Wireless Application Protocol (WAP)

&

iMode

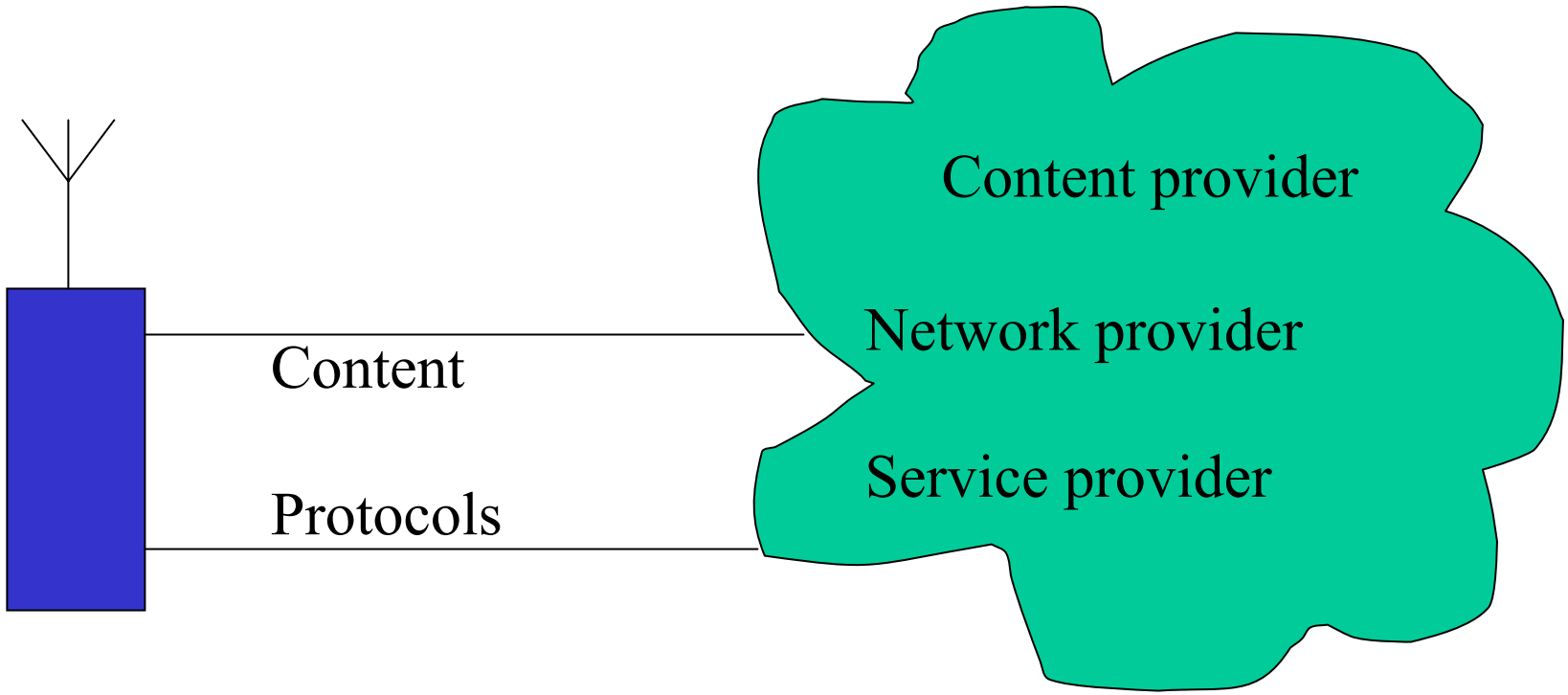
Issues seen by a wireless application user

- Network characteristics
  - Delay: long, variable (Ex.: GSM/800 ms RTT)
    - Packets arrive out of sequence ➔ buffering + reassembly
    - TCP triggers retransmissions ➔ more retransmissions
  - BW: GSM/9.6 Kbps, EDGE/384 Kbps, GPRS/115 Kbps, CDMA2000 1xEVDO/2 Mbps ➔ Application awareness
- Device constraints
  - Memory, CPU power, battery, display
- Service requirements
  - Huge user base ➔ issue of scalability (in real-time apps.)
  - Roaming ➔ Network route changes, but application and source of info don't change ➔ Applications may become useless ➔ Environment awareness

Current web technologies for wireless applications?

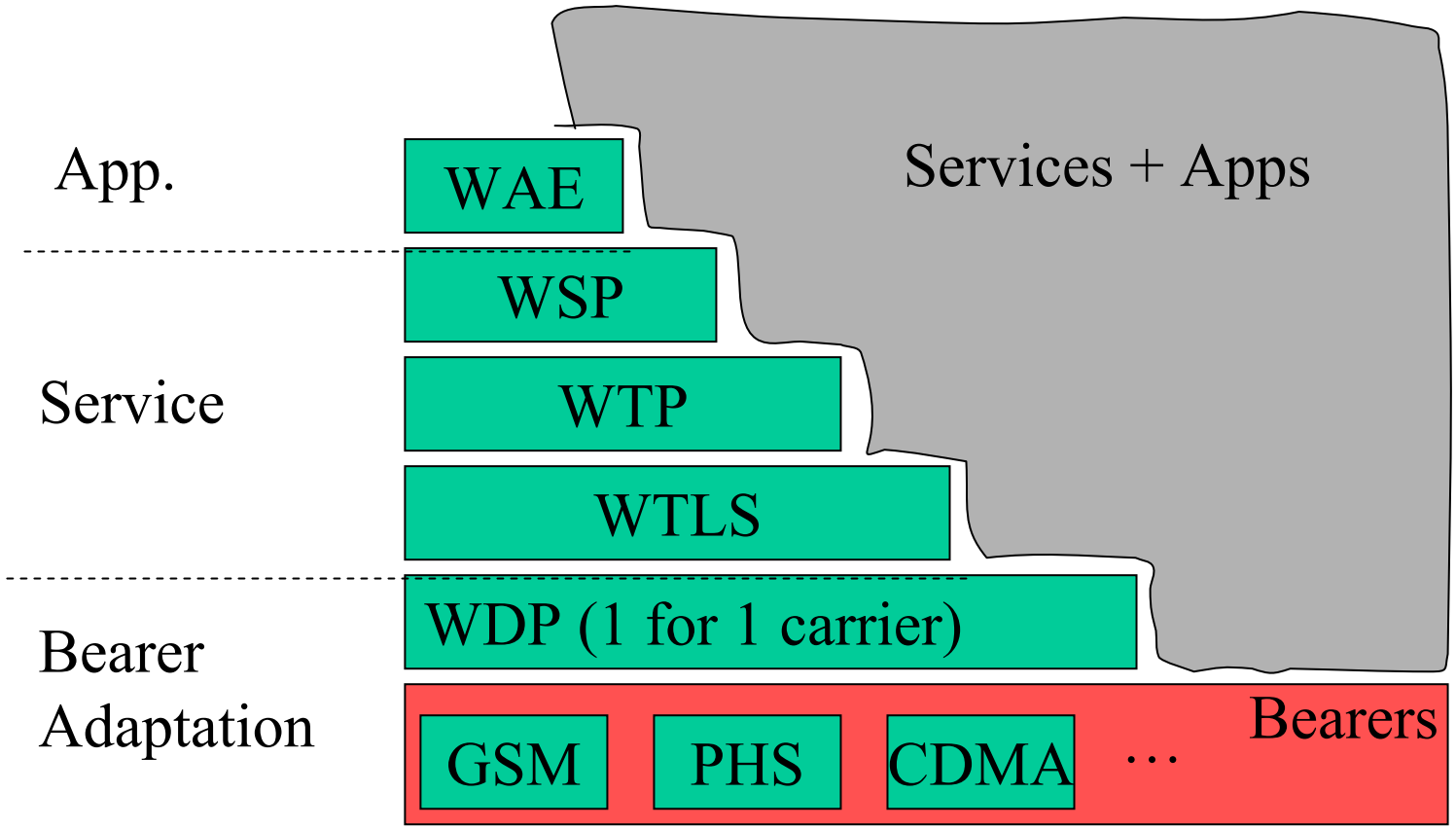
- HTTP (Hypertext Transfer Protocol)
  - High overhead: one network session for each “element”
- HTML: Focus on rendering
  - Most doc in HTML
  - 80+ tags: forms, graphics, lists, links, tables…
  - **Rendering is expensive**
- Java
  - Synonymous with Web
  - Difficult to implement in small wireless devices

WAP standard



WAP standard—how handsets communicate over wireless net and how contents+services are delivered.

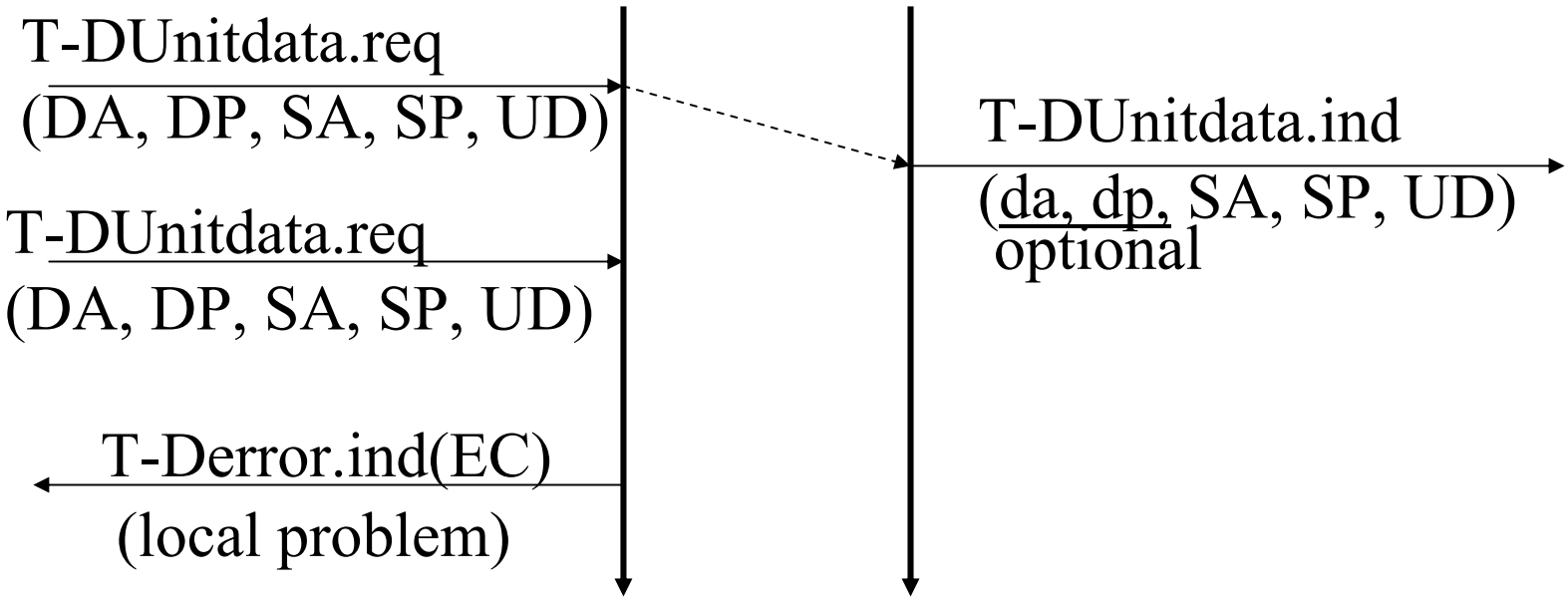
WAP architecture



Components of WAP standard

- Bearer Adaptation
  - Different wireless access ➔ need for common access abstraction
  - Abstraction is achieved by Wireless Datagram Protocol
    - » Point-to-point
    - » No guarantee of quality: reliability, security, timeliness
- Service Protocols (higher-level services)
  - Security and reliability
- Application Environment
  - Browser-based environment supporting content and application portability across devices

