for Standardization, June 2001.
- [174] ISO. Information technology – Coding of audio-visual objects – Part 2: Visual. Standard ISO/IEC 14496-2, International Organization for Standardization, June 2001.
- [175] M. Isomaki and E. Leppanen. An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Manipulating Presence Document Contents. RFC 4827, Internet Engineering Task Force, May 2007.
- [176] ITU-T. Functional description of the Signalling System No. 7 Telephone User Part (TUP). Recommendation Q.721, International Telecommunication Union, November 1988.
- [177] ITU-T. Pulse code modulation (PCM) of voice frequencies. Recommendation G.711, International Telecommunication Union, November 1988.
- [178] ITU-T. 40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM). Recommendation G.726, International Telecommunication Union, December 1990.
- [179] ITU-T. Functional description of the message transfer part (MTP) of Signalling System No. 7. Recommendation Q.701, International Telecommunication Union, March 1993.
- [180] ITU-T. Introduction to CCITT Signalling System No. 7. Recommendation Q.700, International Telecommunication Union, March 1993.

- [181] ITU-T. Video codec for audiovisual services at $p \times 64$ kbit/s. Recommendation H.261, International Telecommunication Union, March 1993.
- [182] ITU-T. Protocol for multimedia application text conversation. Recommendation T.140, International Telecommunication Union, February 1998.
- [183] ITU-T. Video coding for low bit rate communication. Recommendation H.263, International Telecommunication Union, February 1998.
- [184] ITU-T. Narrow-band visual telephone systems and terminal equipment. Recommendation H.320, International Telecommunication Union, May 1999.
- [185] ITU-T. Signalling System No. 7 – ISDN User Part functional description. Recommendation Q.761, International Telecommunication Union, December 1999.
- [186] ITU-T. Bearer Independent Call Control protocol. Recommendation Q.1901, International Telecommunication Union, June 2000.
- [187] ITU-T. Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks. Recommendation X.509, International Telecommunication Union, March 2000.
- [188] ITU-T. H.263 Annex X: Profiles and levels definitions. Recommendation H.263 Annex X, International Telecommunication Union, April 2001.
- [189] ITU-T. Gateway control protocol: Version 2. Recommendation H.248, International Telecommunication Union, May 2002.
- [190] ITU-T. Terminal for low bit-rate multimedia communication. Recommendation H.324, International Telecommunication Union, March 2002.
- [191] ITU-T. Packet-based multimedia communication systems. Recommendation H.323, International Telecommunication Union, July 2003.
- [192] C. Jennings, F. Audet, and J. Elwell. Session Initiation Protocol (SIP) URIs for Applications such as Voicemail and Interactive Voice Response (IVR). RFC 4458, Internet Engineering Task Force, April 2006.
- [193] C. Jennings, R. Mahy, and A. B. Roach. Relay Extensions for the Message Sessions Relay Protocol (MSRP). RFC 4976, Internet Engineering Task Force, September 2007.
- [194] C. Jennings, J. Peterson, and M. Watson. Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks. RFC 3325, Internet Engineering Task Force, November 2002.
- [195] A. Johnston and O. Levin. Session Initiation Protocol (SIP) Call Control – Conferencing for User Agents. RFC 4579, Internet Engineering Task Force, August 2006.
- [196] S. Josefsson. The Base16, Base32, and Base64 Data Encodings. RFC 3548, Internet Engineering Task Force, July 2003.
- [197] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306, Internet Engineering Task Force, December 2005.
- [198] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303, Internet Engineering Task Force, December 2005.
- [199] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). RFC 2406, Internet Engineering Task Force, November 1998.
- [200] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, Internet Engineering Task Force, November 1998.
- [201] J. Klensin. Simple Mail Transfer Protocol. RFC 2821, Internet Engineering Task Force, April 2001.

- [202] J. Klensin. Terminology for Describing Internet Connectivity. RFC 4084, Internet Engineering Task Force, May 2005.
- [203] G. Klyne and D. Atkins. Common Presence and Instant Messaging (CPIM): Message Format. RFC 3862, Internet Engineering Task Force, August 2004.
- [204] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP). RFC 4340, Internet Engineering Task Force, March 2006.
- [205] P. Leach, M. Mealling, and R. Salz. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122, Internet Engineering Task Force, July 2005.
- [206] J. Lennox. Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP). RFC 4572, Internet Engineering Task Force, July 2006.
- [207] O. Levin. Suppression of Session Initiation Protocol (SIP) REFER Method Implicit Subscription. RFC 4488, Internet Engineering Task Force, May 2006.
- [208] M. Lonnfors and K. Kiss. Session Initiation Protocol (SIP) User Agent Capability Extension to Presence Information Data Format (PIDF). Internet-Draft draft-ietf-simple-prescaps-ext-09, Internet Engineering Task Force, March 2008. Work in progress.
- [209] M. Lonnfors, E. Leppanen, H. Khatabil, and J. Urpalainen. Presence Information Data format (PIDF) Extension for Partial Presence. Internet-Draft draft-ietf-simple-partial-pidf-format-10, Internet Engineering Task Force, November 2007. Work in progress.
- [210] J. Loughney. Diameter Command Codes for Third Generation Partnership Project (3GPP) Release 5. RFC 3589, Internet Engineering Task Force, September 2003.
- [211] R. Ludwig and M. Meyer. The Eifel Detection Algorithm for TCP. RFC 3522, Internet Engineering Task Force, April 2003.
- [212] R. Mahy. A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP). RFC 3842, Internet Engineering Task Force, August 2004.
- [213] R. Mahy, B. Biggs, and R. Dean. The Session Initiation Protocol (SIP) Replaces Header. RFC 3891, Internet Engineering Task Force, September 2004.
- [214] A. Mankin, S. Bradner, R. Mahy, D. Willis, J. Ott, and B. Rosen. Change Process for the Session Initiation Protocol (SIP). RFC 3427, Internet Engineering Task Force, December 2002.
- [215] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP). RFC 2408, Internet Engineering Task Force, November 1998.
- [216] R. Moats. URN Syntax. RFC 2141, Internet Engineering Task Force, May 1997.
- [217] P.V. Mockapetris. Domain names – concepts and facilities. RFC 1034, Internet Engineering Task Force, November 1987.
- [218] NENA. Network Interfaces for E9-1-1 and Emerging Technologies. NENA Technical Information Document TID 07-503, National Emergency Number Association, September 2002.
- [219] A. Niemi. Session Initiation Protocol (SIP) Extension for Event State Publication. RFC 3903, Internet Engineering Task Force, October 2004.
- [220] A. Niemi. An Extension to Session Initiation Protocol (SIP) Events for Conditional Event Notification. Internet-Draft draft-ietf-sip-subnot-etags-02, Internet Engineering Task Force, February 2008. Work in progress.
- [221] A. Niemi, J. Arkko, and V. Torvinen. Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA). RFC 3310, Internet Engineering Task Force, September 2002.

