Persistence Schemes

Chakchai So-In
Department of Computer science
Washington University

Outline

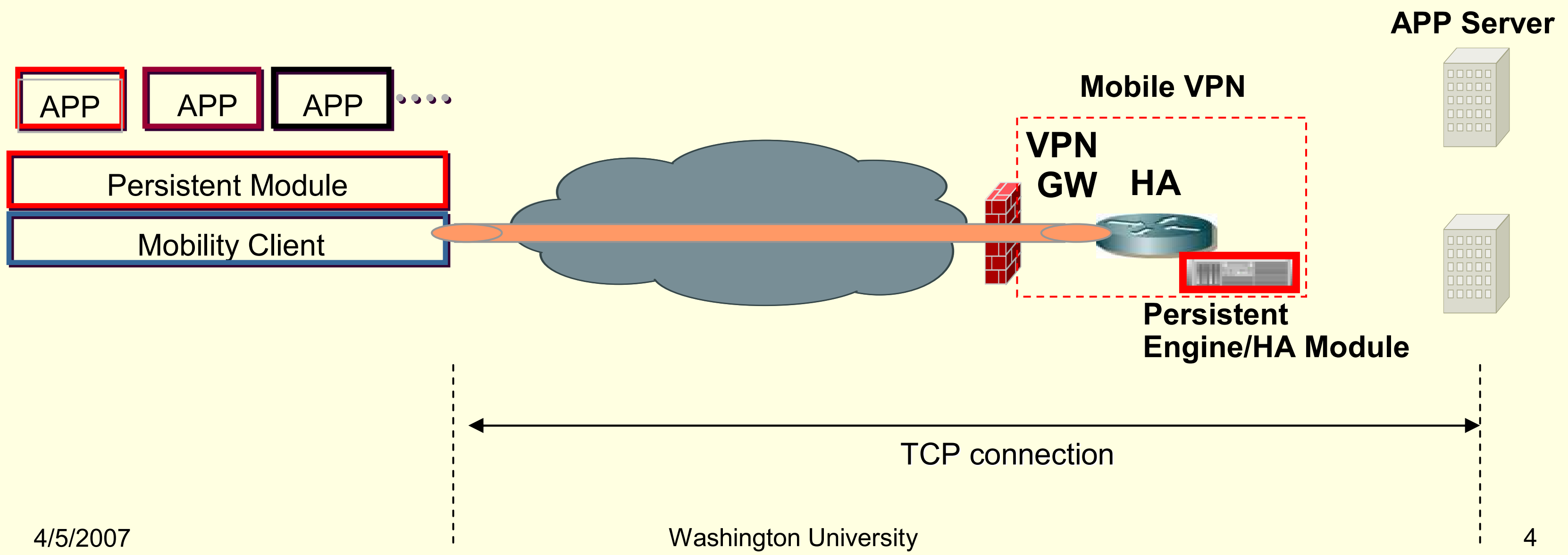
- Problems
- Goals
- General Ideas
- Transport persistence
- Future schemes/ Related Work
- Conclusions

Problems

- Connection lost especially for TCP
  - IP address is changed.
    - Client moves/ Better signal (wireless connection)
  - Handoff issue (L2&L3 handoff)
  - Link is broken: Network down
  - Other issues
    - Network move: Mobile Router
    - Client down/ Server down (TCP fault tolerant)
    - Client transport (move session across media/ adhoc network)

Problems (cont.)

- Move Detection (ICMP/ MobileIP message)
  - We know when the connection will be disconnected.
  - We don’t know in advance.



Goals

- To preserve the connection to be alive. (Given, Client down/ Network down/ Server down, Client move, Server move, the connection is still preserved (designing the framework for this)
  - TCP/ UDP
  - Other protocols such as RTP
- Avoid modifying TCP stack or Proxy issue
- Simplified the idea and easy to code

General Ideas

- Layer Persistence
  - L1 issue: Low power RF and DSP
  - L2 issue: FEC/ARQ
  - VPN issue:
    - IKE/IPsec Re-Keying
    - Allow for Dead-Spot time
    - IKE Keep-alive: Spoof Keep Alive (need to check)
    - Set keep-alive to 0 and account for DPD
  - L3 issue: Mobile IP
  - L4 issue: Maintain transport state
  - L7 issue: freeze application timer issue/ buffering

