Mobile Communications Chapter 10: Support for Mobility

- File systems
- Data bases
- WWW and Mobility
- WAP (Wireless Application Protocol), i-mode & Co.

File systems - Motivation

Goal

- efficient and transparent access to shared files within a mobile environment while maintaining data consistency

Problems

- limited resources of mobile computers (memory, CPU, ...)
- low bandwidth, variable bandwidth, temporary disconnection
- high heterogeneity of hardware and software components (no standard PC architecture)
- wireless network resources and mobile computer are not very reliable
- standard file systems (e.g., NFS, network file system) are very inefficient, almost unusable

Solutions

- replication of data (copying, cloning, caching)
- data collection in advance (hoarding, pre-fetching)

File systems - consistency problems

THE big problem of distributed, loosely coupled systems

- are all views on data the same?
- how and when should changes be propagated to what users?

Weak consistency

- many algorithms offering strong consistency (e.g., via atomic updates) cannot be used in mobile environments
- invalidation of data located in caches through a server is very problematic if the mobile computer is currently not connected to the network
- occasional inconsistencies have to be tolerated, but conflict resolution strategies must be applied afterwards to reach consistency again

Conflict detection

- content independent: version numbering, time-stamps
- content dependent: dependency graphs

File systems for limited connectivity I

Symmetry

- Client/Server or Peer-to-Peer relations
- support in the fixed network and/or mobile computers
- one file system or several file systems
- one namespace for files or several namespaces

Transparency

- hide the mobility support, applications on mobile computers should not notice the mobility
- user should not notice additional mechanisms needed

Consistency model

- optimistic or pessimistic

Caching and Pre-fetching

- single files, directories, subtrees, partitions, ...
- permanent or only at certain points in time

File systems for limited connectivity II

Data management

- management of buffered data and copies of data
- request for updates, validity of data
- detection of changes in data

Conflict solving

- application specific or general
- errors

Several experimental systems exist

- Coda (Carnegie Mellon University), Little Work (University of Michigan), Ficus (UCLA) etc.

Many systems use ideas from distributed file systems such as, e.g., AFS (Andrew File System)

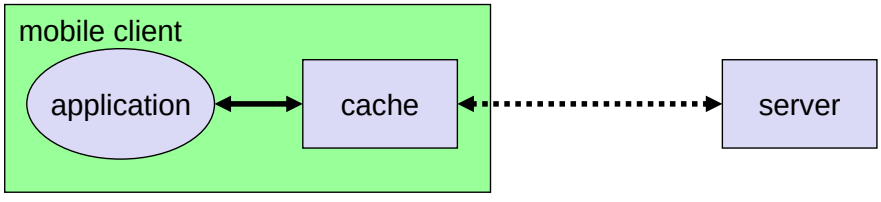
File systems - Coda I

Application transparent extensions of client and server

- changes in the cache manager of a client
- applications use cache replicates of files
- extensive, transparent collection of data in advance for possible future use („Hoarding“)

Consistency

- system keeps a record of changes in files and compares files after reconnection
- if different users have changed the same file a manual reintegration of the file into the system is necessary
- optimistic approach, coarse grained (file size)



File systems - Coda II

Hoarding

- user can pre-determine a file list with priorities
- contents of the cache determined by the list and LRU strategy (Last Recently Used)
- explicit pre-fetching possible
- periodic updating

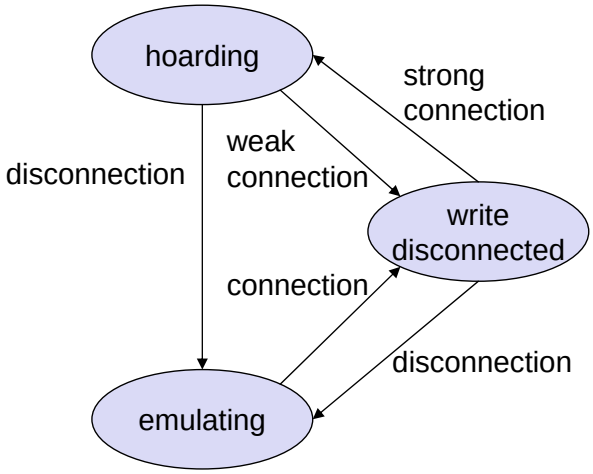
Comparison of files

- asynchronous, background
- system weighs speed of updating against minimization of network traffic

Cache misses

- modeling of user patience: how long can a user wait for data without an error message?
- function of file size and bandwidth

States of a client



File systems - Little Work

- Only changes in the cache manager of the client
- Connection modes and use

	Connected	**Partially Connected**	**Fetch only**	**Disconnected**
Method	normal	delayed write to the server	optimistic replication of files	abort at cache miss
Network requirements	continuous high bandwidth	continuous bandwidth	connection on demand	none
Application	office, WLAN	packet radio	cellular systems (e.g., GSM) with costs per call	independent

File systems - further examples

Mazer/Tardo

- file synchronization layer between application and local file system
- caching of complete subdirectories from the server
- “Redirector” responses to requests locally if necessary, via the network if possible
- periodic consistency checks with bi-directional updating

Ficus

- not a client/server approach
- optimistic approach based on replicates, detection of write conflicts, conflict resolution
- use of „gossip“ protocols: a mobile computer does not necessarily need to have direct connection to a server, with the help of other mobile computers updates can be propagated through the network

MIo-NFS (Mobile Integration of NFS)

- NFS extension, pessimistic approach, only token holder can write
- connected/loosely connected/disconnected

Database systems in mobile environments

Request processing

- power conserving, location dependent, cost efficient
- example: find the fastest way to a hospital

Replication management

- similar to file systems

Location management

- tracking of mobile users to provide replicated or location dependent data in time at the right place (minimize access delays)
- example: with the help of the HLR (Home Location Register) in GSM a mobile user can find a local towing service

Transaction processing

- “mobile” transactions can not necessarily rely on the same models as transactions over fixed networks (ACID: atomicity, consistency, isolation, durability)
- therefore models for “weak” transaction

World Wide Web and mobility

Protocol (HTTP, Hypertext Transfer Protocol) and language (HTML, Hypertext Markup Language) of the Web have not been designed for mobile applications and mobile devices, thus creating many problems!

Typical transfer sizes

- HTTP request: 100-350 byte
- responses avg. <10 kbyte, header 160 byte, GIF 4.1kByte, JPEG 12.8 kbyte, HTML 5.6 kbyte
- but also many large files that cannot be ignored

The Web is no file system

- Web pages are not simple files to download
- static and dynamic content, interaction with servers via forms, content transformation, push technologies etc.
- many hyperlinks, automatic loading and reloading, redirecting
- a single click might have big consequences!

WWW example

Request to port 80:
or:

```
GET / HTTP/1.0
GET / HTTP/1.1
Host: www.inf.fu-berlin.de
```

Response from server

non persistent

```
HTTP/1.1 200 OK
Date: Wed, 30 Oct 2002 19:44:26 GMT
Server: Apache/1.3.12 (Unix) mod_perl/1.24
Last-Modified: Wed, 30 Oct 2002 13:16:31 GMT
ETag: "2d8190-2322-3dbfdbaf"
Accept-Ranges: bytes
Content-Length: 8994
Connection: close
Content-Type: text/html

<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>FU-Berlin: Institut f&uuml;r Informatik</TITLE>
    <base href="http://www.inf.fu-berlin.de">
    <link rel="stylesheet" type="text/css" href="http://www.inf.fu-
    berlin.de/styles/homepage.css">
    <!--script language="JavaScript" src="fuinf.js"-->
    <!--/script-->
  </head>

  <body onResize="self.location.reload();">
...
```

HTTP 1.0 and mobility I

Characteristics

- stateless, client/server, request/response
- needs a connection oriented protocol (TCP), one connection per request (some enhancements in HTTP 1.1)
- primitive caching and security

Problems

- designed for large bandwidth (compared to wireless access) and low delay
- big and redundant protocol headers (readable for humans, stateless, therefore big headers in ASCII)
- uncompressed content transfer
- using TCP
  - huge overhead per request (3-way-handshake) compared with the content, e.g., of a GET request
  - slow-start problematic
- DNS lookup by client causes additional traffic

HTTP 1.0 and mobility II

Caching

- quite often disabled by information providers to be able to create user profiles, usage statistics etc.
- dynamic objects cannot be cached
  - numerous counters, time, date, personalization, ...
- mobility quite often inhibits caches
- security problems
  - how to use SSL/TLS together with proxies?
- today: many user customized pages, dynamically generated on request via CGI, ASP, ...

POSTing (i.e., sending to a server)

- can typically not be buffered, very problematic if currently disconnected

Many unsolved problems!

HTML and mobile devices

HTML

- designed for computers with “high” performance, color high-resolution display, mouse, hard disk
- typically, web pages optimized for design, not for communication

Mobile devices

- often only small, low-resolution displays, very limited input interfaces (small touch-pads, soft-keyboards)

Additional “features”

- animated GIF, Java AWT, Frames, ActiveX Controls, Shockwave, movie clips, audio, ...
- many web pages assume true color, multimedia support, high-resolution and many plug-ins

Web pages ignore the heterogeneity of end-systems!

- e.g., without additional mechanisms, large high-resolution pictures would be transferred to a mobile phone with a low-resolution display causing high costs

Approaches toward WWW for mobile devices

Application gateways, enhanced servers

- simple clients, pre-calculations in the fixed network
- compression, filtering, content extraction
- automatic adaptation to network characteristics

Examples

- picture scaling, color reduction, transformation of the document format (e.g., PS to TXT)
- detail studies, clipping, zoom
- headline extraction, automatic abstract generation
- HDML (handheld device markup language): simple language similar to HTML requiring a special browser
- HDTP (handheld device transport protocol): transport protocol for HDML, developed by Unwired Planet

Problems

- proprietary approaches, require special enhancements for browsers
- heterogeneous devices make approaches more complicated

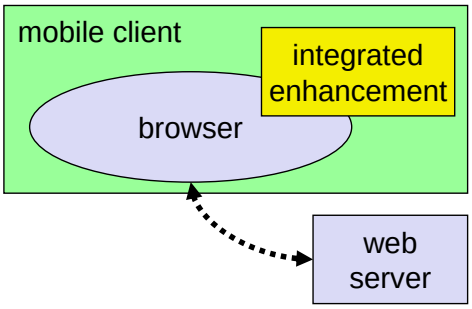
Some new issues that might help mobility?

- Push technology
  - real pushing, not a client pull needed, channels etc.
- HTTP/1.1
  - client/server use the same connection for several request/response transactions
  - multiple requests at beginning of session, several responses in same order
  - enhanced caching of responses (useful if equivalent responses!)
  - semantic transparency not always achievable: disconnected, performance, availability -> most up-to-date version...
  - several more tags and options for controlling caching (public/private, max-age, no-cache etc.)
  - relaxing of transparency on app. request or with warning to user
  - encoding/compression mechanism, integrity check, security of proxies, authentication, authorization...
- Cookies: well..., stateful sessions, not really integrated...

System support for WWW in a mobile world I

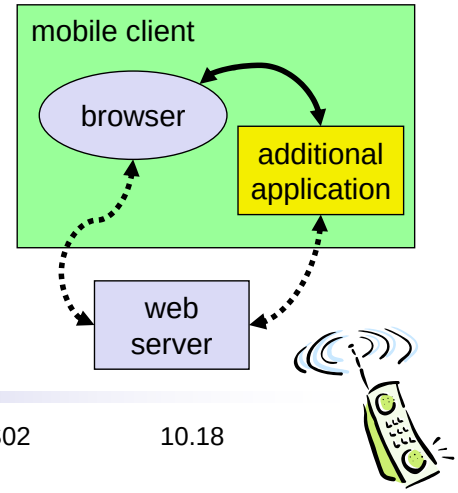
Enhanced browsers

- Pre-fetching, caching, off-line use
- e.g. Internet Explorer



Additional, accompanying application

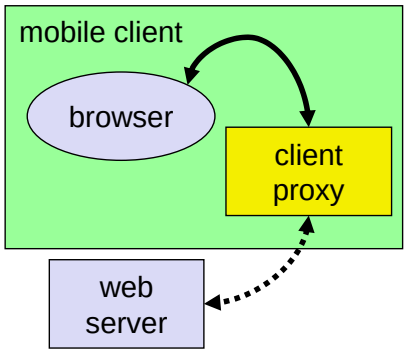
- Pre-fetching, caching, off-line use
- e.g. original WebWhacker



System support for WWW in a mobile world II

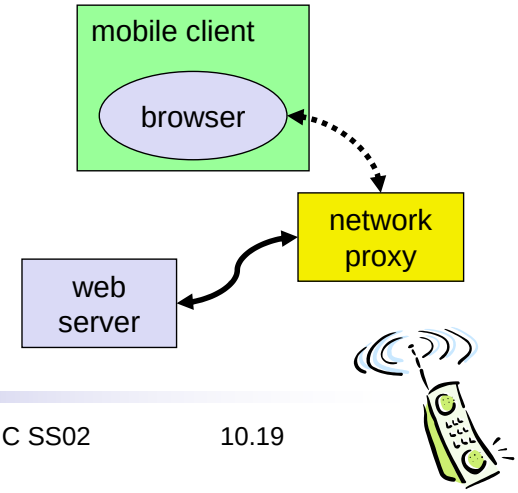
Client Proxy

- Pre-fetching, caching, off-line use
- e.g., Caubweb, TeleWeb, Weblicator, WebWhacker, WebEx, WebMirror, ...



