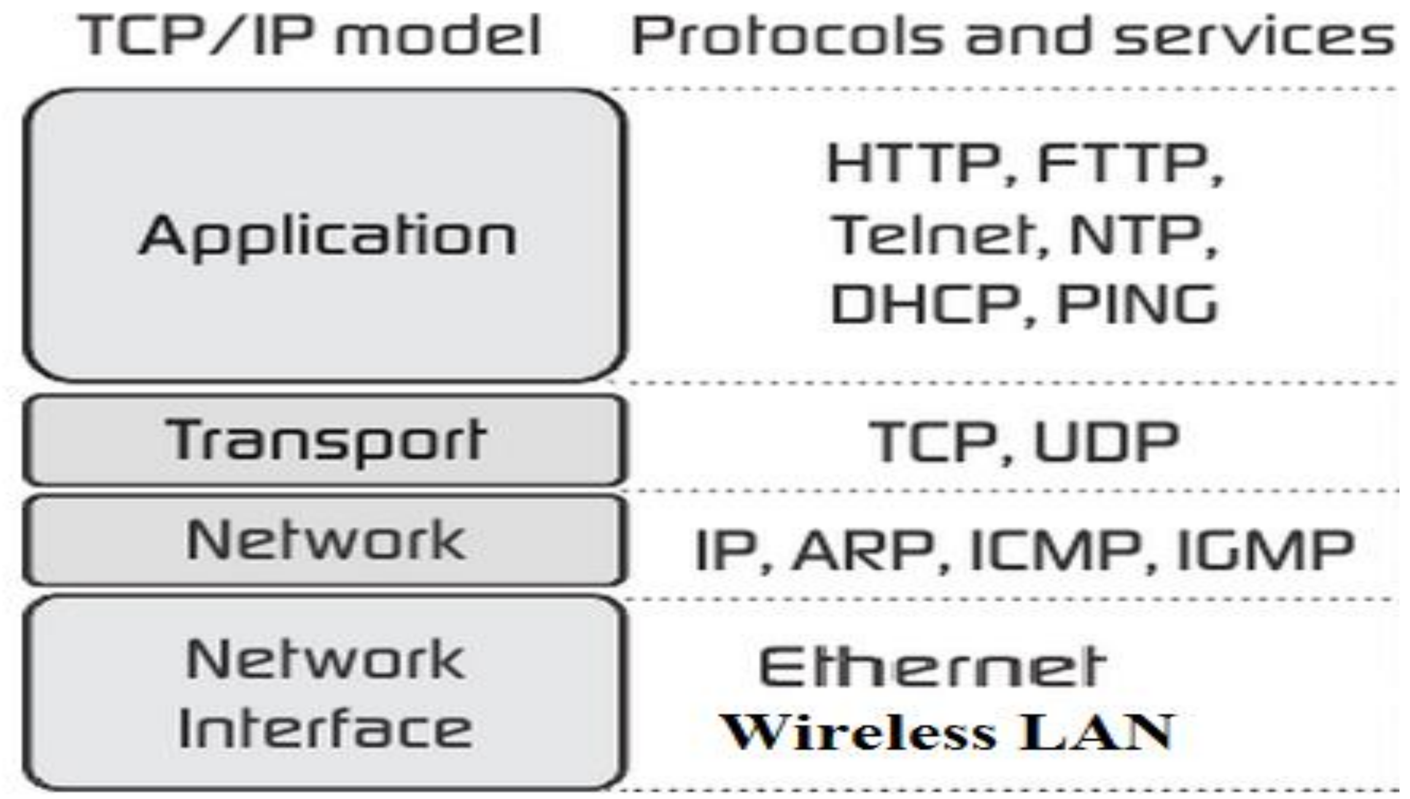
TRADITIONAL TCP

Dr. A. Beulah

AP/CSE

Architecture of TCP/IP

- TCP
  - Telnet, SMTP, FTP
  - HTTP
- UDP
  - DNS
  - SNMP



Terminologies of TCP/IP

- TCP
- IP
- HTTP
- SMTP
- MIME
- FTP
- SNMP

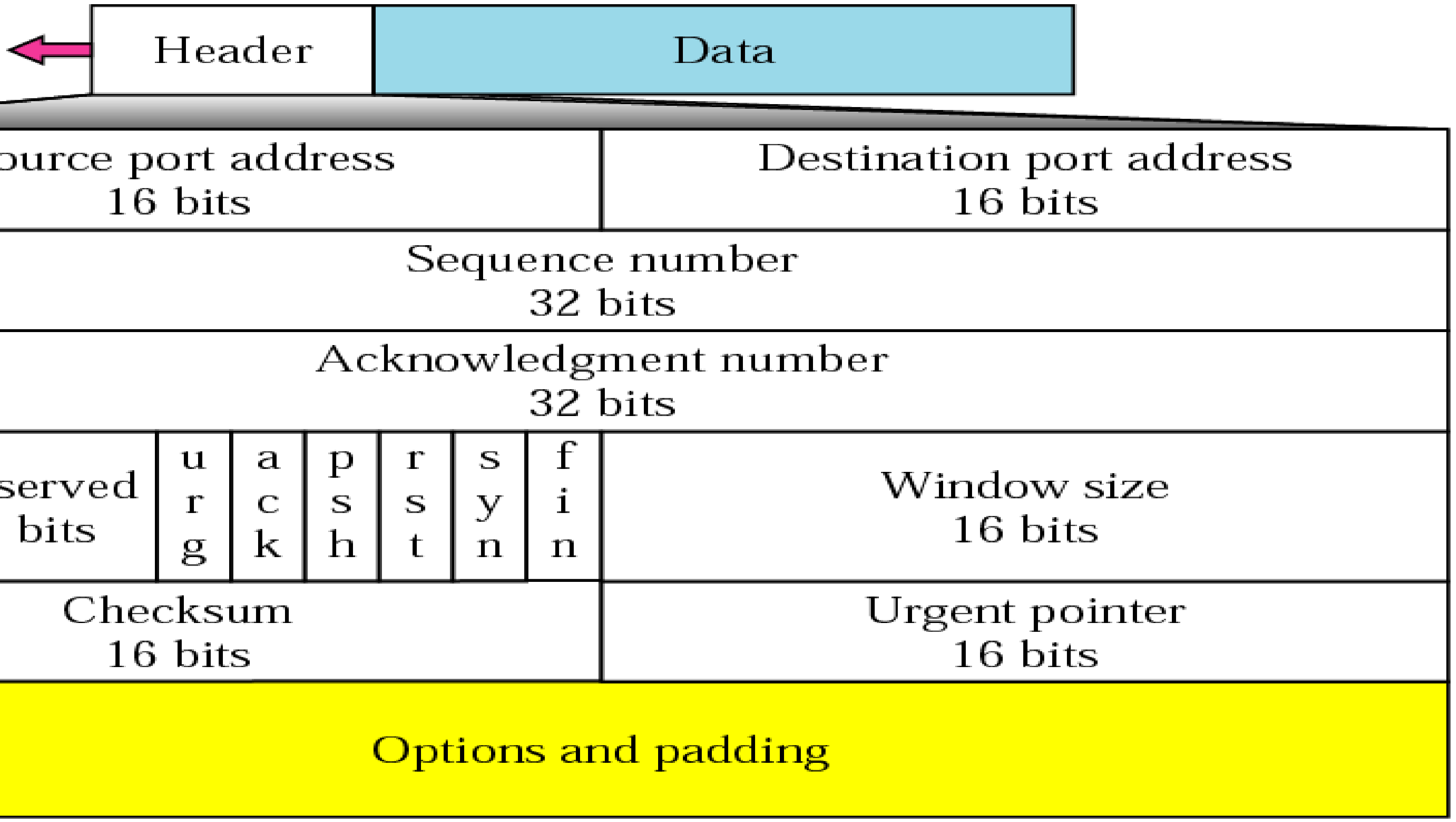
- ARP
- RARP
- BOOTP
- Routers
- DNS
- IP Address
- ICMP
- IGMP

Overview of TCP Operations

- TCP Segment
- Port address
- Data Encapsulation

TCP Segment

- 20 or 60 byte header



Numbering System

- Byte Number
- Sequence Number
- Acknowledgment Number

Byte Number

- Each byte should be numbered
- Random number – 1057
- Data contains 6000 bytes
- 1057 – 7056

Sequence Number

- Sequence number for each segment is the number of the first byte carried in that segment
- Imagine a TCP connection is transferring a file of 6000 bytes. The first byte is numbered 10010. What are the sequence numbers for each segment if data are sent in five segments with the first four segments carrying 1000 bytes and the last segment carrying 2000 bytes?

Sequence Number

- The following shows the sequence number for each segment:

Segment 1 ==> sequence number: 10,010

(range: 10,010 to 11,009)

Segment 2 ==> sequence number: 11,010

(range: 11,010 to 12,009)

Segment 3 ==> sequence number: 12,010

(range: 12,010 to 13,009)

Segment 4 ==> sequence number: 13,010

(range: 13,010 to 14,009)

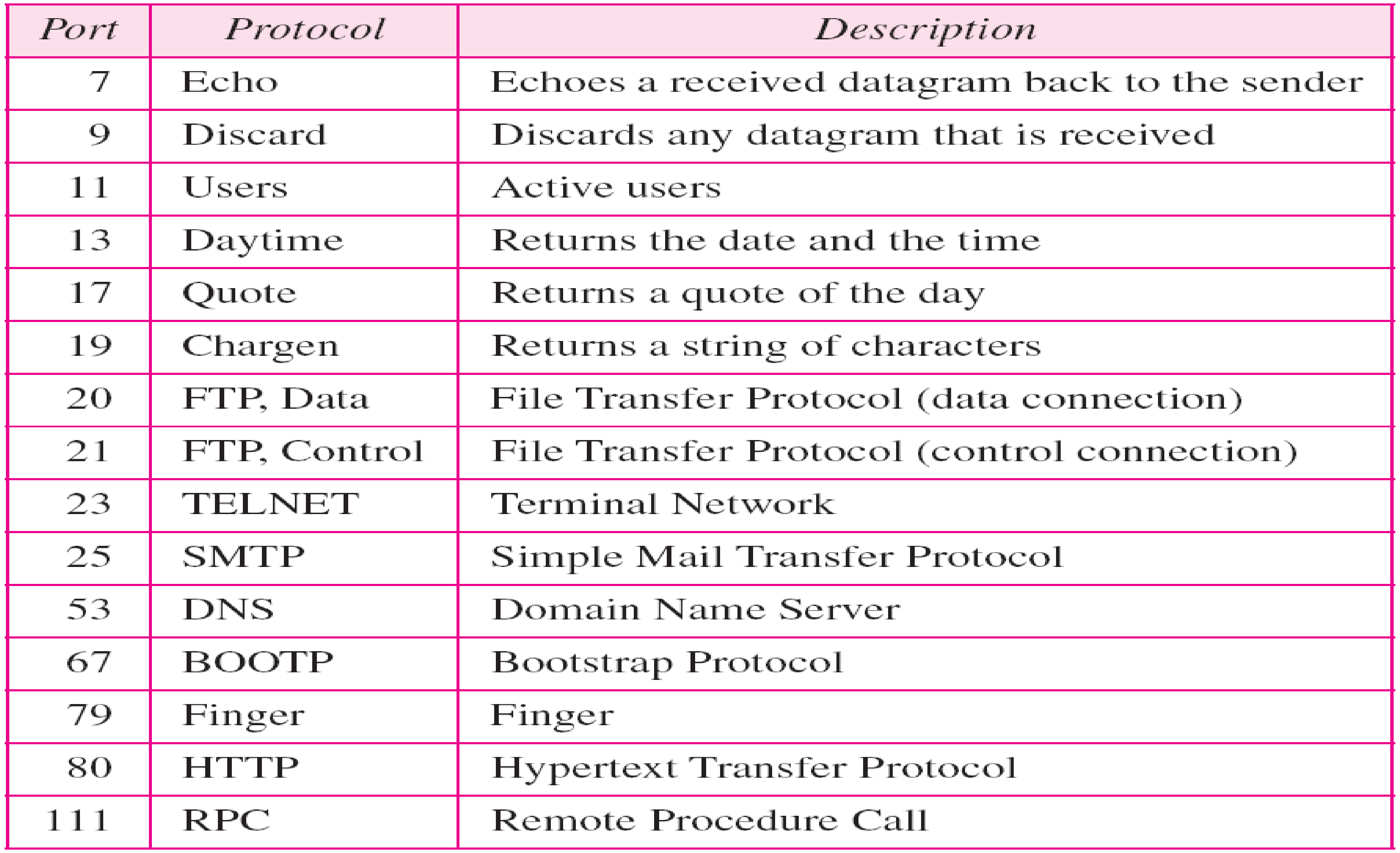
Segment 5 ==> sequence number: 14,010

(range: 14,010 to 16,009)

Acknowledgment Number

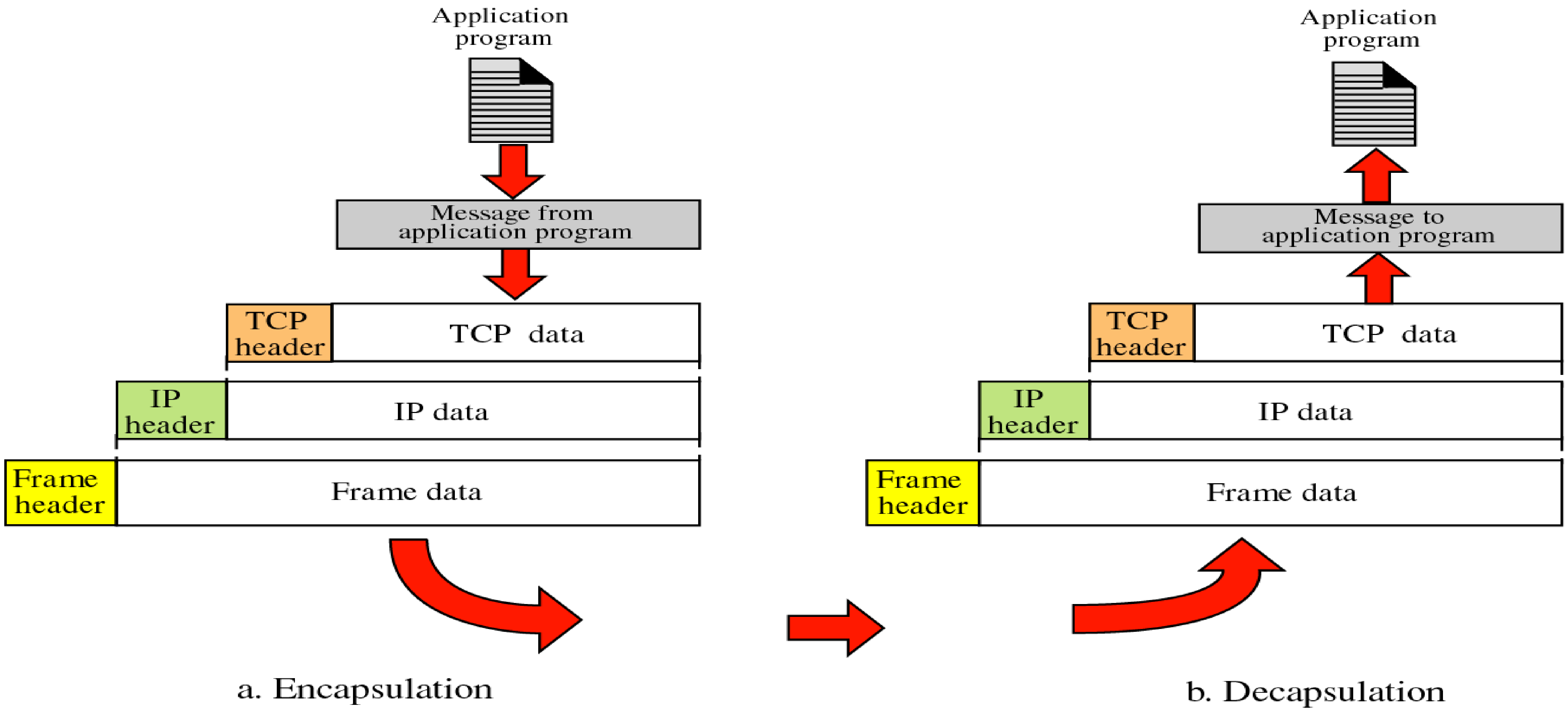
- The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive.
- The acknowledgment number is cumulative.

Port Nos.



Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

Data Encapsulation



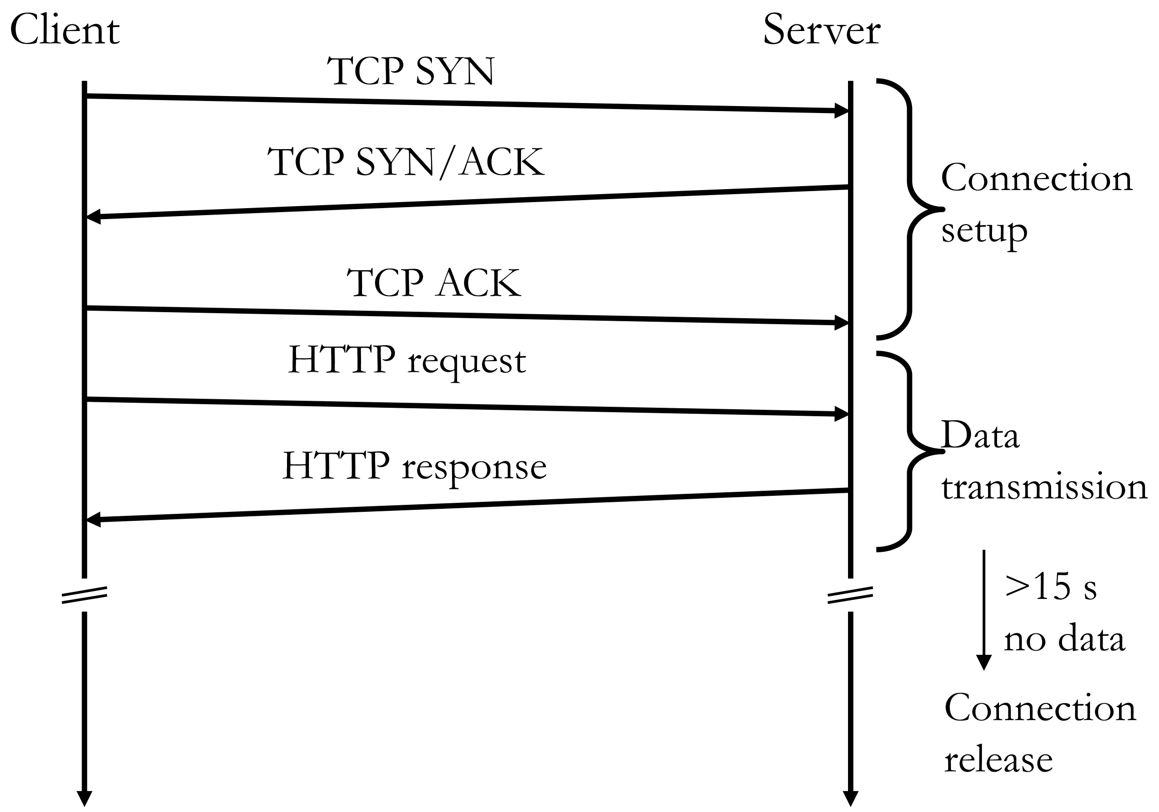
Application Layer Protocols of TCP

- SMTP
- FTP
- Telnet

ADAPTATION OF TCP WINDOW

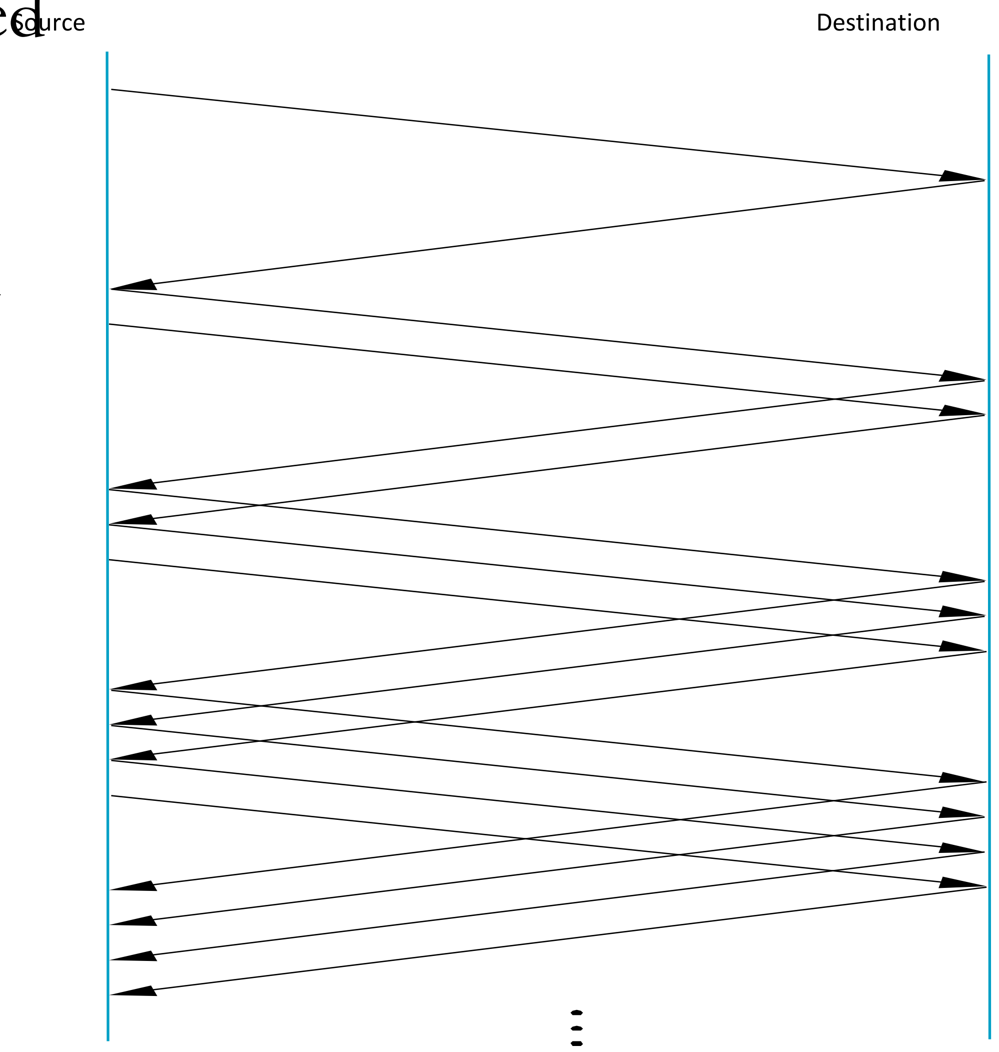
Basic Concepts

- E.g. HTTP (used by web services) typically uses TCP
  - Reliable transport between client and server required
- TCP
  - Stream oriented, not transaction oriented
  - Packet Loss
    ➔ congestion
    ➔ slow down transmission
- Well known – TCP guesses quite often wrong in wireless and mobile networks
  - Packet loss due to transmission errors
  - Packet loss due to change of network
- Result
  - Severe performance degradation



Additive Increase

- Additive Increase is a reaction to perceived available capacity.
- Linear Increase → For each "cwnd's worth" of packets sent, increase cwnd by 1 packet.
- In practice, cwnd is incremented exponentially for each arriving ACK.