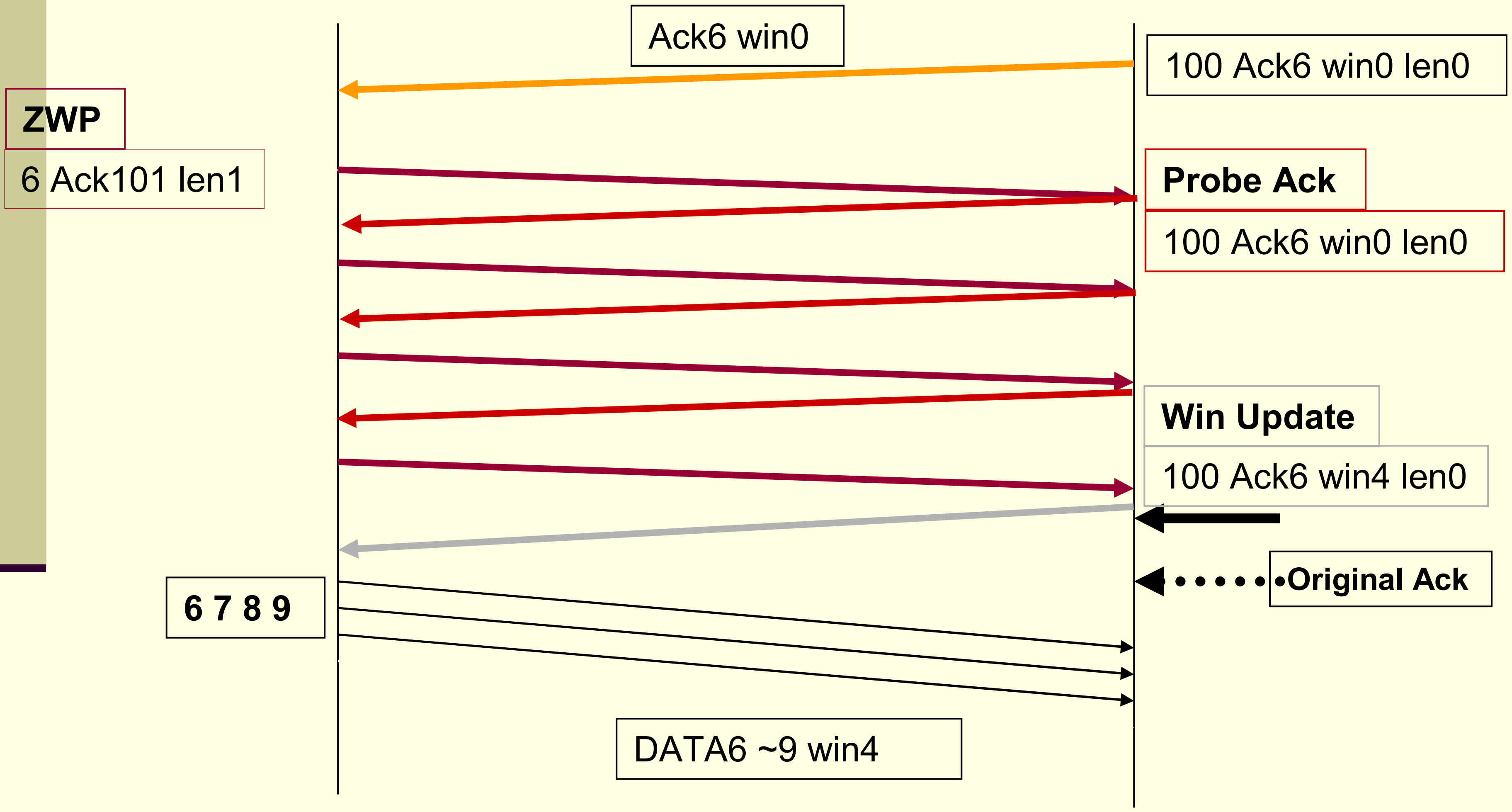
Transport persistence

- UDP Persistence (UDP -> Connectionless)
  - Resent previous UDP packet (like UDP keepalive)
  - MMS and Share Drive
    - Should be ok if control message is alive
- TCP Persistence
  - Retransmission Timeout, Keepalive Connection Timeout, Connection Timeout
  - Hardcode all those parameters
  - Other mechanisms such as TCP freeze

