CS 179i: Project in Computer Science (Networks)

Jiasi Chen
Lectures: 3:10-4pm Watkins 2240
TA: Shahryar Afzal
Lab: Tuesday 4:10-7pm WCH 133

http://www.cs.ucr.edu/~jiasi/cs179i_winter18/
Why Networks?

Supports the applications that we use today...

Social media

Video streaming

Number of Internet users

- 97% of Americans between 18-29
- 40% of the world population → scope for more users

Why Networks?
But also a source of conflict.

Cyber security

Network neutrality

http://www.huffingtonpost.com/eric-dezenhall/a-look-back-at-the-target_b_7000816.html
What is networking?

• Bunch of acronyms?
What is networking?

• Bunch of headers?

Source: https://nmap.org/book/tcpip-ref.html
Networking is...

The search for general principles to guide communication
Major Areas in Networking

- **Wireless**
  - How to provide a one-to-one communication pipe in an inherently broadcast environment?

- **Layering**
  - How to modularize the design to enable easy innovation?

- **Protocols**
  - How to interact within each layer, and talk to other layers?

- **Resource allocation**
  - How to share limited resources between competing users?

---

OSI 5-layer model of the Internet

<table>
<thead>
<tr>
<th>Layer</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>(e.g. video streaming)</td>
</tr>
<tr>
<td>Transport</td>
<td>(e.g. TCP, UDP)</td>
</tr>
<tr>
<td>Network</td>
<td>(e.g. routing)</td>
</tr>
<tr>
<td>Link</td>
<td>(e.g. scheduling)</td>
</tr>
<tr>
<td>Physical</td>
<td>(e.g. OFDM)</td>
</tr>
</tbody>
</table>
Download Booster Using Multiple Interfaces

- Speed up downloads by using multiple interfaces simultaneously (e.g., WiFi, 4G, Ethernet)
- Samsung introduced Download Booster, but it got blocked by major carriers
- Multipath-TCP is another major standardization effort
Transport layer multipath: multipath-TCP

- Extension of TCP to split a single flow into multiple subflows
- Each subflow can use a different interface

- Pros: works for TCP traffic
- Cons: kernel modification

- Control knobs
  - Congestion control
  - Scheduler

- Resources
Application layer: HTTP proxy on client

- Construct a local HTTP proxy that listens on port 8080
- Split GET requests
  - eth0: request bytes 1-50
  - wlan0: request bytes 51-100
- Pros: easy to run
- Cons: only works for HTTP traffic

- Control knobs
  - Split 50/50? 25/75? Depends on link bandwidth
  - What if link bandwidth changes over time?

- Resources
  - Simple Python local proxy will be provided
Competition

- Throughput (how fast can I download?)
- Fairness (how well do I share the link with others?)
- What if throughput changes over time?

Example run:

Metrics

- Throughput (how fast can I download?)
- Fairness (how well do I share the link with others?)
- What if throughput changes over time?
Your Tasks

1. Install and get familiar with Mininet (small assignment)
2. Install multipath-TCP as a baseline
   • Experiment with different congestion control and schedulers
3. Implement the HTTP proxy
   • Design an algorithms to splitting the traffic
4. Develop a GUI to visualize the results
5. Final demos: head-to-head comparison with your classmates
6. Bonus: Run the proxy in real life (e.g., WiFi + Ethernet)
What You Will Learn in this Course

• **Knowledge:** Common networking tools/protocols, depending on your choice of project
  - Software-defined networking
  - Multipath
  - Socket programming

• **Skills**
  - How to work in teams
  - How to lead your own project
  - How to learn on your own
Logistics

• Lecture: Jiasi Chen
  • Slides available on course website
  • Office hours: Thursdays 1-3pm, or by appointment

• Lab: Shahryar Afzal

• Submit assignments on iLearn

• Check class website for latest updates
  • http://www.cs.ucr.edu/~jiasi/cs179i_winter18/
Grading

• Project: 65% total
  • Mininet assignment: 5%
  • Project proposal: 5%
  • Progress update: 10%
  • Final report: 30%
  • Final presentation: 15%

• 4 essays: 20%
  • ABET requirement
  • 2 free late days

• Participation: 15%
  • Attending lecture and lab
  • Giving feedback during other teams’ final presentations
## Calendar

<table>
<thead>
<tr>
<th>Week</th>
<th>Lecture</th>
<th>Assignment Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>MPTCP</td>
<td>Group formation</td>
</tr>
<tr>
<td>3</td>
<td>Proxy</td>
<td>Mininet mini-assignment</td>
</tr>
<tr>
<td>4</td>
<td>Visualization</td>
<td>New trends essay</td>
</tr>
<tr>
<td>5</td>
<td>Progress update / Q&amp;A</td>
<td>Brief (10 minute) presentation per group</td>
</tr>
<tr>
<td>6</td>
<td>Ethics</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Guest lecture</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>TBD</td>
<td>Ethics essay</td>
</tr>
<tr>
<td>9</td>
<td>Final presentations</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Final presentations</td>
<td>Presentation essay</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Finals week</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Teamwork essay, final report due</td>
</tr>
</tbody>
</table>
To do

• Next lecture: Mininet

• To do by next class
  • Form groups (2+) and send one email per group to myself and TA

• Questions?
## Platforms for Network/Systems Teaching

<table>
<thead>
<tr>
<th>Platform</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware Testbed</strong></td>
<td>fast accurate: &quot;ground truth&quot;</td>
<td>expensive shared resource? hard to reconfigure hard to change hard to download</td>
</tr>
<tr>
<td><strong>Simulator</strong></td>
<td>inexpensive, flexible detailed (or abstract!) easy to download virtual time (can be &quot;faster&quot; than reality)</td>
<td>may require app changes might not run OS code detail ! = accuracy may not be &quot;believable&quot; may be slow/non-interactive</td>
</tr>
<tr>
<td><strong>Emulator</strong></td>
<td>inexpensive, flexible real code reasonably accurate easy to download fast/interactive usage</td>
<td>slower than hardware experiments may not fit possible inaccuracy from multiplexing</td>
</tr>
</tbody>
</table>
To start with, a Very Simple Network
Very Simple Network using Full System Virtualization