Silly Window Syndrome

- If a receiver with this problem is unable to process all incoming data, it requests that its sender reduce the amount of data they send at a time.
- MSS/2
- If the receiver continues to be unable to process all incoming data, the window becomes smaller and smaller, sometimes to the point that the data transmitted is smaller than the packet header, making data transmission extremely inefficient.
- The name of this problem is due to the window size shrinking to a "silly" value.
- Nagel's Algorithm

Silly Window Syndrome

Nagel's Algorithm

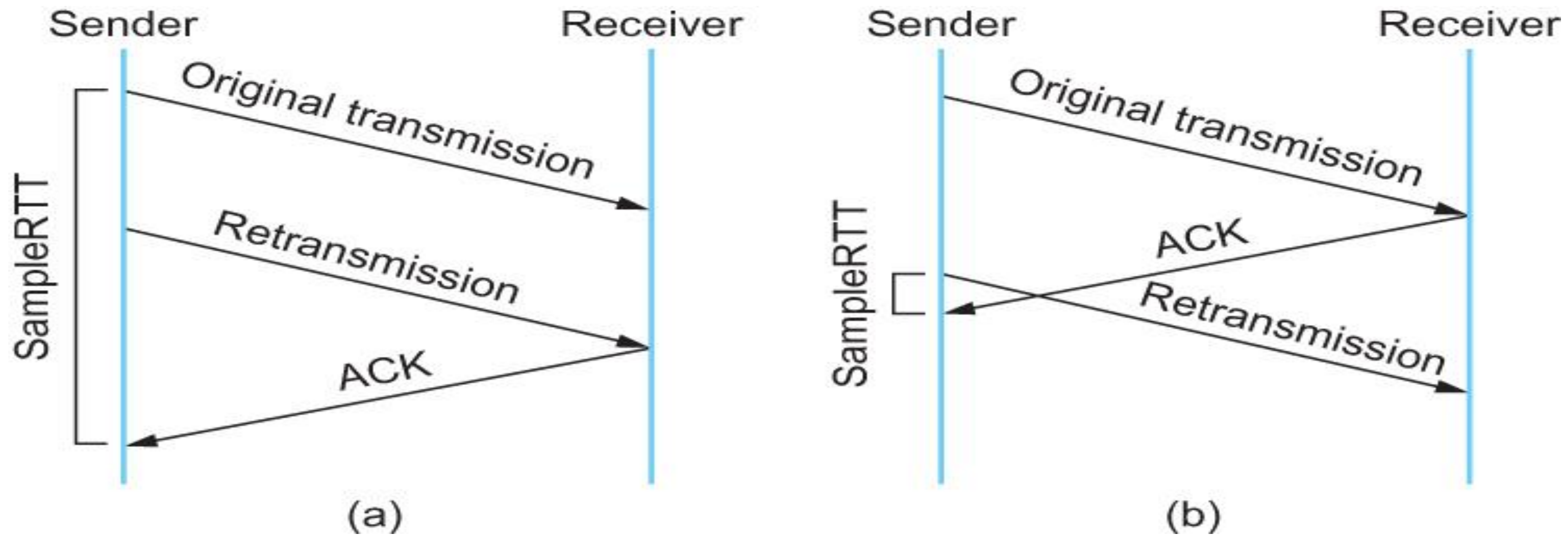
```
if there is new data to send then
    if the window size $\geq$ MSS and available data is $\geq$ MSS then
        send complete MSS segment now
    else
        if there is unconfirmed data still in the pipe then
            enqueue data in the buffer until an acknowledge is received
        else
            send data immediately
        end if
    end if
end if
```

Adaptive Retransmission

- TCP achieves reliability by retransmitting segments after a Timeout

- Choosing the value of the Timeout
  - Set time out as a function as RTT
  - If too small, retransmit unnecessarily
  - If too large, poor throughput
  - Make this adaptive, to respond to changing congestion delays in Internet

Adaptive Retransmission



Associating the ACK with (a) original transmission versus (b) retransmission

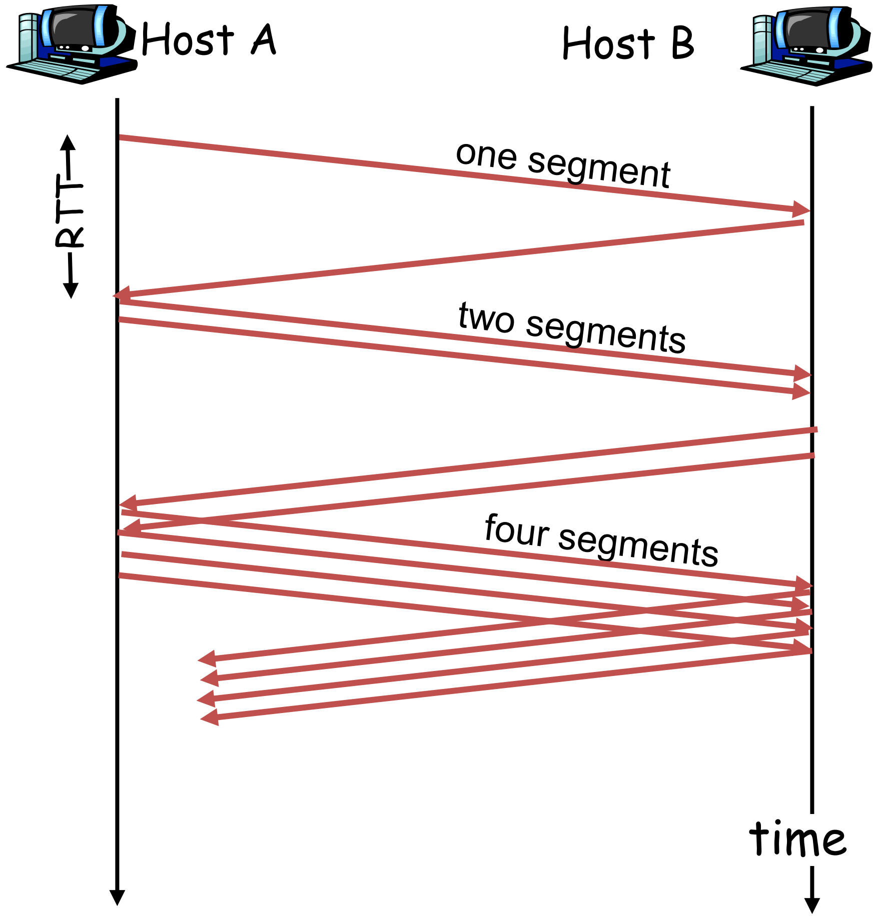
IMPROVEMENT IN TCP PERFORMANCE

Improvement in TCP Performance

- Traditional TCP
  - Slow Start
  - Fast Retransmit and recovery
- Popular TCP Congestion Control Algorithms
  - TCP Tahoe
  - TCP Reno
  - TCP SACK
  - TCP Vegas
- TCP in Mobile Networks

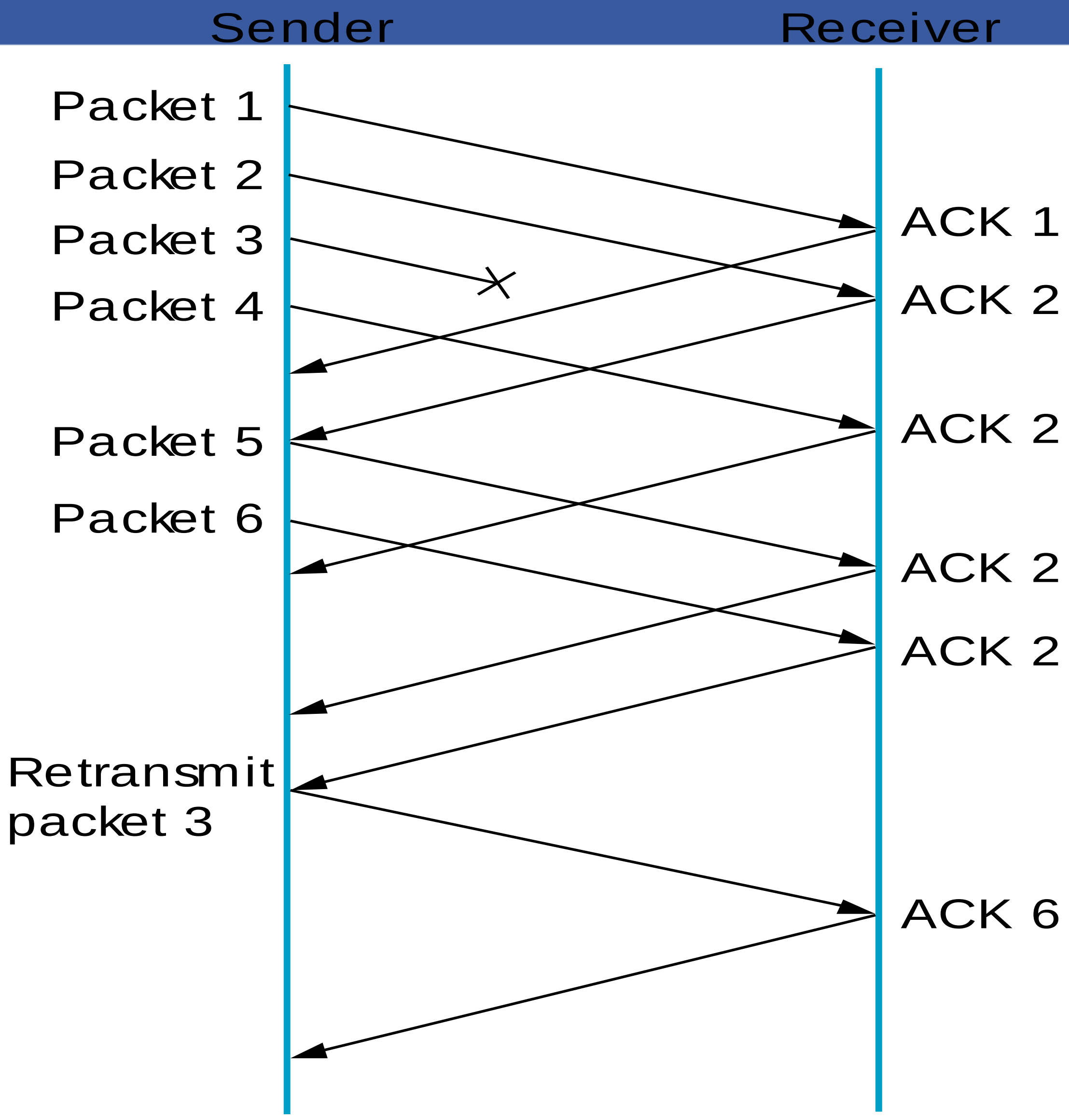
TCP Slow-Start algorithm

- Sender calculates a congestion window for a receiver
- Start with a congestion window size equal to one segment
- Exponential increase of the congestion window up to the congestion threshold, then linear increase
- Missing acknowledgement causes the reduction of the congestion threshold to one half of the current congestion window
- Congestion window starts again with one segment



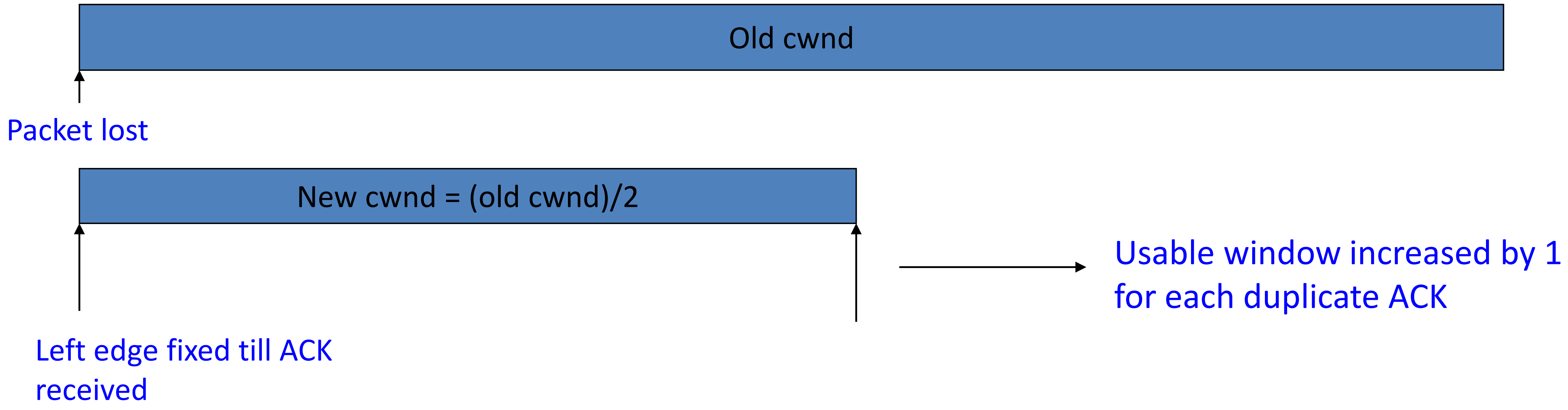
Fast Retransmit

- Coarse timeouts remained a problem, and Fast retransmit was added with TCP.
- Since the receiver responds every time a packet arrives, this implies the sender will see duplicate ACKs.
- Use duplicate ACKs to signal lost packet.
- Upon receipt of *three duplicate ACKs,* the TCP Sender retransmits the lost packet.



Fast Recovery

- Fast recovery was added with TCP.
- When fast retransmit detects three duplicate ACKs, start the recovery process from congestion
- After Fast Retransmit, half the cwnd and commence *recovery from this point using linear additive increase.*


Old cwnd
Packet lost
New cwnd = (old cwnd)/2
Usable window increased by 1 for each duplicate ACK
Left edge fixed till ACK received

Popular TCP Congestion Control Algs

- Comparison
  - Tahoe: Slow start, fast retransmit
  - Reno: Tahoe + fast recovery
  - New-Reno: Reno with modified fast recovery
  - SACK: Reno + selective ACKs
  - Vegas: Modified Slow start, and retransmission

TCP Tahoe

Packet loss is identified in 2 ways:

1. Time out
2. Dup acks

- Slow start, fast retransmit
- Fast retransmit improves channel utilization
- Two slow start situations:
  - At the very beginning of a connection **{cold start}**.
  - When the connection goes dead waiting for a timeout to occur (i.e, the advertized window goes to zero!)

TCP Tahoe

- Coarse timeouts remained a problem, and Fast retransmit was added with **TCP Tahoe**.
- Since the receiver responds every time a packet arrives, this implies the sender will see duplicate ACKs.

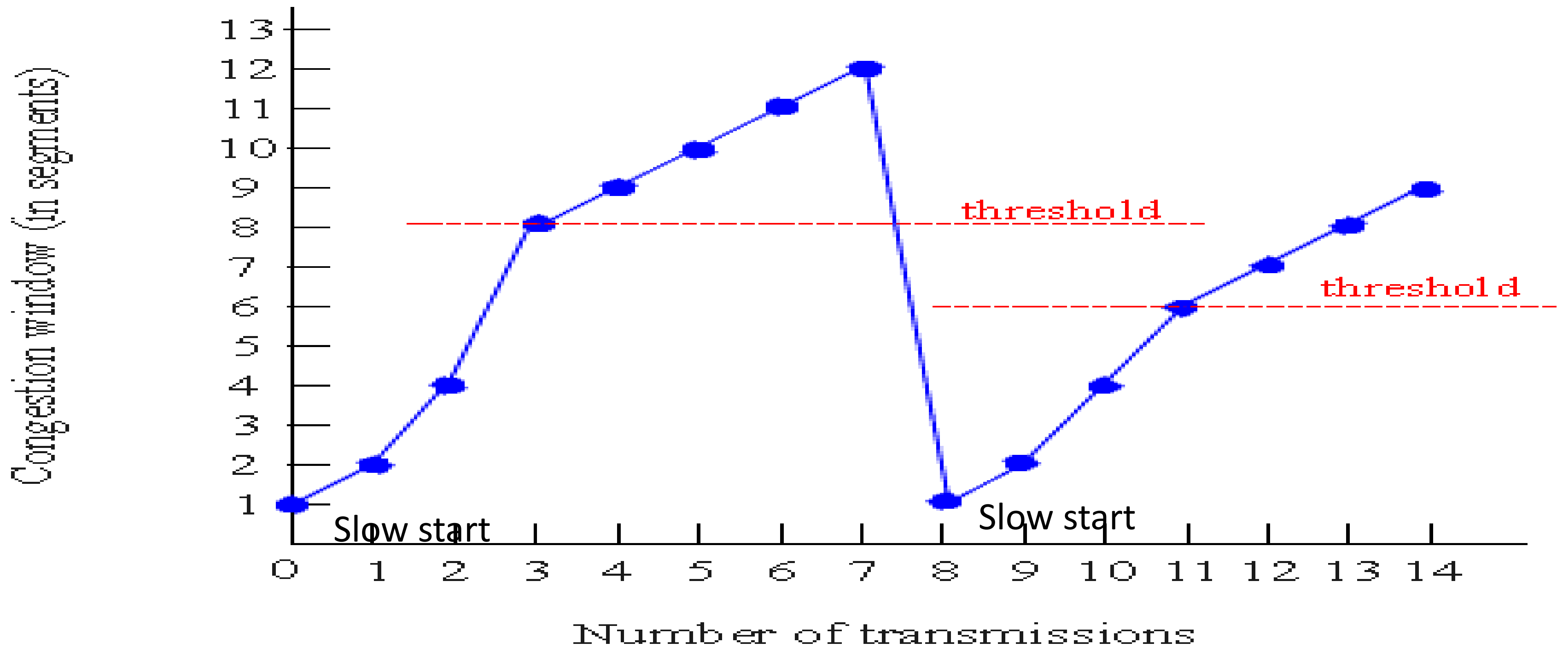
Basic Idea:: *use **duplicate ACKs** to signal lost packet.*

Fast Retransmit

Upon receipt of three duplicate ACKs, the TCP Sender retransmits the lost packet.

TCP Tahoe

- Limitations
  - Too aggressive
  - Returns to slow start on every congestion



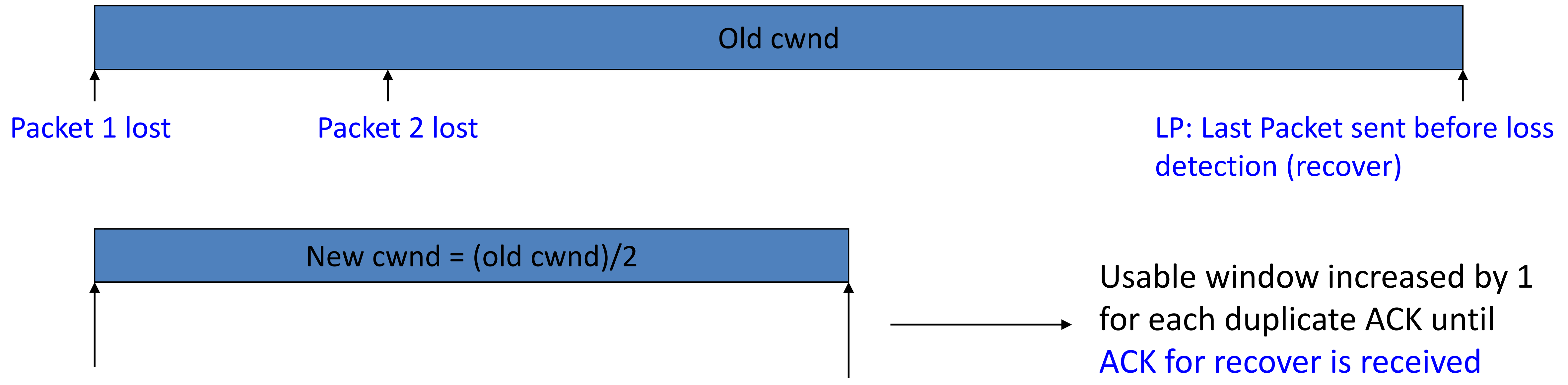
TCP Reno

- Tahoe + fast recovery
- After fast retransmit, reduce cwnd by half, and continue sending segments at this reduced level. (Instead of moving to slow start)
- Limitations
  - Has to wait for 3 Dup-ACKs
  - Inefficient incase of multiple losses

TCP New-Reno

- Reno with modified fast recovery
- New-Reno continues with fast recovery if a *partial ACK* is received
- When duplicate ACKs trigger a retransmission for a lost packet, remember the highest packet sent from window in recover.
- In sender side, upon receiving an ACK,
  - if ACK < recover => partial ACK (for old segment sent)
  - If ACK ≥ recover => new ACK (for previous segment sent)
- Partial ACK implies another lost packet: retransmit next packet, inflate window and stay in fast recovery.
- New ACK implies fast recovery is over: starting from 0.5 x cwnd proceed with congestion avoidance (linear increase).
- New Reno recovers from n losses in n round trips.

