TCP for today’s Web
Race to Increase Connective and Improve End-User Performance
Agenda

• TCP Background

• Changes in Internet and Web pages

• TCP Fast-Open

• TCP
TCP Revisited: Opening a New Connection

• TCP 3-way handshake
TCP Revisited: Recovery From Packet Loss

- TimeOut Based retransmission
- Fast recovery based retransmission
TCP Revisited: Recovery From Packet Loss

• Fast-Retransmit:
  – If 3 duplicate ACKs
  – Then resend
  – Must receive 3 packets

• Time-Out
  – If ACK is not received within ~300ms
  – Then resend

• Fast-Retransmit is much faster than Time-out

• What happens if the last packet is lost?
Agenda

• TCP Background

• Changes in Internet and Web pages

• TCP Fast-Open

• TCP
How was the Internet Changed over time?

- Network connections have more bandwidth
  - Need to rethink TCP’s Parameters

- Network are more secure: widespread use of HTTPS
  - Need to make optimizations for HTTPS
The World is Growing Increasingly Aware of Privacy Violations

WikiLeaks Reveals NSA Spying On Japanese Government Officials

YOU’LL NEVER GUESS HOW THE NSA MAN FOR YAHOO!

National surveillance: AT&T, NSA partner in internet spying
• TLS increase the overhead from 1 RTT to 3 RTT.
How was the Internet Changed over time?

• Network connections have more bandwidth
  – Need to rethink TCP’s Parameters

• Network are more secure: widespread use of HTTPS
  – Need to make optimizations for HTTPS

• Objects on the web are smaller
  – Need to Optimize TCP inefficiencies
What Does A WebPage look like Today?

- Web-page < 320KB
- but objects are small 7.5KB - 2.4KB [25]
- lots of small objects in a page.
- Objects from 7 hosts
- 66%-90% compressed
How was the Internet Changed over time?

• Network connections have more bandwidth
  – Need to rethink TCP’s Parameters

• Network are more secure: widespread use of HTTPS
  – Need to make optimizations for HTTPS

• Objects on the web are smaller
  – Need to Optimize TCP inefficiencies

• Widespread use of Mobile/Wireless connections
  – Need to include Wireless specific optimizations
Agenda

• TCP Background

• Changes in Internet and Web pages

• TCP: New Initial Congestion Window

• TCP Fast-Open (TFO)

• Reducing Web Latency
TCP

• Initial Congestion Window
  – Number of packets sent after the handshake

• Throughput = Window/RTT
  – # of Packets/RTT

• Early 20s IW = 3 pkts
  – Network is highly under-utilized
  – But many pages are small but > 3 packets
New Initial Congestion Window

- Google Proposes: IW=10
- Lots of interesting questions:
  - How does this change the loss rate? Will there be more congestion?
  - Does this actually improve performance?
Agenda

• TCP Background

• Changes in Internet and Web pages

• TCP Fast-Open (TFO)

• TCP
Improving 3 Way Handshake

• Original TCP 3-way handshake

Hello
Let's talk
+Data
Oh, Hello
Our Motivation and Goal

TCP Handshake Significantly Slows Down Connection

Goal: Eliminate the overhead of TCP’s Handshake.

Without introducing new Security Problems.
Improving 3 Way Handshake

• Original TCP 3-way handshake
  Hello
  Let's talk
  +Data
  Oh, Hello

• TCP Fast Open
  Hello+
  Data
  Let's
  talk+Data
  Oh, Hello+Data
Problems?
New Attack: SYN Flood

Attacker Make up a random ip address
Send a bunch of SYN pks with data

• Server maintains state for each connection
• Server uses CPU to process each connection

Note: the attacker uses fake address, the response from the server do not go to the attack there need not maintain state
Old Attack: Reflection Attack

- Attacker makes up a random IP address
- Sends a bunch of SYN packets with data

- Similar to SYN-Flood Attack
- But the main purpose is to Denial of Server attack
  - The data in the SYN could be a GET request for a page (~10KB)
  - The response (SYN-ACK), would be the web-page (~320KB)
- There’s an amplification of data through the server
Attack Model

• Attack work because:
  – Attacker can spoof (make-up) SRC IPs.

• To prevent this:
  – TFO adds a cookie to the protocol
  – Client must include cookie in the handshake
  – The cookie is an encrypted version of source IP
    • Source IP encrypted with the server’s private key
    • Server unencrypts cookie and compares
System Assumptions

Acceptable Changes:

• 1. symmetric crypto
  – (can be done in fast path) but no asymmetric.

• 2. soft state
  – (can't keep permanent state-- scale issues)

• 3. minor App changes
  – Don’t want to prevent adoption
TCP Fast Open

• First Connection to the server is like a normal TCP handshake
  – Use TCP Options to negotiate TFO
  – **No data is sent!!!**

• Cookies exchanged
  – Client has to store the cookie, so no state on server
TCP Fast Open

• All future connections are now different
  – data is sent!!!

• Use cookie in the initial SYN packet
Attacks Revisited

• Reflection
  – To get cookie must compromise host or network
    • If you can then you don’t need reflection

• SYN Flood
  – We limit the number TFO connections
  – So server is still always willing to accept regular TCP connections
What are Some Deployment Issues?
What are Some Deployment Issues?

