Redes Móveis

2014/2015

Segundo Trabalho:

IP Multimedia Subsystem:

taxação de serviços

Rodolfo Oliveira

rado@fct.unl.pt

Versão 3.0

1. Objectivos

Resumo

Este trabalho pretende estudar e estender um *Application Server* (AS) integrado na plataforma OpenIMS. O AS implementa um serviço de taxação de serviços. A taxação do serviço divide-se em dois objectivos distintos, sendo o código do primeiro objectivo disponibilizado aos alunos através de um projecto Netbeans.

Primeiro objectivo – taxação de terminais não registados

O primeiro objectivo de taxação consiste em promover que os terminais estejam activos na rede, para aumentar o nível de disponibilidade de receptores. Dessa forma, os terminais são taxados sempre que se encontram de-registados da rede.

O diagrama simplificado da operação de taxação encontra-se representado na Figura 1. A sua implementação encontra-se no projecto disponibilizado. Basicamente, recorre-se ao serviço "*Subscribe-Notification*" para que o *Application Server* (AS) seja informado sempre que os utilizadores alteram o seu estado de registo.

Inicialmente todos os terminais são iniciados com 500 unidades de crédito. A taxação deve ser de n unidades por minuto, onde n é o número de horas em que o terminal está registado, adicionado de uma unidade. Assim, os utilizadores são mais penalizados à medida que se encontram de-registados há mais tempo (uma unidade por minuto durante a primeira hora de de-registo, duas unidades por minuto após a primeira hora de de-registo, etc.). Sempre que o utilizador se regista na rede, este recebe uma mensagem com o montante do seu saldo.

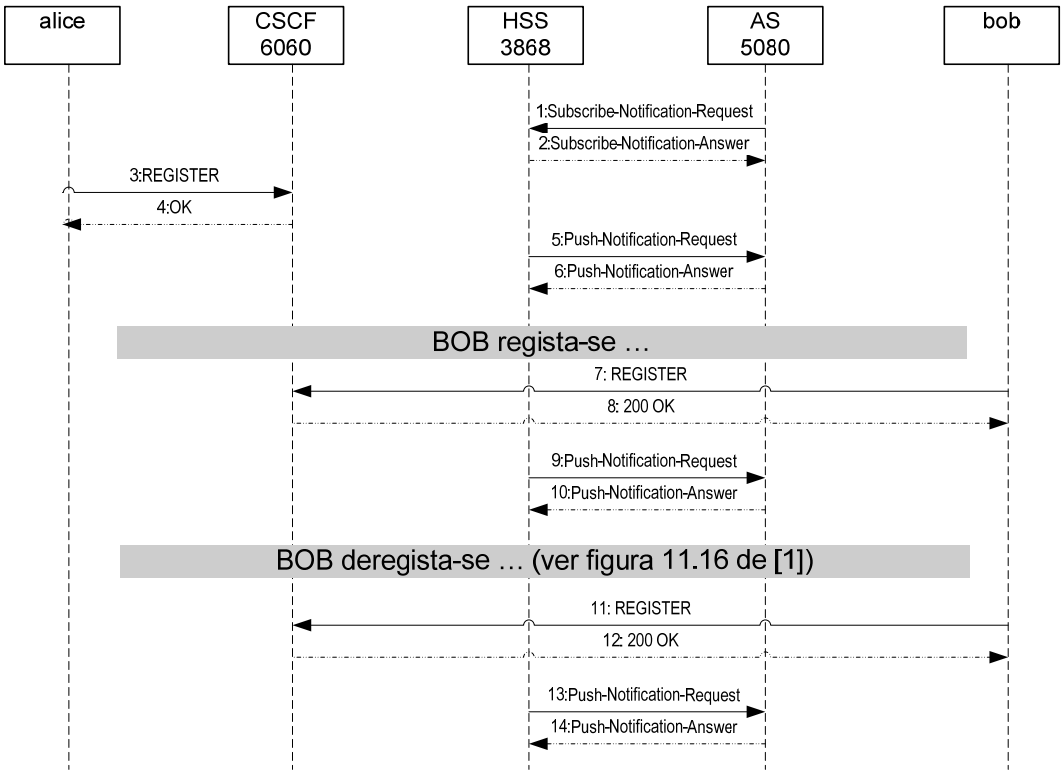


Figura 1 – Diagrama simplificado da operação de taxação de de-registo dos terminais.

Segundo objectivo – taxação online

No segundo objectivo pretende-se implementar um mecanismo de taxação *online* de uma sessão IMS. No início da sessão, o AS recebe a mensagem INVITE, tal como se observa na Figura 2 (mensagem 6). Nesse instante, o AS reserva 30 unidades de crédito, pois uma sessão tem um custo fixo de 30 unidades, sendo depois taxada a 10 unidades por minuto.

Caso a **sessão seja activada com sucesso** (situação a) na Figura 2), o AS reserva somente o crédito de 30 unidades e arma um temporizador com um minuto de conversação (10 unidades). Caso **posteriormente receba um erro** (404 na situação b)), o AS restitui a reserva efectuada no passo 7. Caso **o crédito seja insuficiente para realizar a reserva** (caso C)), o AS envia uma mensagem de erro 402, enviando posteriormente uma mensagem ao utilizador informando-o da insuficiência de crédito.

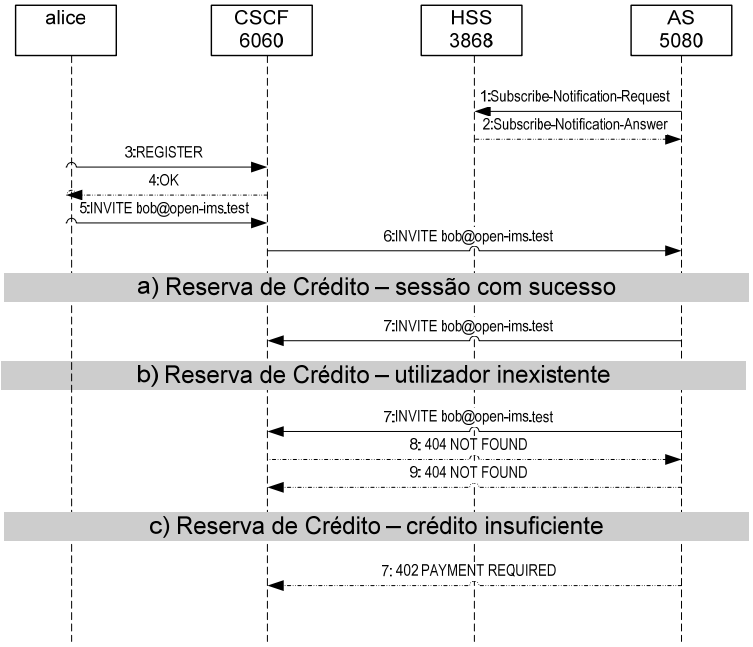


Figura 2 – Fase de reserva de crédito no início do estabelecimento de sessão.

Após o temporizador ser activado com o tempo de crédito existente, a sessão pode ser interrompida por um dos utilizadores (representado na Figura 4), ou caso não seja, o temporizador responsável pelo controlo de crédito irá expirar. Uma vez expirado o temporizador, e sempre que o saldo seja negativo, o AS deve enviar uma mensagem ao chamador, alertando-o para a insuficiência de saldo. Esta mensagem é também enviada sempre que a sessão é interrompida por um dos utilizadores, tal como representado na Figura 4.

Figura 3 – Sessão interrompida – crédito insuficiente.

Figura 4 – Sessão interrompida pelo telefone chamador.

Neste sistema de taxação sempre que o utilizador fica sem saldo, a sessão não é terminada. No entanto, o saldo poderá ficar negativo caso a sessão não seja interrompida, e o utilizador é imediatamente avisado quando em sessão activa for atingido um saldo negativo.

Terceiro Objectivo (extra) – taxação do terminal chamado

Neste objectivo pretende-se que o terminal chamado também seja taxado. O terminal chamado é taxado online, sendo o custo por minuto de sessão de 3 unidades de crédito por minuto.

2. Plataforma OpenIMS

O IMS (IP Multimedia Subsystem) é uma arquitectura para plataformas de entrega de serviços Multimédia baseados em IP (Internet Protocol). Em [1] encontra-se a seguinte definição de IMS:

IMS é uma arquitectura de controlo de serviços global, independente da tecnologia de acesso e baseado em conectividade normalizada IP, a qual permite vários tipos de serviços de multimédia para utilizadores finais que utilizam os mesmos protocolos *Internet-based*.

Neste trabalho adopta-se a plataforma OpenIMS [2], implementada pelo Instituto Fraunhofer (Fokus). O OpenIMS consiste numa implementação de um módulo Call Session Control Function (CSCF) e de uma base de dados que representa uma implementação do módulo Home Subscriber Server (HSS), os quais em conjunto formam os elementos do núcleo das arquitecturas IMS/NGN (IP Multimedia Subsystem / New Generation Networks) especificadas pelos institutos/iniciativas de normalização 3GPP, 3GPP2, ETSI, TISPAN e PacketCable. As quatro componentes que constituem a plataforma baseiam-se em software de código aberto (e.g. o HSS baseia-se em MySQL).

A Figura 5 apresenta a estrutura principal dos blocos definidos pela plataforma OpenIMS, bem como as interfaces definidas entre elas. Os números representados no interior dos rectângulos referem-se à configuração dos portos definidos na configuração do OpenIMS que é disponibilizada na imagem fornecida aos alunos.