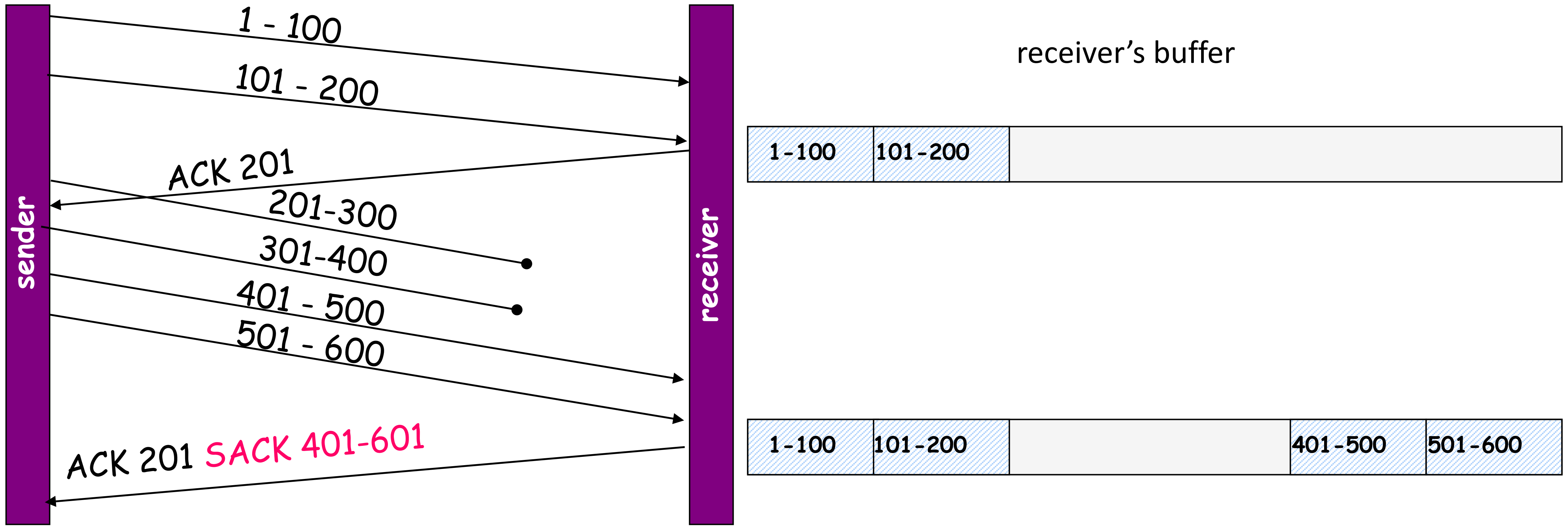
TCP New-Reno



Usable window increased by 1 for each duplicate ACK until ACK for recover is received

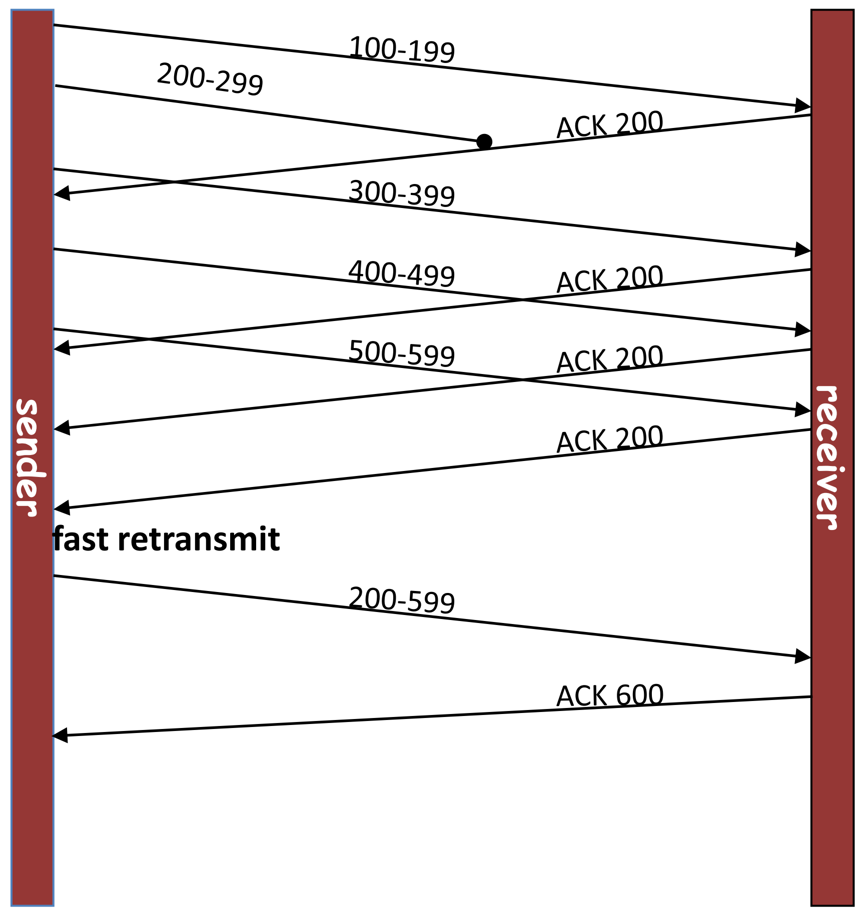
TCP SACK

- Reno + selective ACKs

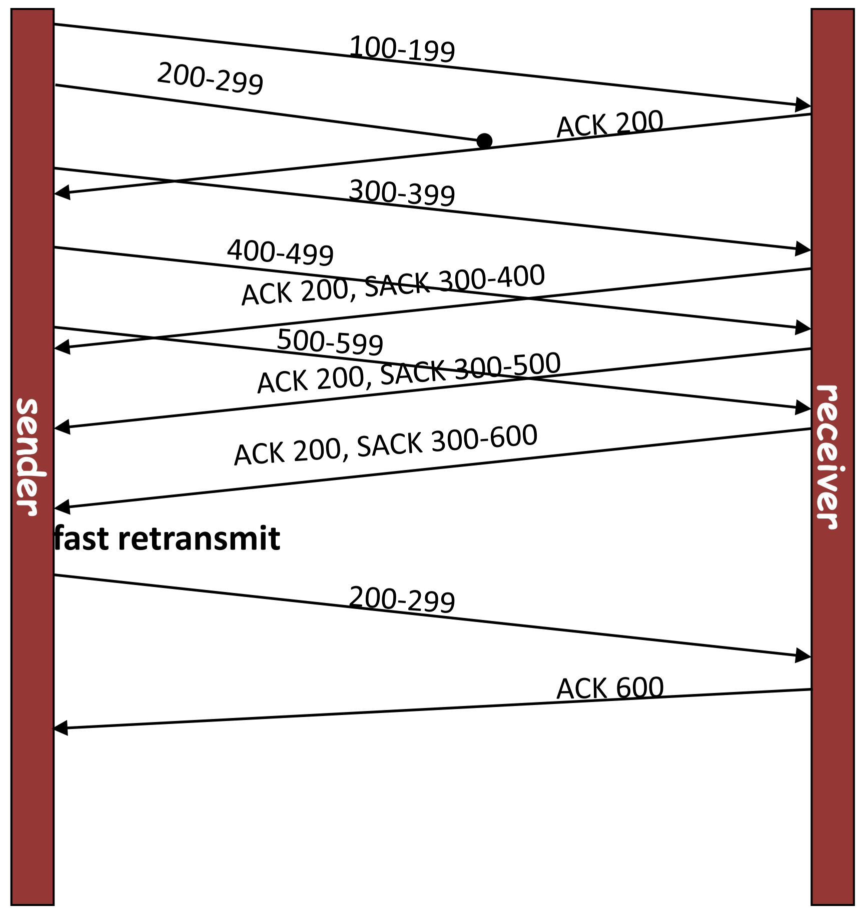


TCP SACK

TCP without SACK



TCP with SACK



TCP Vegas

- The only way Tahoe, Reno and New Reno can detect congestion is by creating congestion!
  - They carefully probe for congestion by slowly increasing their sending rate.
  - When they find (create), congestion, they cut sending rate at least in half!

TCP Vegas

- There are three ways proposed in Vegas to increase delivery throughput and decrease packet loss.
  - Modified Slow-Start Mechanism
    - Cwnd is allowed exponential growth only every other RTT. (doubles the size of cwnd every 2 RTT time while there are no losses).
  - New Congestion Avoidance Mechanism
    - Control the size of cwnd by observing the variation of RTT
    - Congestion in network (high RTT)
  - New Retransmission Mechanism
    - Sender does retransmission after a dupACK received, if RTT estimate > timeout (without waiting for 3 dupACKs)

Summary

- Transmission Control Protocol(TCP)
  - Basic concepts of TCP
  - TCP header
- Adaptation of TCP window
  - Silly window syndrome
  - Adaptive retransmision
- Improvements in TCP (Wired TCP)
  - TCP Tahoe
  - TCP Reno
  - TCP New-Reno
  - TCP SACK
  - TCP Vegas

Test your understanding

- Why do congestion occur in a network?
- How does slow start help improve the performance of TCP?

References

- Prasant Kumar Pattnaik, Rajib Mall, “Fundamentals of Mobile Computing”, PHI Learning Pvt. Ltd, New Delhi – 2012.
- Jochen H. Schller, “Mobile Communications”, Second Edition, Pearson Education, New Delhi, 2007.
- Behrouz A. Forouzan, “Data communication and Networking”, Fourth Edition,Tata McGraw – Hill, 2011.

MOBILE TCP

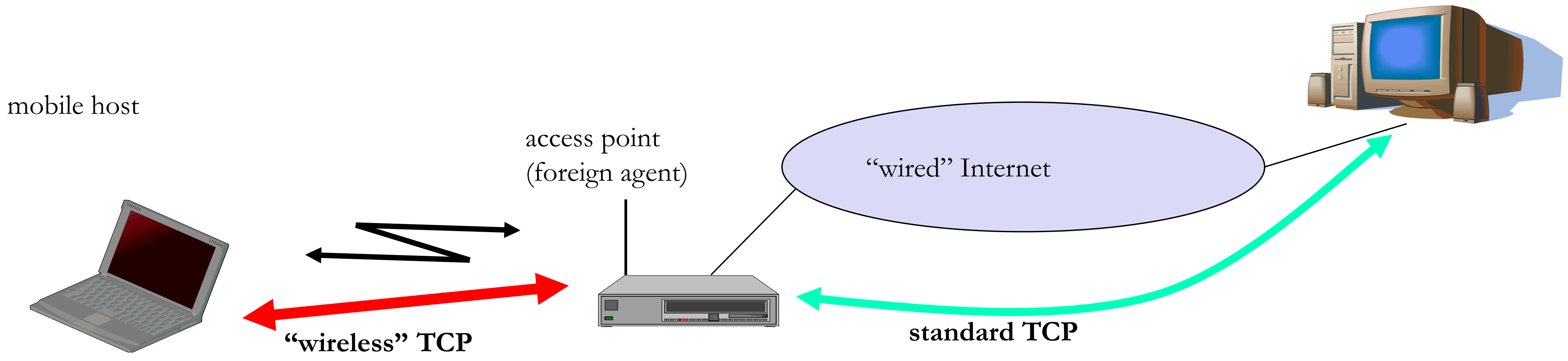
Dr. A. Beulah
AP/CSE

TCP in Mobile Networks

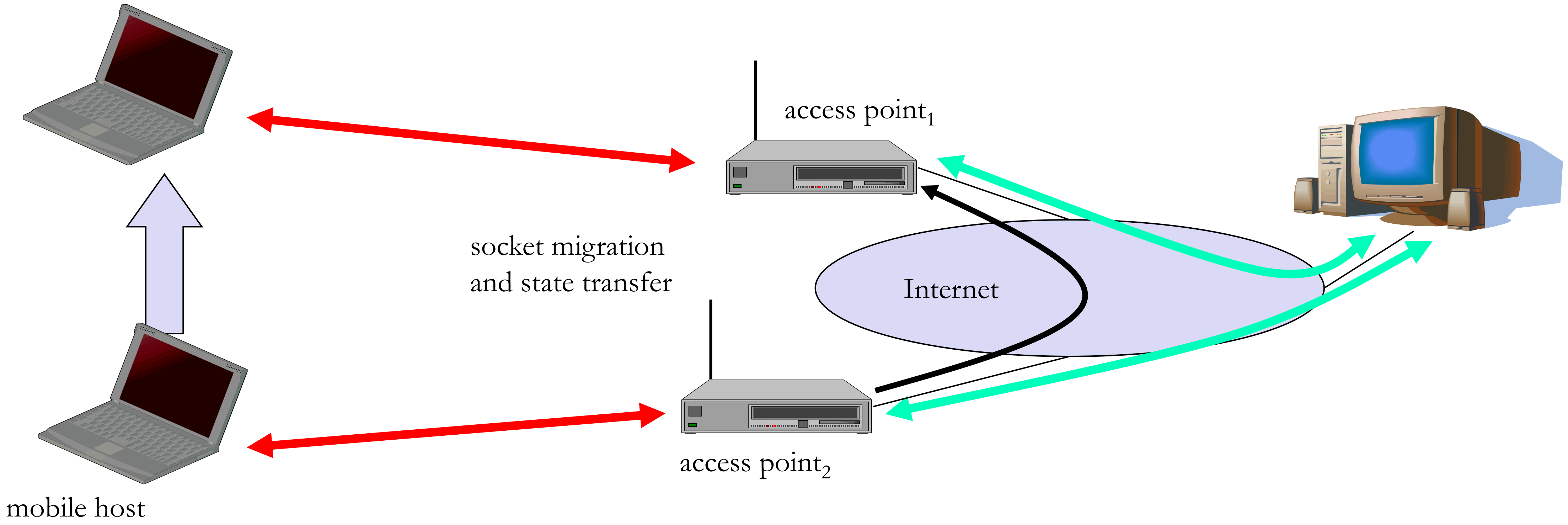
- TCP in Single-Hop Wireless Networks (Wireless LAN)
  - Indirect TCP (I-TCP)
  - Fast Retransmission
  - Snooping TCP (S-TCP)
- Mobile TCP - TCP for Cellular Networks
  - Freeze TCP
- TCP in Multi-Hop Wireless Networks (Ad hoc)
  - TCP Feedback (TCP-F)

Indirect TCP (I-TCP)

- Indirect TCP or I-TCP segments the connection
  - The access point is seen as the mobile host for the fixed host and as the fixed host for the mobile host.
  - Splitting of the TCP connection at, e.g., the foreign agent into 2 TCP connections, no real end-to-end connection any longer
  - Hosts in the fixed part of the net do not notice the characteristics of the wireless part



I- TCP Socket and State Migration



Indirect TCP (I-TCP)

- Advantages
  - No changes in the fixed network necessary, no changes for the hosts (TCP protocol) necessary, all current optimizations to TCP still work
  - Transmission errors on the wireless link do not propagate into the fixed network
  - Simple to control, mobile TCP is used only for one hop between, e.g., a foreign agent and mobile host
  - Very fast retransmission of packets is possible, the short delay on the mobile hop is known
- Disadvantages
  - Loss of end-to-end semantics, an acknowledgement to a sender does now not any longer mean that a receiver really got a packet, foreign agents might crash
  - Higher latency possible due to buffering of data within the foreign agent and forwarding to a new foreign agent

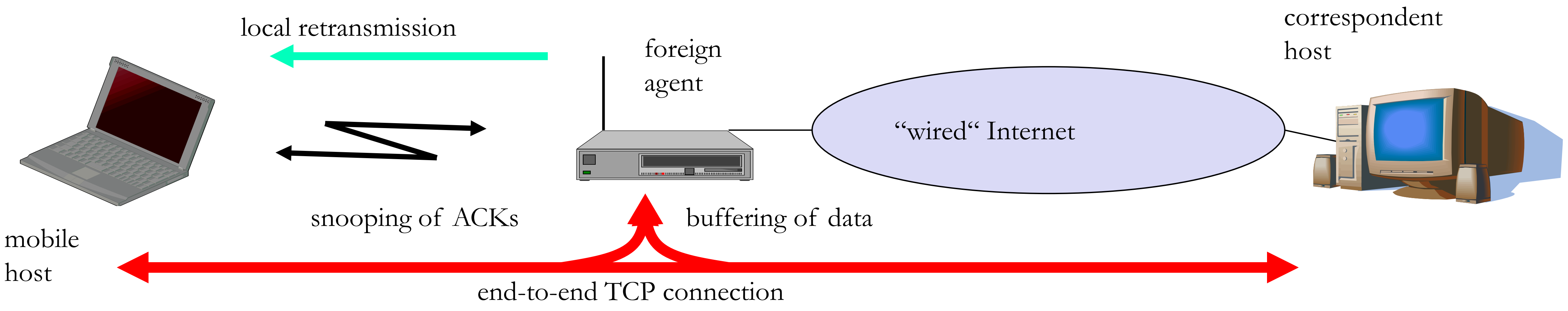
Fast Retransmit/Fast Recovery

- Change of foreign agent often results in packet loss
  - TCP reacts with slow-start although there is no congestion
- Forced Fast retransmit
  - As soon as the mobile host has registered with a new foreign agent, the MH sends duplicated acknowledgements on purpose
  - This forces the fast retransmit mode at the communication partners
  - Additionally, the TCP on the MH is forced to continue sending with the actual window size and not to go into slow-start after registration
- Advantage
  - Simple changes result in significant higher performance
- Disadvantage
  - Further mix of IP and TCP, no transparent approach

Snooping TCP(S-TCP)

- Drawback of I-TCP
  - Segmentation of a single TCP into 2 TCP connections.
- The extension done on traditional TCP is transparent
  - Buffer the packets sent to the mobile host
  - Lost packets on the wireless link will be retransmitted immediately by the mobile host or foreign agent, respectively → "Local" Retransmission
  - Foreign Agent "snoops" the packet flow and recognizes acknowledgements in both directions
  - FA filters the ACKs
  - Extension of TCP is done within the Foreign Agent

Snooping TCP(S-TCP)



Snooping TCP(S-TCP)

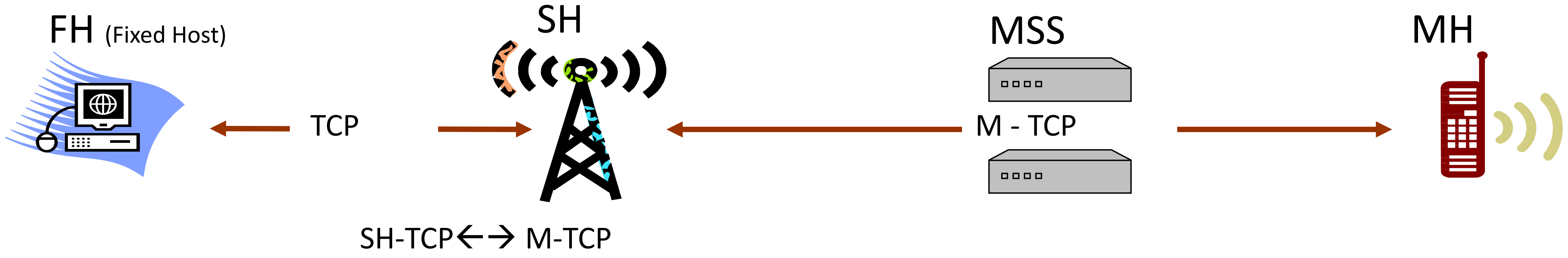
- Data transfer to the mobile host
  - FA buffers data until it receives ACK of the MH, FA detects packet loss via duplicated ACKs or time-out
- Data transfer from the mobile host
  - FA detects packet loss on the wireless link via sequence numbers, FA answers directly with a NACK to the MH
  - MH can now retransmit data with only a very short delay
- Integration of the MAC layer
  - MAC layer often has similar mechanisms to those of TCP
  - thus, the MAC layer can already detect duplicated packets due to retransmissions and discard them

Snooping TCP(S-TCP)

- Advantages
  - End to end TCP semantic is preserved
  - No change in CN. Changes are done only in FA.
- Disadvantages
  - Snooping TCP does not isolate the wireless link as good as I-TCP
  - Snooping might be useless depending on encryption schemes

Mobile TCP

- Major thing to consider → Handling of lengthy and/or frequent disconnections
- Traditional TCP (Disconnections)
  - The sender tries to retransmit an unacknowledged packet every minute and gives up after 12 retransmission.
- I-TCP (Disconnections)
  - The FA / Proxy has to buffer more and more data.
  - Long disconnection period requires lengthy buffer to store more data.



Mobile TCP

- M-TCP splits as I-TCP does (ie. 2 segmentation connection)
  - Unmodified TCP fixed network to Supervisory Host (SH)
  - Optimized TCP between SH and MH
- Supervisory Host
  - No caching, No retransmission
  - Packet loss → Retransmission done by original sender and not by SH. (End to end Semantics)
  - Monitors all packets and acknowledges it.
  - If disconnection detected
    - Set sender window size to 0
    - Sender automatically goes into persistent mode
  - Old or new SH reopen the window

Mobile TCP

- Advantages
  - Maintains End to end semantics
  - SH does not send ACK itself, but forwards ACK from MH
  - Supports disconnection and no buffer forwarding
- Disadvantages
  - Loss on wireless link propagated into fixed network
  - adapted TCP on wireless link

Freeze TCP

- Mobile hosts can be disconnected for a longer time
  - No packet exchange possible, e.g., in a tunnel, disconnection due to overloaded cells or mux. with higher priority traffic
  - TCP disconnects after time-out completely
- TCP freezing
  - MAC layer is often able to detect interruption in advance
  - MAC can inform TCP layer of upcoming loss of connection
  - TCP stops sending, but does now not assume a congested link
  - MAC layer signals again if reconnected
- Advantage
  - Scheme is independent of data
- Disadvantage
  - TCP on mobile host has to be changed, mechanism depends on MAC layer

TCP Feedback (TCP-F)

- TCP-F allows the sender to be informed about a route disconnection.
- When a link in a route is broken, the upstream node that detects the disconnection will send a Route Failure Notification (RFN) message back to the sender.
- Upon receiving this message, the source enters SNOOZE state.

TCP Feedback (TCP-F)

- In SNOOZE state:
  - The sender stops transmitting all data packets(new or retransmitted)
  - The sender freezes all its timers, cwnd size, retransmission timer etc.
- When the route repair complete message is received, data transmission will be resumed and all timers and state variables will be restored.

Summary

Approach	Mechanism	Advantages	Disadvantages
Indirect TCP	splits TCP connection into two connections	isolation of wireless link, simple	loss of TCP semantics, higher latency at handover
Snooping TCP	"snoops" data and acknowledgements, local retransmission	transparent for end-to-end connection, MAC integration possible	problematic with encryption, bad isolation of wireless link
M-TCP	splits TCP connection, chokes sender via window size	Maintains end-to-end semantics, handles long term and frequent disconnections	Bad isolation of wireless link, processing overhead due to bandwidth management
Fast retransmit/ fast recovery	avoids slow-start after roaming	simple and efficient	mixed layers, not transparent
Freeze TCP	freezes TCP state at disconnect, resumes after reconnection	independent of content or encryption, works for longer interrupts	changes in TCP required, MAC dependant

Test your understanding

- How are handoffs handled in snooping TCP??

- Is the following statement true or false?

  The multicast group membership of a packet is defined by the source IP address.

References

Jochen H. Schller, “Mobile Communications”, Second Edition, Pearson Education, New Delhi, 2007.

Prasant Kumar Pattnaik, Rajib Mall, “Fundamentals of Mobile Computing”, PHI Learning Pvt. Ltd, New Delhi – 2012.

TRANSMISSION CONTROL PROTOCOL
By Mr. Vishal Jaiswal

Transmission Control Protocol (TCP) is a connection-oriented, reliable protocol. TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service.

TCP Congestion -

Congestion refers to a network state where - The message traffic becomes so heavy that it slows down the network response time. Congestion is an important issue that can arise in Packet Switched Network. **Congestion leads to the loss of packets in transit.**

- Congestion is an important issue that can arise in **Packet Switched Network.**
- Congestion leads to the loss of packets in transit.
- So, it is necessary to control the congestion in network.
- It is not possible to completely avoid the congestion.

TCP Congestion Control :

The important service done by TCP is providing end to end transmission. This service has a significant role in congestion control in the network. Congestion free network which is an ideal network means taking the advantage of all the bandwidth in the network.

The sender's window size is determined not only by the receiver but also by congestion in the network. The sender has two pieces of information: the receiver-advertised window size and the congestion window size. The actual size of the window is the minimum of these two.

