Security and Project 3

Tapan and Ghaith
HW 4 due 05/23
Project 1 Updated Submission due 11 PM Friday 05/12
  ○ In-person/Zoom OH on Friday:
    ■ 11:30 AM - 12:30 PM (Location TBD)
    ■ 2:30 PM - 3:30 PM (Location TBD)
    ■ Will confirm on ed today
Symmetric (Secret Key) Encryption

- Alice and Bob have the same secret key, $K_{AB}$
  - Anyone with the secret key can encrypt/decrypt

```plaintext

<table>
<thead>
<tr>
<th>Alice</th>
<th>Encrypt</th>
<th>Ciphertext</th>
<th>Hi there</th>
<th>Decrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plaintext</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I love networks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secret key</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plaintext</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I love networks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secret key</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Introduction to Computer Networks
Public Key (Asymmetric) Encryption

- Alice and Bob have public/private key pairs \((K_B, K_{B^{-1}})\)
  - Public keys are well-known, private keys are secret

Alice

Bob

Bob’s public key

\(K_B\)

Encrypt

Ciphertext

\(Hi\ there\)

Decrypt

Bob’s private key

Introduction to Computer Networks
Public Key Encryption (2)

- Alice encrypts with Bob’s pubkey $K_B$; anyone can send.
- Bob decrypts with his private key $K_B^{-1}$; only he can.

**Diagram:**

Alice Decrypts Hi-there with her private key $K_A$. Bob Receives Hi-there ciphertext and Decrypts with his private key $K_B^{-1}$.
How can we trust a public key?
Certificates

• A certificate binds pubkey to identity, e.g., domain
• Distributes public keys when signed by a party you trust
• Commonly in a format called X.509

I hereby certify that the public key
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
belongs to
Robert John Smith
12345 University Avenue
Berkeley, CA 94702
Birthday: July 4, 1958
Email: bob@superdupernet.com

Signed by CA
PKI (Public Key Infrastructure)

• Adds hierarchy to certificates to let parties issue
• Issuing parties are called CAs (Certificate Authorities)

I certified the ABC website!

RA 2 is approved. Its public key is 47383AE349...
Root's signature

RA 2 is approved. Its public key is B3EAE49...
Root's signature

CA 5 is approved. Its public key is 6384AF863B...
RA 2’s signature
PKI (2)

- Need public key of PKI root and trust in servers on path to verify a public key of website ABC
  - Browser has Root’s public key
  - \{RA1’s key is X\} signed Root
  - \{CA1’s key is Y\} signed RA1
  - \{ABC’s key Z\} signed CA1
PKI (3)

• Browser/OS has public keys of the trusted roots of PKI
  • >100 root certificates!
  • That’s a problem …
  • Inspect your web browser

Certificate for wikipedia.org issued by DigiCert
PKI (4)

- Real-world complication:
  - Public keys may be compromised
  - Certificates must then be revoked
- PKI includes a CRL (Certificate Revocation List)
  - Browsers use to weed out bad keys
<table>
<thead>
<tr>
<th></th>
<th><a href="http://www.google.com">www.google.com</a></th>
<th>GTS CA 1C3</th>
<th>GTS Root R1</th>
<th>GlobalSign Root CA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subject Name</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td>BE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td>GlobalSign nv-sa</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organizational Unit</td>
<td>Root CA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Common Name</td>
<td>GlobalSign Root CA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Issuer Name</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td>BE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td>GlobalSign nv-sa</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organizational Unit</td>
<td>Root CA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Common Name</td>
<td>GlobalSign Root CA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Validity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Before</td>
<td>Tue, 01 Sep 1998 12:00:00 GMT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not After</td>
<td>Fri, 28 Jan 2028 12:00:00 GMT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bufferbloat

“Bufferbloat is a cause of high latency in packet-switched networks caused by excess buffering of packets” – Wikipedia
Let’s revisit network delays

● Previously we studied how to calculate delays for a network
  ○ RTT: ~typically defines the latency for a small packet round-trip
  ○ Assumption: Network is unloaded (nothing else going on)
  ○ This is not often true in practice
● What changes when something else is going on?
  ○ [https://www.waveform.com/tools/bufferbloat](https://www.waveform.com/tools/bufferbloat)
  ○ Observe unloaded vs loaded latency
● What’s going on here?
Let’s investigate this further…

- Recall the equations to calculate latency
  
  \[
  \text{Latency} = \text{Propagation} + \text{Transmit} + \text{Queue}
  \]
  \[
  \text{Propagation} = \frac{\text{Distance}}{\text{SpeedOfLight}}
  \]
  \[
  \text{Transmit} = \frac{\text{Size}}{\text{Bandwidth}}
  \]

- Queueing delay: How long packet spends in switch buffers
  
  - Queueing delay \(\propto\) # packets queued
  - For unloaded network: 0
  - For loaded network: depends how packets are sent
What happens with TCP Reno

- Host doesn’t know the bandwidth of the bottleneck link.
- TCP Reno relies solely on packet losses to guide how fast to send.
  - It keeps sending faster and faster until a packet drops.
- With a queue, this can fill up the queue pretty quickly.
- Larger the switch queue capacity, more the delay
  - Called Bufferbloat
Bufferbloat – Problem

- Suppose h1 knows to send at 1.5 Mb/s, what’s the RTT when the queue is full?
  - ...when it’s not full?
- TCP at the end of the day will operate at the bottleneck bandwidth, but is it necessary to fill up the queue?
How do solve this this?

- Problem: Packet loss is a bad signal of congestion
  - Aside: packet losses because of other reasons can also mess up performance

- What else can we do?

- Proposal #1: Active Queue Management
  - Switch tells sender that it’s running close to congestion
  - Explicit Congestion Notification (ECN): Mark a bit in the packet IP header
  - Challenges: Requires switch hardware support; If mixed with regular TCP flows, it takes over all the bandwidth because it works too well
  - Used in datacenters (DCTCP), not on the internet

- Proposal #2: Try to find the optimal point using measurements
  - BBR from Google is one deployed example
Bufferbloat aware transport: BBR

- **Optimum operating point:**
  - (Max BW, Min RTT)
  - Product is called Bandwidth Delay Product (BDP)

- **BBR**
  - Estimate min RTT, bottleneck bandwidth via probing and measuring performance.
  - Keep inflight bytes equal to BDP

---

**FIGURE 1: DELIVERY RATE AND ROUND-TRIP TIME VS. INFIGHT**

- **BDP**
  - App limited
  - Bandwidth limited
  - Buffer limited

- **BDP+ BtlBw**
  - Optimum operating point is here
  - Loss-based congestion control operates here

- **RTprop**
  - Slope = 1/BtlBw

- **Link Capacity**
  - Delivery rate
  - Round trip time
  - Amount inflight
Project 3 – Goal

- Simulate bufferbloat problem.
- See the worse performance when queue size is larger.
- See the difference between TCP Reno and TCP BBR.
Experiment Setup

● Long-lived TCP flow from h1 to h2  
  ○ Simulate background traffic  

● Back-to-back ping from h1 to h2  
  ○ Measure RTT  

● Spawn a webserver on h1 and periodically fetch a page  
  ○ Simulate more important load  
  ○ Measure time  

● Plot time series of RTT and number of queued packets.

● Run the experiment with  
  ○ Q=20 and Q=100  
  ○ Reno and BBR  
  ○ 4 experiments total
Detour – Hypothesize

In groups of 3-4,...

- In your own words, what is bufferbloat problem?
- For each of the 4 experiments (Q=20 or 100; and with Reno or BBR),
  - How do the webpage fetch time compare?
- How would plot between queue size and time look like for TCP Reno?
In groups of 3-4,...

- In your own words, what is bufferbloat problem?
- For each of the 4 experiments (Q=20 or 100; and with Reno or BBR),
  - How do the webpage fetch time compare?
- How would plot between queue utilization and time look like for TCP Reno?

<table>
<thead>
<tr>
<th></th>
<th>Q=20</th>
<th>Q=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reno</td>
<td>&lt;;=;&gt;</td>
<td>&lt;;=;&gt;</td>
</tr>
<tr>
<td>BBR</td>
<td>&lt;;=;&gt;</td>
<td></td>
</tr>
</tbody>
</table>
Setup

- Use Mininet VM (same as Project 2)
- Get the starter code and install dependencies

