Introduction to Wireless TCP

Choong Seon Hong

Kyung Hee University
cshong@khu.ac.kr
Contents

✓ Background
✓ Introduction
✓ Problems of Original TCP in Mobile Environments
✓ Strengths and Drawbacks of Existing Solutions
✓ Our Proposed Solution: Adaptive TCP
✓ Conclusions
Background: IP Protocol Stack

- **Network Layer (Layer 3):**
  - Internet Protocol (IPv4, IPv6)
  - Packet (IP datagram) transmission between end-systems [hosts]
    (packet size up to 65535 bytes, often restricted by Layer 2 protocols)
  - Routing using 32 bit addresses (v4)
  - Unreliable, connectionless transmission: loss, duplication, reordering can occur

- **Transport Layer (Layer 4):**
  - Most frequent transport protocols
    - Transmission Control Protocol (TCP)
    - User Datagram Protocol (UDP)

- **Application Layer/Services (Layer 5-7):**
  - TCP based:
    - HTTP (HyperText Transfer Protocol),
    - FTP (File Transmission Protocol),
    - SMTP (Simple Mail Transfer Protocol), Telnet, ...
  - UDP based:
    - DNS (Domain Name Service),
    - Streaming media, VoIP, ...
    - SNMP (Simple Network Management Protocol)
Goal: data transfer between application (processes) in end-systems

- support of multiplexing/de-multiplexing
  e.g. socket API

Data stream/connection identified by:
  two IP addresses, protocol number, two port numbers
Background : Transport Layer Protocols

- User Datagram Protocol (UDP) : RFC 768
  - Connectionless
  - Unreliable
  - No flow/congestion control

- Transmission Control Protocol (TCP) : RFC 793, 1122, 1323, 2018, 2581
  - Connection-oriented (full duplex)
  - Reliable, in-order byte-stream delivery
  - Flow/congestion control

- Stream Control Transport Protocol (SCTP)
  - Connection oriented (full-duplex associations)
  - Reliable message delivery, support of multiple streams
  - Support of multi-homing, flow/congestion control

- Real-Time Transport Protocol (RTP)
  - Uses UDP
  - Provides: Time-stamps, sequence numbers
  - Supports: codecs, codec translation, mixing of multi-media streams
TCP: Basics

- Point-to-point, bi-directional connections (between end-systems)
- Reliable, in-order transport of byte-stream using
  - Sequence Numbers
  - Acknowlegements

- Flow/Congestion Control:
  - Prevent flooding of
    - Receiver
    - Intermediate Systems

- Important 'new' Header Fields
  - Sequence number: number of first data byte transmitted in the segment
  - ACK number: number of next byte expected in the reverse data flow
  - Window size: number of bytes host is willing to accept in reverse data flow
A state transition diagram for TCP

Now connection is closed in one direction.
Introduction

- Internet access and wireless access are resulting in a strong combination.
- TCP is a transport layer protocol designed for the wired links.
- It will incur end-to-end performance degradation in mobile environments.
Introduction (cont’d) : Mobile Environment

- **MH** (mobile Host)
- **FH** (Fixed Host)
- **BS** (Base Station)
- **wireless links**
- **wired links**

- **cell 1**
- **cell 2**
Introduction (cont’d)

✓ The challenges for TCP in mobile environments are:

1. High Bit Error Rates (BER)
2. Long time disconnections
3. Frequent disconnections
Problems of original TCP (cont’d)

For High-Bandwidth Links:

- CWND doubles after each round-trip time (RTT) in slow-start phase
- CWND increases by one segment size in each RTT during congestion avoidance phase

Congestion detected:

- **timeout** → set ssthresh to cwnd/2, perform slow-start
- **three dup Acks** → half cwnd, ssthresh=cwnd (Fast Retransmit/Recovery)

Throughput = CWND*MSS / RTT (for high-bandwidth links)
Problems of original TCP (cont’d)