Actual window size = minimum (rwnd, cwnd)

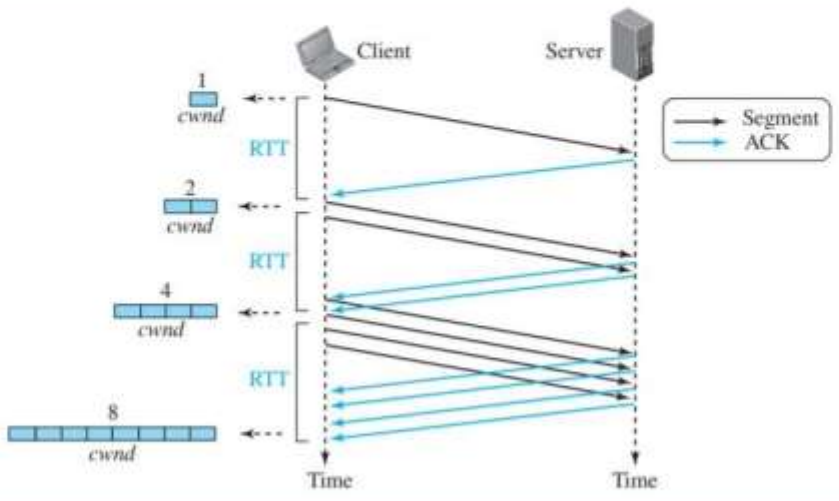
Congestion Policy :

TCP's general policy for handling congestion is based on three phases: slow start, congestion avoidance, and congestion detection. In the slow-start phase, the sender starts with a very slow rate of transmission, but increases the rate rapidly to reach a threshold. When the threshold is reached, the data rate is reduced to avoid congestion. Finally if congestion is detected, the sender goes back to the slow-start or congestion avoidance phase based on how the congestion is detected.

Slow Start: Exponential Increase :

The slow-start algorithm is based on the idea that the size of the congestion window (cwnd) starts with one maximum segment size (MSS), but it increases one MSS each time an acknowledgment arrives.

Figure 24.29 *Slow start, exponential increase*



the size of the congestion window in this algorithm is a function of the number of ACKs arrived and can be determined as follows.

If an ACK arrives, $cwnd = cwnd + 1$.

If we look at the size of the *cwnd* in terms of round-trip times (RTTs), we find that the growth rate is exponential in terms of each round trip time, which is a very aggressive approach:

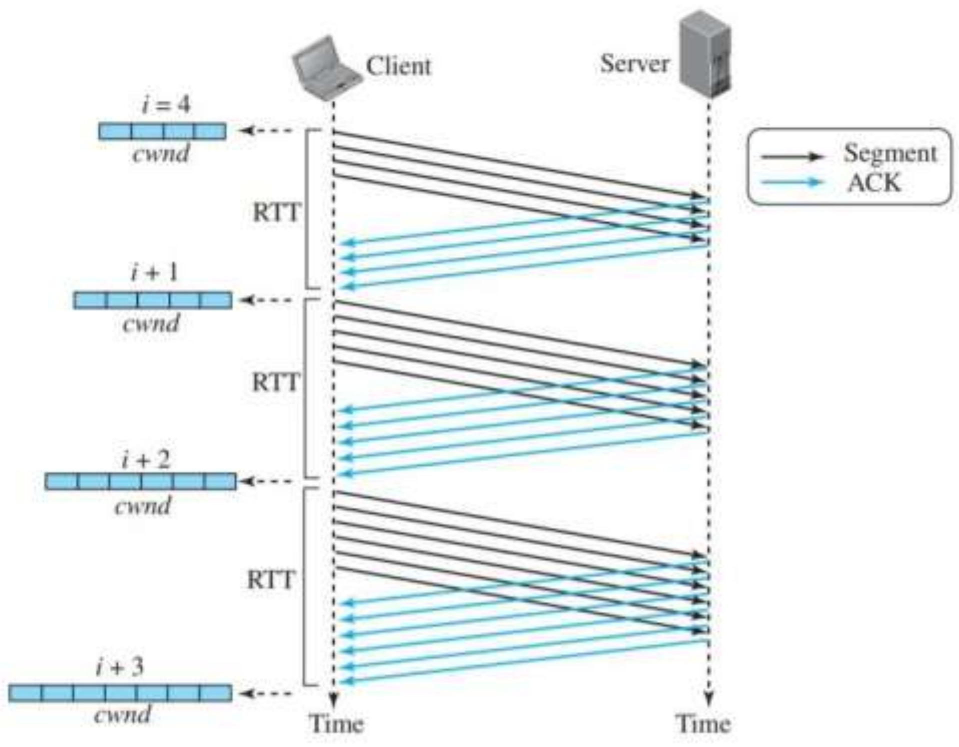
Start	$\rightarrow$	$cwnd = 1 \rightarrow 2^0$
After 1 RTT	$\rightarrow$	$cwnd = cwnd + 1 = 1 + 1 = 2 \rightarrow 2^1$
After 2 RTT	$\rightarrow$	$cwnd = cwnd + 2 = 2 + 2 = 4 \rightarrow 2^2$
After 3 RTT	$\rightarrow$	$cwnd = cwnd + 4 = 4 + 4 = 8 \rightarrow 2^3$

“ In the slow-start algorithm, the size of the congestion window increases exponentially until it reaches a threshold ”

Congestion Avoidance: Additive Increase :

To avoid congestion before it happens, we must slow down this exponential growth. TCP defines another algorithm called congestion avoidance, which increases the cwnd additively instead of exponentially. When the size of the congestion window reaches the slow-start threshold in the case where cwnd = i, the slow-start phase stops and the additive phase begins. In this algorithm, each time the whole “window” of segments is acknowledged, the size of the congestion window is increased by one. A window is the number of segments transmitted during RTT.

Figure 24.30 *Congestion avoidance, additive increase*



The sender starts with cwnd = 4. This means that the sender can send only four segments. After four ACKs arrive, the acknowledged segments are purged from the window, which means there is now one extra empty segment slot in the window. The size of the congestion window is also increased by 1. The size of window is now 5.

After sending five segments and receiving five acknowledgments for them, the size of the congestion window now becomes 6, and so on. In other words, the size of the congestion window in this algorithm is also a function of the number of ACKs that have arrived and can be determined as follows: If an ACK arrives, cwnd 5 cwnd 1 (1/cwnd). The size of the window increases only 1/cwnd portion of MSS (in bytes). In other words, all segments in the previous window should be acknowledged to increase the window 1 MSS bytes. If we look at the size of the cwnd in terms of round-trip times (RTTs), we find that the growth rate is linear in terms of each round-trip time, which is much more conservative than the slow-start approach.

Start	$\rightarrow$	$cwnd = i$
After 1 RTT	$\rightarrow$	$cwnd = i + 1$
After 2 RTT	$\rightarrow$	$cwnd = i + 2$
After 3 RTT	$\rightarrow$	$cwnd = i + 3$

In the congestion-avoidance algorithm, the size of the congestion window increases additively until congestion is detected.

Congestion Detection: Multiplicative Decrease :

If congestion occurs, the congestion window size must be decreased. The only way the sender can guess that congestion has occurred is by the need to retransmit a segment. However, retransmission can occur in one of two cases: when a timer times out or when three Duplicate ACKs are received. In both cases, the size of the threshold is dropped to one-half, a multiplicative decrease.

TCP implementations have two reactions :

1. If a time-out occurs, there is a stronger possibility of congestion; a segment has probably been dropped in the network, and there is no news about the sent segments.

In this case TCP reacts strongly:

a. It sets the value of the threshold to one-half of the current window size.

b. It sets cwnd to the size of one segment.

c. It starts the slow-start phase again.

2. If three ACKs are received, there is a weaker possibility of congestion; a segment may have been dropped, but some segments after that may have arrived safely since three ACKs are received. This is called fast transmission and fast recovery.

In this case, TCP has a weaker reaction:

a. It sets the value of the threshold to one-half of the current window size.

b. It sets cwnd to the value of the threshold.

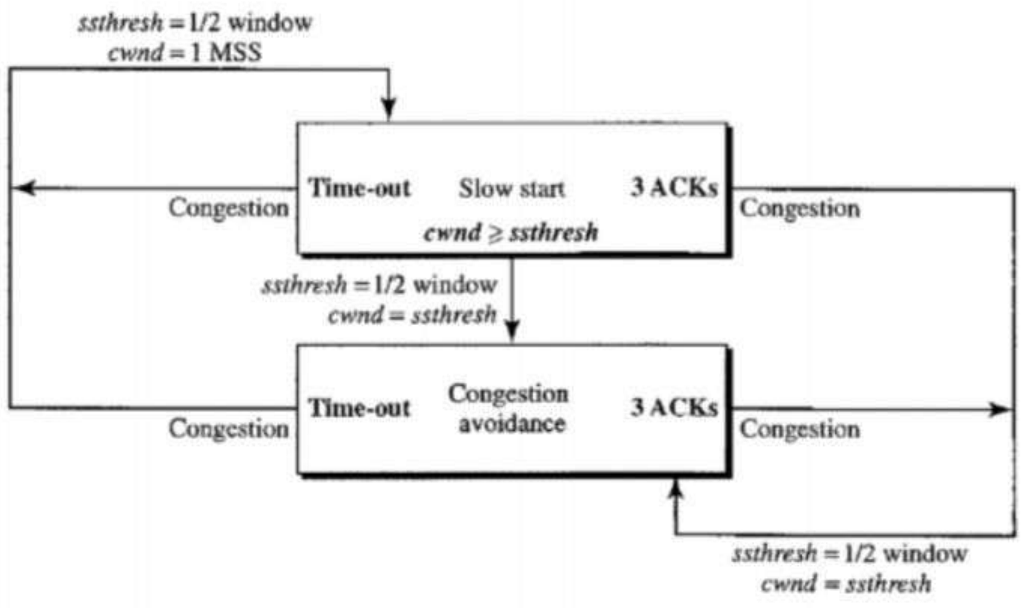
c. It starts the congestion avoidance phase.

An implementations reacts to congestion detection in one of the following ways :

If detection is by time-out, a new slow-start phase starts.

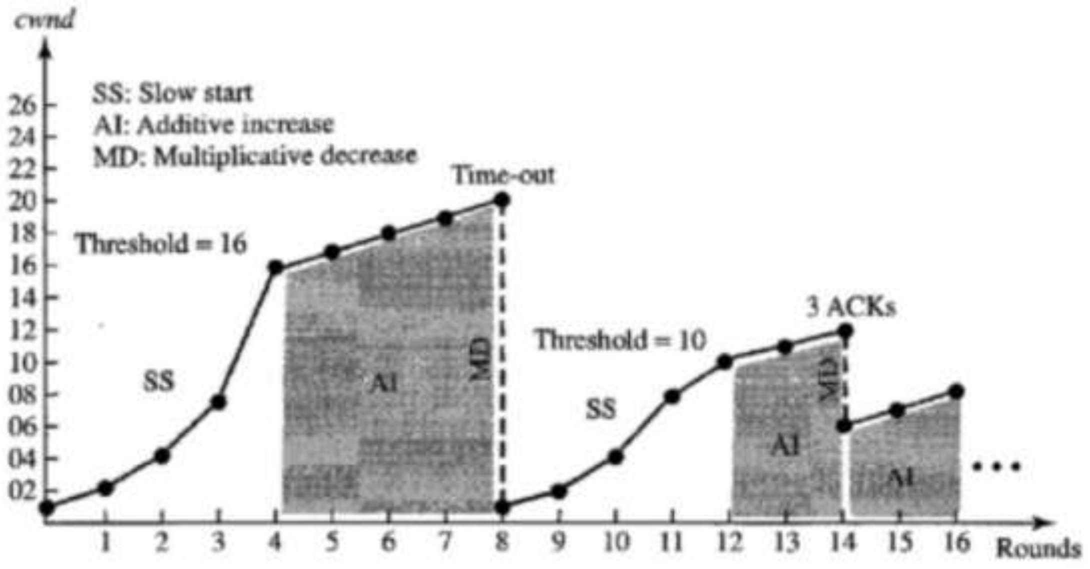
If detection is by three ACKs, a new congestion avoidance phase starts.

Figure 24.10 *TCP congestion policy summary*



Example : We assume that the maximum window size is 32 segments. The threshold is set to 16 segments (one-half of the maximum window size). In the slow-start phase the window size starts from 1 and grows exponentially until it reaches the threshold. After it reaches the threshold, the congestion avoidance (additive increase) procedure allows the window size to increase linearly until a timeout occurs or the maximum window size is reached. In Figure 24.11, the time-out occurs when the window size is 20. At this moment, the multiplicative decrease procedure takes over and reduces the threshold to one-half of the previous window size. The previous window size was 20 when the time-out happened so the new threshold is now 10. TCP moves to slow start again and starts with a window size of 1, and TCP moves to additive increase when the new threshold is reached. When the window size is 12, a three duplicate ACKs event happens. The multiplicative decrease procedure takes over again. The threshold is set to 6 and TCP goes to the additive increase phase this time. It remains in this phase until another time- out or another three duplicate ACKs happen.

Figure 24.11 *Congestion example*



Fast Recovery :

The **fast-recovery** algorithm is optional in TCP. The old version of TCP did not use it, but the new versions try to use it. It starts when three duplicate ACKs arrive, which is interpreted as light congestion in the network. Like congestion avoidance, this algorithm is also an additive increase, but it increases the size of the congestion window when a duplicate ACK arrives (after the three duplicate ACKs that trigger the use of this algorithm).

We can say :

" If a duplicate ACK arrives, cwnd 5 cwnd 1 (1 / cwnd) "

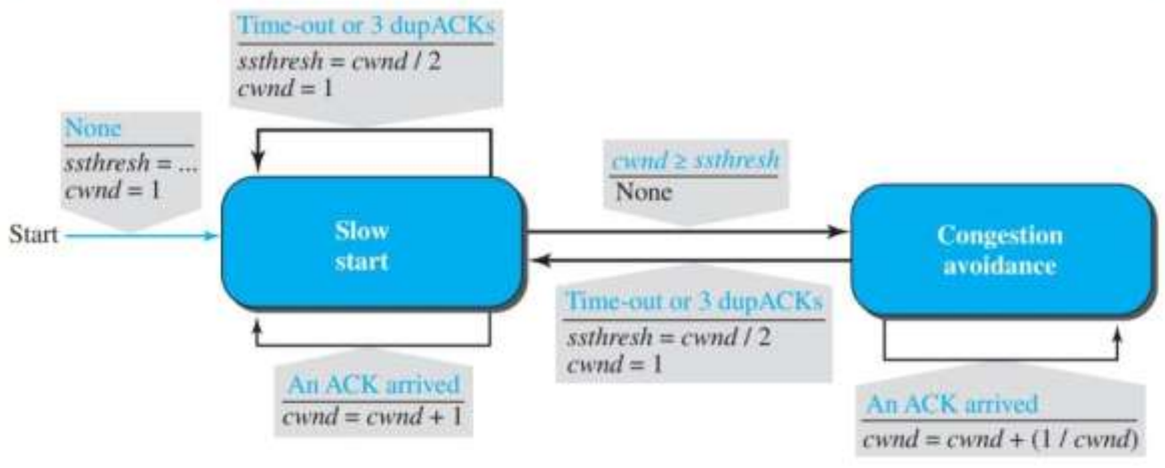
Policy Transition :

We discussed three congestion policies in TCP. Now the question is when each of these policies is used and when TCP moves from one policy to another. To answer these questions, we need to refer to three versions of TCP: Taho TCP, Reno TCP, and New Reno TCP.

Taho TCP :

The early TCP, known as Taho TCP, used only two different algorithms in their congestion policy: slow start and congestion avoidance. We use Figure 24.31 to show the FSM for this version of TCP. However, we need to mention that we have deleted some small trivial actions, such as incrementing and resetting the number of duplicate ACKs, to make the FSM less crowded and simpler.

Figure 24.31 *FSM for Taho TCP*



Taho TCP treats the two signs used for congestion detection, time-out and three duplicate ACKs, in the same way. In this version, when the connection is established, TCP starts the slow-start algorithm and sets the ssthresh variable to a pre-agreed value (normally a multiple of MSS) and the cwnd to 1 MSS. In this state, as we said before, each time an ACK arrives, the size of the congestion window is incremented by 1. We know that this policy is very aggressive and exponentially increases the size of the window, which may result in congestion. If congestion is detected (occurrence of time-out or arrival of three duplicate ACKs), TCP immediately interrupts this aggressive growth and restarts a new slow start algorithm by limiting the threshold to half of the current cwnd and resetting the congestion window to 1. In other words, not only does TCP restart from scratch, but it also learns how to adjust the threshold. If no congestion is detected while reaching the threshold, TCP learns that the ceiling of its ambition is reached; it should not continue at this speed. It moves to the congestion avoidance state and continues in that state.

In the congestion-avoidance state, the size of the congestion window is increased by 1 each time a number of ACKs equal to the current size of the window has been received. For example, if the window size is now 5 MSS, five more ACKs should be received before the size of the window becomes 6 MSS. Note that there is no ceiling for the size of the congestion window in this state; the conservative additive growth of the congestion window continues to the end of the data transfer phase unless congestion is detected.

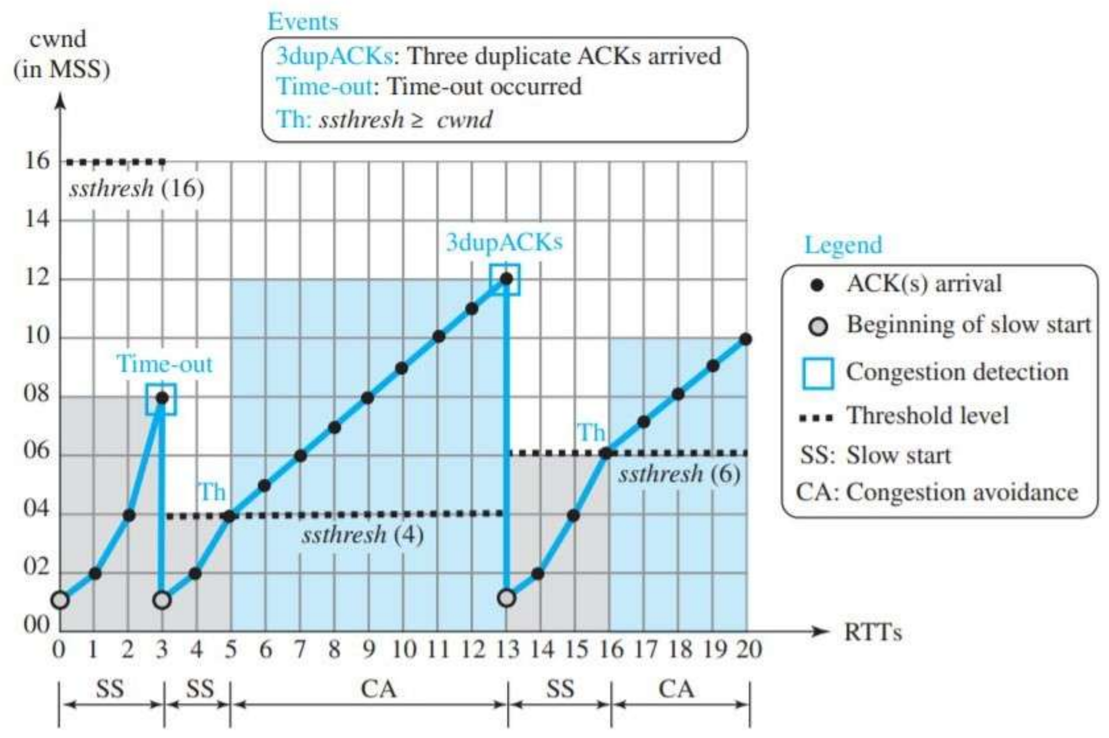
If congestion is detected in this state, TCP again resets the value of the ssthresh to half of the current cwnd and moves to the slow-start state again. Although in this version of TCP the size of ssthresh is continuously adjusted in each congestion detection, this does not mean that it necessarily becomes lower than the previous value. For example, if the original ssthresh value is 8 MSS and congestion is detected when TCP is in the congestion avoidance state and the value of the cwnd is 20, the new value of the ssthresh is now 10, which means it has been increased.

Example :

Figure 24.32 shows an example of congestion control in a Taho TCP. TCP starts data transfer and sets the ssthresh variable to an ambitious value of 16 MSS. TCP begins at the slow-start (SS) state with the cwnd = 1. The congestion window grows exponentially, but a time-out occurs after the third RTT (before reaching the threshold). TCP assumes that there is congestion in the network. It immediately sets the new ssthresh = 4 MSS (half of the current cwnd, which is 8) and begins a new slow-start (SA) state with cwnd = 1 MSS. The congestion window grows exponentially until it reaches the newly set threshold. TCP now moves to the congestion-avoidance (CA) state and the congestion window grows additively until it reaches cwnd = 12 MSS. At this moment, three duplicate ACKs arrive, another indication of congestion in the network. TCP again halves the value of ssthresh to 6 MSS and begins a new slow-start

(SS) state. The exponential growth of the cwnd continues. After RTT 15, the size of cwnd is 4 MSS. After sending four segments and receiving only two ACKs, the size of the window reaches the ssthresh (6) and TCP moves to the congestion-avoidance state. The data transfer now continues in the congestion avoidance (CA) state until the connection is terminated after RTT 20.