- [222] A. Niemi, M. García-Martín, and G. Arne Sandbakken. Multi-party Instant Message (IM) Sessions Using the Message Session Relay Protocol (MSRP). Internet-Draft draft-ietf-simple-chat-02, Internet Engineering Task Force, February 2008. Work in progress.
- [223] A. Niemi, M. Lonnfors, and E. Leppanen. Publication of Partial Presence Information. Internet-Draft draft-ietf-simple-partial-publish-07, Internet Engineering Task Force, February 2008. Work in progress.
- [224] O. Novo, G. Camarillo, D. Morgan, and R. Even. Conference Information Data Model for Centralized Conferencing (XCON). Internet-Draft draft-ietf-xcon-common-data-model-10, Internet Engineering Task Force, March 2008. Work in progress.
- [225] Open Geospatial Consortium, Inc. OpenGIS Geography Markup Language (GML) Encoding Standard. OGC 07-036 Version 3.2.1, Open Geospatial Consortium, Inc., August 2007.
- [226] Open Mobile Alliance. <http://www.openmobilealliance.org>.
- [227] Open Mobile Alliance. OMA Provisioning Architecture Overview Version 1.1. TS, Open Mobile Alliance, November 2002.
- [228] Open Mobile Alliance. Enabler Release Definition for IMS in OMA Version 1.0. TS, Open Mobile Alliance, February 2005.
- [229] Open Mobile Alliance. Enabler Release Definition for Push to Talk Over Cellular Version 1.0. TS, Open Mobile Alliance, April 2005.
- [230] Open Mobile Alliance. IMS in OMA Version 1.0. Candidate Enabler Release, Open Mobile Alliance, February 2005.
- [231] Open Mobile Alliance. OMA Device Management Protocol 1.2. TS, Open Mobile Alliance, March 2005.
- [232] Open Mobile Alliance. Push to Talk Over Cellular Version 1.0. Candidate Enabler Release, Open Mobile Alliance, May 2005.
- [233] Open Mobile Alliance. Push to Talk Over Cellular Version 1.0 – Architecture. TS, Open Mobile Alliance, April 2005.
- [234] Open Mobile Alliance. Push to Talk Over Cellular Version 1.0 – Control Plane Specification. TS, Open Mobile Alliance, April 2005.
- [235] Open Mobile Alliance. Push to Talk Over Cellular Version 1.0 – Requirements. TS, Open Mobile Alliance, March 2005.
- [236] Open Mobile Alliance. Push to Talk Over Cellular Version 1.0 – User Plane Version. TS, Open Mobile Alliance, April 2005.
- [237] Open Mobile Alliance. Push to Talk Over Cellular Version 1.0 – XDM Specification. TS, Open Mobile Alliance, April 2005.
- [238] Open Mobile Alliance. Utilization of IMS Capabilities Version 1.0 – Architecture. TS, Open Mobile Alliance, February 2005.
- [239] Open Mobile Alliance. Utilization of IMS Capabilities Version 1.0 – Requirements. TS, Open Mobile Alliance, February 2005.
- [240] Open Mobile Alliance. Converged IP Messaging Requirements 1.0. Draft, Open Mobile Alliance, September 2007.
- [241] Open Mobile Alliance. Device Management Version 1.2. Approved Enabler Release, Open Mobile Alliance, September 2007.
- [242] Open Mobile Alliance. Enabler Release Definition for SIMPLE IM 1.0. Candidate Enabler Release, Open Mobile Alliance, August 2007.

- [243] Open Mobile Alliance. Push to Talk Over Cellular Version 2.0. Candidate Enabler Release, Open Mobile Alliance, December 2007.
- [244] Open Mobile Alliance. XML Document Management Version 2.0. Candidate Enabler Release, Open Mobile Alliance, July 2007.
- [245] Open Mobile Alliance. XML Document Management (XDM) Specification. Candidate Enabler Release, Open Mobile Alliance, July 2007.
- [246] Open Mobile Alliance. Presence Simple Version 2.0. Draft Enabler Release, Open Mobile Alliance, March 2008.
- [247] J. Peterson. A Privacy Mechanism for the Session Initiation Protocol (SIP). RFC 3323, Internet Engineering Task Force, November 2002.
- [248] J. Peterson. Common Profile for Presence (CPP). RFC 3859, Internet Engineering Task Force, August 2004.
- [249] J. Peterson. Session Initiation Protocol (SIP) Authenticated Identity Body (AIB) Format. RFC 3893, Internet Engineering Task Force, September 2004.
- [250] J. Peterson. A Presence-based GEOPRIV Location Object Format. RFC 4119, Internet Engineering Task Force, December 2005.
- [251] J. Peterson and C. Jennings. Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP). RFC 4474, Internet Engineering Task Force, August 2006.
- [252] D. Piper. The Internet IP Security Domain of Interpretation for ISAKMP. RFC 2407, Internet Engineering Task Force, November 1998.
- [253] J. Polk and B. Rosen. Location Conveyance for the Session Initiation Protocol. Internet-Draft draft-ietf-sip-location-conveyance-10, Internet Engineering Task Force, February 2008. Work in progress.
- [254] J. Polk, J. Schnizlein, and M. Linsner. Dynamic Host Configuration Protocol Option for Coordinate-based Location Configuration Information. RFC 3825, Internet Engineering Task Force, July 2004.
- [255] J. Postel. User Datagram Protocol. RFC 0768, Internet Engineering Task Force, August 1980.
- [256] J. Postel. Internet Protocol. RFC 0791, Internet Engineering Task Force, September 1981.
- [257] J. Postel. Transmission Control Protocol. RFC 0793, Internet Engineering Task Force, September 1981.
- [258] R. Price, C. Bormann, J. Christoffersson, H. Hanu, Z. Liu, and J. Rosenberg. Signaling Compression (SigComp). RFC 3320, Internet Engineering Task Force, January 2003.
- [259] B. Ramsdell. S/MIME Version 3 Message Specification. RFC 2633, Internet Engineering Task Force, June 1999.
- [260] C. Rigney, A. Rubens, W. Simpson, and S. Willens. Remote Authentication Dial In User Service (RADIUS). RFC 2058, Internet Engineering Task Force, January 1997.
- [261] C. Rigney, A. Rubens, W. Simpson, and S. Willens. Remote Authentication Dial In User Service (RADIUS). RFC 2138, Internet Engineering Task Force, April 1997.
- [262] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, Internet Engineering Task Force, June 2000.
- [263] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, Internet Engineering Task Force, April 1992.
- [264] A. B. Roach. Session Initiation Protocol (SIP) – Specific Event Notification. RFC 3265, Internet Engineering Task Force, June 2002.

- [265] A. B. Roach, B. Campbell, and J. Rosenberg. A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists. RFC 4662, Internet Engineering Task Force, August 2006.
- [266] B. Rosen. Dial String Parameter for the Session Initiation Protocol Uniform Resource Identifier. RFC 4967, Internet Engineering Task Force, July 2007.
- [267] J. Rosenberg. The Session Initiation Protocol (SIP) UPDATE Method. RFC 3311, Internet Engineering Task Force, October 2002.
- [268] J. Rosenberg. A Presence Event Package for the Session Initiation Protocol (SIP). RFC 3856, Internet Engineering Task Force, August 2004.
- [269] J. Rosenberg. A Session Initiation Protocol (SIP) Event Package for Registrations. RFC 3680, Internet Engineering Task Force, March 2004.
- [270] J. Rosenberg. A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP). RFC 3857, Internet Engineering Task Force, August 2004.
- [271] J. Rosenberg. A Data Model for Presence. RFC 4479, Internet Engineering Task Force, July 2006.
- [272] J. Rosenberg. A Framework for Conferencing with the Session Initiation Protocol (SIP). RFC 4353, Internet Engineering Task Force, February 2006.
- [273] J. Rosenberg. Extensible Markup Language (XML) Formats for Representing Resource Lists. RFC 4826, Internet Engineering Task Force, May 2007.
- [274] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. Internet-Draft draft-ietf-mmusic-ice-19, Internet Engineering Task Force, October 2007. Work in progress.
- [275] J. Rosenberg. Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP). Internet-Draft draft-ietf-sip-gruu-15, Internet Engineering Task Force, October 2007. Work in progress.
- [276] J. Rosenberg. Presence Authorization Rules. RFC 5025, Internet Engineering Task Force, December 2007.
- [277] J. Rosenberg. The Extensible Markup Language (XML) Configuration Access Protocol (XCAP). RFC 4825, Internet Engineering Task Force, May 2007.
- [278] J. Rosenberg. Applying Loose Routing to Session Initiation Protocol (SIP) User Agents (UA). Internet-Draft draft-rosenberg-sip-ua-loose-route-02, Internet Engineering Task Force, January 2008. Work in progress.
- [279] J. Rosenberg, G. Camarillo, and D. Willis. A Framework for Consent-based Communications in the Session Initiation Protocol (SIP). Internet-Draft draft-ietf-sip-consent-framework-04, Internet Engineering Task Force, January 2008. Work in progress.
- [280] J. Rosenberg, R. Mahy, and P. Matthews. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). Internet-Draft draft-ietf-behave-turn-07, Internet Engineering Task Force, February 2008. Work in progress.
- [281] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for (NAT) (STUN). Internet-Draft draft-ietf-behave-rfc3489bis-15, Internet Engineering Task Force, February 2008. Work in progress.
- [282] J. Rosenberg, J. Peterson, H. Schulzrinne, and G. Camarillo. Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP). RFC 3725, Internet Engineering Task Force, April 2004.
- [283] J. Rosenberg and H. Schulzrinne. An Offer/Answer Model with Session Description Protocol (SDP). RFC 3264, Internet Engineering Task Force, June 2002.