Network Proxy

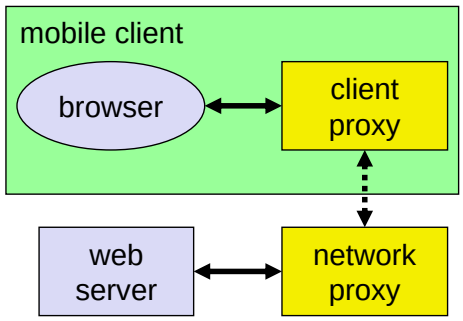
- adaptive content transformation for bad connections, pre-fetching, caching
- e.g., TranSend, Digestor



System support for WWW in a mobile world III

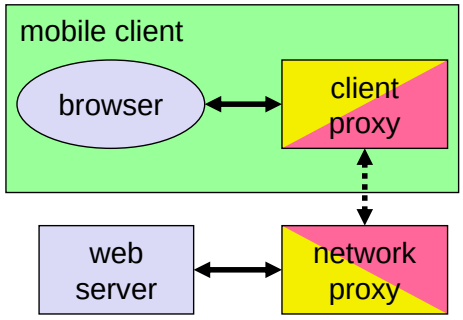
Client and network proxy

- combination of benefits plus simplified protocols
- e.g., MobiScape, WebExpress



Special network subsystem

- adaptive content transformation for bad connections, pre-fetching, caching
- e.g., Mowgli



Additional many proprietary server extensions possible

- “channels”, content negotiation, ...

WAP - Wireless Application Protocol

Goals

- deliver Internet content and enhanced services to mobile devices and users (mobile phones, PDAs)
- independence from wireless network standards
- open for everyone to participate, protocol specifications will be proposed to standardization bodies
- applications should scale well beyond current transport media and device types and should also be applicable to future developments

Platforms

- e.g., GSM (900, 1800, 1900), CDMA IS-95, TDMA IS-136, 3rd generation systems (IMT-2000, UMTS, W-CDMA, cdma2000 1x EV-DO, …)

Forum

- was: WAP Forum, co-founded by Ericsson, Motorola, Nokia, Unwired Planet, further information www.wapforum.org
- now: **Open Mobile Alliance** www.openmobilealliance.org (Open Mobile Architecture + WAP Forum + SyncML + …)

WAP - scope of standardization

Browser

- “micro browser”, similar to existing, well-known browsers in the Internet

Script language

- similar to Java script, adapted to the mobile environment

WTA/WTAI

- Wireless Telephony Application (Interface): access to all telephone functions

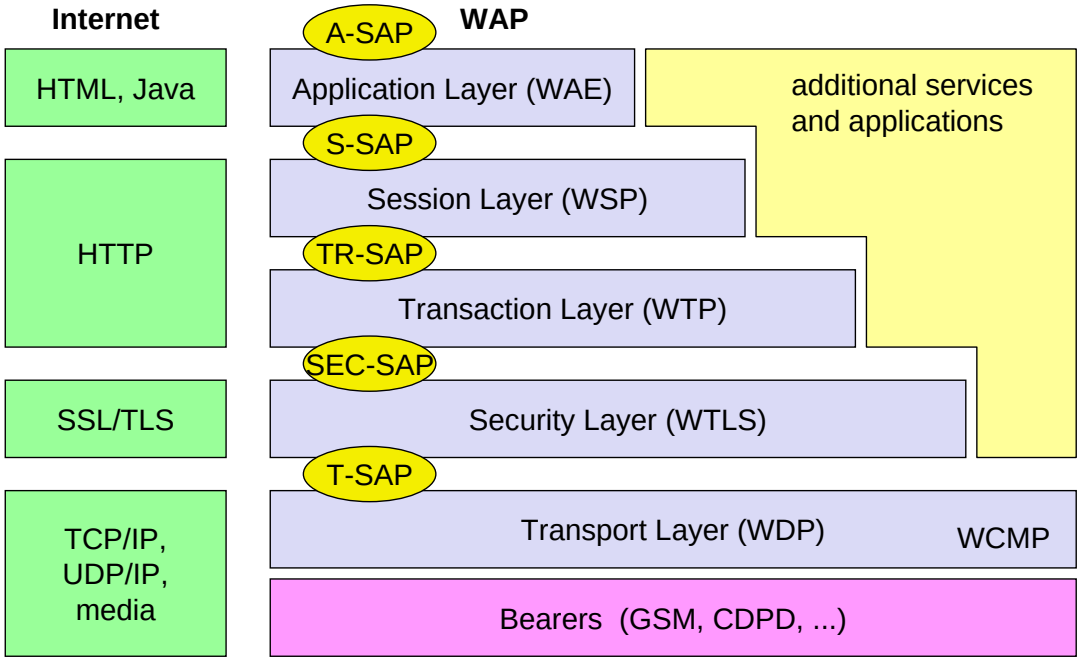
Content formats

- e.g., business cards (vCard), calendar events (vCalender)

Protocol layers

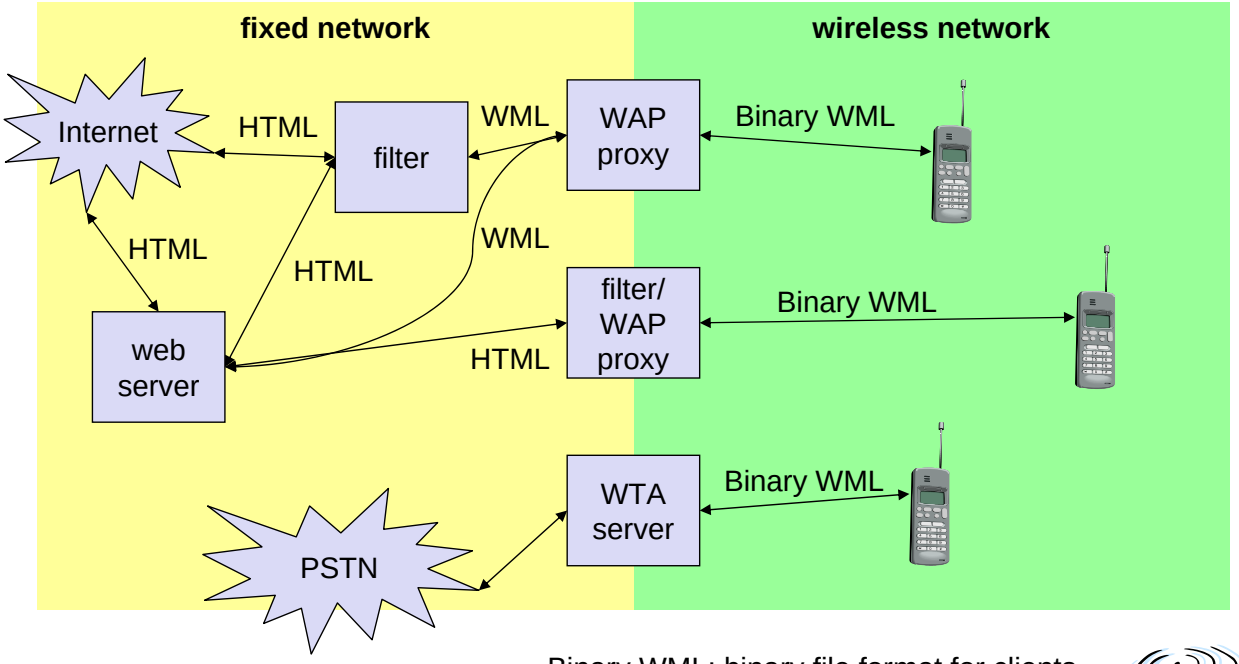
- transport layer, security layer, session layer etc.

WAP 1.x - reference model and protocols



WAE comprises WML (Wireless Markup Language), WML Script, WTAI etc.

WAP - network elements



Binary WML: binary file format for clients

WDP - Wireless Datagram Protocol

Protocol of the transport layer within the WAP architecture

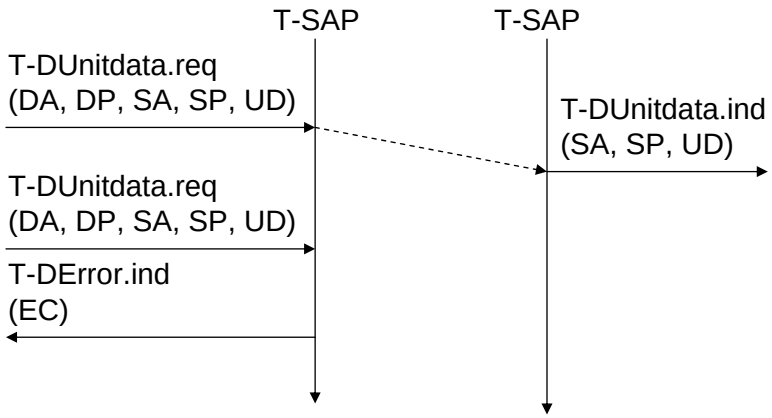
- uses directly transports mechanisms of different network technologies
- offers a common interface for higher layer protocols
- allows for transparent communication using different transport technologies (GSM [SMS, CSD, USSD, GPRS, ...], IS-136, TETRA, DECT, PHS, IS-95, ...)

Goals of WDP

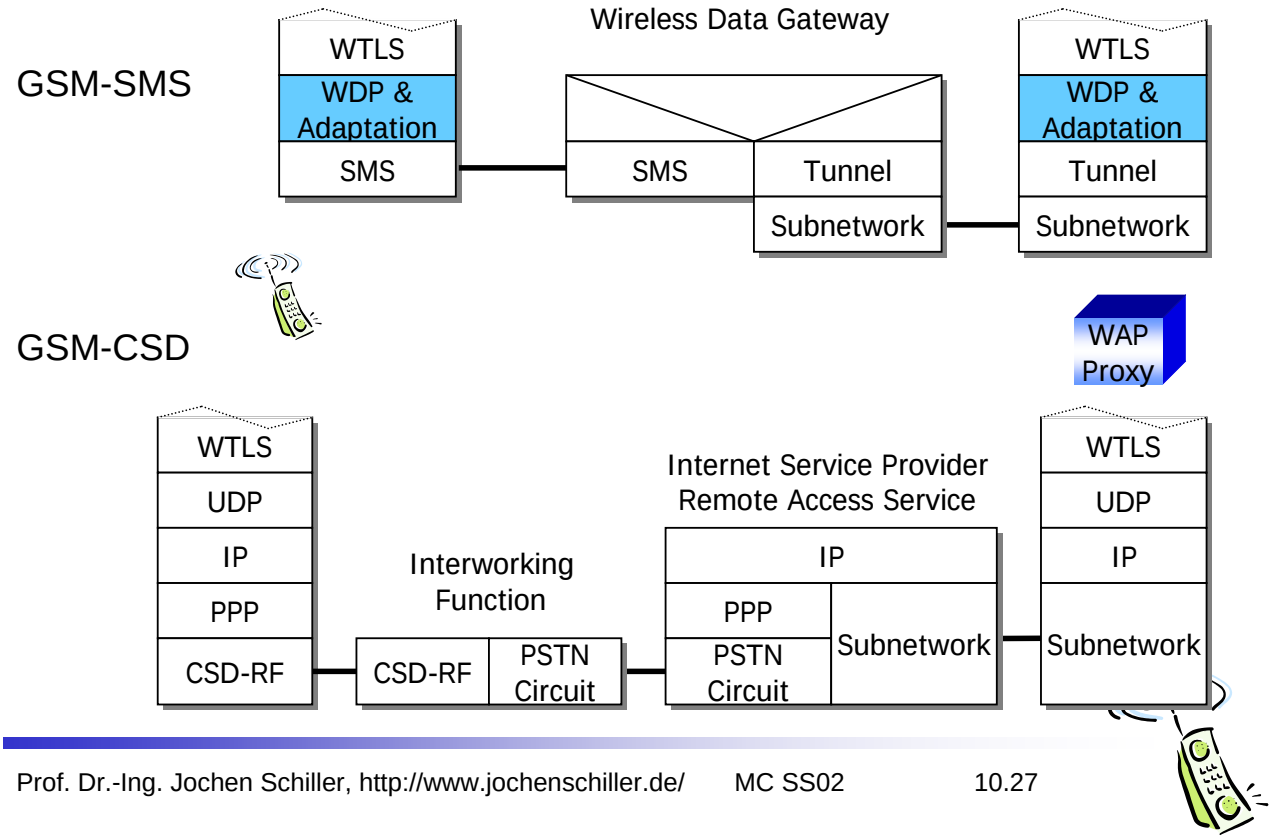
- create a worldwide interoperable transport system with the help of WDP adapted to the different underlying technologies
- transmission services such as SMS, GPRS in GSM might change, new services can replace the old ones

