Lecture 8: Cross-Layer Design in TCP

Hung-Yu Wei
National Taiwan University
TCP in Wireless and Mobile Networks

• References
TCP Overview
TCP operations

- Establishing TCP connection: 3-way handshake
  - Source sends TCP msg (SYN flag set) to Destination
    - SN=i (Sequence #)
  - Destination responds with ACK flag set
    - AN=i+1 (Acknowledgement #), SN=j
- Data
  - AN=j+1

SN: sequence number
AN: acknowledge number
TCP operations

- Terminating TCP connection: 3-way handshake
  - (1) FIN flag set. ACK flag set
    - Indicate the intention for termination
  - (2) FIN flag set. ACK flag set
    - Respond the intended termination
  - (3) ACK flag not set
    - Last message
Wireless TCP

• Challenges in wireless and mobile environment
  – High BER (bit-error rate) in wireless medium
  – Mobility

• How transmission error affect TCP?
  – TCP thinks packet losses are due to congestion
    • Reduce congestion window size
    • Back off its retransmission timer
    • Reduce throughput
Transmission Control Protocol

- Provide reliable transport layer mechanism
- Congestion control and avoidance
- Window-based flow control
- Use ACKs to acknowledge all received bytes
- Use byte sequence number
TCP window flow control

- Additive increase and multiplicative decrease
- Increase window (2 phase)
  - Slow start
  - Congestion avoidance
- Decrease window
  - When detecting packet loss
  - Assume congestion occur
Slow Start

- Additive increase
- Congestion window cwnd
  - Initial value: cwnd=1 MSS (Maximum Segment Size)
  - Receiving every ACK: cwnd=cwnd+1
  - cwnd grows exponentially
- Slow start stops when reaching slow start threshold
Congestion Avoidance

- When receive every ACK
  \[ cwnd = cwnd + \frac{1}{cwnd} \]
TCP Fast Retransmission

• Use duplicate ACKs
  – Repeating ACKs for the same packet indicates loss of the packet

• Fast Retransmission
  – When sender receives duplicate ACKs
  – Don’t wait until timeout
TCP in Lossy Link

TCP Source

Packets Stored at Sender

1 2 3 4 5

congestion window = 5

Packets in Flight

TCP Receiver

Acknowledgments Returning

Base Station

Lossy Link

TCP in Lossy Link
Variants of TCP and TCP mechanisms
TCP (Transport Control Protocol)

• End-to-end transport layer protocol

• Functionalities
  - Congestion control
  - Flow control
  - Reliable transmission
  - Sequenced delivery
Window Flow Control

- ~ W packets per RTT
- Lost packet detected by missing ACK
Effect of Congestion

- Packet loss
- Retransmission
- Reduced throughput
- Congestion collapse due to
  - Unnecessarily retransmitted packets
  - Undelivered or unusable packets
- Congestion may continue after the overload!
Congestion Control

• TCP seeks to
  – Achieve high utilization
  – Avoid congestion
  – Share bandwidth

• Window flow control
  – Source rate = \( \frac{W}{RTT} \) packets/sec
  – Adapt \( W \) to network (and conditions)
    \( W = BW \times RTT \)
TCP Window Flow Controls

- **Receiver flow control**
  - Avoid overloading receiver
  - Set by receiver
  - $awnd$: receiver (advertised) window

- **Network flow control**
  - Avoid overloading network
  - Set by sender
  - Infer available network capacity
  - $cwnd$: congestion window

- Set $W = \min (cwnd, awnd)$
Receiver Flow Control

• Receiver advertises `awnd` with each ACK
• Window `awnd`
  – closed when data is received and ack’d
  – opened when data is read
• Size of `awnd` can be *the performance limit* (e.g. on a LAN)
  – sensible default ~16kB
Network Flow Control

- Source calculates *cwnd* from indication of network congestion
- Congestion indications
  - Losses
  - Delay
  - Marks
- Algorithms to calculate *cwnd*
  - Tahoe, Reno, Vegas, RED, REM ...
TCP Flavors