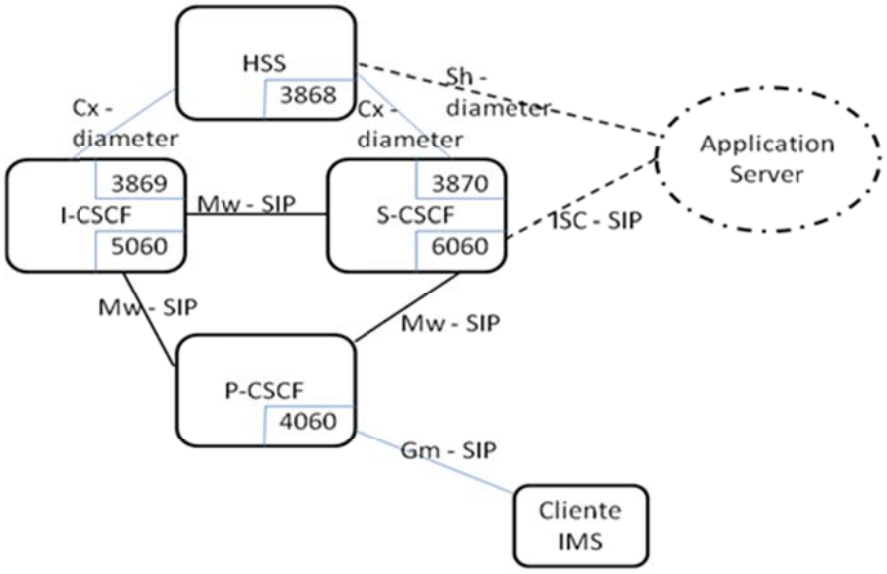


Figura 5 – Configuração dos blocos da plataforma OpenIMS disponibilizados na imagem Ubuntu.

A configuração da Figura 5 é muitas vezes resumida a dois módulos: o módulo CSCF e o módulo HSS.

CSCF

O módulo Call Session Control Function (CSCF) consiste num conjunto específico de sub-elementos que gerem a sinalização associada com o estabelecimento, terminação e troca de mensagens SIP das chamadas. O CSCF é constituído por 3 sub-elementos:

- P-CSCF – Proxy Call Session Control Function
- I-CSCF – Interrogating Call Session Control Function
- S-CSCF – Serving Call Session Control Function

Estes elementos actuam como servidores ou *proxies* SIP e são o núcleo de processamento dos pacotes de sinalização SIP numa rede IMS.

O **Proxy-CSCF (P-CSCF)**, como o nome indica, é um proxy SIP que funciona como o primeiro e exclusivo ponto de contacto de um terminal IMS (User Equipment). Por ele passam todas as mensagens SIP enviadas ou recebidas pelo terminal (embora algumas redes usam um SBC – Session Border Control para desempenhar esta função). O P-CSCF é indicado ao terminal IMS quando este se regista e não muda durante o período do registo. É, também, o responsável por gerar os registos de utilização para taxação. O P-CSCF pode ser responsável pela implementação de QoS, gestão de largura de banda e controlo de acessos. O P-CSCF desempenha também um papel na segurança da rede, sendo o responsável pela autenticação e estabelecimento de um túnel IPSEC com o terminal. O P-CSCF previne ataques através da alteração dos pacotes SIP e protege a privacidade do utilizador. Os outros componentes da rede estabelecem uma relação de confiança com o P-CSCF e não autenticam o utilizador novamente. Em suma, o P-CSCF executa as seguintes tarefas:

- É ponto de contacto com o UE; faz as primeiras verificações de segurança do cliente através dos dados existentes no HSS;

- Estabelece o protocolo IPSEC para criar um domínio seguro (*trusted)*;

- Gere dados de *billing*;

- Implementa QoS;

O **Interrogating-CSCF (I-CSCF)** tem um endereço IP público e é através dele que servidores remotos contactam um determinado domínio. O I-CSCF interroga o HSS usando a interface Diameter Cx para conhecer a localização de um utilizador e encaminha os pedidos SIP para o S-CSCF que serve esse cliente. Até a *Release* 6 da normalização IMS, podia ser usado como o ponto de entrada, escondendo a rede do domínio público, tornando-se num Topology Hiding Inter-network Gateway (THIG). A partir da *Release* 7, este papel é desempenhado pelo Interconnection Border Control Function (IBCF). Em suma o I-CSCF executa as seguintes tarefas:

- Obtém o nome do próximo elemento a contactar (AS ou S-CSCF) a partir do HSS;

- Atribui um S-CSCF baseado na informação obtida a partir do HSS. Esta atribuição sucede quando o utilizador se regista na rede ou um utilizador não se encontra registado na rede mas possui serviços registados (e.g. *voice mail*);

- Encaminha pedidos que lhe cheguem *a posteriori* para o S-CSCF anteriormente atribuído.

O **Serving-CSCF (S-CSCF)** é o centro de encaminhamento do sistema, sendo o responsável pelos registos SIP que fazem a interligação entre o terminal e o respectivo endereço SIP. Por ele passam todas as mensagens de sinalização SIP e é o responsável por determinar para que *Application Server* (AS) deve encaminhar as mensagens SIP de modo a fornecer o serviço desejado. O S-CSCF usa as interfaces Diameter Cx e Dx para contactar o HSS e, é este último que define o S-CSCF que o utilizador deve utilizar quando é contactado por uma rede externa via I-CSCF. O S-CSCF está localizado na *home network*. Em suma, o Serving-CSCF executa as seguintes tarefas:

- É o responsável pelo controlo da sessão;

- Evita a utilização não autorizada dos serviços;

- É o responsável pelo mapeamento entre a localização do utilizador e o seu endereço público (por exemplo, o mapeamento entre o número de telefone e o IP do terminal);

Home Subscriber Server (HSS)

O Home Subscriber Server (HSS) é a base de dados central de utilizadores de uma rede IMS. O seu papel é semelhante ao Home Location Register (HLR) e ao Authentication Center (AuC) encontrados nas redes GSM. Contem o perfil de utilizador, informação de autenticação, autorização e localização física do mesmo. De acordo com a especificação 3GPP, o HSS é a entidade que armazena a informação relativa aos utilizadores para permitir que os elementos de rede possam fazer o tratamento das sessões. Proporciona suporte à gestão de mobilidade, estabelecimento de chamadas e/ou sessões, geração de informação de segurança dos utilizadores, identificação e autorização de acesso dos mesmos. O HSS guarda chaves secretas para os clientes móveis e gera informação de segurança dinâmica para cada um deles. Também fornece funções de aprovisionamento e autorização dos serviços.

O HSS também pode mapear o número de telefone do utilizador na rede PLMN e guardar informação como qual o tipo de terminal e a localização de cada cliente. Nele podem constar as identidades IP Multimedia Public Identity (IMPU), IP Multimedia Private Identity (IMPI), International Mobile Subscriber Identity (IMSI), Mobile Subscriber ISDN Number (MSISDN), entre outras para cada utilizador.

É no HSS que se definem e configuram os utilizadores e os seus privilégios, bem como os filtros das mensagens que se desejam encaminhar para os diferentes ASes (triggers). No caso da plataforma OpenIMS, pode aceder à configuração do HSS através do endereço (user: hssAdmin password: hss):

http://hss.open-ims.test:8080/hss.web.console/

Os anexos III e IV exemplificam a definição de novos utilizadores e a definição de *trigers* e de outra informação necessária para a integração de um AS no OpenIMS.

Application Server (AS)

Uma das principais vantagens do IMS está no valor acrescentado dos seus serviços multimédia. O *Application Server* (AS), como o seu nome indica, é o servidor responsável pelo armazenamento e execução desses serviços multimédia. Mas dependendo do tipo de serviço, o AS pode funcionar como proxy, SIP *User Agent* ou B2BUA (*back-to-back user agent*).

O SIP AS comunica com o S-CSCF que lhe encaminha determinadas sessões baseado em informações obtidas a partir do HSS. De acordo com regras criadas para a sessão, o SIP AS decide qual das suas aplicações deverá tratar a sessão. Durante a execução da lógica do serviço, o SIP AS pode questionar o HSS para obter mais informação sobre o cliente. Para melhorar a experiência do serviço, a rede IMS antecipa as interacções necessárias do serviço e faz uso de facilitadores, como serviços de localização, presença e gestão de grupos.

Os *Application Servers* podem permitir o acesso dos serviços a outros elementos da rede IMS através de *Servlets* SIP. Como a rede IMS é modular, se um novo serviço necessitar de algum requisito não disponível, é apenas necessário actualizar ou substituir o AS onde o serviço vai ficar alojado. Esta implementação contrasta com os modelos antigos das redes de telecomunicações em que cada serviço tinha os requisitos específicos de equipamento proprietário.

Para aumentar a flexibilidade na criação de serviços, um AS pode estar localizado na *home network* ou numa rede de terceiros (de um fornecedor de serviços). Se estiver localizada na *home network* pode **comunicar directamente com o HSS através da interface *Diameter Sh* no caso de ser um SIP-AS**.

3. Plataforma Mobicents SIP Servlets

Os ASes que implementam os serviços de taxação *Online* e *Offline* especificadas neste trabalho são executados na plataforma Mobicents SIP Servlets. A plataforma Mobicents SIP Servlets é a primeira implementação *Open Source* certificada de acordo com especificação SIP Servlet 1.1. As Servlets (aplicações AS) são executadas sobre *containers* (servidores de aplicações), que no caso podem ser do tipo servidores de aplicação TomCat ou JBOSS. Neste trabalho utilizamos o servidor de aplicações JAVA "JBOSS", dado a plataforma Mobicents estar mais robusta para este tipo de *container*.