1. Bear Adaptation (WDP)



Destination Address ex.: MSISDN (GSM), IP addr

WCMP: destination unreachable
parameter problem (header)
packet too big, …

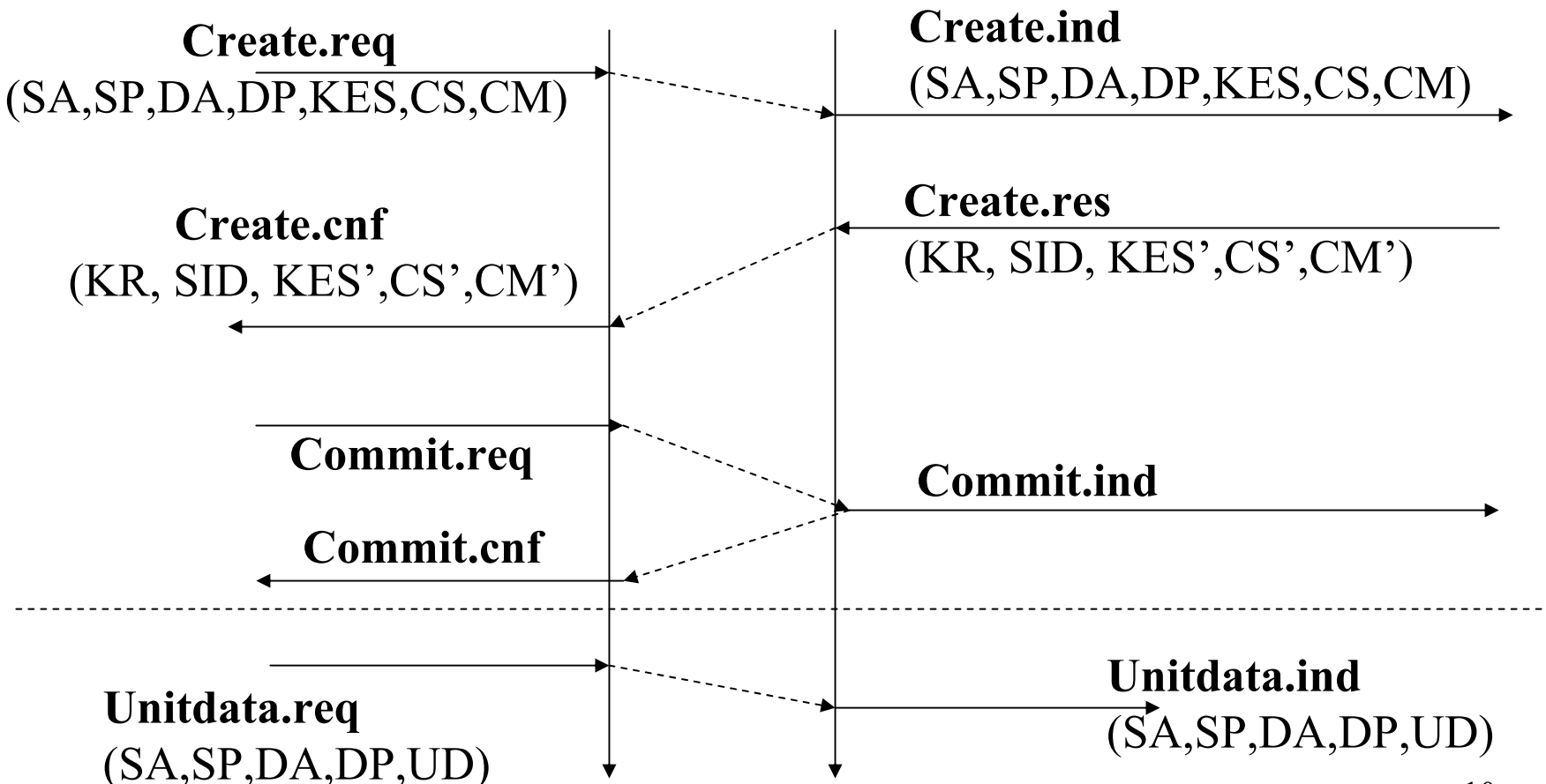
2. Service Protocols

- Desired **services** for application dev.
  - Reliability
  - Ordering
  - Timeliness
  - Security
- Protocols used
  - Wireless Transport Layer Security (WTLS)
  - Wireless Transaction Protocol (WTP)
  - Wireless Session Protocol

Wireless Transport Layer Security (WTLS)

- Supports
  - **authentication** using client and server **certificates**
  - **encryption**
- Consumes
  - computation and bandwidth
- Bottom position
  - quick identification and elimination
- Optional

WTLS establishing a **secure session**



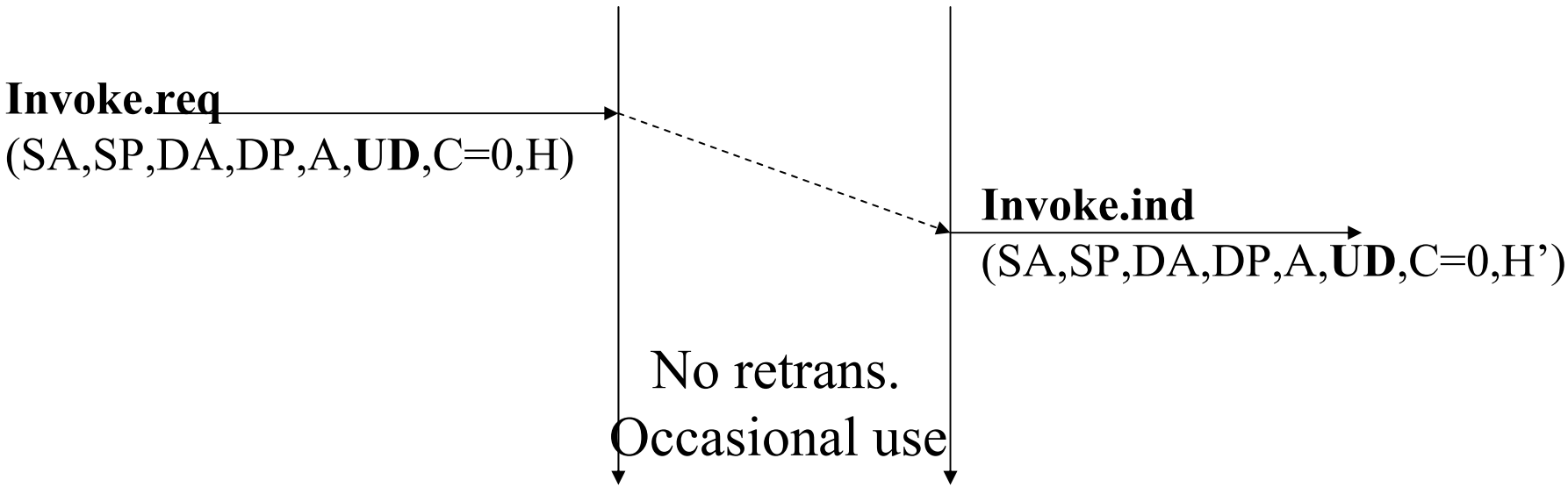
Secure session

- KES: Key Exchange Suite (RSA, Diffie/Hellman, ..)
- CS: Cipher Suite (DES, AES, …)
- CM: Compression Method (currently not specified)
- SNM: Sequence Number Mode
- KR: Key Refresh cycle

Wireless Transaction Protocol

- Introduces the notion of a **transaction:**
  - Explicit pairing between client request and server response
- Supports **reliable** client/server message exchange
  - Duplicate removal and Retransmission
  - Confirmation of reception (ACK)
- User-level ACK (coming from WTP user)
- Minimizes bandwidth consumption
    - » **Piggybacking** of fresh data with ACK sent
    - » **Delayed transmission** (hope for concatenation)
- Classes of WTP: for different quality and cost
  - Class 0, 1, 2