- **TCP-Tahoe**
  - $W=1$ adaptation on congestion

- **TCP-Reno**
  - $W=W/2$ adaptation on fast retransmit, $W=1$ on timeout

- **TCP-newReno**
  - TCP-Reno + fast recovery

- **TCP-Vegas**

- **TCP-SACK**
Slow Start

- Start with $cwnd = 1$ (slow start)
- On each successful ACK increment $cwnd$
  $$cwnd \leftarrow cnwd + 1$$
- Exponential growth of $cwnd$
  each RTT: $cwnd \leftarrow 2 \times cwnd$
- Enter \textit{CA} when $cwnd \geq ssthresh$
Slow Start

cwnd ← cwnd + 1 (for each ACK)
Congestion Avoidance

- Starts when $cwnd \geq ssthresh$
- On each successful ACK:
  \[ cwnd \leftarrow cwnd + \frac{1}{cwnd} \]
- Linear growth of $cwnd$
  each RTT: \[ cwnd \leftarrow cwnd + 1 \]
Congestion Avoidance

cwnd ← cwnd + 1 (for each cwnd ACKS)
Packet Loss

- **Assumption:** loss indicates congestion
- **Packet loss detected by**
  - Retransmission TimeOuts (RTO timer)
  - Duplicate ACKs (at least 3)

![Diagram showing packet loss and acknowledgments]
TCP Tahoe (Jacobson 1988)

SS: Slow Start
CA: Congestion Avoidance
Summary: Tahoe

- Basic ideas
  - Gently probe network for spare capacity
  - Drastically reduce rate on congestion
  - Windowing: self-clocking
  - Other functions: round trip time estimation, error recovery

```plaintext
for every ACK {
    if (W < ssthresh) then W++ (SS)
    else W += 1/W (CA)
}
for every loss {
    ssthresh = W/2
    W = 1
}
```
TCP Reno (Jacobson 1990)

SS: Slow Start
CA: Congestion Avoidance

Fast retransmission/fast recovery
Fast Retransmit

- Wait for a timeout is quite long
- Immediately retransmits after 3 dupACKs without waiting for timeout
- Adjusts ssthresh
  \[ \text{flightsize} = \min(\text{awnd}, \text{cwnd}) \]
  \[ \text{ssthresh} \leftarrow \max(\text{flightsize}/2, 2) \]
- Enter Slow Start (cwnd = 1)
Fast recovery

- Motivation: prevent `pipe' from emptying after fast retransmit
- Idea: each dupACK represents a packet having left the pipe (successfully received)
- Enter FR/FR after 3 dupACKs
  - Step (I): window inflation
  - Step (II): window deflation
  - Step (III): enter CA
Implications

• Congestion measure = end-to-end queueing delay

• At equilibrium
  – Zero loss
  – Stable window at full utilization
  – Approximately weighted proportional fairness
  – Nonzero queue, larger for more sources

• Convergence to equilibrium
  – Converges if sufficient network buffer
  – Oscillates like Reno otherwise
Issues in Wireless Environment
Wireless Transmissions

• Limited bandwidth
• Long Round Trip Time
  - Severe issue in satellite communications
  - Current 3G cellular also tries to reduce RTT
• High BER
  - Strong FEC might be inefficient
• Short Flows
  - HTTP applications need several small packet transmissions
• Power consumption
  - Long TCP session might be power inefficient
TCP with mobile hosts

- Mobile IP
  - Designed for handoff less frequent than 1 per sec
- Handoff results in packet loss
- Might need to distinguish packet losses due to congestion and mobility
Problems

- TCP considers packet losses as indication of congestion
  - Trigger congestion control
- In wireless and mobile networks, packet losses could be due to
  - High BER of wireless transmission
  - Handoff
- Frequent TCP congestion window backoff results in
  - Small window size
  - Limited throughput
Classifications of Solutions

• Link Layer
  - (1) Robust link layer
    • FEC/ARQ
  - (2) TCP-Aware Link Layer solution
    • Snoop
  - (3) TCP-Unaware Link Layer solution
    • TULIP
    • Delayed Duplicate ACK

• Split connection (Transport Layer)
  - Indirect TCP (I-TCP)
  - M-TCP
Classifications of Solutions (2)

- **Modified version of TCP**
  - TCP SACK
  - TCP FACK
  - TCP Santa Cruz

- **Other changes to TCP**
  - Change ACK strategy
  - Increase TCP’s Initial Congestion Window
  - Explicit Congestion Notification
  - Explicit Loss Notification
  - Fast Retransmit

- **New Transport Layer**
  - Wireless TCP

- **WAP protocol stacks**
Link Layer Solution
Link Layer Schemes

• Hide Wireless Link Error from TCP
  – Emulate a wired link
    • Reduce BER with FEC
    • Hide retransmission (L2) from L4
  – Reduce the error rate in wireless medium
    • Automatic Repeat Request (ARQ)
    • Forward Error Correction (FEC)
    • Hybrid FEC/ARQ scheme