A Figura 6 exemplifica a execução do AS sobre este tipo de plataforma. O servidor de aplicações JBOSS dá suporte à plataforma Mobicents, a qual utilizando um Multiplexer de comunicações que implementa o protocolo Diameter acede à rede externa, bem como ao AS (*External Application*).

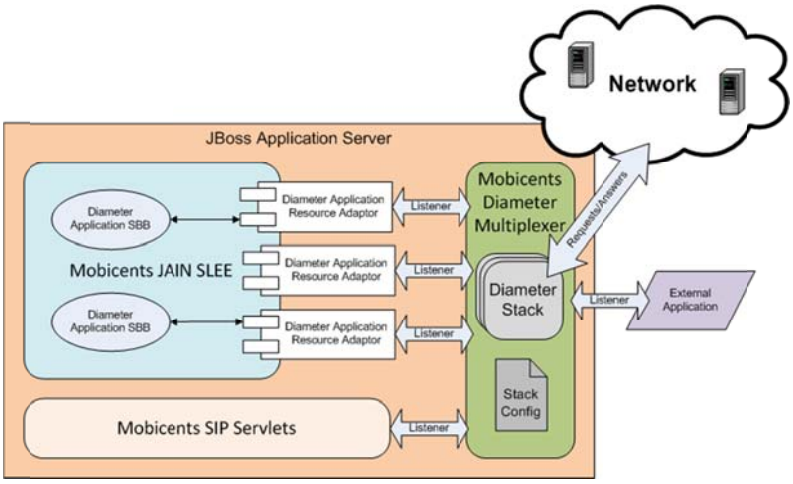


Figura 6 – Arquitectura das entidades envolvidas na execução do AS.

Na imagem Ubuntu disponibilizada aos alunos, o acesso à interface de gestão do JBOSS realiza-se através do endereço (user: admin password: admin):

http://localhost:8081/admin-console/login.seam;jsessionid=3CC2D624625DC7024EB40B6868F1AE95?conversationId=1

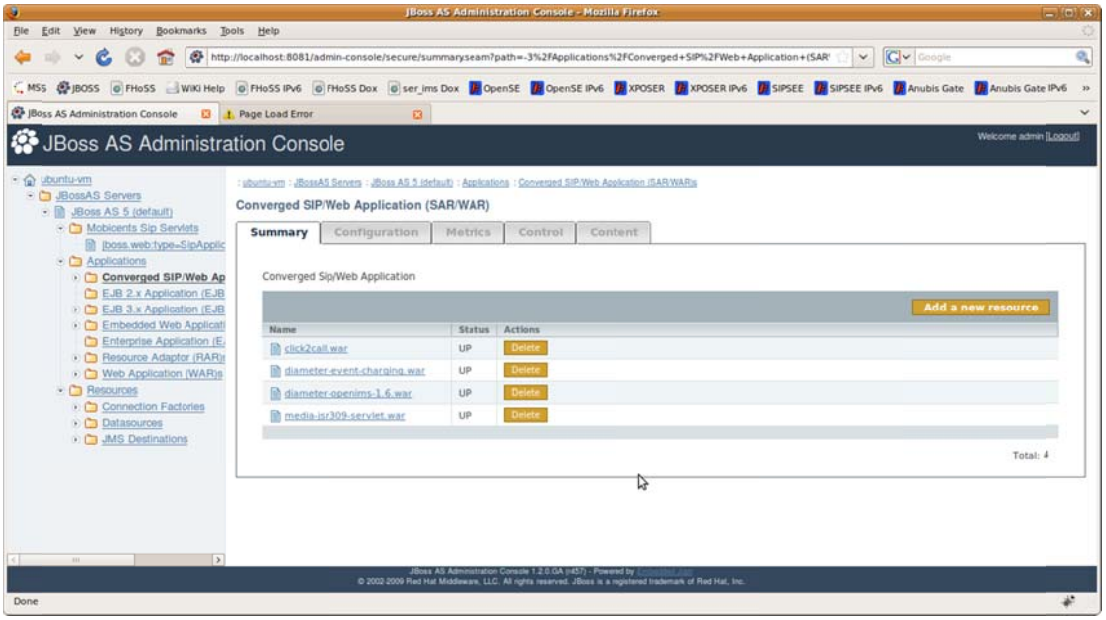


Figura 7 – Aspecto da consola de gestão da plataforma JBOSS.

Na Figura 7 ilustra-se o tipo de aplicações SIP/Web que o servidor JBOSS se encontra a executar. Essas aplicações (Servlets) encontram-se sobre a forma de um ficheiro compactado com extensão .war (na realidade é um ficheiro do tipo zip) na pasta

/opt/mss-1.2-jboss-5.1.0.GA/server/default/deploy

As servlets SIP desenvolvidas na plataforma mobicents são geridas na consola de gestão disponibilizada no endereço:

http://localhost:8081/sip-servlets-management/org.mobicents.servlet.management.SipServletsManagement/SipServletsManagement.html

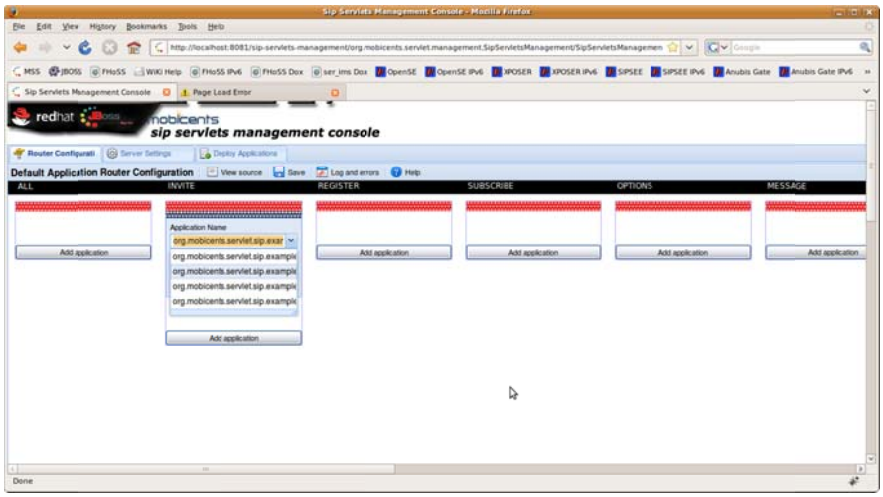


Figura 8 – Aspecto da consola de gestão da plataforma Mobicents.

SIP Servlets

As Servlets SIP [7] são aplicações baseadas em Java, as quais são geridas por *containers* SIP Servlet que realizam toda a sinalização SIP necessária ao seu funcionamento. Tal como sucede noutros componentes Java, as Servlets são classes Java independentes da plataforma, que podem ser chamadas dinamicamente por um servidor de aplicações habilitado a executar aplicações SIP. Os *containers*, também denominados de motores de Servlets são extensões do servidor que disponibilizam as funcionalidades às Servlets.

Ciclo de Vida de uma Servlet

As Servlets seguem um ciclo de vida que se encontra esquematizado na Figura 9. O *container* da Servlet chama a classe da Servlet, instancia-a e invoca o método init(), passando toda a informação de configuração da mesma através de um objecto do tipo ServletConfig. Tendo sido inicializada com sucesso, a Servlet fica disponível para ser executar as suas funções, sendo que o *container* invoca repetidamente o método service() com os argumentos que representam os pedidos chegados e as suas respostas. Quando o *container* decide desactivar a Servlet, invoca o método destroy() e a Servlet liberta os recursos alocados no método init().

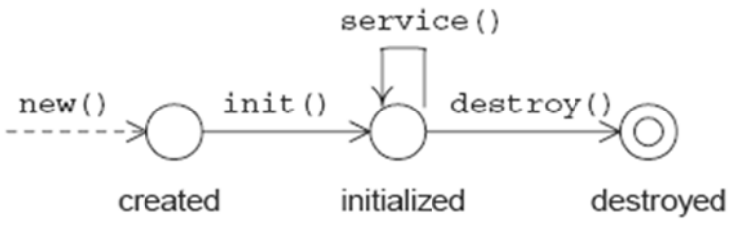


Figura 9 – Ciclo de vida de uma Servlet.

Processamento de mensagens SIP

A interface da Servlet define o método service() para tratar os pedidos dos clientes. Este método é invocado para cada mensagem que o *container* da Servlet encaminhe para uma instância da mesma. Geralmente o *container* da Servlet trata concorrentemente dos pedidos dos clientes realizados à mesma Servlet através da execução concorrente do método service() em diferentes *threads.* As Servlets SIP são invocadas para tratar quer os pedidos (*requests*) quer as respostas (*response/answer*) dos clientes. Em ambos os casos a mensagem é entregue através do método service() da interface da Servlet:

```
void service(ServletRequest req, ServletResponse res) throws
ServletException, java.io.IOException;
```

Se o evento for um pedido, então o argumento *request* é *null* e vice-versa. Quando invocado para processar um evento SIP, os argumentos têm de implementar as interfaces SipServletRequest e SipServletResponse. A implementação do método service() da SipServlet entrega as mensagens chegadas ao servidor para os métodos doRequest() e doResponse(), caso sejam pedidos ou respostas, respectivamente.

```
protected void doRequest(SipServletRequest req);
protected void doResponse(SipServletResponse resp);
```

Estes métodos tratam depois das mensagens como se descreve de seguida.