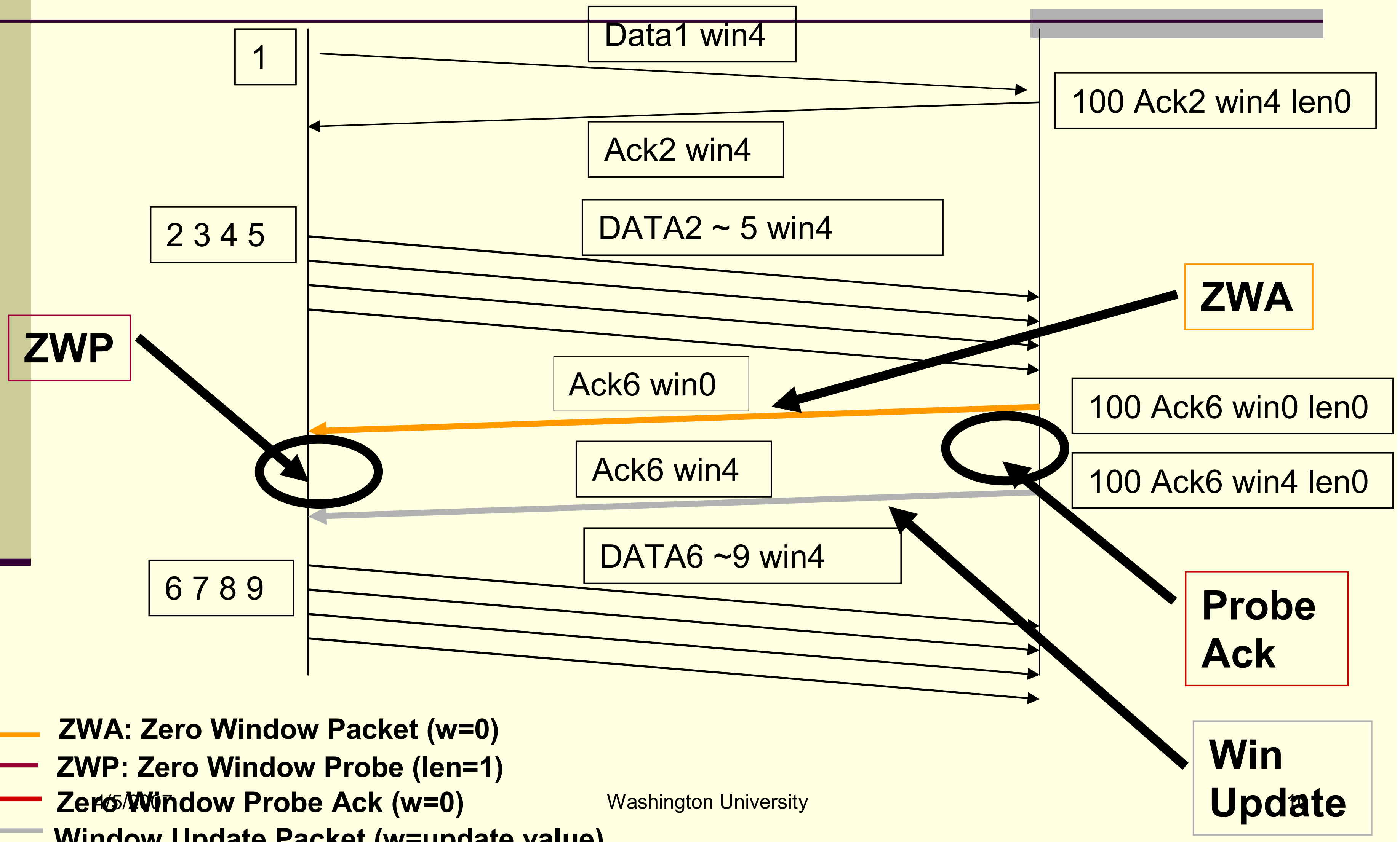
TCP freeze overview

- To freeze all TCP timers so we can keep TCP connection alive. (Build in TCP stack when the buffer is full)
- Once link is disconnected, a receiver sends Zero Window Option to freeze TCP timeout parameters as a result APP stops transferring data: ZWA
- When APP receives ZWA, it will generate TCP Probe back to check if it can continue sending data : ZWP
- As received TCP Probe, the receiver sends Probe Ack back if it still wants to freeze transmission but sends Window update packets to return to normal state.

TCP freeze overview (cont.)



TCP freeze overview (cont.)



🟧	ZWA: Zero Window Packet (w=0)
🟥	ZWP: Zero Window Probe (len=1)
🟥	Zero Window Probe Ack (w=0)
⬜	Window Update Packet (w=update value)

TCP freeze overview (cont.)

- System Testing
  - Telnet: OK with probe ack
  - FTP: Mostly OK
- Constrain: TCP timeout depends to TCP application; Window FTP client (Set an option, “how long the client will disconnect regarding the received ZeroW packet”)

Idea Implementation

- Stateful approach
  - We keep updating both sequence and acknowledgement number for each flow. Once the link is disconnected, we can send the zero window packet based on the last ack.
- Pros: correct information (real ack)
- Cons: too many states and we have to keep the states although we are in normal operation: slow (the link is connected)

Idea Implementation (cont.)

