

Wireless Application Protocol

Sridhar Iyer

KR School of Information Technology

IIT Bombay

<http://www.it.iitb.ernet.in/~sri>

Jan 2001

Outline

- Mobile applications
- How are mobile/wireless environments different?
- What is WAP?
- WAP Architecture
- WAE (WML/WMLScript)
- WTA Framework
- WAP Push Services
- WAP Protocol Stack
- Hype v/s Reality
- References and Resources

Mobile Applications - 1

- **Vehicles**
 - transmission of news, road condition etc
 - ad-hoc network with near vehicles to prevent accidents

- **Emergencies**
 - early transmission of patient data to the hospital
 - ad-hoc network in case of earthquakes, cyclones
 - military ...

- **Traveling salesmen**
 - direct access to central customer files
 - consistent databases for all agents
 - mobile office

Mobile Applications - 2

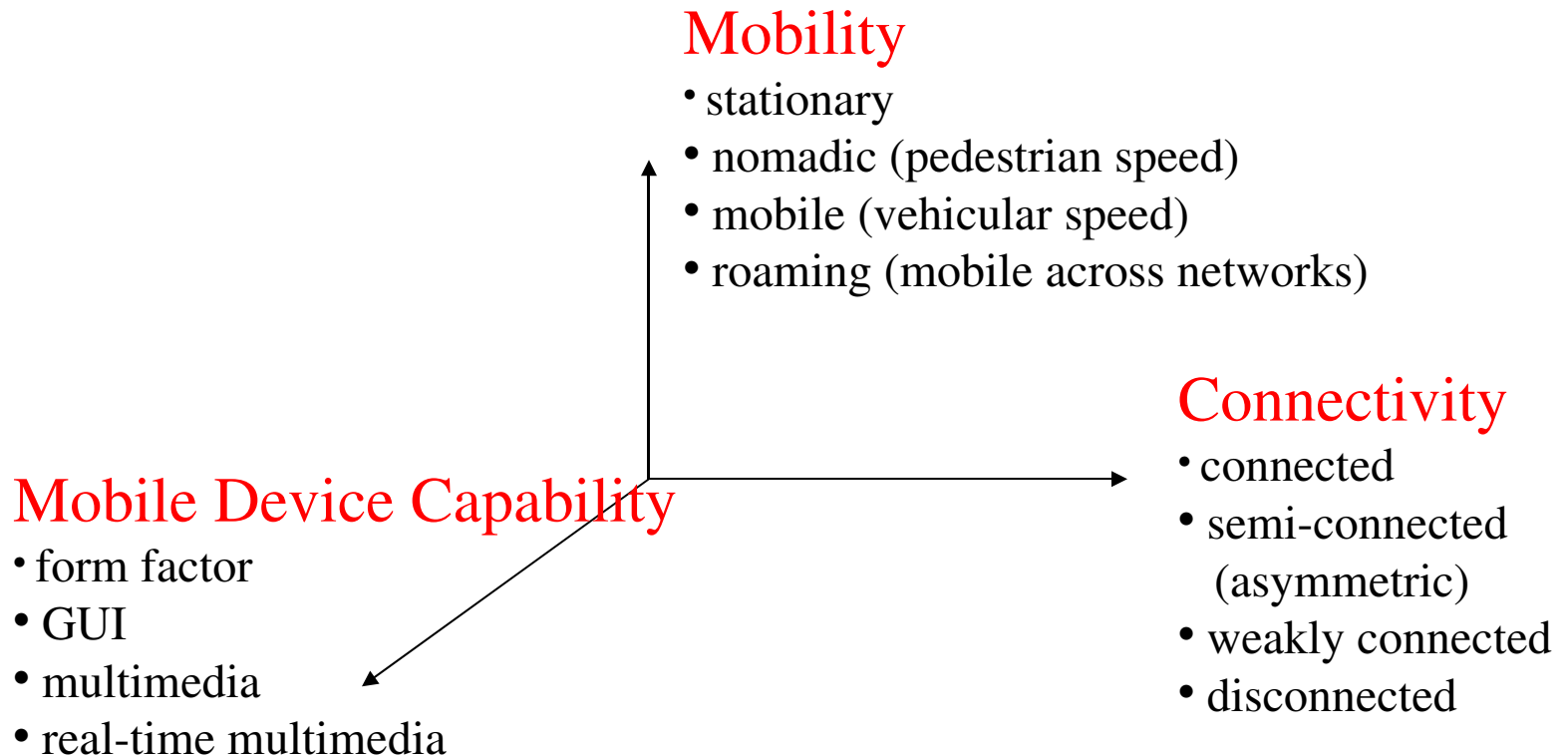
- **Web access**
 - outdoor Internet access
 - intelligent travel guide with up-to-date location dependent information

- **Information services**
 - push: stock quotes; pull: nearest cash ATM

- **Disconnected operations**
 - file-system caching for off-line work
 - mobile agents, e.g., shopping

- **Entertainment**
 - games, etc

Variability of the Mobile Environment



World Wide Web and Mobility

HTTP/HTML have not been designed for mobile applications/devices

■ HTTP 1.0 characteristics

- designed for large bandwidth, low delay
- stateless, client/server, request/response communication
- connection oriented, one connection per request
- TCP 3-way handshake, DNS lookup overheads
- big protocol headers, uncompressed content transfer
- primitive caching (often disabled, dynamic objects)
- security problems (using SSL/TLS with proxies)

■ HTML characteristics

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

System Support for Mobile WWW

- **Enhanced browsers**
 - client-aware support for mobility
- **Proxies**
 - Client proxy: pre-fetching, caching, off-line use
 - Network proxy: adaptive content transformation for connections
 - Client and network proxy
- **Enhanced servers**
 - server-aware support for mobility
 - serve the content in multiple ways, depending on client capabilities
- **New protocols/languages**
 - WAP/WML

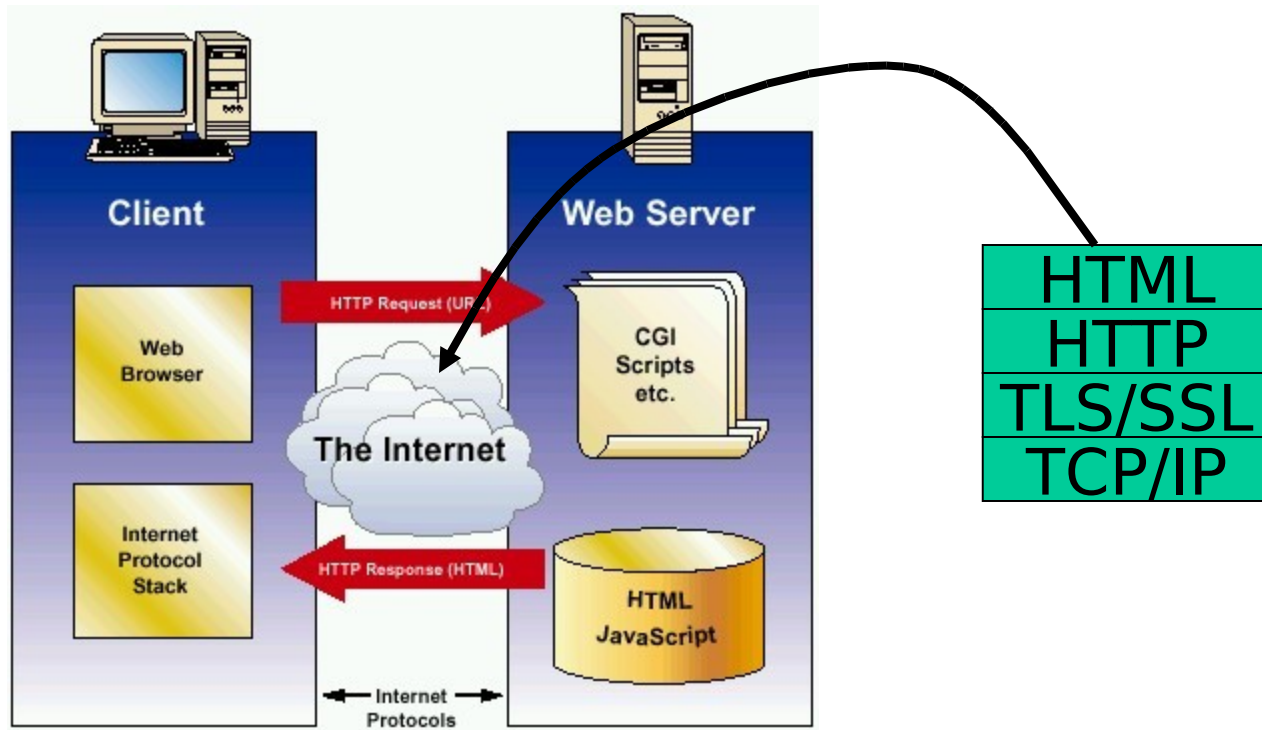
Wireless Application Protocol (WAP)

- Empowers mobile users with wireless devices to easily access and interact with information and services.
- A “standard” created by wireless and Internet companies to enable Internet access from a cellular phone
- wapforum.org:
 - co-founded by Ericsson, Motorola, Nokia, Phone.com
 - 450 members in 2000, comprise of Handset manufacturers, Wireless service providers, ISPs, Software companies in the wireless industry
 - Goals
 - deliver Internet services to mobile devices
 - enable applications to scale across a variety of transport options and device types
 - independence from wireless network standards
 - GSM, CDMA IS-95, TDMA IS-136, 3G systems (UMTS, W-CDMA)

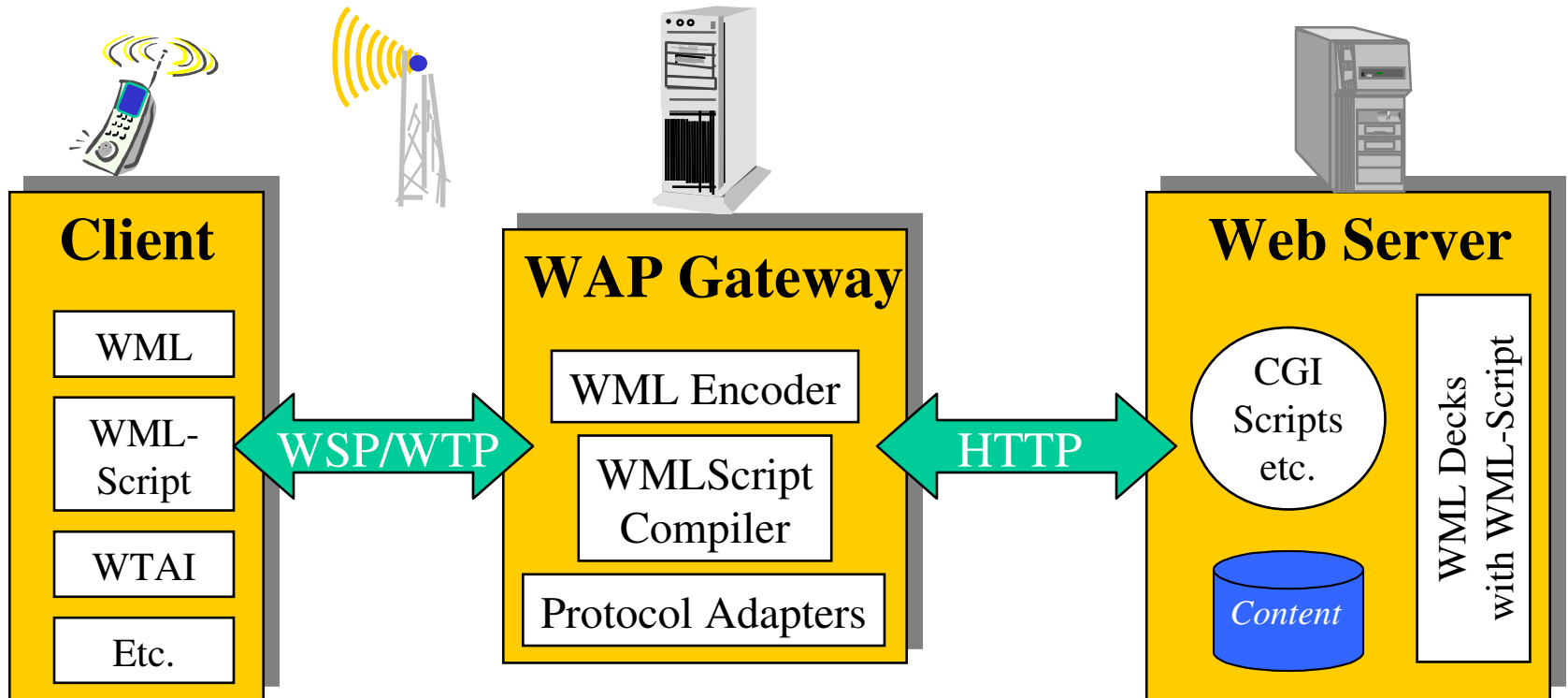
WAP: Main Features

- **Browser**
 - “Micro browser”, similar to existing web browsers
- **Markup language**
 - Similar to HTML, adapted to mobile devices
- **Script language**
 - Similar to Javascript, adapted to mobile devices
- **Gateway**
 - Transition from wireless to wired world
- **Server**
 - “Wap/Origin server”, similar to existing web servers
- **Protocol layers**
 - Transport layer, security layer, session layer etc.
- **Telephony application interface**
 - Access to telephony functions