Figure 24.32 *Example 24.9*



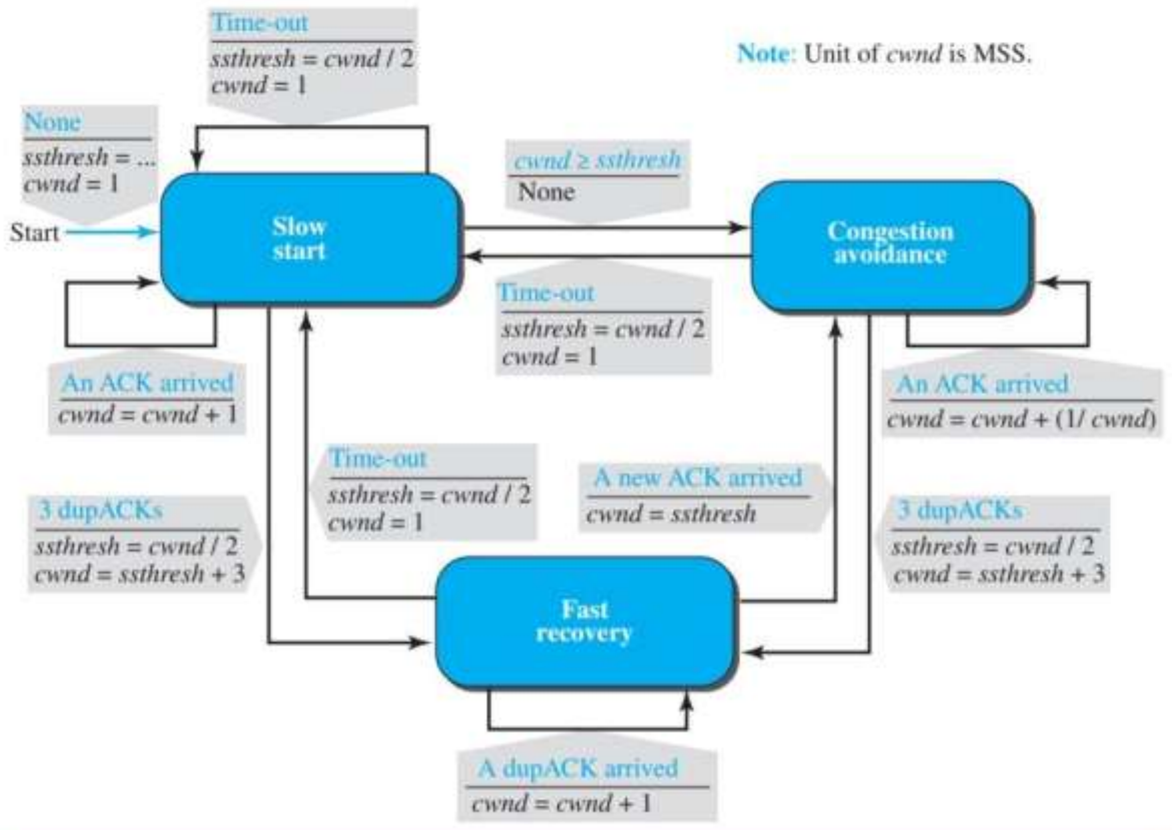
Reno TCP :

A newer version of TCP, called Reno TCP, added a new state to the congestion-control FSM, called the fast-recovery state. This version treated the two signals of congestion,time-out and the arrival of three duplicate ACKs, differently. In this version, if a time-out occurs, TCP moves to the slow-start state (or starts a new round if it is already in this state); on the other hand, if three duplicate ACKs arrive, TCP moves to the fast-recovery state and remains there as long as more duplicate ACKs arrive. The fast-recovery state is a state somewhere between the slow-start and the congestion-avoidance states. It behaves like the slow start, in which the cwnd grows exponentially, but the cwnd starts with the value of ssthresh plus 3 MSS (instead of 1). When TCP enters the fast-recovery state, three major events may occur. If duplicate ACKs continue to arrive, TCP stays in this state, but the cwnd grows exponentially. If a time-out occurs, TCP assumes that there is real congestion in the network and moves to the slow-start state. If a new (non duplicate) ACK arrives, TCP moves to the congestion-avoidance state, but deflates the size of the cwnd to the ssthresh value, as though the three duplicate ACKs have not occurred, and transition is from the slow-start state to the congestion-avoidance state. Figure 24.33 shows the simplified FSM for Reno TCP. Again, we have removed some trivial events to simplify the figure and discussion.

Example :

Figure 24.34 shows the same situation as Figure 24.32, but in Reno TCP. The changes in the congestion window are the same until RTT 13 when three duplicate ACKs arrive. At this moment, Reno TCP drops the ssthresh to 6 MSS (same as Taho TCP), but it sets the cwnd to a much higher value (ssthresh + 3 = 9 MSS) instead of 1 MSS. Reno TCP now moves to the fast recovery state. We assume that two more duplicate ACKs arrive until RTT 15, where cwnd grows exponentially. In this moment, a new ACK (not duplicate) arrives that announces the receipt of the lost segment. Reno TCP now moves to the congestion-avoidance state, but first deflates the congestion window to 6 MSS (the ssthresh value) as though ignoring the whole fast-recovery state and moving back to the previous track.

Figure 24.33 *FSM for Reno TCP*



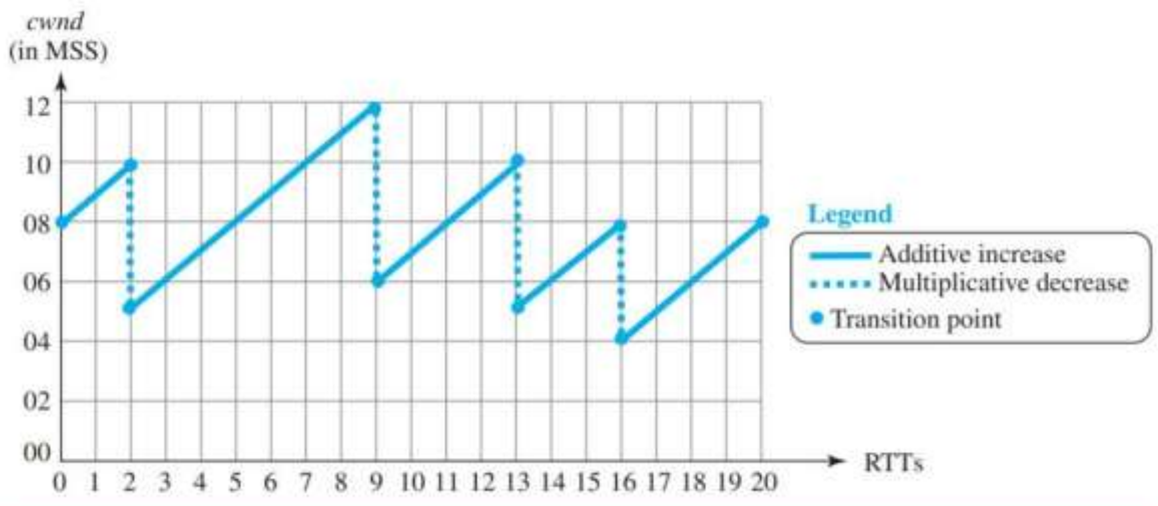
New Reno TCP :

A later version of TCP, called NewReno TCP, made an extra optimization on the Reno TCP. In this version, TCP checks to see if more than one segment is lost in the current window when three duplicate ACKs arrive. When TCP receives three duplicate ACKs, it retransmits the lost segment until a new ACK (not duplicate) arrives. If the new ACK defines the end of the window when the congestion was detected, TCP is certain that only one segment was lost. However, if the ACK number defines a position between the retransmitted segment and the end of the window, it is possible that the segment defined by the ACK is also lost. NewReno TCP retransmits this segment to avoid receiving more and more duplicate ACKs for it.

Additive Increase, Multiplicative Decrease :

Out of the three versions of TCP, the Reno version is most common today. It has been observed that, in this version, most of the time the congestion is detected and taken care of by observing the three duplicate ACKs. Even if there are some time-out events, TCP recovers from them by aggressive exponential growth. In other words, in a long TCP connection, if we ignore the slow-start states and short exponential growth during fast recovery, the TCP congestion window is cwnd = cwnd + (1 / cwnd) when an ACK arrives (congestion avoidance), and cwnd = cwnd / 2 when congestion is detected, as though SS does not exist and the length of FR is reduced to zero. The first is called additive increase; the second is called multiplicative decrease. This means that the congestion window size, after it passes the initial slow-start state, follows a saw tooth pattern called additive increase, multiplicative decrease (AIMD), as shown in Figure 24.35.

Figure 24.35 *Additive increase, multiplicative decrease (AIMD)*



Comparison Among TCP Taho , Reno TCP and New Reno TCP :

Table 1. The four phases of congestion control algorithm

Phase 1	Slow Start Algorithm
Phase 2	Congestion avoidance Algorithm
Phase 3	Fast retransmission Algorithm
Phase 4	Fast recovery Algorithm

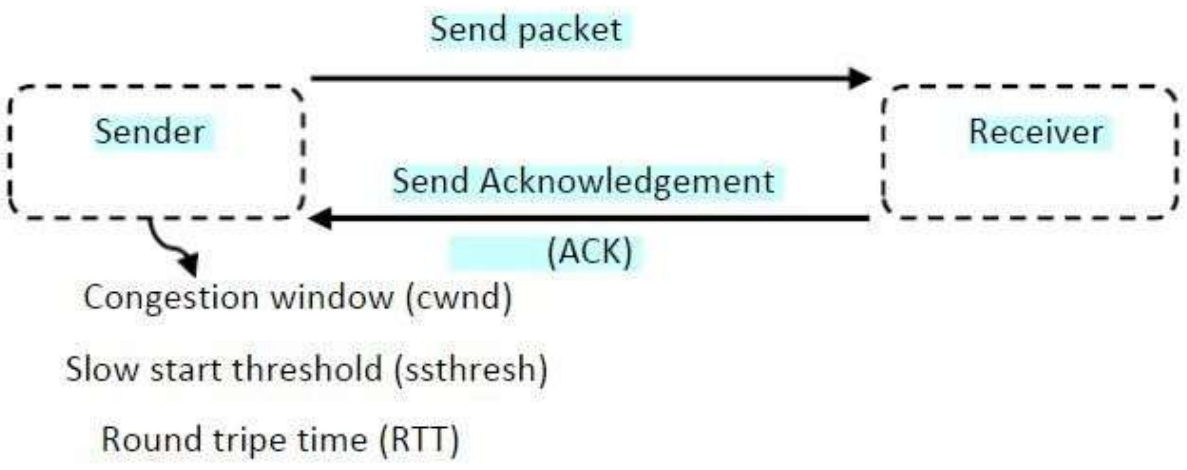


Fig. 1 Common factors used in congestion control

TCP Taho :

TCP Taho uses only the first three algorithm mentioned in Table 1 which are Slow Start, Congestion Avoidance, and Fast Retransmit . To simplify the idea discussed in algorithm steps with and without congestion will be shown here.

A) The procedure without congestion or loss packet will be as following (All terms taken from Fig.1). The procedure start with congestion window of size equal to one packet and any time we have a complete transmission (the source received an acknowledgement for sent packet before timeout status) the congestion window grow by one packet.

cwnd<= ssthresh
Connection start : cwnd=1 packet.
Each ACK : cwnd+=1 packet.

B) The procedure when we have any one of the following states: the congestion window exceeds the ssthresh, timeout, or receives three duplicate acknowledgements by source.

ssthresh = cwnd/2
cwnd = 1
Congestion avoidance
Fast retransmission
Slow start state: when ACK received for retransmitted packet.

TCP Reno :

TCP Reno uses the entire four algorithms for congestion control mentioned in Table 1. The new procedure looks like the one used in TCP Tahoe when respond to timeout. But for three duplicate acknowledgements another procedure will be used which is like following:

ssthresh = cwnd/2.
cwnd = ssthresh + 3. (fast recovery)
first ACK: cwnd +=1.
next ACK cwnd = ssthresh.

The simulation uses two senders and two receivers to test different scenarios of congestion control and Round-trip time RTT. According to the results of simulation, the raise of RTT occurs only when there is a congestion in the network. The raise in RTT property is used to timing the sender with the difference of RTT in current cycle and the previous one. The negative sum of RTT differences means we have no congestion in the current cycle of transmission while the positive sum of RTT differences means the transmission is in congestion state or will have a congestion in next cycles. The algorithm of TCP Reno will be like the following:

```
RTT_diff_sum=0;
when the sender recieves a ACK;
    RTT_diff = RTT - RIT_last;
    RTT_diff_sum + = RTT_diff;
    if(dup_ACKs> =3)
  (if(RTT_diff_sum >O) llin congestion state
    {ssthresh=min(cwnd,rcv_window )/2;
cwnd = ssthresh+ 3*MSS; //lower transmission rate
    retransmit the lost packet;
    } else  //in non-congestion state
    { only retransmit the lost packet;
```

```
        }
       }
```

TCP Reno is not the best choice to use with wireless network. The new procedure doesn't provide a way to differentiate between packet loss caused by congestion in network and packet loss when network suffer from random bit error in wireless links.

TCP New Reno :

A very novel TCP version has been proposed. The design provides a new mechanism to differentiate between packets loss caused by high bit error and packets loss caused by network congestion. Sender, receiver and middle router all of these parts cooperate to detect congestion and control it.

The new modified Transmission Control Protocol (TCP Reno) is able to monitor the loss of wireless packets in real time. By detecting a router's buffer mechanism in response to congestion occupancy, the modified Reno is able to monitor wireless packet loss; thus being able to react accordingly and decrease the rate of wireless package loss. This is important because when high volumes of information packets are sent it can have problems reaching the desired recipient. Communication over wireless links, between computer networks and various systems, is filled with random rates of high bit error and connectivity that is intermittent due to the frequency of handoffs.

Mechanisms such as random early detection (RED) were used to find possible problematic packets in their early stages.

The modified TCP Reno uses Explicit Congestion Notification (ECN), which is an extension of REDs. This mechanism can allow the system to properly tell the difference between random wireless link errors and errors caused by the congestion of network links. It can also monitor the rate at which wireless packets are lost in a manner that helps the sender select the appropriate segment size at the right moment when packet loss is identified. These modifications to Reno, after plenty of tests and simulations, have shown to have merit and even improve the TCP efficiency; it can do this with no changes in the protocol itself which makes the modified Reno easy enough to use.

WIRELESS APPLICATION PROTOCOL (WAP)

Dr. A. Beulah

AP/CSE

Introduction

- 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 + …)

Introduction

- A protocol suite should enable global wireless communication across different wireless network technologies, e.g., GSM, CDPD, UMTS etc.
- **Interoperable**
  - allowing terminals and software from different vendors to communicate with networks from different providers;
- **Scalable**
  - protocols and services should scale with customer needs and number of customers;
- **Efficient**
  - provision of QoS suited to the characteristics of the wireless and mobile networks;
- **Reliable**
  - provision of a consistent and predictable platform for deploying services;
- **Secure**
  - preservation of the integrity of user data, protection of devices and services from security problems.

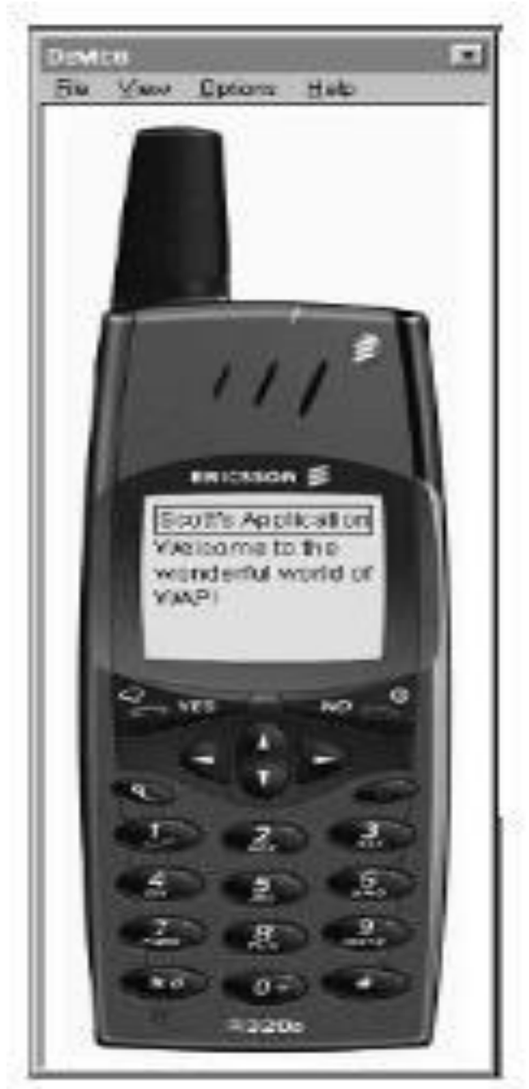
WAP - scope of standardization

- 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 Device Characteristics

- WAP device has limited CPU power, RAM, and ROM.
- Exact numbers are not given in the specification

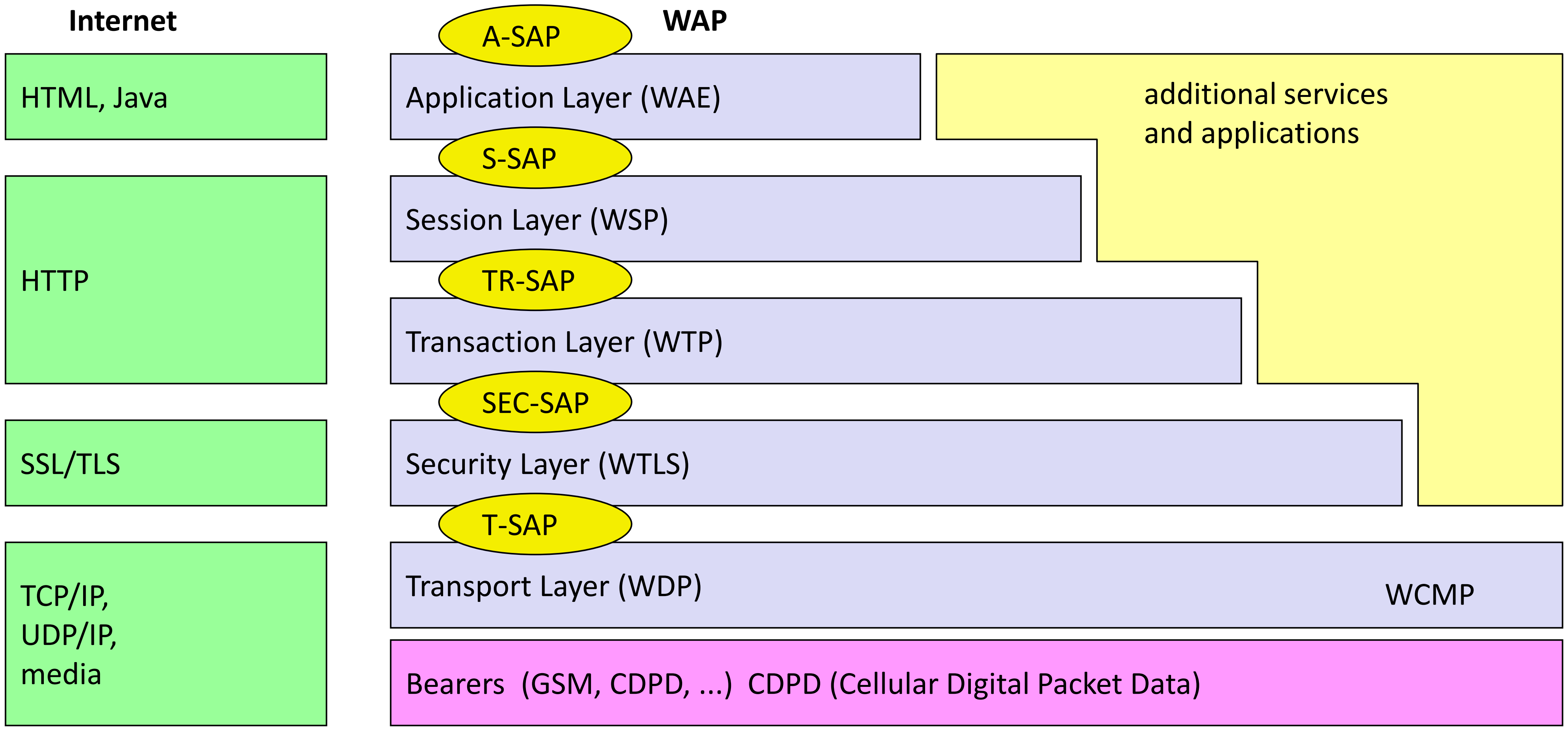
- The telephone has “Yes” and “No” buttons and arrow keys for navigating from one screen to another.



Accessing a Web Site

- Here's what happens when accessing a Web site using a WAP-enabled device:
  - Turn on the device and open the minibrowser.
  - The device sends out a radio signal, searching for service.
  - A connection is made with the service provider.
  - Select a Web site to view.
  - A request is sent to a gateway server using WAP.
  - The gateway server retrieves the information via HTTP from the Web site.
  - The gateway server encodes the HTTP data as WML.
  - The WML-encoded data is sent to the device.
  - The wireless Internet version of the Web page selected is visible now.

WAP Architecture



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

WAP Architecture

- **WAE** (Wireless Application Environment)
  - The Wireless Application Environment holds the tools that wireless Internet content developers use.
  - These include WML and WMLScript, which is a scripting language used in conjunction with WML. It functions much like JavaScript.
- **WSP** (Wireless Session Protocol)
  - The Wireless Session Protocol determines whether a session between the device and the network will be **connection-oriented** or **connectionless**.
  - In a connection-oriented session, data is passed both ways between the device and the network; WSP then sends the packet to the Wireless Transaction Protocol layer
  - If the session is connectionless, commonly used when information is being broadcast or **streamed** from the network to the device, then WSP redirects the packet to the Wireless Datagram Protocol layer

WAP Architecture

- **WTP** (Wireless Transaction Protocol)
  - The **WTP** offers a lightweight transaction service at the **transaction SAP (TR-SAP)** **(**transaction request- Service Access Point**)**
  - It also determines how to classify each transaction request: Reliable two-way, Reliable one-way, Unreliable one-way
- **WTLS** (Wireless Transport Layer Security)
  - The **WTLS offers its service at the security SAP (SEC-SAP).**
  - **WTLS** is based on the transport layer security (TLS, formerly SSL, secure sockets layer)
  - WTLS has been optimized for use in wireless networks
  - It can offer data integrity, privacy, authentication, and (some) denial-of-service protection.

