CSCI-1680
DNS
Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, Scott Shenker and John Jannotti
• TCP Milestone II: Should do your meeting by today
  – If you’re stuck, that’s okay—bring questions!
• HW4: Due Friday
• TCP officially due next Tuesday (11/22)
  – I will add hours tomorrow (or come by when my door is open)
  – Same for Monday/Tuesday

• Class next Tuesday => should be fun
What happens the rest of term?

- There will be a HW5
- Final project **NO LATE DAYS.**
  - Open-ended, you pick the topic
  - Can form new groups, or keep the same group
  - Designed to be MUCH lighter than TCP
  - Due just before exam period (12/12) (subject to change)
  - Document + group form to go out soon, start after Thanksgiving
Congestion control: the story so far

J NEED TO MAKE SURE WE DON'T OVERWHELM THE NETWORK.

Diagram:
- Y-axis: Throughput, Load, Link Capacity
- X-axis: Load
- Drop Packets
- Drops (Retransmits)
Slow start every time?!

- Losses have large effect on throughput
- **Fast Recovery (TCP Reno)**
  - Same as TCP Tahoe on Timeout: $w = 1$, slow start
  - On triple duplicate ACKs: $w = w/2$
  - Retransmit missing segment (fast retransmit)
  - Stay in Congestion Avoidance mode
- Why 3 dup-acks instead of just 1?
This is just the beginning...

Lots of congestion control schemes, with different strategies/goals:

- Tahoe (1988)
- Reno (1990)
- Vegas (1994): Detect based on RTT
- New Reno: Better recovery multiple losses
- Cubic (2006): Linux default, window size scales by cubic function
- BBR (2016): Used by Google, measures bandwidth/RTT
- ...
BBR

• Problem: can’t measure both $\text{RTT}_\text{prop}$ and Bottleneck BW at the same time

• BBR:
  – Slow start
  – Measure throughput when RTT starts to increase
  – Measure RTT when throughput is still increasing
  – Pace packets at the BDP
  – Probe by sending faster for 1RTT, then slower to compensate
BBR

From: https://labs.ripe.net/Members/gih/bbr-tcp
Help from the network

- What if routers could tell TCP that congestion is happening?
  - Congestion causes queues to grow: rate mismatch
- TCP responds to drops
- Idea: Random Early Drop (RED)
  - Rather than wait for queue to become full, drop packet with some probability that increases with queue length
  - TCP will react by reducing cwnd
  - Could also mark instead of dropping: ECN
Help from the network

• What if routers could tell TCP that congestion is happening?
  – Congestion causes queues to grow: rate mismatch

Know: TCP responds to drops

  \[ \Rightarrow \text{Routes, switches can inform TCP flows of congestion.} \]

• Idea: Random Early Drop (RED)
  – Rather than wait for queue to become full, drop packet with some probability that increases with queue length
  – TCP will react by reducing cwnd

\[ \Rightarrow \text{Any router can do it!} \]

  \[ \Rightarrow \text{But: perf impact of dropped packets.} \]
RED Advantages

- Probability of dropping a packet of a particular flow is roughly proportional to the share of the bandwidth that flow is currently getting
- Higher network utilization with low delays
- Average queue length small, but can absorb bursts

But can we do better?
What if we didn’t have to drop packets?

- Routers/switches set bits in packet to indicate congestion

\[ \text{If a router detects congestion, it sets the ECN bit in the IP header.} \]

\[ \text{When a receiver receives an ACK, it sets the TCP header's fast retransmit bit.} \]

\[ \Rightarrow \text{Requires some router support} \]

\[ \Rightarrow \text{Hosts need to actually use it.} \]
What if we didn’t have to drop packets?

- Routers/switches set bits in packet to indicate congestion
- When sender sees congestion bit, scales back cwnd
- Must be supported by both sender and receiver

=> Avoids retransmissions optionally dropped packets
Special purpose example: DCTCP (2012)?

- A datacenter is a DIFF environment
  - The global internet
    - Very low latency (<100ms)
    - But, complete control of stack!

⇒ Requires ECN at routers, use this as primary indicator of congestion.

⇒ Doesn't play well w/ others.
DNS
The story so far

Transport layer: send packets to IP:port,
eg. 128.148.10.3 port 80

Is this how users interact with the network? No!
A new abstraction

What we have: **IP Addresses**
- Numerical address appreciated by routers
- Fixed length, binary numbers
- Hierarchical, related to host’s location in the network

Examples: 128.148.32.110, 212.58.224.138

Want: **Host names**
- Mnemonics appreciated by humans
- Variable length, string characters
- Provide little (if any) information about location

Examples: google.com, www.cs.brown.edu, bbc.co.uk
Separating Naming and Addressing

cs.brown.edu => 128.148.32.110

\[ \text{makes sense to human users.} \]
\[ \text{CS Department @ Brown} \]
Separating Naming and Addressing

\[ \text{cs.brown.edu} \Rightarrow 128.148.32.110 \]

Why?
- Names are easier to remember
- Addresses can change underneath
  - e.g., renumbering when changing providers
- Useful Multiplexing/sharing
  - One name -> multiple addresses
  - Multiple names -> one address

Can adjust IP without changing how end user interacts with service!
Another Change in Layers...