```bash
cd ~
unzip project3.zip

sudo apt-get update

sudo apt install python3-pip

sudo python3 -m pip install mininet matplotlib
```
Starter Code

- **run.sh**
  - Run the entire experiment
    - Run `bufferbloat.py` on `q=20` and `q=100`
    - Generate latency and queue length graphs

- **bufferbloat.py**
  - Complete the TODOs
    - Setup the mininet topology and the experiment
    - Write shell commands to do the measurements
Long-lived TCP Flow

- Starter code sets up iperf server on h2
- Goal: start iperf client on h1, connect to h2
  - Should be “long-lasting”, i.e. for time specified by --time parameter
- How do I connect to a certain IP or make the connection long-lasting?
  - man pages are your friend!
  - type `man iperf` in a Linux terminal
Ping Train

- **Goal**: Start “ping train” between h1 and h2
  - Pings should occur at 10 per second interval
  - Should run for entire experiment

- **How do I specify the ping interval and how long the ping train runs?**
  - man pages are your friend!
  - type `man ping` in a Linux terminal

- **Write the RTTs recorded from `ping` to `{args.dir}/ping.txt**
  - See starter code comments for more detail
Download Webpage with curl

- Starter code spawns webserver on h1
- Goal: Use `curl` to measure fetch time to download webpage from h1
  - Starter code has hint on formatting curl command
  - Make sure `curl` doesn’t output an error
    - Errors report very small latency
- No need to plot fetch times; just need to report average fetch time for each experiment.
Plotting

- Starter code contains scripts for plotting, `plot_queue.py`, `plot_ping.py`:
  - Expects queue occupancy in $dir/q.txt, ping latency in $dir/ping.txt
  - Plots are useful for debugging!

- Part 3, run same experiments with TCP BBR instead of TCP Reno:
  - How do you expect the graph outputs to differ?