✓ The idle time $T$ of the sender in slow start stage is given by:

$$T = P(\text{RTT}) - (2^P - 1)S/R$$

- $S$ means the MSS (maximum segment size) is $S$ bits.
- $\text{RTT}$ is the round-trip time.
- $R$ denotes the transmission rate of the link from the sender to the receiver.
- $P = \min\{Q, K-1\}$
  . $K$ is the number of windows that cover the object
  . $Q$ is the number of times the sender would stall
Problems of original TCP (cont’d)

- If the slow-start scheme is not triggered, the number of extra packets that can be transferred is given by:

  \[ \text{Extra Segments} = \frac{W^2}{8} + W \log W - \frac{5W}{4} + 1; \]

  - \( W \) is the unACKed packets that can be sent in a congestion window.

* The wrong assumption on packet loss in mobile environment will definitely decrease the performance of TCP.
Wireless (Infrastructure) Networks: Challenges

- Wireless links tend to have the following properties:
  - Large delays
  - Low throughput
  - Bit errors / packet losses due to poor channel conditions (noise, interference, fading)

- Impact of mobility
  - Delay/losses due to hand-over events

- Protocols in IP family are not originally designed for such links
  - Increased volume due to headers
  - Deficiencies of TCP flow/congestion control
  - ... many more (e.g. applications HTTP→WAP)

- Protocol Enhancements are being developed, e.g.
  - Robust Header Compression (RoHC)
  - Enhancements for Wireless TCP
TCP-mechanisms in wireless settings

TCP assumes congestion if packets are dropped

- possibly wrong in wireless networks, here we often have packet loss due to *transmission errors*
- furthermore, *mobility* itself can cause packet loss (handover losses) or temporary connection disruptions (timeouts or even broken TCP connections)

→ performance of an unchanged TCP degrades severely

- however, TCP cannot be changed fundamentally due to the large base of installation in the fixed network, TCP for mobility has to remain compatible
- the basic TCP mechanisms are needed to for congestion control in the wired network parts
Wireless TCP: Common Approaches for Enhancement

✓ Link-Layer Approach
  ➣ Local Retransmission of lost packets
  ➣ Hide losses from sender

✓ Explicit Notification Approach
  ➣ Explicitly notify TCP sender of the condition of the network / type of the loss

✓ End to End Approach
  ➣ Enhance TCP Protocol stack at sender and receiver to achieve better throughput (e.g. TCP SACK)

✓ Split-Connection Approach
  ➣ End-to-End flow control terminated before wireless link
Wireless TCP: Split Connection

Fixed host (FH)

Wired network

TCP

IP

Wire Network Interface

Server Applications

Split-TCP Daemon

TCP

TCP/Enhanced protocol

IP

Wire Network Interface

Enhancing Proxy (PEP)

Wireless Network Interface

Wireless Network I/F

Wireless Network

Mobile host (MH)

Applications

TCP/Enhanced protocol

IP

Wireless Network I/F

TCP connection

connection

✓ Proxy is located between the 2 end-hosts to split the TCP connection into 2 parts
✓ Different implementations: Indirect TCP (state-full), Mobile TCP (stateless, send recv=0 when disconnection detected)
Wireless TCP: Split Connection (cont’d)

✓ Advantages
  F No Changes in wired network and correspondent host necessary
  F Shields the end-host in the wired network from the wireless network characteristics: Wireless Local Recovery by Proxy
  F Possibly reduced header size for optimized transport protocol over wireless link
  F Allows for ‘Smaller’ and simpler wireless protocol between MH and Proxy

✓ Disadvantages
  F Loss of end-to-end TCP semantics
  F Not usable with end-2-end encryption (e.g. IPsec), since access to the TCP header needed
  F For ‘state-full’ versions:
    F Requires buffer management at Proxy
    F Requires state transfer when hand-over to different proxy occurs
    F Hard state: Failures of proxy can result in loss of data
Wireless TCP: Link-Layer Approach

- Proxy detects losses over the wireless link
- Proxy does local retransmission before the sender timeouts. Hides the wireless losses from the Sender
- TCP-unaware approaches: local retransmission
- TCP-aware approaches: e.g. SNOOP (but: filtering of dup Acks!)
Wireless TCP: Link-Layer Approach