- Iteration technique (based on δ)
  - Once the link is disconnected, we keep sending 3 duplication acknowledgement packets back (ACK = current ack – δ) which δ is usually a maximum window size number. We can estimate δ from *max(recvW, sendW)* when first the connection is established.
- Pros: correct information (real ack)
- Cons: if δ is very big, we have spend too much time to keep sending ACK—till we get the last real ack. It might cause the application timeout be expired.

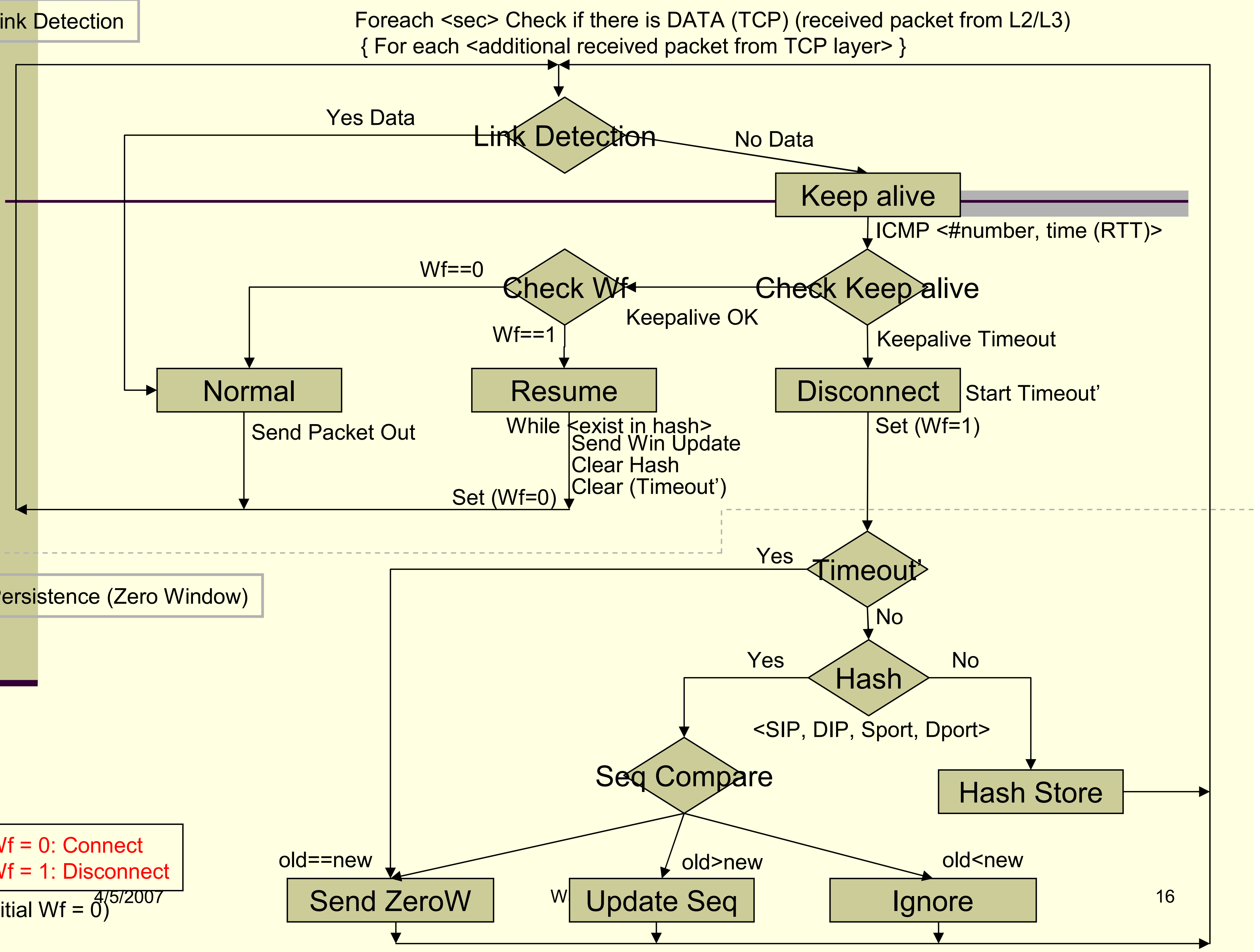
Idea Implementation (cont.)

- Best guess based on traffic and retransmission packets
  - We keep tracking of the retransmission packets. Once we receive a second retransmission packet, we send the zero window packet back based on the least sequence.
  - Once the link is disconnected and the application retransmits the packet (RTO is expired), CWND is set to 1 (It means the application can send at most only one packet). So they can retransmit only the first real ack.

Idea Implementation (cont.)

- Pros: lesser states; we don’t need to keep the state information during the normal operation.
- Cons: there is an error with this algorithm but the probability is too low. For example, if the application sends duplicated packets (Digital fountain technique), we would think it’s the retransmission packet.