• TCP-aware Link Layer Solution
  – Snoop
Link Level Retransmissions

TCP connection

application
transport
network
link
physical

application
transport
network
link
physical

application
transport
network
link
physical

Link layer state

Local Retx

wireless

Local Retx
**Link Layer: Strength/Weakness**

**Advantages**
- No connection state to be maintained at BS
- Layered structure
- Shielding the loss to higher-layer protocols

**Disadvantages**
- Do not solve the mobility issue
- Small timeout value for ARQ
- Duplicate re-transmission effort
Related work

• **FEC**
  - Most communication systems use FEC

• **ARQ**
  - Cellular system typically uses ARQ
  - In some systems, ARQ is optional (e.g. 802.16)

• **AIRMAIL (1995)**
  - FEC+ARQ

• **Hybrid ARQ**
  - FEC+ARQ
  - Soft combining
  - Promising technique in many systems
Snoop

H. Balakrishnan et al.
Snoop

- Blakrishnan, Padmanabhan, Seshan, Katz
  - SIGCOMM 1996
- TCP-aware Link Layer solution
- Snoop Agent locates at BS
- No change required in wired network
- Snoop Agent monitors all TCP session
  - Maintain un-acknowledged TCP segment information at cache
  - Monitor both uplink/downlink
- Soft state
Snoop Agent Operation

• Packet loss detection
  – Receive duplicate ACKs
  – Local timeout

• Hide the packet loss in wireless network
  – Retransmit cached packet
  – Dropping duplicate ACKs at BS to prevent from fast retransmission from source
Snoop Protocol

Per TCP-connection state

TCP connection

- application
- transport
- network
- link
- physical

- application
- transport
- network
- link
- physical

- application
- transport
- network
- link
- physical

FH - BS - MH

wireless
Example assumes delayed ack - every other packet ack’d
Snoop: Example
Snoop : Example

Duplicate acks are not delayed
Snoop: Example

Duplicate acks
Discard dupack

Dupack triggers retransmission of packet 37 from base station

BS needs to be TCP-aware to be able to interpret TCP headers
Snoop : Example
TCP sender does not fast retransmit
TCP sender does not fast retransmit
Snoop : Example
Design Goal

• TCP sender should not take any congestion control when encounter a packet loss due to wireless transmission error
• TCP sender should just retransmit the packet due to wireless transmission error
• Sender should be shielded by wireless transmission error
Snoop: Strength/Weakness

**Advantages**
- Shielding sender from wireless link error
- Soft state connection is used

**Disadvantages**
- Link layer overhead
- Violate modular layer design principle
- Mobile-side TCP need to be modified
- Modification at BS
Split Connection Approach

• Split TCP end-to-end connection into two connections
  – Sender to BS
  – BS to receiver

• Handle wireline and wireless TCP separately

![Diagram showing FN, BS, and MH with wireless link between FN and BS, and BS and MH.]

Fixed Node  Base Station  Mobile Node
Split Connection: I-TCP

ICDCS 95
Bakre & Badrinath
Split Connection Schemes

• Separate packet loss in wireless link and wired link
  – Independent TCP flow control in wireline and wireless linisk
  – May use special transport layer protocol and control parameters in wireless/wired link

• I-TCP (Indirect TCP)
  – Bakre and Badrinath.
    • ICDCS 95
  – Use two standard TCP connections
I-TCP

• MSR (Mobility Support Router)
  - Split TCP operation: 2 logical connections
  - MSR acknowledges on behalf of the mobile host

• Operations
  - Normal TCP between MSR and Sender
  - "wireless TCP" between MSR and mobile host
    • Separate "wireless link" and "user mobility"

• Performance
  - Sender sees more stable connection
  - Recover loss packets faster
  - Handle mobility
Split Connection Approach

TCP connection

application
transport
network
link
physical

TCP connection

application
transport
network
link
physical

Per-TCP connection state

rxmt

wireless

63
Split Connection: Strength/Weakness

- **Advantages**
  - Do not modify wireline network
  - Possible for special TCP design in wireless network
    - Light-weighted
    - Energy-efficient

- **Disadvantages**
  - Modification at BS
  - Hard state maintain at BS
  - Violate TCP end-to-end paradigm
  - User mobility will cause problems
Split Connection: M-TCP

ACM CCR 97
Brown & Singh
\textbf{M-TCP}

- M-TCP uses TCP Persist Mode
  - When a \textbf{new} ack is received with receiver’s advertised window $= 0$, the sender enters persist mode
  - Sender does not send any data in persist mode
    - except when persist timer goes off
  - When a positive window advertisement is received, sender exits persist mode
  - On exiting persist mode, \textbf{RTO} and \textbf{cwnd} are same as before the persist mode
M-TCP