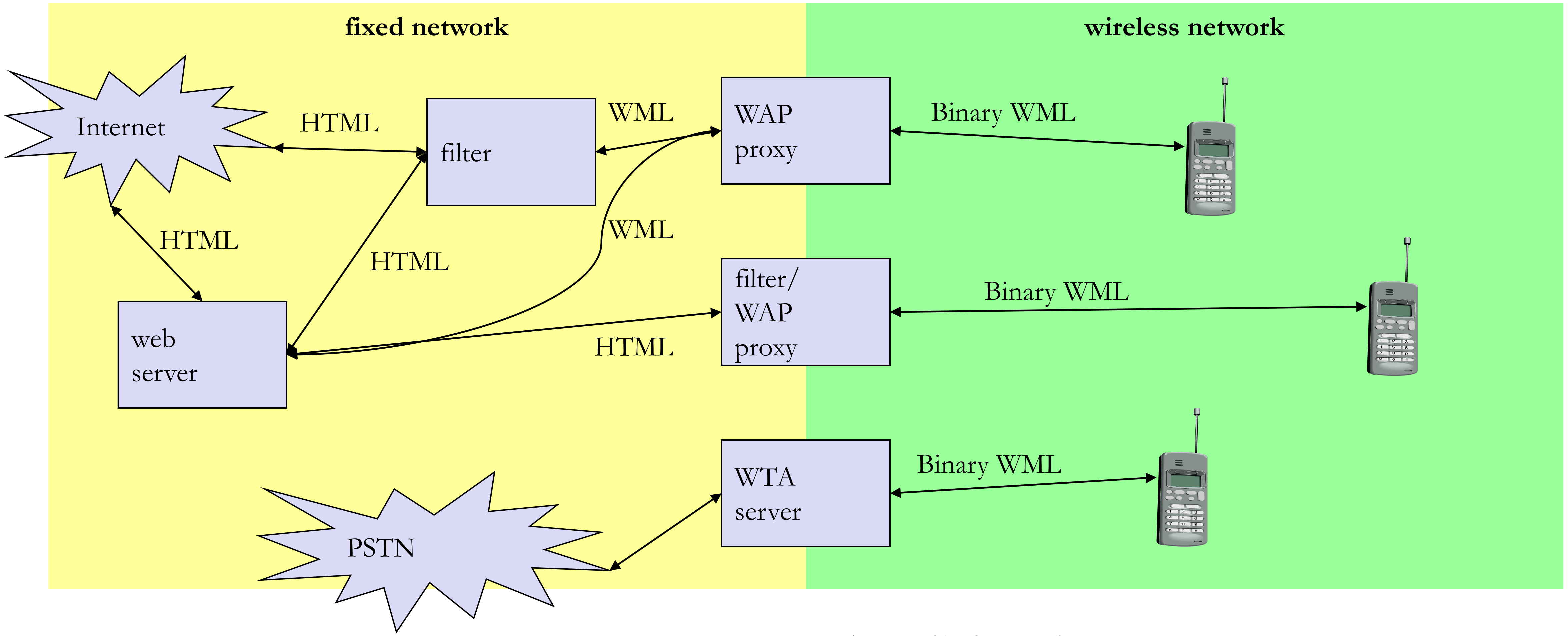
WAP Architecture

- **WDP** (Wireless Datagram Protocol)
  - works in conjunction with the network **bearer** layer
  - WDP makes it easy to adapt WAP to a variety of bearers because all that needs to change is the information maintained at this level.
  - Communication is done transparently over one of the available bearer services.
  - The **transport layer service access point (T-SAP) is the** common interface to be used by higher layers independent of the underlying network

WAP Architecture

- **Network bearers**
  - Also called **bearers**, these can be any of the existing technologies that wireless providers use.
  - The basis for transmission of data is formed by different **bearer services.**
  - **WAP** does not specify bearer services, but uses existing data services and will integrate further services.
  - Examples are message services, such as:
    - short message service (SMS) of GSM,
    - circuit-switched data, such as high-speed circuit switched data (HSCSD) in GSM
    - packet switched data, such as general packet radio service (GPRS) in GSM

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, ...)

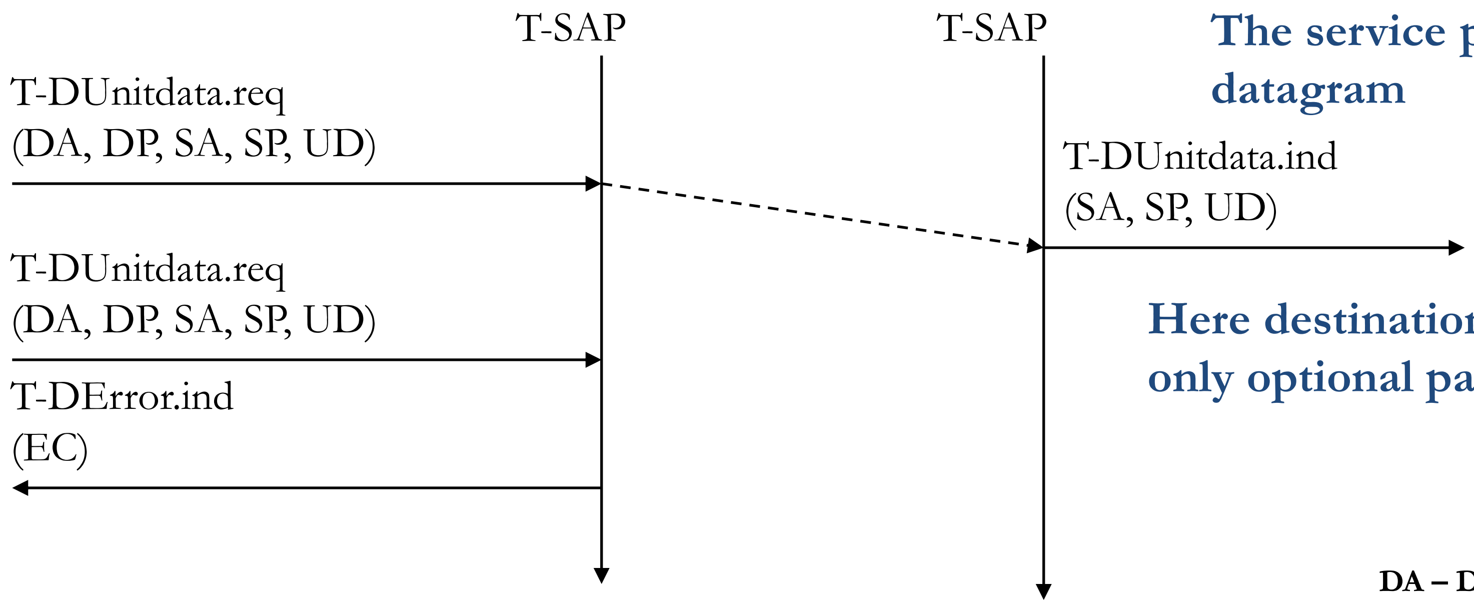
- Goals of WDP
  - create a worldwide interoperable transport system with the help of WDP adapted to the different underlying technologies
  - transmission services such as SMS, GPRS in GSM might change, new services can replace the old ones

WDP - Wireless Datagram Protocol

- WDP offers source and destination port numbers used for multiplexing and demultiplexing of data respectively.
- The service primitive to send a datagram **is TDUnitdata.req** with the destination address (DA), destination port (DP), Source address (SA), source port (SP), and user data (UD) as mandatory parameters
- Destination and source address are unique addresses for the receiver and sender of the user data.
- These could be MSISDNs (i.e., a telephone number), IP addresses, or any other unique identifiers.
- The **T-DUnitdata.ind** service primitive indicates the reception of data. Here destination address and port are only optional parameters.
- If a higher layer requests a service the WDP cannot fulfill, this error is indicated with the **T-DError.ind** service primitive.
- An error code (EC) is returned indicating the reason for the error to the higher layer.

WDP - Service Primitives

The service primitive to send a datagram



The service primitive to receive a datagram

Here destination address and port are only optional parameters

DA –	**Destination Address**
DP –	**Destination Port**
SA –	**Source Address**
SP –	**Source Port**
UD –	**User Data**
EC –	**Error Code**

WDP - Wireless Datagram Protocol

- Additionally, WCMP (Wireless Control Message Protocol) is used for control/error report (similar to ICMP in the TCP/IP protocol suite)
- Typical WCMP messages are
  - destination unreachable (route, port, address unreachable),
  - parameter problem (errors in the packet header),
  - message too big,
  - reassembly failure,
  - echo request/reply.

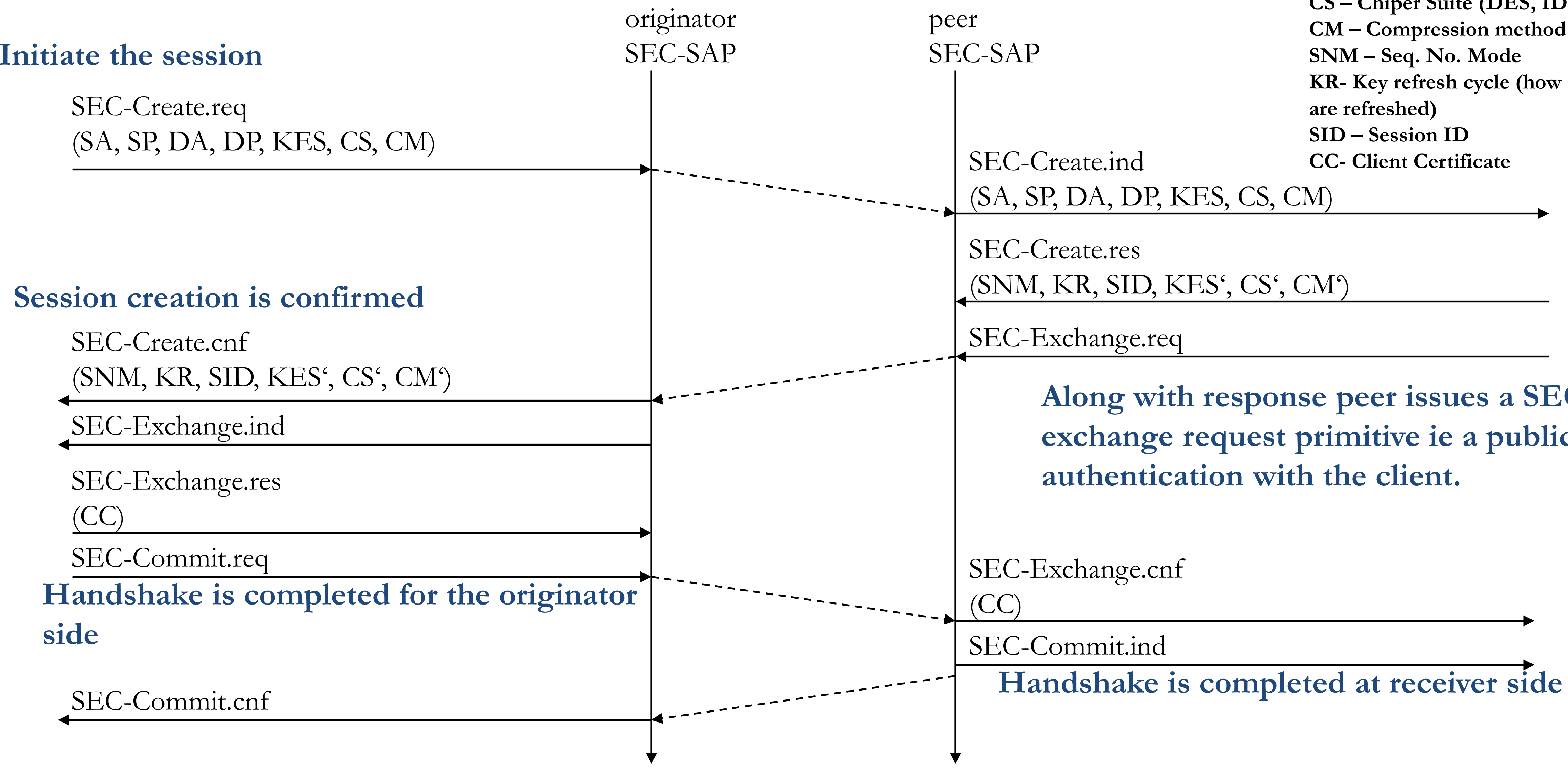
WTLS (Wireless Transport Layer Security)

- 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
  - Before data can be exchanged via WTLS, a secure session has to be established.

WTLS (Wireless Transport Layer Security)

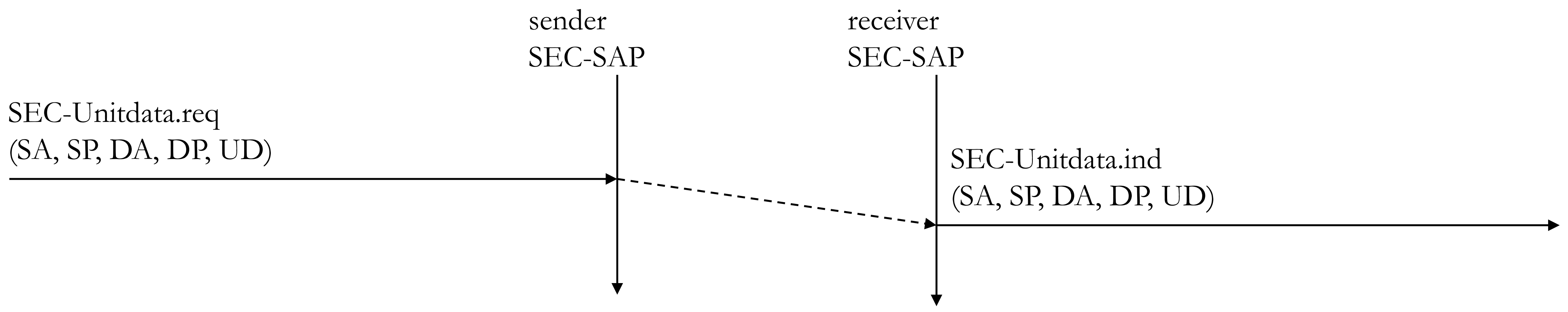
KES- Key exchange suite (RSA, DH, ECC)
CS – Chiper Suite (DES, IDEA)
CM – Compression method
SNM – Seq. No. Mode
KR- Key refresh cycle (how often the keys are refreshed)
SID – Session ID
CC- Client Certificate


originator SEC-SAP
peer SEC-SAP
Initiate the session
SEC-Create.req (SA, SP, DA, DP, KES, CS, CM)
SEC-Create.ind (SA, SP, DA, DP, KES, CS, CM)
SEC-Create.res (SNM, KR, SID, KES‘, CS‘, CM‘)
SEC-Exchange.req
Session creation is confirmed
SEC-Create.cnf (SNM, KR, SID, KES‘, CS‘, CM‘)
SEC-Exchange.ind
Along with response peer issues a SEC-exchange request primitive ie a public key authentication with the client.
SEC-Exchange.res (CC)
SEC-Commit.req
Handshake is completed for the originator side
SEC-Exchange.cnf (CC)
SEC-Commit.ind
Handshake is completed at receiver side
SEC-Commit.cnf

WTLS establishing a secure session

WTLS (Wireless Transport Layer Security)

WTLS datagram transfer



Wireless Transaction Protocol (WTP)

- Goals
  - Different transaction services, offloads applications
    - Advantages to higher layers: reliability over datagram services, improved efficiency over connection oriented services.
  - Support of different communication scenarios
    - class 0: unreliable message transfer (unreliable one way)
    - class 1: reliable message transfer without result message (reliable one way)
    - class 2: reliable message transfer with exactly one reliable result message (reliable two way)
  - 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)

Wireless Transaction Protocol (WTP)

- 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

Wireless Transaction Protocol (WTP)

- 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)
- PDUs exchanged between two WTP entities
  - invoke PDU, ack PDU, and result PDU

Wireless Transaction Protocol (WTP)

WTP Class 0 transaction

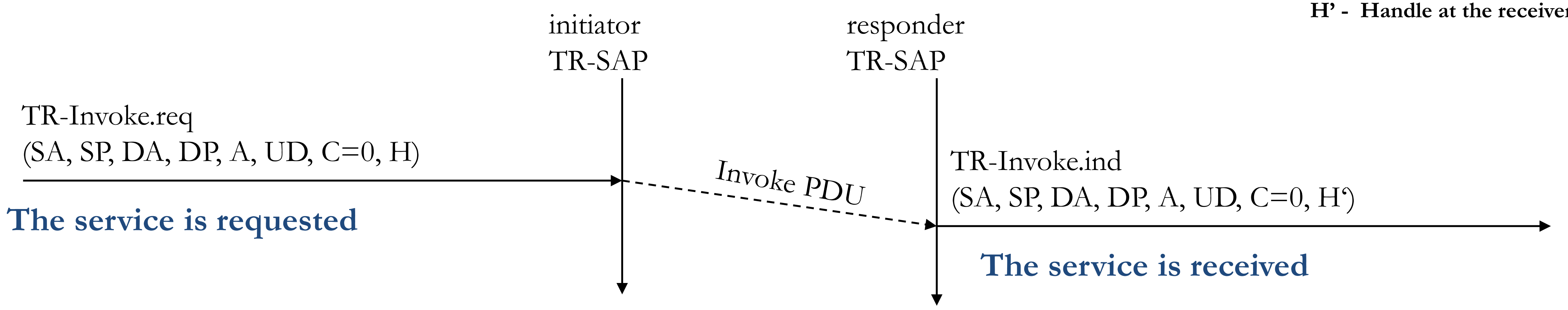
A – flag – If the responder should generate an ACK or not

C – Class

H – Handle – Unique identifier for the transaction

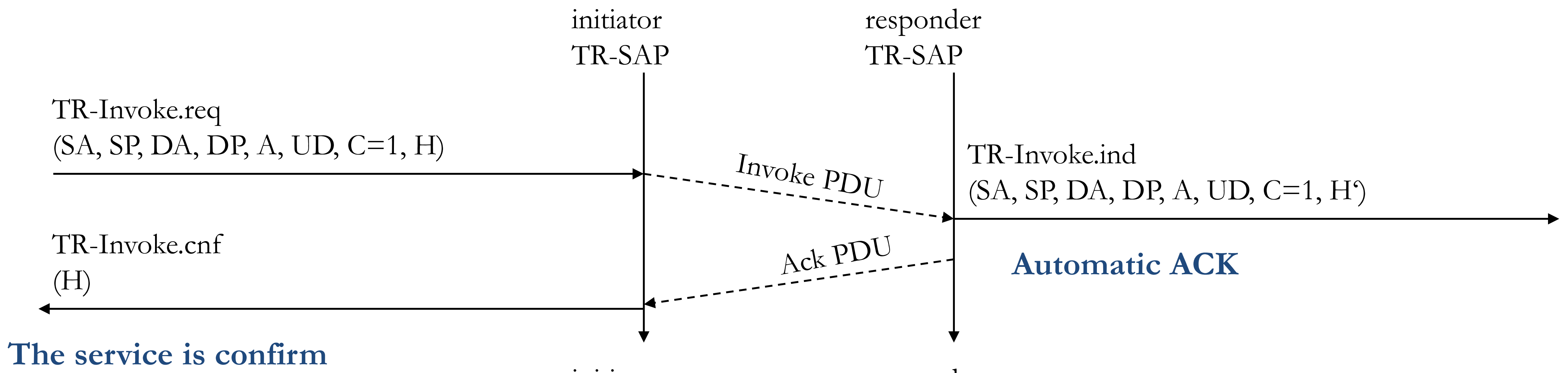
H' - Handle at the receiver side

Unreliable service without result message

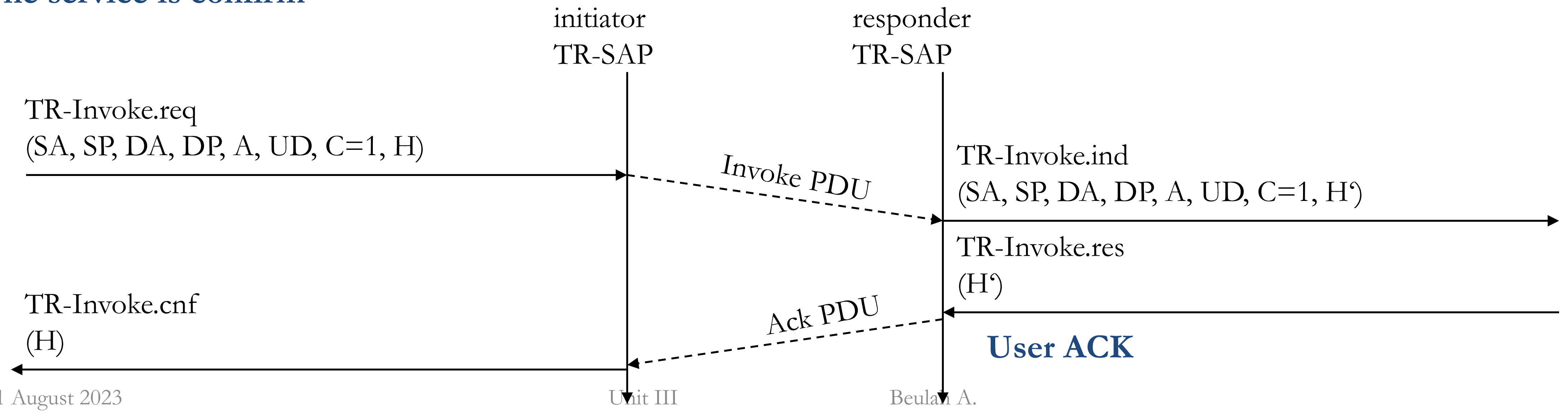


Wireless Transaction Protocol (WTP)