Link Detection



Persistence (Zero Window)

Wf = 0: Connect
Wf = 1: Disconnect

(Initial Wf = 0)

Future schemes/ Related work

- Maintaining TCP connection due to the disconnectivity (connection ID)
  - Migratory TCP (M-TCP): Changing client IP [1]
  - Robust TCP Connection [2]
  - TCP-R (TCP Redirection) [3]
  - A Mobile Socket [4]
- Checkpoint/ Process state backup
  - Persistent connection [5]

Future schemes/ Related work

- Making Server Fault Tolerant
  - TCP Connection Migration : Changing Server [6]
  - Fault tolerant TCP (FT-TCP) [7]
- Proxy-assisted
  - MSOCK [8]
- Context Migration
  - Mobile Service: Context-Aware Service Migration in Ad Hoc Networks [9]

Future schemes/ Relate work

- Application Mobility
  - Application-layer Mobility support for Streaming Real-time Media [10]
  - Application-Layer Mobility Using SIP [11]
- Session Mobility
  - SLM, A Framework for Session Layer Mobility Management [12]
  - TESLA: A Transparent, Extensible Session-Layer Architecture [13]

Common Ideas

- Hide information from an application layer
  - Virtual Port and Virtual connection ID
- Present a unique connection ID securely mapping to previous connection (synchronize seq and bytes)
- Buffering/ a mechanism to block non-blocking write
- Find a way to freeze all timers (L4 to L7)

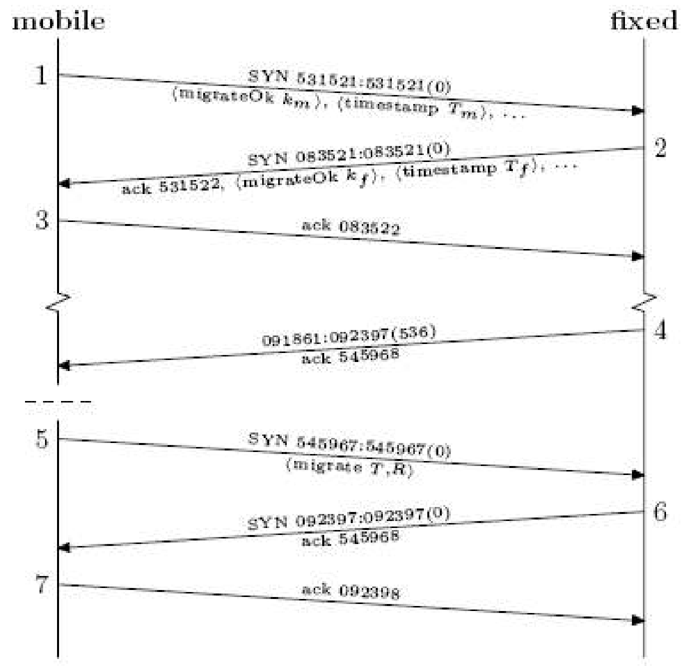


Figure 1: TCP Connection Migration

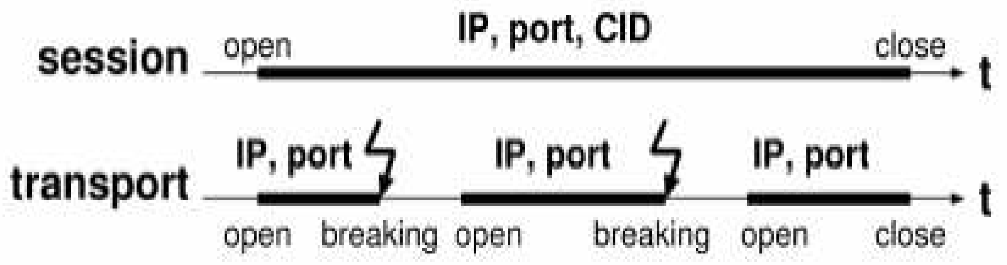


Fig. 2. Lifetime of session and transport layer connections.