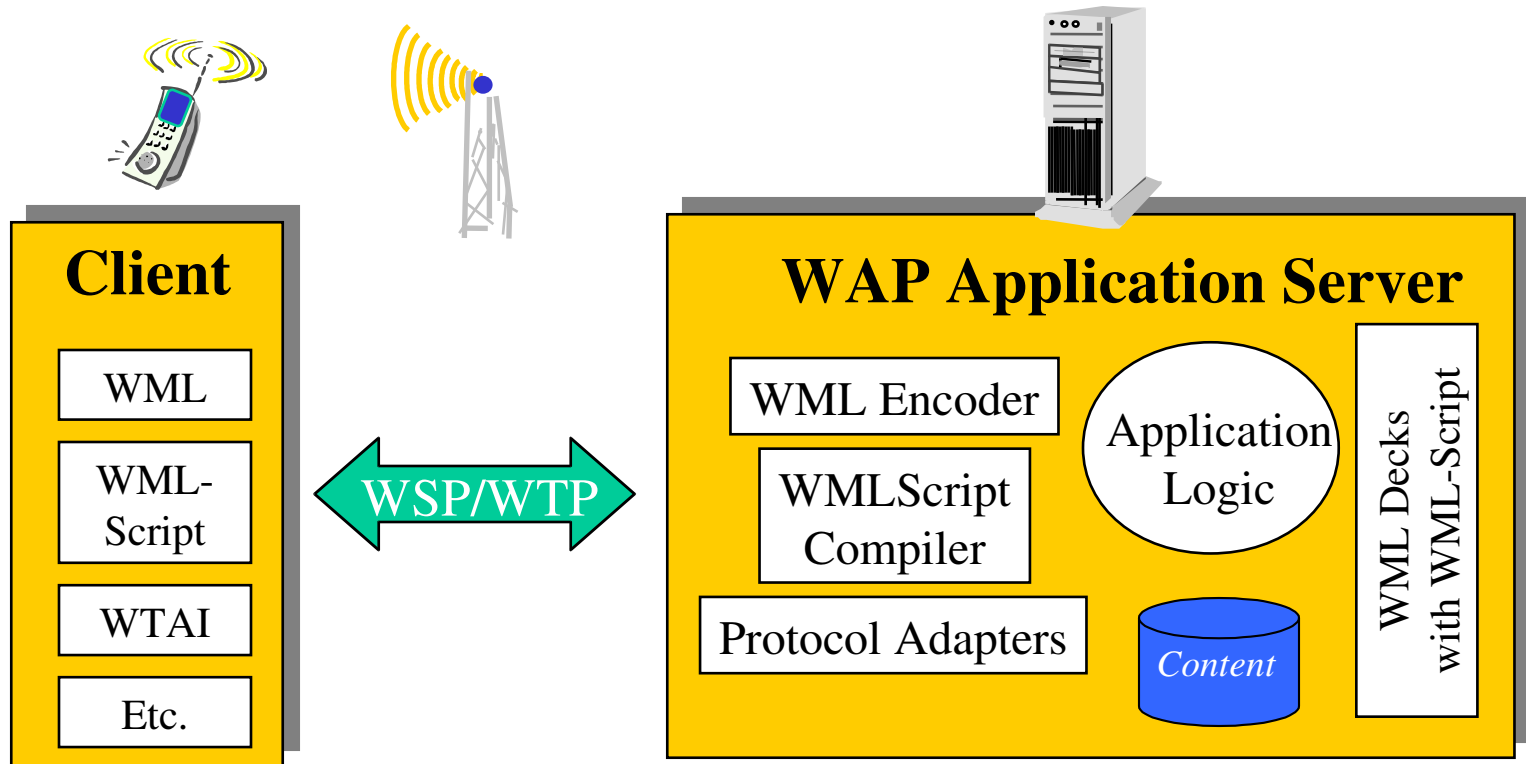
Internet Model



WAP Architecture

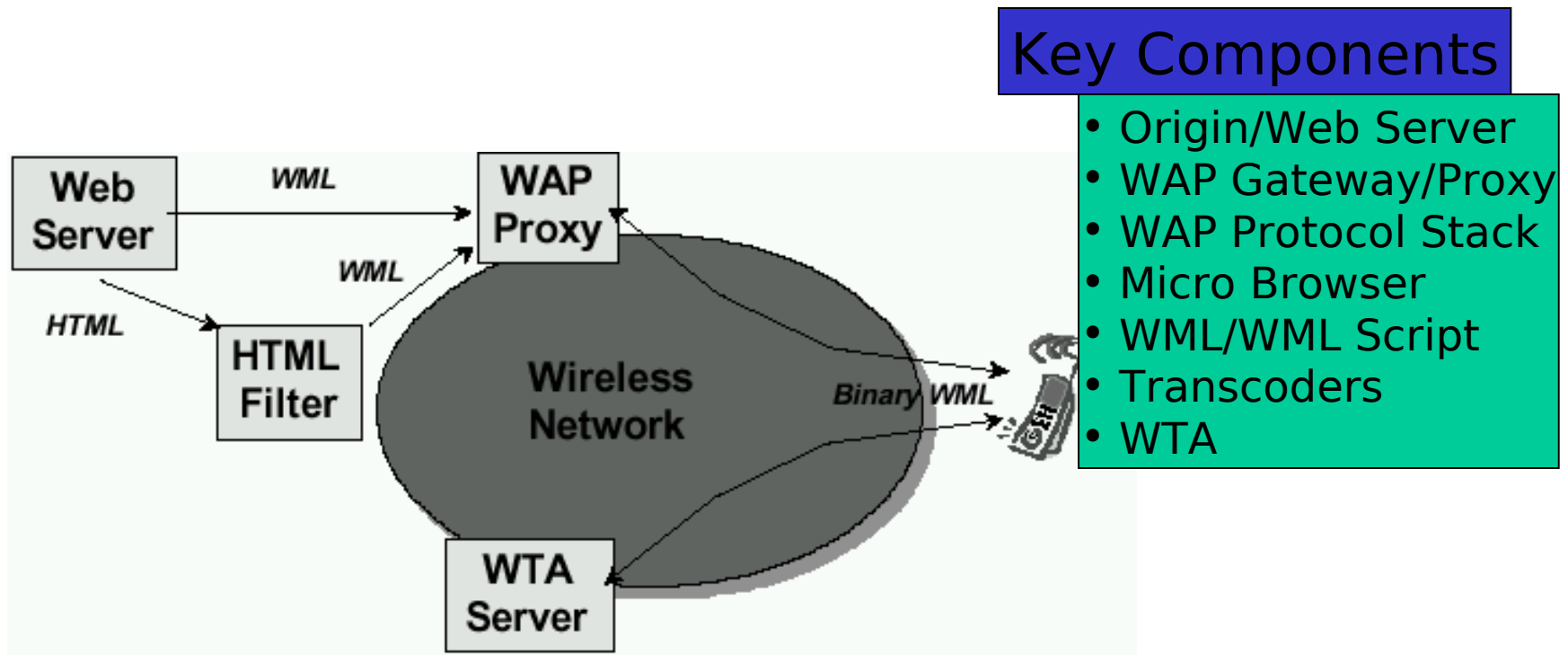


WAP Application Server

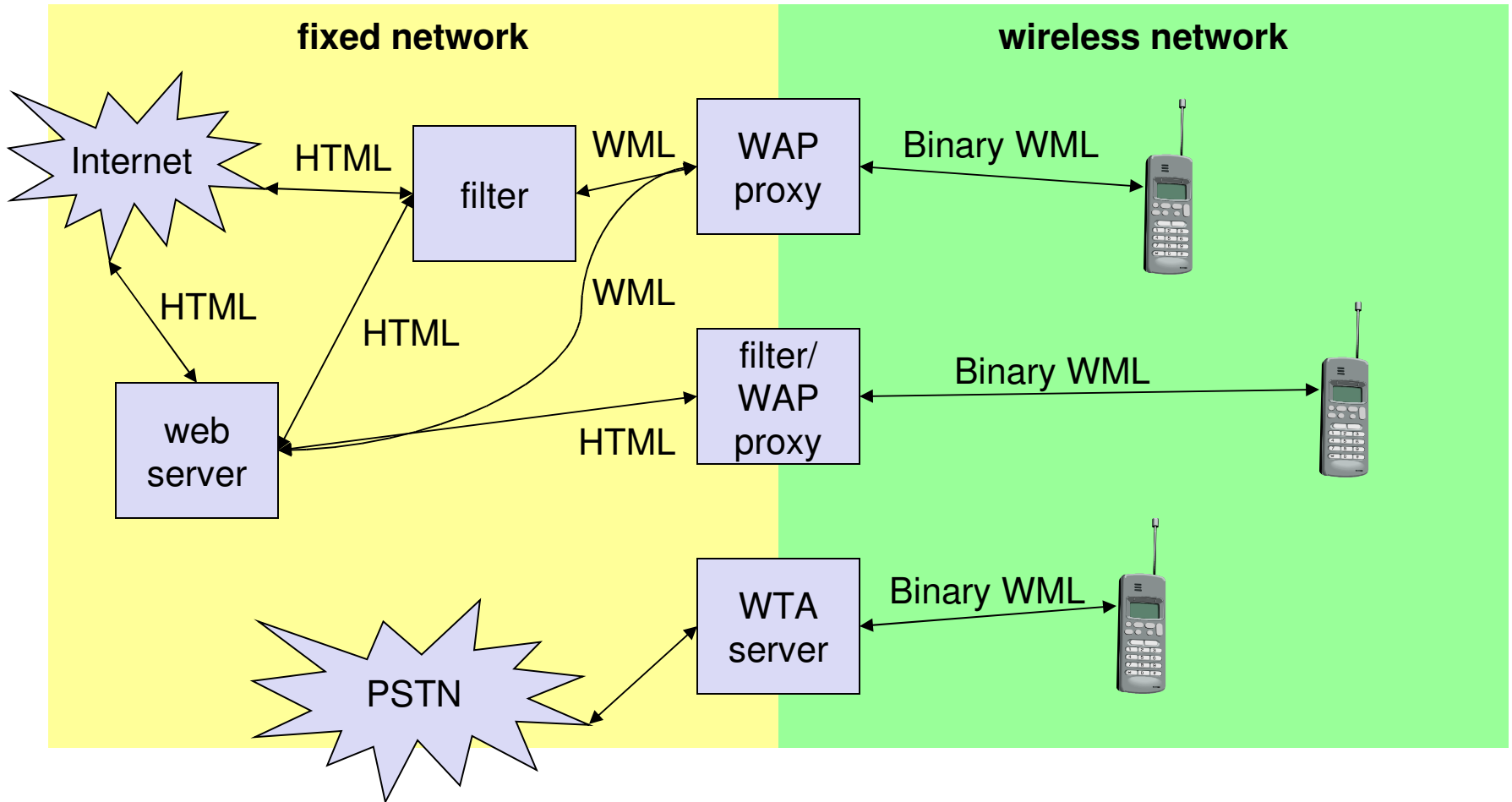


WAP Architecture

- Another look



WAP: Network Elements



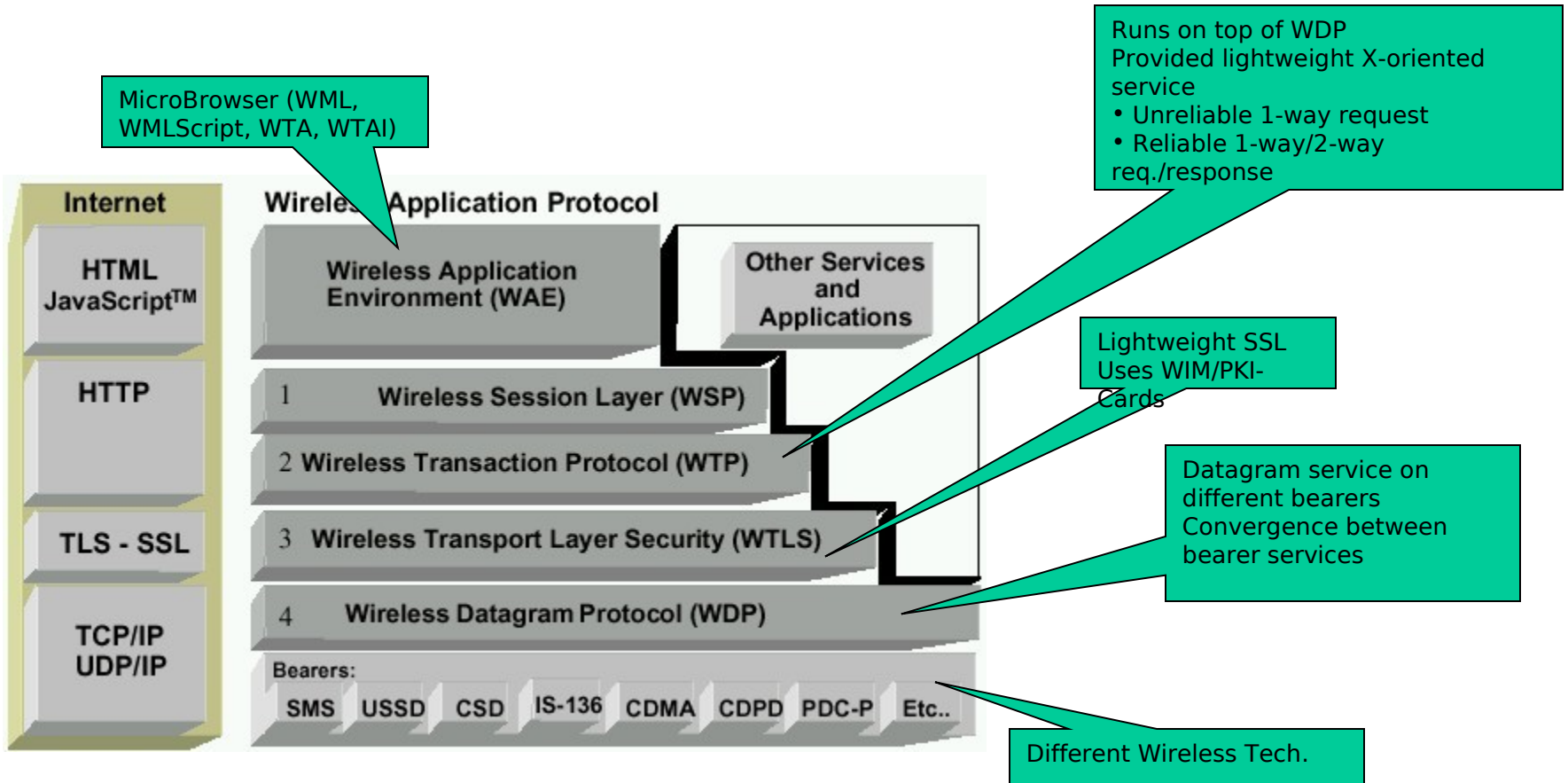
Binary WML: binary file format for clients

WAP Specifies

- **Wireless Application Environment**
 - WML Microbrowser
 - WMLScript Virtual Machine
 - WMLScript Standard Library
 - Wireless Telephony Application Interface (WTAI)
 - WAP content types

- **Wireless Protocol Stack**
 - Wireless Session Protocol (WSP)
 - Wireless Transport Layer Security (WTLS)
 - Wireless Transaction Protocol (WTP)
 - Wireless Datagram Protocol (WDP)
 - Wireless network interface definitions

WAP Stack



WAP Stack

- **WAE (Wireless Application Environment):**
 - Architecture: application model, browser, gateway, server
 - WML: XML-Syntax, based on card stacks, variables, ...
 - WTA: telephone services, such as call control, phone book etc.
- **WSP (Wireless Session Protocol):**
 - Provides HTTP 1.1 functionality
 - Supports session management, security, etc.
- **WTP (Wireless Transaction Protocol):**
 - Provides reliable message transfer mechanisms
 - Based on ideas from TCP/RPC
- **WTLS (Wireless Transport Layer Security):**
 - Provides data integrity, privacy, authentication functions
 - Based on ideas from TLS/SSL
- **WDP (Wireless Datagram Protocol):**
 - Provides transport layer functions
 - Based on ideas from UDP

Content encoding, optimized for low-bandwidth channels, simple devices

WHY WAP?

- Wireless networks and phones
 - have specific needs and requirements
 - not addressed by existing Internet technologies
- WAP
 - Enables any data transport
 - TCP/IP, UDP/IP, GUTS (IS-135/6), SMS, or USSD.
 - Optimizes the content and air-link protocols
 - Utilizes plain Web HTTP 1.1 servers
 - leverages existing development methodologies
 - utilizes standard Internet markup language technology (XML)
 - all WML content is accessed via HTTP 1.1 requests
 - WML UI components map well onto existing mobile phone user interfaces
 - no re-education of the end-users
 - leveraging market penetration of mobile devices
 - Several modular entities together form a fully compliant Internet entity

Why is HTTP/HTML not enough?

Big pipe - small pipe syndrome

Internet

HTTP/HTML

```
<HTML>
<HEAD>
<TITLE>NNN Interactive</TITLE>
<META HTTP-EQUIV="Refresh" CONTENT="1800,
URL=/index.html">
</HEAD>
<BODY BGCOLOR="#FFFFFF"
BACKGROUND="/images/9607/bgbar5.gif" LINK="#0A3990"
ALINK="#FF0000" VLINK="#FF0000" TEXT="000000"
ONLOAD="if(parent.frames.length!
=0)top.location='http://nnn.com';">
<A NAME="#top"></A>
<TABLE WIDTH=599 BORDER="0">
<TR ALIGN=LEFT>
<TD WIDTH=117 VALIGN=TOP ALIGN=LEFT>
```

```
<HTML>
<HEAD>
<TITLE
>NNN
Intera
ctive<
/TITLE
>
<META
HTTP-
EQUIV=
"Refre
sh"
CONTEN
T="180
0,
URL=/i
ndex.h
tml">
```

Wireless network

WAP

```
<WML>
<CARD>
<DO TYPE="ACCEPT">
<GO URL="/submit?Name=$N"/>
</DO>
Enter name:
<INPUT TYPE="TEXT" KEY="N"/>
</CARD>
</WML>
```

Content encoding

```
010011
010011
110110
010011
011011
011101
010010
011010
```

WAP: “Killer” Applications

- **Location-based services**
 - Real-time traffic reporting, Event/restaurant recommendation
- **Enterprise solutions**
 - Email access, Database access, “global” intranet access
 - Information updates “pushed” to WAP devices
- **Financial services**
 - Banking, Bill-paying, Stock trading, Funds transfers
- **Travel services**
 - Schedules and rescheduling, Reservations
- **Gaming and Entertainment**
 - Online, real-time, multi-player games
 - Downloadable horoscopes, cartoons, quotes, advice
- **M-Commerce**
 - Shopping on the go
 - Instant comparison shopping
 - Location-based special offers and sales

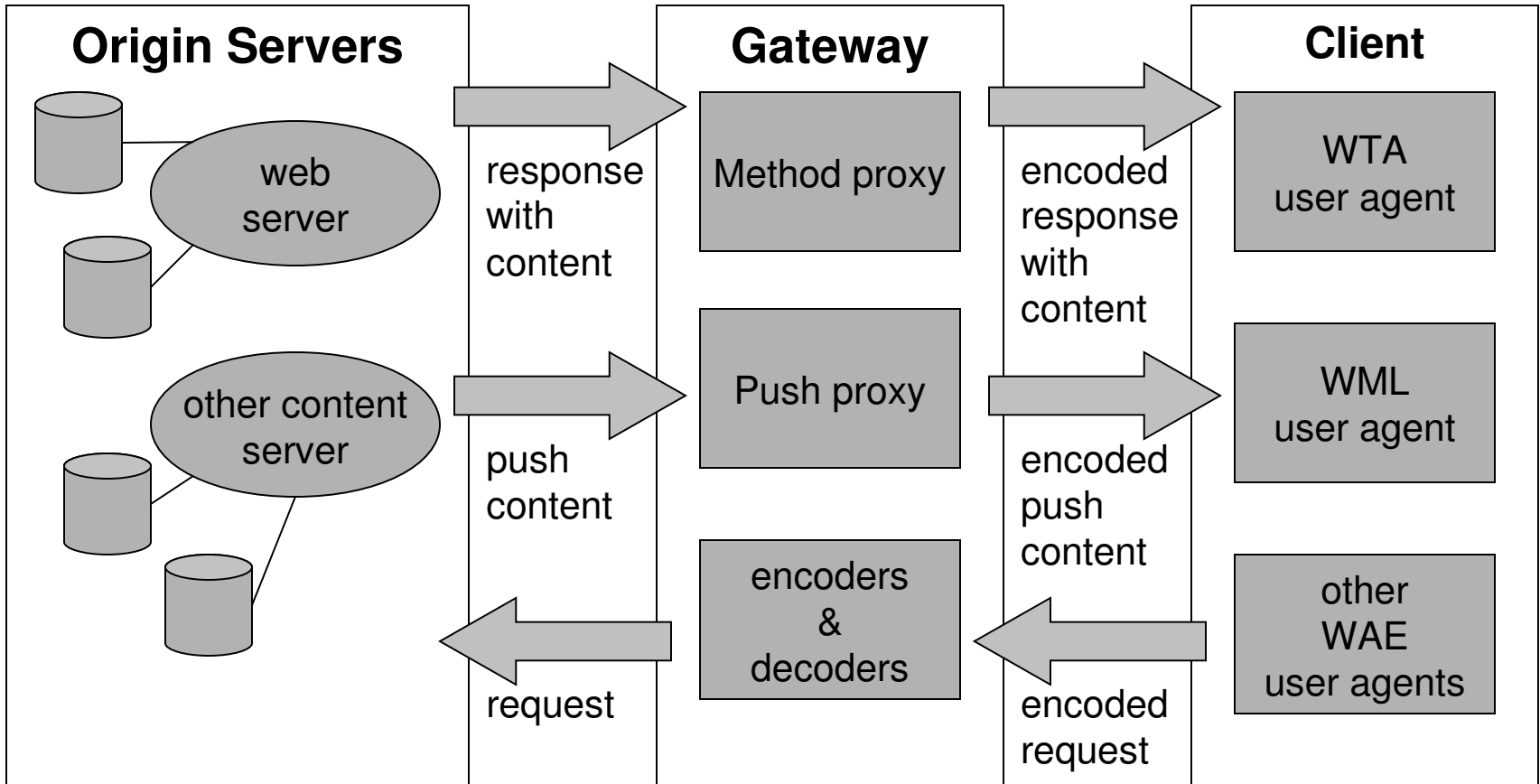
Wireless Application Environment (WAE)

- Goals
 - device and network independent application environment
 - for low-bandwidth, wireless devices
 - considerations of slow links, limited memory, low computing power, small display, simple user interface (compared to desktops)
 - integrated Internet/WWW programming model
 - high interoperability

WAE Components

- **Architecture**
 - Application model, Microbrowser, Gateway, Server
- **User Agents**
 - WML/WTA/Others
 - content formats: vCard, vCalendar, Wireless Bitmap, WML, ...
- **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)
- **Proxy (Method/Push)**

WAE: Logical Model



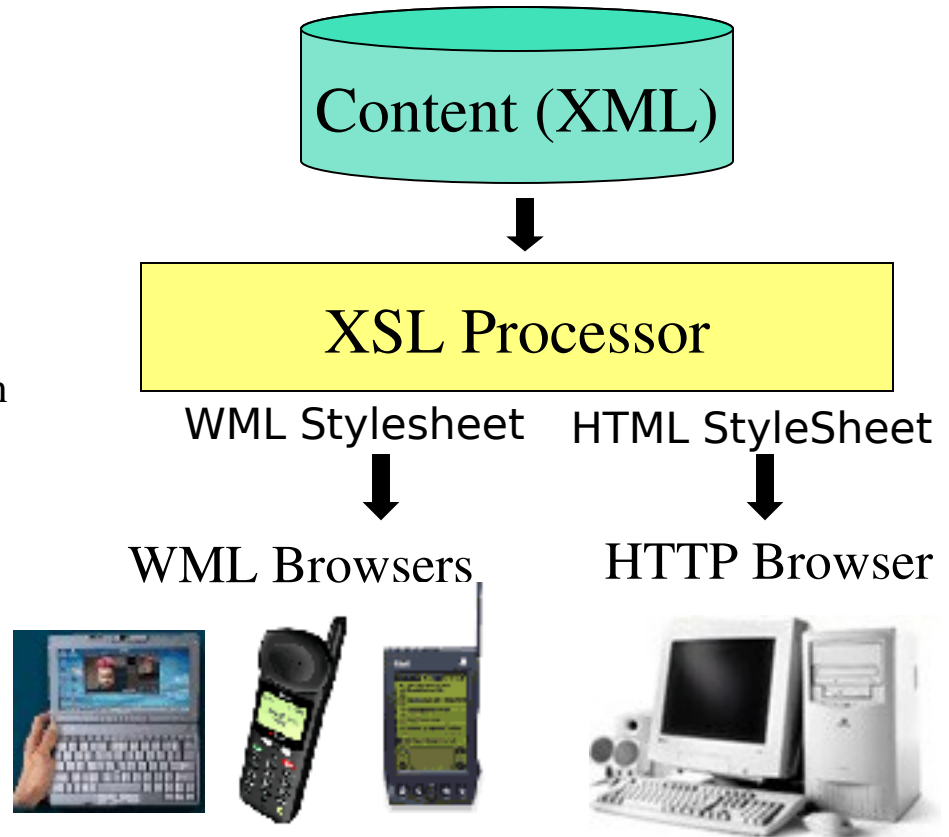
WAP Microbrowser



- Optimized for wireless devices
- Minimal RAM, ROM, Display, CPU and keys
- Provides consistent service UI across devices
- Provides Internet compatibility
- Enables wide array of available content and applications

WML: Wireless Markup Language

- Tag-based browsing language:
 - Screen management (text, images)
 - Data input (text, selection lists, etc.)
 - Hyperlinks & navigation support
- Takes into account limited display, navigation capabilities of devices
- XML-based language
 - describes only intent of interaction in an abstract manner
 - presentation depends upon device capabilities
- Cards and Decks
 - document consists of many cards
 - User interactions are split into cards
 - Explicit navigation between cards
 - cards are grouped to decks
 - deck is similar to HTML page, unit of content transmission
- Events, variables and state mgmt



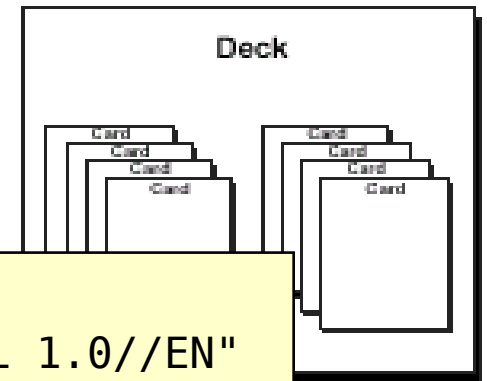
WML

- The basic unit is a **card**. Cards are grouped together into **Decks** Document ~ Deck (unit of transfer)
- All decks must contain
 - Document prologue
 - XML & document type declaration
 - <WML> element
 - Must contain one or more cards

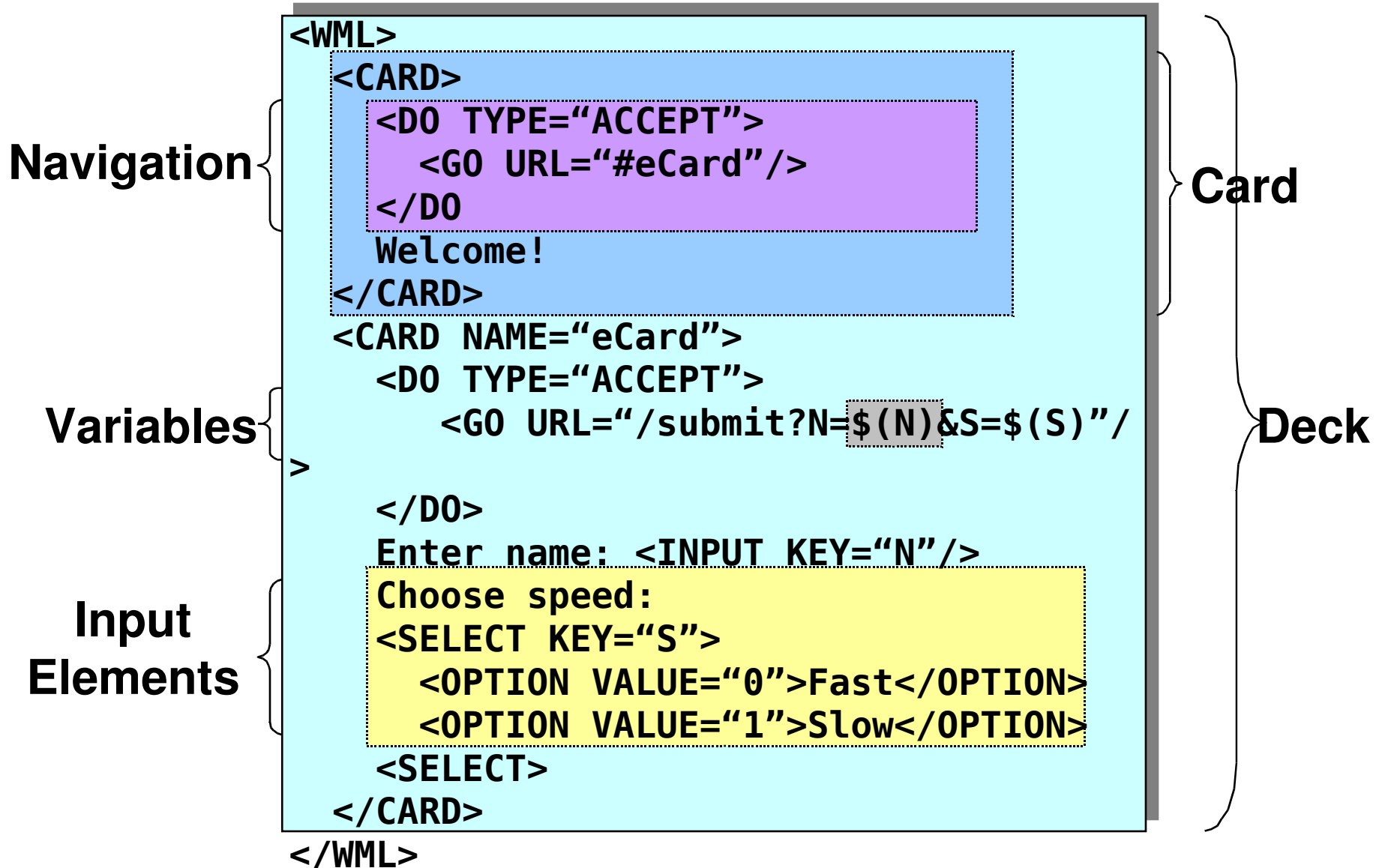
WML File Structure

```
<?xml version="1.0"?>
<!DOCTYPE WML PUBLIC "-//WAPFORUM//DTD WML 1.0//EN"
    "http://www.wapforum.org/DTD/wml.xml">

<WML>
    ...
</WML>
```