- [284] J. Rosenberg and H. Schulzrinne. Reliability of Provisional Responses in Session Initiation Protocol (SIP). RFC 3262, Internet Engineering Task Force, June 2002.
- [285] J. Rosenberg and H. Schulzrinne. Session Initiation Protocol (SIP): Locating SIP Servers. RFC 3263, Internet Engineering Task Force, June 2002.
- [286] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force, June 2002.
- [287] J. Rosenberg, H. Schulzrinne, and P. Kyzivat. Caller Preferences for the Session Initiation Protocol (SIP). RFC 3841, Internet Engineering Task Force, August 2004.
- [288] J. Rosenberg, H. Schulzrinne, and P. Kyzivat. Indicating User Agent Capabilities in the Session Initiation Protocol (SIP). RFC 3840, Internet Engineering Task Force, August 2004.
- [289] J. Rosenberg, H. Schulzrinne, and O. Levin. A Session Initiation Protocol (SIP) Event Package for Conference State. RFC 4575, Internet Engineering Task Force, August 2006.
- [290] J. Rosenberg, H. Schulzrinne, and R. Mahy. An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP). RFC 4235, Internet Engineering Task Force, November 2005.
- [291] J. Rosenberg and J. Urpalainen. An Extensible Markup Language (XML) Document Format for Indicating A Change in XML Configuration Access Protocol (XCAP) Resources. Internet-Draft draft-ietf-simple-xcap-diff-08, Internet Engineering Task Force, February 2008. Work in progress.
- [292] K. Rosenbrock, R. Sanmugam, S. Bradner, and J. Klensin. 3GPP-IETF Standardization Collaboration. RFC 3113, Internet Engineering Task Force, June 2001.
- [293] R. Sahita, S. Hahn, K. Chan, and K. McCloghrie. Framework Policy Information Base. RFC 3318, Internet Engineering Task Force, March 2003.
- [294] J. Schaad and R. Housley. Advanced Encryption Standard (AES) Key Wrap Algorithm. RFC 3394, Internet Engineering Task Force, September 2002.
- [295] H. Schulzrinne. The tel URI for Telephone Numbers. RFC 3966, Internet Engineering Task Force, December 2004.
- [296] H. Schulzrinne. CIPID: Contact Information for the Presence Information Data Format. RFC 4482, Internet Engineering Task Force, July 2006.
- [297] H. Schulzrinne. Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information. RFC 4776, Internet Engineering Task Force, November 2006.
- [298] H. Schulzrinne. Timed Presence Extensions to the Presence Information Data Format (PIDF) to Indicate Status Information for Past and Future Time Intervals. RFC 4481, Internet Engineering Task Force, July 2006.
- [299] H. Schulzrinne. A Uniform Resource Name (URN) for Emergency and Other Well-Known Services. RFC 5031, Internet Engineering Task Force, January 2008.
- [300] H. Schulzrinne and S. Casner. RTP Profile for Audio and Video Conferences with Minimal Control. RFC 3551, Internet Engineering Task Force, July 2003.
- [301] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, Internet Engineering Task Force, July 2003.
- [302] H. Schulzrinne, V. Gurbani, P. Kyzivat, and J. Rosenberg. RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF). RFC 4480, Internet Engineering Task Force, July 2006.
- [303] H. Schulzrinne and S. Petrack. RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals. RFC 2833, Internet Engineering Task Force, May 2000.

- [304] H. Schulzrinne, H. Tschofenig, J. Morris, J. Cuellar, J. Polk, and J. Rosenberg. Common Policy: A Document Format for Expressing Privacy Preferences. RFC 4745, Internet Engineering Task Force, February 2007.
- [305] H. Schulzrinne and B. Volz. Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers. RFC 3319, Internet Engineering Task Force, July 2003.
- [306] R. Sparks. The Session Initiation Protocol (SIP) Refer Method. RFC 3515, Internet Engineering Task Force, April 2003.
- [307] R. Sparks. The Session Initiation Protocol (SIP) Referred-By Mechanism. RFC 3892, Internet Engineering Task Force, September 2004.
- [308] R. Stewart, Q. Xie, K. Morneau, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960, Internet Engineering Task Force, October 2000.
- [309] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson. Presence Information Data Format (PIDF). RFC 3863, Internet Engineering Task Force, August 2004.
- [310] M. Thomson and J. Winterbottom. Revised Civic Location Format for Presence Information Data Format Location Object (PIDF-LO). RFC 5139, Internet Engineering Task Force, February 2008.
- [311] J. Urpalainen. An Extensible Markup Language (XML) Patch Operations Framework Utilizing XML Path Language (XPath) Selectors. Internet-Draft draft-ietf-simple-xml-patch-ops-04, Internet Engineering Task Force, November 2007. Work in progress.
- [312] J. Urpalainen. An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Diff Event Package. Internet-Draft draft-ietf-sip-xcapedevent-02, Internet Engineering Task Force, April 2008. Work in progress.
- [313] A. Vemuri and J. Peterson. Session Initiation Protocol for Telephones (SIP-T): Context and Architectures. RFC 3372, Internet Engineering Task Force, September 2002.
- [314] WAP Forum. WAP Architecture. Recommendation WAP Architecture, Wireless Application Protocol Forum, July 2001.
- [315] D. Willis and A. Allen. Requesting Answering Modes for the Session Initiation Protocol (SIP). Internet-Draft draft-ietf-sip-answermode-06, Internet Engineering Task Force, September 2007. Work in progress.
- [316] D. Willis and B. Hoeneisen. Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts. RFC 3327, Internet Engineering Task Force, December 2002.
- [317] D. Willis and B. Hoeneisen. Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration. RFC 3608, Internet Engineering Task Force, October 2003.
- [318] T. Ylonen and C. Lonwick. The Secure Shell (SSH) Protocol Architecture. RFC 4251, Internet Engineering Task Force, January 2006.
- [319] D. Yon and G. Camarillo. TCP-Based Media Transport in the Session Description Protocol (SDP). RFC 4145, Internet Engineering Task Force, September 2005.

Index

- 3G Americas, 14
3GPP, xxi, xxiii–xxv, xxvii–xxix, xxxi, xxxiii, 9, 10, 14–20, 22, 25–32, 36–38, 40, 41, 43–46, 48, 49, 51–53, 111–113, 115, 117, 122, 125, 136, 140, 144, 145, 150–152, 167–171, 180, 184, 188, 189, 191, 192, 196, 209–213, 229, 232, 235, 236, 238, 248, 249, 251–253, 256, 258, 260, 265, 266, 271, 272, 274, 293, 294, 297, 298, 301–303, 306, 329, 331, 345, 348, 353, 356, 357, 369, 391, 392, 437, 443, 477, 478, 499, 502, 510, 524, 528–530, 536, 539, 540, 542–545, 548, 549, 551, 553, 558, 559, 577–580, 582, 584–586
PCG, 15
TR, 15, 113, 293, 303, 510, 528, 584, 585
TS, 15, 26–28, 31, 36–38, 40, 46, 48, 52, 111–113, 125, 144, 169–171, 180, 184, 189, 191, 192, 196, 213, 229, 238, 248, 249, 251, 252, 256, 258, 260, 265, 271, 272, 274, 293, 306, 329, 331, 345, 348, 357, 391, 392, 443, 477, 478, 499, 502, 529, 530, 536, 539, 540, 542–545, 548, 549, 551, 553, 558, 559, 577–580, 582, 584–587
TSG, 16–18
3GPP2, xxi, xxv, xxvii, xxix, xxxi, xxxiii, 9, 10, 15–20, 22, 48, 49, 144, 192, 297, 348, 356, 524
SC, 16, 17
AAA, 11, 30, 169, 215–218, 221, 222, 227, 229, 235, 244, 247, 453
ADSL, xxvii, 29, 31, 49, 52, 112, 188, 314
AIB (Authenticated Identity Body), 285, 287, 288, 291
Anchoring, 560, 561, 563–569, 571, 574
ANSI, 15, 314
ARIB, 14, 16
BICC, 29, 30, 39, 40, 191, 584
Cable modem, 29
CAMEL, 26, 36, 37, 170, 171, 561, 562, 564–567, 570, 571, 574, 580, 582
CAP (CAMEL Application Part), 36, 171, 260, 582
CCSA, 16
CDMA 2000, 15, 17, 348
Certificates, 280–282, 284, 285, 303, 310
CGI, 30, 167, 234
Charging, xxi, xxviii, 7, 15, 26, 32, 34, 54, 121, 140, 145, 146, 150, 230, 234, 235, 239, 242–244, 251–258, 260–263, 271, 479, 528, 546, 547, 553, 555, 579, 582–584, 587
ABMF (Account Balance Management Function), 260
CCF (Charging Collector Function), 235, 256
CDF (Charging Data Function), 251, 254, 256, 258, 528
CDR (Charging Data Record), 251, 252, 258, 528
CDR (Charging Data Records), 252
CGF (Charging Gateway Function), 251, 252
CTF (Charging Trigger Function), 251, 258, 261, 262, 265
EBCF (Event Based Charging Function), 260
ECUR (Event Charging with Unit Reservation), 261, 262
ICID (IMS Charging Identity), 253, 254
IEC (Immediate Event Charging), 261, 262, 354
IOI (Inter Operator Identifier), 253, 254