TCP-aware approach: SNOOP

- Modify the IP layer in the BS, and let BS cache the TCP packets sending from CN to the MN.
- If the packet lost on wireless link, IP layer on the BS will retransmit the packet.
- BS suppress DUPACKs sent from MN to CN.
- BS use shorter local timer for local timeout.
Wireless TCP: Link-Layer Approach (cont’d)

- Changes are restricted to BS and optionally to MN as well
- A (snoop) layer is added to the routing code at BS which keep track of packets in both directions
- Packets meant to MN are buffered at BS, and if needed, retransmitted in the wireless link

Discussion: Link-Layer Approaches

✓ Advantages
  - No modification to TCP layer in end hosts
  - Prevent propagation of losses/errors to TCP source

✓ Disadvantages
  - TCP-aware link layer solutions cannot be used with IPSec
  - Link-layer retransmission cause RTT variations (and thus possibly TCP time-outs)
  - Not usable if data and Ack traverse different paths
Wirless TCP: End-to-end approach

✓ TCP Protocol Stack at Sender and Receiver enhanced
e.g. SACK, FACK, D-SACK, Eifel, Freeze TCP

✓ Possible enhancements
  ✶ Differentiate Losses
e.g. no slow-start for wireless losses
  ✶ Efficient Utilisation of Bandwidth
e.g. no initial slow-start
  ✶ Detection of Multiple Losses

✓ Advantages
  ✶ End-to-End semantics and layered architecture of network protocols are preserved
  ✶ IP packet encryption can be used

✓ Disadvantage
  ✶ Modification at end host
End-2-End Enhancements: SACK

TCP Reno inefficient for multiple packet losses within congestion window

Extension: Selective Acknowledgement (SACK)

SACK principles:

✓ Several (3) SACK blocks mark successfully received sets of data (in TCP Option header extension)
✓ Sender maintains scoreboard of successfully transmitted packets
✓ Special treatments of partial Acks during Fast Recovery
→ Allows retransmission of more than one segment per RTT
End-2-End Enhancements: Freeze TCP

Principles

- Freeze-TCP does not need help from base station
- The receiver sends Zero Window Advertisement (ZWA)
- End-to-end enhancement
- TCP receiver monitors signal strengths and detects an impending handoff
- Advertise a zero window size to force the sender into the ZWP mode and prevent it from dropping its congestion window

Zero Window Probe (ZWP)

- An advertised window size is zero $\rightarrow$ the sender freeze all retransmit timers and enter a persist mode
- Sender sends ZWPs until the receiver’s window opens up
- The interval between successive probes grows exponentially until it reaches 1 minute

- Receiver sends 3 DUPACKs when route is re-established
End-2-End Enhancements: Freeze TCP

✓ TR-ACKs

☞ Prevent substantial idle time after a reconnection (ZWP is exponentially backed off)

☞ As soon as a connection is re-established, the receiver sends 3 copies of the ACK for the last data segment it received prior to the disconnection

✓ Advantages

☞ It is a true end-to-end signalling scheme and does not require any intermediaries (such as base stations) to participate in the flow control

☞ Changes to TCP code are confined entirely to the receiver side and are easy to implement
End-2-End Enhancements: Freeze TCP

✓ Disadvantages

- Whether it is appropriate to restart transmission at the full rate with the old window size upon entering a new, unknown environment?

- It needs the receiver to predict impending disconnection
End-2-End Enhancements: Freeze TCP

Regular TCP

Freeze TCP
Existing Solutions - Summary

- Indirect TCP (I-TCP) [A. Bakre 1995]
- Mobile TCP (M-TCP) [K. Brown 1997]
- Explicit bad-state notification (EBSN) [N. Vaidya 1999]
- Snoop protocol [H. Balakrishnan 1995]
- New Snoop protocol [Jian-Hao Hu 2000]
- Freeze-TCP [Goff 2000]
- TCP HACK: TCP Header Checksum Option [R. K. Balan 2001]
- Adaptive Bandwidth Share Estimation [R. Wang 2002]
## Strengths and Drawbacks of Existing Solutions