WML Example



A Deck of Cards

```
<WML>
  <CARD>
    <DO TYPE="ACCEPT" LABEL="Next">
      <GO URL="#card2"/>
    </DO>
    Acme Inc.<BR/>Directory
  </CARD>

  <CARD NAME="card2">
    <DO TYPE="ACCEPT">
      <GO URL="?send=$type"/>
    </DO>
    Services
    <SELECT KEY="type">
      <OPTION VALUE="em">Email</OPTION>
      <OPTION VALUE="ph">Phone</OPTION>
      <OPTION VALUE="fx">Fax</OPTION>
    </SELECT>
  </CARD>
</WML>
```

**Acme Inc.
Directory**

Next



Services
1>Email
2 Phone

OK

The DO Element

- Binds a task to a user action
 - Action type: *ACCEPT, OPTIONS, HELP
PREV, DELETE, RESET*
 - Label: *Text string or image (optional)*
 - Task: *GO
PREV, REFRESH, NOOP*
 - Destination: *URL*
 - Post data: *if METHOD=POST*

```
<DO TYPE="ACCEPT" LABEL="Next">  
  <GO URL="http://www.mysite.com/myapp.wml"/>  
</DO>
```

Anchored Links

- Bind a task to the ACCEPT action, when cursor points to a link
 - **TITLE=** sets the label string (default = “Link”)
 - Links are not allowed in select list options

```
<CARD>
  Please visit our
    <A TITLE="Visit">
      <GO URL="home.wml"/>home page</A>
    for details.
</CARD>
```

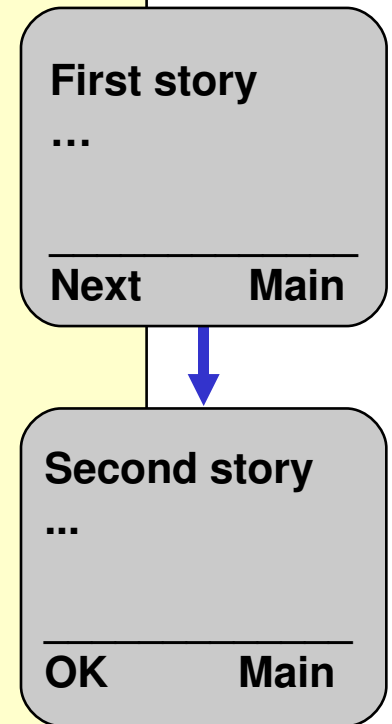
Please visit
▶ **our home**
page for

Visit

The TEMPLATE Element

- Defines actions & events for all cards in a deck

```
<WML>
  <TEMPLATE>
    <DO TYPE="OPTIONS" LABEL="Main">
      <GO URL="main_menu.wml"/>
    </DO>
  </TEMPLATE>
  <CARD NAME="msg1">
    <DO TYPE="ACCEPT" LABEL="Next">
      <GO URL="#msg2"/>
    </DO>
    First story
  </CARD>
  <CARD NAME="msg2">
    Second story
  </CARD>
</WML>
```




Handling User Input

- Select lists
 - Choose from a list of options
- Input fields
 - Enter a string of text or numbers
- KEY variables
 - Set by **SELECT** and **INPUT** elements
 - How user input is passed to other cards and the application server

The SELECT Element

- Display a list of options
 - Each option may set the KEY variable and/or bind a task to the ACCEPT key
 - TITLE= dynamically sets the label string
 - MULTIPLE="TRUE": Allows user to pick multiple items

```
<CARD>
  <DO TYPE="ACCEPT" LABEL="View">
    <GO URL="getcity.cgi?location=$city"/>
  </DO>
  Forecast
  <SELECT KEY="city">
    <OPTION VALUE="ber">Berlin</OPTION>
    <OPTION VALUE="rom">Rome</OPTION>
    <OPTION TITLE="Find" ONCLICK="find.cgi">New City</OPTION>
  </SELECT>
</CARD>
```



The INPUT Element

- Prompts user to enter a string of text
 - `DEFAULT=key_value`; Default KEY variable (displayed to user)
 - `FORMAT=format_specifier`; If omitted, free-form entry is allowed
 - `EMPTYOK="TRUE"`; Browser will accept null input
 - `TYPE="PASSWORD"`; Special entry mode handled by the browser
 - `MAXLENGTH=number`; Maximum number of allowed characters

```
<CARD>
  <D0 TYPE="ACCEPT">
    <GO URL="?get=person"
      METHOD="POST" POSTDATA="userid=$ssn"/>
  </D0>
  Soc Security:
  <INPUT KEY="ssn" FORMAT="NNN\ -NN\ -NNNN"/>
</CARD>
```

Soc. Security:

287-33- _

NUM

Soc. Security:

287-33- 7629

OK

WML Content Formats

- Common interchange formats, for interoperability
- Formats:
 - Business cards: IMC vCard standard
 - Calendar: IMC vCalendar standard
 - Images: WBMP (Wireless BitMaP)
 - Compiled WML, WMLScript
- Newly defined formats:
 - WML text and tokenized format
 - WMLScript text and bytecode format
 - WBMP image format
- Binary format for size reduction
 - Bytecodes/tokens for common values and operators
 - Compressed headers
 - Data compression (e.g. images)
- General-purpose transport compression can still be applied

Displaying Images

- Insert app images or local icons within display text
 - 1-bit BMP format
- Images are ignored by non-bitmapped devices
 - Check `HTTP_ACCEPT` for “image/bmp”

```
<CARD>
  <DO TYPE="ACCEPT">
    <GO URL="#c2"/>
  </DO>
  Continue <IMG LOCALSRC="righthand"
             ALT="forward..."/>
</CARD>

<CARD NAME="c2">
  <IMG SRC="../images/logo.wbmp"
         ALT="Unwired Planet"/>
  <BR/>Welcome!
</CARD>
```



WML (other features)

- Setting card styles to create forms
- Using variables to cache user data
- Using card intrinsic events to trigger transparent tasks
- Using timers
- Securing WML decks
- Bookmarking decks

WMLScript

- Complement to WML
 - Derived from JavaScript™
- Provides general scripting capabilities
 - Procedural logic, loops, conditionals, etc.
 - Optimized for small-memory, small-cpu devices
- Features
 - local user interaction, validity check of user input
 - access to device facilities (phone call, address book etc.)
 - extensions to the device software
 - configure device, download new functionality after deployment
- Bytecode-based virtual machine
 - Stack-oriented design, ROM-able
 - Designed for simple, low-impact implementation
- WMLScript compiler resides in the network

WMLScript Libraries

- Lang - VM constants, general-purpose math functionality, etc.
- String - string processing functions
- URL - URL processing
- Browser - WML browser interface
- Dialog - simple user interface
- Float - floating point functions

WMLScript Example

Functions

```
function currencyConvertor(currency, exchRate)
{
    return currency*exchangeRate;
}
```

Variables

```
function myDay(sunShines) {
    var myDay;
    if (sunShines) {
        myDay = "Good";
    } else {
        myDay = "Not so good";
    };
    return myDay;
}
```

Programming
Constructs

Wireless Telephony Application (WTA)

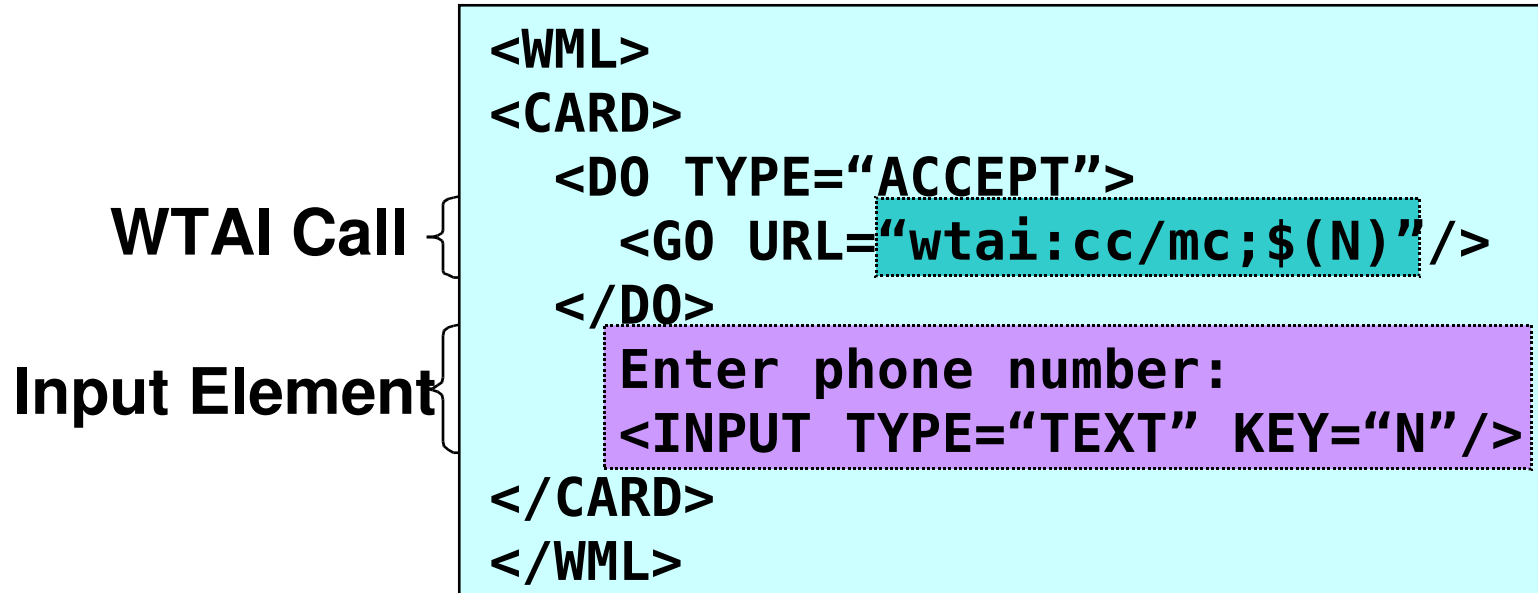
- Collection of telephony specific extensions
 - designed primarily for network operators
- Example
 - calling a number (WML)
`wtai://wp/mc;07216086415`
 - calling a number (WMLScript)
`WTAPublic.makeCall("07216086415");`
- Implementation
 - Extension of basic WAE application model
 - Extensions added to standard WML/WMLScript browser
 - Exposes additional API (WTAI)

WTA Features

- Extension of basic WAE application model
 - network model for interaction
 - client requests to server
 - event signaling: server can push content to the client
 - event handling
 - table indicating how to react on certain events from the network
 - client may now be able to handle unknown events
 - telephony functions
 - some application on the client may access telephony functions
- WTAI includes:
 - Call control
 - Network text messaging
 - Phone book interface
 - Event processing
- Security model: segregation
 - Separate WTA browser
 - Separate WTA port

WTA Example (WML)

Placing an outgoing call with WTAI:



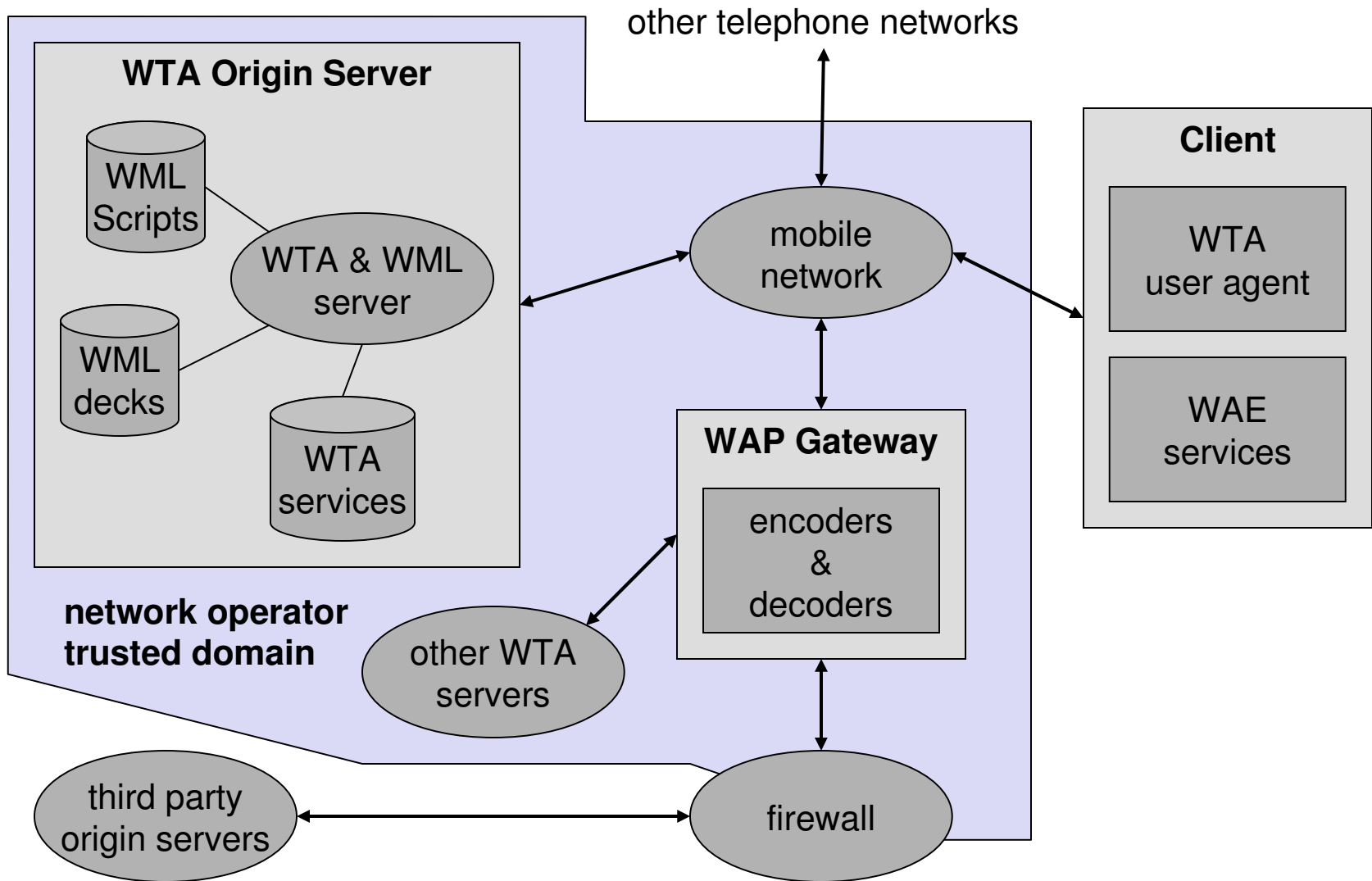
WTA Example (WMLScript)

Placing an outgoing call with WTAI:

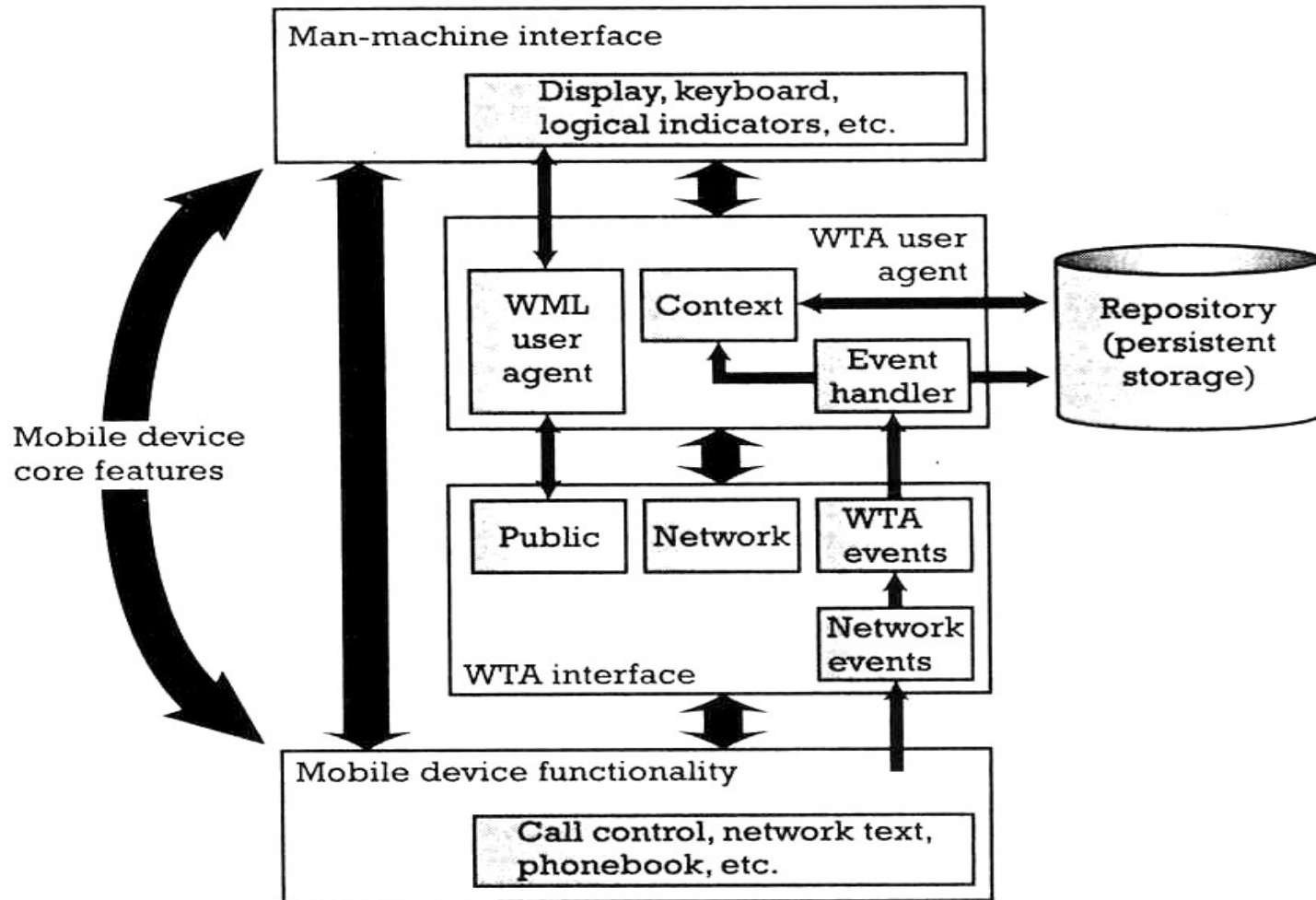
WTAI Call {

```
function checkNumber(N) {  
    if (Lang.isInt(N))  
        WTAI.makeCall(N);  
    else  
        Dialog.alert("Bad phone number");  
}
```

WTA Logical Architecture



WTA Framework Components



WTA User Agent

- **WTA User Agent**
 - WML User agent with extended functionality
 - can access mobile device's telephony functions through WTAI
 - can store WTA service content persistently in a repository
 - handles events originating in the mobile network
- **WTA User Agent Context**
 - Abstraction of execution space
 - Holds current parameters, navigation history, state of user agent
 - Similar to activation record in a process address space
- Uses connection-mode and connectionless services offered by WSP
- Specific, secure WDP ports on the WAP gateway