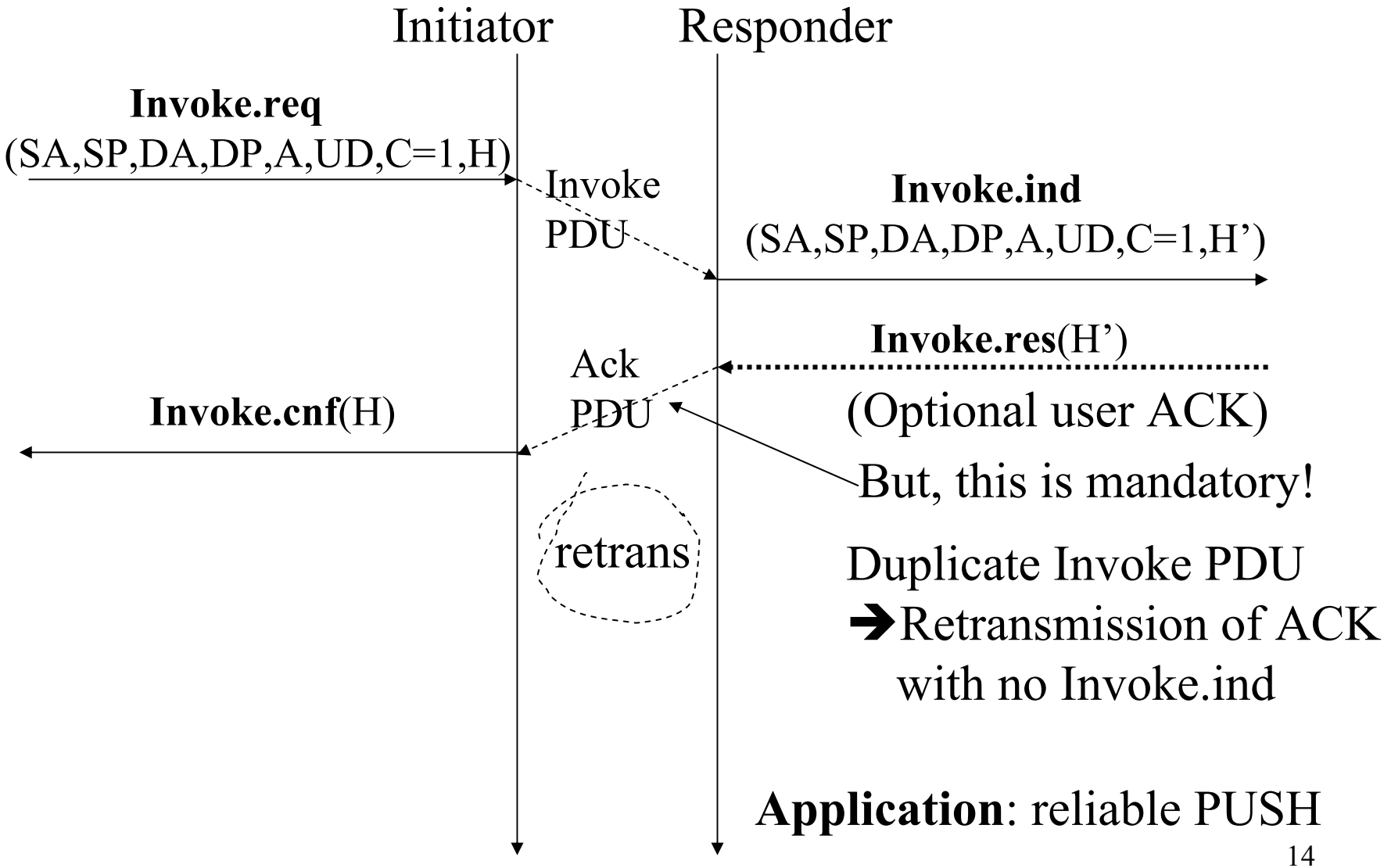
WTP: Class 0



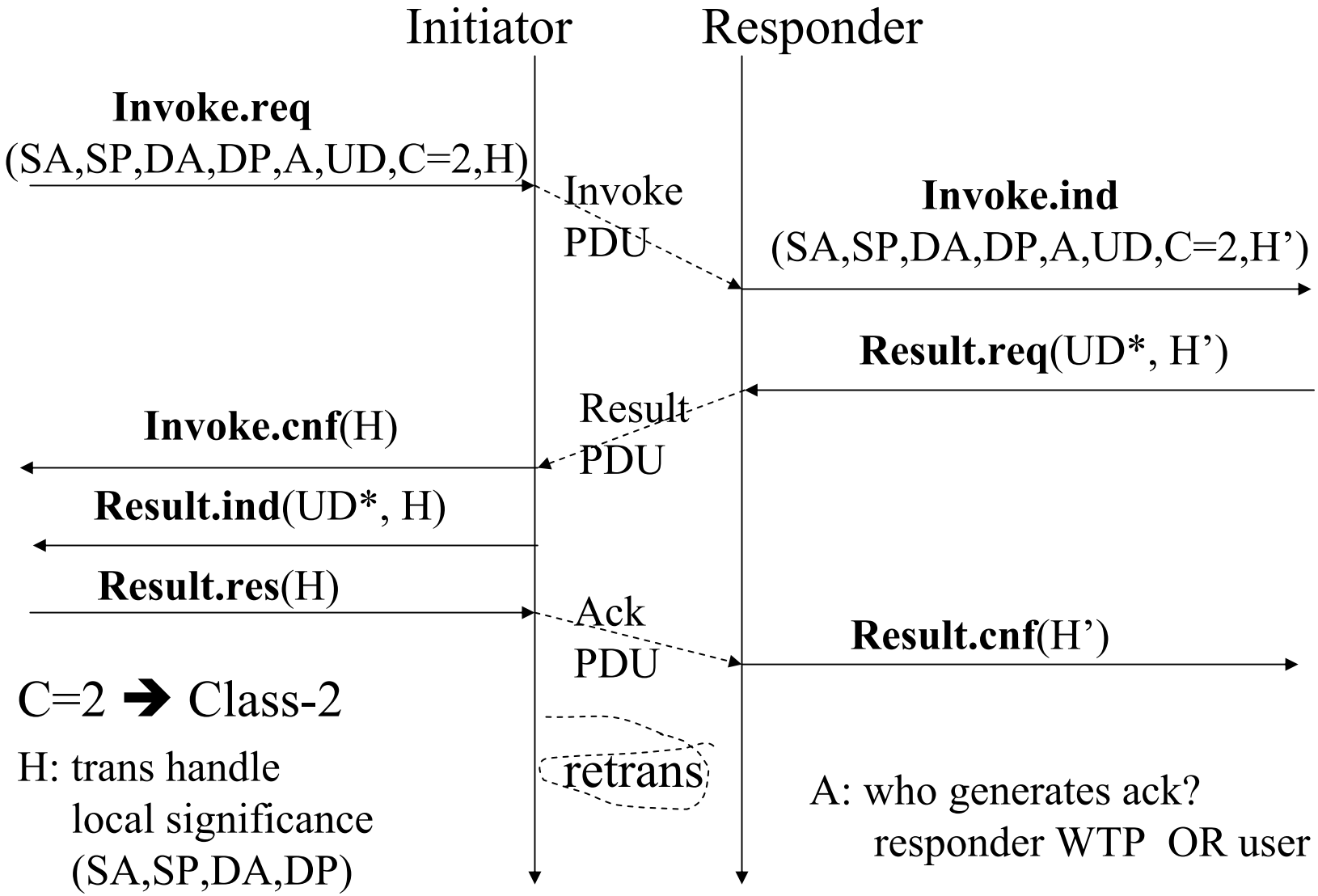
H: an alias for
(SA,SP,DA,DP)

A: Who generates ACK--WTP entity or user (not used here)

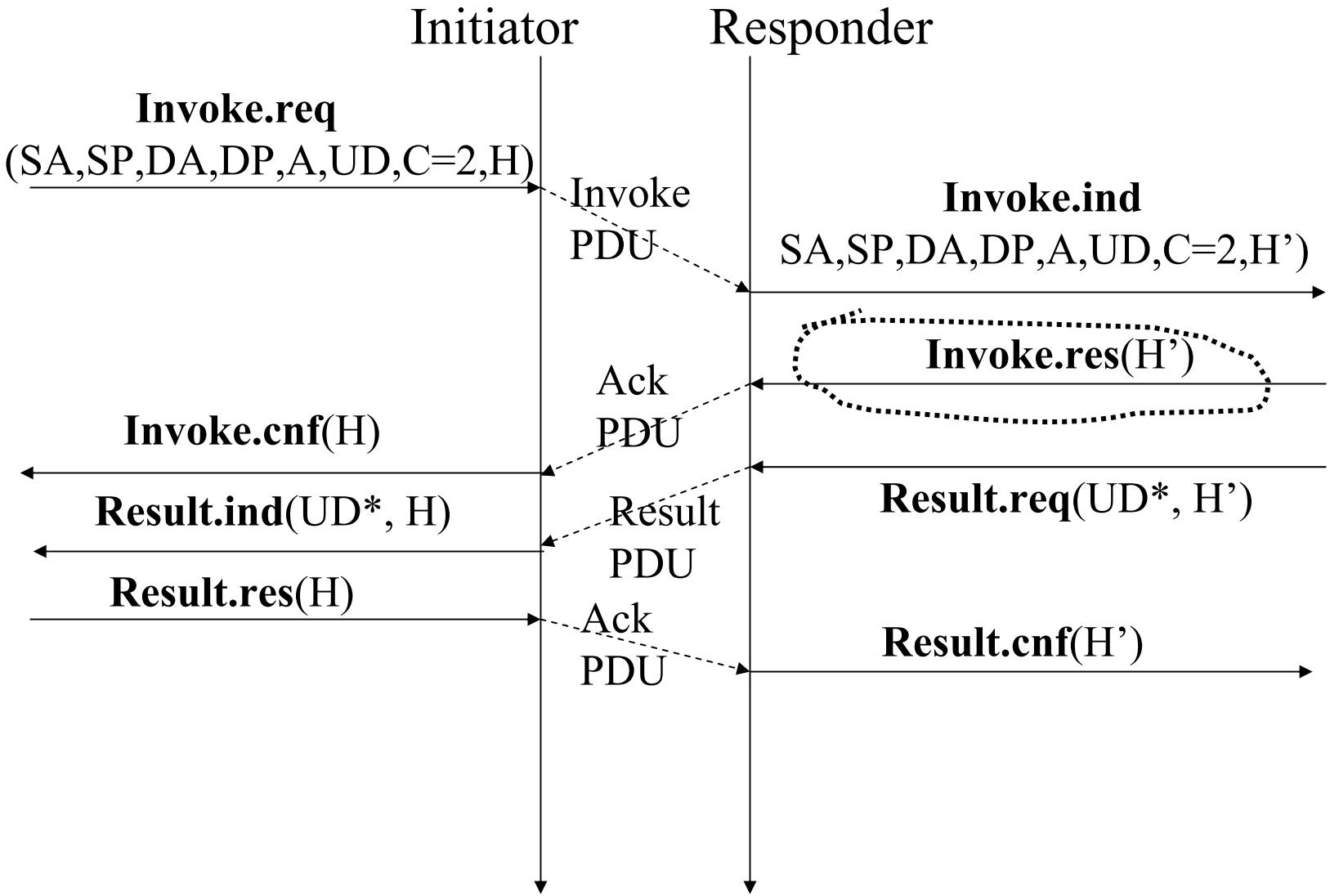
WTP **class-1 (reliable, but no result msg)**