Additionally, WCMP (wireless Control Message Protocol) is used for control/error report (similar to ICMP in the TCP/IP protocol suite)

WDP - Service Primitives



Usage of WDP



WTLS - Wireless Transport Layer Security

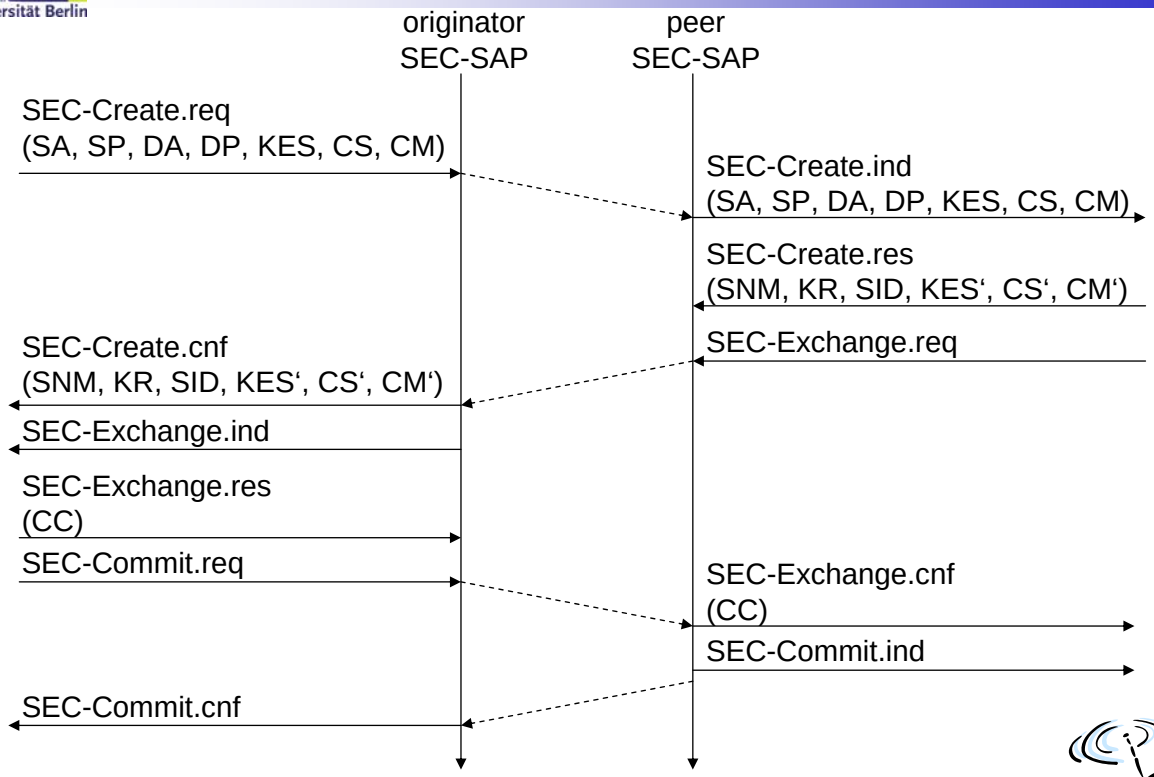
Goals

- data integrity
  - prevention of changes in data
- privacy
  - prevention of tapping
- authentication
  - creation of authenticated relations between a mobile device and a server
- protection against denial-of-service attacks
  - protection against repetition of data and unverified data

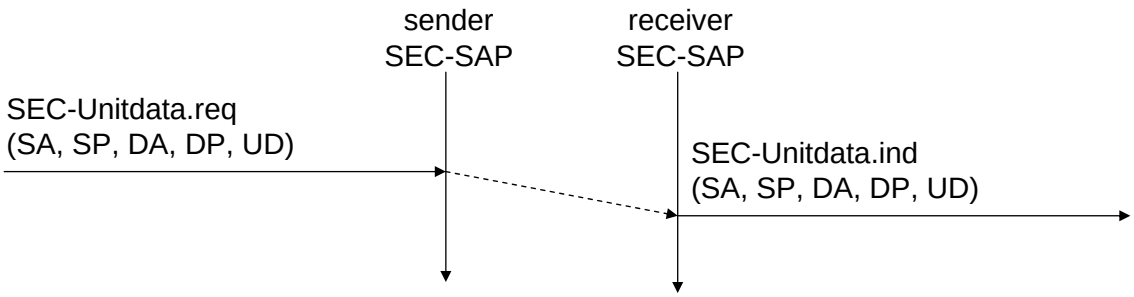
WTLS

- is based on the TLS (Transport Layer Security) protocol (former SSL, Secure Sockets Layer)
- optimized for low-bandwidth communication channels

Secure session, full handshake



SEC-Unitdata - transferring datagrams



WTP - Wireless Transaction Protocol

Goals

- different transaction services, offloads applications
  - application can select reliability, efficiency
- support of different communication scenarios
  - *class 0:* unreliable message transfer
  - *class 1:* reliable message transfer without result message
  - *class 2:* reliable message transfer with exactly one reliable result message
- supports peer-to-peer, client/server and multicast applications
- low memory requirements, suited to simple devices (< 10kbyte)
- efficient for wireless transmission
  - segmentation/reassembly
  - selective retransmission
  - header compression
  - optimized connection setup (setup with data transfer)

Details of WTP I

Support of different communication scenarios

- Class 0: unreliable message transfer
  - Example: push service
- Class 1: reliable request
  - An invoke message is not followed by a result message
  - Example: reliable push service
- Class 2: reliable request/response
  - An invoke message is followed by exactly one result message
  - With and without ACK
  - Example: typical web browsing

No explicit connection setup or release is available

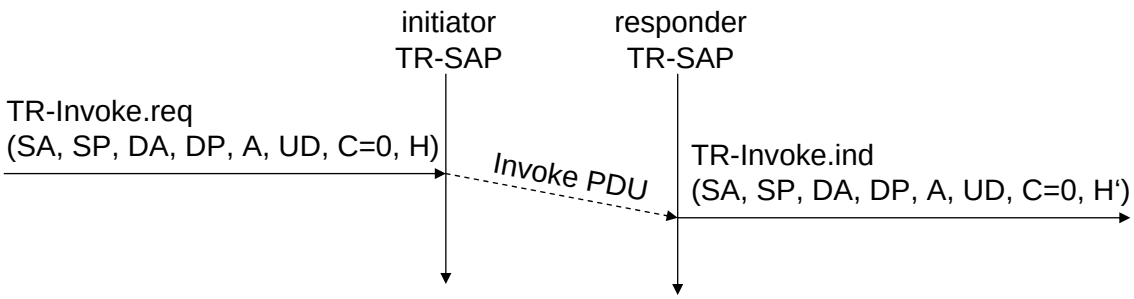
Services for higher layers are called events

Details of WTP II

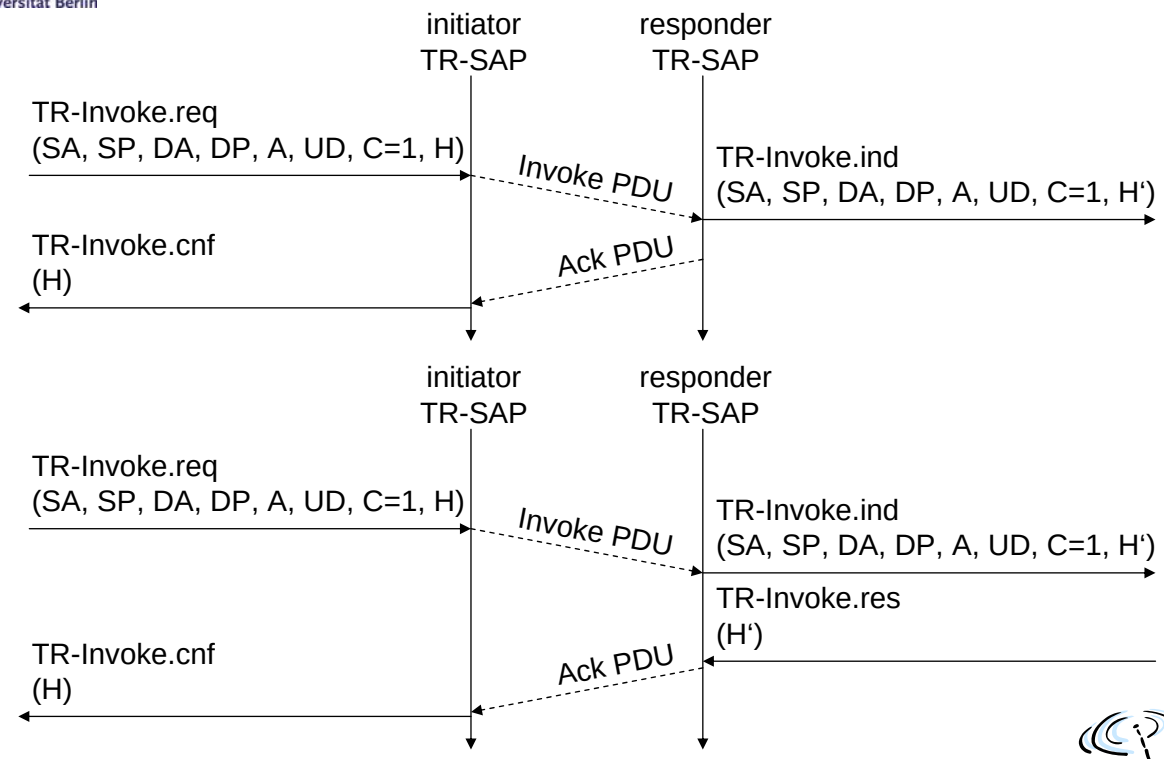
Used Mechanisms

- Reliability
  - Unique transaction identifiers (TID)
  - Acknowledgements
  - Selective retransmission
  - Duplicate removal
- Optional: concatenation & separation of messages
- Optional: segmentation & reassembly of messages
- Asynchronous transactions
- Transaction abort, error handling
- Optimized connection setup (includes data transmission)