WTP Class 1 transaction, Automatic ack & user ack

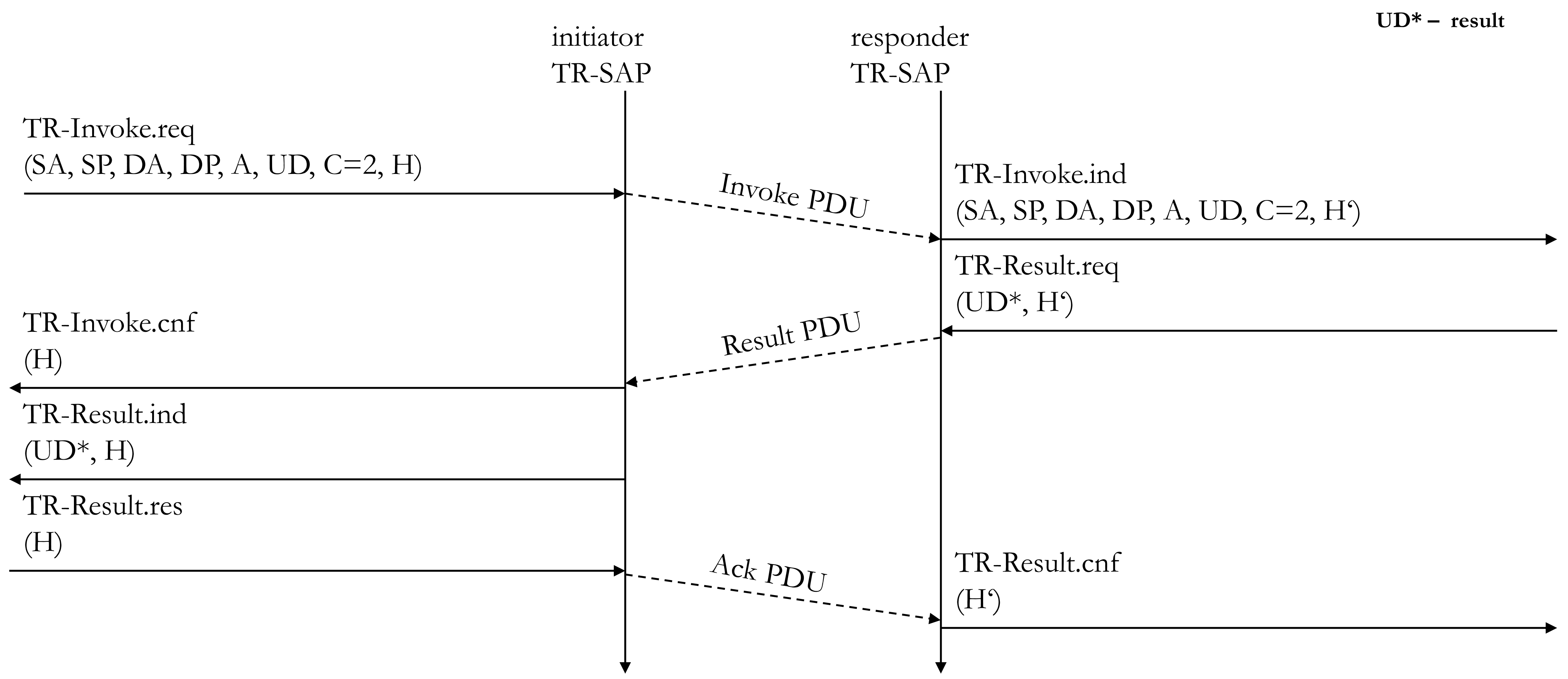


The service is confirm



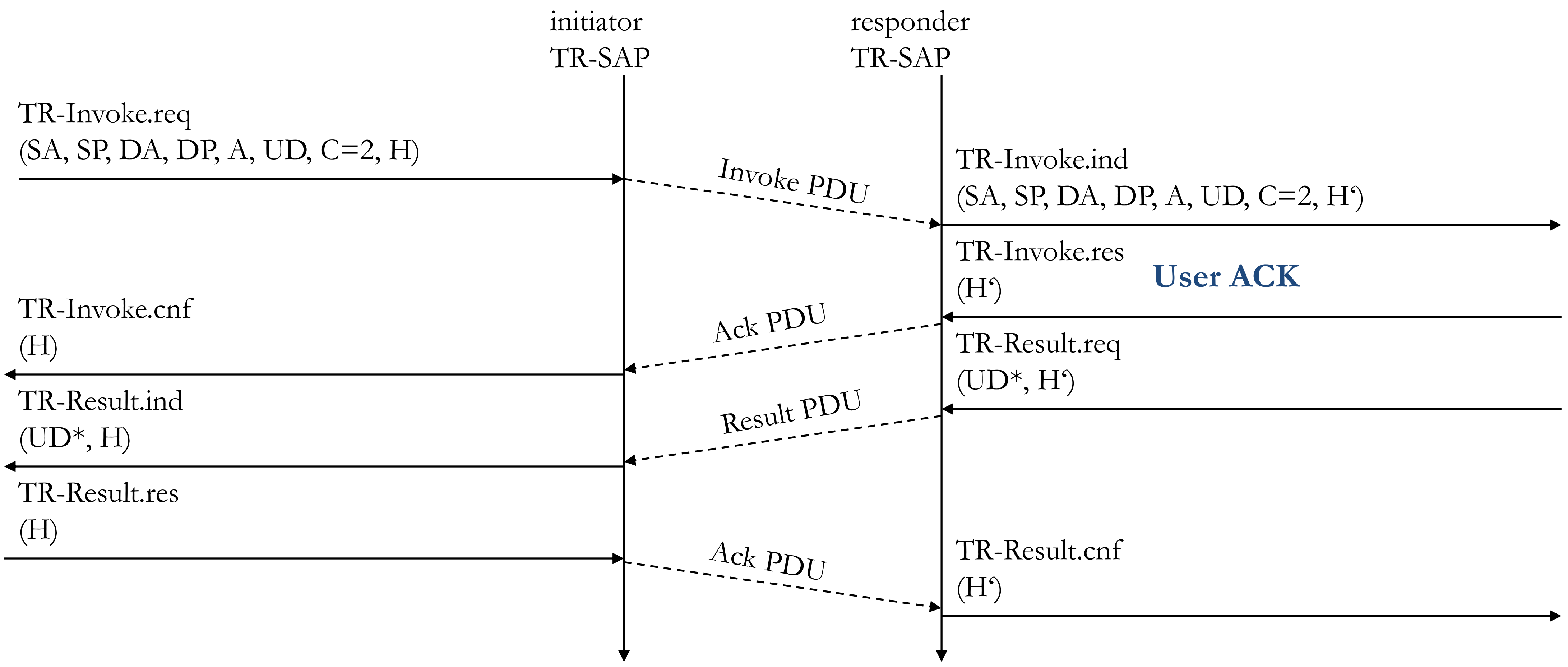
Wireless Transaction Protocol (WTP)

WTP Class 2 transaction, no user ack



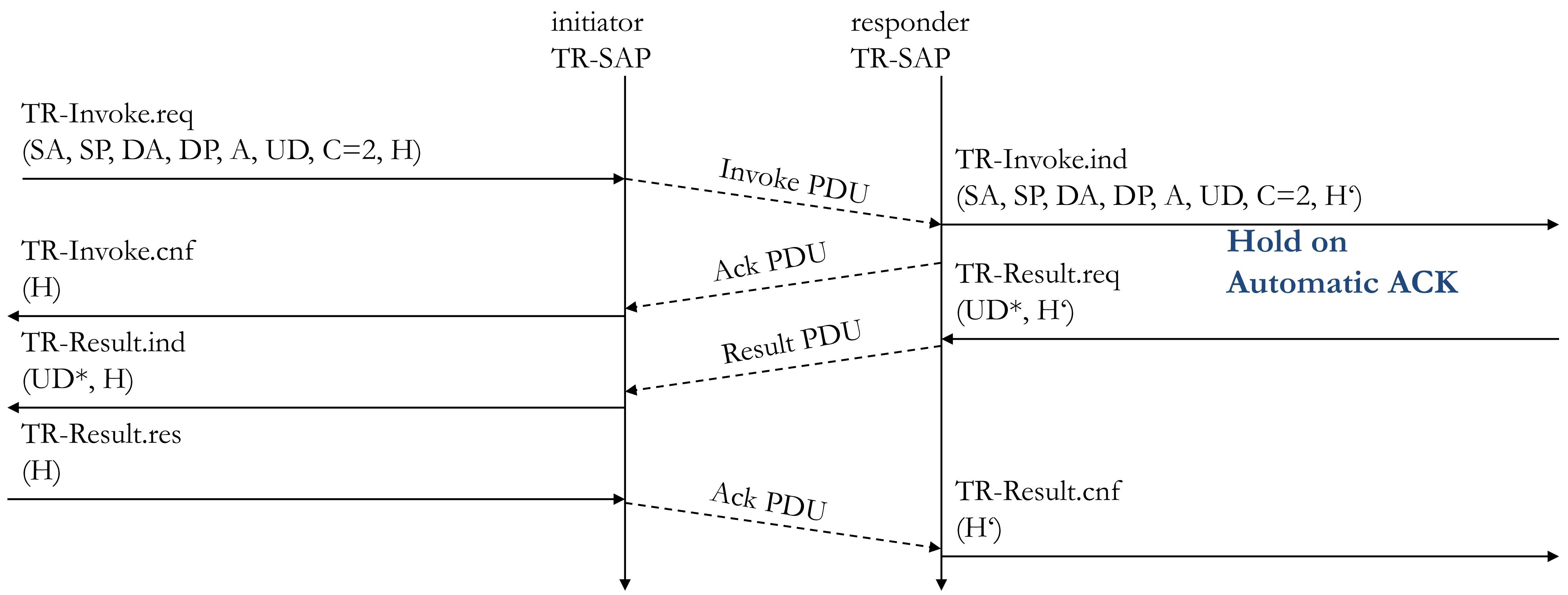
Wireless Transaction Protocol (WTP)

WTP Class 2 transaction, user ack



Wireless Transaction Protocol (WTP)

WTP Class 2 transaction, hold on, no user ack



Hold on → to prevent a retransmission of the invoke PDU (If no result is sent before the timer expire)

Wireless Session Protocol (WSP)

- WSP → general purpose session protocol
- WSP/B → protocols and services suitable for browsing
- 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

Wireless Session Protocol (WSP)

- WSP/B (browsing) over WTP
  - Uses 3 services classes of WTP
  - Class 0 is used for:
    - Unconfirmed push
    - Session resume, and
    - Session management.
  - Class 1 is used for:
    - Confirmed push
  - Class 2 is used for:
    - Confirmed push's
      - method invocation,
      - Session resume
      - Session management

Wireless Session Protocol (WSP)

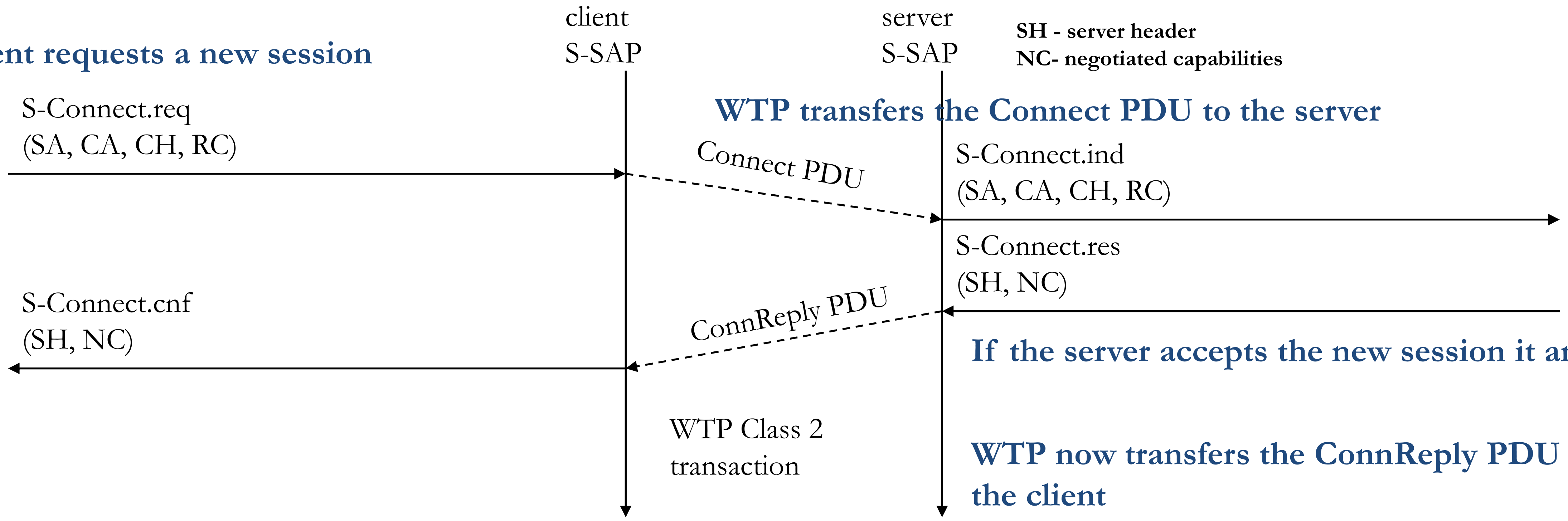
WSP/B session establishment using WTP class 2

SA server address
CA - client address
CH - optional client header
User to user information compatible with HTTP header

RC- requested capabilities
S&C SDU size, outstanding request, protocol options

SH - server header
NC- negotiated capabilities

Client requests a new session



WTP transfers the Connect PDU to the server

If the server accepts the new session it answers

WTP now transfers the ConnReply PDU back to the client

Wireless Session Protocol (WSP)

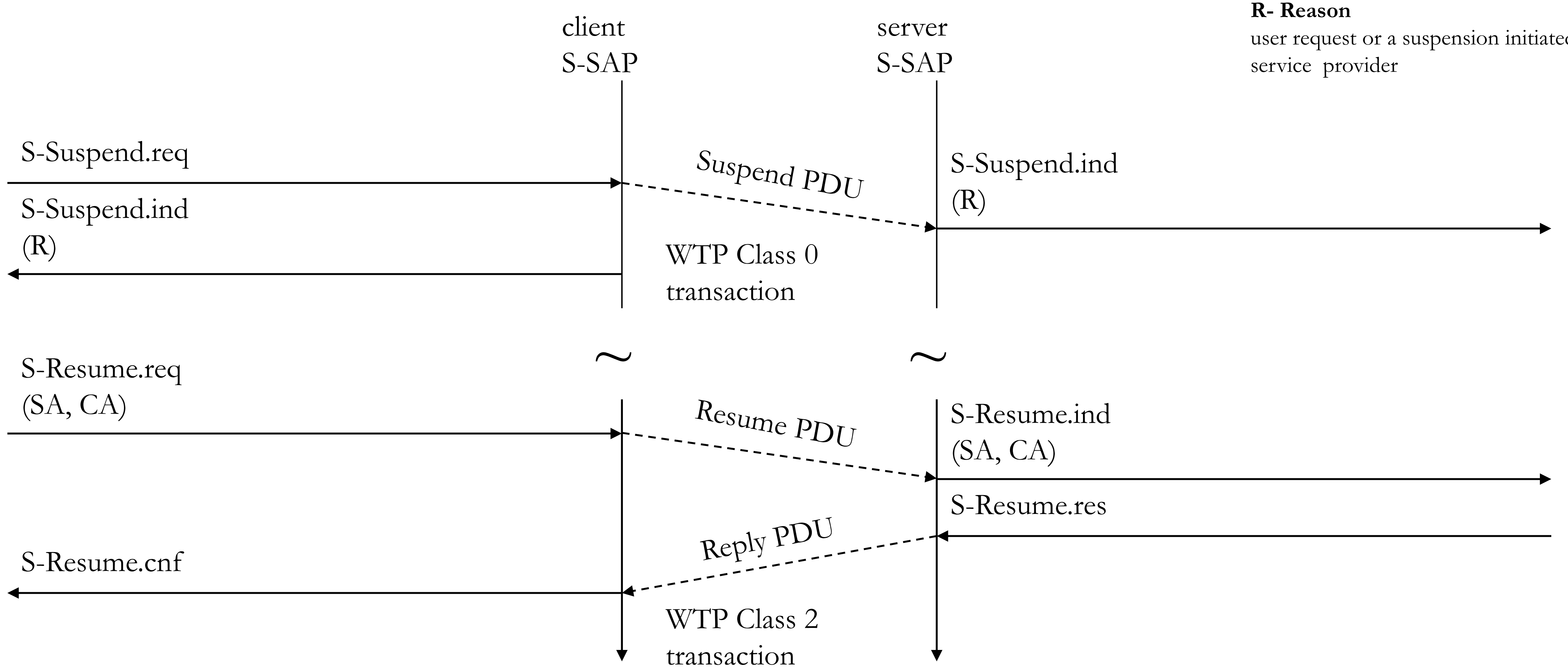
WSP/B session suspend/resume

- A client notices that it will soon be unavailable,
  - Ex: The bearer network will be unavailable due to roaming to another network
  - or the user switches off the device,
- So, the client can suspend the session.
- Session suspension will automatically abort all data transmission and freeze the current state of the session on the client and server side.

Wireless Session Protocol (WSP)

WSP/B session suspend/resume

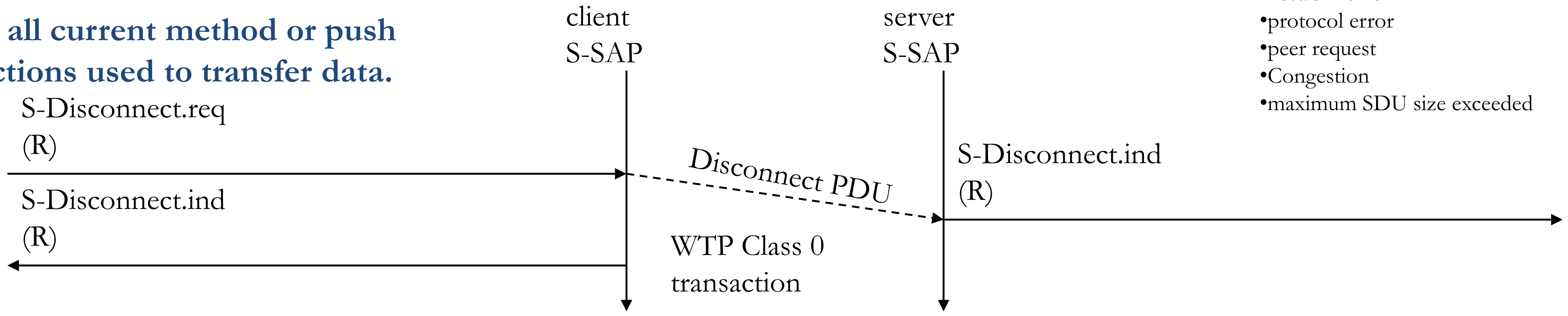
R- Reason
user request or a suspension initiated by the service provider



Wireless Session Protocol (WSP)

WSP/B session termination

Aborts all current method or push transactions used to transfer data.



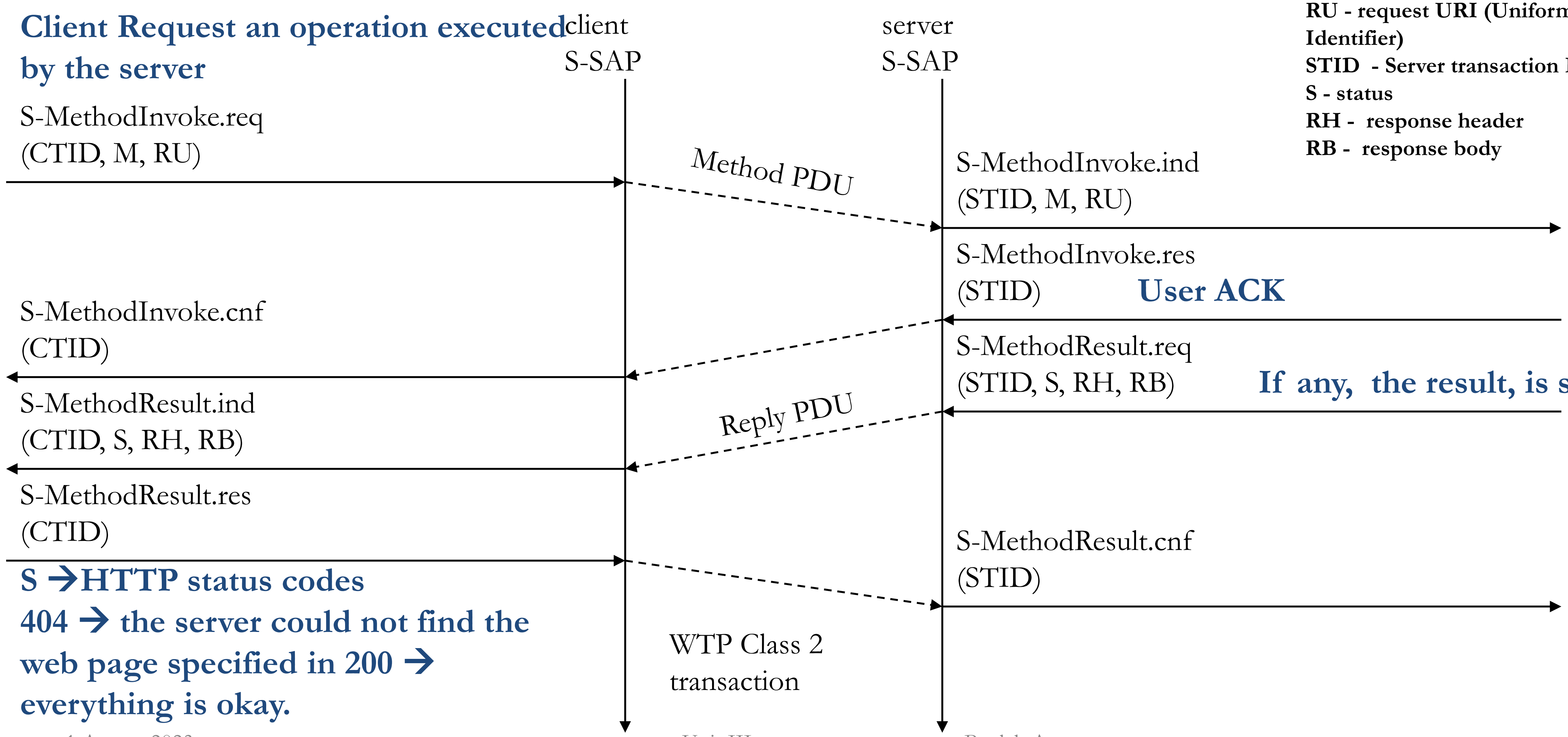
R- Reason
- network error
- protocol error
- peer request
- Congestion
- maximum SDU size exceeded

Wireless Session Protocol (WSP)

WSP/B method invoke

CTID - client transaction identifier (to distinguish between pending transactions)
M - method
RU - request URI (Uniform Resource Identifier)
STID - Server transaction ID
S - status
RH - response header
RB - response body

Client Request an operation executed by the server

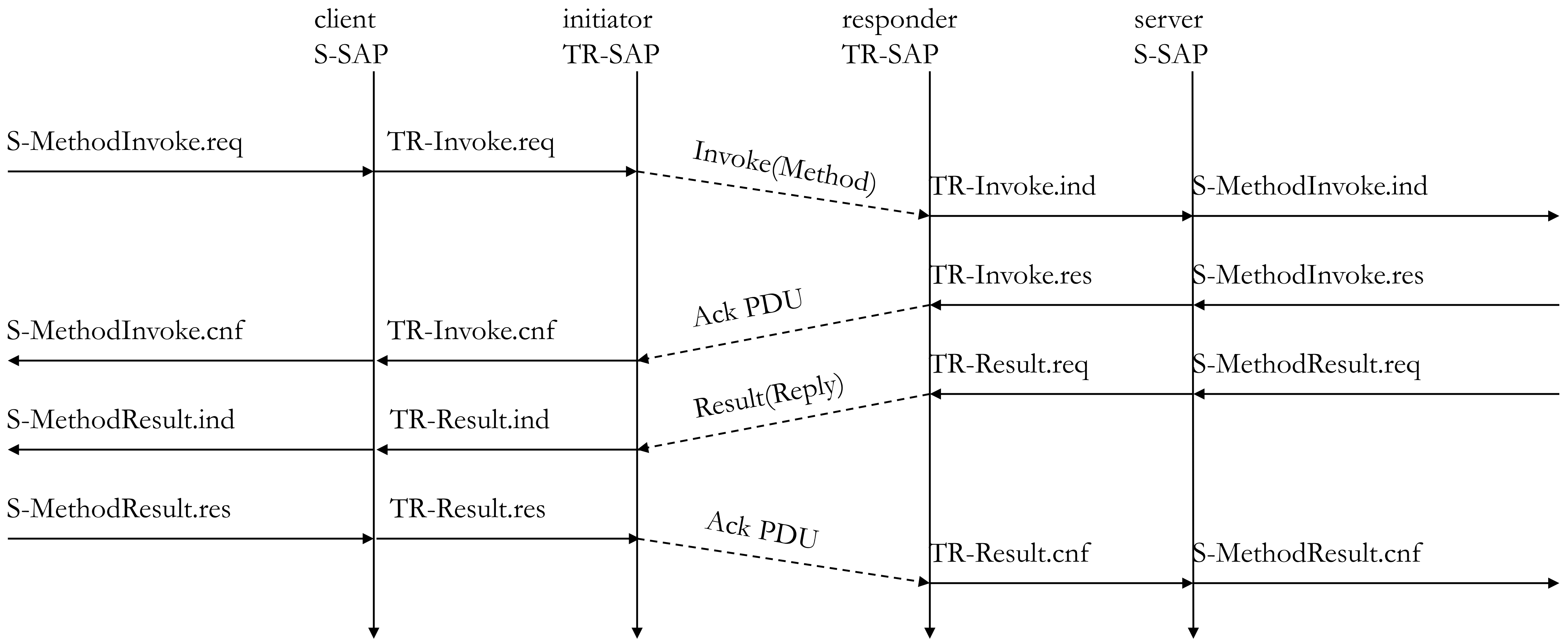

client S-SAP
server S-SAP
S-MethodInvoke.req (CTID, M, RU)
Method PDU
S-MethodInvoke.ind (STID, M, RU)
S-MethodInvoke.res (STID)
User ACK
S-MethodInvoke.cnf (CTID)
S-MethodResult.req (STID, S, RH, RB)
S-MethodResult.ind (CTID, S, RH, RB)
Reply PDU
S-MethodResult.res (CTID)
S-MethodResult.cnf (STID)
WTP Class 2 transaction

If any, the result, is sent back

S →HTTP status codes
404 → the server could not find the web page specified in 200 → everything is okay.

