Discussion 5:

A2 Help, More TCP Questions

By Joey Buiteweg

(some slide idea credits to Yiwen Zhang)

Discussion Outline

By the end of this discussion we will:

- Better understand what is required in A2, and how to test it
- Answer some more sample questions about TCP

A2 Hints and Help

What You Should Test (i.e what we will test)

Part 1: Proxy by itself, one Web Server, no DNS

- Log format is correct
- Bitrate adaptation is working
- Initial bitrate starts from lowest option
- Final bitrate stabilizes to correct values based on network conditions
- Outputting reasonable number of log lines
- Multiple browsers supported

What You Should Test (i.e what we will test)

Part 2: DNS server by itself

- Round Robin behavior correct, run many times to ensure this
- Geographical, test different topologies even circles
- For Geo, no need to test more than 20 Nodes, 50 links

Your mininet topology can be a simple wheel-spoke for testing

Geographical, only the text file matters.

This portion can also be done in parallel with part 1 (proxy)

Don’t worry about network conditions here, they don’t really affect behavior

What You Should Test (i.e what we will test)

Part 3: Both parts together

- Verify the proxy works with both RR and GEO DNS settings
- Verify multiple browsers can talk to the proxy (i)
- Verify multiple proxy servers can talk to a single DNS (ii)
- Case (i) and case (ii) won’t be tested at the same time on the AG
- Bitrate adaptation should still work
- AG will test with real DNS topologies represented in text file, but this isn’t necessary for your testing

Mininet Tips

Mininet has a Python API, it’s how we setup topologies

Example wheel-spoke topology for testing purposes

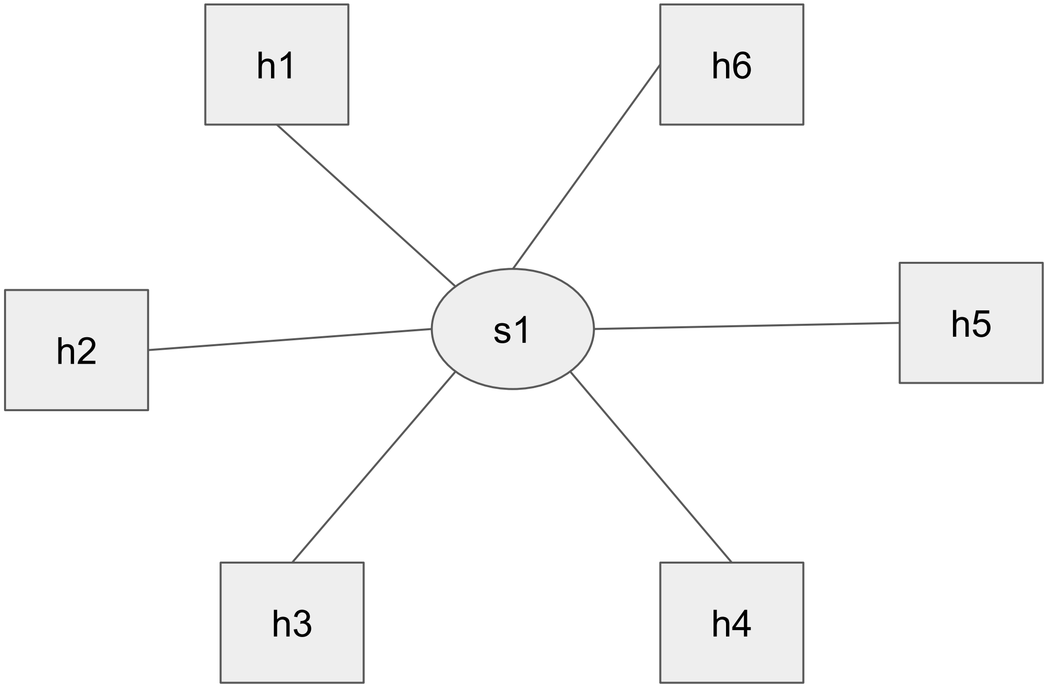
```
# wheelspoke_topo.py
# necessary imports here, see assignment1_topology.py
class WheelSpokeTopo(Topo):

    def __init__(self, **opts):
        Topo.__init__(self, **opts)
        h1 = self.addHost(‘h1’)
        # add more hosts here here

        self.addLink(h1, s1, bw=opts[‘bw’], delay=opts[‘delay’]
        # for rest of hosts, add link to s1

        # Notice we only really need one switch
```