WTA Events and Repository

■ WTA Events

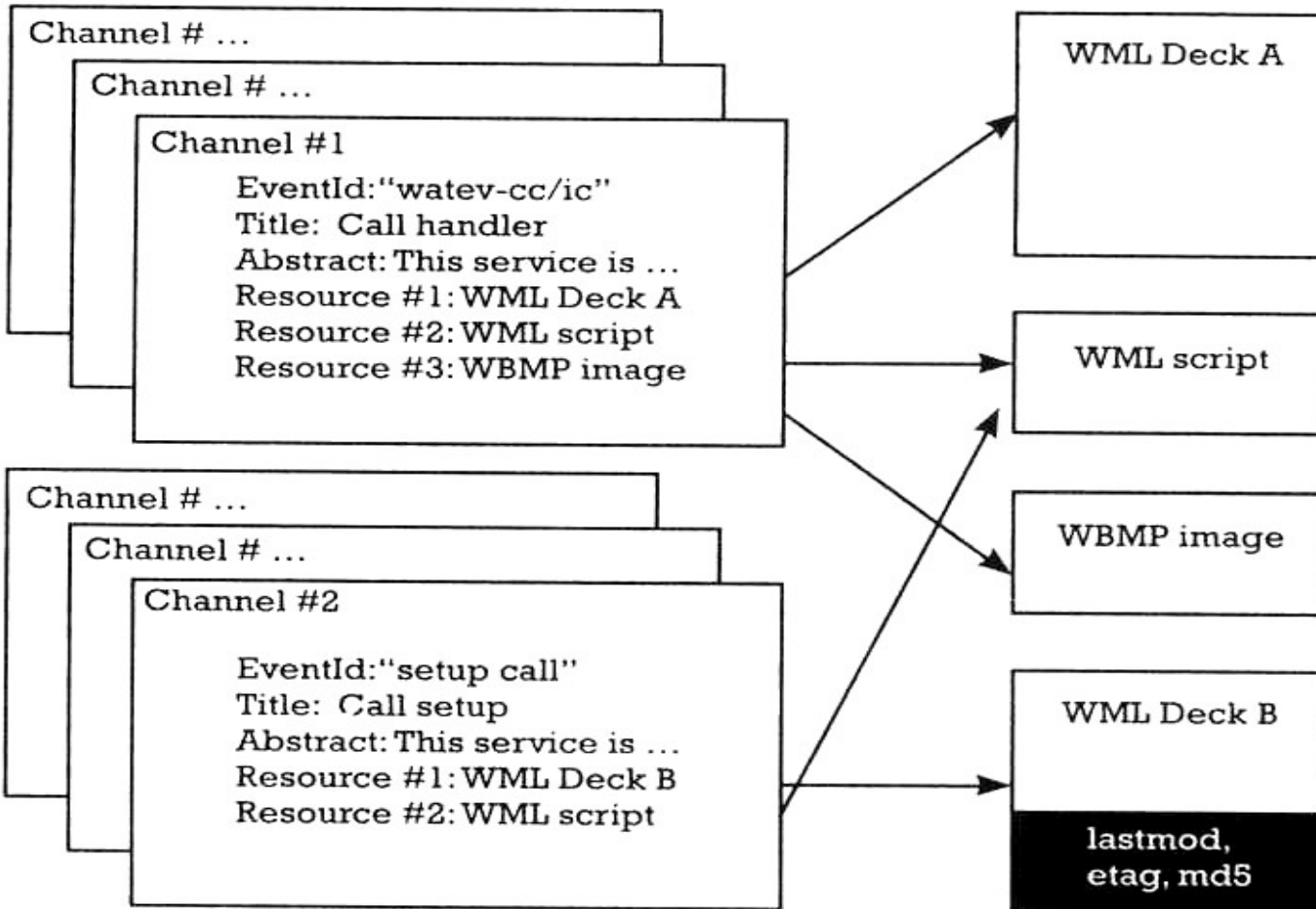
- Network notifies device of event (such as incoming call)
- WTA events map to device's native events
- WTA services are aware of and able to act on these events
- example: incoming call indication, call cleared, call connected

■ WTA Repository

- local store for content related to WTA services (minimize network traffic)
- Channels: define the service
 - content format defining a WTA service stored in repository
 - XML document specifying eventid, title, abstract, and resources that implement a service
- Resources: execution scripts for a service
 - could be WML decks, WML Scripts, WBMP images..
 - downloaded from WTA server and stored in repository before service is referenced
- Server can also initiate download of a channel

WTA Channels and Resources

Repository



WTA Interface (public)

■ WTA Interface

- generic, high-level interface to mobile's telephony functions
- setting up phone calls, reading and writing entries in phonebook..

■ Public WTAI

- for third party WML content providers
- restricted set of telephony functions available to any WAE User Agent
- library functions
 - **make call**: allows application to setup call to a valid tel number
 - **send DTMF tones**: send DTMF tones through the setup call
- user notified to grant permission for service execution
- cannot be triggered by network events
- example: Yellow pages service with “make call” feature

WTA Interface (network)

■ Network Common WTAI

- WTA service provider is in operator's domain
- all WTAI features are accessible, including the interface to WTA events
- library functions
 - **Voice-call control**: setup call, accept, release, send DTMF tones
 - **Network text**: send text, read text, remove text (SMS)
 - **Phonebook**: write, read, remove phonebook entry
 - **Call logs**: last dialed numbers, missed calls, received calls
 - **Miscellaneous**: terminate WTA user agent, protect context
- user can give blanket permission to invoke a function
- example: Voice mail service

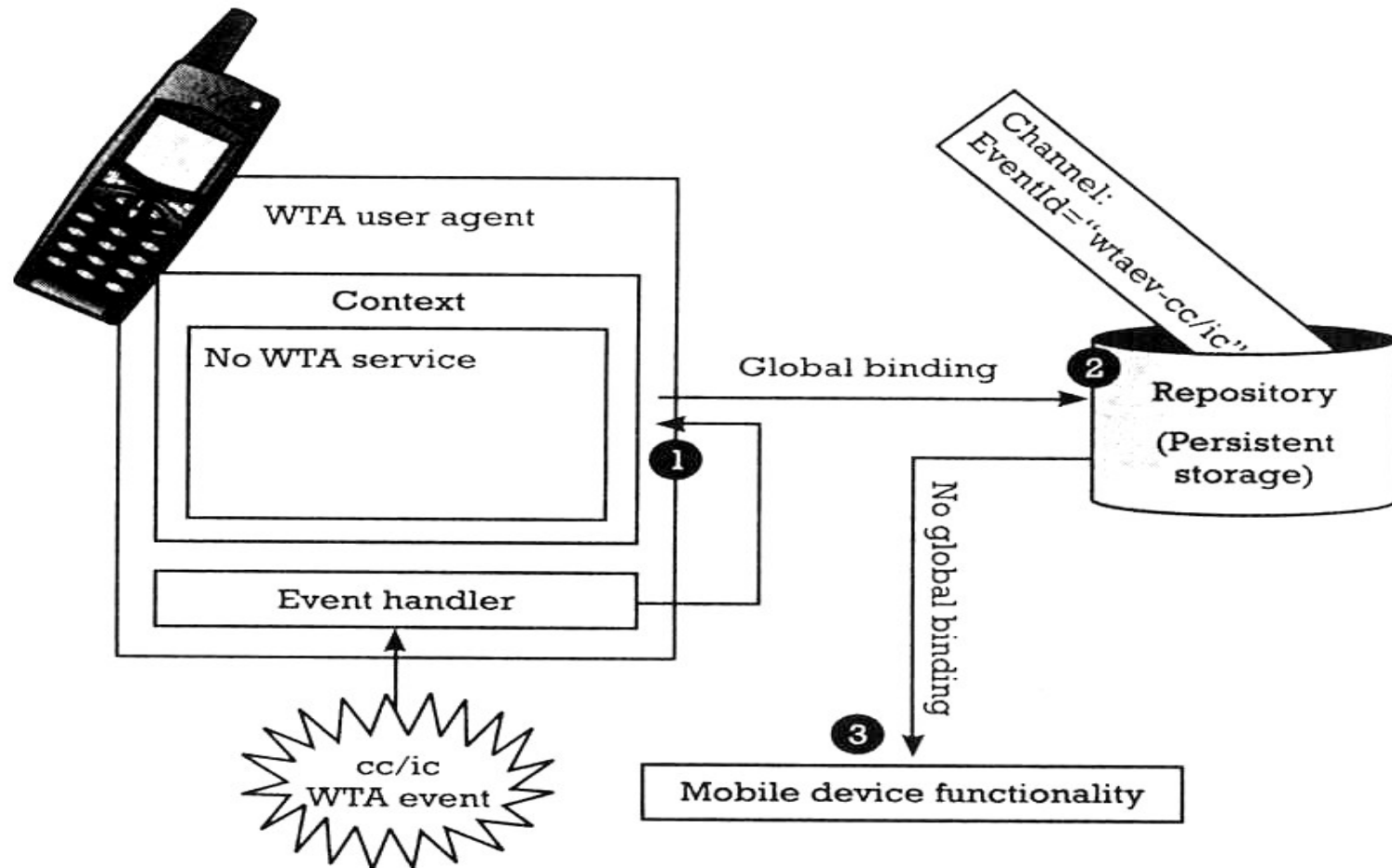
■ Network Specific WTAI

- specific to type of bearer network
- example:
 - GSM: call reject, call hold, call transfer, join multiparty, send USSD

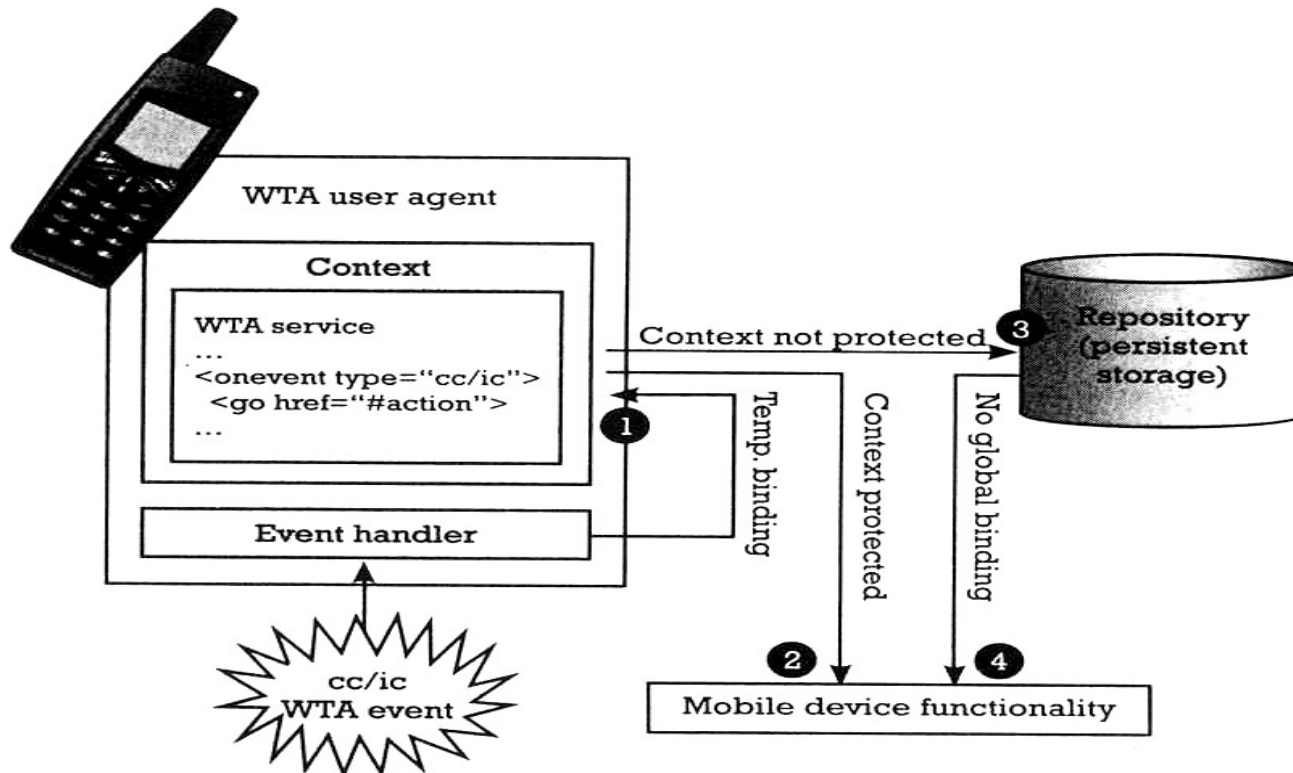
WTA Event Handling

- **Event occurrence**
 - WTA user agent could be executing and expecting the event
 - WTA user agent could be executing and a different event occurs
 - No service is executing
- **Event handling**
 - channel for each event defines the content to be processed upon reception of that event
- **Event binding**
 - association of an event with the corresponding handler (channel)
 - **Global binding:**
 - channel corresponding to the event is stored in the repository
 - event causes execution of resources defined by the channel
 - example: voice mail service
 - **Temporary binding:**
 - resources to be executed are defined by the already executing service
 - example: yellow pages lookup and call establishment

Event Handling (no service in execution)

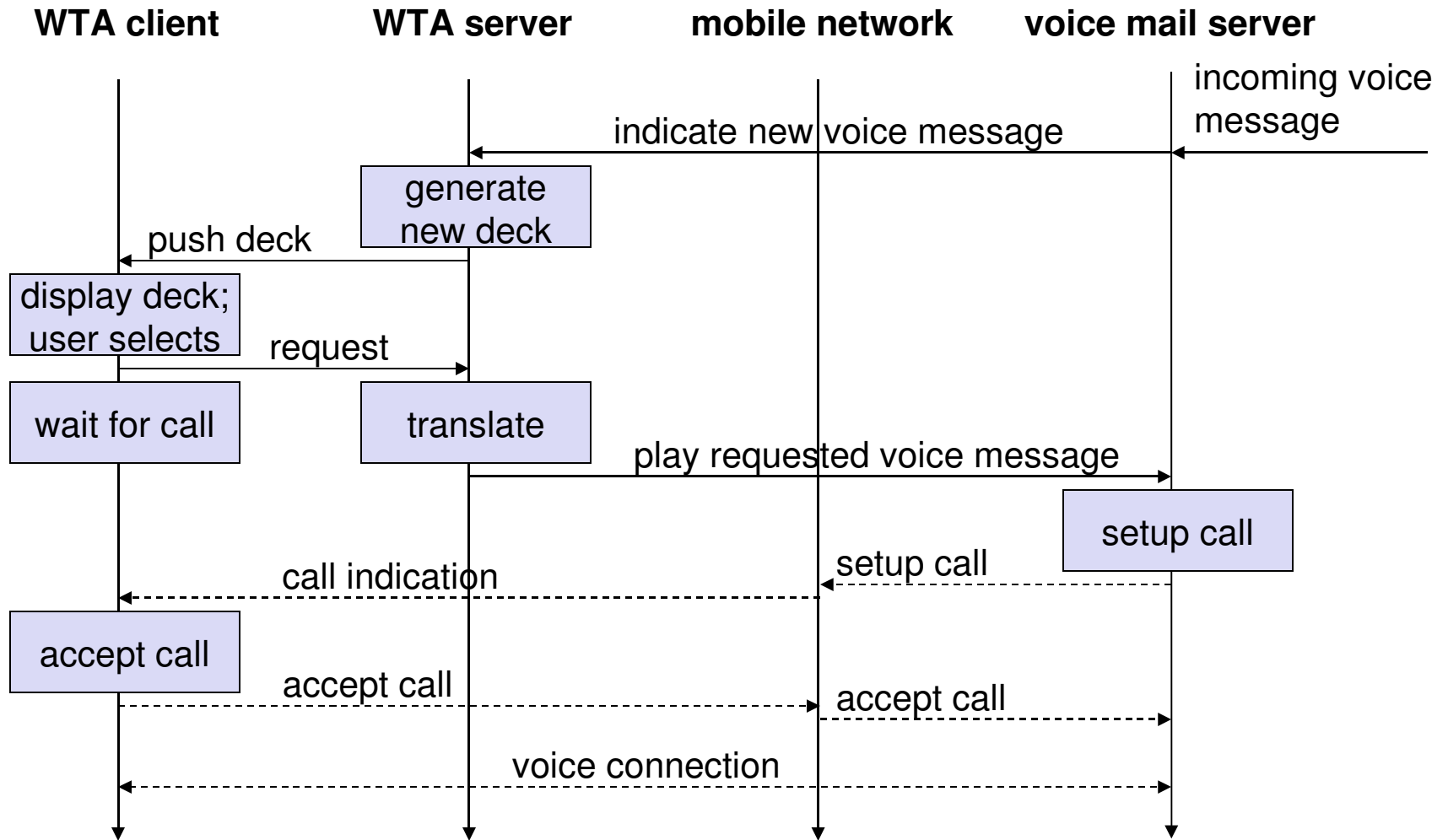


Event Handling (service already execution)



- 1: Temporary binding exists
- 2: No temporary binding and context is protected
- 3: No temporary binding and context is not protected

WTA: Voice mail Example



WTA Application: Example (using WML)

- <WML>
- <CARD>
- <DO TYPE="ACCEPT" TASK="GO" URL="#voteChamp"/>
- Please vote for your champion!
- </CARD>

- <CARD NAME="voteChamp">
- <DO TYPE="ACCEPT" TASK="GO" URL="wtai://cc/sc;\$voteNo;1"/>
- Please choose:
- <SELECT KEY="voteNo">
- <OPTION VALUE="6086415">Mickey</OPTION>
- <OPTION VALUE="6086416">Donald</OPTION>
- <OPTION VALUE="6086417">Pluto</OPTION>
- </SELECT>
- </CARD>
- </WML>

WTA: Example with WML and WMLScript

```
▪ 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");  
▪ }  
▪
```

WTA: Example with WML and WMLScript

- <WML>
- <CARD>
- <DO TYPE="ACCEPT" TASK="GO" URL="#voteChamp"/>
- Please vote for your champion!
- </CARD>
- <CARD NAME="voteChamp">
- <DO TYPE="ACCEPT" TASK="GO"
- URL="/script#voteCall(\$voteNo)"/>
- Please choose:
- <SELECT KEY="voteNo">
- <OPTION VALUE="6086415">Mickey</OPTION>
- <OPTION VALUE="6086416">Donald</OPTION>
- <OPTION VALUE="6086417">Pluto</OPTION>
- </SELECT>
- </CARD>
- <CARD NAME="showResult">
- Status of your call: \$Message \$No
- </CARD>
- </WML>

WAP Push Services

■ Web push

- Scheduled pull by client (browser)
 - example: Active Channels
- no real-time alerting/response
 - example: stock quotes

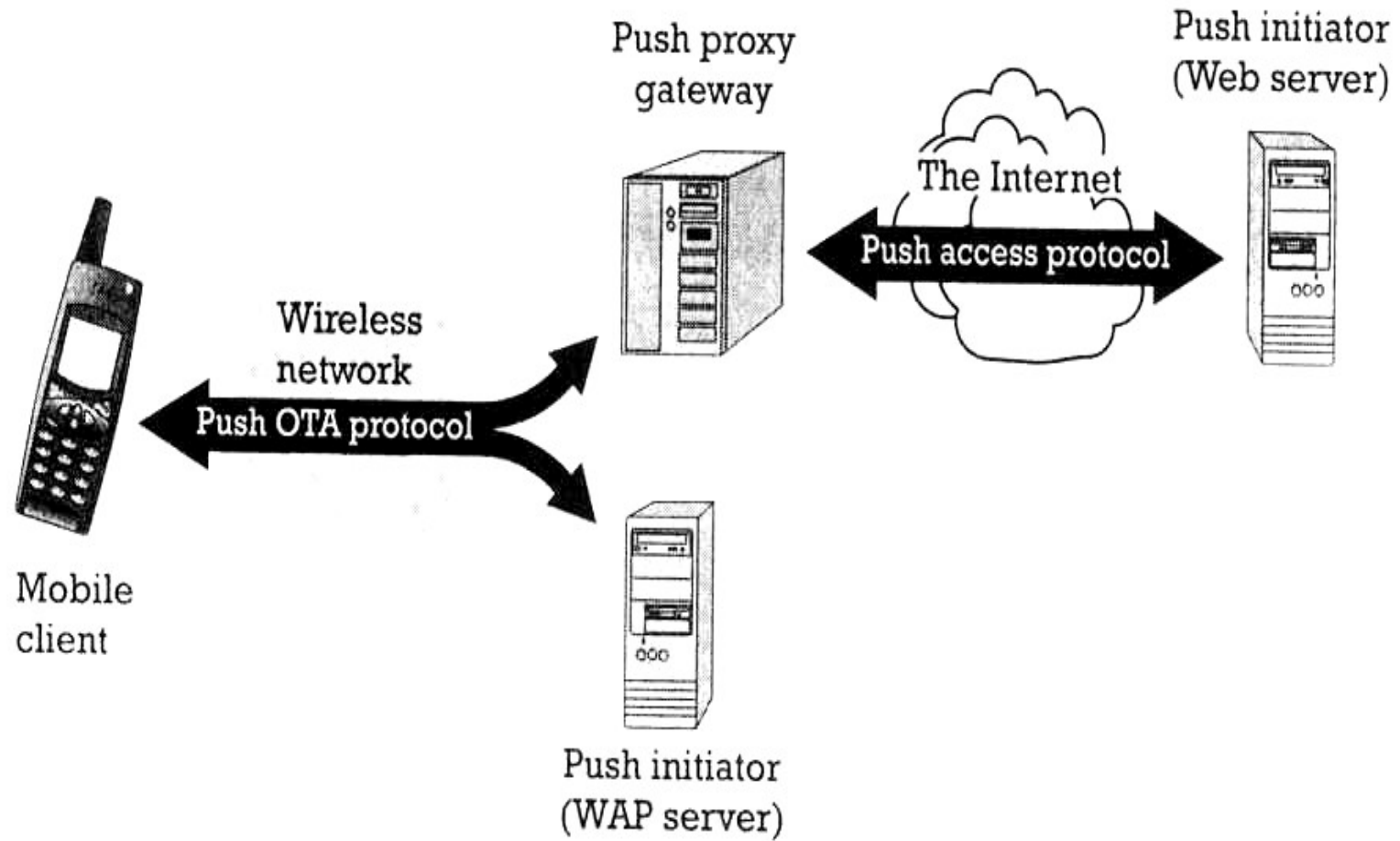
■ Wireless push

- accomplished by using the network itself
 - example: SMS
- limited to simple text, cannot be used as starting point for service
 - example: if SMS contains news, user cannot request specific news item