- Remember ARP
  - ARP: maps IP addresses to MAC addresses
Scalable (Address <-> Name) Mappings

Original way: one file: `hosts.txt`
- Flat namespace
- Central administrator kept master copy (for the Internet)
- To add a host, emailed admin
- Downloaded file regularly
<table>
<thead>
<tr>
<th>HOST NAME</th>
<th>HOST ADDRESS</th>
<th>SPONSOR</th>
<th>LIAISON</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>10.2.0.54</td>
<td>VDH ARPA</td>
<td>Lockwood, Gregory (LOCKWOOD@BBBNIC)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Associated Computer Consultants</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>414 East Cota Street</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Santa Barbara, California 93101</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(805) 965-1023</td>
</tr>
<tr>
<td>CPUtype: PDP-11/70 (UNIX)</td>
<td>10.2.0.35</td>
<td>ARPA McBride, William T. (MCBRIDE@USC-ISI)</td>
<td>Naval Ocean Systems Center</td>
</tr>
<tr>
<td>ACCAT-TIP</td>
<td></td>
<td></td>
<td>Code 8321</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>271 Catalina Boulevard</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>San Diego, California 92152</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(714) 225-2883 (AV) 953-2083</td>
</tr>
<tr>
<td>CPUtype: H-316</td>
<td>AEROSPACE</td>
<td>10.2.0.65</td>
<td>AFSC Nelson, Louis C. (LOUSAEROSPACE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Aerospace Corporation A2/1013</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P.O. Box 92957</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Los Angeles, California 90009</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(213) 615-4424</td>
</tr>
<tr>
<td>CPUtype: VAX-11/780 (UNIX)</td>
<td>AFGL</td>
<td>10.1.0.66</td>
<td>AFSC Cosentino, Antonio (COSENTINO@AFSC-HQ)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Air Force Geophysics Laboratory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUNA Mail Stop 30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hanscom Air Force Base,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Massachusetts 01731</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(617) 861-4161 (AV) 478-4161</td>
</tr>
<tr>
<td>CPUtype: PDP-11/50 (RISX1M)</td>
<td>AFGL-TAC</td>
<td>10.2.0.66</td>
<td>AFSC Cosentino, Antonio (COSENTINO@AFSC-HQ)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Air Force Geophysics Laboratory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUNA Mail Stop 30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hanscom Air Force Base,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Massachusetts 01731</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(617) 861-4161 (AV) 478-4161</td>
</tr>
<tr>
<td>CPUtype: C/30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subject: Hostname table, 10-June-82
Scalable (Address <-> Name) Mappings

Original way: one file: `hosts.txt`
- Flat namespace
- Central administrator kept master copy (for the Internet)
- To add a host, emailed admin
- Downloaded file regularly

Is this feasible today? Lol no.
Domain Name System (DNS)

- Originally proposed by RFC882, RFC883 (1983)
- Distributed key-value store, before it was cool

- Distributed protocol to translate hostnames -> IP addresses
  - Human-readable names
  - Load-balancing/content delivery
  - So much more…
Goals for DNS

• Scalability
  – Must handle a huge number of records
    • With some software synthesizing names on the fly
  – Must sustain update and lookup load
Goals for DNS

- **Scalability**
  - Must handle a huge number of records
    - With some software synthesizing names on the fly
  - Must sustain update and lookup load

- **Distributed Control**
  - Let people control their own names
Goals for DNS

• Scalability
  – Must handle a huge number of records
    • With some software synthesizing names on the fly
  – Must sustain update and lookup load

• Distributed Control
  – Let people control their own names

• Fault Tolerance
  – Minimize lookup failures in face of other network problems
The good news

- Properties that make these goals easier to achieve
  1. Read-mostly database
     Lookups MUCH more frequent than updates
  2. Loose consistency
     When adding a machine, not end of the world if it takes minutes or hours to propagate

→ LOT OF CACHING, OR IF UPDATES TAKE MINUTED OR LONGER.
The good news

• Properties that make these goals easier to achieve
  1. Read-mostly database
     Lookups MUCH more frequent than updates
  2. Loose consistency
     When adding a machine, not end of the world if it takes minutes or hours to propagate

• These suggest aggressive caching
  – Once you’ve lookup up a hostname, remember
  – Don’t have to look again in the near future
How it works

Hierarchical namespace broken into zones

- Names can be arbitrarily long
- Hierarchy of names separated by dots

cslab1a.cs.brown.edu

From there, can allocate more names and more hierarchy

Root
  Top-level domain (TLD)
  Second-level domain (service or org)