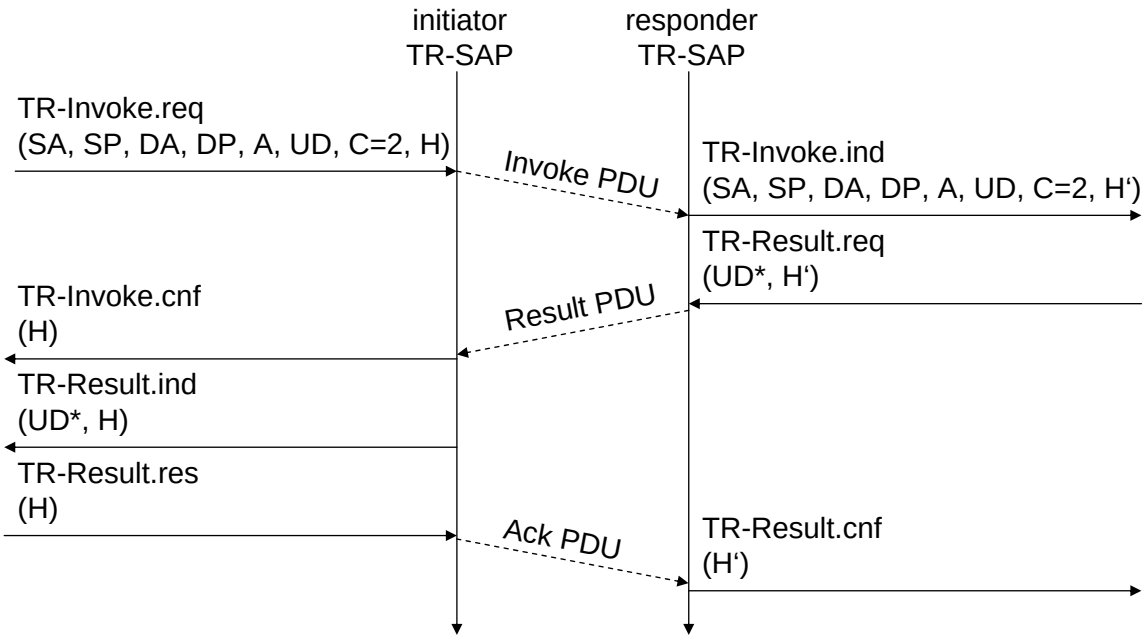
WTP Class 0 transaction



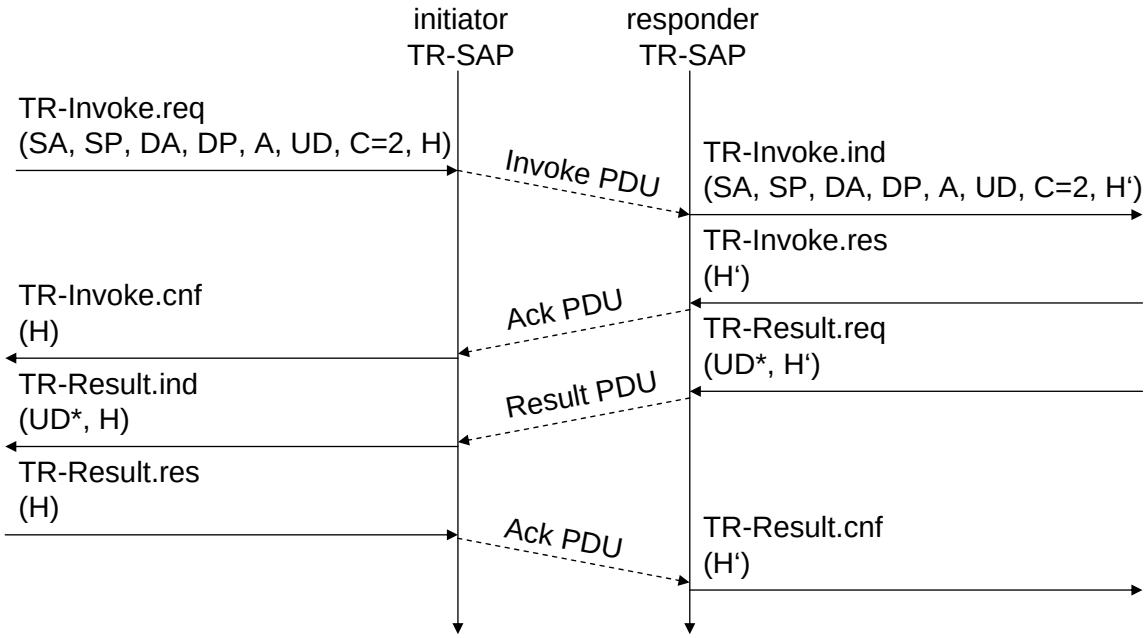
WTP Class 1 transaction, no user ack & user ack



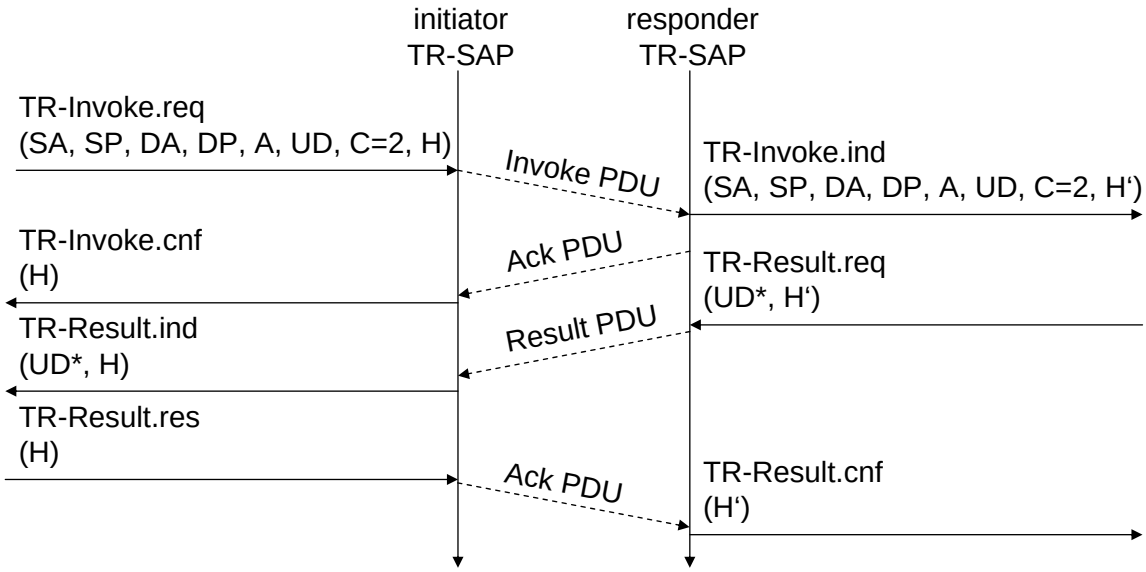
WTP Class 2 transaction, no user ack, no hold on



WTP Class 2 transaction, user ack



WTP Class 2 transaction, hold on, no user ack



WSP - Wireless Session Protocol

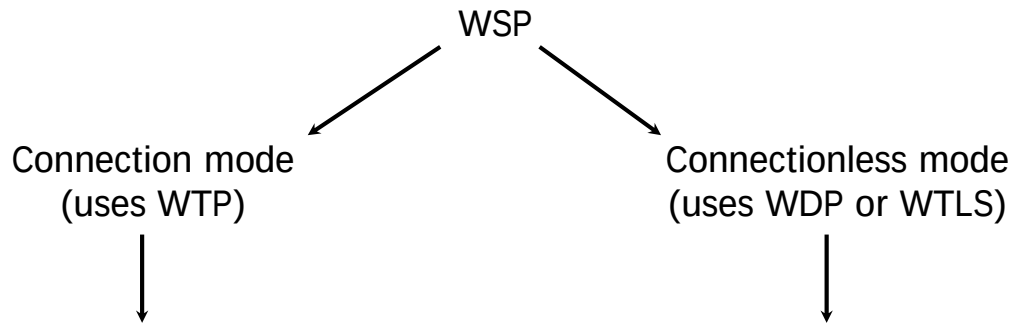
Goals

- HTTP 1.1 functionality
  - Request/reply, content type negotiation, ...
- support of client/server, transactions, push technology
- key management, authentication, Internet security services
- session management (interruption, resume,...)

Open topics

- QoS support)
- Group communication
- Isochronous media objects
- management

WSP protocols

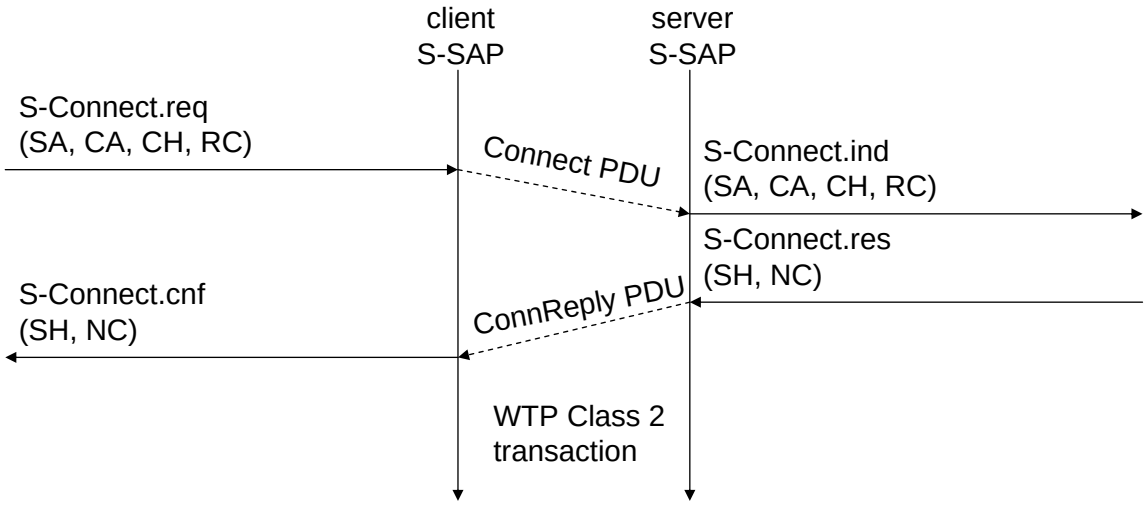


- Session Management (class 0, 2)
- Method Invocation (Kl. 2)
- Error Report
- Push (class 0)
- Confirmed Push (class 1)
- Session suspend/resume (class 0, 2)

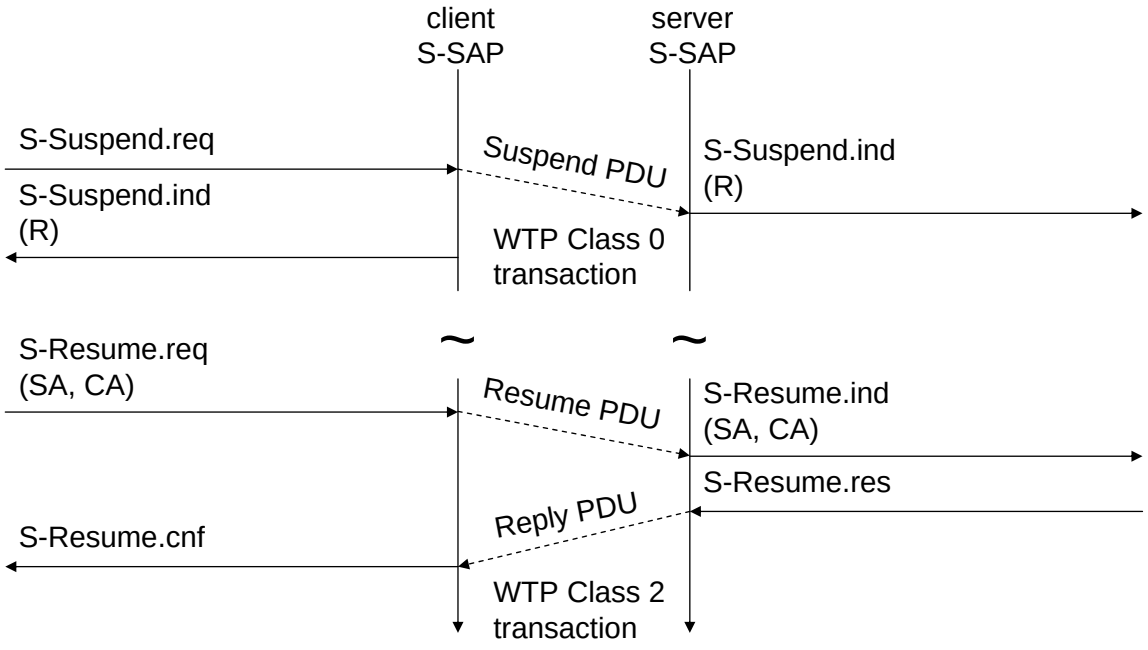
- Method Invocation
- Push

(in general unreliable)