<table>
<thead>
<tr>
<th></th>
<th>I-TCP</th>
<th>M-TCP</th>
<th>Snoop</th>
<th>New Snoop</th>
<th>EBSN</th>
<th>Freeze-TCP</th>
<th>ABSE</th>
<th>TCP Hack</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-to-end TCP semantics?</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Require the intermediate node to modify TCP?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Handle encrypted traffic?</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Need symmetric routing?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Handle long disconnections?</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Handle frequent disconnections?</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Handle high bit error rate?</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Drawbacks of using Performance Enhancing Proxies (PEPs)

- The intermediary will become the bottleneck
- Add the third point of failure besides the endpoints themselves
- Cannot handle end-to-end encrypted traffic
- Need a symmetric routing
Proposed Solution: Extended TCP HACK

- The special ACK in TCP HACK may be lost.
- Extended TCP HACK:
  - Add a buffer in the TCP receiver (denoted s_buffer).
  - Then, save all sequence numbers into s_buffer and these sequence numbers are recovered from those packets transmitted in the same window (Data are corrupted but the sequence numbers in headers can be recovered.)
  - Send special ACK to the sender containing all the sequence numbers in the s_buffer.
  - Sequence numbers in the s_buffer will be cleared if the retransmitted packets are received correctly or the timers expire.
### Extended TCP HACK (cont’d)

<table>
<thead>
<tr>
<th>Kind=14</th>
<th>Length=4</th>
<th>1’s complement checksum of TCP header and pseudo-IP header</th>
</tr>
</thead>
</table>

**TCP Header Checksum option**

- **Kind=16**
  - **Length**: Variable
  - 1st 32-bit sequence number of corrupted segment to resend
  - 2nd 32-bit sequence number of corrupted segment to resend
  - ….
  - nth 32-bit sequence number of corrupted segment to resend

**Extended TCP header Checksum ACK option**
• The segments with sequence A, C and E are data corrupted but their headers can be recovered.
• The segment with sequence B is correct
• The segment with sequence D is corrupted in the header.
Extended TCP HACK (cont’d)

The process of extended TCP HACK when receiving the segment with sequence number B

Normal ACK generated by receiving segment B

sender

s_buffer

\[ X \]: packet with sequence number X
Extended TCP HACK (cont'd)

The process of extended TCP HACK when receiving the segment with sequence number C.
The process of extended TCP HACK when receiving the segment with sequence number E
The process of extended TCP HACK when receiving the retransmitted segment with sequence number A
Extended TCP HACK (cont’d)

TCP Sender

- When sending out segment

- Header checksum option enabled?
  - No: Continue as normal TCP
  - Yes:
    1) Calculate header checksum of segment
    2) Place the header checksum into the header checksum option
    3) Continue as normal TCP

Modifications to the TCP sender
Extended TCP HACK (cont’d)

TCP Receiver

- when receiving a data segment
  - TCP segment corrupted?
    - Yes
      - Header portion corrupted?
        - Yes
          - Discard Packet
        - No
          - 1) Recover sequence number of corrupted segment from header.
          - 2) Save this sequence number in the s_buffer
          - 3) Generate special ACK containing all the sequence numbers in the s_buffer.
    - No
      - Continue as normal
Extended TCP HACK (cont’d)

ACK Processing

- ACK segment received

Is this a special ACK (with option 16)?

- No: Continue as normal
- Yes:
  1) Extract sequence numbers from the ACK option
  2) Selectively retransmit these segments
  3) ACK is discarded without further processing

Modification to the ACK processing
Improved Freeze-TCP

Problem of Freeze-TCP

- Since the ZWPs (Zero Window Probe) are exponentially backed off, there is the possibility of substantial idle time after a reconnection.

- If the disconnection period was long and the reconnection happened immediately after losing a ZWP from the sender. The sender will go into a long back-off before sending the next probe.