■ WAP push

- Network supported push of WML content
 - example: Alerts or service indications
- Pre-caching of data (channels/resources)

WAP Push Framework



Push Access Protocol

- Based on request/response model
- Push initiator is the client
- Push proxy is the server
- Initiator uses HTTP POST to send push message to proxy
- Initiator sends control information as an XML document, and content for mobile (as WML)
- Proxy sends XML entity in response indicating submission status
- Initiator can
 - cancel previous push
 - query status of push
 - query status/capabilities of device

Push Proxy Gateway

- WAP stack (communication with mobile device)
- TCP/IP stack (communication with Internet push initiator)
- Proxy layer does
 - control information parsing
 - content transformation
 - session management
 - client capabilities
 - store and forward
 - prioritization
 - address resolution
 - management function

Over the Air (OTA) Protocol

- Extends WSP with push-specific functionality
- Application ID uniquely identifies a particular application in the client (referenced as a URI)
- **Connection-oriented mode**
 - client informs proxy of application IDs in a session
- **Connectionless mode**
 - well known ports, one for secure and other for non-secure push
- **Session Initiation Application (SIA)**
 - unconfirmed push from proxy to client
 - request to create a session for a specific user agent and bearer

WAE Summary

- **WML**
 - analogous to HTML (optimized for wireless)
 - event based, microbrowser user agent
- **WMLScript**
 - analogous to JavaScript
 - features of compiler in the network
- **WTA**
 - WTAI: different access rights for different applications/agents
 - WTA User Agent (analogy with operating systems)
 - Context – Activation Record
 - Channel – Interrupt Handler
 - Resource – Shared routines invoked by interrupt handlers
 - Repository – Library of interrupt handlers
 - feature of dynamically pushing the interrupt handler before the event
- **Push**
 - no analogy in Internet

WAP Gateway Summary

▪ Encoders

- translate between binary (WML) and text (HTML/WML)

▪ Filters

- transcoding between WML (wireless) and HTML (wired)

▪ Method Proxy

- similar to standard proxy services
- WAP stack on wireless interface and TCP/IP stack on Internet interface

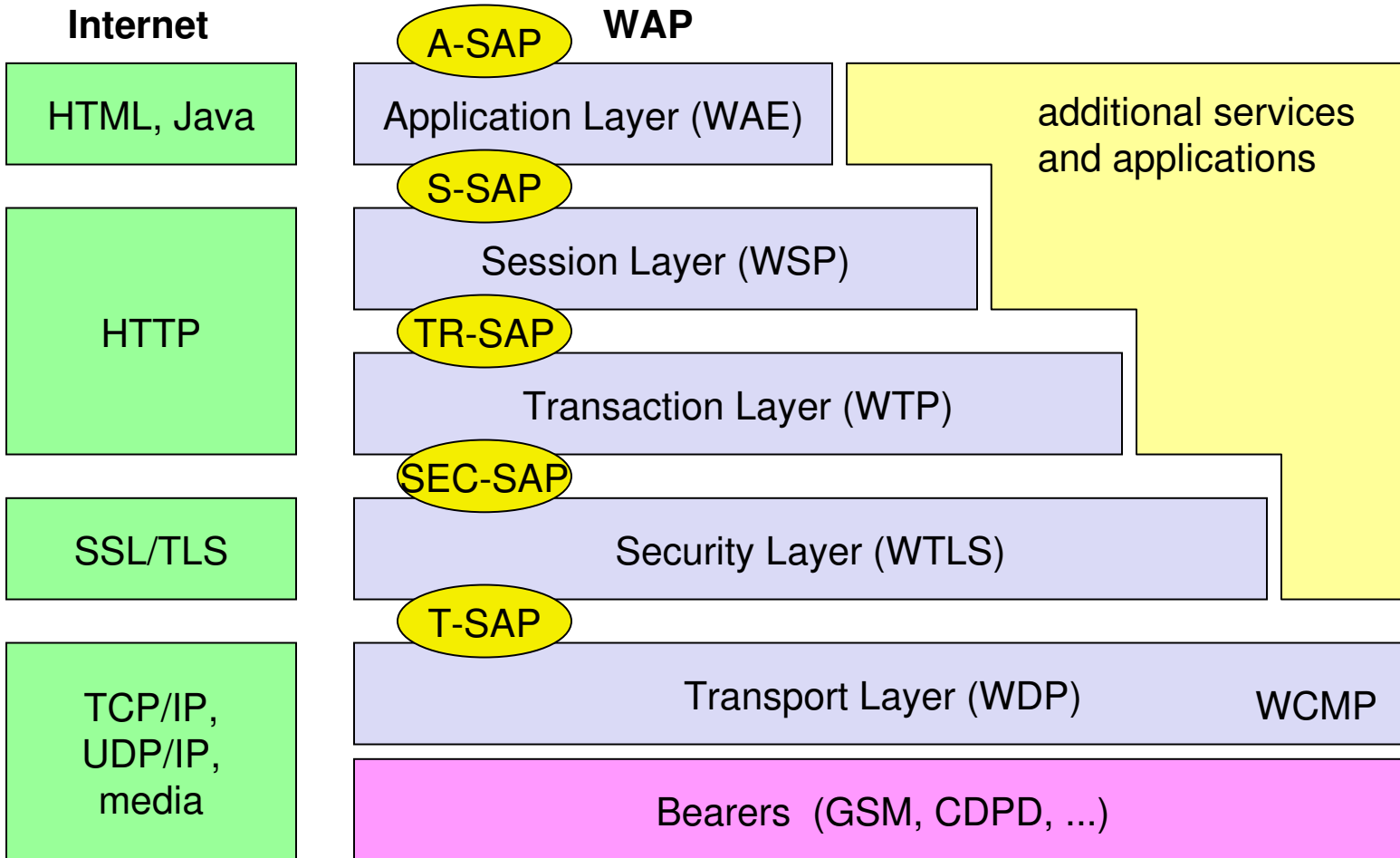
▪ Push Proxy

- Push Access Protocol with Internet Push Initiator (Web Server)
- Over the Air Protocol with mobile device (and WAP Push Initiator)
- Performs necessary filtering, translation etc.

WAP Servers Summary

- **Origin Server**
 - Web server with HTML/WML contents
 - Runs TCP/IP stack, needs PAP protocol for push, no end-to-end security
- **WAP Server**
 - Serves WML content
 - Runs WAP stack, uses OTA protocol for push, end-to-end security possible
- **WTA Server**
 - Specialized for telephony applications (runs WAP stack, uses push extensively)
 - Client initiated (make call “hyperlink” from a Yellow pages service)
 - Server initiated (incoming call from a Voice mail service)

WAP: Protocol Stack



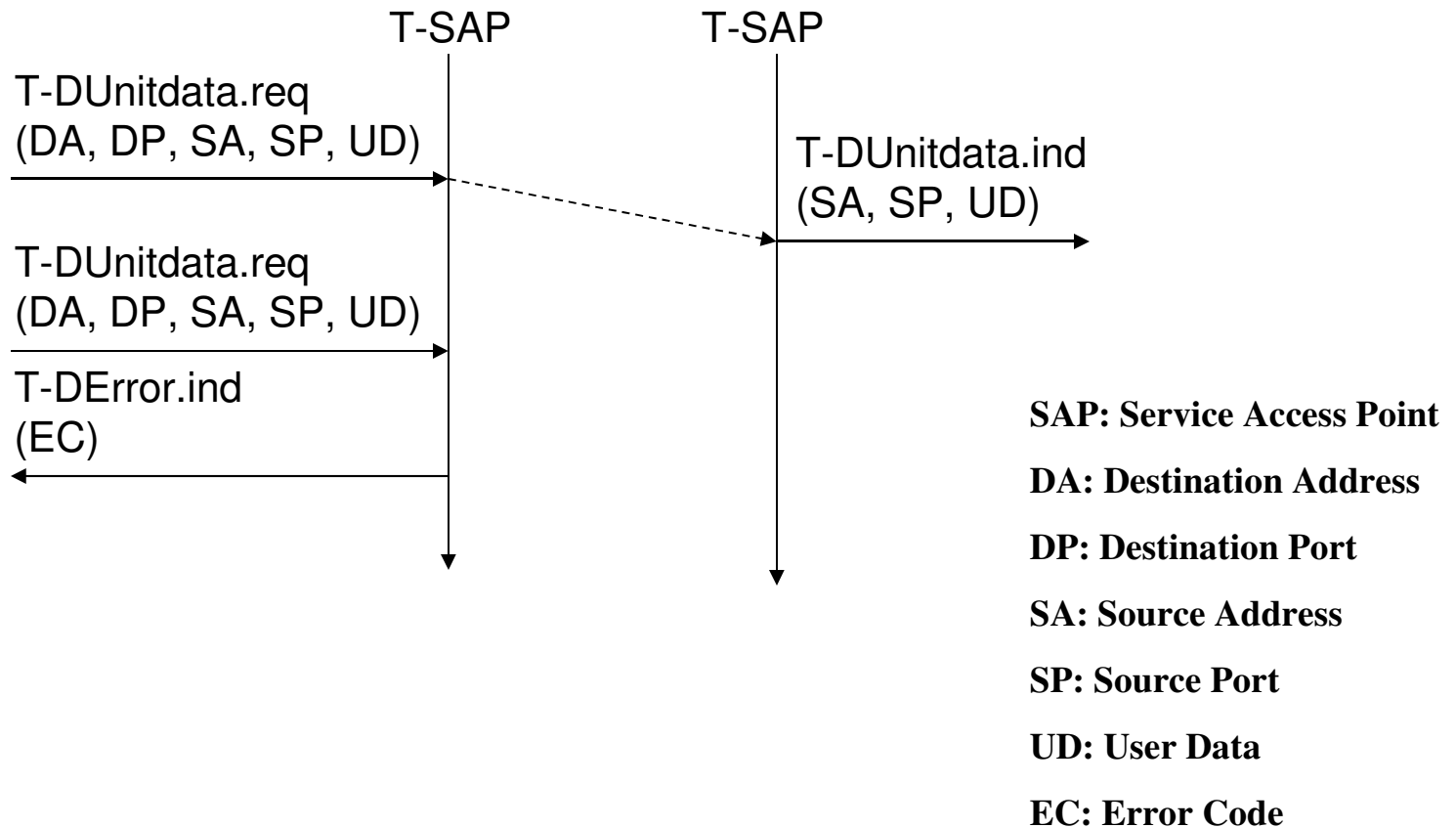
WAE comprises WML (Wireless Markup Language), WML Script, WTAI etc.

WDP: Wireless Datagram Protocol

- Goals
 - create a worldwide interoperable transport system by adapting WDP to the different underlying technologies
 - transmission services, such as SMS in GSM might change, new services can replace the old ones

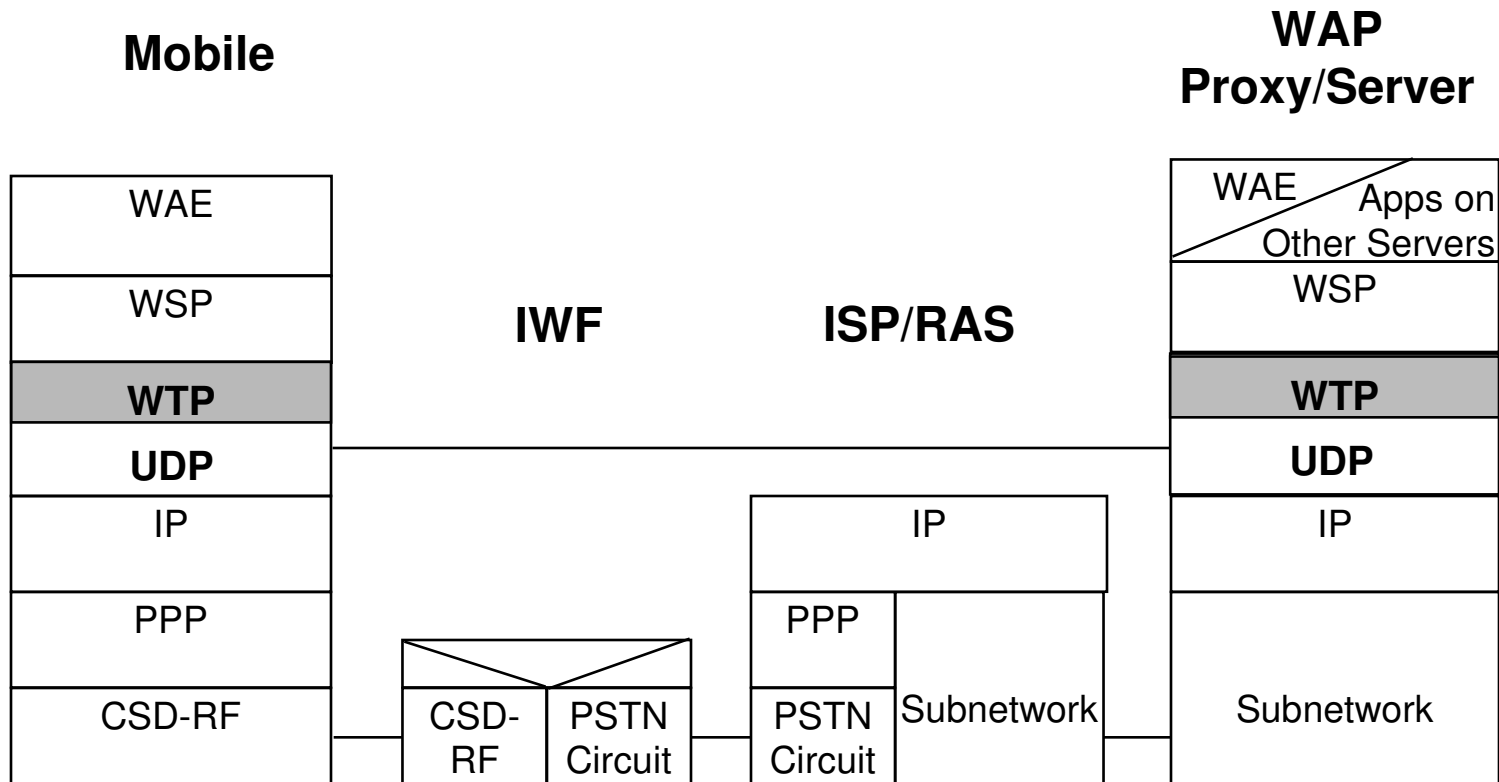
- WDP
 - Transport layer protocol within the WAP architecture
 - uses the Service Primitive
 - T-UnitData.req .ind
 - uses transport mechanisms of different bearer technologies
 - offers a common interface for higher layer protocols
 - allows for transparent communication despite different technologies
 - addressing uses port numbers
 - WDP over IP is UDP/IP