1(left), 2(right)
3(left), 4(right)

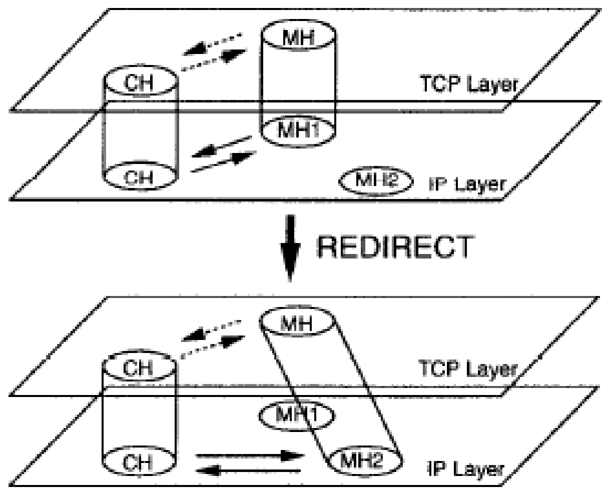


Figure 1: Redirection on Active TCP Connections

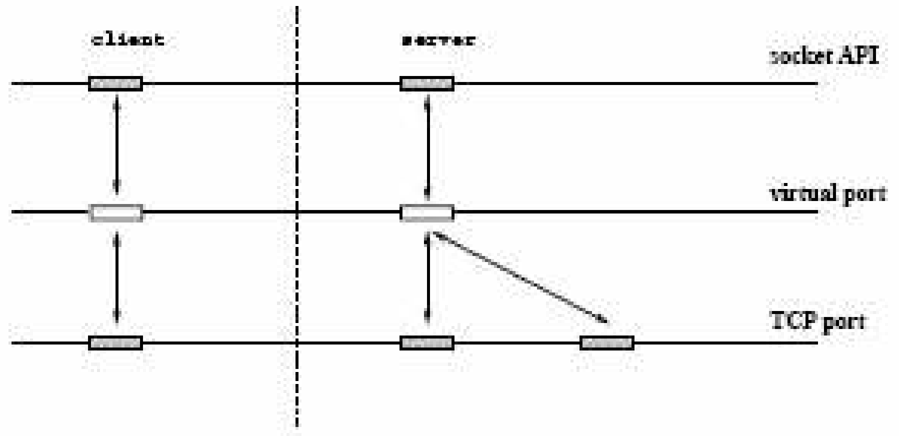


Figure 18: Virtual Port

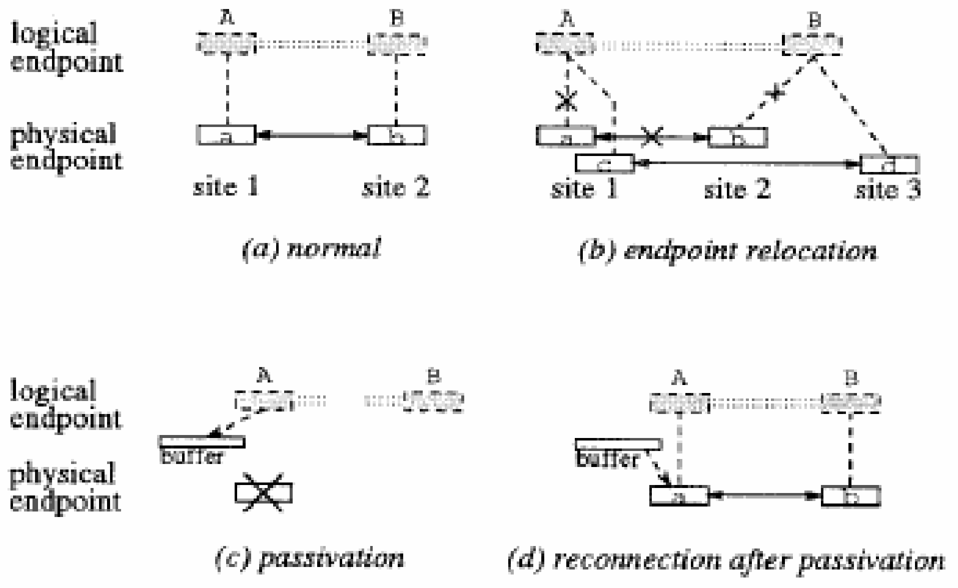
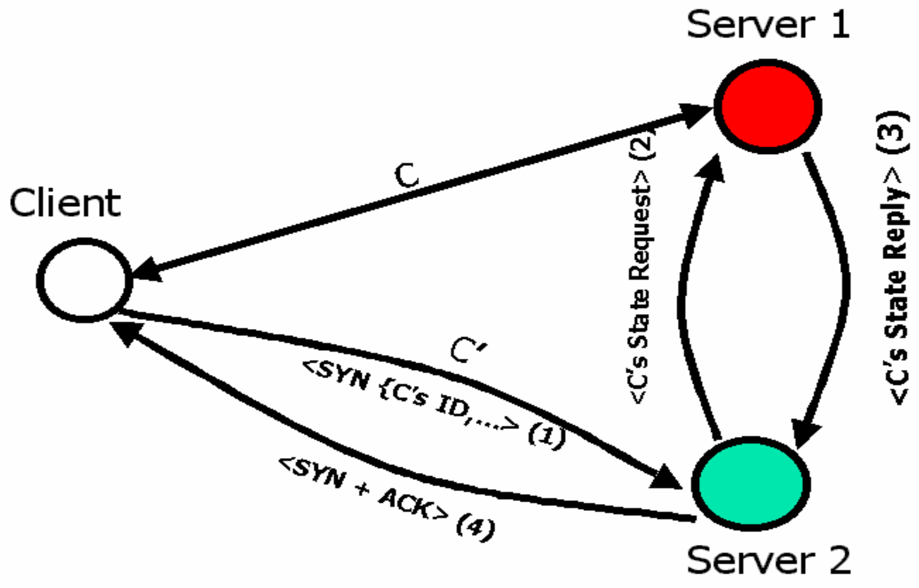