Métodos para tratamento de Pedidos SIP

A subclasse abstracta SipServlet define um número de métodos para além dos que estão disponíveis na interface Servlet. Estes métodos são automaticamente invocados pelo medo doRequest() (e indirectamente pelo método service()) na classe SipServlet para processar os pedidos SIP recebidos. Estes métodos são os seguintes:

`doInvite` para processar SIP INVITE requests
`doAck` para processar SIP ACK requests
`doOptions` para processar SIP OPTIONS requests
`doBye` para processar SIP BYE requests
`doCancel` para processar SIP CANCEL requests
`doRegister` para processar SIP REGISTER requests
`doPrack` para processar SIP PRACK requests
`doSubscribe` para processar SIP SUBSCRIBE requests
`doNotify` para processar SIP NOTIFY requests
`doMessage` para processar SIP MESSAGE requests
`doInfo` para processar SIP INFO requests

Métodos para tratamento de Respostas SIP

O método doResponse() invoca um dos métodos seguintes de acordo com o código do estado da resposta que a Servlet receba:

`doProvisionalResponse` para processar SIP 1xx informational responses

`doSuccessResponse` para processar SIP 2xx responses
`doRedirectResponse` para processar SIP 3xx responses
`doErrorResponse` para processar SIP 4xx, 5xx, and 6xx responses

O Anexo V descreve a lista de *requests*/*responses* utilizadas nas mensagens SIP. Para mais informações consulte a referência [7].

Interface Diameter Sh

A aplicação (AS) pode necessitar de dados relacionados com uma identidade de um utilizador em particular ou relacionados com uma determinada identidade pública. O AS poderá necessitar de saber qual o S-CSCF para o qual deverá enviar um pedido SIP. Este tipo de informação é guardado no HSS. Além disso, existe um ponto de referência entre o HSS e o AS, denominado Sh, o qual usa o protocolo Diameter [8]. O procedimento para realizar este tipo de comunicações encontra-se dividido em duas categorias: **tratamento de dados** e ***subscription/notification***. A Figura 10 sumariza os comandos disponíveis na interface Sh. O HSS mantém uma lista de ASs que estão autorizados a obter ou guardar este tipo de dados.

Command-Name	Purpose	Abbreviation	Source	Destination
User-Data-Request/Answer	User-Data-Request/Answer (UDR/UDA) commands are used to deliver the user data of a particular user	UDR UDA	AS HSS	HSS AS
Profile-Update-Request/Answer	Profile-Update-Request/Answer (PUR/PUA) commands are used to update transparent data in the HSS	PUR PUA	AS HSS	HSS AS
Subscribe-Notifications-Request/Answer	Subscribe-Notifications-Request/Answer commands are used to make a subscription/ cancel a subscription to user's data on which notifications of change are required	SNR SNA	AS HSS	HSS AS
Push-Notification-Request/Answer	Push-Notification-Request/ Answer commands are used to send the changed data to the AS	PNR PNA	HSS AS	AS HSS

Figura 10 – Comandos da Interface Sh.

Os procedimentos de tratamento de dados contêm a possibilidade de obter dados a partir do HSS. Estes dados podem conter aspectos relacionados com os serviços, informação de registo, identidades públicas dos utilizadores, inital filter criteria, o nome do servidor S-CSCF que suporta o utilizador ou a identidade pública, endereços de funções de *charging,* etc. Estes representam dados que um AS pode guardar no HSS para suportar a lógica que implementa o serviço. Os ASs usam o comando User-Data-Request (UDR) para requerer dados ao HSS. O HSS responde com o uso do comando User-Data-Answer (UDA). O AS também pode actualizar dados transparentemente no HSS

utilizando o comando Profile-Update-Request (PUR), o qual contém os dados a serem alterados. O comando PUR é seguido de um comando Profile-Update-Answer (PUA), o qual indica simplesmente o resultado da operação.

Os serviços de Subscription/Notification permitem que o AS receba uma notificação quando existe uma actualização no HSS dos dados de um determinado utilizador. O AS começa por enviar um comando Subscribe-Notifications-Request (SNR) para posteriormente receber uma notificação quando os dados indicados no SNR foram alterados no HSS. O HSS informa do pedido de subscrição através do comando Subscribe-Notifications-Answer (SNA), o qual explicita o resultado da operação. Se o AS enviou comandos SNR e pediu uma notificação com tipo de pedido subscrito, então o HSS envia um comando Push-Notification-Request (PNR) para o AS sempre que os dados especificados no pedido são alterados no HSS. O comando PNR é depois confirmado pelo comando Push-Notification-Answer.

4. Descrição do Projecto Netbeans fornecido na imagem Ubuntu

O projecto Netbeans fornecido em

```
/opt/project/mobicents-read-only/servers/sip-servlets/sip-servlets-examples/RM_T2_charging/
```

implementa o esquema ilustrado na Figura 1. Este deverá ser compreendido de forma a servir de base para as alterações pedidas neste trabalho.

Para gerir a execução do *container* JBOSS/Mobicents existem os comandos:

```
/opt/x_start_jboss_mobicents
```

```
/opt/x_stop_jboss_mobicents
```

Para copiar para o *container* o ficheiro .WAR (Servlet) gerado a partir do editor Netbeans, deve utilizar o comando:

```
/opt/x_deploy_servlet
```

5. Desenvolvimento do trabalho

O desenvolvimento deste trabalho decorre durante quatro semanas. A sua avaliação será realizada após a entrega e através de uma discussão oral. Enumera-se de seguida uma orientação cronológica dos objectivos a atingir no final de cada semana:

O desenvolvimento deste trabalho decorre durante seis semanas. A sua avaliação será realizada no após a entrega e através de uma discussão oral com os alunos. Enumera-se de seguida uma orientação cronológica dos objectivos a atingir no final de cada semana:

> **Semana 1 (15/22 Abril – turno(s) de terça/quarta)** – compreensão da arquitectura; Utilização de ferramenta de *sniffing* para monitorização da sinalização SIP.
>
> **Semana 2 (21/28 de Abril)** – integração do ficheiro binário que implementa a Servlet da aplicação de taxação; Configuração de triggers e integração da Servlet na plataforma OpenIMS (configuração do HSS); Criação de novos utilizadores (bucha e estica);

Semanas 3 e 4 (5/6 e 12/13 de Maio – sincronização das aulas práticas) – compreensão do código da Servlet que implementa o AS disponibilizado na imagem Ubuntu; **Finalização do primeiro objectivo:** Realização da implementação da política de taxação no cenário de de-registo (desconto variável de acordo com o número de horas de-registado);

Segundo objectivo: implementação da reserva e libertação (em caso de insucesso) de crédito no estabelecimento de sessão. Envio de SMS ao utilizador informando-o de falta de saldo.

Semana 5 (19/20 de Maio) – implementação da reserva de crédito (temporizador) numa sessão activa; caso o crédito seja nulo (expiração do temporizador), realizar o envio de SMS a informar o utilizador da situação de crédito nulo. **Objectivo extra**

Semana 6 (26/27 de Maio) – **Objectivo extra** Envio de SMS ao utilizador informando-o do crédito disponível após a realização da sessão; realização de testes finais e aprimoramento do código;

A data para entrega do trabalho é dia 1 de Junho, até às 19:00. Deverá entregar o projecto Netbeans compactado (entrega por email: rado@fct.unl.pt).

Bibliografia

[1] Miikka Poikselka Aki Niemi, Hisham Khartabil & Georg Mayer, “The IMS: IP Multimedia Concepts and Services”, Third Edition, Wiley, 2008.

[2] OpenIMS, http://www.openimscore.org/ , 2010.

[3] Leonard Broman, “IMS platform prototype”, Master Thesis, Lulea University of Technology, Suécia, 2008. Disponível em http://epubl.luth.se/1402-1617/2008/210/LTU-EX-08210-SE.pdf.

[4] Luis Reis, “A criação de redes de próxima geração IMS usando produtos *open source”,* Dissertação de Mestrado, FEUP/UP, 2008.