WDP: Service Primitives



Service, Protocol, and Bearer Example

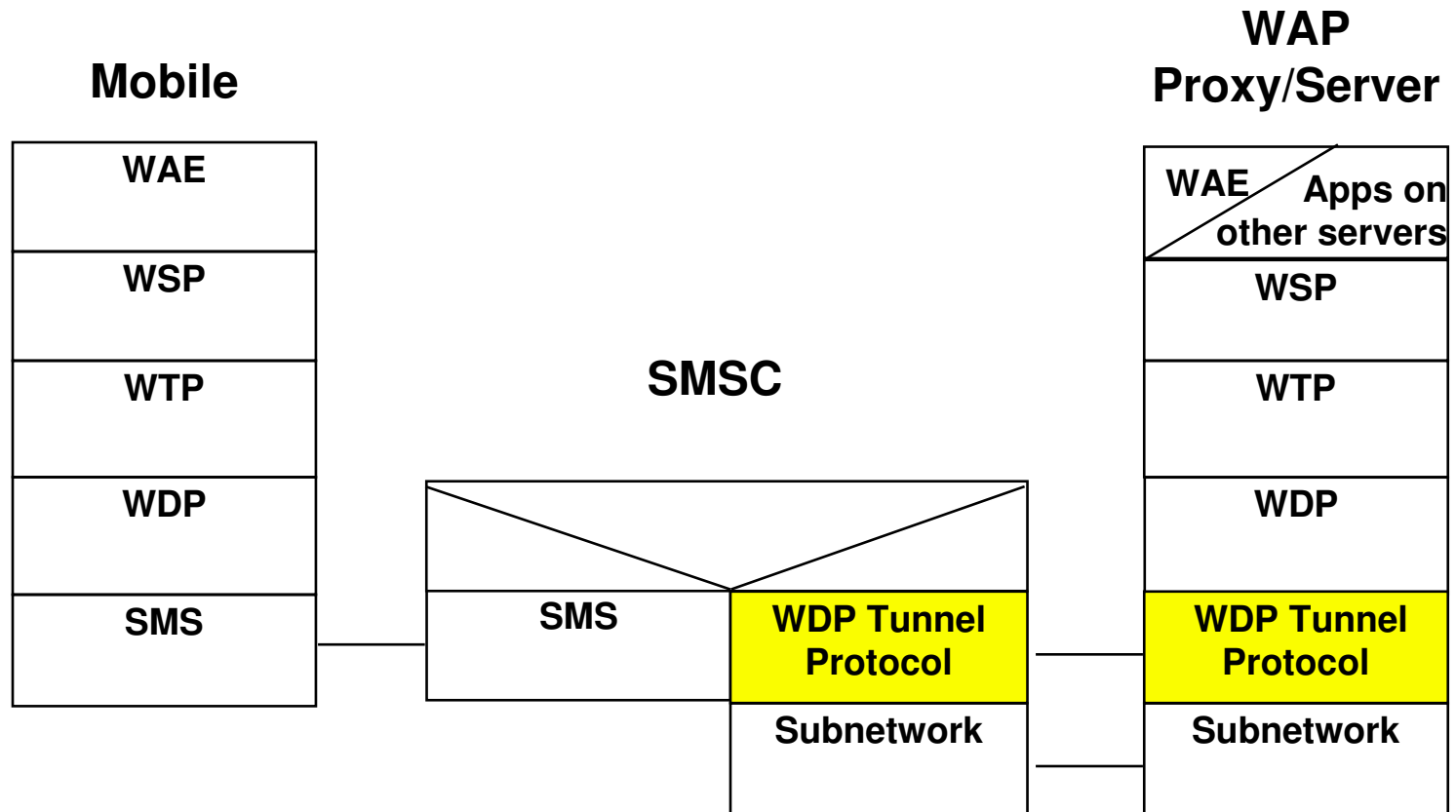
WAP Over GSM Circuit-Switched



RAS - Remote Access Server
IWF - InterWorking Function

Service, Protocol, and Bearer Example

WAP Over GSM Short Message Service



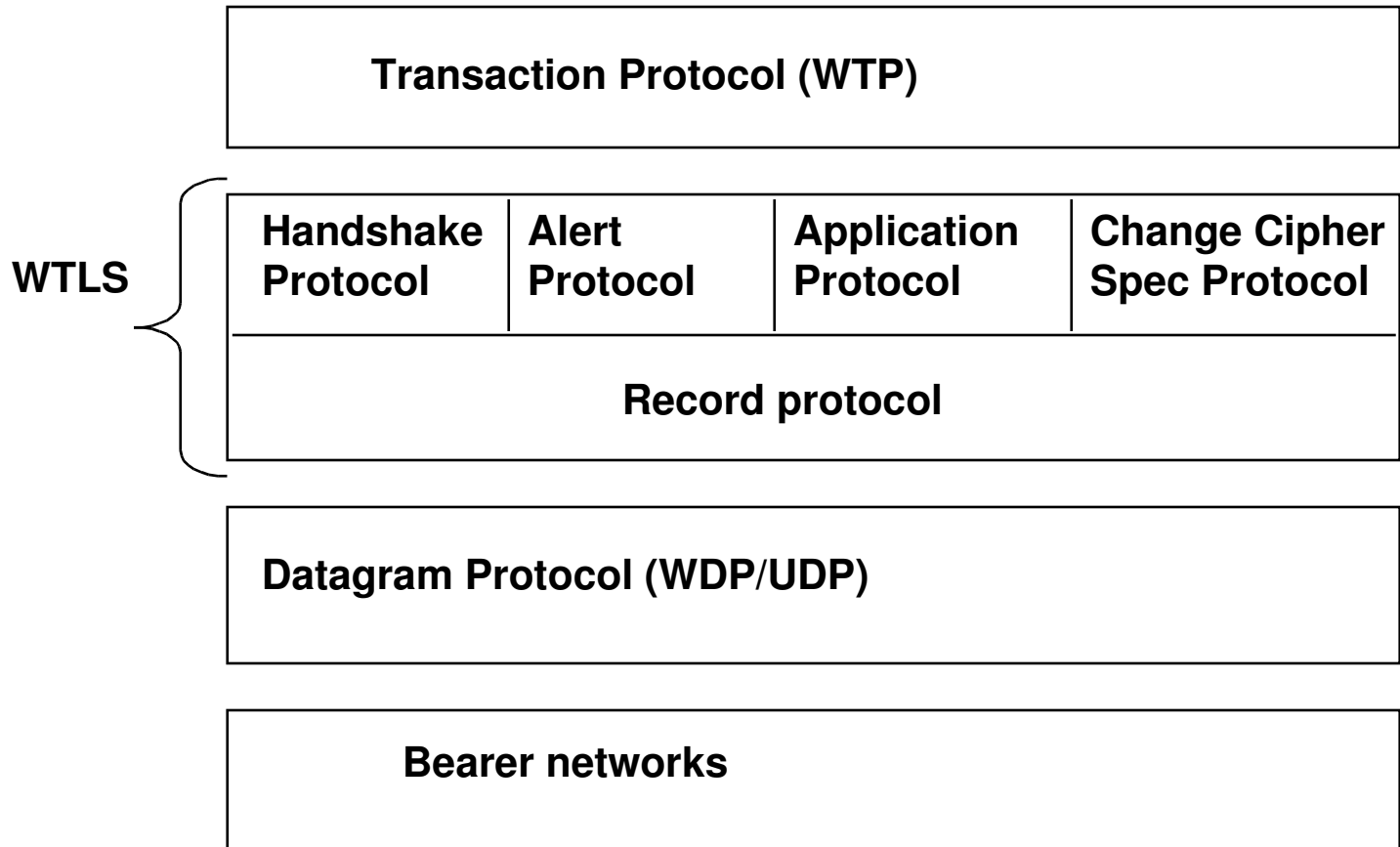
under development

WTLS: Wireless Transport Layer Security

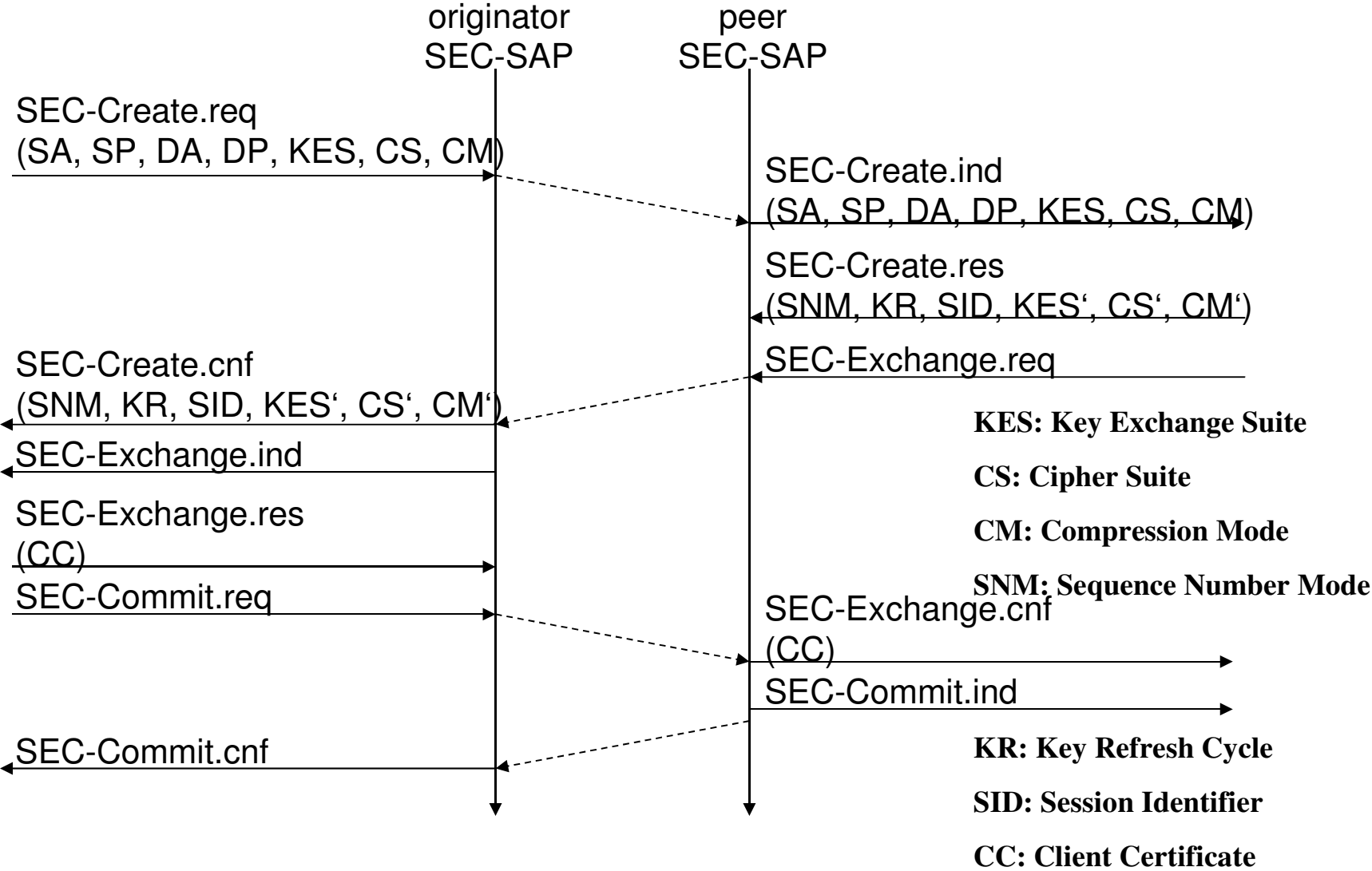
- Goals
 - Provide mechanisms for secure transfer of content, for applications needing privacy, identification, message integrity and non-repudiation
 - Provide support for protection against denial-of-service attacks

- WTLS
 - is based on the TLS/SSL (Transport Layer Security) protocol
 - optimized for low-bandwidth communication channels
 - provides
 - privacy (encryption)
 - data integrity (MACs)
 - authentication (public-key and symmetric)
 - Employs special adapted mechanisms for wireless usage
 - Long lived secure sessions
 - Optimised handshake procedures
 - Provides simple data reliability for operation over datagram bearers

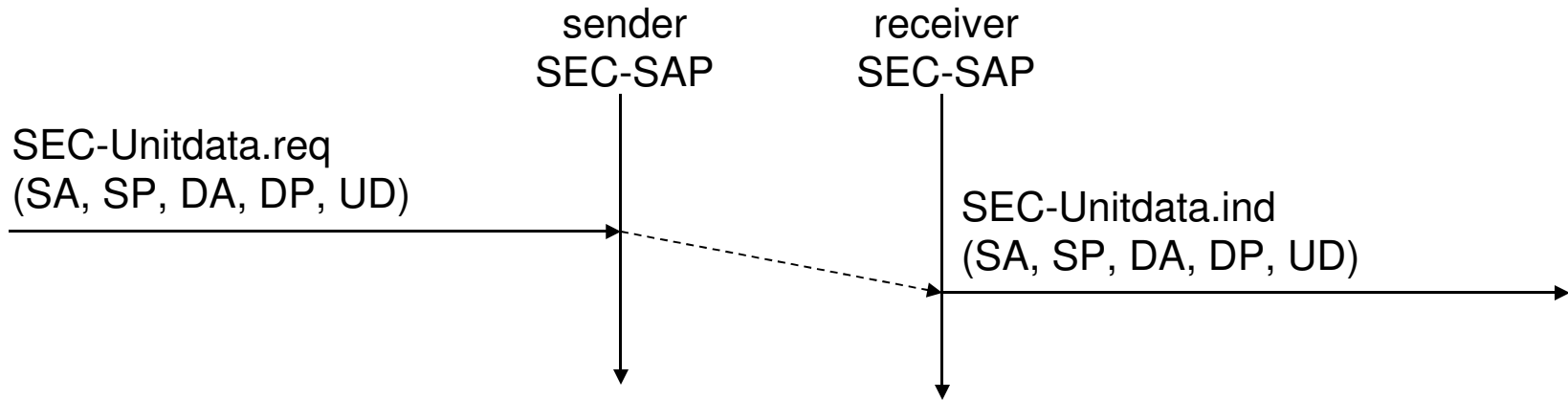
WTLS Internal Architecture



WTLS: Secure session, Full handshake



WTLS: Transferring Datagrams



WTP: Wireless Transaction Protocol

■ Goals

- different transaction services that enable applications to select reliability, efficiency levels
- low memory requirements, suited to simple devices (< 10kbyte)
- efficiency for wireless transmission

■ WTP

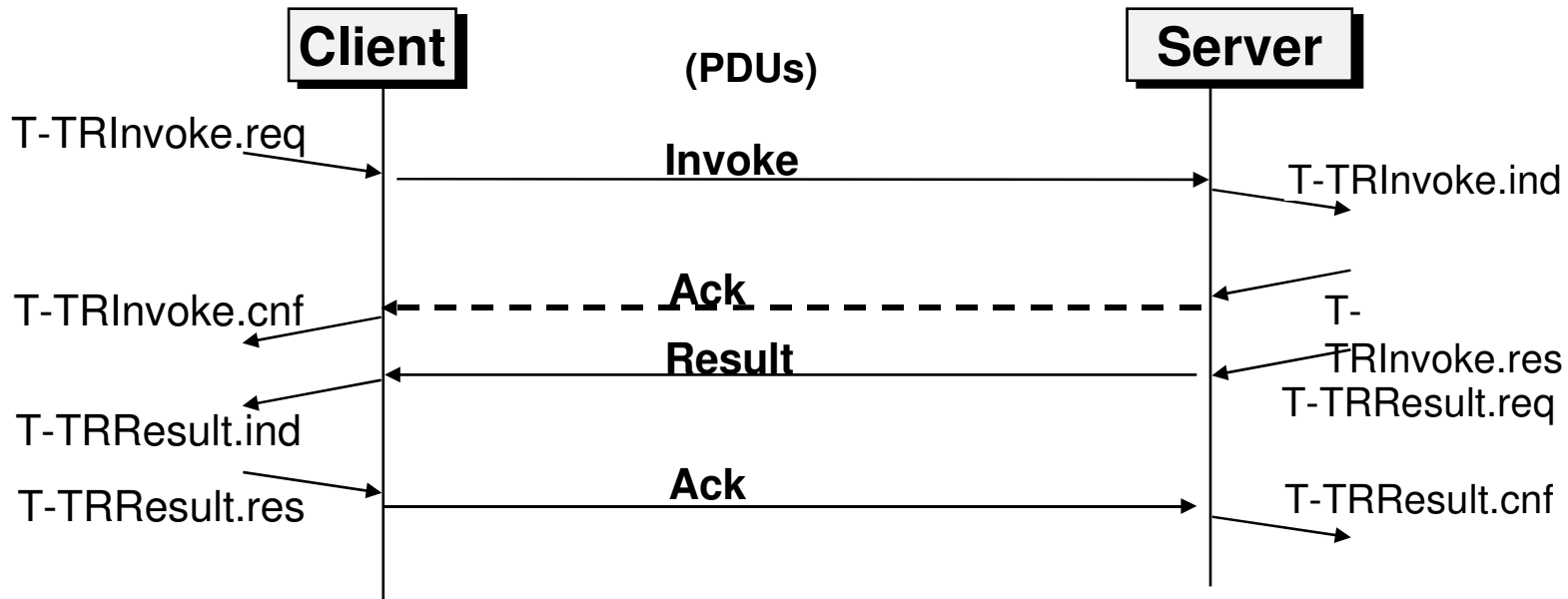
- supports peer-to-peer, client/server and multicast applications
- efficient for wireless transmission
- support for different communication scenarios
- **class 0**: unreliable message transfer
 - unconfirmed Invoke message with no Result message
 - a datagram that can be sent within the context of an existing Session
- **class 1**: reliable message transfer without result message
 - confirmed Invoke message with no Result message
 - used for data push, where no response from the destination is expected
- **class 2**: reliable message transfer with exactly one reliable result message
 - confirmed Invoke message with one confirmed Result message
 - a single request produces a single reply

WTP Services and Protocols

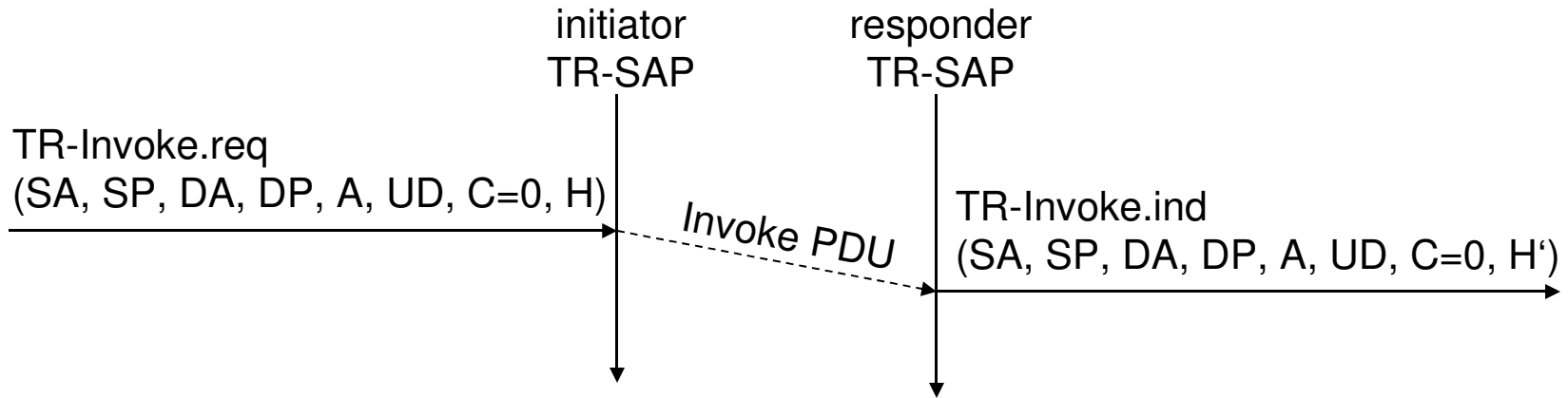
- WTP (Transaction)
 - provides reliable data transfer based on request/reply paradigm
 - no explicit connection setup or tear down
 - optimized setup (data carried in first packet of protocol exchange)
 - seeks to reduce 3-way handshake on initial request
 - supports
 - header compression
 - segmentation /re-assembly
 - retransmission of lost packets
 - selective-retransmission
 - port number addressing (UDP ports numbers)
 - flow control
 - message oriented (not stream)
 - supports an Abort function for outstanding requests
 - supports concatenation of PDUs
 - supports User acknowledgement or Stack acknowledgement option
 - acks may be forced from the WTP user (upper layer)
 - default is stack ack

WTP Services and Protocols

- uses the service primitives
 - T-TRInvoke.req .cnf .ind .res
 - T-TRResult.req .cnf .ind .res
 - T-Abort.req .ind



WTP Class 0 Transaction

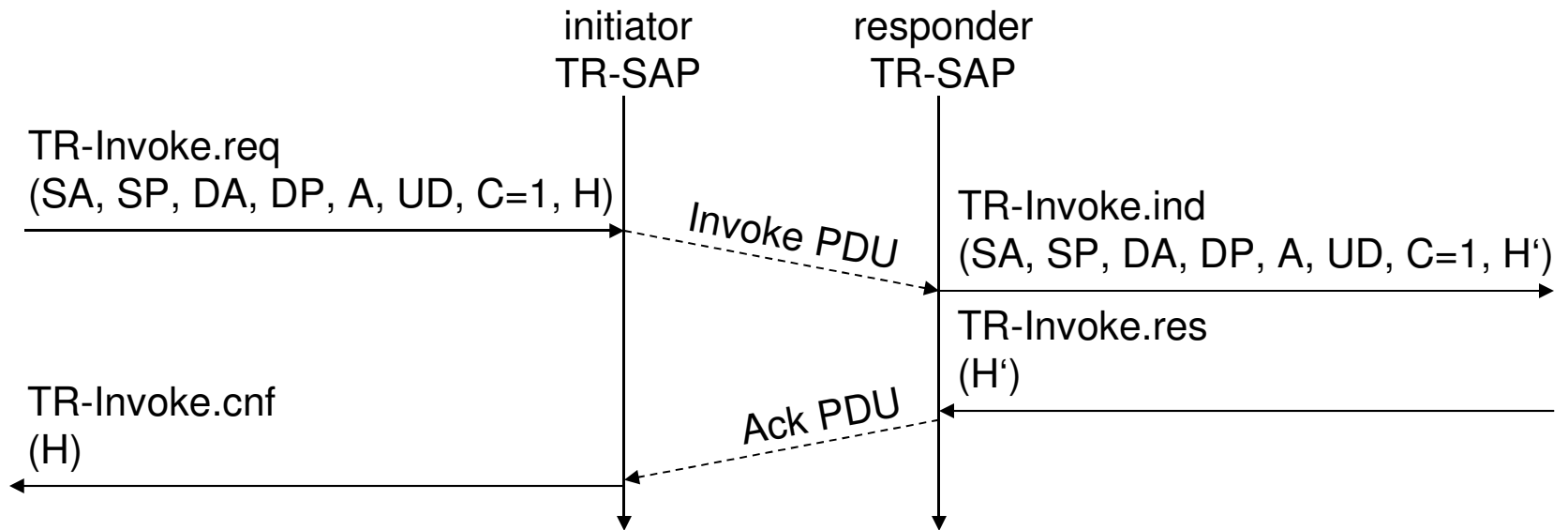
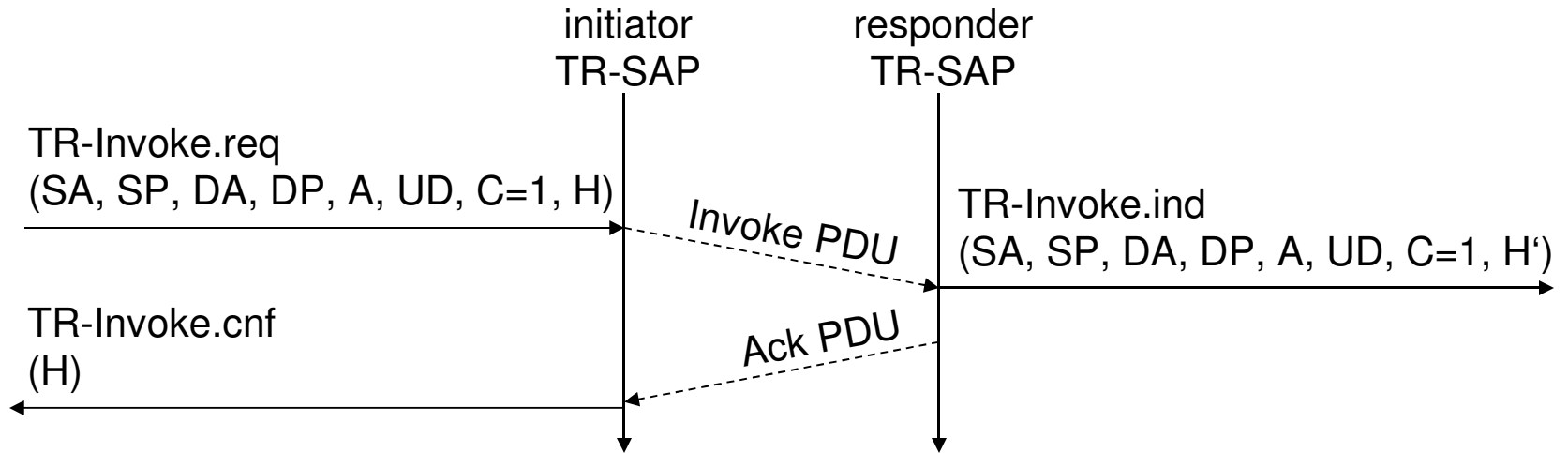


A: Acknowledgement Type
(WTP/User)

