this lecture *could not be* better timed with respect to the 6.033 schedule
6.033 Spring 2022

Lecture #8: Introduction to Networking

Katrina’s favorite lecture

Katrina LaCurts | lacurts@mit.edu | 6.033 2022
how do modules of a system communicate if they’re on separate machines?

**Client**

**Class Browser** (on machine 1)

```python
def main():
    html = browser_load_url(URL)
    ...  
```

```python
def browser_load_url(url):
    msg = url  # could reformat
    send request
    wait for reply
    html = reply  # could reformat
    return html  
```

**Server**

**Class Server** (on machine 2)

```python
def server_load_url():
    ...
    return html
```

```python
def handle_server_load_url(url):
    wait for request
    url = request
    html = server_load_url(URL)
    reply = html
    send reply
```
how do modules of a system communicate if they’re on separate machines?
how do modules of a system communicate if they’re on separate machines?

point-to-point links: get a source to talk to a directly-connected destination
how do modules of a system communicate if they’re on separate machines?

**switches:** help forward data to destinations that are far away

switches do other things, too
how do modules of a system communicate if they’re on separate machines?

as this system grows, we need to think about how to turn this set of links into a network

communication between two directly-connected nodes
examples: ethernet, bluetooth, 802.11 (wifi)
how do modules of a system communicate if they’re on separate machines?

as this system grows, we need to think about how to turn this set of links into a network

application: the things that actually generate traffic
transport: sharing the network, reliability (or not)
network: naming, addressing, routing
link: communication between two directly-connected nodes
examples: ethernet, bluetooth, 802.11 (wifi)
1970s:

ARPAnet

**Application**

the things that actually generate traffic

**Transport**

sharing the network, reliability (or not)

**Network**

naming, addressing, routing

**Link**

communication between two directly-connected nodes

examples: ethernet, bluetooth, 802.11 (wifi)

---

**FIGURE 6.1** Drawing of September 1969
(Courtesy of Alex McKenzie)

https://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/historical.html
1970s: ARPAnet

hosts.txt

Application: the things that actually generate traffic
Transport: sharing the network, reliability (or not)
Network: naming, addressing, routing
Link: communication between two directly-connected nodes
Examples: ethernet, bluetooth, 802.11 (wifi)

https://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/historical.html
1970s:
ARPAnet

hosts.txt
distance-vector routing

ARPANET LOGICAL MAP, MARCH 1977

application the things that actually generate traffic

transport sharing the network, reliability (or not)

network naming, addressing, routing

link communication between two directly-connected nodes
examples: ethernet, bluetooth, 802.11 (wifi)

https://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/historical.html
with a **layered model**, we can swap out protocols at one layer without much (or perhaps any) change to protocols at other layers.
with a **layered model**, we can swap out protocols at one layer without much (or perhaps any) change to protocols at other layers

https://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/historical.html
with a **layered model**, we can swap out protocols at one layer without much (or perhaps any) change to protocols at other layers.

1970s: ARPAnet

1978: flexibility and layering

1980s: growth → change

1990s: growth → problems

1993: commercialization

**application**
- the things that actually generate traffic

**transport**
- sharing the network, reliability (or not)
  - examples: TCP, UDP

**network**
- naming, addressing, routing
  - examples: IP

**link**
- communication between two directly-connected nodes
  - examples: ethernet, bluetooth, 802.11 (wifi)

**TCP, UDP**
- with a layered model, we can swap out protocols at one layer without much (or perhaps any) change to protocols at other layers

**hosts.txt**
- distance-vector routing

**TCP, UDP**
- OSPF, EGP, DNS

**congestion collapse**
- policy routing

**CIDR**

https://www.vox.com/a/internet-maps
with a **layered model**, we can swap out protocols at one layer without much (or perhaps any) change to protocols at other layers.

on the Internet, we have to solve all of the “normal” networking problems (addressing, routing, transport) at massive scale, while supporting a diverse group of applications and competing economic interests.

application: the things that actually generate traffic

transport: sharing the network, reliability (or not)
examples: TCP, UDP

network: naming, addressing, routing
examples: IP

link: communication between two directly-connected nodes
examples: ethernet, bluetooth, 802.11 (wifi)