Wheel-Spoke Topology



Mininet Tips

Mininet allows us to run commands on hosts

Lets us automate starting different processes, very useful for A2

```
# get host from topology, can also use net['h1']
h1 = net.get('h1')
h1_cmd = "echo hello from h1"

# Two options for running command on host
# Option 1: host.cmd function, wrapper around host.popen
output = h1.cmd(h1_cmd) # output == "hello from h1"

import subprocess
# Option 2: host.popen, more fine grained control, very useful!
h1_proc = h1.popen(h1_cmd, stdout=subprocess.PIPE,
                   stderr=subprocess.STDOUT)
h1_done = h1_proc.poll() # Can see if process is done or not
h1_stdout, h1_stderr = h1_proc.communicate() # h1_stderr may be None, see
docs
# Note: same interface that's provided by subprocess.Popen(), in this case
we # are running on specific mininet host!
```

Mininet Tips

```
# proxy_simple.py, host.cmd demo
from wheelspoke_topo import WheelSpokeTopo

if __name__ == '__main__':
    setLogLevel('info')
    # Create data network
    topo = WheelSpokeTopo(bw=1, delay="3ms")
    net = Mininet(topo=topo, link=TCLink, autoSetMacs=True,
                  autoStaticArp=True)
    # Run network
    net.start()

    h1 = net.get('h1') # h1 is firefox client, can also be net['h1']
    h1_cmd = "python launch_firefox.py 1 10.0.0.2:8888/index.html &"

    h2 = net.get('h2') # h2 is proxy, can also be net['h2']
    h2_cmd = "../miProxy/miProxy [flags here] > h2-out.txt &"

    h3 = net.get('h3')
    h3_cmd = "python start_server.py 3" # h3 is the web server
    # ---------------------------------------------------->
```

```
# Collect outputs
print h3.cmd(h3_cmd)
print h2.cmd(h2_cmd)
print h1.cmd(h1_cmd)

# CLI to interact w/net
# optional
CLI(net)

net.stop()
```

Documentation Links

Mininet Python Wiki: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet

https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#running-programs-in-hosts

Python (2.7) subprocess : https://docs.python.org/2/library/subprocess.html#module-subprocess

Quick Demo of A2 in Action

Lecture Based Questions

Q1 TCP File Transfer 1

Consider transferring an enormous file of ***L*** bytes from host A to host B

What is the maximum value of L such that we don't run out of TCP sequence numbers? Note: TCP sequence number is 4 bytes in the header 2^32

Given ***L = 2^32 bytes,*** find how long it takes to transmit the file

Assume:

- **MSS** (max TCP segment size), of **536 bytes**
- **66 bytes of header data** for each segment for all layers
- **155 Mbps link from A to B**
- Ignore flow and congestion control, assume A sends as fast as possible contiguously

$$\frac{(2^{32} + \frac{2^{32}}{536} * 66)}{(155 * \frac{10^6}{8})}$$

Total bytes to send (data + headers)

Transmission speed in bytes/second

Q2 TCP Segment Metadata

Host A (sender) and B (receiver) are communicating over a TCP connection

Assume the following events happen in order:

- B has **received** the first **127 bytes** of the flow from A, this consumes seq num 0-126
- A then sends two segments, **S1 (80 bytes of data)**, **S2 (40 bytes of data)**
- S1 has **sequence num 127, source port 302, destination port 80**
- S1 and S2 arrive in order
- B sends ACK1 and ACK2 to A when it receives S1 and S2, respectively

For S2, what are the sequence num, source port and destination port?

Seq num: 207, Source port: 302, Destination port: 80

For ACK1, what are the ack num, source port and destination port?

Ack num: 207, Source port: 80, Destination port: 302

Now assume S1 and S2 come out of order

For ACK1, what are the ack num, source port and destination port?

Ack num: 127, Source port: 80, Destination port: 302

Q3 TCP CWND

Consider sending a large file over a lossless TCP connection
Assume:

- TCP uses AIMD for congestion control without slow start
- CWND increases by 1 MSS for every batch of ACKs received
- Approximately constant RTT
- CWND starts at 5 MSS

How long does it take for CWND to increase from 6 MSS to 12 MSS? 6 RTT

What is the average throughput (in terms of MSS and RTT) through time = 6 RTT

$$\frac{5+6+7+8+9+10}{6} = 7.5\frac{MSS}{RTT}$$

Q4 TCP File Transfer 2

Consider sending a large file over a single TCP Reno connection

Assume:

- Link speed of 10 Mbps, no buffering/queueing (an ideal switch)
- RWND >> CWND
- Normal TCP segment/data size is 1460 bytes, but assume 1500 bytes here
- RTT of 300 milliseconds
- Connection starts with congestion avoidance (not slow start)

What is the max window size (in segments) this TCP connection can achieve?

$$\frac{10 * 10^6 * .3}{8 * 1500} = \frac{BDP}{8 * MSS} = 250$$