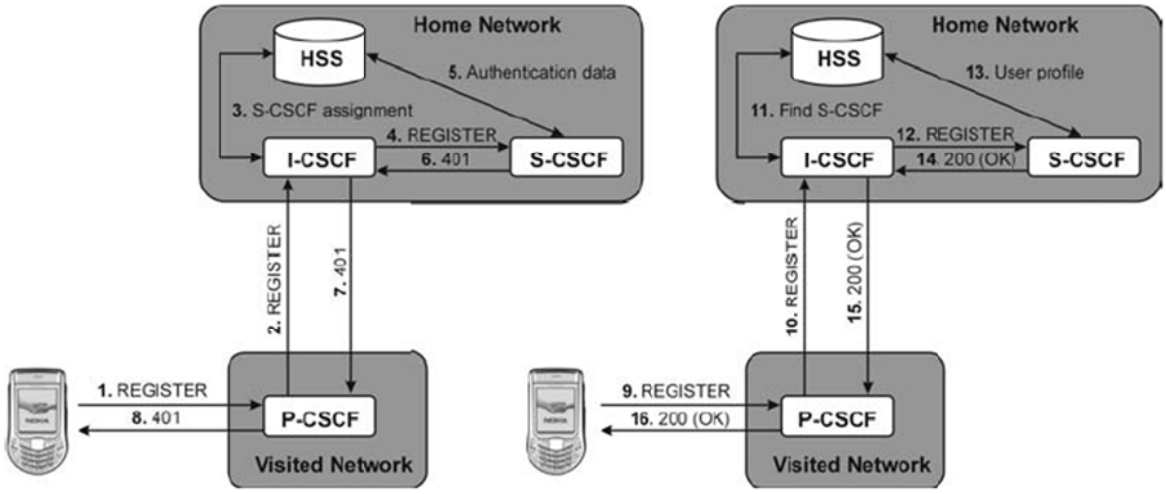
[5] Mobicents, http://www.mobicents.org/products_sip_servlets.html

[6] JBOSS, http://www.jboss.org/jbossas

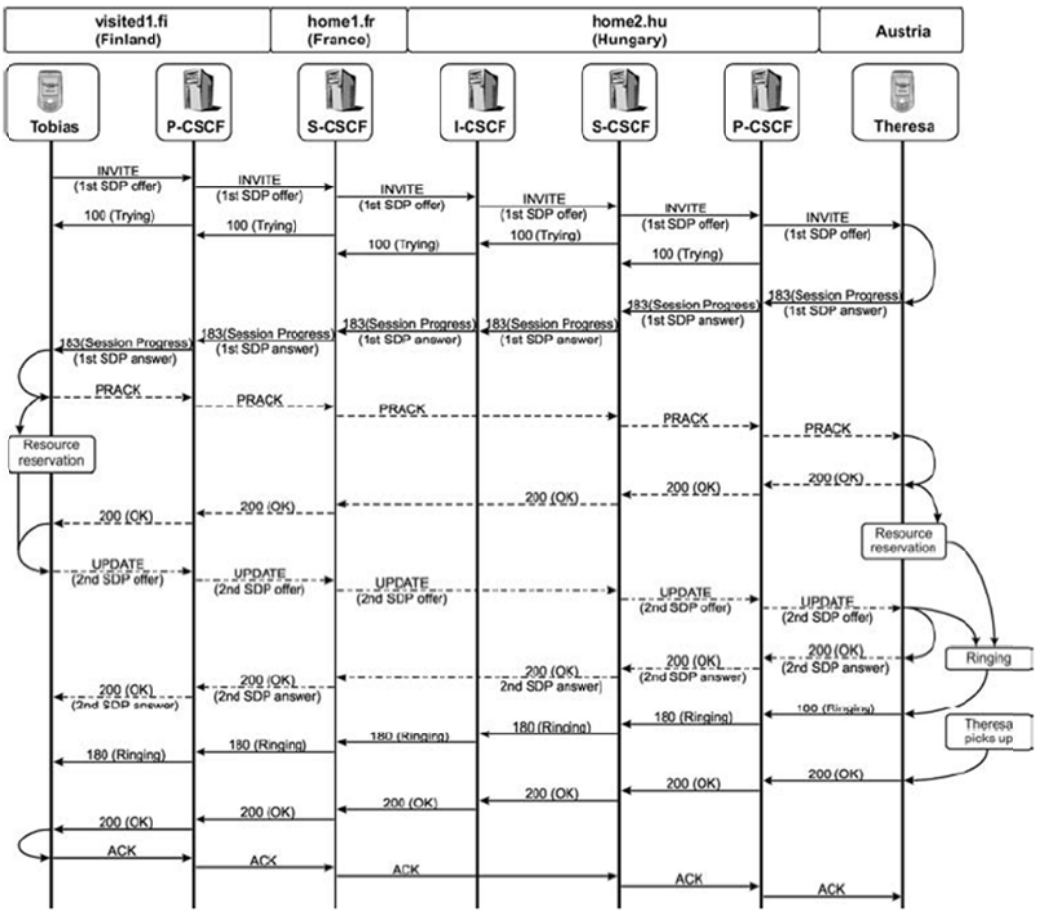
[7] Anders Kristensen, SIP Servlet API Version 1.0, 2003.

[8] RFC 3588 (rfc3588) - Diameter Base Protocol, 2003.

Anexo I – Operação de registo no IMS



Anexo II – Operação de inicio de sessão



Anexo III – Definição de novos utilizadores na plataforma OpenIMS

A definição de um utilizador na arquitectura IMS é representada por uma IMS Subscription, a qual está associada a uma única identidade privada (Private User Identity), sendo possível ter associadas várias identidades públicas (no exemplo da Figura AIII.1 existem 4 identidades públicas distintas para a mesma subscrição, estando agrupadas em dois perfis de serviço diferenciados).

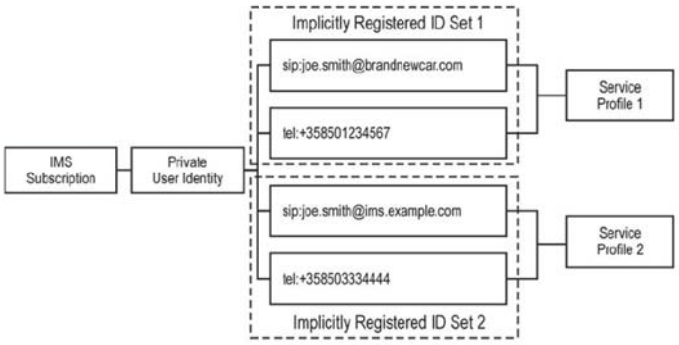


Figura AIII.1 – Exemplo de uma subscrição de um utilizador na arquitectura IMS.

A definição de novos utilizadores na plataforma OpenIMS realiza-se acedendo à configuração do HSS através do endereço (user: hssAdmin password: hss):

http://hss.open-ims.test:8080/hss.web.console/

Inicialmente definem-se os utilizadores clicando na opção "USER IDENTITIES". Começa-se por definir a Private User Identity (IMPI). Este é uma identidade única e global definida pelo operador da Home Network, a qual pode ser usada no interior da Home Network para unicamente identificar o utilizador na perspectiva da rede. O identificador IMPI não identifica o utilizador, mas sim a subscrição do utilizador. Esta identidade é usada principalmente com objectivos de autenticação. Primeiro, deve especificar-se o utilizador (exemplo sip:bucha@open-ims.test ou sip:estica@open-ims.test), devendo-se depois especificar os esquemas de autenticação que este utilizador poderá suportar. A figura AIII.2 exemplifica a configuração desta identidade.

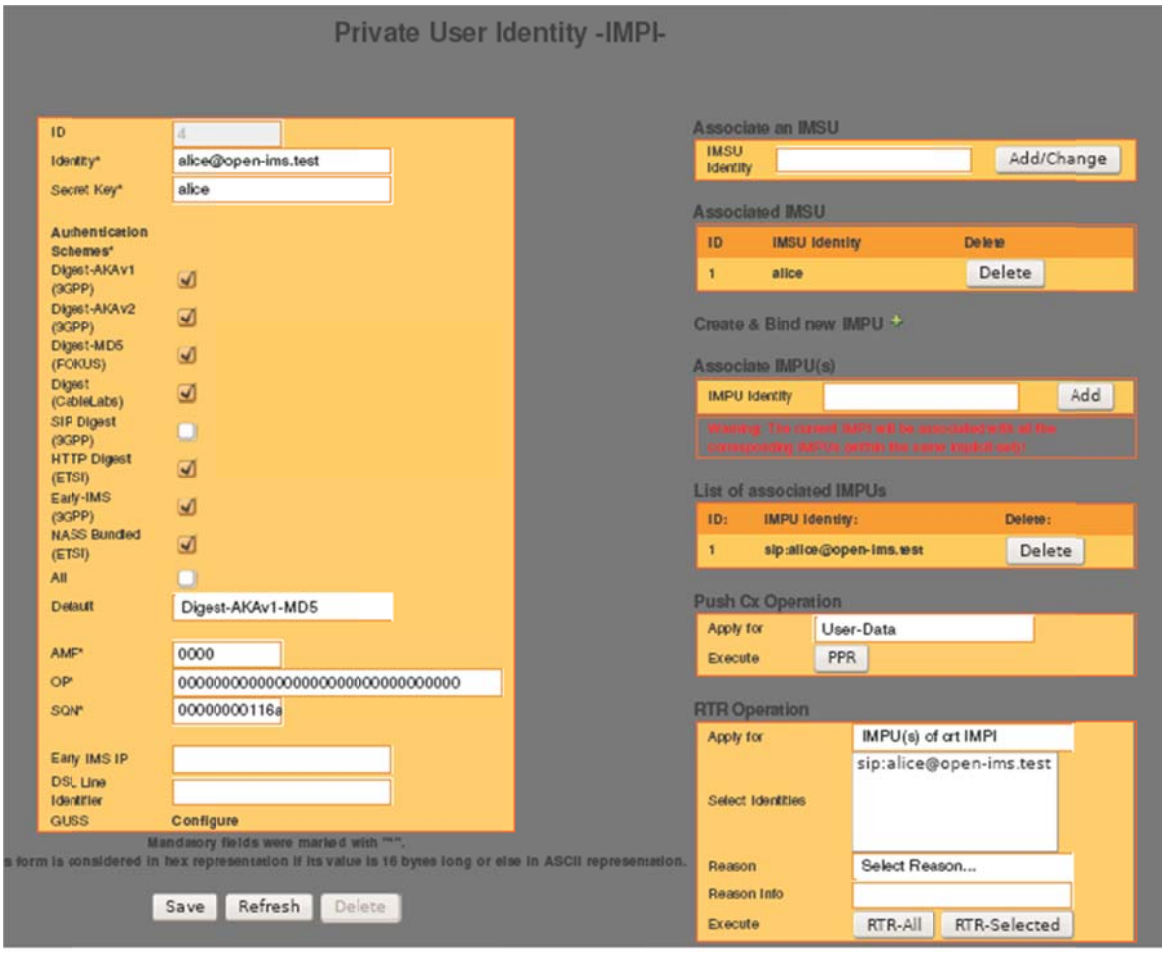


Figura AIII.2 – Configuração da identidade privada do utilizador.