• Middleboxes are HORRIBLE
  – They drop new TCP options
  – NAT changes IP addresses so cookies can’t work

• Fall back to normal TCP
Agenda

- TCP Background
- Changes in Internet and Web pages
- TCP Fast-Open (TFO)
- TCP: New Initial Congestion Window
- Reducing Web Latency
Ways to Reduce Latency: The State of the Art

High loss, high delay
Ways to Reduce Latency: The State of the Art

Improve the proximity of services to the user
Leverage multi-stage connections

High loss, shorter delay
Low loss, multiplexed
Evaluating TCP Performance

High loss, shorter delay

Low loss, multiplexed

Analyzed billions of flows carrying Web traffic between Google and clients
Transfers With Loss Are Too Slow

Loss makes Web latency 5 times slower

Delays caused by TCP loss detection and recovery

6% of transfers between Google and clients are lossy
Retransmission Timeouts Are Expensive

77% of losses are recovered by retransmission timeouts.

Retransmission timeouts can be 200 times larger than the RTT.

Caused by high RTT variance, or lack of samples.
Tail Drops Are Expensive

(Single) tail packet drop is very common

Tail packets are twice as likely to be dropped compared to packets early in a burst

35% of lossy bursts observe only one packet loss
Our Motivation and Goal

Loss significantly slows down transfers. Due to frequent recovery via slow RTOs. Caused by tail loss.

Goal: Approaching the ideal of loss detection and recovery without delay.

Without making the protocol too aggressive.
Setting

Preference for solutions without client changes and middlebox compatibility

Controlling server only

Latency-sensitive traffic is a small portion of traffic mix

Controlling client and server
Setting

Frontend

Server

Backend

Server

Private Network

Public Network

Reactive

Trigger fast retransmit by retransmitting the tail packet early

Proactive

Avoid retransmissions through packet duplication

Corrective

Add redundancy to enable recovery without retransmission, or trigger fast retransmit
Setting
Frontend Server
Backend Server
Private Network
Public Network

Reactive
- Trigger fast retransmit by retransmitting the tail packet early

Proactive
- Avoid retransmissions through packet duplication

Corrective
- Add redundancy to enable recovery without retransmission, or trigger fast retransmit
Reactive

Wait time until RTO

Receiver does not know about the loss and therefore cannot send signals back

Wait time until RTO
Reactive

Wait for two RTTs

Fast retransmit
Retransmit new packet or previous (tail) packet after two RTTs
Can trigger selective acknowledgement indicating loss
Speeds up loss detection
Reactive: Detecting Masked Losses

- Wait for two RTTs
- Cannot ignore the case where a packet loss is recovered by the Reactive probe
- Count ACKs and reduce congestion window if only one ACK for tail packet received
Reactive: Detecting Masked Losses

Wait for two RTTs

One ACK only:
Loss → Reduce congestion window

Two ACKs:
No loss

Wait for two RTTs
Frontend Server

Public Network

Private Network

Reactive
- Trigger fast retransmit by retransmitting the tail packet early

Proactive
- Avoid retransmissions through packet duplication

Corrective
- Add redundancy to enable recovery without retransmission, or trigger fast retransmit
Proactive

Wait time until RTO
Avoid almost all retransmissions through packet duplication

Wait time until RTO

1 - 3
3
Proactive

Avoid almost all retransmissions through packet duplication

Duplicates are used if original transmission was lost

Avoids loss detection and recovery
A/B Experiment Setup
Frontend Server
Backend Server
Reactive
Proactive
Experimented in production environment
serving billions of queries
(millions of queries are sampled)
15-day experiment, 2.6 million queries sampled:
mean response time reduced by 23%
99th percentile response time reduced by 47%

Impact of Proactive:
Retransmission rates on the backend connection dropped from 0.99% to 0.09%

Impact of Reactive:
Almost 50% of retransmission timeouts on the frontend connection are converted to fast retransmits
Corrective: The Middle Way

Reactive speeds up loss detection, but still requires recovery.

Proactive avoids loss detection and recovery, but has 100% overhead.
Setting

Frontend

Server

Backend

Server

Private Network

Public Network

Reactive

Trigger fast retransmit by retransmitting the tail packet early

Proactive

Avoid retransmissions through packet duplication

Corrective

Add redundancy to enable recovery without retransmission, or trigger fast retransmit
Corrective: 
Forward Error Correction in TCP

Add redundancy to enable recovery without retransmission

Wait time until RTO
Corrective:
Forward Error Correction in TCP

Encodes previously transmitted segments in few coded segments

XOR coding can recover single packet loss at the receiver

Signaling of recovery status to the sender to enforce congestion control or fast retransmit

No loss detection required

Speeds up loss detection and recovery
What are Some Deployment Issues?
What are Some Deployment Issues?

• Middleboxes are HORRIBLE
  – They drop new TCP options
  – They rewrite sequence numbers

• Fall back to normal TCP
Tail latency reduced by more than 20% but: performance slightly worse on loss-free connections.
Lessons

• A webpage === lots of small objects
  – Harder to ameliorate overheads
    • TCP Handshake overhead
    • 2 RTT $\Rightarrow$ Loss detection

• Most common loss pattern in WAN
  – Last packet in small connection
  – Use redundancy (FEC) to overcome this.

• Eliminate Handshake overhead
  – Send packet during the handshake
  – Insert secret cookie in handshake to eliminate attacks
Concluding Remarks

• Google Fiber is growing and internet connectivity is growing against. Do we need a new IW?

• Optimizations are specific to N. America, Europe and parts of S.E Asia
  – Do we need specific optimizations for other parts of the world?
  – Look up Facebook’s Internet.org