- Meantime the receiver has already reconnected, but the connection remains idle until the sender transmits its next probe.
**Improved Freeze-TCP (cont’d)**

✓ **Solution of Freeze-TCP:**

- As soon as a connection is re-established, the receiver sends 3 copies of the ACK for the last data segment it received prior to the disconnection.
- This causes the sender to immediately re-transmit one segment, which eliminates the waiting period.
- Fast re-transmit /fast recovery algorithms will set the congestion window to a half of its current window.
- This will degrade the performance of TCP.
Our solution

- Use the extended TCP header checksum ACK option proposed in Extended TCP HACK
- As soon as a connection is re-established, the receiver sends at least one copy of the special ACK with option kind,16, but no data in the ACK option.
- When sender receives the special ACK with no data, it restarts to send the packets, does not shrink its congestion window. Then discards these special ACK without further processing.
Improved Freeze-TCP (cont’d)

The process of improved Freeze- TCP HACK when connection is re-established

Extended TCP Header

sender

receiver

s_buffer

kind=16    len=32
Adaptive TCP

- MH (Mobile Hosts)
- FH (Fixed Host)
- BS (Base Station)
- Wired links
- Wireless links

Extended TCP HACK is triggered if BER is high.

Improved Freeze-TCP is triggered if handoff occurs.
Work flow of adaptive TCP

1. add extended TCP HACK to original TCP
2. extended TCP HACK
3. disconnections occurs
4. Triggers improved Freeze-TCP
5. Disconnections recovered?
   - No
   - Yes

Note: The diagram shows the process of adding an extended TCP HACK to the original TCP, triggering improved Freeze-TCP, and checking if disconnections are recovered.
Simulation Model

Simulation Parameters
(simulation tool: OPNET modeler v8.0)

Bandwidth of wireless link: 384Kbytes
MSS: 536bytes

Burst length: 3 packets
Size of s_buffer: 40Byte
Throughput for various packet loss rate

Packet loss only on the forward path

- Original TCP
- TCP HACK
- Adaptive TCP

Throughput (packets/s) vs. Percentage Packet Loss (%)

- Throughput decreases as packet loss increases.
- TCP HACK exhibits improved performance compared to Original TCP and Adaptive TCP under high packet loss conditions.
Throughput for various packet loss rate

Packet loss on both forward and reverse paths

![Graph showing throughput (packets/s) vs. percentage packet loss (%)]

- Original TCP
- TCP HACK
- Adaptive TCP
Transfer time for various disconnect time

- Original TCP
- Freeze-TCP
- Adaptive TCP

Transfer time vs Disconnect time

- 300s
- 150s
- 120s
- 90s
- 60s
- 30s

- 0.5ms
- 1ms
- 10ms
- 0.1s
- 0.5s
- 1s
- 3s
## Transfer time for various protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Time needed to finish all the transmissions (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original TCP</td>
<td>2200</td>
</tr>
<tr>
<td>Freeze-TCP</td>
<td>1300</td>
</tr>
<tr>
<td>TCP HACK</td>
<td>980</td>
</tr>
<tr>
<td>Adaptive TCP</td>
<td>320</td>
</tr>
</tbody>
</table>
1. **Drawbacks**

- The software overload;
- The improved Freeze-TCP needs the receiver to predict an impending disconnection.
Conclusions

✓ With our proposal, we can get an enhanced TCP which can work efficiently in mobile environments.
✓ Our proposed protocol has following characteristics:

1. An end-to-end TCP
2. Does not require an intermediate node to do TCP modifications
3. Handles end-to-end encrypted traffic
4. Does not need symmetric routing
5. Handles long time disconnections
6. Handles frequent disconnections
7. Handles high BER (bit error rate)
References

[20] Jinsheng Sun, Moshe Zukerman‡, King-Tim Ko, Guanrong Chen and Sammy Chan, Effect of Large Buffers on TCP Queueing Behavior, IEEE Infocom 2004
[22] Ren Wang, Giovanni Pau, Kenshin Yamada, M.Y. Sanadidi, and Mario Gerla, TCP Startup Performance in Large Bandwidth Delay Networks, IEEE Infocom 2004
Thanks!