- OCF (Online Charging Function), 260–262, 264, 265
 OFCS (Offline Charging System), 244, 252
 PCRF (Policy and Charging Rule Function), 243–249, 256, 271
 RF (Rating Function), 260, 265
 SBCF (Session Based Charging Function), 260
 SCUR (Session Charging with Unit Reservation), 261, 262
 CIF (Common Intermediate Format), 355
 Circuit-switched, xxi, 6, 8, 25–29, 37, 39, 40, 46, 47, 118, 149, 150, 168, 192, 196, 216, 238, 239, 241, 297, 306, 329–331, 335, 341, 346, 523, 559–564, 567, 569–571, 574, 576, 582
 CMS (Cryptographic Message Syntax), 282, 284, 286
 Codecs
 - Audio
 - AMR, 28, 40, 144, 145, 154, 159, 345–353, 357, 524
 - AMR-WB, 348, 357
 - CELP, 347, 353
 - EVRC, 144, 348, 524
 - G.711, 28, 40, 60, 61, 145, 246, 342, 362, 363
 - LPC (Linear Predictive Coding), 343, 344, 347, 353
 - LTP (Long-Term Predictors), 344
 - Video
 - Chrominance, 355, 356
 - DivX, 354
 - H.261, 60, 61, 354, 362
 - H.263, 144, 154, 156, 159, 353, 355–357
 - Luminance, 355, 356
 - MPEG, 70, 144, 156, 159, 354
 - XviD, 354- COPS, xxxii, 19, 243
 - COPS-PR, 19
- CPIM (Common Presence and Instant Messaging), 456, 459, 460, 474
- DCCP, xxxii, 359–361, 364
- DCT, 356
- DHCP, 19, 113, 115–117, 313, 314, 322, 336
 - DHCPv6, 113, 115, 116
- Diameter, xxiii, xxxi, 15, 19, 30, 31, 33–37, 52, 120–123, 128, 137, 151, 169, 184, 216–225, 227–232, 234, 236, 238–244, 249–254, 258, 260, 264–266, 294, 295, 298, 306–308, 336, 444, 447, 582, 586
- AVPs
 - 3GPP-RAT-Type, 251, 253
 - Abort-Cause, 250
 - Acceptable-Service-Info, 250
 - Access-Network-Charging-Address, 250, 253
 - Access-Network-Charging-Identifier, 250, 252, 253
 - Access-Network-Charging-Identifier-Value, 250, 253
 - Accounting-Record-Number, 259
 - Accounting-Record-Type, 258, 259
 - Acct-Application-Id, 225, 259
 - Acct-Interim-Interval, 259
 - ActualTime, 265
 - AF-Application-Identifier, 250
 - AF-Charging-Identifier, 250, 253
 - AllowedUnits, 265
 - Auth-Session-State, 225
 - Authorization-Lifetime, 224
 - BasicPrice, 265
 - BasicPriceTimeStamp, 265
 - Bearer-Control-Mode, 252
 - Bearer-Identifier, 252
 - Bearer-Operation, 252
 - Bearer-Usage, 252
 - BeginTime, 265
 - BillingInfo, 265
 - Called-Station-ID, 253
 - CC-Request-Number, 253
 - CC-Request-Type, 253, 262
 - Charging-Information, 234, 235, 253
 - Charging-Rule-Report, 252
 - Codec-Data, 250
 - Confidentiality, 234, 235
 - ConsumedUnits, 265
 - ConsumedUnitsAfterTariffSwitch, 265
 - Counter, 265
 - CounterChange, 265
 - CounterChangeForFirstChargeable-TimeUnit, 265
 - CounterChangeForFirstChargeable-TimeUnitAfterSwitch, 265

- CounterChangePerChargeableVolumeUnit, 265
CounterChangePerChargeableVolumeUnitAfterSwitch, 265
CounterChangePerConsumedService-Unit, 265
CounterChangePerSession, 265
CounterChangePerSubsequentChargeableTimeUnit, 265
CounterChangePerSubsequentChargeableTimeUnitAfter-Switch, 265
CounterExpiryDate, 265
CounterID, 265
CounterPrice, 265
CounterTariff, 265
CounterThreshold, 265
CounterType, 265
CounterValue, 265
CounterValueBegin, 265
CounterValueChange, 265
CounterValueEnd, 265
Current-Location, 241
Data-Reference, 241
Deregistration-Reason, 234, 235
Destination-Realm, 224, 259
DestinationID, 265
DestinationIDData, 265
DestinationIDType, 265
EParameterE1, 266
EParameterE2, 266
EParameterE3, 266
EParameterE4, 266
EParameterE5, 266
EParameterE6, 266
EParameterE7, 266
Error-Reporting-Host, 259
Event-Timestamp, 259
Event-Trigger, 252
ExpiryTime, 266
Extension, 266
FirstRequest, 266
Flow-Description, 250, 253
Flow-Number, 250
Flow-Usage, 250
Flows, 250, 253
Flows-Status, 250, 253
Framed-IP-Address, 251, 253
Framed-IPv6-Prefix, 251, 253
Guaranteed-Bitrate-DL, 252
Guaranteed-Bitrate-UL, 252
ImpactOnCounter, 266
Integrity-Key, 234, 235
IP-CAN-Type, 251, 252
Mandatory-Capability, 233, 234
Max-Requested-Bandwidth-DL, 250, 253
Max-Requested-Bandwidth-UL, 250, 253
Media-Component-Description, 250
Media-Component-Number, 250
Media-Sub-Component, 250
Media-Type, 250
Metering-Method, 252
MinimalRequestedUnits, 266
MonetaryQuota, 266
MonetaryTariff, 266
MonetaryTariffAfterValidUnits, 266
MSISDN, 241
Network-Request-Support, 252
NextMonetaryTariff, 266
Offline, 252
Online, 252
Optional-Capability, 233, 234
Origin-Host, 224, 259
Origin-Realm, 224, 259
Origin-State-Id, 259
PCC-Rule-Status, 252
Precedence, 252
Price, 266
Primary-Charging-Collection-Function-Name, 234
Primary-Event-Charging-Function-Name, 234
Proxy-Host, 225
Proxy-Info, 225
Proxy-State, 225
Public-Identity, 151, 233, 234, 241
QoS-Class-Identifier, 252
QoS-Information, 252
Rating-Group, 253
Reason-Code, 234, 235
Reason-Info, 234, 235
Reporting-Level, 252
Requested-Domain, 241