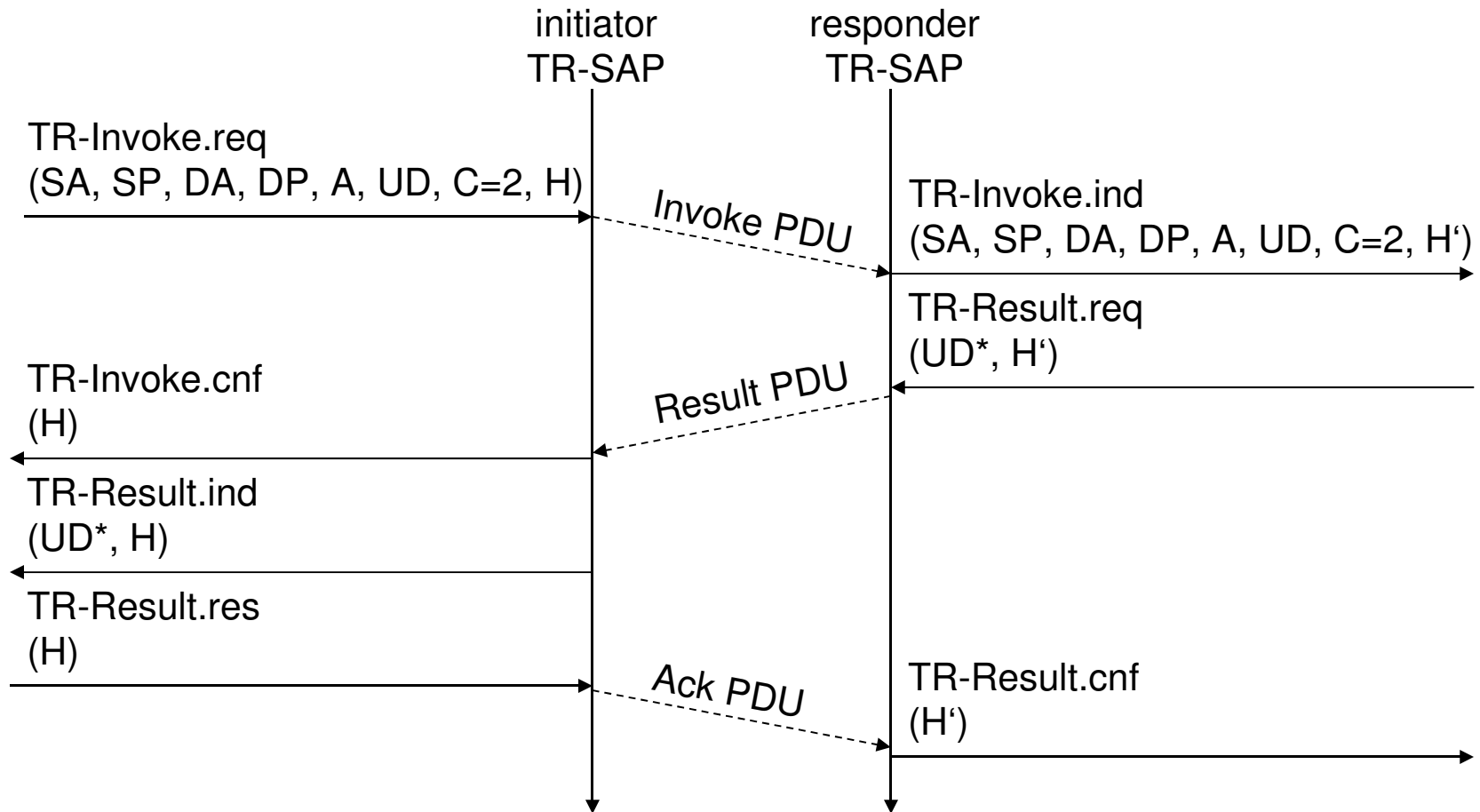
C: Class (0,1,2)

H: Handle (socket alias)

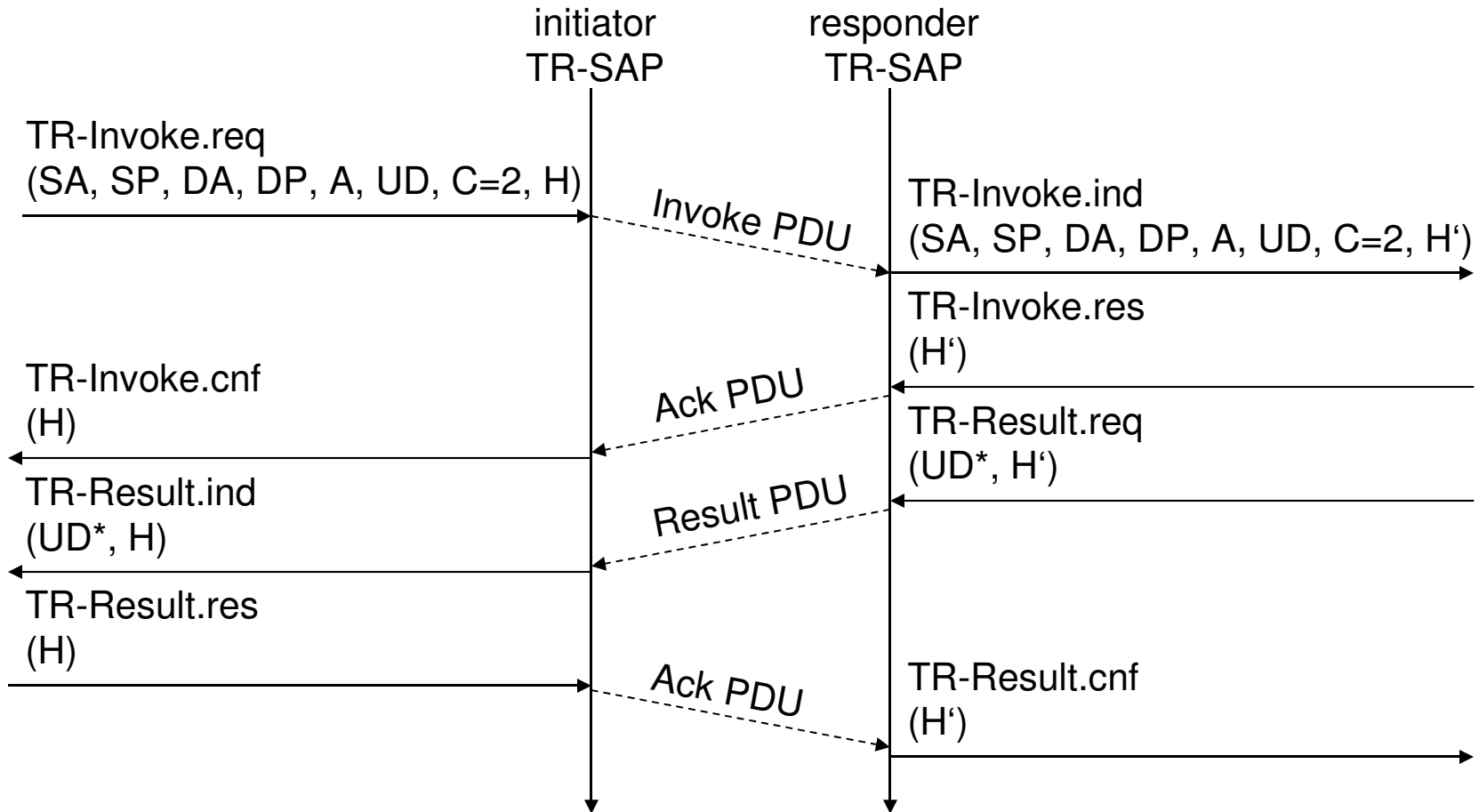
WTP Class 1 Transaction, no user ack & user ack



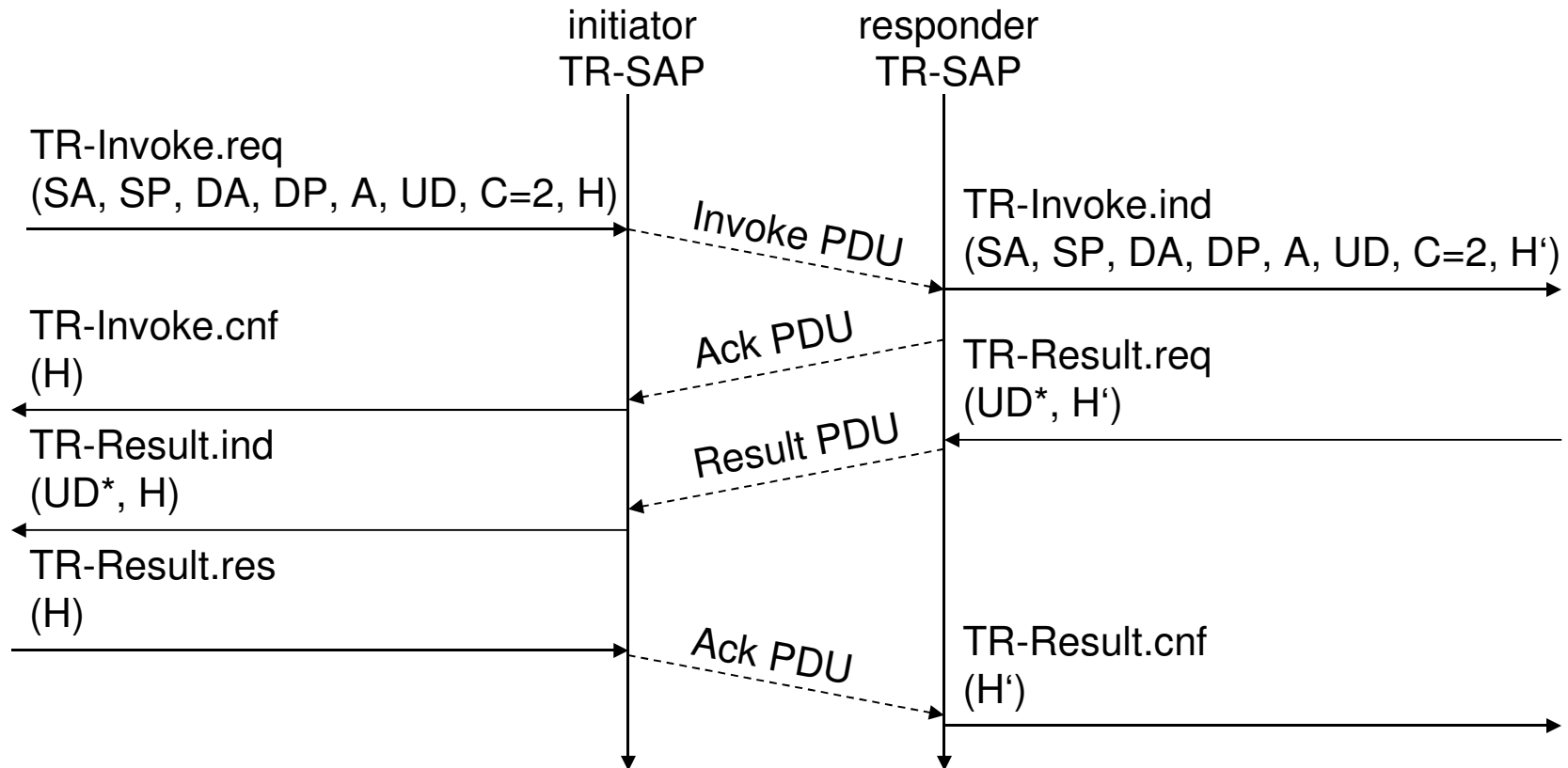
WTP Class 2 Transaction, no user ack, no hold on



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

- WSP Services
 - provides shared state between client and server, optimizes content transfer
 - session management (establish, release, suspend, resume)
 - efficient capability negotiation
 - content encoding
 - push

- WSP/B (Browsing)
 - HTTP/1.1 functionality - but binary encoded
 - exchange of session headers
 - push and pull data transfer
 - asynchronous requests

HTTP 1.1 and WSP

- **HTTP 1.1**
 - extensible request/reply methods
 - extensible request/reply headers
 - content typing
 - composite objects
 - asynchronous requests
- **WSP enhancements beyond HTTP**
 - binary header encoding
 - session headers
 - confirmed and non-confirmed data push
 - capability negotiation
 - suspend and resume
 - fully asynchronous requests
 - connectionless service
- **Why Not HTTP?**
 - encoding not compact enough, inefficient capability negotiation
 - no push facility

WSP Overview

- **Header Encoding**
 - compact binary encoding of headers, content type identifiers and other well-known textual or structured values
 - reduces the data actually sent over the network
- **Capabilities** (are defined for):
 - message size, client and server
 - protocol options: Confirmed Push Facility, Push Facility, Session Suspend Facility, Acknowledgement headers
 - maximum outstanding requests
 - extended methods
 - header code pages
- **Suspend and Resume**
 - server knows when client can accept a push
 - multi-bearer devices
 - dynamic addressing
 - allows the release of underlying bearer resources

WSP Sessions

- **Session Context and Push**

- push can take advantage of session headers
- server knows when client can accept a push