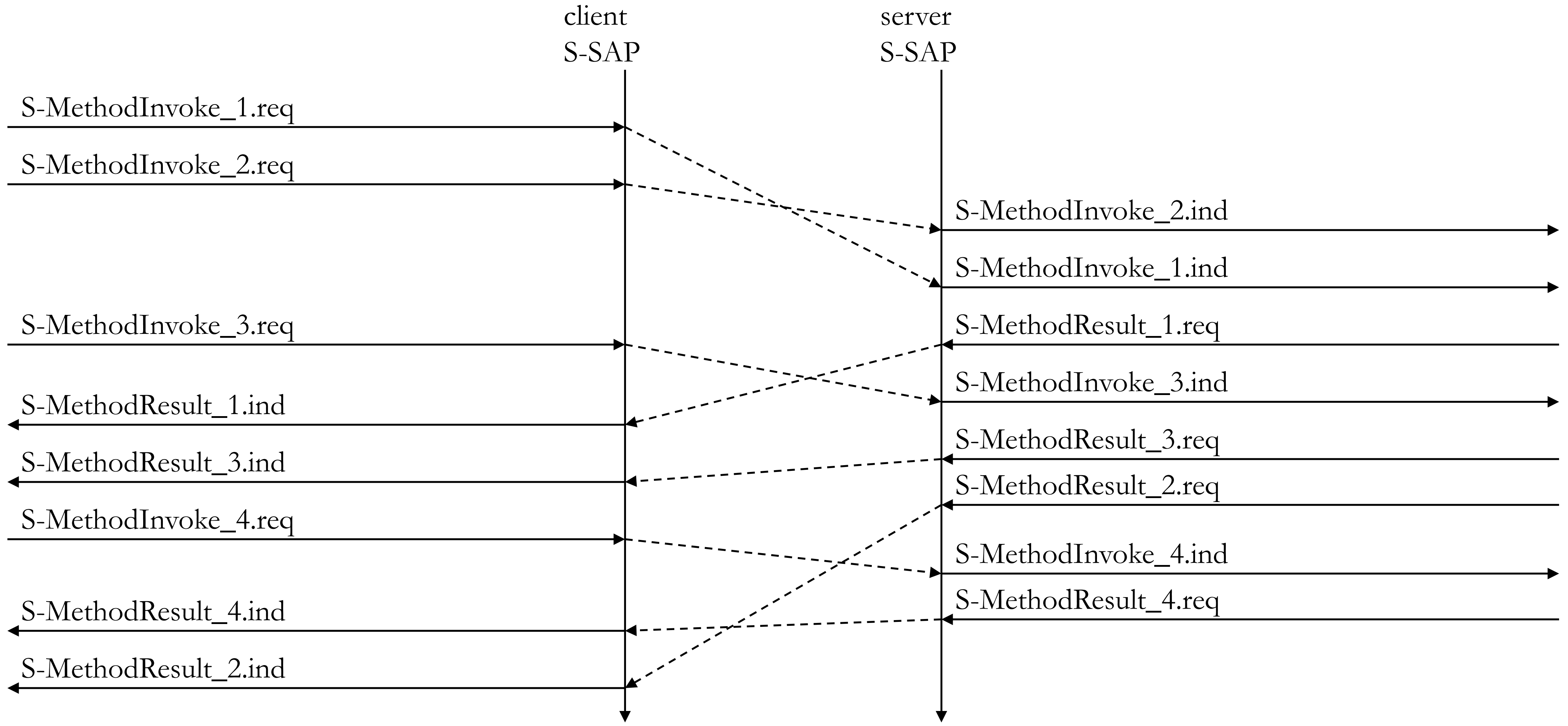
Wireless Session Protocol (WSP)

WSP/B over WTP - method invocation



Wireless Session Protocol (WSP)

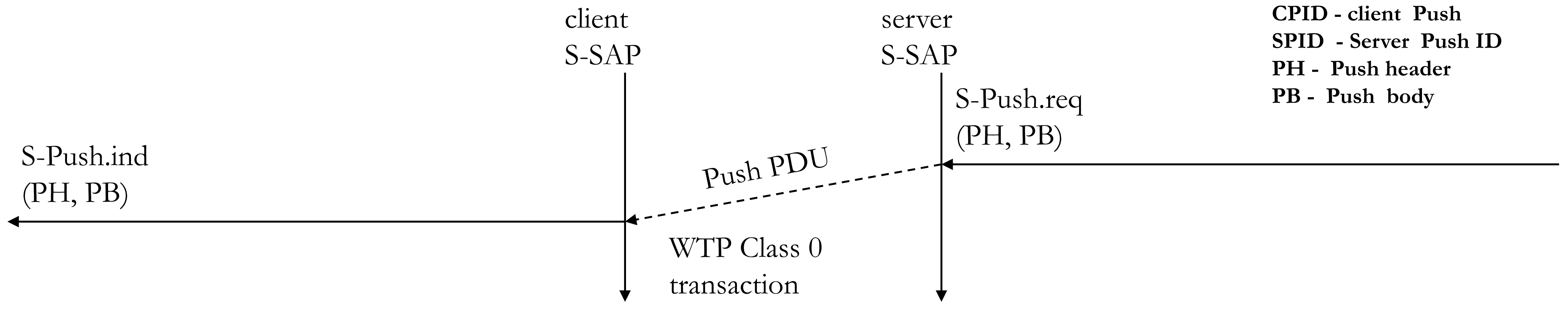
WSP/B over WTP - asynchronous, unordered requests

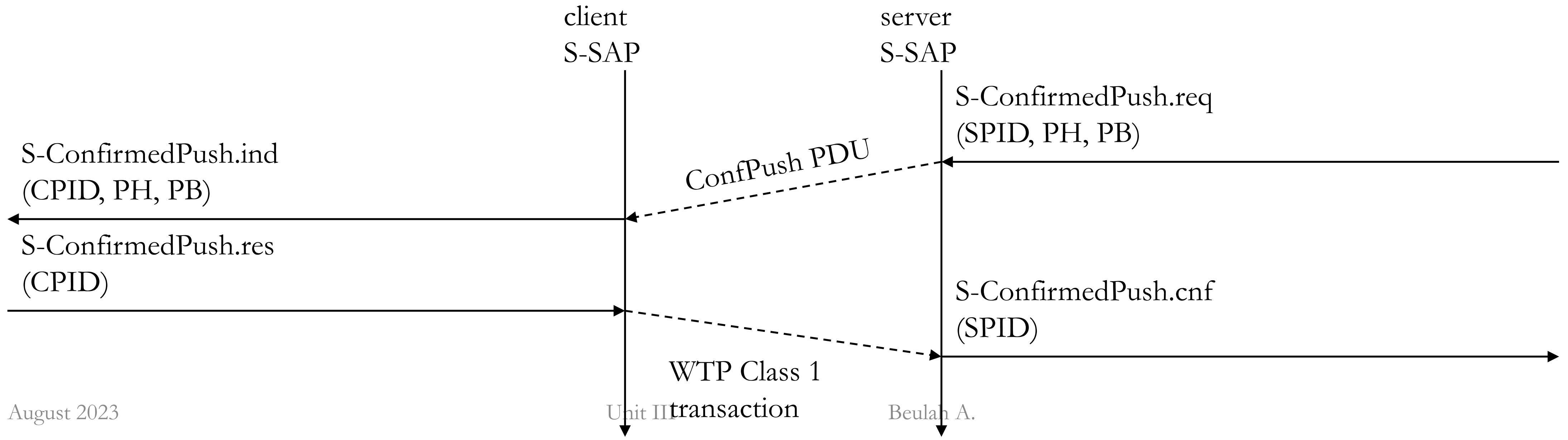


Wireless Session Protocol (WSP)

WSP/B - confirmed/non-confirmed push

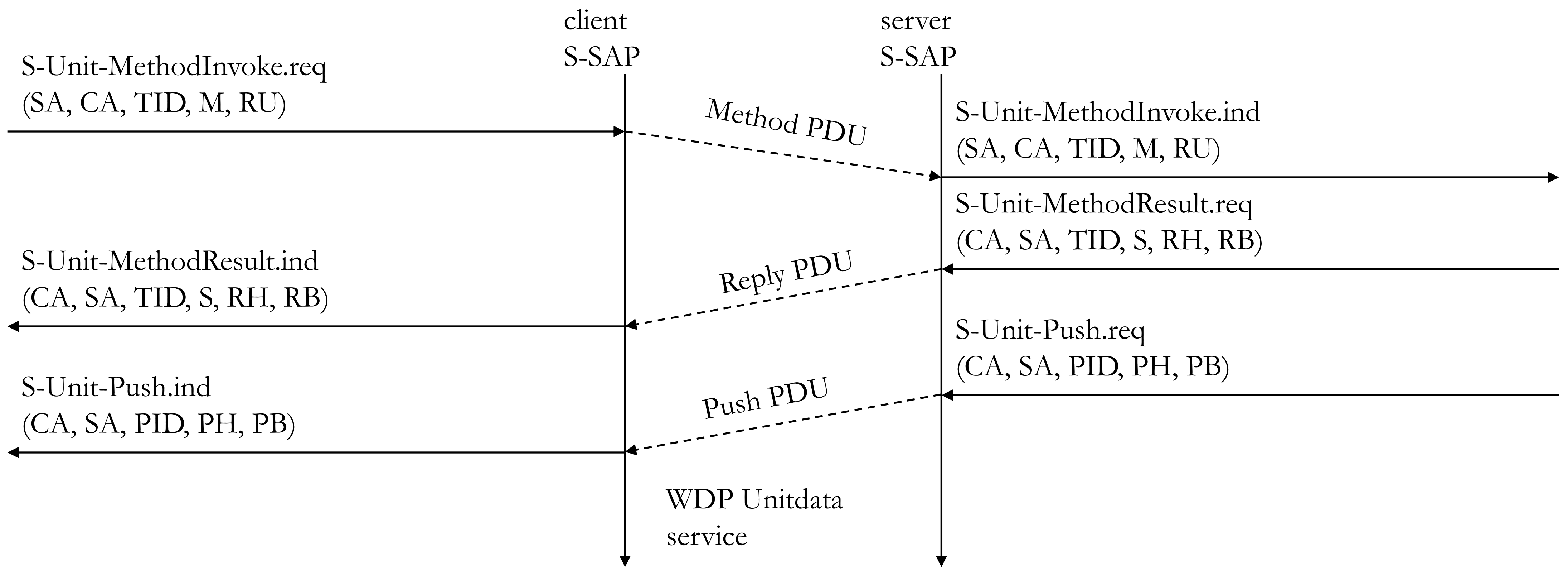
CPID - client Push
SPID - Server Push ID
PH - Push header
PB - Push body





Wireless Session Protocol (WSP)

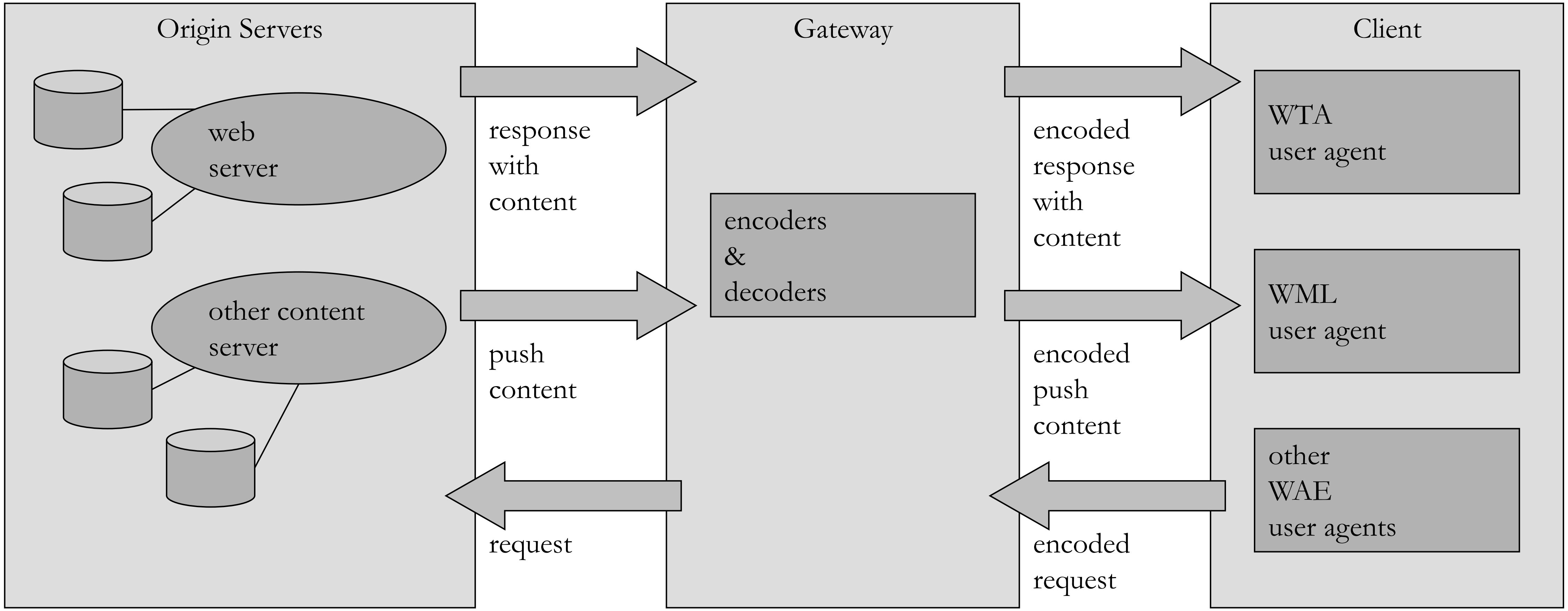
WSP/B over WDP



Wireless Application Environment (WAE)

- 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, ...

Wireless Application Environment (WAE)



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

Wireless Markup Language (WML)

```
<?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>
```

Wireless Markup Language (WML)

```
<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

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

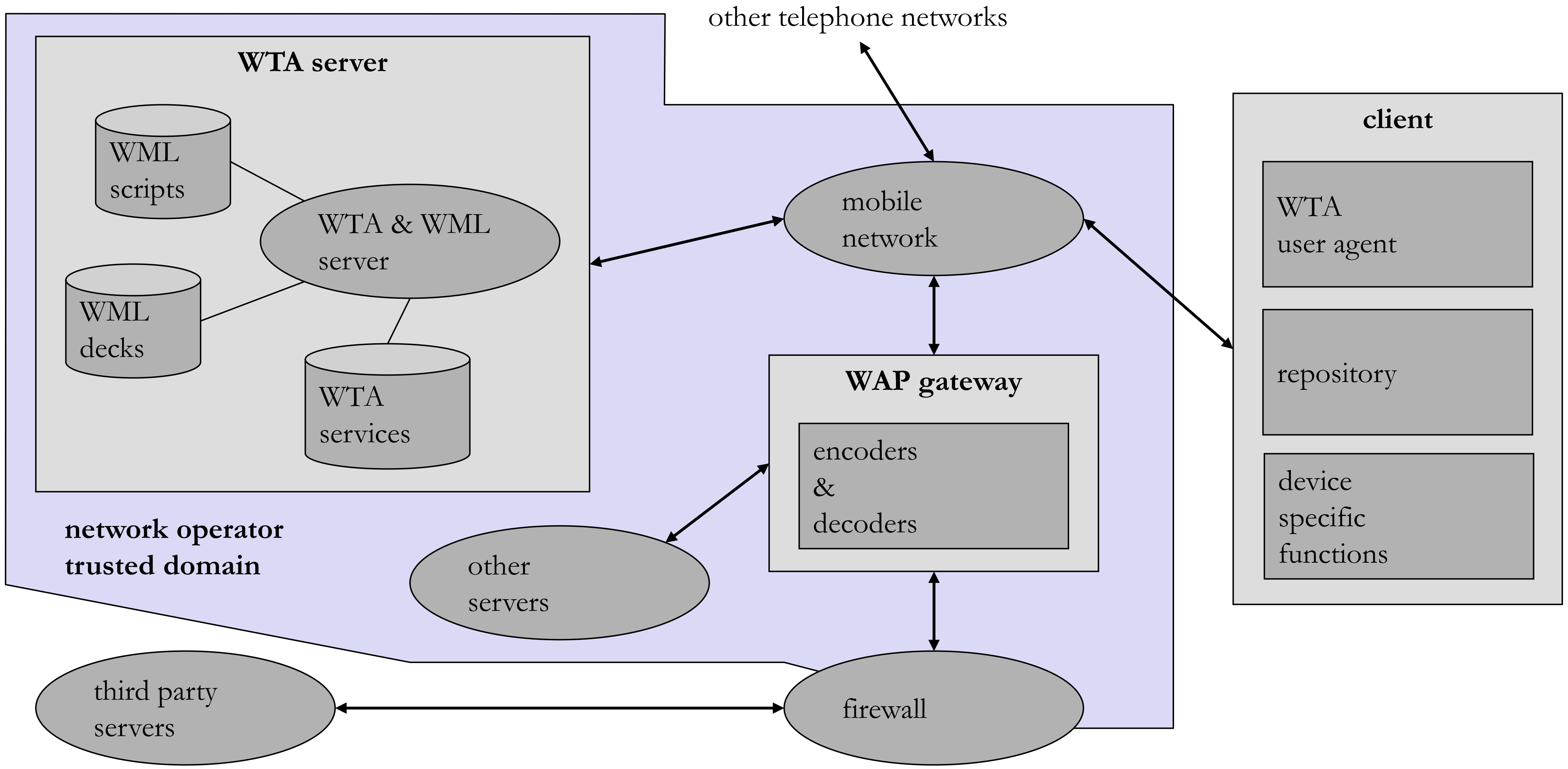
Wireless Telephony Application (WTA)

- 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)

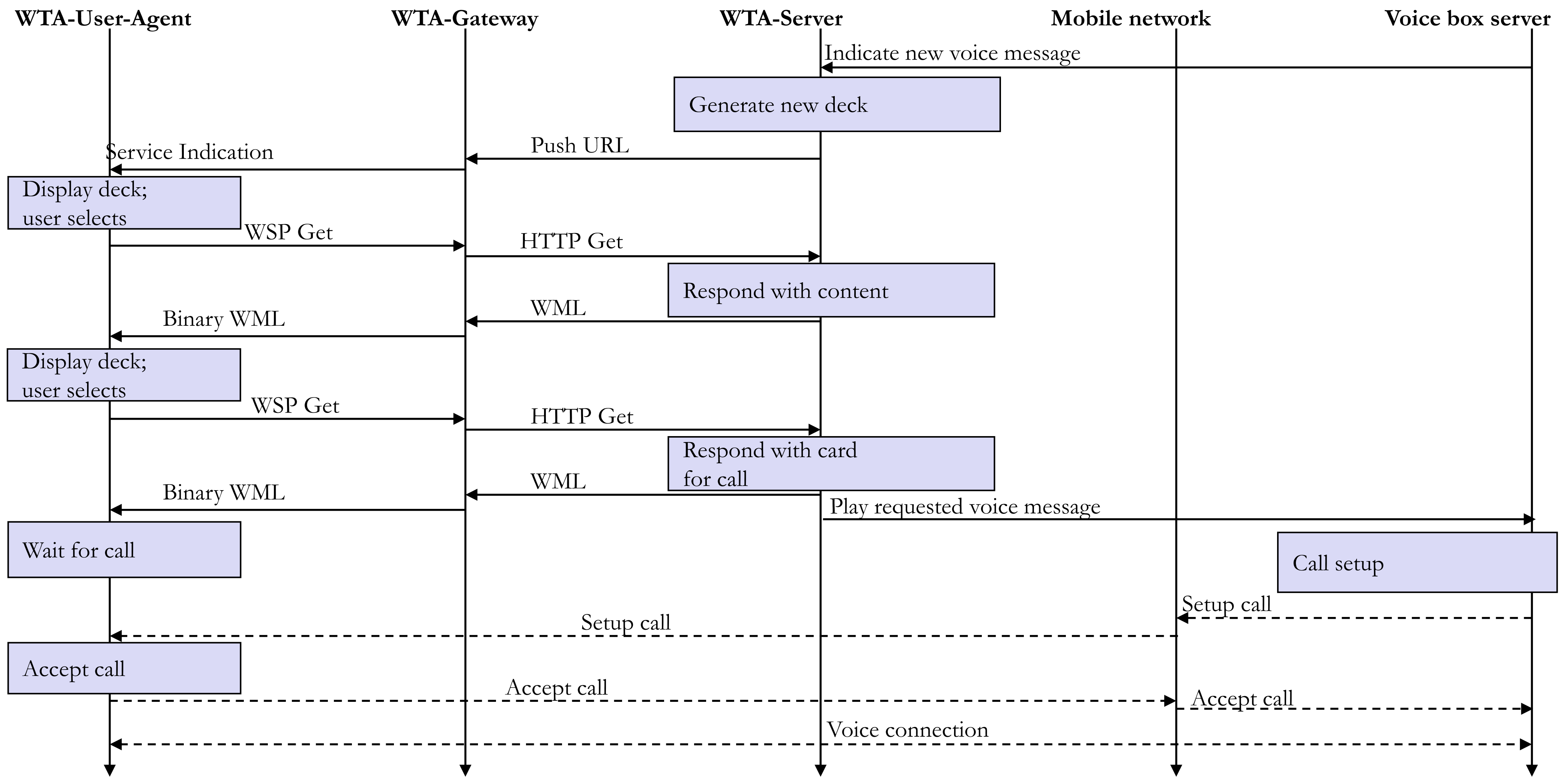
    ```
    wtai://wp/mc;07216086415
    ```
  - calling a number (WMLScript)

    ```
    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

```
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

```
<?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>
```

Test your Knowledge

- What are the primary goals of the WAP forum efforts and how they reflected in the initial WAP protocol architecture?

Summary

- Wireless application protocol (version 1.x)
  - 10.3.1 Architecture
  - Wireless datagram protocol
  - Wireless transport layer security
  - Wireless transaction protocol
  - Wireless session protocol
  - Wireless application environment
  - Wireless markup language
  - WMLScript
  - Wireless telephony application

References

Jochen H. Schller, “Mobile Communications”, Second Edition, Pearson Education, New Delhi, 2007.