- RequestedCounter, 266
 RequestedUnits, 266
 RequestSubType, 266
 Reservation-Priority, 251
 Result-Code, 224, 230, 259
 Route-Record, 225, 259
 RR-Bandwidth, 250
 RS-Bandwidth, 250
 Secondary-Charging-Collection-Function-Name, 234
 Secondary-Event-Charging-Function-Name, 234
 Server-Assignment-Type, 234, 235
 Server-Capabilities, 233, 234
 Server-Name, 151, 233, 234, 241
 Service-Context-Id, 259
 Service-ID, 452
 Service-Identifier, 253, 266
 Service-Indication, 241
 Service-Info-Status-AVP, 250
 Service-Information, 259
 Service-Rating, 266
 Service-URN, 250
 ServiceInformation, 266
 Session-ID, 224, 259
 SetCounterTo, 266
 SIP-Auth-Data-Item, 234
 SIP-Authenticate, 234, 235
 SIP-Authentication-Context, 234, 235
 SIP-Authentication-Scheme, 234, 235
 SIP-Authorization, 234, 235
 SIP-Forking-Indication, 250
 SIP-Item-Number, 234, 235
 SIP-Number-Auth-Items, 234
 Specific-Action, 250
 Subs-Req-Type, 241
 Subscription-Id, 253, 266
 Subscription-Id-Data, 266
 Subscription-Id-Type, 266
 Tariff-Switch-Definition, 264
 TariffSwitchTime, 266
 TFT-Filter, 252
 TFT-Packet-Filter-Information, 252
 ToS-Traffic-Class, 252
 User-Authorization-Type, 234, 235
 User-Data, 234, 236, 239, 241
 User-Data-Already-Available, 234
 User-Data-Request-Type, 234
 User-Equipment-Info, 253
 User-Identity, 241
 User-Name, 224, 236, 259
 ValidUnits, 266
 Vendor-ID, 220, 221, 225, 232
 Vendor-Specific-Application-ID, 225
 Visited-Network-Identifier, 232–234
Commands
 AAA, 250
 AAR, 244, 250
 ACA, 222, 223, 258, 259, 264
 ACR, 222, 223, 258–260
 ASA, 222, 223, 250, 264
 ASR, 222, 223, 250, 264
 CCA, 245, 251, 262, 264
 CCR, 245, 251, 262, 264
 CEA, 222, 223, 264
 CER, 222, 223, 264
 DPA, 222, 223, 264
 DPR, 222, 223, 264
 DWA, 222, 223, 264
 DWR, 222, 223, 264
 LIA, 151, 229, 231, 232
 LIR, 151, 229, 231, 232
 MAA, 122, 123, 128, 229–231, 295, 306, 308
 MAR, 122, 229–231, 295, 306, 308
 PNA, 239–241
 PNR, 239–241
 PPA, 229, 232, 233
 PPR, 229, 232, 233, 236
 PRQ, 264
 PRS, 264
 PUA, 239, 240
 PUR, 239, 240
 RAA, 222, 224, 245, 250, 251, 264
 RAR, 222, 224, 245, 250, 251, 264
 RTA, 229, 232, 233
 RTR, 229, 232, 233
 SAA, 229–232, 236, 254, 306, 308
 SAR, 123, 229–232, 236, 306, 308
 SNA, 239, 240
 SNR, 239, 240
 STA, 222, 224, 250
 STR, 222, 224, 250
 SUQ, 264
 SUS, 264
 TRQ, 264
 TRS, 264
 UAA, 120–122, 229–231, 306
 UAR, 120, 122, 128, 229–231, 235, 306

- UDA, 239, 307
UDR, 239, 307
DiffServ, 267, 269, 271, 275
DLS, 6, 114, 188, 189, 307, 308
DNS, xxiii, 18, 19, 34, 35, 115–117, 119, 120, 122, 127, 128, 142, 148–150, 195, 230, 281, 310, 314
NAPTR, 149, 310
SRV, 310
DoS, 277, 280, 359, 506
DSCP, 269, 270, 275
DSS-1, 189

EDGE, 15, 140
EIA-41, 15
Entity tag, 380–382, 384–386, 439
ENUM, 149
ETSI, xxv, 10, 14, 29, 49, 344, 502, 529, 530, 536, 539, 540, 542–546, 548, 549, 551, 553, 558, 578, 584

Filter Criteria, 123, 146, 148, 151, 173, 174, 180, 181, 183, 184, 186, 187, 211, 228, 232, 236, 238, 240, 444, 447, 477, 513, 514, 532–534, 536, 538–540, 544, 547, 550, 563, 564, 568, 569, 571, 573

GCID (GPRS Charging ID), 258
GERAN, 15, 140
GGSN, 34, 40, 41, 114, 115, 142, 161, 242, 243, 252, 256, 258, 271, 275, 303–305, 445
GMLC (Gateway Mobile Location Center), 331
GPRS, 8, 19, 26, 27, 29, 31, 34, 40, 112–115, 142, 161, 197, 258, 260, 272, 274, 293, 294, 303, 304, 445, 559
GRUU, 96–100, 206, 208, 209, 322, 325, 332, 334
GSM, xxi, 14, 15, 25, 26, 29, 32, 36, 40, 42, 43, 45, 46, 53, 112, 119, 130, 131, 140, 149, 171, 312, 318, 344–346, 559
GSM Association, 14
GTP (GPRS Tunneling Protocol), 260

H.248, 31, 37, 40, 186, 191, 192, 480, 489, 493, 564, 586
H.323, 30, 53, 492
HELD (HTTP Enabled Location Delivery), 315, 316, 318, 319, 322, 325

HFC (Hybrid Fiber Coax), 29
HLR, 32, 53, 114, 446, 571
HTML, 453, 456, 459, 460, 464
HTTP, xxviii, xxix, 30, 67, 92, 95, 115, 167, 168, 186, 188, 189, 235, 277–279, 293–298, 300, 302, 303, 313, 315, 316, 318–320, 359, 360, 371–373, 376–382, 385, 389–402, 419, 423, 448, 450, 455, 464, 466, 530, 532–534, 544, 558
Methods
 DELETE, 380, 558
 GET, 316, 318, 320, 380–382, 391–397, 534, 544, 558
 POST, 315, 316, 319, 372, 397–402
 PUT, 372, 373, 378, 379, 381, 382, 448, 558
HTTP Digest Access Authentication, xxviii, 188, 189, 235, 277–279, 293–298, 300, 302, 303, 392

IARI (IMS Application Reference Identifier), xxviii, 212
ICE, xxviii, 38, 105, 106, 109, 213
ICSI (IMS Communication Service Identifier), xxviii, 209–212
IETF, xxi, xxiii–xxv, xxix, xxxi, xxxii, 9–12, 18–20, 22, 23, 29–31, 59, 80, 111, 113, 131, 140, 143, 192, 215, 216, 220, 225, 229, 247, 274, 314, 319, 359, 360, 369, 372, 374, 376, 412, 427, 435, 437, 454, 479, 483, 499, 504, 548, 577
 BCP, 12, 13
 Draft Standard, 13
 Historic, 13
 IAB, 11
 IESG, 11, 12
 Informational, 13
 Internet Standard, 13
 Proposed Standard, 13
 Working Groups
 MMUSIC, 20, 483
 ROHC, 20
 SIMPLE, 11, 20, 22, 443, 502, 584, 585, 587
 SIP, 19, 20
 SIPPING, xxi, xxiv, 11, 20, 483–486, 488–490, 492, 499, 504
 XCON, 483, 484, 489–493, 495, 499, 504

- working groups
 - SIPPING, xxv
- IKE, 287, 291
- IMS
 - IMSI (International Mobile Subscriber Identifier), 43, 46, 119, 126–128, 184, 300, 303–306
 - Interfaces
 - Bx, 252
 - Cx, 19, 188, 227–229, 232–234, 236, 241, 582
 - Dx, 227–229, 582
 - e2, 52, 336
 - Ga, 252, 258
 - Go, 19, 243, 582
 - Gq, 52, 582
 - Gq', 52
 - Gx, 40, 243, 249, 251–253, 582
 - Gy, 243
 - Gz, 244, 252
 - ISC, 167, 170, 171, 260, 444, 561–563
 - Ix, 38
 - Ma, 168, 169, 561, 562
 - Mb, 192
 - Mg, 191
 - Ml, 331
 - Mn, 192, 582
 - Mq, 336
 - Mx, 38
 - Rc, 260
 - Re, 260, 264–266
 - Rf, 251, 258, 528
 - Ro, 19, 260, 264, 265, 528
 - Rx, 40, 52, 243, 248–251, 266, 582
 - Sh, 19, 169, 170, 186, 227, 238, 239, 241, 444, 561–563, 582
 - Si, 171
 - Sp, 243
 - Ut, 168, 169, 444, 542, 543, 557, 581, 585
 - V3, 563
 - Za, 309, 310
 - Zb, 309
 - Nodes
 - ABMF, 260
 - AF, 243, 249, 271
 - AS, 35–37, 44, 51, 53, 54, 123, 131, 135, 151, 167–181, 183, 184, 186, 187, 197, 209, 210, 212, 227, 236, 238–241, 243, 260, 264, 445, 447, 477, 478, 489, 502, 529, 532–540, 542–553, 555, 558, 561, 565, 574
 - BFCP, 37, 39, 51, 150, 192, 251, 309, 331, 333, 571
 - CCF, 235, 256
 - CDF, 251, 254, 256, 258, 528
 - CGF, 251, 252
 - CSAF, 561, 562, 564–567, 569, 571, 574, 575
 - CTF, 251, 258, 261, 262, 265
 - DSF, 561, 562, 568, 569, 571
 - DTF, 561, 563–565, 567–576
 - E-CSCF, 331–333
 - EBCF, 260
 - ECF, 235, 261
 - HSS, 32–37, 43, 51, 53, 117, 120–124, 128, 131, 137, 148, 151, 169–171, 181, 184, 186, 188, 189, 209, 218, 227–236, 238–241, 254, 293–296, 298, 300, 301, 304, 306, 308, 309, 446, 447, 502, 561–563
 - I-CSCF, 33–35, 38, 53, 119–125, 127, 128, 130, 137, 138, 146, 148, 150, 151, 158, 168, 169, 173, 176, 177, 188, 192, 210, 227–233, 235, 251, 295, 296, 298, 305, 306, 308, 334, 447, 502, 561, 562, 564, 565, 571, 574
 - IBCF, 38, 53, 54, 212, 213, 309, 310, 331
 - IM-SSF AS, 36, 37, 170–172, 184
 - LRF, 331, 333, 336
 - MGCF, 32, 40, 51, 191–194, 251, 333, 499, 551, 560, 563–566, 570–572, 574–576, 582
 - MGW, 32, 40, 51, 192, 560, 563–565, 570, 571, 574, 582
 - MRF, 37, 184, 480, 534
 - MRFC, 31, 37, 51, 184, 186, 251, 260, 264, 479–481, 499, 502, 534, 535
 - MRFP, 31, 37, 51, 186, 479–481, 499, 502, 534, 535, 586
 - OSA-SCS AS, 36, 37, 170, 171, 238
 - P-CSCF, 33, 34, 40, 41, 48, 52, 53, 112, 113, 115–119, 122–125, 127, 128, 130, 131, 133, 135, 137, 138, 140, 141, 143, 145, 146, 148, 150, 152, 154, 156, 158, 172, 174, 176, 177, 187–189, 210, 213, 228, 230, 232, 233, 235, 243–249, 251, 253,