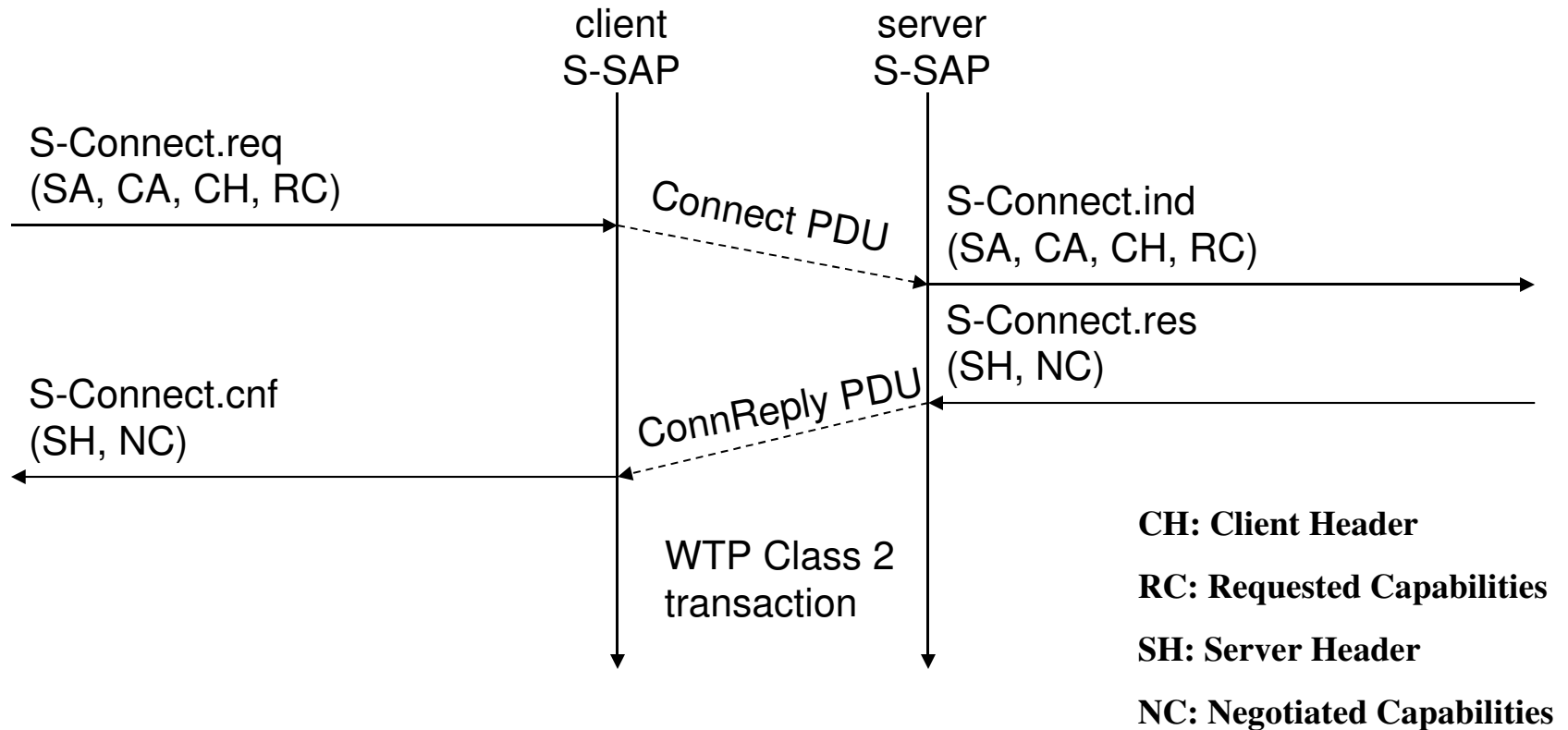
- **Connection-mode**

- long-lived communication, benefits of the session state, reliability

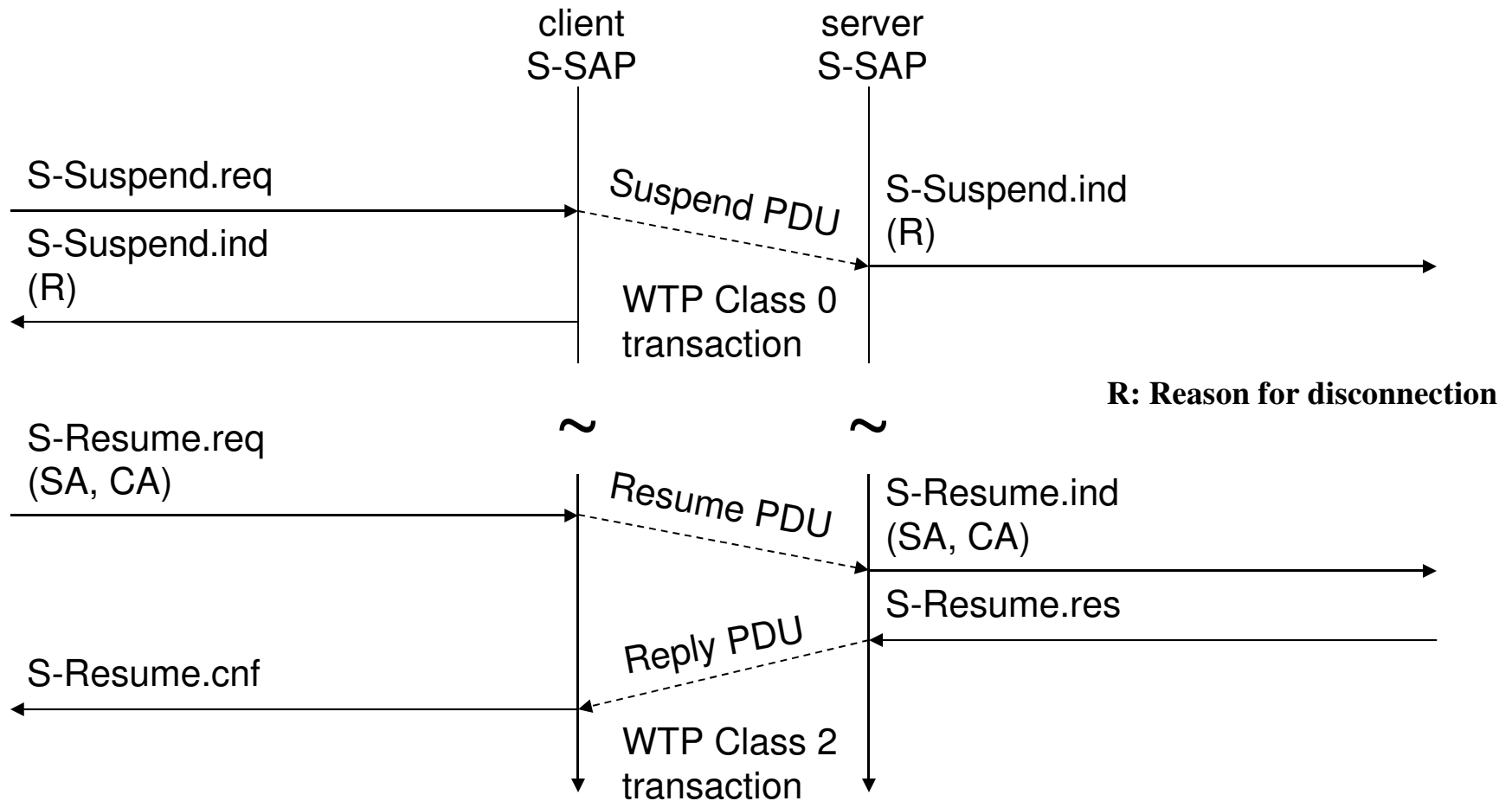
- **Connectionless-mode**

- stateless applications, no session creation overhead, no reliability overhead

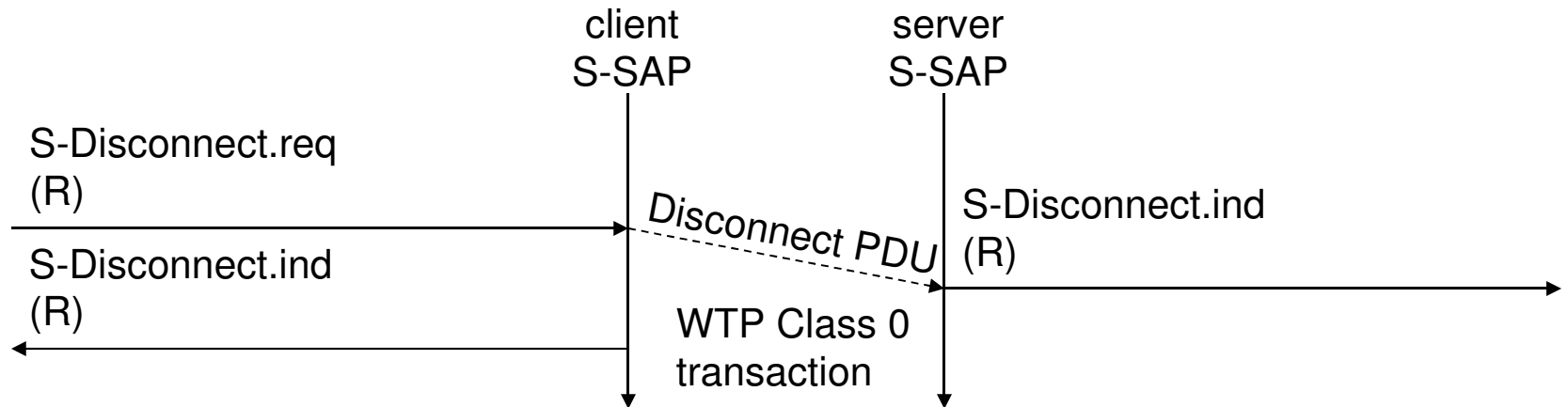
WSP/B session establishment



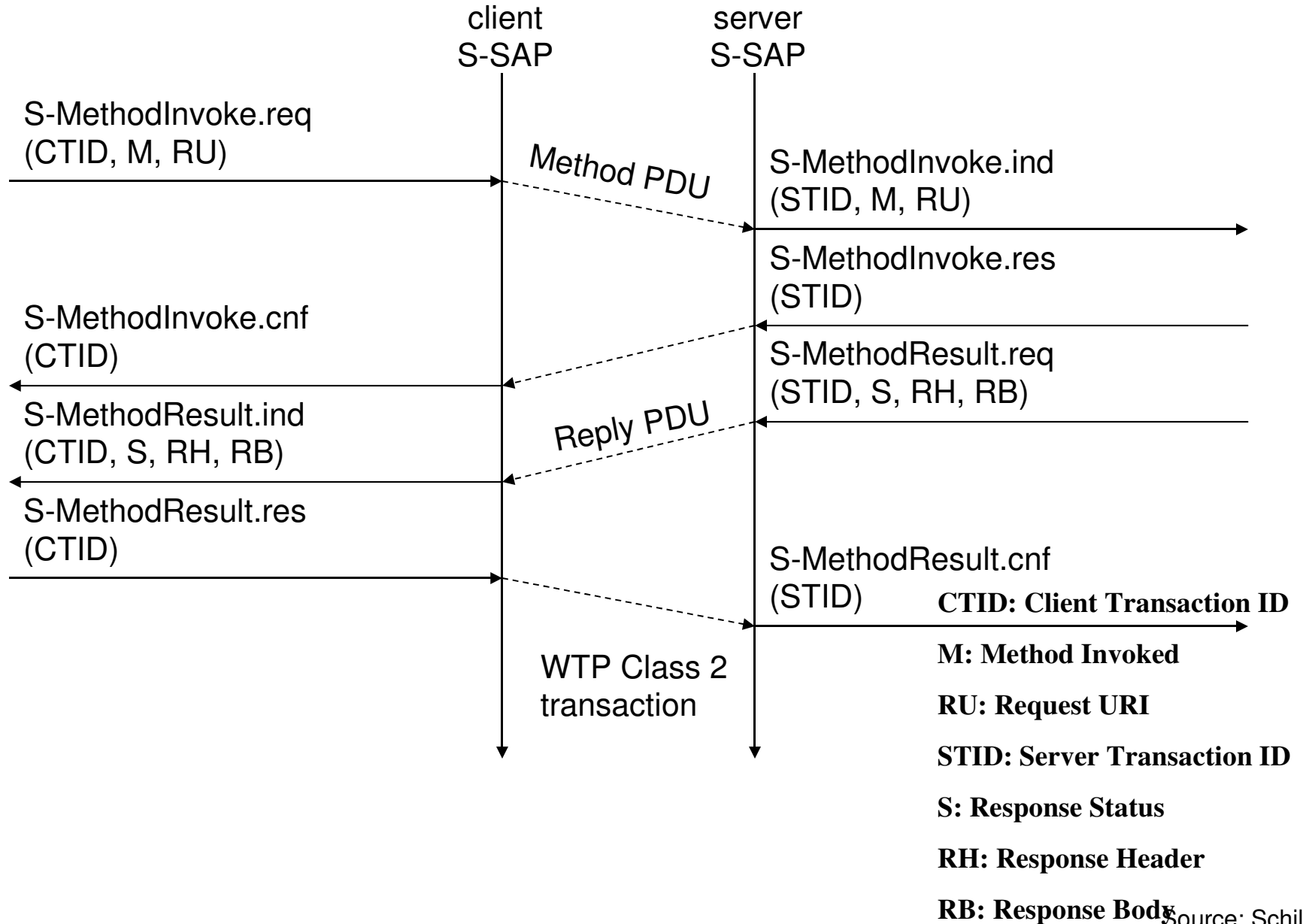
WSP/B session suspend/resume



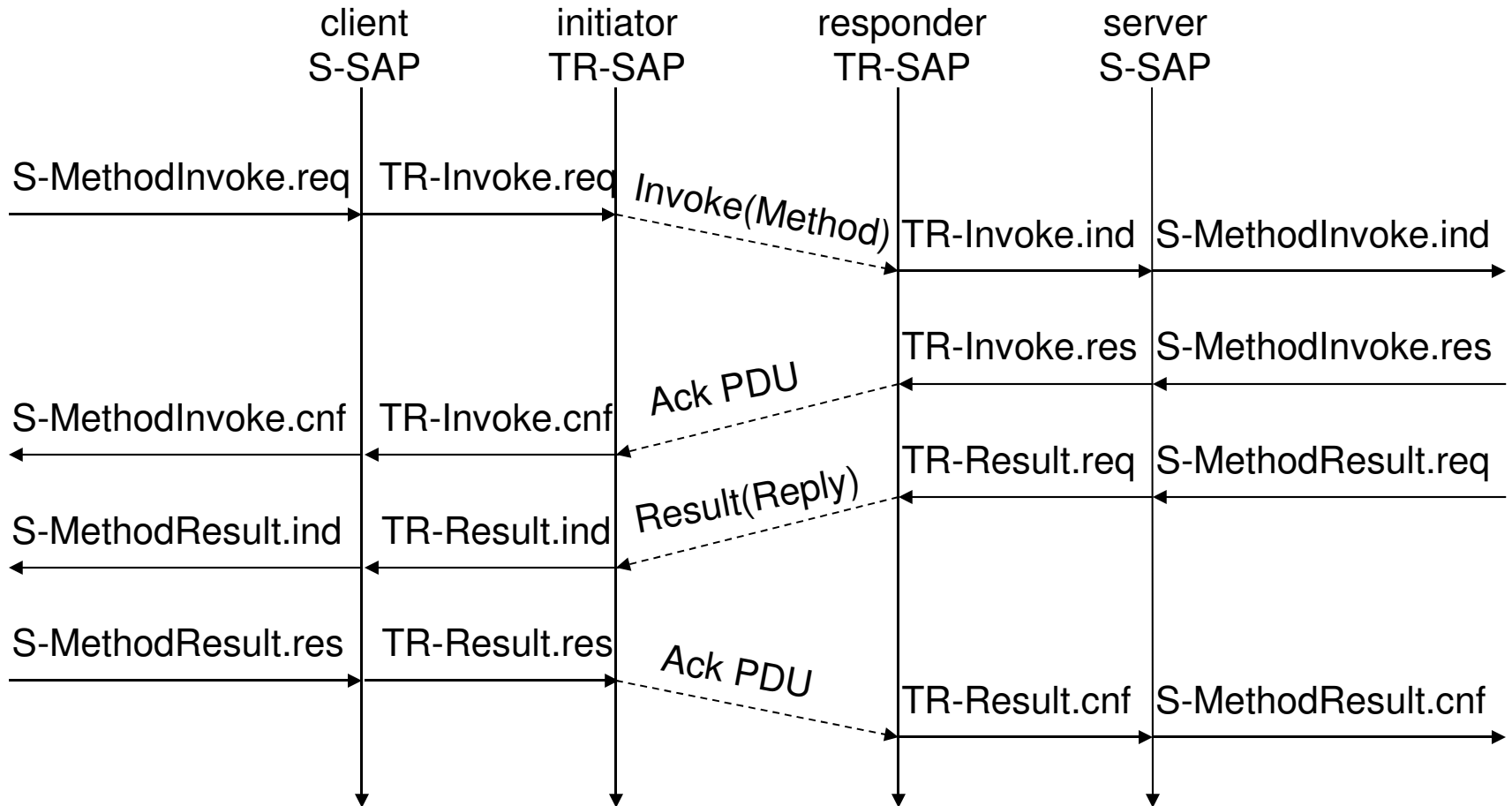
WSP/B session termination



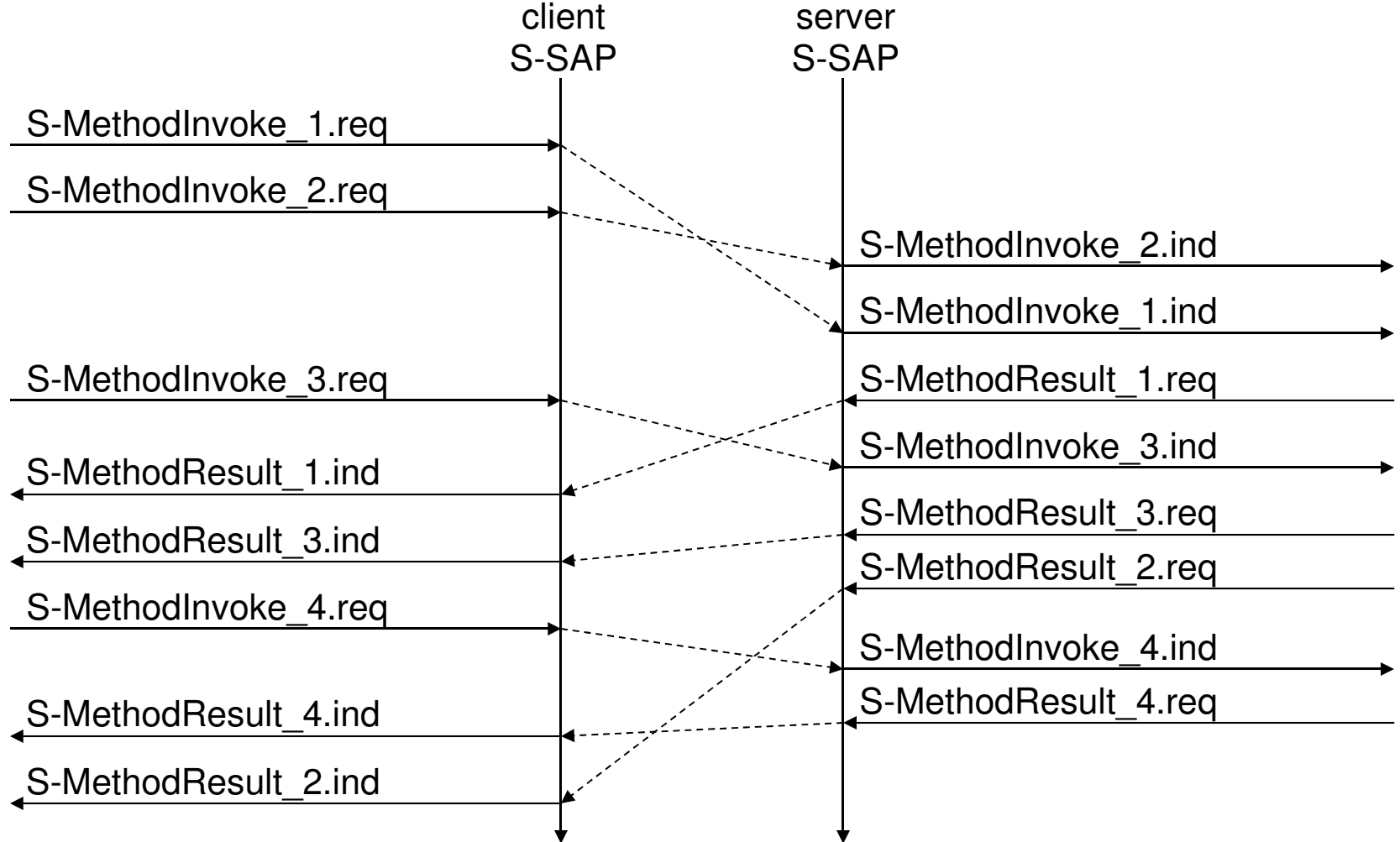
WSP/B method invoke



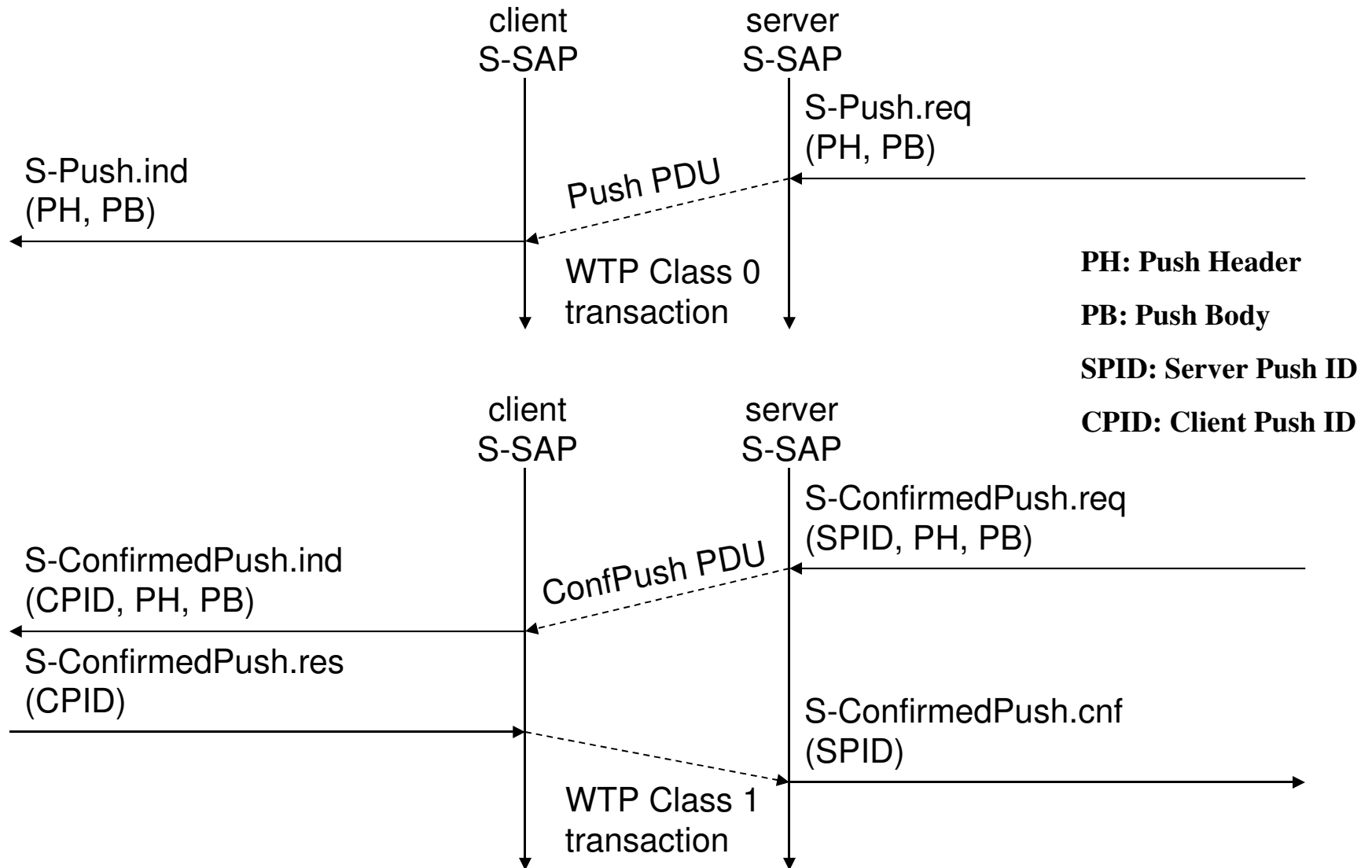
WSP/B over WTP - method invocation



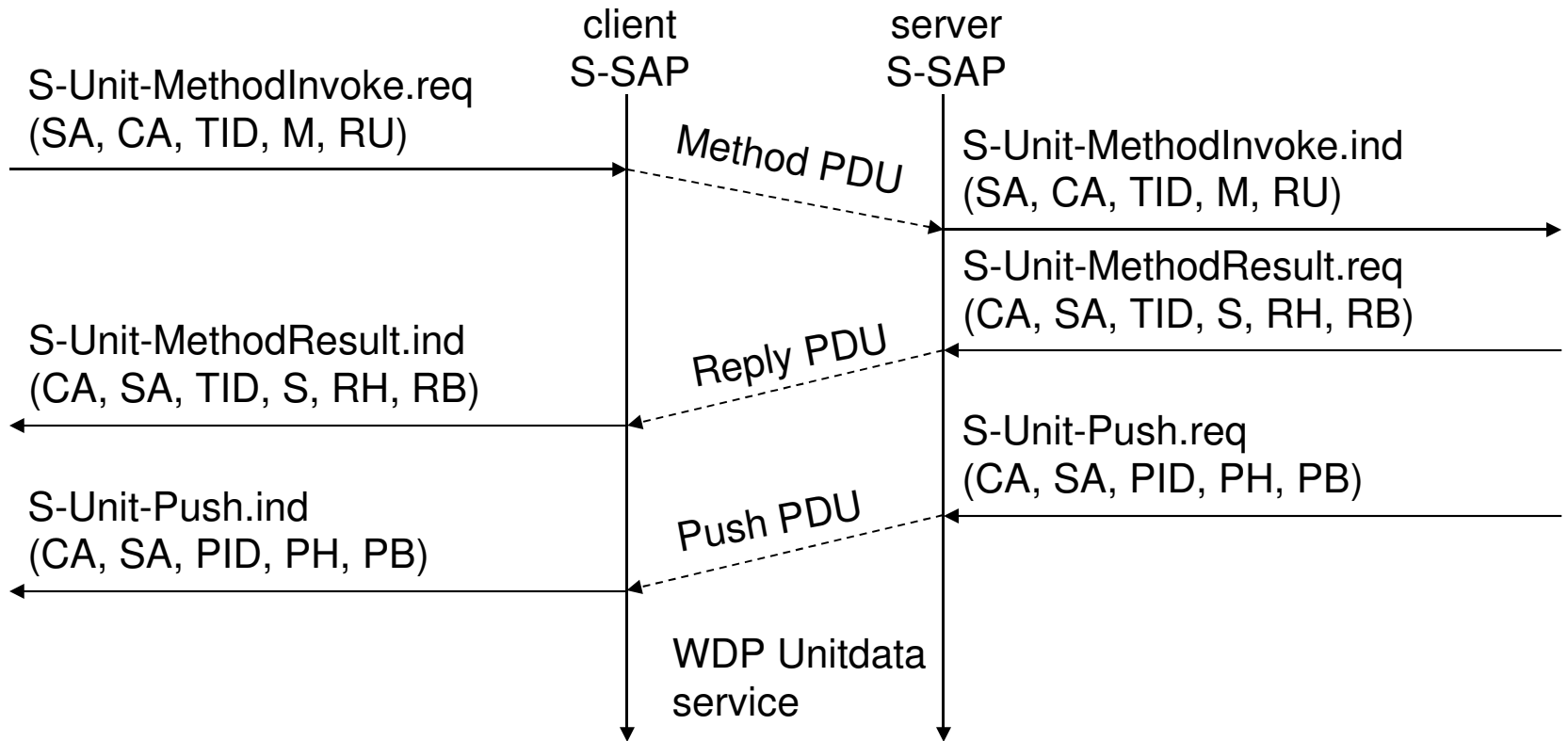
WSP/B over WTP - asynchronous, unordered requests



WSP/B - confirmed/non-confirmed push



WSP/B over WDP



WAP Stack Summary

- **WDP**
 - functionality similar to UDP in IP networks
- **WTLS**
 - functionality similar to SSL/TLS (optimized for wireless)
- **WTP**
 - Class 0: analogous to UDP
 - Class 1: analogous to TCP (without connection setup overheads)
 - Class 2: analogous to RPC (optimized for wireless)
 - features of “user acknowledgement”, “hold on”
- **WSP**
 - WSP/B: analogous to http 1.1 (add features of suspend/resume)
 - method: analogous to RPC/RMI
 - features of asynchronous invocations, push (confirmed/unconfirmed)

WAP: Ongoing Work

- WDP
 - Tunnel to support WAP where no (end-to-end) IP bearer available
- WTLS
 - support for end-to-end security (extending WTLS endpoint beyond WAP Gateway)
 - interoperable between WAP and Internet (public key infrastructure)
 - integrating Smart Cards for security functions
- WTP
 - efficient transport over wireless links (wireless TCP)
 - bearer selection/switching
 - quality of service definitions
- WSP
 - quality of service parameters
 - multicast data, multimedia support
- WAE
 - User agent profiles: personalize for device characteristics, preferences etc
 - Push architecture, asynchronous applications
 - Billing

WAP: Hype vs Reality

- **Low-bandwidth wireless links**
 - tcp/ip over wireless can also address these problems
 - encoding in http can also reduce data transfer on wireless links
- **Limited device capabilities**
 - Microbrowser is appropriate to address this problem
 - WTAI features are not present in tcp/ip domain
- **Challenges in WAP**
 - adapting to applications rich in content and interaction
 - service guarantees
 - interface design and usability
- **Other approaches for WWW access through mobiles**
 - i-Mode (from NTT DoCoMo)
 - WAP is a TRAP (<http://www.freeprotocols.org/wapTrap>)

References and Resources

- Books
 - Mobile communications: Jochen Schiller, Addison Wesley 2000
 - Understanding WAP:

- Official Website (specifications)
 - www.wapforum.org

- Technical/Developer Info and tools
 - www.palopt.com.au/wap
 - www.wap.net

- Major players
 - www.nokia.com/wap
 - www.ericsson.se/wap
 - phone.com

- OpenSource effort
 - www.wapgateway.org (Kannel WAP gateway project)

Thank You

This presentation is available online from

<http://www.it.iitb.ernet.in/~sri/talks>

Sridhar Iyer

KR School of Information Technology

IIT Bombay