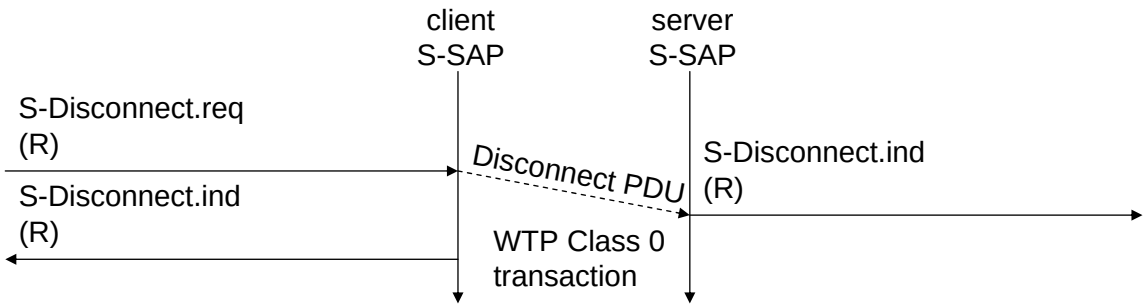
WSP/B session establishment



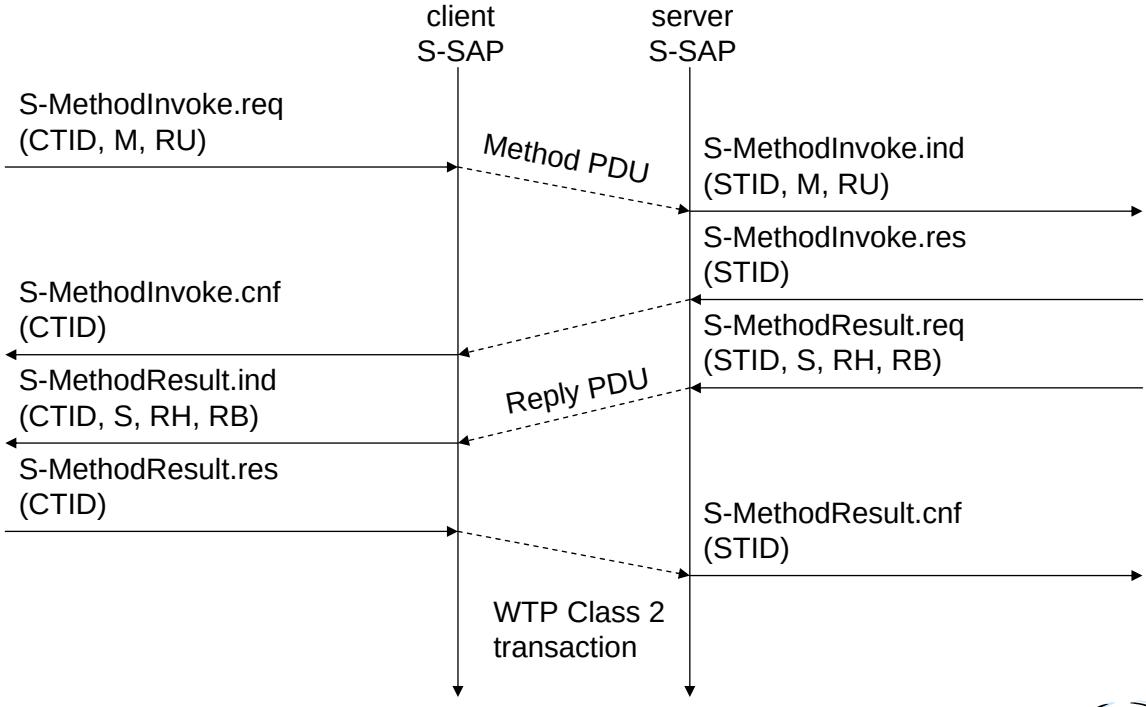
WSP/B session suspend/resume



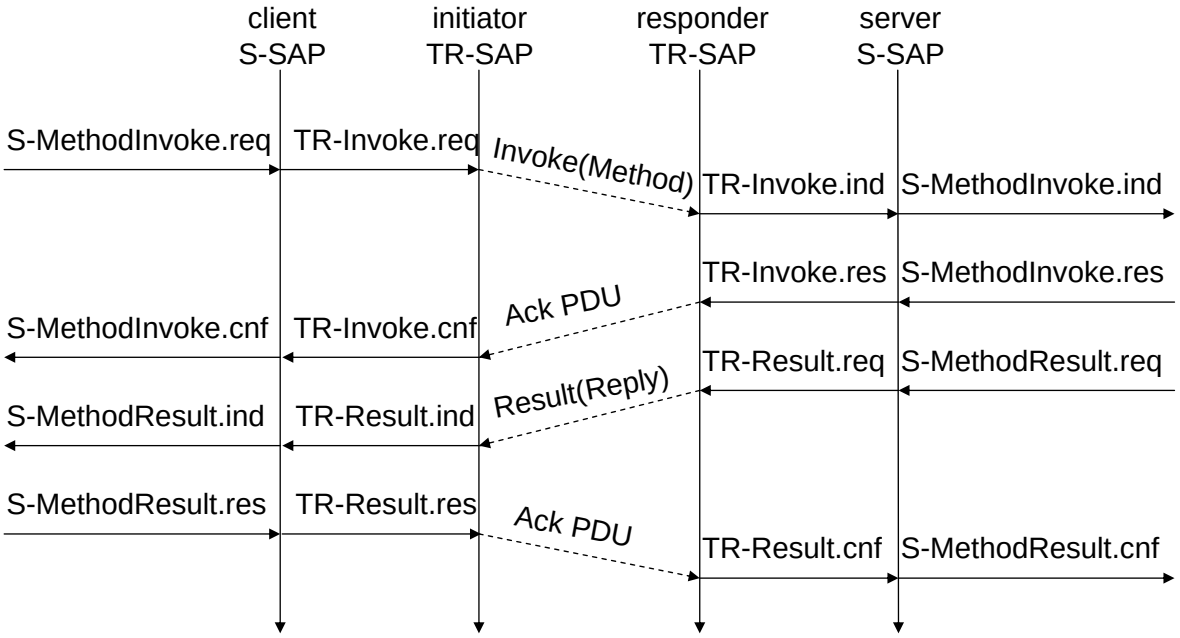
WSP/B session termination



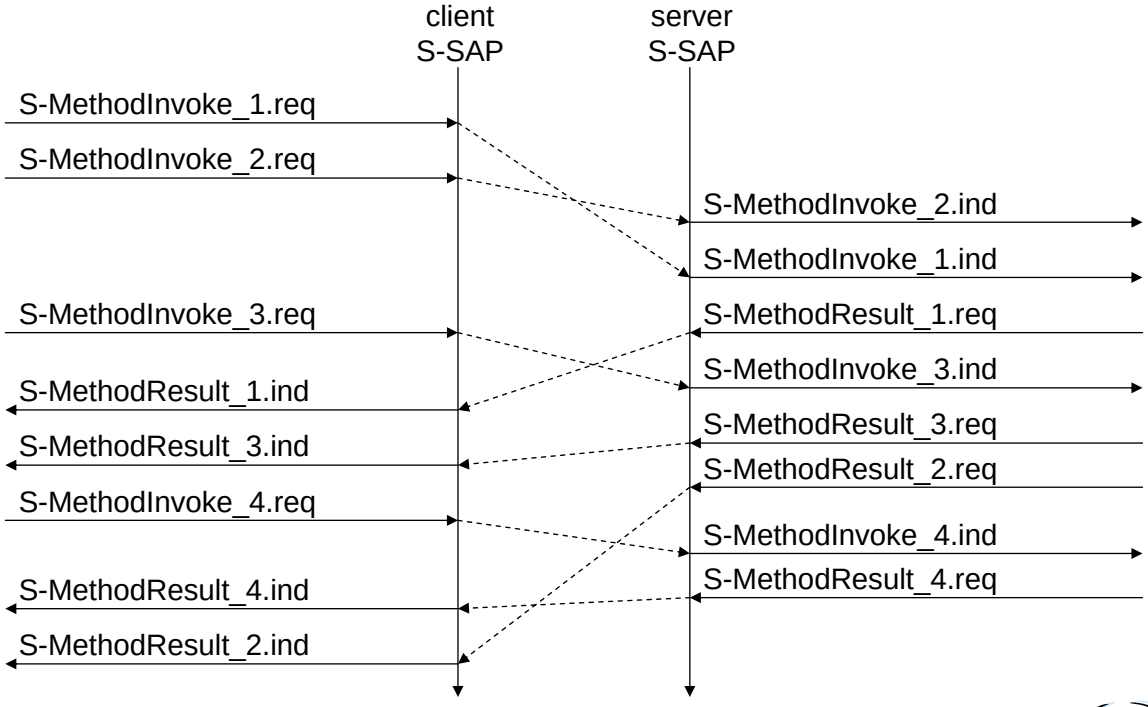
WSP/B method invoke



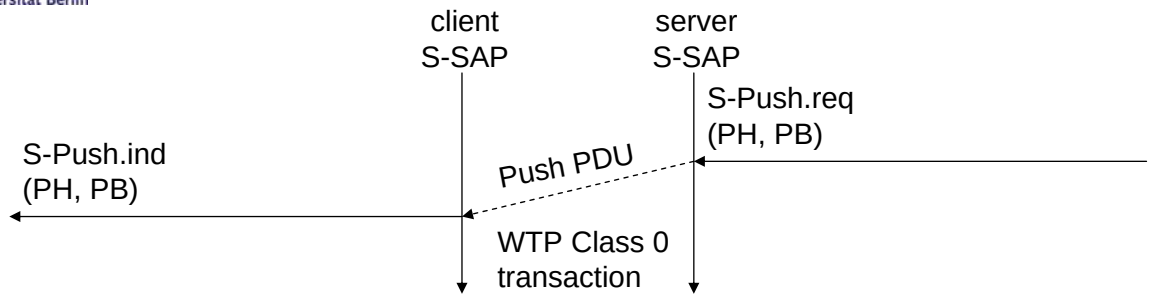
WSP/B over WTP - method invocation

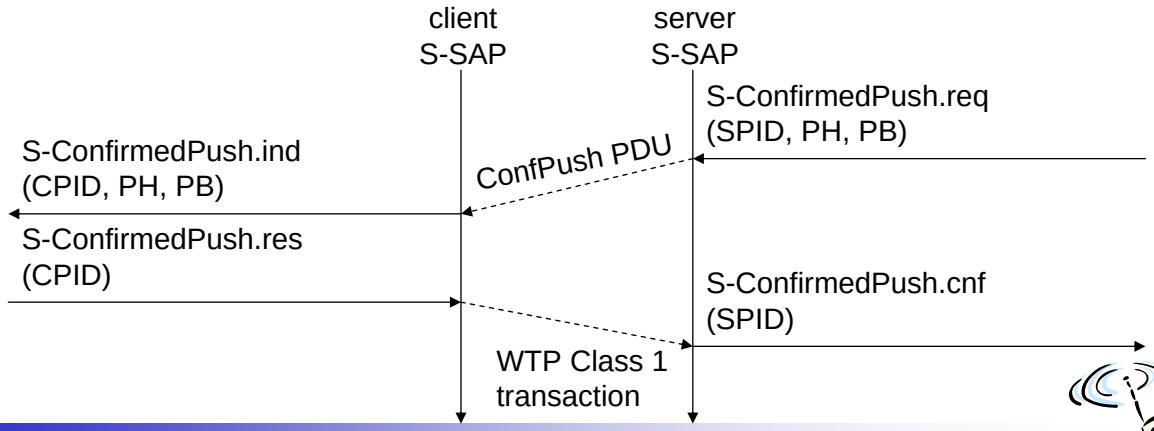


WSP/B over WTP - asynchronous, unordered requests

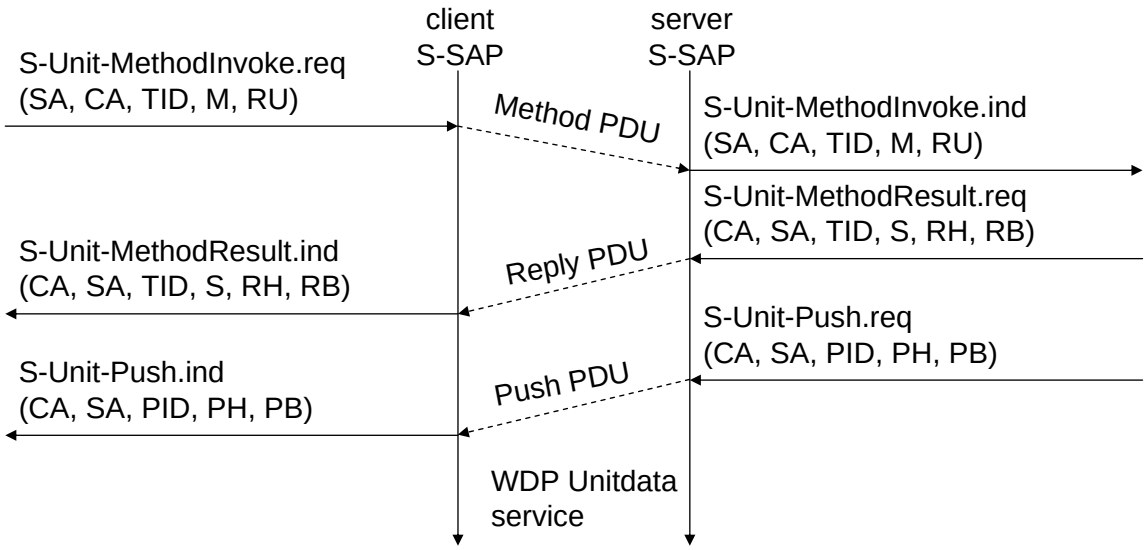


WSP/B - confirmend/non-confirmed push





WSP/B over WDP



WAE - Wireless Application Environment

Goals

- network independent application environment for low-bandwidth, wireless devices
- integrated Internet/WWW programming model with high interoperability