Host VM

- cupsd
- bash
- init
- Linux Kernel
- eth0
- tap0

10.0.0.1

VM Server

openvswitch kernel module

Host VM

- cupsd
- bash
- init
- Linux Kernel
- eth0
- tap1

10.0.0.2

ovs-vswitchd

firefox

httpd

Linux Kernel

bash

init

tap0

tap1
Very Simple Network using Lightweight Virtualization

Server (or VM!)

Network Namespace 1
- firefox
- eth0
- veth1

Network Namespace 2
- httpd
- eth0
- veth2
- ovs-vswitchd

Linux Kernel
- openvswitch kernel module

10.0.0.1
10.0.0.2

10.0.0.1
10.0.0.2
Mechanism: Network Namespaces and Virtual Ethernet Pairs

```
Network Namespace 1
  firefox
  eth0
  veth1
  10.0.0.1

Network Namespace 2
  httpd
  eth0
  veth2
  10.0.0.2

Root Namespace
  Software Switch
  virtual Ethernet pairs
```
Creating it with Linux

```bash
sudo bash
# Create host namespaces
ip netns add h1
ip netns add h2

# Create switch
ovs-vswitch add-br s1

# Create links
ip link add h1-eth0 type veth peer name s1-eth1
ip link add h2-eth0 type veth peer name s1-eth2
ip link show

# Move host ports into namespaces
ip link set h1-eth0 netns h1
ip link set h2-eth0 netns h2
ip netns exec h1 ip link show
ip netns exec h2 ip link show

# Connect switch ports to OVS
ovs-vswitch add-port s1 s1-eth1
ovs-vswitch add-port s1 s1-eth2
ovs-vswitch show

# Set up OpenFlow controller
ovs-vswitch set-controller s1 tcp:127.0.0.1
oxv-controller ptcp: &
ovs-vswitch show

# Configure network
ip netns exec h1 ifconfig h1-eth0 10.1
ip netns exec h1 ifconfig lo up
ip netns exec h2 ifconfig h2-eth0 10.2
ip netns exec h1 ifconfig lo up
ifconfig s1-eth1 up
ifconfig s1-eth2 up

# Test network
ip netns exec h1 ping -c 1 10.2
```
Wouldn’t it be great if...

● We had a simple command-line tool and/or API that did this for us automatically?

● It allowed us to easily create topologies of varying size, up to hundreds of nodes, and run tests on them?

● It was already included in Ubuntu?
Mininet
An Instant Virtual Network on your Laptop (or other PC)

Mininet creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native), in seconds, with a single command:

```
> sudo mn
```

Because you can easily interact with your network using the Mininet CLI (and API), customize it, share it with others, or deploy it on real hardware, Mininet is useful for development, teaching, and research.

Mininet is also a great way to develop, share, and experiment with OpenFlow and Software-Defined Networking systems.

Mininet is actively developed and supported, and is released under a permissive BSD Open Source license. We encourage you to contribute code, bug reports/fixes, documentation, and anything else that can improve the system!

**Get Started**
Download a Mininet VM, do the walkthrough and run the OpenFlow tutorial.

**Support**
Read the FAQ, read the documentation, and join our mailing list, mininet-discuss.

**Contribute**
File a bug, download the source, or submit a pull request - all on GitHub.
Mininet command line tool and CLI demo

# mn
# mn --topo tree,depth=3,fanout=3 --link=tc,bw=10
mininet> xterm h1 h2
h1# wireshark &
h2# python -m SimpleHTTPServer 80 &
h1# firefox &
# mn --topo linear,100
# mn --custom custom.py --topo mytopo
Mininet's Python API

Core of Mininet!! Everything is built on it.
Python >> JSON/XML/etc.
Easy and (hopefully) fun
Python is used for *orchestration*, but emulation is performed by compiled C code (Linux + switches + apps)

[api.mininet.org](http://api.mininet.org)
[docs.mininet.org](http://docs.mininet.org)
[Introduction to Mininet](http://www.mininet.org/introduction.html)
Mininet API basics

```python
net = Mininet() # net is a Mininet() object
h1 = net.addHost( 'h1' ) # h1 is a Host() object
h2 = net.addHost( 'h2' ) # h2 is a Host()
s1 = net.addSwitch( 's1' ) # s1 is a Switch() object
c0 = net.addController( 'c0' ) # c0 is a Controller()
net.addLink( h1, s1 ) # creates a Link() object
net.addLink( h2, s1 )
net.start() #net is a Mininet() object
h2.cmd( 'python -m SimpleHTTPServer 80 &' )
sleep( 2 )
h1.cmd( 'curl', h2.IP() )
CLI( net )
h2.cmd('kill %python')
net.stop()
```
Performance modeling in Mininet

```python
# Use performance-modeling link and host classes
net = Mininet(link=TCLink, host=CPULimitedHost)
# Limit link bandwidth and add delay
net.addLink(h2, s1, bw=10, delay='50ms')
# Limit CPU bandwidth
net.addHost('h1', cpu=.2)
```

![Network diagram]

- **s1**: Switch
- **h1**: Host with 20% CPU limit
- **h2**: Host with 10.0.0.2 address
- **Controller**: Central control point

Network configuration:
- s1 to h2: 10 Mbps, 50 ms delay