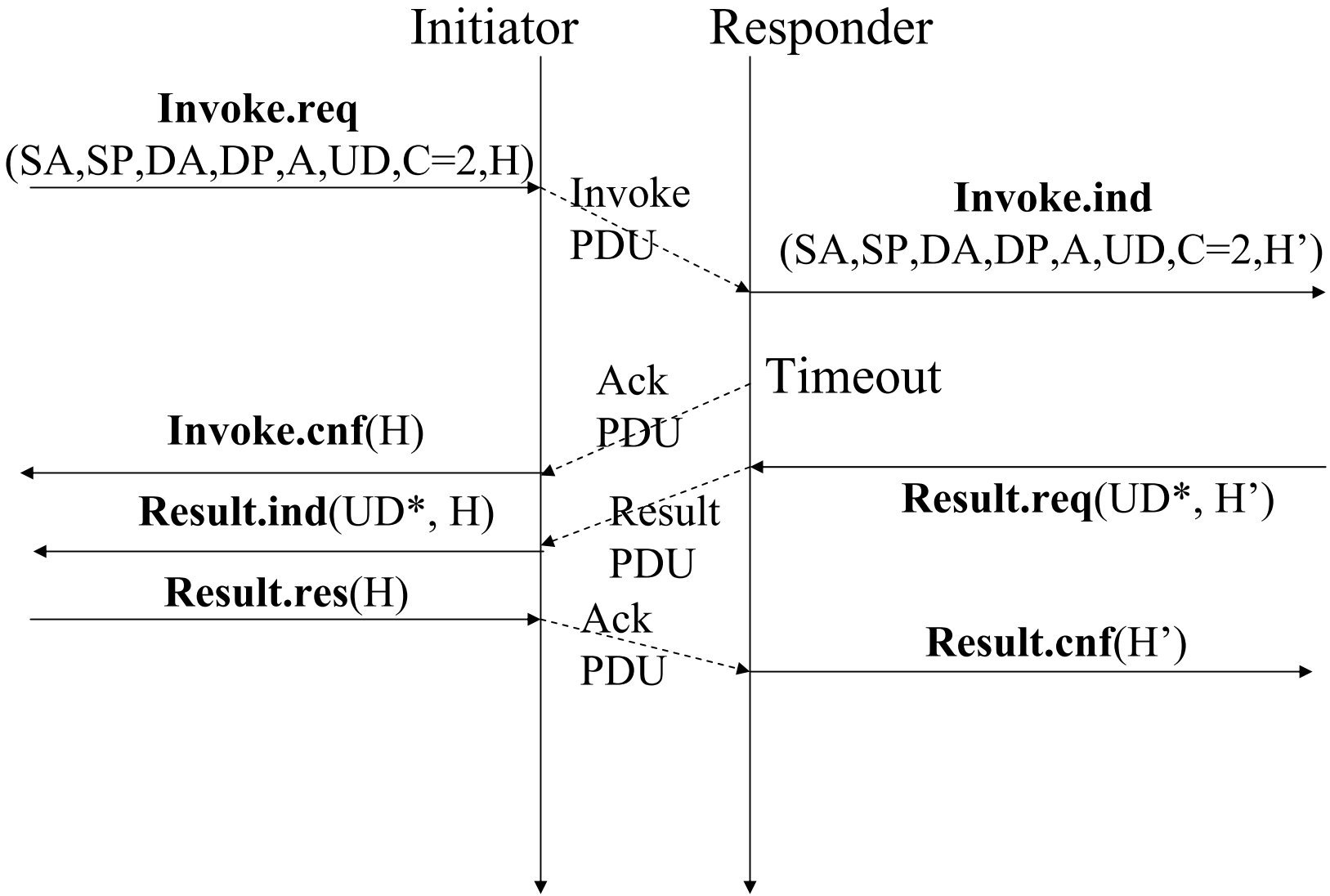
WTP **class-2 (reliable req/res)**



WTP **class-2 (user Ack)**



WTP **class-2 (“hold on”, reduce retrans.)**



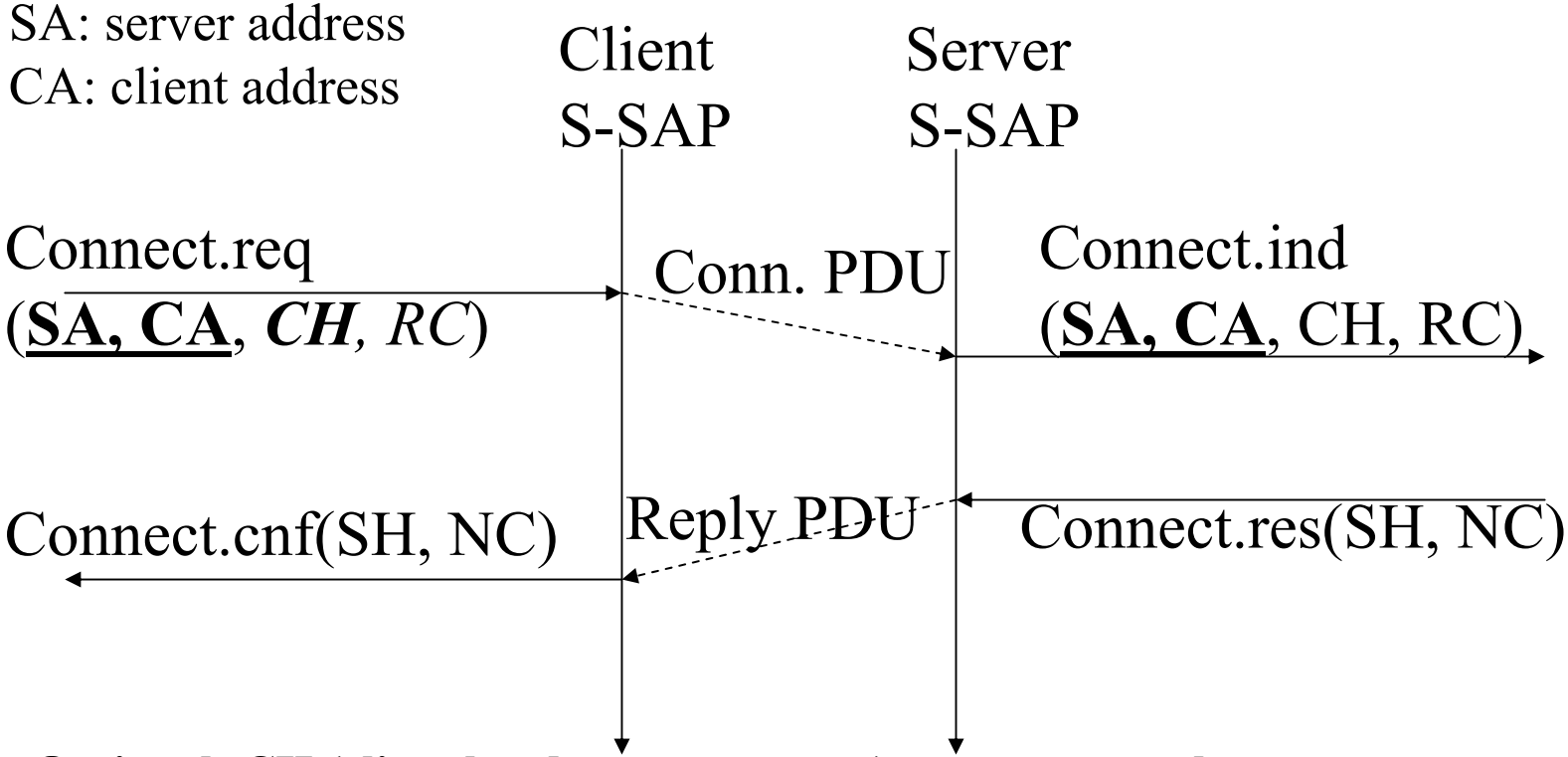
Wireless **Session** Protocol (WSP/Browsing)

- **Session**: **lifetime** of a user's interaction with content **servers**.
  - Suspend/resume sessions: Ability to continue from where you left.
  - Capability neg.: max SDU (client/server) size, max outstanding req.
- Provides a **shared state** between client and server
  - To optimize content transfer
  - To cater to the special need of mobile users
    - » Network interruption
    - » Customized environment

WSP/B: Establish a session

(using Class-2 WTP)