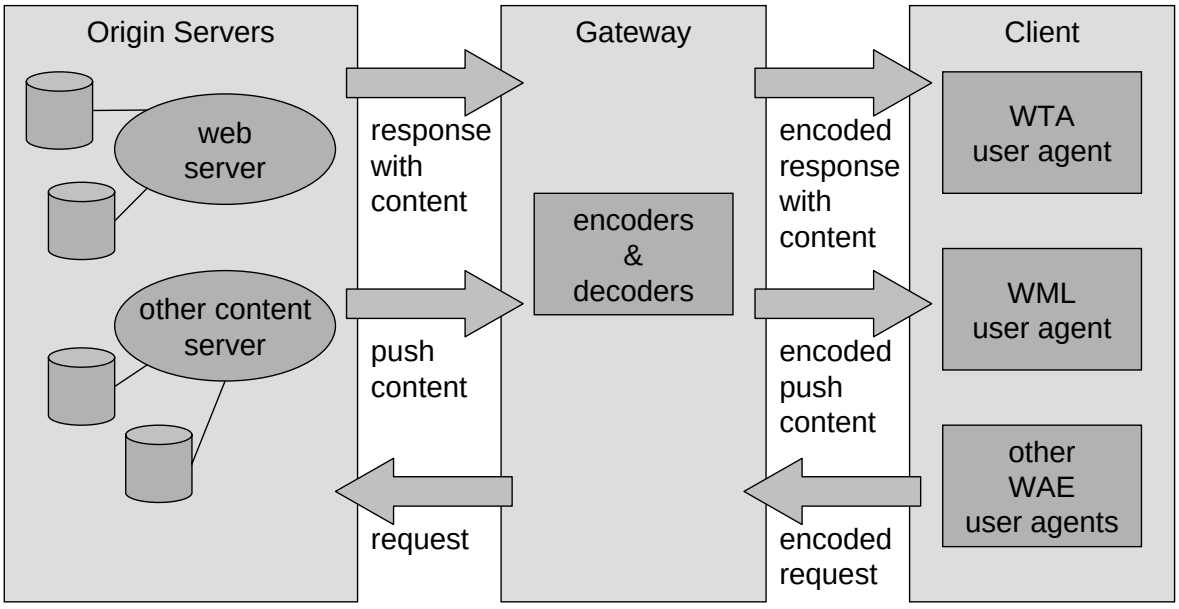
Requirements

- device and network independent, international support
- manufacturers can determine look-and-feel, user interface
- considerations of slow links, limited memory, low computing power, small display, simple user interface (compared to desktop computers)

Components

- architecture: application model, browser, gateway, server
- WML: XML-Syntax, based on card stacks, variables, ...
- WMLScript: procedural, loops, conditions, ... (similar to JavaScript)
- WTA: telephone services, such as call control, text messages, phone book, ... (accessible from WML/WMLScript)
- content formats: vCard, vCalendar, Wireless Bitmap, WML, ...

WAE logical model



Wireless Markup Language (WML)

WML follows deck and card metaphor

- WML document consists of many cards, cards are grouped to decks
- a deck is similar to an HTML page, unit of content transmission
- WML describes only intent of interaction in an abstract manner
- presentation depends on device capabilities

Features

- text and images
- user interaction
- navigation
- context management

WML – example I

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
            "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card id="card_one" title="simple example">
        <do type="accept">
            <go href="#card_two"/>
        </do>
        <p>
        This is a simple first card!
        <br/>
        On the next one you can choose ...
        </p>
    </card>
```

WML – example II

```
<card id="card_two" title="Pizza selection">
        <do type="accept" label="cont">
            <go href="#card_three"/>
        </do>
        <p>
        ... your favorite pizza!
        <select value="Mar" name="PIZZA">
            <option value="Mar">Margherita</option>
            <option value="Fun">Funghi</option>
            <option value="Vul">Vulcano</option>
        </select>
        </p>
    </card>
    <card id="card_three" title="Your Pizza!">
        <p>
        Your personal pizza parameter is <b>$(PIZZA)</b>!
        </p>
    </card>
</wml>
```

WMLScript

Complement to WML

Provides general scripting capabilities

Features

- validity check of user input
  - check input before sent to server
- access to device facilities
  - hardware and software (phone call, address book etc.)
- local user interaction
  - interaction without round-trip delay
- extensions to the device software
  - configure device, download new functionality after deployment

WMLScript - example

```
function pizza_test(pizza_type) {
   var taste = "unknown";
   if (pizza_type = "Margherita") {
        taste = "well... ";
   }
   else {
        if (pizza_type = "Vulcano") {
               taste = "quite hot";
        };
   };
   return taste;
};
```

Wireless Telephony Application (WTA)

Collection of telephony specific extensions

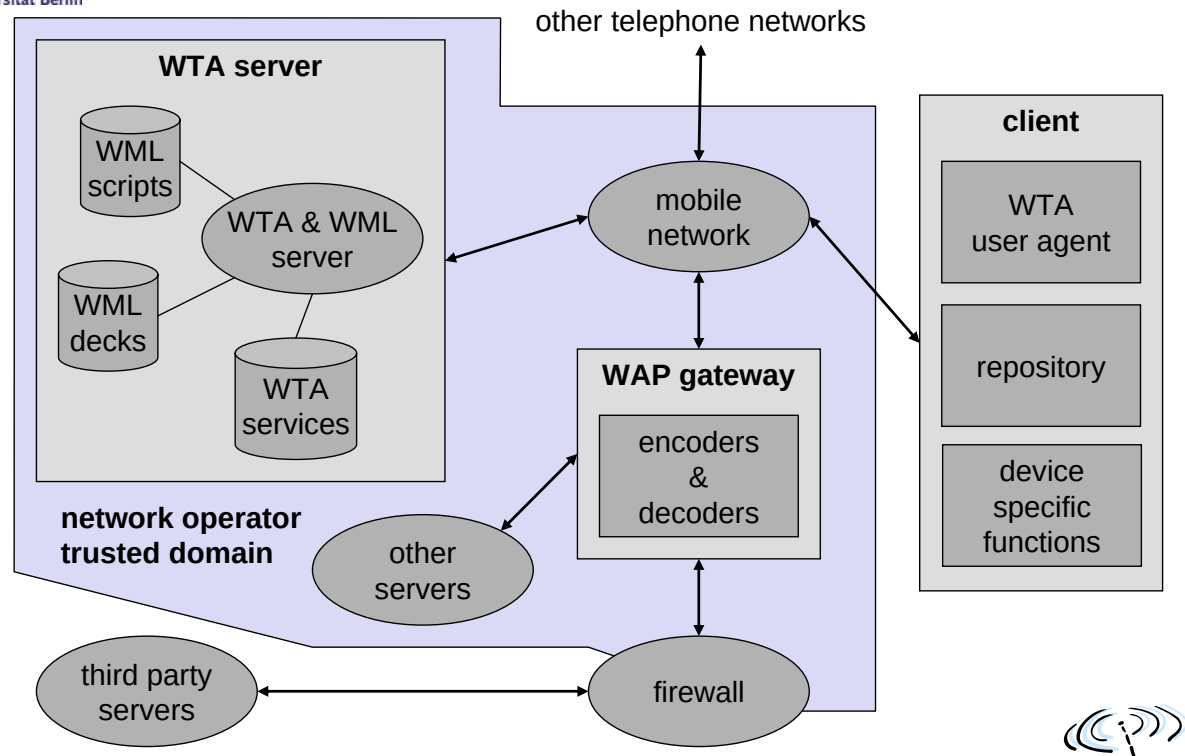
Extension of basic WAE application model

- content push
  - server can push content to the client
  - client may now be able to handle unknown events
- handling of network events
  - table indicating how to react on certain events from the network
- access to telephony functions
  - any application on the client may access telephony functions

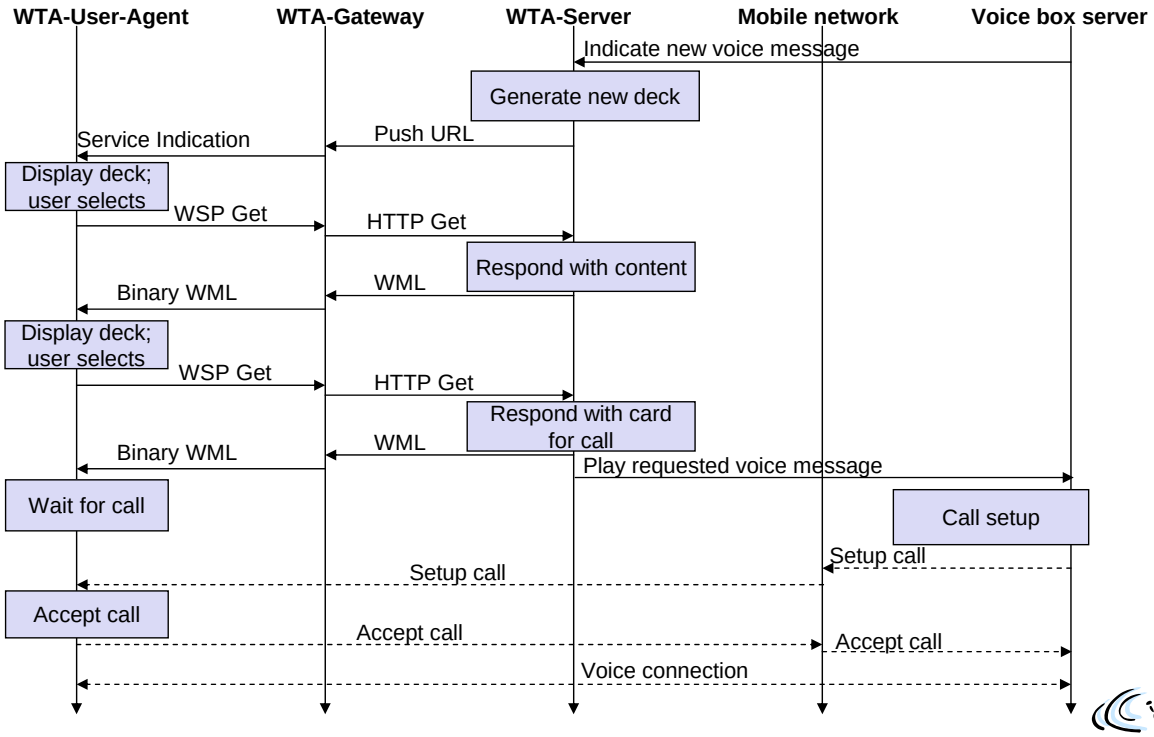
Example

- calling a number (WML)
  `wtai://wp/mc;07216086415`
- calling a number (WMLScript)
  `WTAPublic.makeCall("07216086415");`

WTA logical architecture



Voice box example



WTAI - example with WML only

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
              "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card id="card_one" title="Tele voting">
        <do type="accept">
            <go href="#card_two"/>
        </do>
        <p> Please choose your candidate! </p>
    </card>
    <card id="card_two" title="Your selection">
        <do type="accept">
            <go href="wtai://wp/mc;$dialno"/>
        </do>
        <p> Your selection:
        <select name="dialno">
            <option value="01376685">Mickey</option>
            <option value="01376686">Donald</option>
            <option value="01376687">Pluto</option>
        </select>
        </p>
    </card>
</wml>
```