- 254, 256, 258, 271–274, 293, 296, 298, 300–303, 305, 307, 308, 313, 329, 331–336, 450, 460, 478, 480, 540, 542, 550, 551, 563, 567, 571, 573, 576
PCEF, 243–245, 249, 271
PCRF, 243–249, 256, 271
PDF, 34
RF, 260, 265
S-CSCF, 32–38, 43, 120–126, 128, 130, 131, 133, 135–138, 145, 146, 148–153, 158, 167, 170–177, 180, 181, 184, 186–189, 191, 192, 206, 208–211, 218, 227–236, 238, 240, 246, 248, 251, 253, 254, 256, 258, 260, 293–296, 298–301, 303, 306, 308, 331, 334, 335, 444, 446, 447, 477, 478, 480, 510, 514, 532–536, 538–540, 542, 546, 547, 550–552, 561–565, 568, 569, 571, 573–576
SBCF, 260
SGW, 32, 39, 191
SLF, 32–34, 53, 117, 131, 227, 228
SPR, 243, 244
THIG, 34, 35
UE, 31, 332, 478, 508, 542, 560
Release
R5, 44, 579–583
R6, 44, 579, 580, 582, 583
R7, 579, 580, 582
IMT-2000, 9
IN, 26
IP-CAN, 40, 112, 113, 115, 116, 142, 161, 166, 197, 204, 293, 294, 303, 306, 331, 334, 335, 452, 559
IPsec, xxxi, 33, 138, 143, 146, 282, 287, 290, 291, 293, 294, 298, 300–303, 309, 310
AH, 287, 289
ESP, 287, 289, 290, 309
IPv4, 37–39, 52–54, 113–117, 192, 195, 196, 213, 217, 220, 221, 269, 270, 304–306, 313, 580
IPv6, xxiv, 14, 16, 18, 19, 37–39, 53, 54, 113–117, 143, 192, 195, 196, 213, 221, 251, 253, 270, 304–306, 313
IPv6 Forum, 14, 16
ISDN, xxviii, 6, 29, 42, 47, 49, 51, 354, 391, 502, 529–531, 557, 558, 581, 584–586
ISIM, 45, 47, 48, 117–119, 125–128, 294, 297–300, 334, 583
ISUP, xxi, 29, 30, 39, 40, 51, 190, 191, 331, 333, 551, 564, 566, 567, 570, 571, 574, 576, 584, 585
ITU, xxi, xxxi, 9, 10, 28–31, 39, 40, 342, 343, 354–357, 584
JPEG, 70, 459
LDAP, 243, 419
Line identifier, 188, 189, 307, 308, 314, 332, 333, 336
LoST (Location-to-Service Translation), 319
LoST (Location-to-Service Translation), 319
LoST (Location-to-Service Translation), 319–322, 325, 326
MAP (Mobile Application Part), 37, 171, 571
MCC (Mobile Country Code), 126, 127
MD5, 235, 278, 296, 299
MEGACO, 31
MGCP, 191
MIME, 70, 71, 92, 282, 323, 325, 378, 380–382, 384, 395, 397, 399, 406, 410, 413, 423, 429, 436, 437, 454, 456, 459–461, 471, 472, 476, 546, 547, 558
MMS (Multimedia Messaging Service), xxxi, 6, 8, 22, 47, 48
MNC (Mobile Network Code), 126, 127
MSC (Mobile Switching Function), 26, 445, 563, 571, 572, 574
MSIN (Mobile Subscriber Identification Number), 126, 127
MSISDN (Mobile Subscriber ISDN), 42, 47, 197, 241, 303, 304, 570
MSRP, 22, 455–470, 473, 474, 476, 479–482, 508–510, 523
Header fields
Byte-Range, 461, 463
Failure-Report, 462–464
From-Path, 460, 465
Message-ID, 461, 463
Status, 463, 464
Success-Report, 462, 463
To-Path, 460, 465, 467
Use-Nickname, 473
Methods
AUTH, 456, 465, 466
NICKNAME, 473
REPORT, 456, 462–464, 467, 468

- SEND, 456, 458–464, 467–470, 476, 480, 481
- MTP, 39
- Multicast, 115, 269, 484, 489
- NAI (Network Access Identifier), 43, 224
- NASS (Network Attachment Subsystem), xxviii, 49, 52, 188, 189, 293, 303, 306–308, 332, 336, 584
- NAT (Network Address Translator), xxviii, 38, 39, 52, 53, 92, 100, 101, 103–108, 113, 196, 213, 456
- NGN (Next Generation Network), xxix, 10, 49–54, 188, 189, 558, 578, 581, 584–586
- NGN (Next Generation Networks), xxv
- OMA (Open Mobile Alliance), xxviii, xxix, 10, 20–23, 117, 389, 390, 394, 395, 399, 443, 450–452, 502–504, 506, 508, 510, 524, 525, 578, 584–587
- OSA (Open Service Access), 36, 37, 170, 227, 238
- Packet-switched, 6–8, 25–28, 46, 47, 118, 196, 238, 239, 241, 297, 300, 559, 581
- PCC (Policy and Charging Control), xxviii, 243, 244, 271, 528
- PCC (Poly and Charging Control), 243–245, 248
- PCM (Pulse Code Modulation), 40, 342, 343
- PDA, 64, 65, 73
- PDP (Policy Decision Point), 243
- PDP Context, 114, 115, 142, 245, 256, 258, 271, 272, 274, 275, 304
- PEP (Policy Enforcement Point), 243
- PHB (Per Hop Behavior), 269
- PoC (Push-to-talk over Cellular), xxviii, xxix, 20, 21, 137, 206, 371, 391, 502–504, 506, 508–528, 580, 584, 587
- Presence
- CIPID, 419, 421, 438
 - PA, 405, 406, 409, 410, 412, 428, 435, 438, 439, 441, 444
 - PIDF, 322, 323, 332, 333, 384, 385, 410, 412–414, 416, 417, 419, 421–424, 426, 428, 431, 434, 436–439, 444, 445, 448, 450, 451
 - PIDF-LO (Location Object), 322, 323, 332, 333, 424, 426, 428
- PNA (Presence Network Agent), 445
- Presentity, 173, 405–409, 413–419, 421–424, 427–435, 438, 439, 441, 443, 444, 447, 448, 452
- PS (Presence Server), 406, 444, 445, 447, 448, 450
- PUA, 405, 412, 428, 429, 439, 444, 445
- RPID, 412, 417–420, 437–439
- Watcher, 88, 173, 374, 406–410, 421, 422, 427–431, 433–438, 441, 443, 444, 446–450, 470, 508
- Private User Identity, 43, 44, 46, 48, 119, 120, 123, 126, 127, 181, 188, 224, 230, 236, 294, 300, 303, 306, 308, 334
- PSAP (Public Safe Answering Point), 320
- PSAP (Public Safe Answering Point), 311–313, 319–323, 325, 326, 329–331, 333, 334
- PSI (Public Service Identity), 43, 44, 168, 209, 501, 502, 560, 564, 571, 573, 574
- PSTN, xxiii, xxviii, 25, 28, 32, 37, 39, 40, 42, 49, 51, 69, 149, 150, 189–192, 195, 330, 331, 333, 341, 342, 391, 499, 502, 529–531, 539, 551, 557, 558, 560, 563, 564, 570, 581, 584–586
- PSTN/ISDN emulation, 49, 51, 584
- PSTN/ISDN simulation, xxviii, 51, 391, 529–531, 557, 558, 581, 585, 586
- PSTN/ISDN Simulation Services, xxviii, 391, 529–531, 557, 558, 581, 585, 586
- ACR, 531, 543, 581, 585
- AoC, 531, 545–548, 581
- CB, 531, 543–545, 581, 585
- CCBS, 531, 548, 549, 581
- CCNR, 531, 548, 549, 581
- CD, 536
- CDIV, 531, 536–539, 580, 585
- CDIVN, 537–539
- CFB, 531, 536
- CFNL, 531, 536, 539
- CFNR, 531, 536
- CFU, 531, 536
- CONF, 502, 531, 539, 581, 585
- CW, 531, 581
- ECT, 531, 553, 555, 581, 585
- HOLD, 531, 551, 552, 555, 581, 585
- ICB, 531, 543, 545
- MCID, 531, 549–551, 581, 585
- MWI, 531, 539–541, 581, 585
- OCB, 531, 544, 545