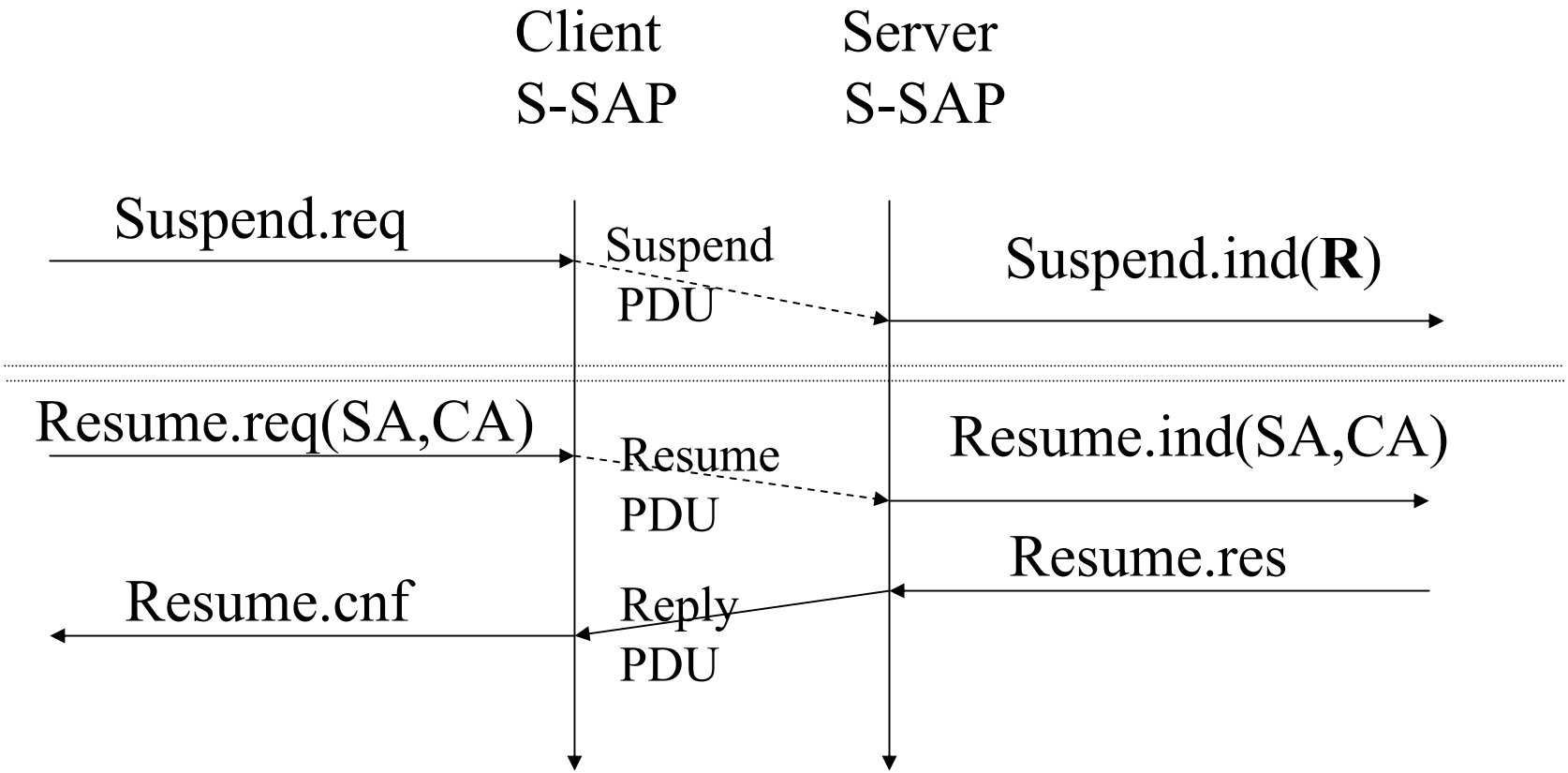
SA: server address
CA: client address



Optional: CH (client header, user-to-user): content type, languages, device capabilities

RC: requested capabilities (confirmed push with session, non-confirmed push with session, push with no session, suspend/resume)

WSP/B (**suspend/resume**)

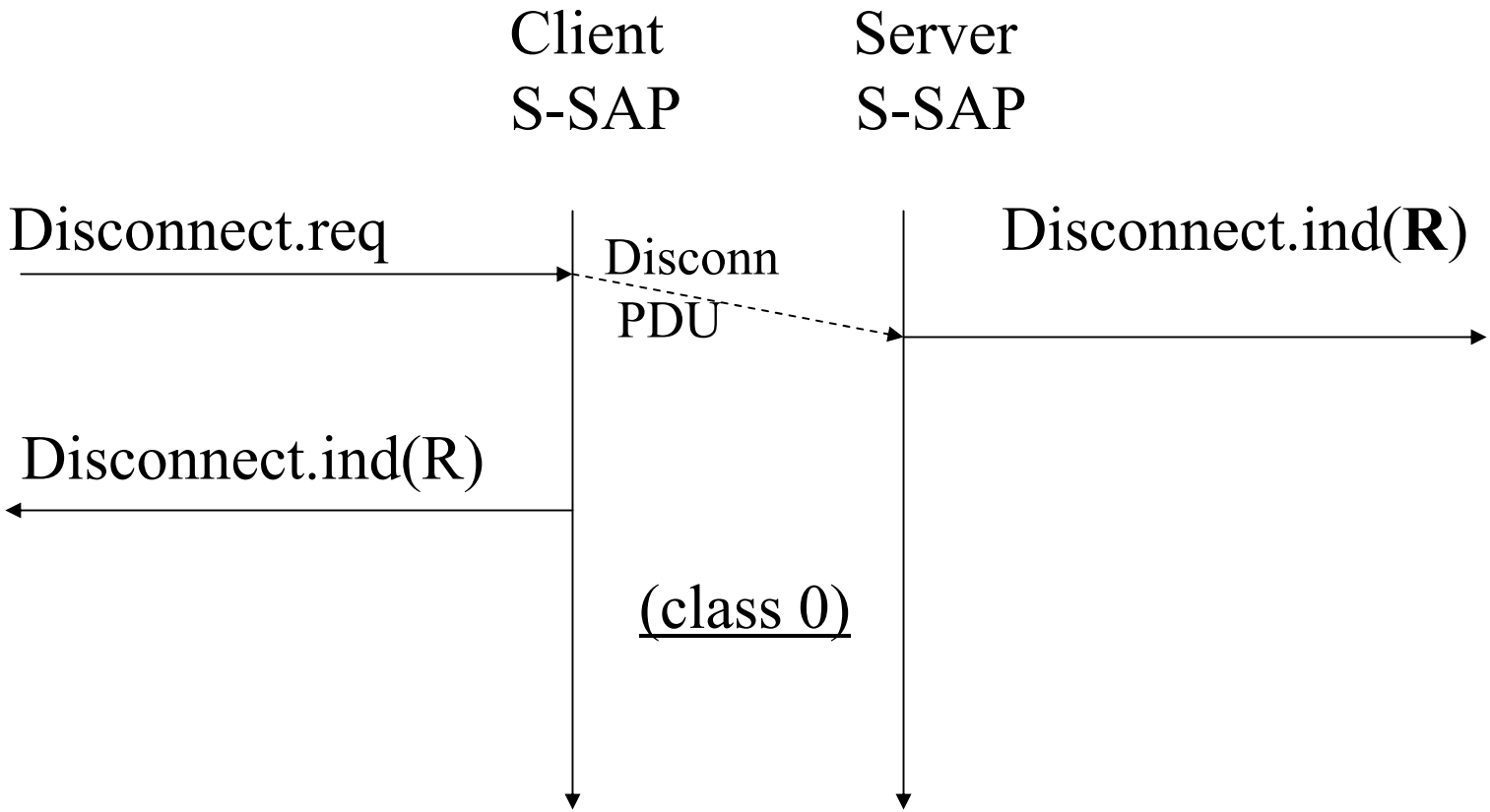


Suspension: abort data trans., freeze state

Class 0: unconfirmed and unreliable

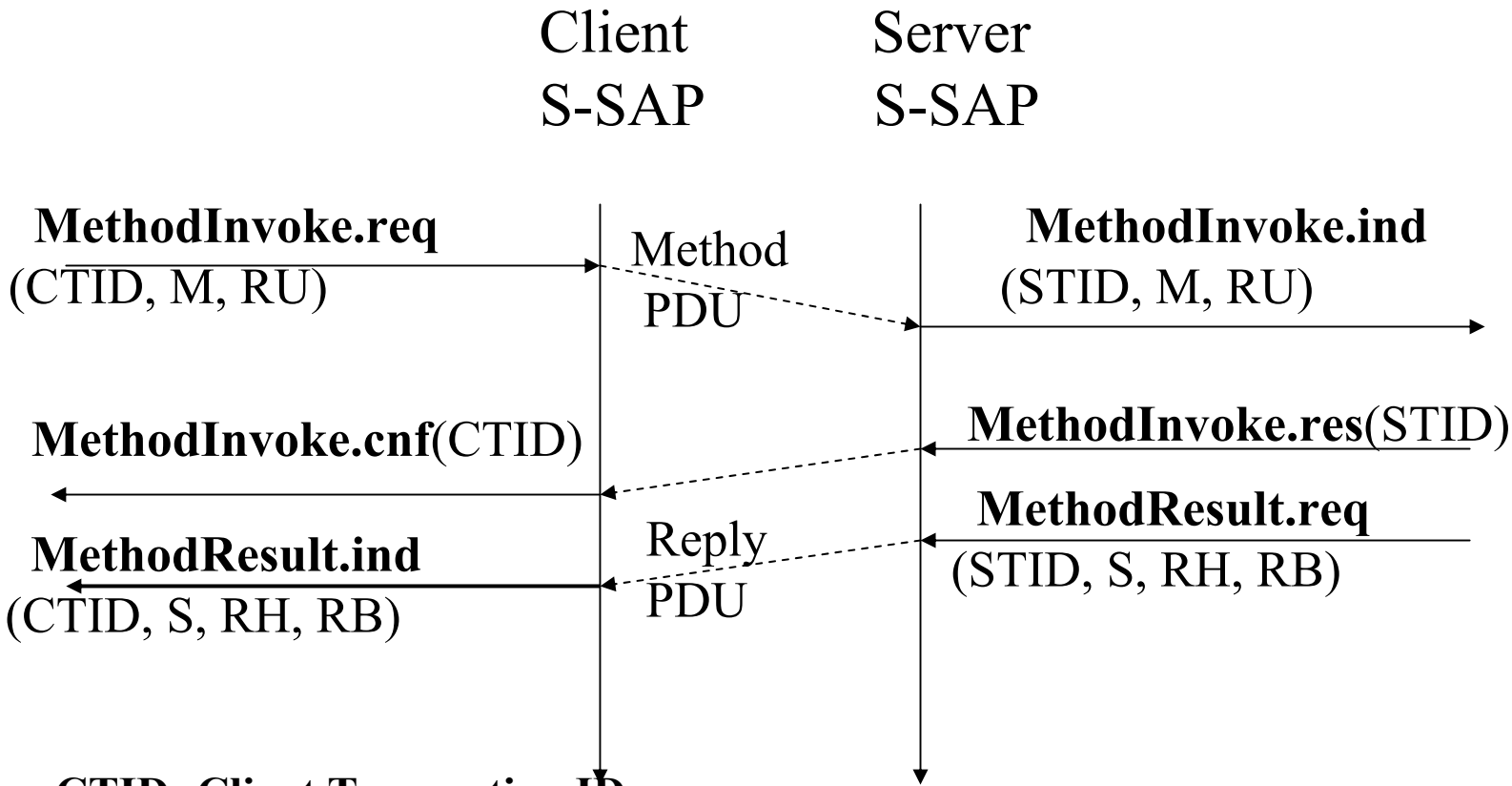
R : reason (service provider/user)

WSP/B (**disconnect**)