- Split connection: 2 logical connections
- BS acknowledges “after” receiving ACKs from MH
- Handoff
  - Withheld ack sent with window advertisement = 0, if MH moves away (handoff in progress)
  - Sender is in persist mode during handoff
  - Sender exits persist mode after handoff, and starts sending packets using same cwnd as before handoff
**Compressed M-TCP**

- Modified version of M-TCP
  - Proposed by the same authors
- Apply compression to improve performance under limited bandwidth
Modification: TCP SACK
Selective Acknowledgements

• TCP-SACK (Selective Acknowledgement)
• Contain sequence number information of the packet that cause the cumulative ACKS
• TCP-SACK can be used along with schemes mentioned before (LL, split-connection, end-to-end)
**SACK** (RFC 2018, RFC 2883)

- **Motivation:** Reno & NewReno retransmit at most 1 lost packet per RTT
  - Pipe can be emptied during FR/FR with multiple losses
- **Idea:** SACK provides better estimate of packets in pipe
  - SACK TCP option describes received packets
  - On 3 dupACKs: retransmits, halves window, enters FR
  - Updates $\text{pipe} = \text{packets in pipe}$
    - Increment when lost or new packets sent
    - Decrement when dupACK received
  - Transmits a (lost or new) packet when $\text{pipe} < \text{cwnd}$
  - Exit FR when all packets outstanding when FR was entered are acknowledged
SACK Operation

• Selective ACK
• Establishment
  - Need to negotiate during TCP connection setup
• SACK operations
  - Active when sender receives 3 duplicated ACKs
  - Retransmit the 1st un-ACKed and un-SACKed segment
• SACK performs better in several scenarios
  - Not in all scenarios
• SACK is optional
  - But implement in many OSs (e.g. MS Windows)
SACK : Strength/Weakness

• **Advantages**
  – Could be used with other schemes
  – End-to-end

• **Disadvantages**
  – May not perform well with high loss-rate
  – Change TCP mechanism
End-to-End Approaches: ELN
End-to-End Schemes

- Explicit Loss Notification (ELN)
  - Add ELN option to TCP ACK
  - When a packet is dropped by wireless link, ELN in ACK which corresponding to the packet is set
  - When sender receives the ELN set, it retransmit packet without reducing congestion window
  - Indicate non-congestion packet loss
  - Get Link status from Link Layer
  - Useful in multi-hop networks
End-to-End : Strength/Weakness

• **Advantages**
  – Easily distinguish congestion loss and non-congestion loss

• **Disadvantages**
  – Wireline-side TCP (i.e. TCP source node) needs to be modified
  – Slow recovery of lost packets
Compared to Snooping

- Similar to Snoop
  - TCP-aware agent to observe packet loss due to wireless transmission
    - Same drawback (need to modify BS)
- Modification of TCP source node
- Protocol dependent (TCP-aware)
New Transport Protocol: WTCP

Sinha et al.
ACM Mobicom 1999
Wireless Transmission Control Protocol

- Design for WWAN

- Idea
  - Decouple "rate control" and "reliability"
    - Rate control information
    - SACK

- Receiver
  - Measure inter-packet delay
  - Estimate reason of segment loss (transmission error or congestion)
  - Ask the sender to increase or decrease or maintain the sending rate
WTCP

• **Advantages**
  - Perform well in certain scenarios

• **Disadvantage**
  - **Complexity**
    - Complicated operation
    - New protocol
  - **Might not power efficient**
    - Receiver complexity
Conclusions
TCP in “Wireless” Networks

• Link Layer Approach
  - FEC, ARQ

• Split Approach
  - Link Layer Split (Snoop)
  - Transport Layer Split (I-TCP)

• End-to-End Approach
  - Modify TCP (TCP SACK)
  - Change ACK algorithm
  - New Transport Layer Protocol
    • New protocol suite (e.g. WAP)
TCP in “Mobile” Networks

- Techniques to alleviate TCP misbehaviors caused by “wireless” packet loss might NOT work
  - Some techniques still work
    - Decouple “packet loss == congestion”
  - Split approaches have trouble in “mobile” environment
- Reduce packet losses during handoff
  - Seamless handoff
    - Reduce handoff latency
    - Packet forwarding is useful
Other Issues

• Link Asymmetry
• Long RTT
  - TCP in 3G
• Data and ACK compete for transmission
  - E.g. 802.11
• Frequent route change
  - Ad hoc networks
  - Moving fast result in more trouble?
    • Answer: it depends....
• 802.11-based multihop network is bad for TCP connections