- OIP, 531, 542, 543, 581, 585
- OIR, 531, 542, 543, 581, 585
- TIP, 531, 543, 581, 585
- TIR, 531, 543, 581, 585
- Public User Identity, 35, 42, 43, 47, 48, 118–120, 125–128, 130, 131, 133, 135, 138, 140, 146, 152, 154, 158, 188, 206, 208, 227, 229, 230, 233, 236, 241, 300, 304–306, 330–335, 391, 398, 447, 448, 538
- QoS, xxi, 7, 8, 27, 52, 84, 85, 145, 156, 158, 159, 161, 198, 200, 204, 243, 247, 252, 267, 271, 274, 275, 579, 580, 582
- RADIUS, 30, 215–218, 220, 222, 304, 306
- RDF (Routing Determination Function), 331
- Resource List, 374, 383, 384, 435–437, 444, 447, 470–472, 478, 587
- RLS (Resource List Server), 384, 385, 435, 436, 444, 446–448, 587
- RSVP, 267–269, 271, 275
- RTCP, xxxii, 31, 38, 113, 258, 354, 361, 363, 364, 508, 523–525
- RTP, xxiii, xxxii, 31, 38, 40, 54, 113, 144, 190, 192, 258, 268, 292, 346, 361–364, 455, 508–510, 523, 559, 560, 570, 574
- S/MIME, 247, 274, 280, 282–285
- SCTP, 39, 81, 138, 190, 191, 217, 222, 359, 360, 454, 455
- SDP, xxiii
- SDP (Session Description Protocol), xxviii, xxxi, 15, 38, 52, 59–62, 70, 71, 84, 85, 91–94, 113, 142–146, 148, 154, 156, 158, 159, 161, 165, 166, 178, 180, 183, 186, 198–200, 204, 206, 213, 218, 236, 243, 246, 247, 249, 258, 271, 272, 284, 287, 292, 322, 323, 325, 333, 360, 363, 364, 450, 456, 458–460, 465, 467, 473, 476, 479, 483, 494, 495, 506, 514, 528, 534, 535, 547, 551, 552, 580, 584
- Servlets, 30, 167
- SGSN, 40, 41, 114, 115, 161, 253, 304, 445
- SHA1, 278
- SigComp, 90–93, 131, 138, 143, 146, 158, 189, 450
- SIM (Subscriber Identity Module), 43, 45, 46, 48, 188, 297, 298
- SIP
 - dialog, 75, 78–80, 84, 125, 138, 142, 146, 148, 150, 151, 154, 159, 175, 181, 195, 209, 212, 454, 530, 532, 534–536, 548, 549, 555, 563, 564, 574–576
 - Entities
 - B2BUA, 35, 38, 172, 178–180, 260, 521, 535, 536, 546, 547, 552, 561, 564, 565, 568, 569, 571, 574
 - Proxy, 33–36, 40, 63–66, 70, 72–75, 78, 80–82, 84, 90, 96, 99, 100, 112, 115, 116, 119, 124, 138, 141, 142, 146, 148, 149, 154, 172, 174–178, 181, 186, 192, 195, 217, 218, 220, 247, 259, 272, 274, 277, 278, 280–282, 287, 291, 309, 312–314, 319, 321–323, 325–327, 389–395, 397–399, 401–403, 406, 448, 454, 455, 457, 508–510, 536, 562
 - Redirect Server, 36, 66, 67, 172, 177, 178, 277
 - Registrar, 35, 63–66, 72, 73, 80, 97, 98, 130, 133, 206, 277
 - UA, 35, 81, 86, 90, 92, 96–98, 130, 138, 172–175, 188, 326, 454, 455, 538
 - Event Packages
 - conference, 476, 492, 495, 502, 505, 539
 - dialog, 548, 549
 - message-summary, 86, 540, 542
 - presence, 447, 449
 - reg, 130–132, 206, 208, 563
 - winfo, 427, 428, 447
 - xcap-diff, 383–385, 387, 403
 - Forking, 65, 66, 70, 78, 95
 - Header fields
 - Accept, 315, 316, 410, 438
 - Accept-Contact, 80, 81, 518
 - Alert-Info, 530, 533, 534, 544, 545
 - Allow, 80, 143
 - Answer-Mode, 508, 520
 - Authentication-Info, 296, 297, 393
 - Authorization, 119, 127, 188, 227, 235, 277, 294–296, 300, 303, 305–308, 334, 392, 401, 466
 - Call-ID, 70, 141, 178, 208, 287, 536, 564