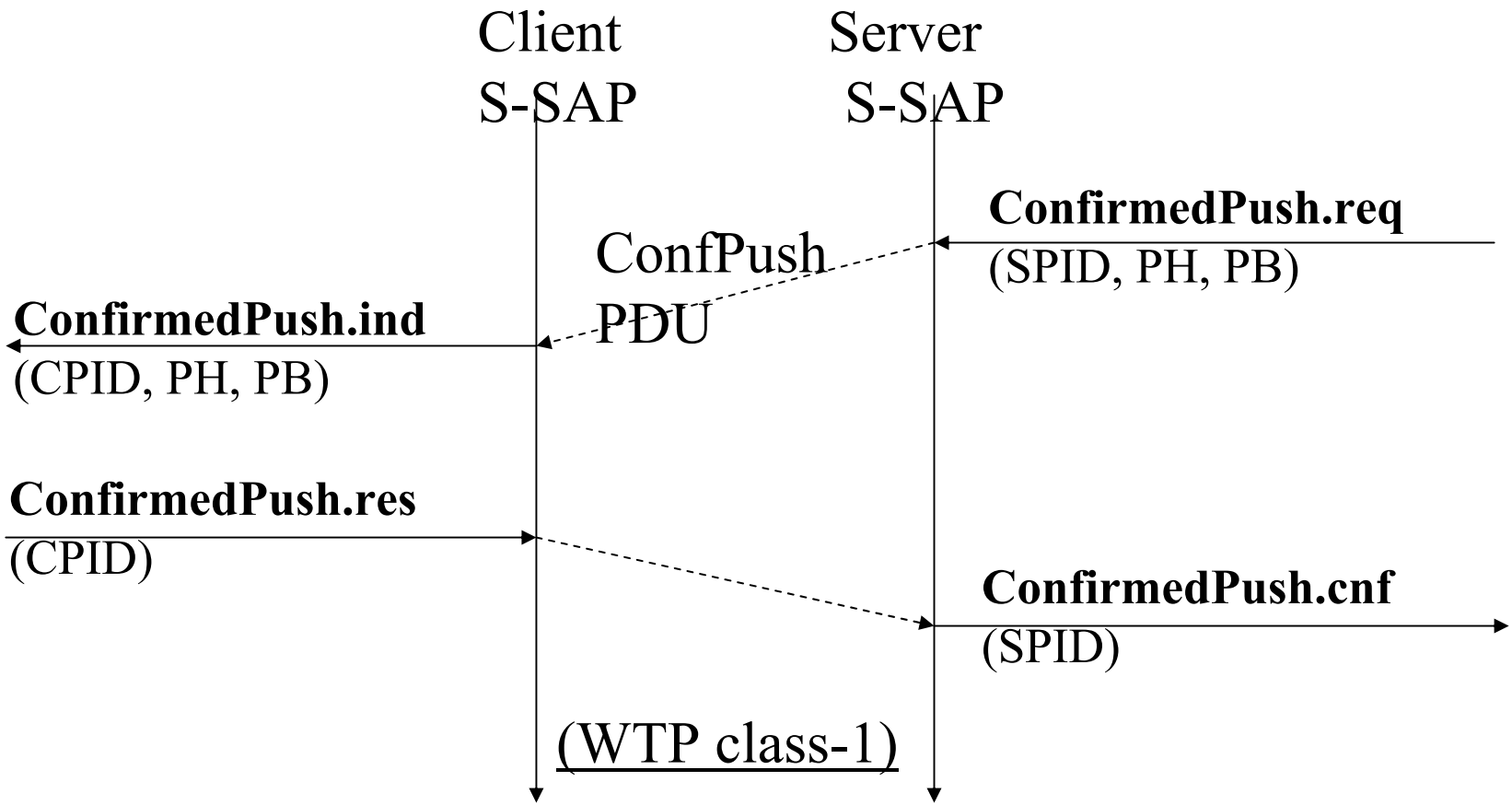
R: network error, peer request, congestion, …

WSP/B (**request an operation**)



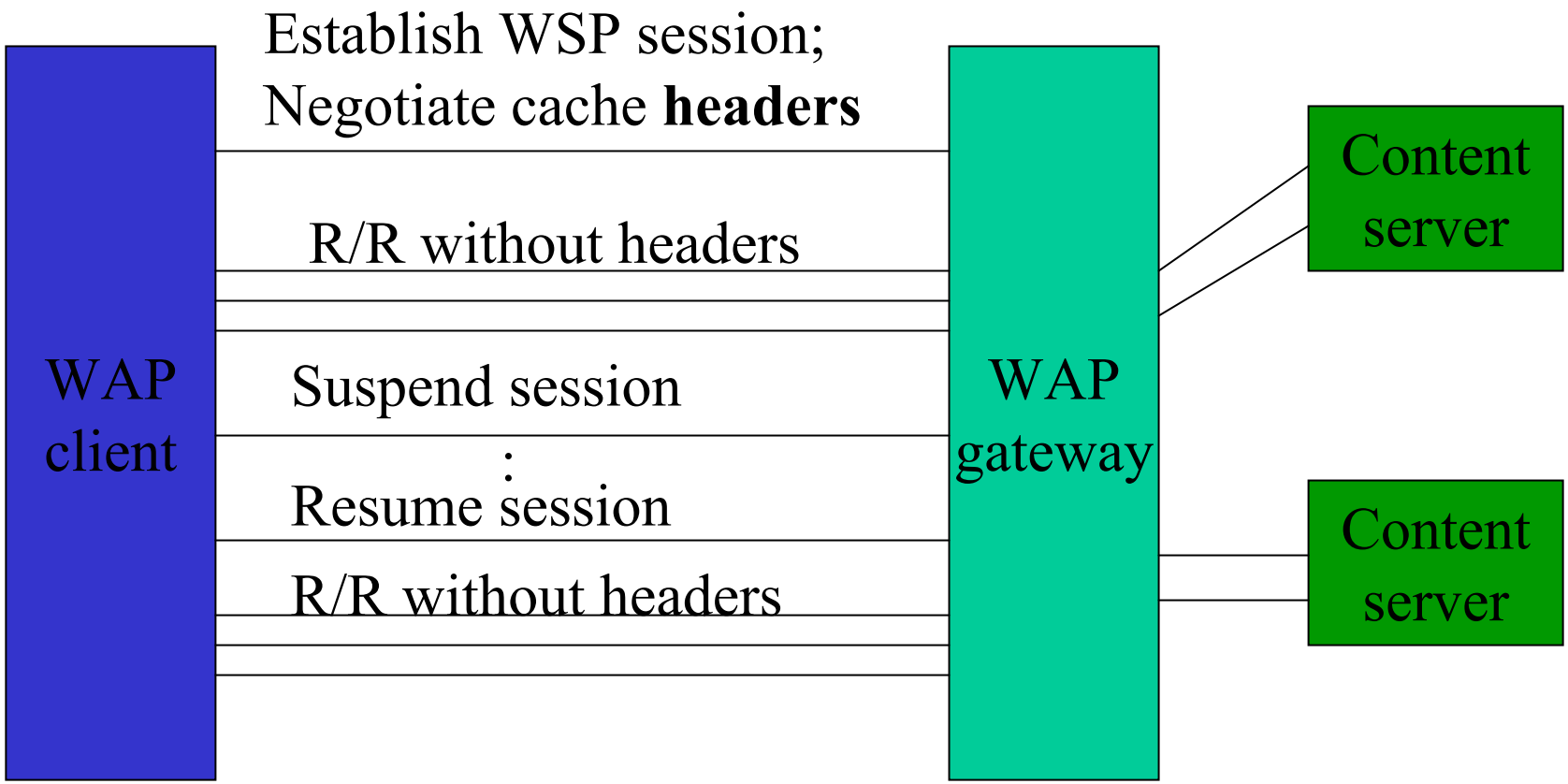
CTID: Client Transaction ID
RU: Request URI (Identifier) ➔ URLs
M : Method ➔ requested operation at server (get/post)
S : Status (server not found: 404)
RH/RB: Response header/body

WSP/B (confirmed **push**)



(Similarly, you can have an unconfirmed push that uses WTP class-0 service.)

Long-lived session between a WAP client and a WAP gateway



Add cached headers and send HTTP req. →

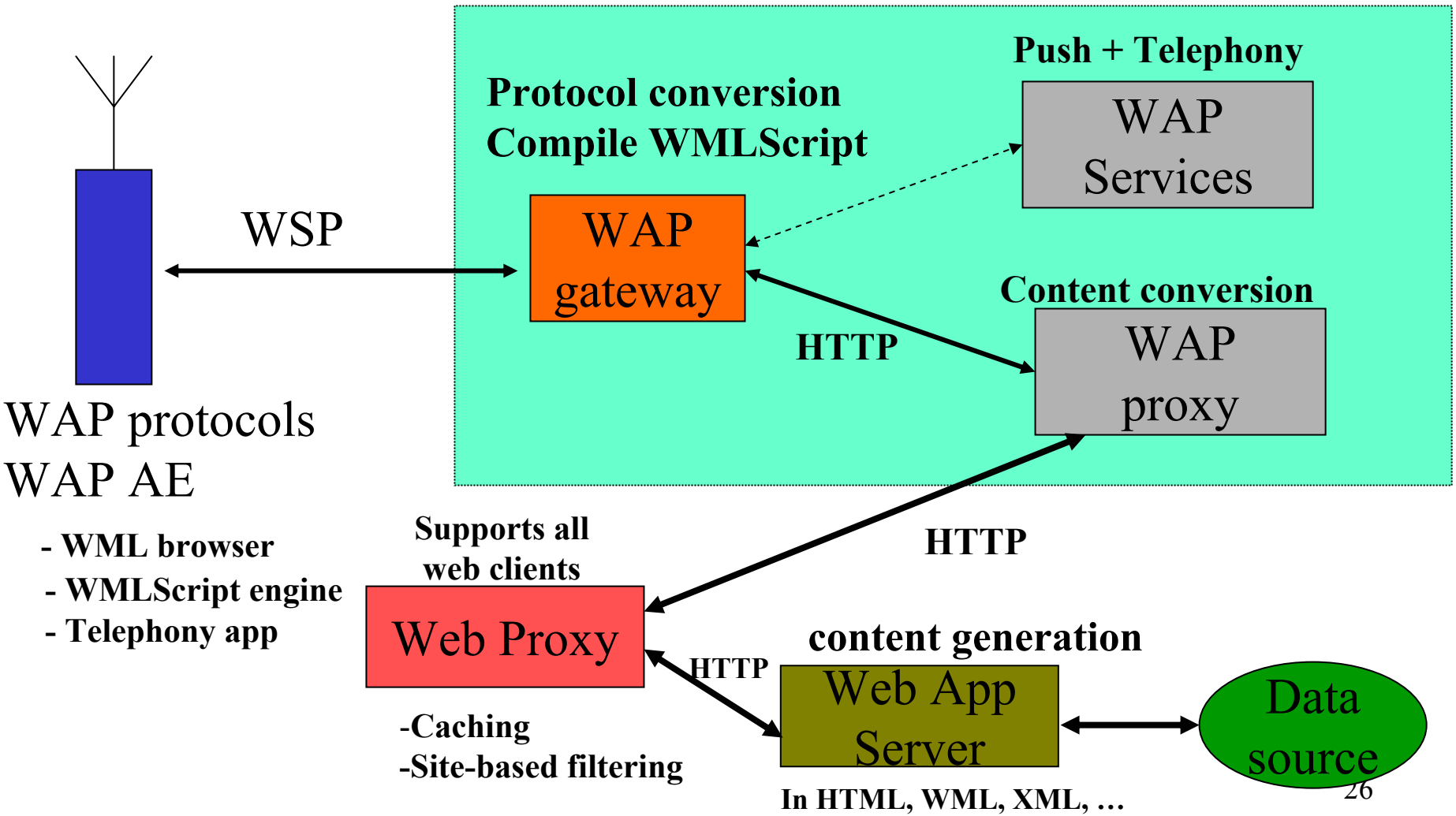
Remove cached headers and send WSP resp. ←