Figure 2: Handling relocation and passivation



5(left), 6(right)
7(left), 8(right)

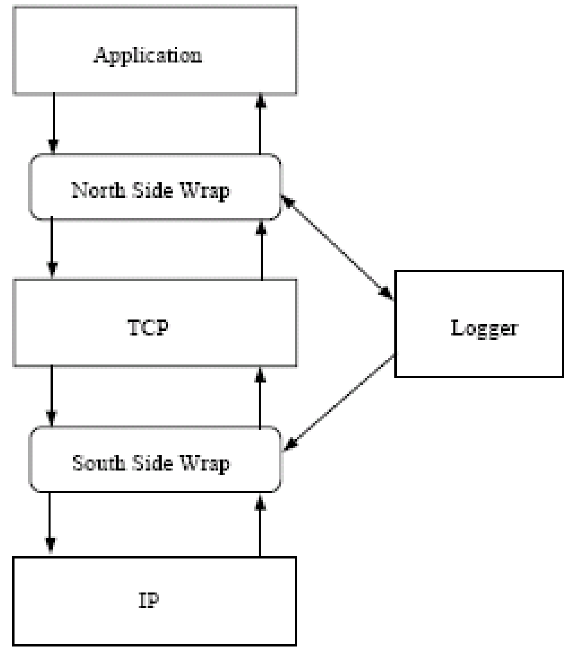


Fig. 1. FT-TCP architecture.

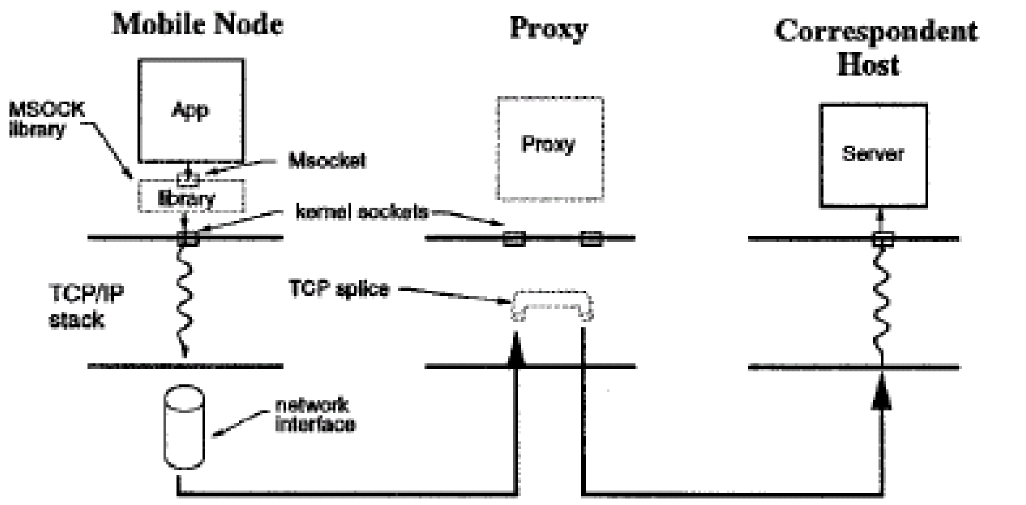


Fig. 2. The MSOCKS architecture. Parts shown in gray are where MSOCKS alterations are made to the standard parts of proxy based client/server system.