Seguidamente define-se a identidade pública do utilizador (IMPU). A identidade pública é utilizada em pedidos de comunicação com outros utilizadores. São estas identidades que podem ser publicadas, por exemplo em listas telefónicas, paginas Web, etc. A Figura AIII.3 apresenta um exemplo de configuração do utilizador Alice.

Finalmente a Figura AIII.4 apresenta a configuração da subscrição IMS.

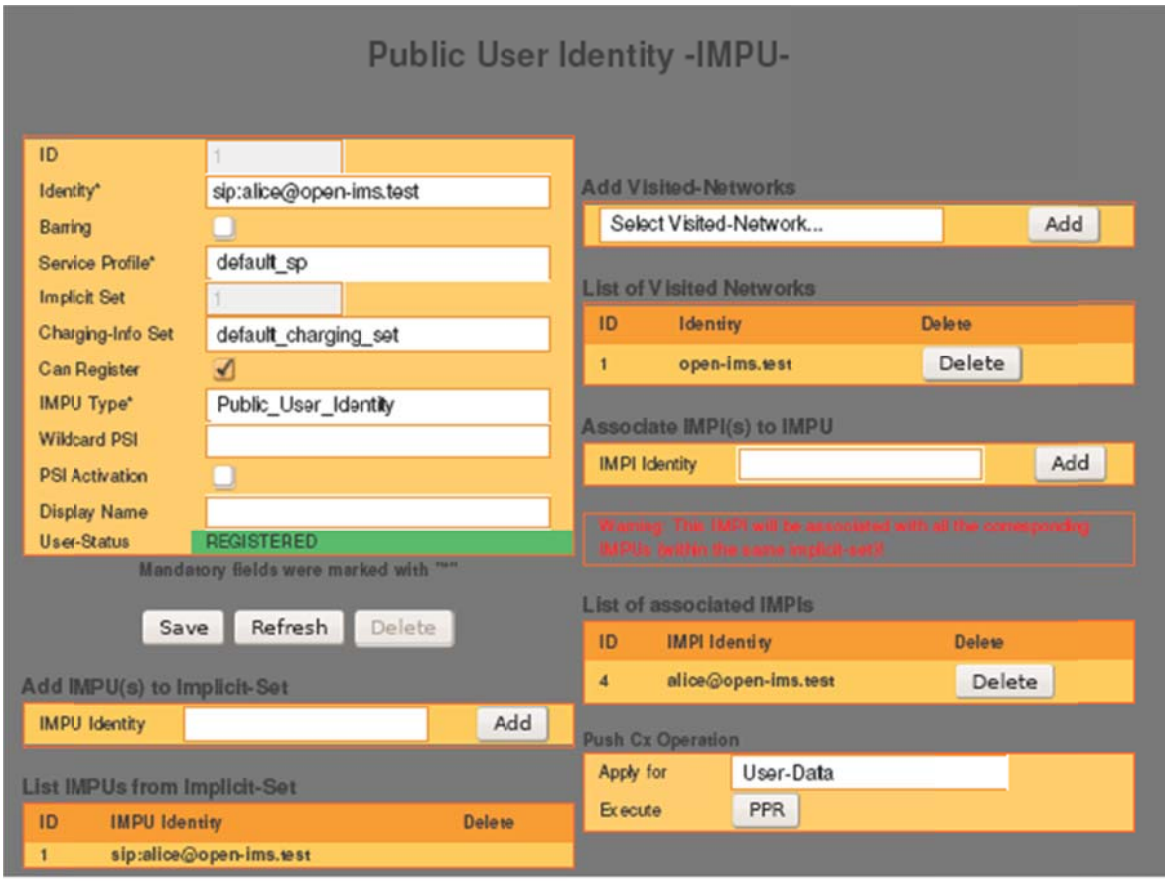


Figura AIII.3 – Configuração da identidade pública do utilizador.

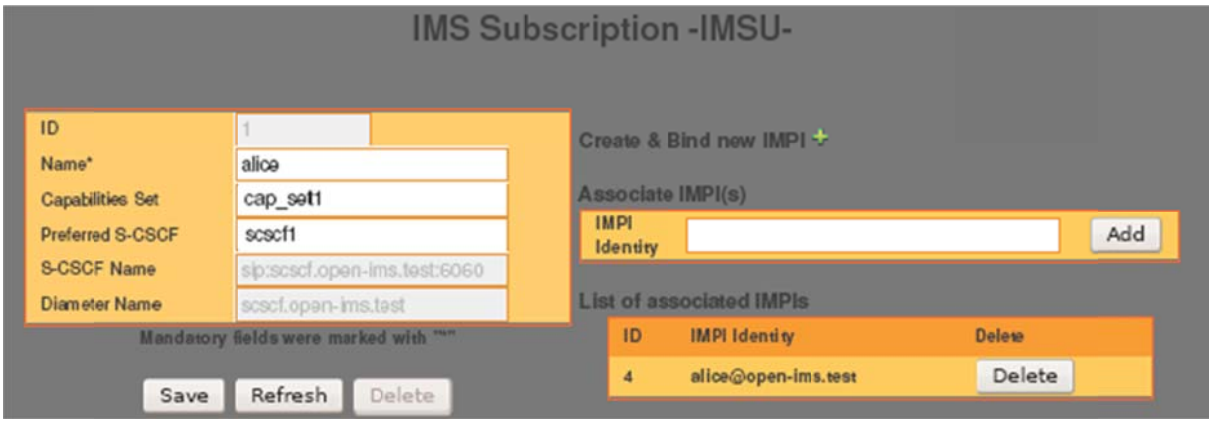


Figura AIII.4 – Configuração da subscrição do utilizador IMS.

Anexo IV – Definição de um novo AS na plataforma OpenIMS

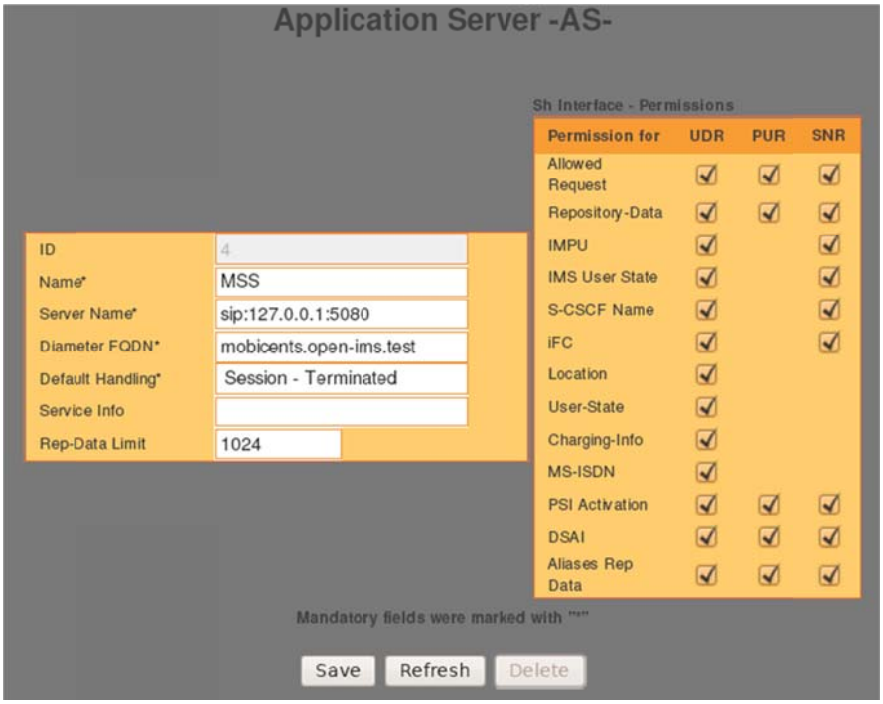
A definição de um novo AS na plataforma OpenIMS realiza-se acedendo à configuração do HSS através do endereço (user: hssAdmin password: hss):

http://hss.open-ims.test:8080/hss.web.console/

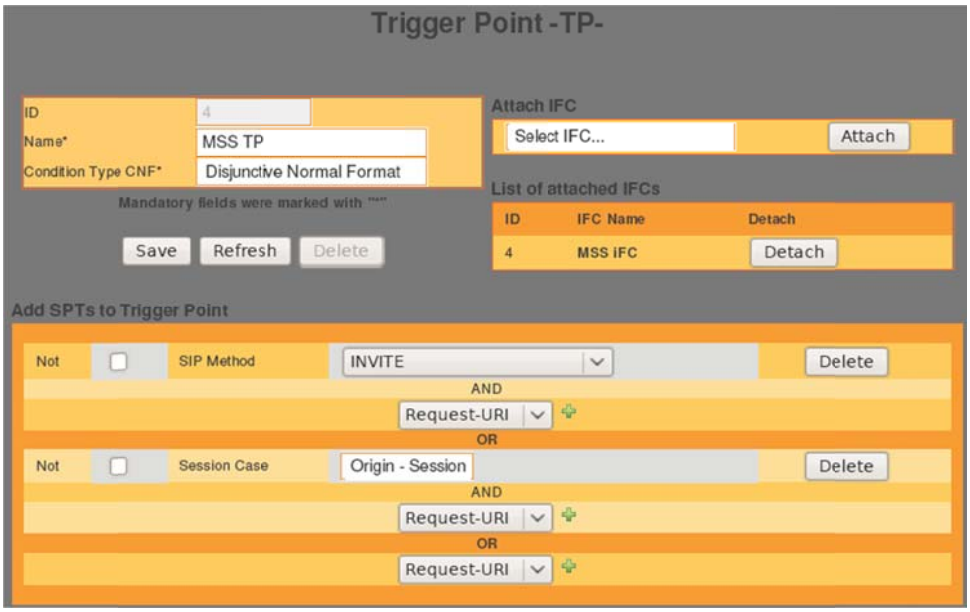
Após clicar no Tab "Services", deve clicar na opção "Create" dos Aplication Servers localizada no painel esquerdo. Aí deve:

- Introduzir o nome do AS: MSS
- Introduzir no Server Name o endereço SIP onde a aplicação se encontra, indicando também o porto de escuta
- No campo Diameter FQDN (Fully Qualified Domain Name) deve colocar-se o endereço de destino do *destination host*.
- Devem ainda ser configuradas as permissões de acesso ao AS, onde UDR, PUR e SNR significam respectivamente User-Data-Request, Profile-Update-Request e Subscribe-Notification-Request, representando os comandos da interface Sh que o AS pode utilizar para comunicar com o HSS (explicados anteriormente na Secção 3 – Interface Diameter Sh).

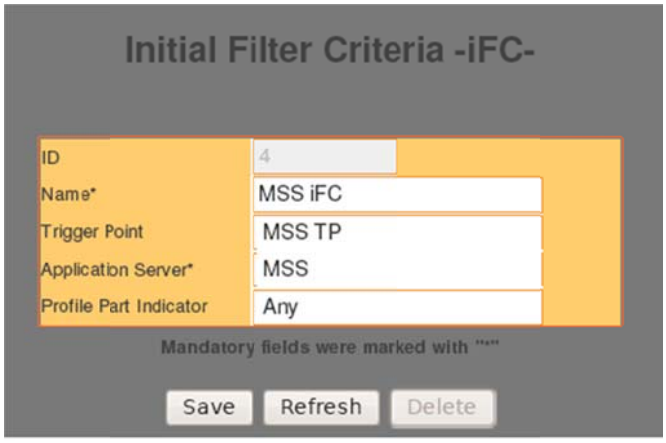
A figura seguinte auxilia a definição do AS.



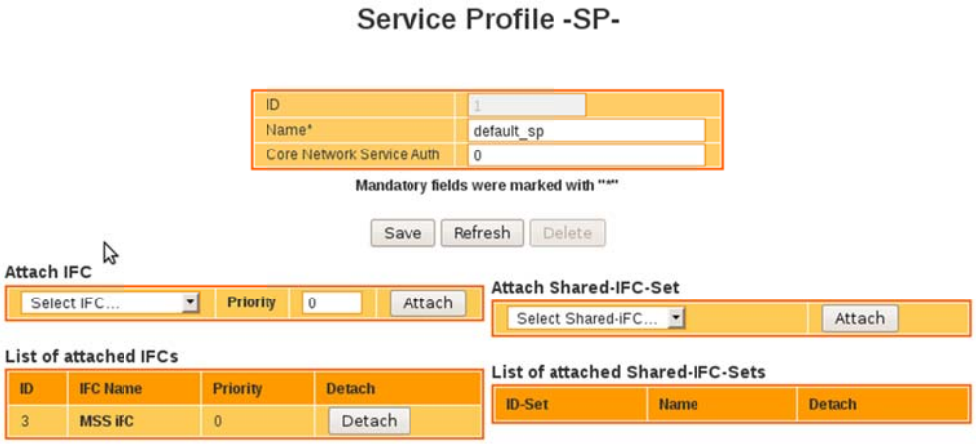
Seguidamente devem especificar-se os *triggers* que o AS deverá utilizar. No caso da aplicação de exemplificada na figura, o AS só pretende receber eventos relacionados com o método SIP INVITE ou com a sessão do terminal que origina a sessão (*Origin*). A figura seguinte auxilia a configuração.



A associação dos triggers TP ao AS é realizado através da definição de um Initial Filter Criteria, como é apresentado na figura seguinte.



Para finalizar a operação, deve clicar no botão Search no campo Service Profiles e clicando novamente no botão Search deverá depois selecionar o perfil default_sp. Neste passo deverá realizar o attach do MSS iFC, tal como se apresenta na figura seguinte.



Anexo V – Mensagens SIP

Pedidos

Method	Description
REGISTER	This method is used to provides the Registrar with information specifying the UA's location and available for incoming SIP requests. When the user agent's location changes, another REGISTER message is sent to update the Registrar's database.
INVITE	This method is used to initiate a communication session between two UA peers. sent message is sent by a user to initiate a session with another peer user. INVITE can also be used to initiate a multi party call.
ACK	This method is used for acknowledgement. It indicates that the final response has been received.
CANCEL	This method is used to terminate pending requests. A calling party can cancel an INVITE message before it receives the final response.
OPTIONS	This method is used to query a server on its capabilities. For example, it can be used to query if a to-be-called party can support a particular type of media.
BYE	This method is used to indicate the termination of a session.
INFO	This method (an extension - RFC 2976) is used to communicate additional information about an active session. For example, it can be used to inform a user agent when available pre-paid balance is approaching zero.
REFER	This method (an extension - RFC 3515) is used to provide the recipient with a third party's contact information. This method can be used for call transfers.
UPDATE	This new method (RFC 3311) is used to change session information (such as a change in codec) before a final response to a SIP INVITE message has been received. Typically, UPDATE messages contain an SDP body that lists the modified session parameters.
SUBSCRIBE and NOTIFY	These are two new SIP methods (RFC 3265) that are used in conjunction with each other for SIP based event notification. A user can subscribe to events such as the Presence information of another user. Subscribe Method s sent to the SIP Presence Server, so when the second user becomes available, the Presence Server sends a NOTIFY.

Respostas

Code range	Description
1xx	► Provisional/Informative response Provisional response indicate that the associated request was received and being processed. Upon receipt of a provisional response, the request sender should stop retransmitting the request. For example, a proxy server can send a response message with a Status-code of 100 upon receipt of an INVITE request. Provisional responses need not be acknowledged with an ACK.
2xx	► Success Success responses with Status-codes in the range from 200 to 299 indicate that the request was received, understood and successfully processed. For example, a 200 OK response is sent to a User Agent Client when an INVITE request is successfully processed.
3xx	► Redirection When further action such as a different location is needed to complete a request, redirection responses are used to provide the new location or an alternative service that would satisfy the request. Redirection responses are usually sent by redirect servers. When a redirect server receives a request, it maps the destination address in the request message to one or more addresses retrieved from the Registrar location database and returns the new address list to the originator of the request. The originator is then supposed to re-send the request to the new location.
4xx	► Client error Client error response Status-codes are sent when requests cannot be processed. The request failure could be because of bad syntax in the request message or simply because the request cannot be fulfilled by the responding server.
5xx	► Server error Server error response Status-codes are sent in cases where the request is valid but the server s unable to fulfill the request. Server internal error (500) and Not implemented (501) are two examples of Server error response Status-codes.
6xx	► Global failure When a request cannot be fulfilled by any server, the Global failure response Status-codes are returned. A User Agent Server can return a global failure response with Status-code 603 to Decline a request to participate in a session.

Códigos de respostas

1xx—Informational Responses

100 Trying - extended search being performed may take a significant time so a forking proxy must send a 100 Trying response
180 Ringing
181 Call Is Being Forwarded
182 Queued
183 Session Progress

2xx—Successful Responses

200 OK
202 accepted: It Indicates that the request has been understood but actually can't be processed

3xx—Redirection Responses

300 Multiple Choices
301 Moved Permanently
302 Moved Temporarily
305 Use Proxy
380 Alternative Service

4xx—Client Failure Responses

400 Bad Request
401 Unauthorized (Used only by registrars or user agents. Proxies should use proxy authorization 407)
402 Payment Required (Reserved for future use)
403 Forbidden
404 Not Found (User not found)
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout (Couldn't find the user in time)
409 Conflict
410 Gone (The user existed once, but is not available here any more.)
412 Conditional Request Failed
413 Request Entity Too Large
414 Request-URI Too Long
415 Unsupported Media Type
416 Unsupported URI Scheme
417 Unknown Resource-Priority
420 Bad Extension (Bad SIP Protocol Extension used, not understood by the server)
421 Extension Required
422 Session Interval Too Small
423 Interval Too Brief
424 Bad Location Information
428 Use Identity Header
429 Provide Referrer Identity
433 Anonymity Disallowed
436 Bad Identity-Info
437 Unsupported Certificate
438 Invalid Identity Header
480 Temporarily Unavailable
481 Call/Transaction Does Not Exist
482 Loop Detected
483 Too Many Hops
484 Address Incomplete
485 Ambiguous
486 Busy Here
487 Request Terminated
488 Not Acceptable Here
489 Bad Event
491 Request Pending
493 Undecipherable (Could not decrypt S/MIME body part)
494 Security Agreement Required