- Call-Info, 530, 532, 535
- Contact, 52, 73, 75, 78–80, 97–99, 119, 123, 125, 133, 138, 142, 143, 152, 158, 177, 180, 206, 209–212, 285, 287, 322, 325, 332, 334, 413, 416, 499, 510, 542, 551, 565
- Content-Disposition, 70, 472
- Content-Length, 143, 144
- Content-Type, 70, 71, 143, 144, 285, 315, 379, 380, 412, 423, 429, 438, 439, 461, 546, 558
- CSeq, 70, 141, 208, 287, 564
- Date, 287
- Error-Info, 535
- Event, 86, 133, 384, 406, 409, 427, 428, 444, 447, 538, 540
- Expires, 125, 409, 410
- From, 70, 127, 141, 146, 178, 180, 285, 287, 291, 306, 322, 325, 332, 334, 335, 407, 421, 536, 542, 551, 574
- History-Info, 188, 536, 537, 539, 551, 566, 567
- Identity, 431
- Max-Forwards, 70
- P-Access-Network-Info, 140, 141, 150, 156, 188, 189, 308, 332, 333, 452
- P-Answer-State, 508, 521, 522
- P-Asserted-Identity, 140, 141, 146, 150, 151, 154, 156, 158, 184, 197, 210, 212, 291, 310, 333, 542, 543, 551
- P-Asserted-Service, 210–212, 310
- P-Associated-URI, 124, 125, 130, 300, 306
- P-Called-Party-ID, 152–154, 156, 158, 188
- P-Early-Media, 530, 535
- P-Preferred-Identity, 138, 140, 146, 156, 210, 333, 336, 542
- P-Preferred-Service, 210, 211
- P-Visited-Network-ID, 119, 232
- Path, 119, 123, 125, 152, 334
- Priv-Answer-Mode, 520
- Privacy, 141, 154, 156, 158, 291, 418, 536, 542, 543
- Reason, 535
- Record-Route, 78, 79, 138, 146, 150–152, 154, 158, 159, 161, 175, 192, 195, 407, 542
- Refer-To, 96, 99, 505, 553, 555, 556
- Referred-By, 551, 553
- Reject-Contact, 80, 81
- Replaces, 539, 553, 555
- Request-Disposition, 80, 81
- Require, 78–80, 141, 142, 156, 159, 165, 198, 200, 423, 447, 472, 555
- Route, 69, 78, 79, 90, 133, 138, 140, 145, 146, 152, 159, 174–177, 184, 186, 231, 282, 322, 323, 325, 326, 332, 333, 407, 539, 563, 564, 568, 569
- RSeq, 200
- Security-Client, 301, 302, 305, 307
- Security-Server, 302
- Security-Verify, 142, 302, 305, 307
- Service-Route, 124, 125, 138, 145, 333, 334
- SIP-ETag, 88, 89, 412, 439
- SIP-If-Match, 439
- Subscription-State, 88, 89, 410, 434, 448
- Supported, 78, 79, 97, 141, 142, 154, 198, 323, 423, 547
- To, 69, 70, 72, 119, 127, 141, 152, 178, 285, 306, 322, 325, 332, 334, 407, 472, 474, 536, 551, 564
- Unsupported, 78
- Via, 52, 70, 90, 138, 140, 146, 158, 268, 282, 301, 305, 306, 392, 542
- WWW-Authenticate, 122, 235, 277, 296, 299, 334, 392, 465
- Methods
- ACK, 69, 71, 72, 75, 77, 78, 80–82, 84, 90, 138, 143, 166, 167, 200, 206, 308, 458, 547, 575, 576
- BYE, 68, 69, 71, 75, 77, 78, 80, 81, 91, 138, 143, 180, 181, 218, 249, 260, 458, 546, 548, 555, 557, 574, 576
- CANCEL, 69, 71–73, 80, 81, 143, 308
- INFO, 69, 551
- INVITE, 68, 69, 71–73, 75, 76, 78, 80–82, 84, 85, 90–92, 95, 96, 99, 100, 121, 138–143, 145–152, 154–156, 158, 159, 161, 165, 166, 172–177, 179–182, 184, 186, 187, 192, 196, 198, 201, 204–206, 209, 211, 212, 218, 232, 244–249, 253–256, 258, 273, 278, 279, 287,

- 288, 291, 312, 313, 318, 321–327, 329, 331–336, 454–458, 465–467, 473, 475, 476, 479, 480, 484, 499, 501, 502, 505–508, 514–519, 521, 526–528, 530, 532–537, 539, 544–547, 550–553, 555, 557, 563–571, 573–575
- MESSAGE, xxviii, 22, 69, 209, 210, 454, 455, 470–472, 477, 478, 506, 517, 518
- NOTIFY, 69, 85–89, 94, 95, 131, 133, 134, 181, 206, 385, 387, 403, 406, 409–411, 413, 428, 433–436, 438, 447, 448, 450, 506, 519, 520, 523, 539–541, 555, 557
- OPTIONS, 69, 80, 81, 181, 182, 196, 454
- PRACK, 69, 83, 84, 138, 142, 143, 148, 156, 159–161, 165, 166, 181, 199, 206, 479, 520, 535, 547
- PUBLISH, 69, 173, 374, 407, 412, 413, 438–441, 444, 450, 508, 522, 523
- REFER, 69, 93–96, 99, 143, 501, 505, 506, 516, 519, 520, 553, 555–557
- REGISTER, 43, 69, 72, 73, 80, 97, 98, 117–131, 138, 145, 152, 184, 188, 189, 206, 208–210, 212, 229, 230, 233, 294–296, 298–303, 305–308, 334, 412, 454, 477, 510, 563
- SUBSCRIBE, 69, 75, 85–88, 95, 131, 148, 150, 173, 181, 182, 209, 212, 384, 386, 403, 406, 408–411, 428, 435, 436, 438, 441, 446, 447, 450, 454, 505, 538, 540
- UPDATE, 69, 75, 85, 142, 143, 148, 161, 163, 165, 181, 200, 203, 256, 258, 479, 528
- Offer/answer, 60, 61, 106, 108, 159, 161, 243, 246, 292, 363, 476, 494, 495, 551
- Option Tags
100rel, 83, 84, 141, 142, 156, 159, 165, 198, 200, 547
- eventlist, 447
- precondition, 84, 141, 142, 154, 156, 198
- sec-agree, 141
- Parameters
comp, 90, 143, 146, 158
- Request-URI, 68, 73, 99, 100, 119, 122, 127, 131, 138, 148, 149, 151, 152, 154, 176, 177, 182, 186, 188, 279, 282, 318, 322, 323, 325–327, 331–333, 378–381, 384, 391, 392, 394, 396–399, 401, 402, 407, 410, 447, 514, 516, 536, 539, 551, 555, 564, 568, 569, 571, 573, 574
- SIPS URI, 61, 62, 282, 310, 407, 415, 423, 453
- SMS (Short Message Service), xxxi, 6, 47, 117, 454
- SRF (Single Reservation Flow), 271–274
- SRTP, xxxii, 292, 364, 365
- SS7, 30, 189
- STUN, xxviii, 105–108, 213
- TCP, 81, 92, 138, 148, 217, 280, 281, 290, 301, 359–361, 365, 454–456, 458–460, 480, 481, 494, 495
- TD-SCDMA Forum, 14
- TEL URI, 42–44, 148–150, 176, 197, 233, 236, 319, 320, 322, 331–333, 407, 414, 423, 431, 560, 561, 571, 574
- TFRC (TCP-Friendly Rate Control), 361
- TIA, 15, 16, 314
- TLS, xxviii, 62, 280–282, 293, 294, 302, 303, 309, 310, 392, 455, 456, 459, 465–467, 494
- TTA, 14, 16
- TTC, 16
- TURN, xxviii, 105–108
- UDP, 81, 84, 92, 138, 148, 156, 216, 217, 281, 300, 301, 346, 359–361, 363, 364, 454, 455
- UDVM (Universal Decompressor Virtual Machine), 90
- UICC (Universal Integrated Circuit Card), 45, 46, 48, 117, 118, 122, 125–128, 188, 294, 297, 298, 300, 306, 318, 329, 334, 335
- UMTS, 14, 29, 43, 46, 112, 140, 297
- UMTS Forum, 14
- URI (Uniform Resource Identifier), xxviii, 35, 42, 44, 48, 62–64, 68–70, 72, 73, 75, 78, 92–97, 99, 100, 117, 119–127, 130, 131, 133, 138, 140–143, 146, 148–152, 154, 158, 168, 174–177, 182–184, 186, 188, 197, 206, 221, 230, 231, 233, 236, 279, 282, 300, 306, 310–313, 315,

- 318–323, 325–327, 331–336, 372, 376–381, 384, 385, 391, 392, 394, 396–399, 401, 402, 407, 408, 410, 413–416, 418, 419, 423, 424, 434–437, 447, 453, 455, 470–472, 474, 478, 499–502, 504–506, 509, 514–517, 530, 534, 536, 539, 544, 553, 555, 557, 560, 561, 563–565, 568, 569, 571, 573, 574
aaa, 221, 222
aaas, 221, 222
im, 453
msrp, 456
pres, 407, 415, 453
sip, 43, 44, 48, 61, 62
sips, 62, 282, 310, 407, 415, 423, 453
URI List, 471, 506, 509, 514–516
URI-List Service, 435, 436, 470, 471, 505, 506
URL (Uniform Resource Locator), 61, 319, 320, 322, 331–333, 456, 458–460, 465, 467, 530, 532, 535, 560, 561, 571, 574
URN (Uniform Resource Name), 97, 99, 209, 210, 212, 250, 318, 322, 323, 325–327, 331, 332, 376, 416, 418
VCC (Voice Call Continuity), xxviii, 559–574, 576, 579
XCAP, xxviii, 21, 168, 371–374, 376–385, 387–391, 394–398, 403, 408, 429, 430, 433, 434, 436, 439, 444, 448, 450, 470, 471, 492, 508–510, 557, 558, 563, 581, 585
AUID, 374, 393–399, 433, 448, 558
XDM (XML Document Management), xxviii, 21, 389–391, 393–399, 403, 443, 508–510, 584, 586, 587
XDMC (XDM Client), 389–397, 399, 401, 403, 444, 508, 509
XMDS (XDM Server), 389–391, 393, 394, 397–399, 401, 402, 444, 447, 448, 450, 509, 510
XPath, 386
XQuery, 390, 398, 399, 402