WTAI - example with WML and WMLScript I

```
function voteCall(Nr) {
   var j = WTACallControl.setup(Nr,1);
   if (j>=0) {
        WMLBrowser.setVar("Message", "Called");
        WMLBrowser.setVar("No", Nr);
   }
   else {
        WMLBrowser.setVar("Message", "Error!");
        WMLBrowser.setVar("No", j);
   }
   WMLBrowser.go("showResult");
}
```

WTAI - example with WML and WMLScript II

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
              "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card id="card_one" title="Tele voting">
        <do type="accept"> <go href="#card_two"/> </do>
        <p> Please choose your candidate! </p>
    </card>
    <card id="card_two" title="Your selection">
        <do type="accept">
            <go href="/myscripts#voteCall($dialno)"/> </do>
        <p> Your selection:
        <select name="dialno">
            <option value="01376685">Mickey</option>
            <option value="01376686">Donald</option>
            <option value="01376687">Pluto</option>
        </select> </p>
    </card>
    <card id="showResult" title="Result">
        <p> Status: $Message $No </p>
    </card>
</wml>
```

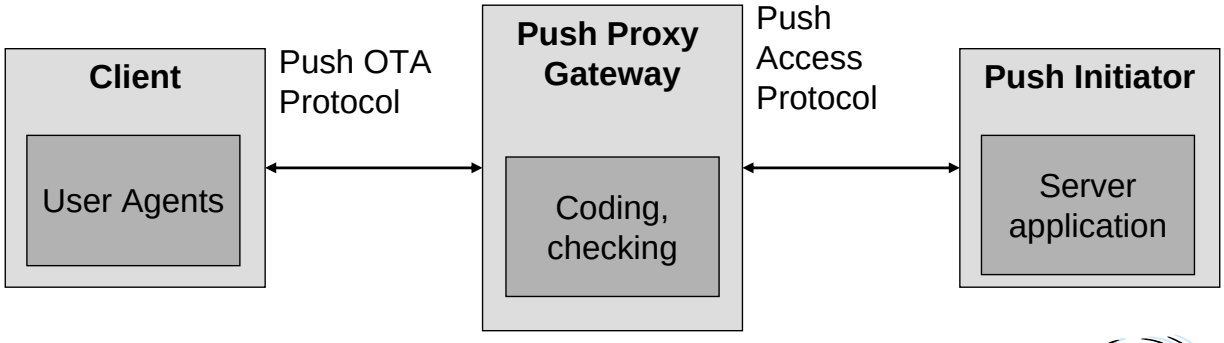
WAP push architecture with proxy gateway

Push Access Protocol

- Content transmission between server and PPG
- First version uses HTTP

Push OTA (Over The Air) Protocol

- Simple, optimized
- Mapped onto WSP



Push/Pull services in WAP I

Service Indication

- Service announcement using a pushed short message
- Service usage via a pull
- Service identification via a URI

```
<?xml version="1.0"?>
  <!DOCTYPE si PUBLIC "-//WAPFORUM//DTD SI 1.0//EN"
    "http://www.wapforum.org/DTD/si.dtd">
<si>
  <indication href="http://www.piiiizza4u.de/offer/salad.wml"
     created="2002-10-30T17:45:32Z"
     si-expires="2000-10-30T17:50:31Z">
     Salad special: The 5 minute offer
  </indication>
</si>
```

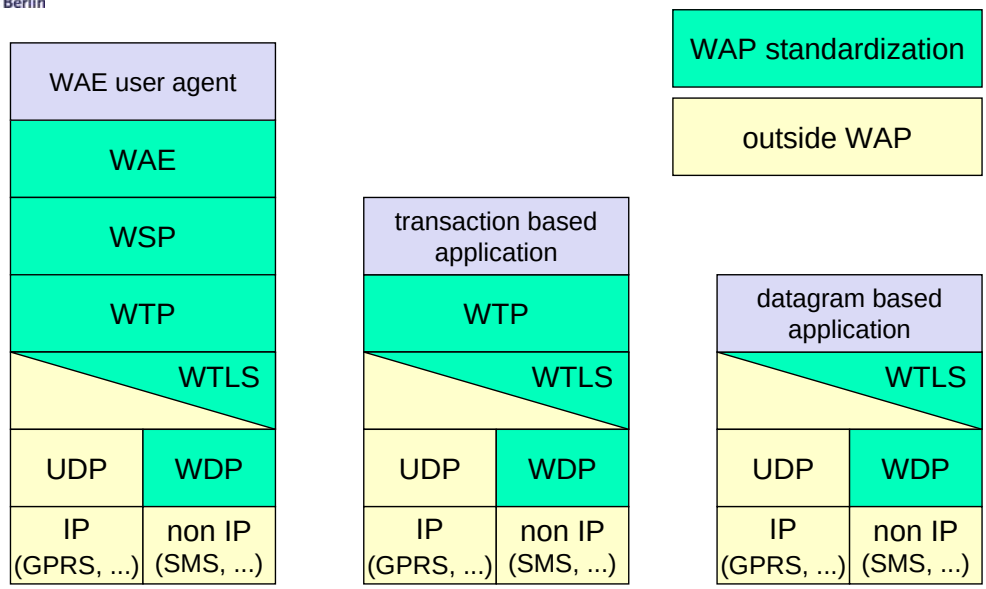
Push/Pull services in WAP II

Service Loading

- short message pushed to a client containing a URI
- User agent decides whether to use the URI via a pull
- Transparent for users, always looks like a push