5xx—Server Failure Responses

500 Server Internal Error
501 Not Implemented: The SIP request method is not implemented here
502 Bad Gateway
503 Service Unavailable
504 Server Time-out
505 Version Not Supported: The server does not support this version of the SIP protocol
513 Message Too Large
580 Precondition Failure

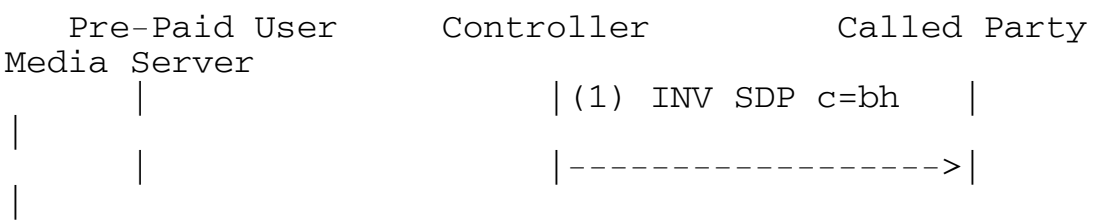
6xx—Global Failure Responses

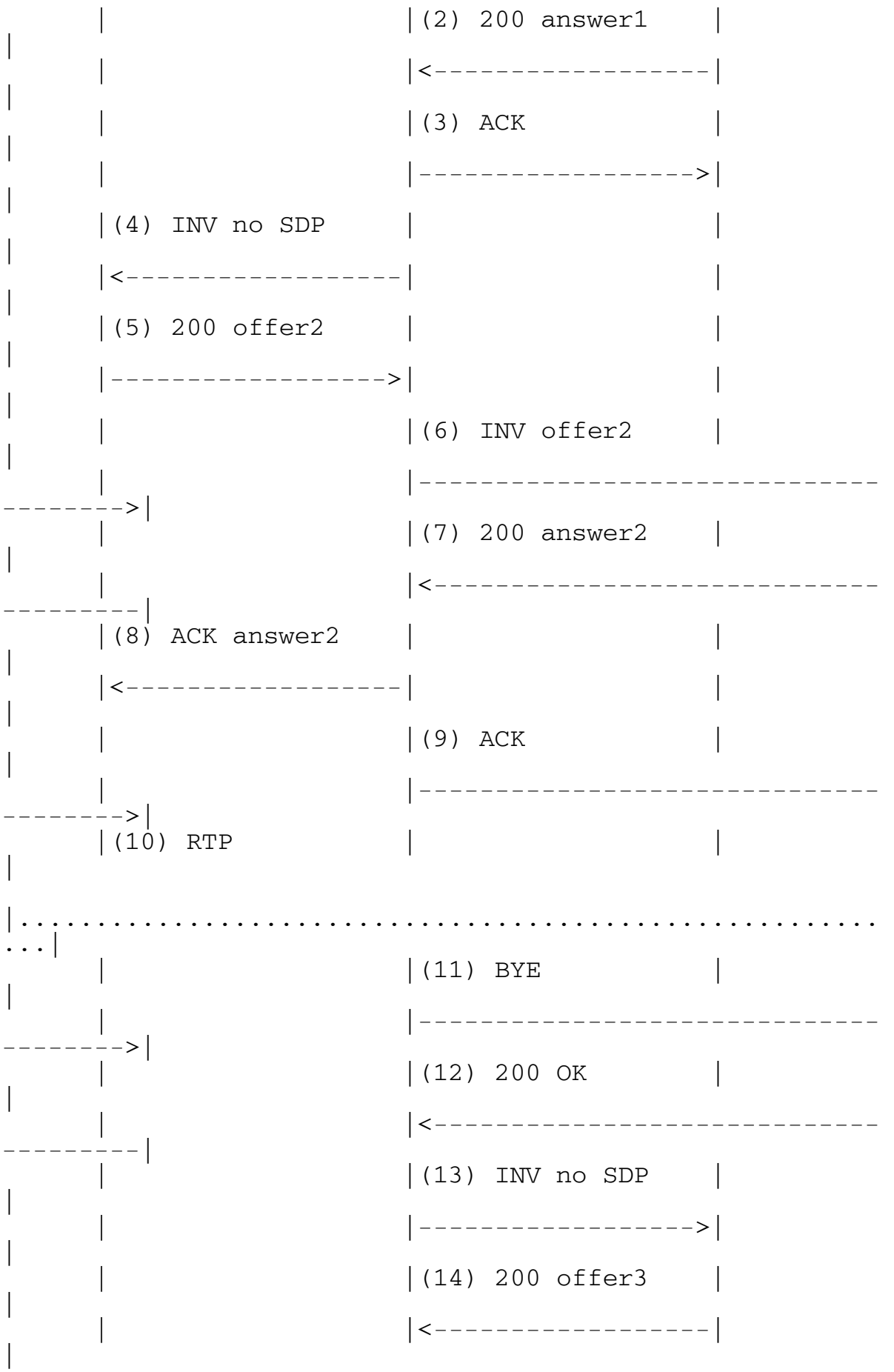
600 Busy Everywhere
603 Decline
604 Does Not Exist Anywhere
606 Not Acceptable

10.2. Mid-Call Announcement Capability

The third party call control mechanism described here can also be used to enable mid-call announcements. Consider a service for pre-paid calling cards. Once the pre-paid call is established, the system needs to set a timer to fire when they run out of minutes. When this timer fires, we would like the user to hear an announcement which tells them to enter a credit card to continue. Once they enter the credit card info, more money is added to the pre-paid card, and the user is reconnected to the destination party.

We consider here the usage of third party call control just for playing the mid-call dialog to collect the credit card information.




(2) 200 answer1
(3) ACK
(4) INV no SDP
(5) 200 offer2
(6) INV offer2
(7) 200 answer2
(8) ACK answer2
(9) ACK
(10) RTP
(11) BYE
(12) 200 OK
(13) INV no SDP
(14) 200 offer3

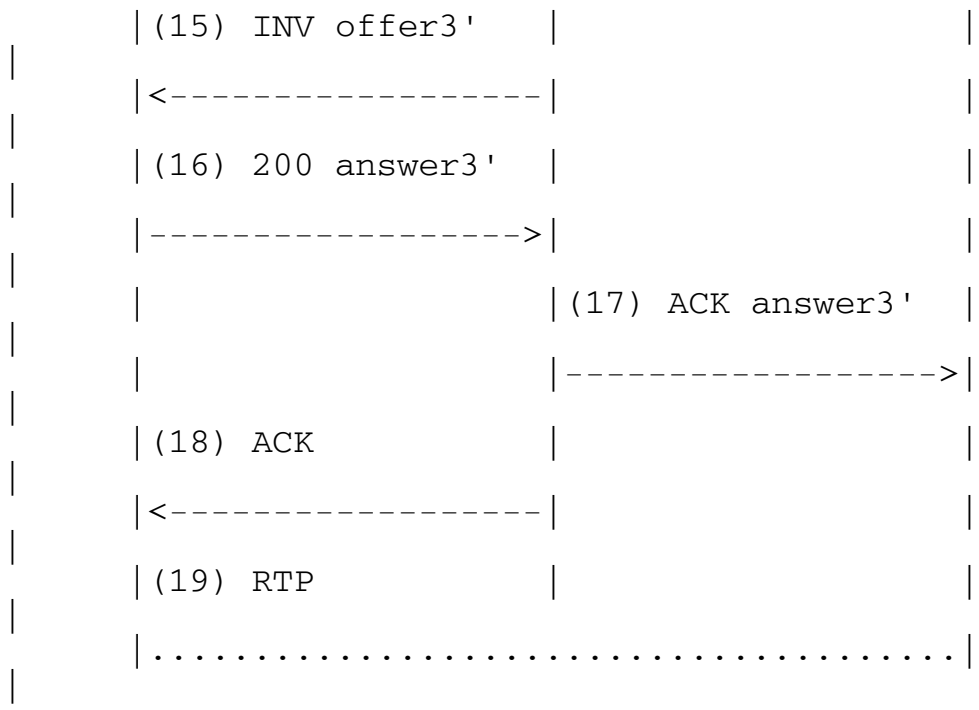


Figure 13

We assume the call is set up so that the controller is in the call as a B2BUA. When the timer fires, we wish to connect the caller to a media server. The flow for this is shown in Figure 13. When the timer expires, the controller places the called party with a connection address of 0.0.0.0 (1). This effectively "disconnects" the called party. The controller then sends an INVITE without SDP to

the pre-paid caller (4). The offer returned from the caller (5) is used in an INVITE to the media server which will be collecting digits (6). This is an instantiation of Flow I. This flow can only be used here because the media server is an automata, and will answer the

INVITE immediately. If the controller was connecting the pre-paid user with another end user, Flow III would need to be used. The media server returns an immediate 200 OK (7) with an answer, which is passed to the caller in an ACK (8). The result is that the media server and the pre-paid caller have their media streams connected.

The media server plays an announcement, and prompts the user to enter a credit card number. After collecting the number, the card number is validated. The media server then passes the card number to the controller (using some means outside the scope of this specification), and then hangs up the call (11).

After hanging up with the media server, the controller reconnects the user to the original called party. To do this, the controller sends an INVITE without SDP to the called party (13). The 200 OK (14) contains an offer, offer3. The controller modifies the SDP (as is done in Flow III), and passes the offer in an INVITE to the pre-paid user (15). The pre-paid user generates an answer in a 200 OK (16) which the controller passes to user B in the ACK (17). At this point, the caller and called party are reconnected.