3. Application Environment

- Main subsystems
  - Network infrastructure
  - Content format, user interactions
    - Format of content (WML)
    - Format of downloadable client logic (WMLScript)
      - Executed within the client browser
      - Check validity of user input
      - App-controlled selecting and navigating of URLs
      - Accessing facilities on device (telephony)

Infrastructure Supporting WAP Clients

(a logical view of deployment chain)



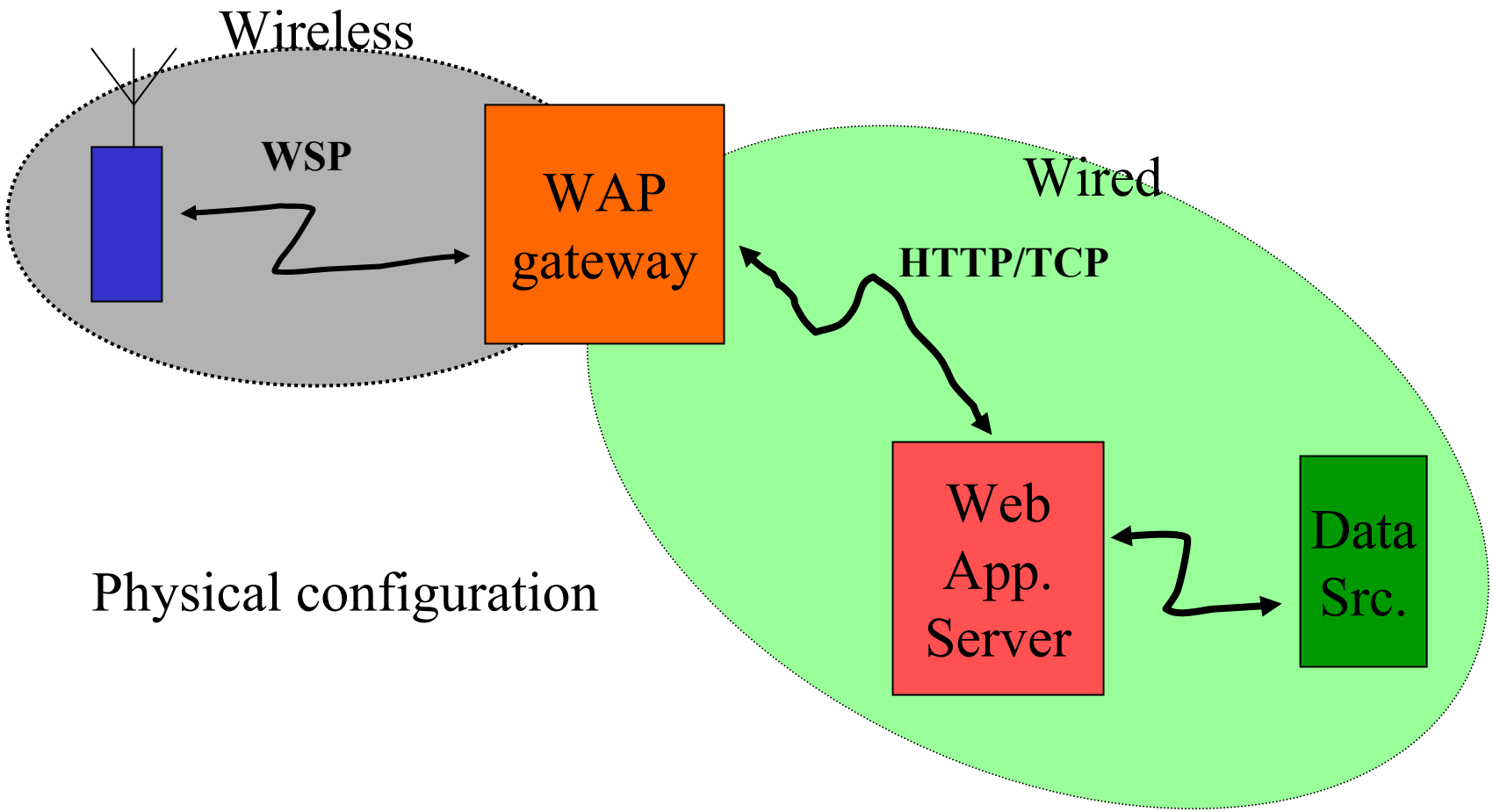
Network Infrastructure …

- WAP client
  - Executes WAP AE: WML browser, WMLScript engine, push client, telephony app environment
- WAP gateway
  - Bridge: wireless network protocol ←→ wireline network
  - WSP ←→HTTP requests/responses
  - Compiles WMLScript into byte code representation
- WAP proxy
  - Content filtering, etc.

Network Infrastructure …

- Various WAP services
  - Capabilities not supported by existing Web infrastructure
  - Telephony server
- Web proxy
  - Content
- Web server
  - Generates content in a form that can be delivered to client
  - Generates WML, HTML, XML, WMLScript, …
- Data source
  - Data repository (IBM's DB2, …)

End-to-end WAP application



Physical configuration

User Agents

- A client may host several **user agents**
  - **WML browser (XML + WMLScript)**
  - **WTA (wireless telephony app)**
  - **(more …)**

WTA (extending basic WAE)

- Content push
  - Server pushes WML decks and WMLScript
  - To enable client to handle new network events
- Handling of network events
  - Incoming events: call, messages
- Access to library functions from WML/WMLScript
  - » **Call Control: set up, accept, release calls**
  - » **Network text: send, read, delete messages**
  - » **Phonebook: read, write, delete entries**
  - » **Miscellaneous: indicate incoming data, e-mail, fax**

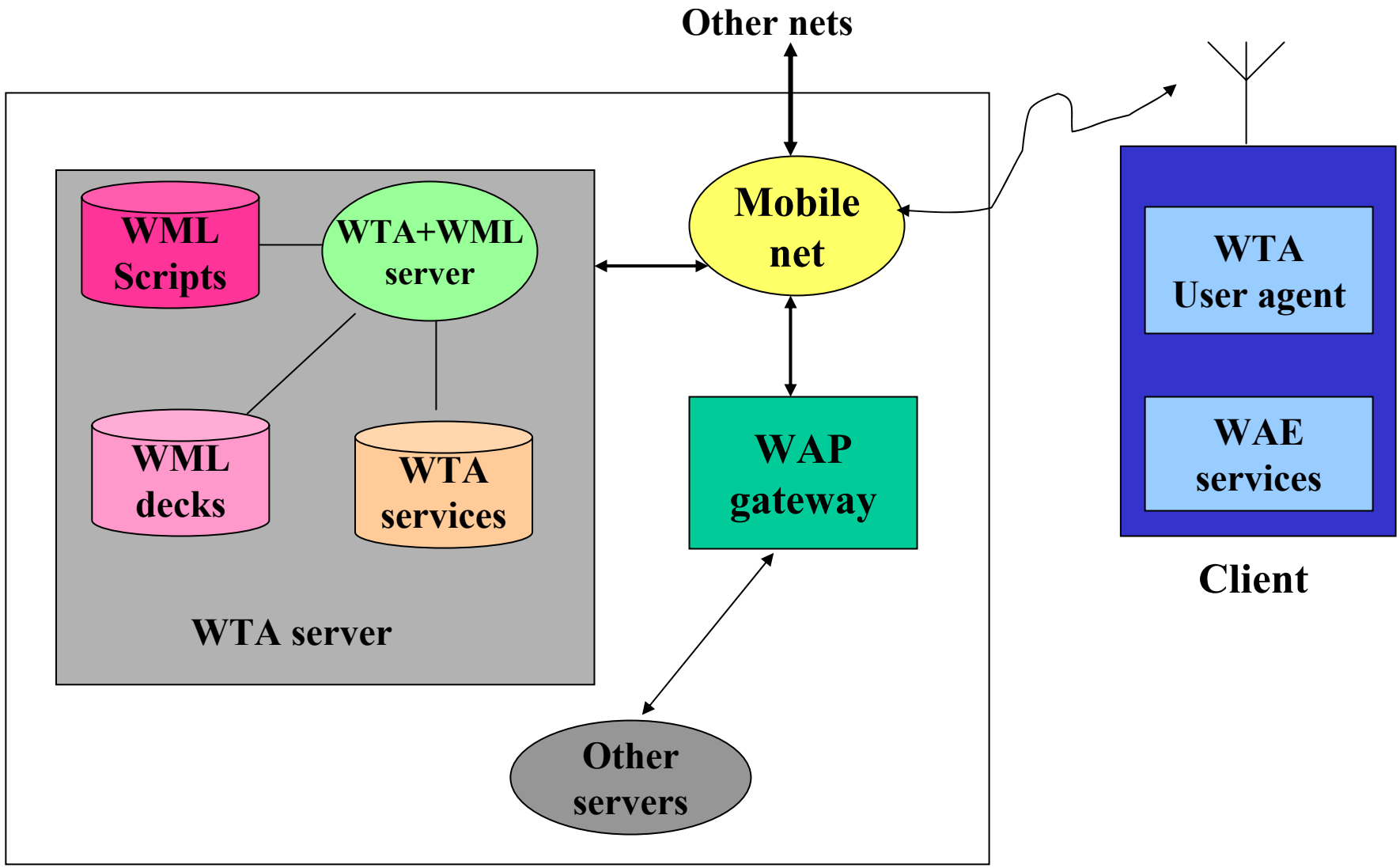
Using WTA libraries

- wtai://<library>/<function>;<parameters>;!<result>
- Library
  - cc: WTACallControl
  - wp: WTAPublic
- Function
  - sc: set up call
  - mc: make call
- In a WML card: **wtai://wp/mc;8884567**
- In WMLScript: WTAIPublic.makeCall(“8884567”)