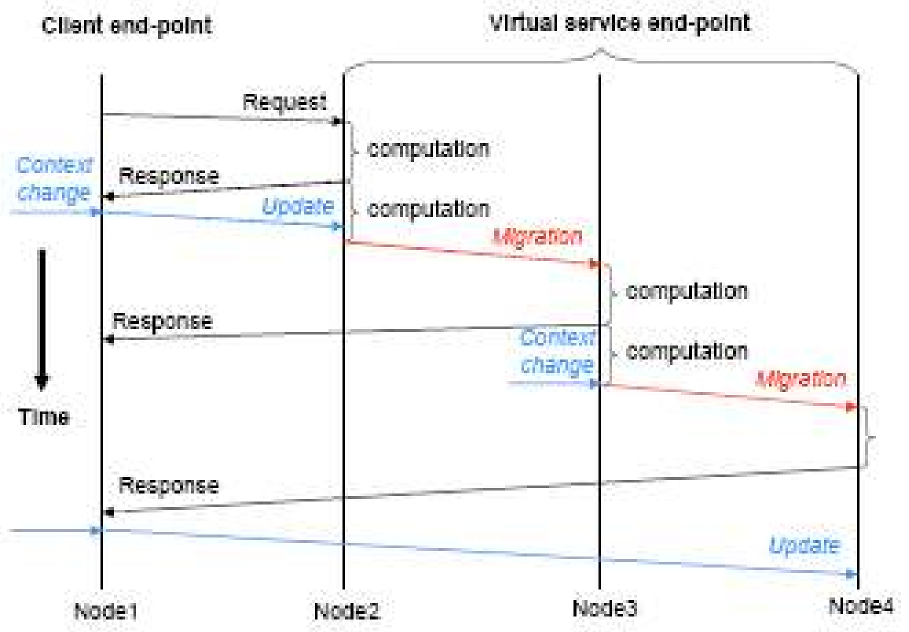
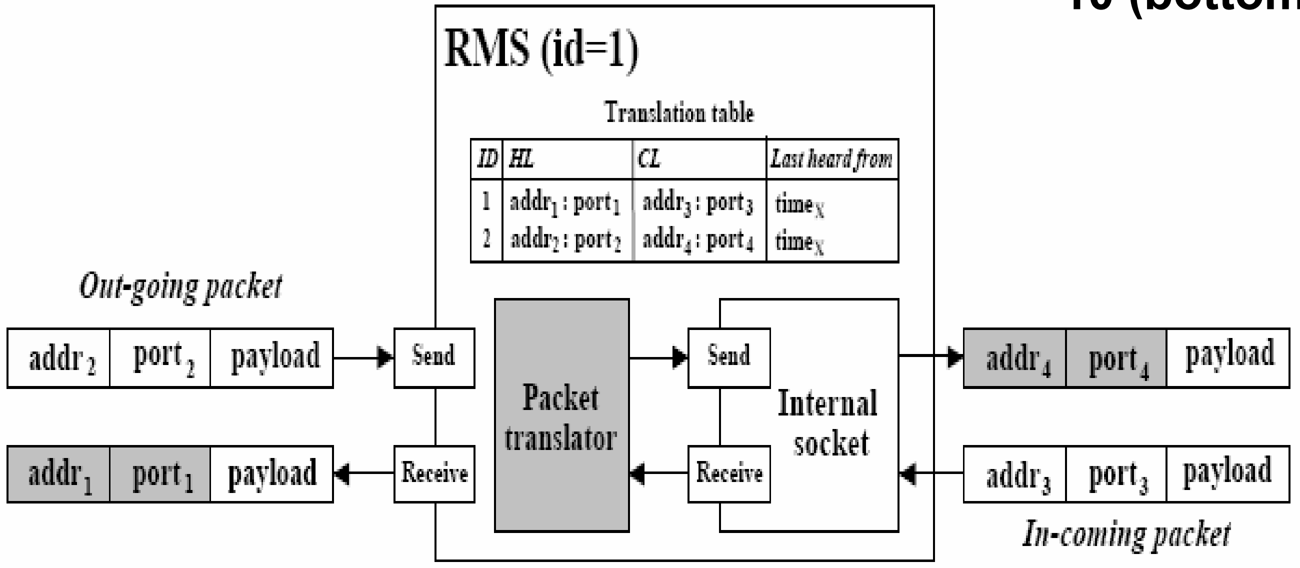


Figure 2. Context-Aware Service Migration

9 (top)
10 (bottom)


RMS (id=1)
Out-going packet
addr$_2$ | port$_2$ | payload
addr$_1$ | port$_1$ | payload
Send
Receive
Packet translator
Send
Receive
Internal socket
addr$_4$ | port$_4$ | payload
addr$_3$ | port$_3$ | payload
In-coming packet

Translation table

ID	HL	CL	Last heard from
1	addr$_1$: port$_1$	addr$_3$: port$_3$	time$_X$
2	addr$_2$: port$_2$	addr$_4$: port$_4$	time$_X$

Fig. 2. A packet translation process.

Conclusions

- Persistence Schemes
- Always-on connection (Client down/ Network down/ Server down, Client move/Server move, Network move, and Client transport)
- Designing a framework for application-persistent mobility (if possible)