```
<?xml version="1.0"?>
  <!DOCTYPE sl PUBLIC "-//WAPFORUM//DTD SL 1.0//EN"
    "http://www.wapforum.org/DTD/sl.dtd">
<sl
  href="http://www.piiiizza4u.de/offer/salad.wml">
</sl>
```

Examples for WAP protocol stacks (WAP 1.x)



1.

typical WAP application with complete protocol stack

2.

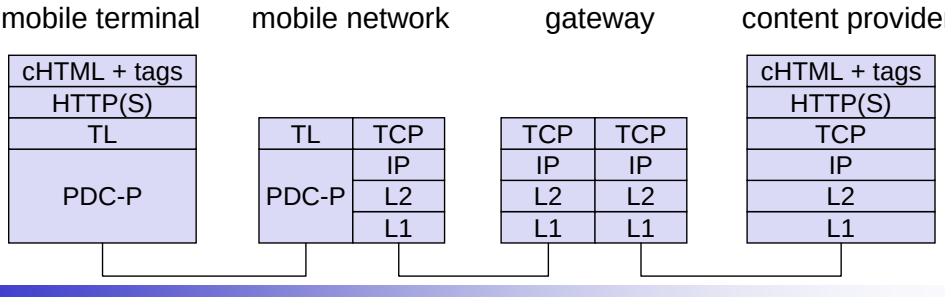
3.

pure data application with/without additional security

i-mode – first of all a business model!

Access to Internet services in Japan provided by NTT DoCoMo

- Services
  - Email, short messages, web, picture exchange, horoscope, ...
- Big success – more than 30 million users
  - Many use i-mode as PC replacement
  - For many this is the first Internet contact
  - Very simple to use, convenient
- Technology
  - 9.6 kbit/s (enhancements with 28.8 kbit/s), packet oriented (PDC-P)
  - Compact HTML plus proprietary tags, special transport layer (Stop/go, ARQ, push, connection oriented)



Email example: i-mode push with SMS

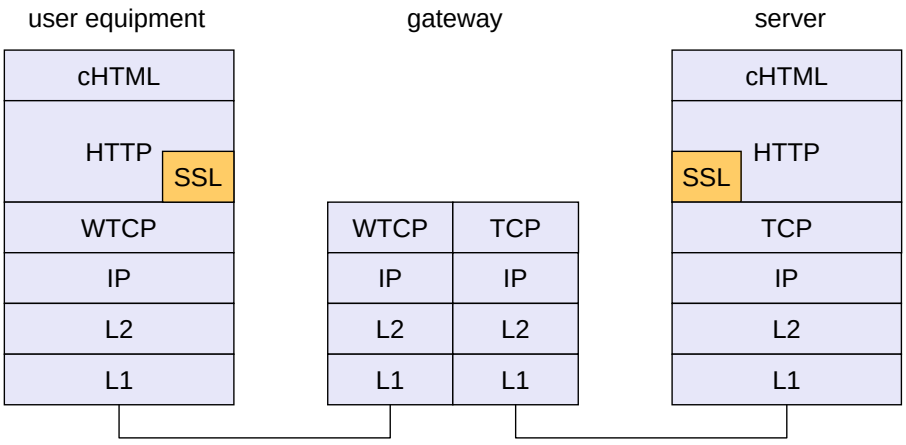
application
WSP
WTP
WDP
SMS

Operator sends an SMS containing a push message if a new email has arrived. If the user wants to read the email, an HTTP get follows with the email as response.

Popular misconception:
WAP was a failure, i-mode is different and a success – wrong from a technology point of view, right from a business point of view…

i-mode as a **business model**:
- content providers get >80% of the revenue.
- independent of technology (GSM/GPRS in Europe, PDC-P in Japan – but also UMTS!)

i-mode protocol stack based on WAP 2.0

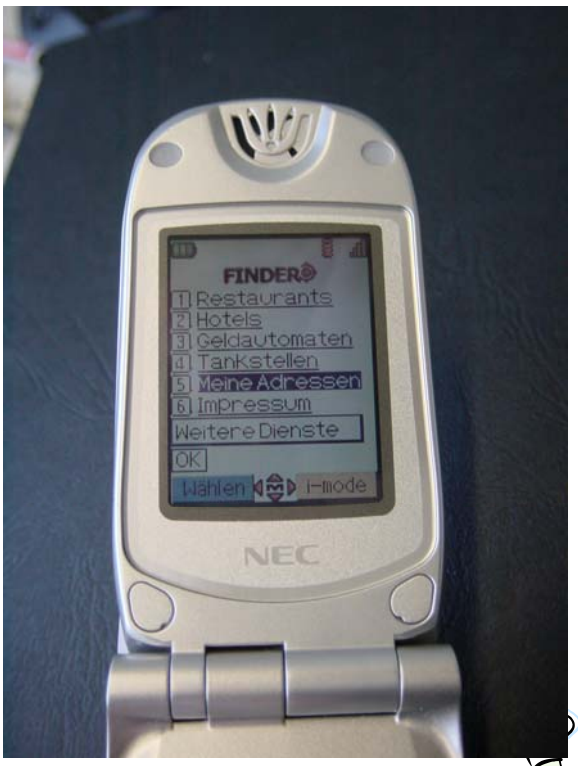


i-mode can use WAP 2.0/Internet protocols (example: i-mode in Germany over GSM/GPRS)

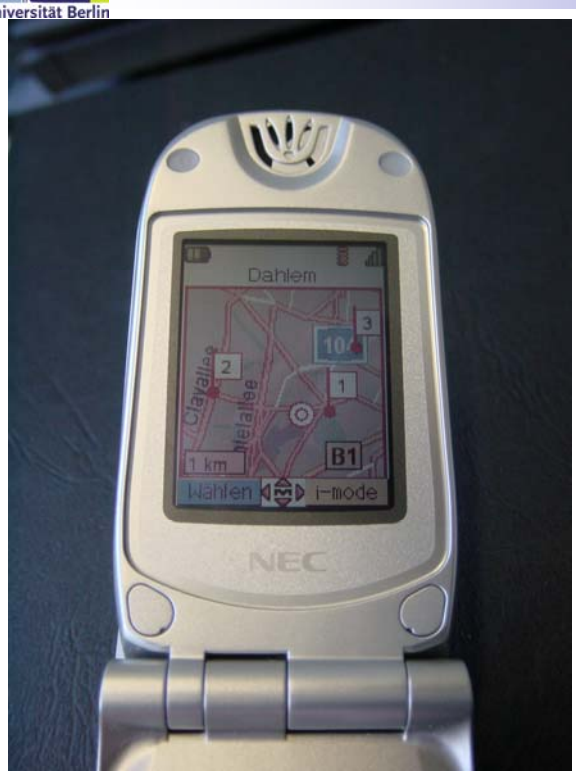
i-mode – technical requirements

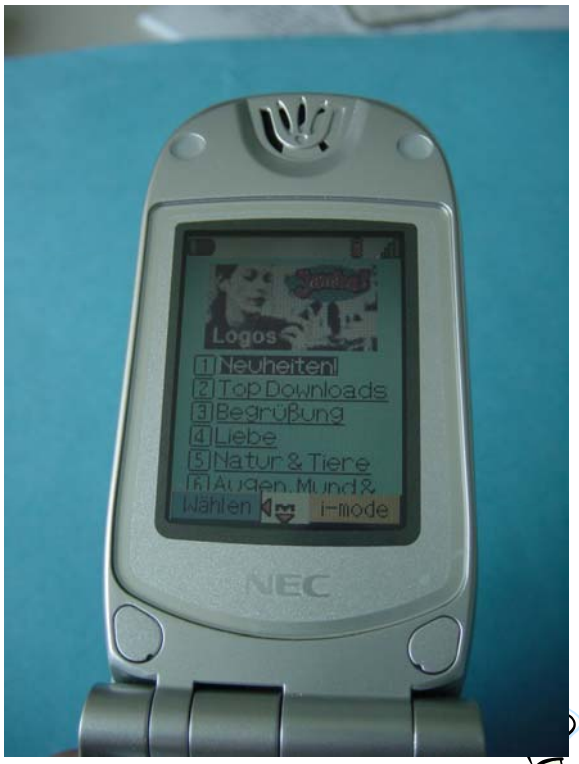
Functions	Descriptions	Status	Requirement
WEB Access	Portal Site / Internet Access	M	i-mode HTML (cHTML+tags)
E-mail	Internet e-mail and inter-terminal email	M	HTTP 1.1
Security	End-End security	O	SSL (Version 2, 3), TLS 1
Java	Java application made available	O	Compatible i-mode JAVA
Ringing tone download	Ringing melody download	M	SMF based
Image download	Stand-by screen download	M	GIF (O: JPEG)
Voice call notification during i-mode session	Voice termination notified and responded during i-mode communications	M	3GPP standard system
Content charge billing	Per content charge billed to user	M	Specifications depend on each operator's billing system
Third party payment collection	Content charge collection on behalf of Content Provider	M	Specifications depend on each operator's billing system
Reverse billing	Packet usage charges can be billed to third party	O	Specifications depend on each operator's billing system
Subscriber ID transmission	Hashed subscriber ID from the operator's portal to the CP transmission on each content access	M	The ID generation algorithm should be determined by each operator and has to be secret
Number of characters per e-mail	Number of characters (byte) per e-mail	M	To be defined by operators (e.g. 500 byte, 1K byte, 10K byte)
Character code set supported	Character code set supported by browser and used to develop content	M	To be defined by operators
User Agent	Browser specifications to be notified	M	HTTP 1.1
i-mode button	Dedicated button	O	Hard or soft key

i-mode examples I



i-mode examples II





i-mode examples III

WAP 2.0 (July 2001)

New for developers

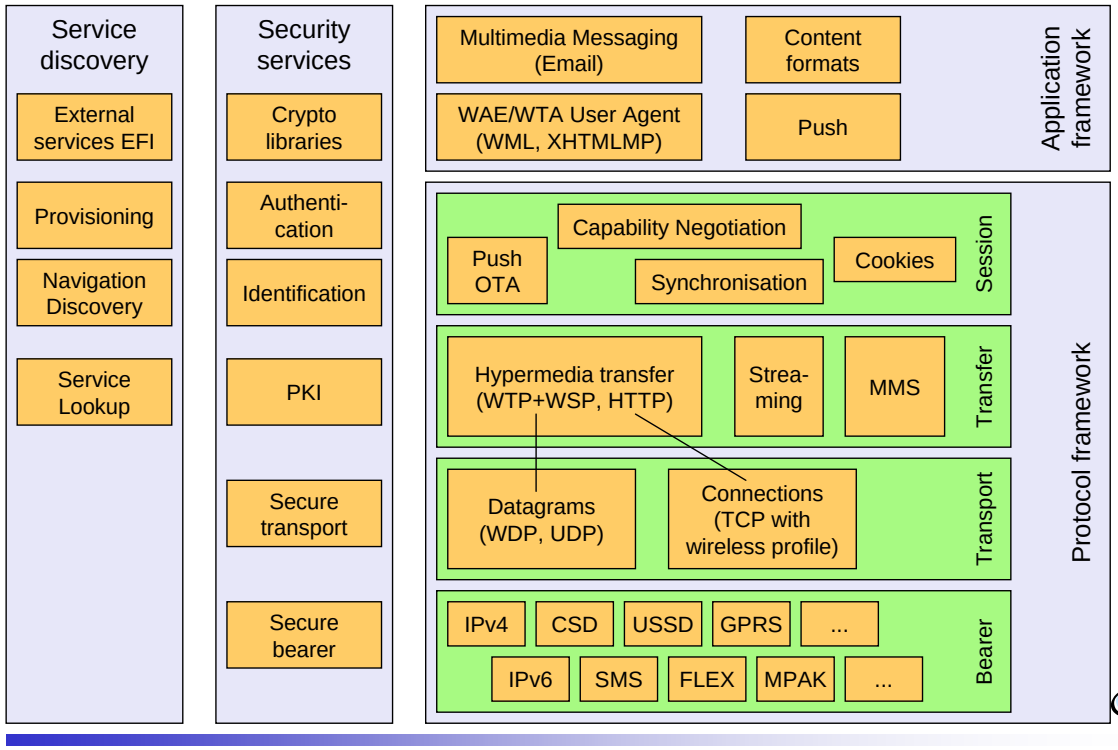
- XHTML
- TCP with „Wireless Profile“
- HTTP

New applications

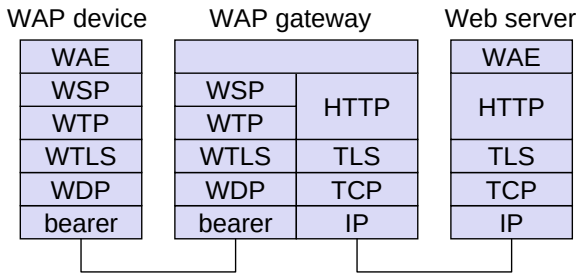
- Color graphics
- Animation
- Large file download
- Location based services
- Synchronization with PIMs
- Pop-up/context sensitive menus

Goal: integration of WWW, Internet, WAP, i-mode

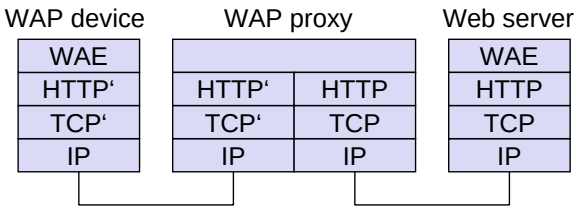
WAP 2.0 architecture



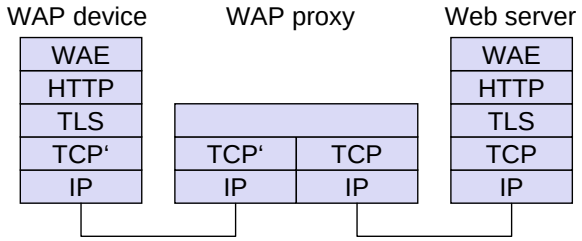
WAP 2.0 example protocol stacks



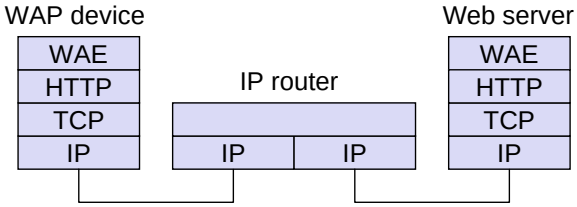
WAP 1.x Server/Gateway/Client



WAP HTTP Proxy with profiled TCP and HTTP



WAP Proxy with TLS tunneling



WAP direct access

Java 2 Platform Micro Edition

„Java-Boom expected“ (?)

- Desktop: over 90% standard PC architecture, Intel x86 compatible, typically MS Windows systems
- Do really many people care about platform independent applications?

BUT: Heterogeneous, “small“ devices

- Internet appliances, cellular phones, embedded control, car radios, ...
- Technical necessities (temperature range, form factor, power consumption, …) and economic reasons result in different hardware

J2ME

- Provides a uniform platform
- Restricted functionality compared to standard java platform (JVM)

Applications of J2ME

Example cellular phones

- NTT DoCoMo introduced iαppli
- Applications on PDA, mobile phone, ...
- Game download, multimedia applications, encryption, system updates
- Load additional functionality with a push on a button (and pay for it)!

Embedded control

- Household devices, vehicles, surveillance systems, device control
- System update is an important factor

αPPLI

Characteristics and architecture

Java Virtual Machine

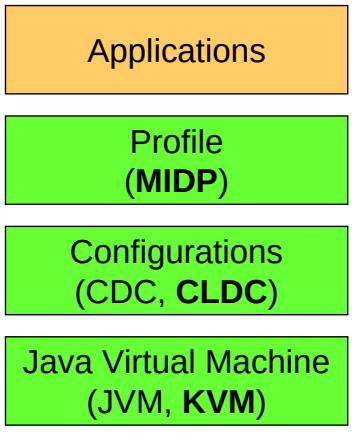
- Virtual Hardware (Processor)
- KVM (K Virtual Machine)
  - Min. 128 kByte, typ. 256 kByte
  - Optimized for low performance devices
  - Might be a co-processor

Configurations

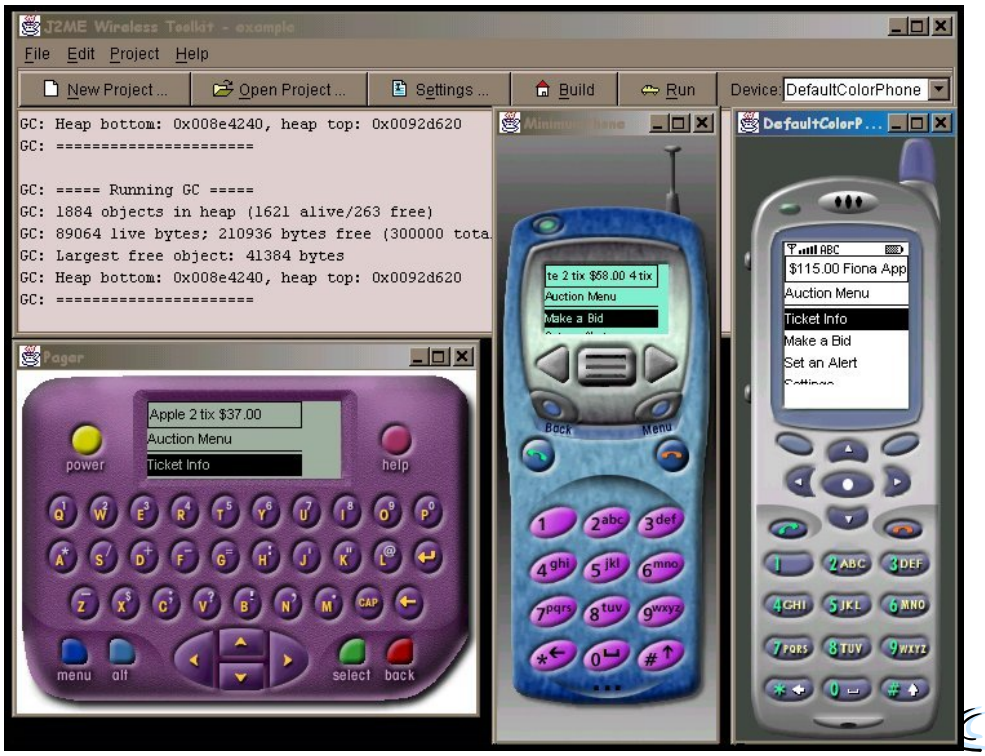
- Subset of standard Java libraries depending technical hardware parameters (memory, CPU)
- CLDC (Connected Limited Device Configuration)
  - Basic libraries, input/output, security – describes Java support for mobile devices

Profiles

- Interoperability of heterogeneous devices belonging to the same category
- MIDP (Mobile Information Device Profile)
  - Defines interfaces for GUIs, HTTP, application support, …



Hardware independent development



Summary J2ME

Idea is more than WAP 1.x or i-mode

- Full applications on mobile phones, not only a browser
- Includes system updates, end-to-end encryption

Platform independent via virtualization

- As long as certain common interfaces are used
- Not valid for hardware specific functions

Limited functionality compared to JVM

- Thus, maybe an intermediate solution only – until embedded systems, mobile phones are as powerful as today's desktop systems