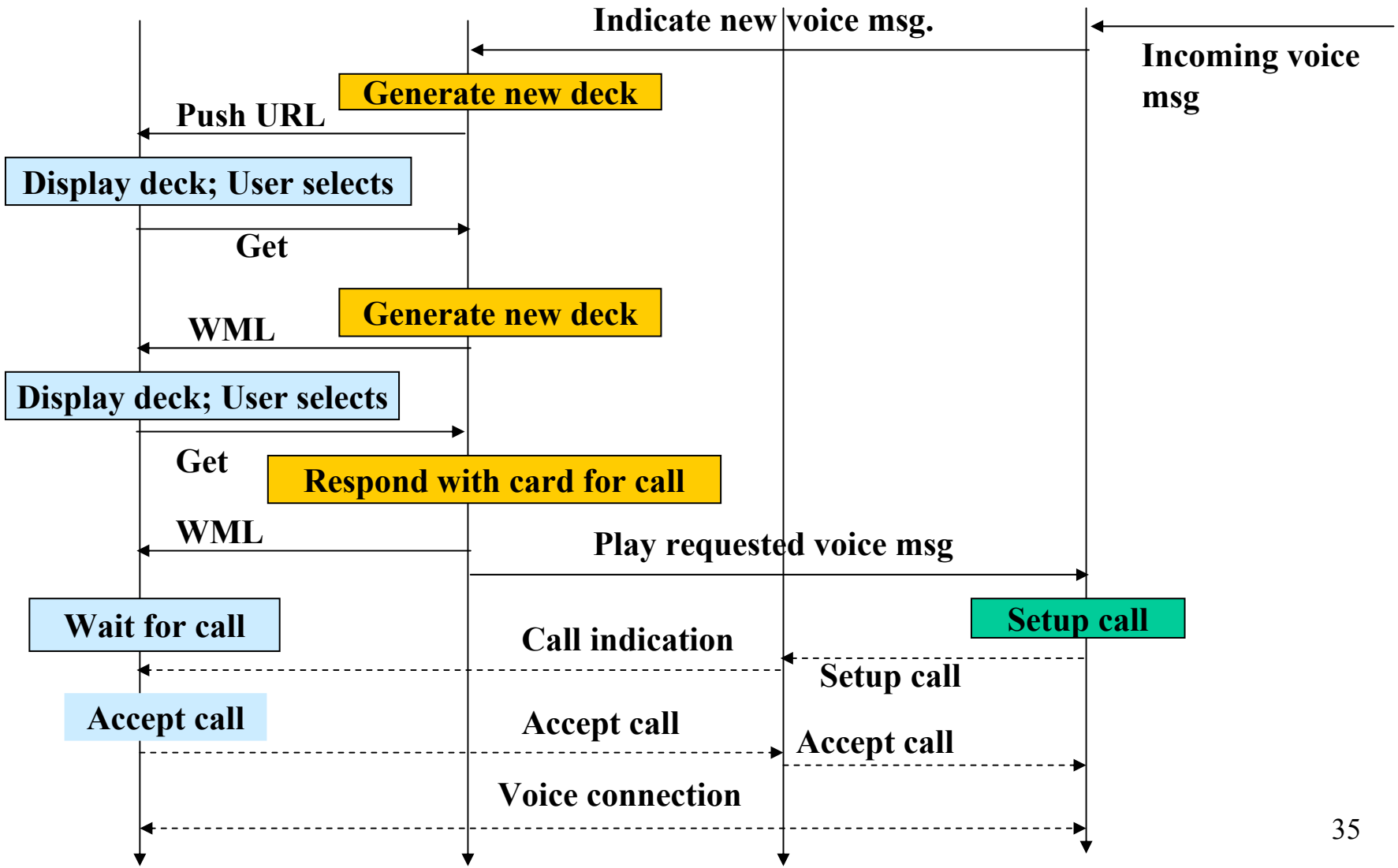
Event handling

- Sources of event
  - From mobile network ➔ WTA event
  - From WTA server (WSP push service)
- Handling of events: **event/ URL** binding

WTA Logical Architecture



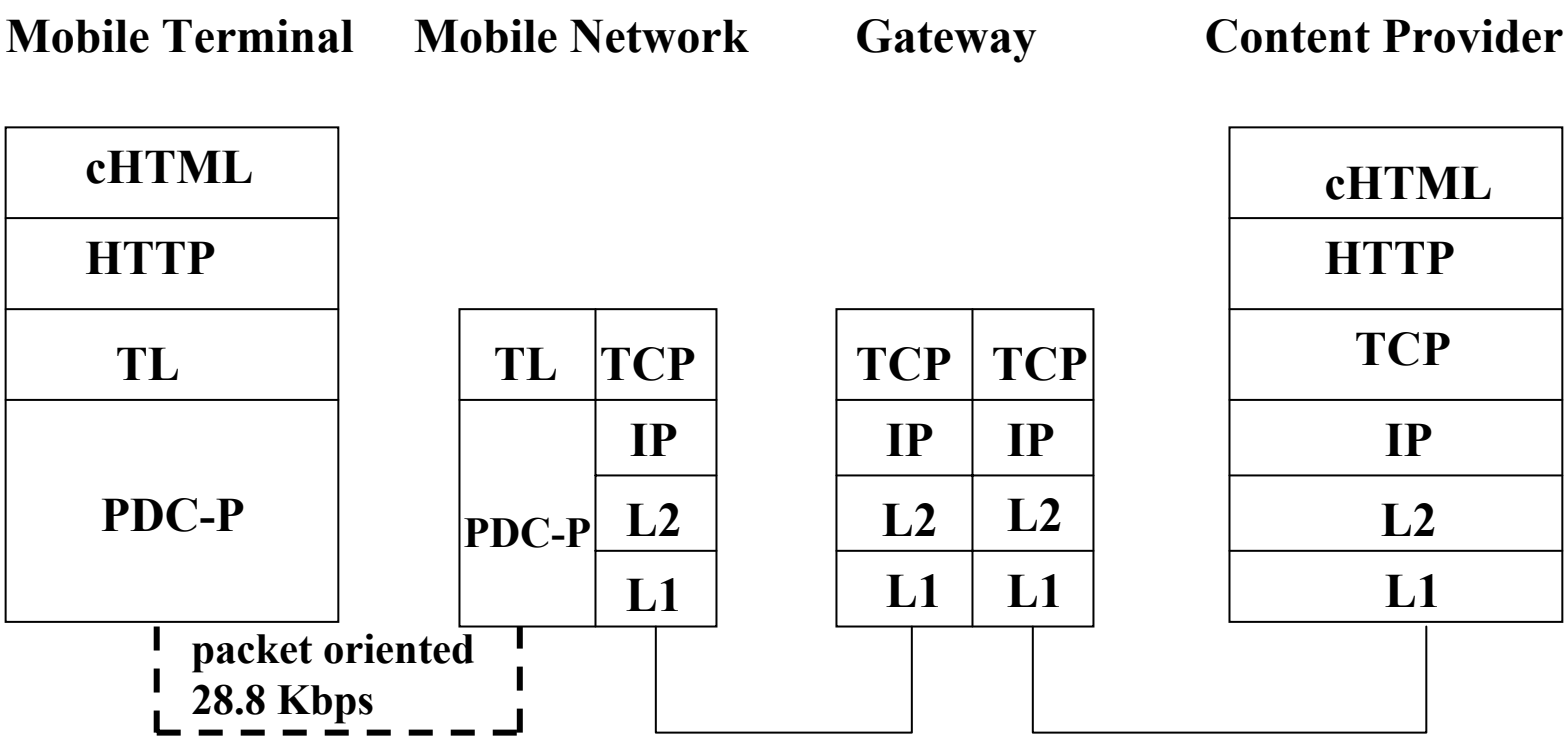
WTA example: voice messaging



iMode

- Introduced by NTT DoCoMo in Japan in 1999
- Services offered
  - » **e-mail, web access, picture exchange**
- Big success: 30+ million users in 3 years!

iMode Protocol Stack



Transport Layer: Connection-oriented, stop-and-go flow control, ARQ

cHTML + tags: Compact HTML with tags for e-mail, telephony, …
Supports animated color pictures in GIF format

Initial WAP vs. iMode

- WAP
  - **Announced as Internet on mobile phone**
    - **➔ High expectation**
  - **Started with connection-oriented bearer service**
- iMode
  - **Just a new service**
    - **➔ Quick acceptance**
  - **Started with packet-oriented bearer**

WML/ WMLScript will not be covered in the class

The following book is a good source:

The Wireless Application Protocol
S. Singhal, et. al.
Addison Wesley

WML

- Design philosophy
  - Very limited capacity of wireless channel
  - Small displays
  - Limited user input facilities
  - Limited memory
  - Low performance computational resource

WML

- WML document: multiple **cards**
- **Deck**
  - a collection of **cards**
  - Similar to an HTML page (identified by an URL)
  - Unit of content transmission
- WML browser:
  - Fetches decks
  - User navigates through a series of cards
  - User **reviews** contents, **enters** data, **makes choices**, …

WML Example

```
<WML>
    <CARD>
      <DO TYPE="ACCEPT"> <GO URL="#card_two"/> </DO>
      This is a simple first card …..Choose on the next one.
    </CARD>

    <CARD NAME="card_two">
        … favorite pizza:
      <SELECT KEY="PIZZA">
          <OPTION VALUE="Mar">Margherita</OPTION>
          <OPTION VALUE="Fun">Funghi</OPTION>
          <OPTION VALUE="Vul">Vulcano</OPTION>
      </SELECT>
    </CARD>
</WML>
```

WMLScript

- WML content: **static**
- Capabilities offered by WMLScript
  - **Validity check** of user input: **save** bandwidth and latency
  - Access to device facilities: phone, address book, messages
  - Local user interaction: several interactions ➔ one message to server
- Language
  - **Weakly-typed**
  - **Functions, expressions, control** (*while, if, for, return*)
  - **Value passing**

Example WMLScript

```
function pizza_test(pizza_type) {
        var taste = “unknown”;
        if (pizza_type = “Mar”) {
            taste = “well …”;
        }
        else {
            if (pizza_type = “Vul”) {
                    taste = “quite hot”;
            };
        };
        return taste;
};
```