Introduction to Computer Networks

Arvind Krishnamurthy

Computer Science & Engineering

UNIVERSITY of WASHINGTON
Focus of the course
Focus of the course (2)

- Three “networking” topics:

<table>
<thead>
<tr>
<th>Distributed systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networking</td>
</tr>
<tr>
<td>Communications</td>
</tr>
</tbody>
</table>

- We’re in the middle
The Main Point

1. To learn how the Internet works »
   - What really happens when you “browse the web”?  
   - What are TCP/IP, DNS, HTTP, NAT, VPNs, 802.11 etc. anyway?

2. To learn the fundamentals of computer networks
Why learn about the Internet?

1. Curiosity »
2. Impact on our world »
3. Job prospects!
From this experimental network ... ARPANET ~1970

(a) Dec. 1969.  
(b) July 1970.  
(c) March 1971.
To this!
Internet ~2005

- An everyday institution used at work, home, and on-the-go
- Visualization contains millions of links
Question

• What do you think are the issues that one has to tackle to grow from a small network to an extremely large network?
Internet – Societal Impact

• An enabler of societal change
  – Easy access to knowledge
  – Electronic commerce
  – Personal relationships
  – Discussion without censorship
Internet – Economic impact

- An engine of economic growth
  - Advertising-sponsored search
  - “Long tail” online stores
  - Online marketplaces
  - Crowdsourcing
The Main Point (2)

1. To learn how the Internet works
2. To learn the fundamentals of computer networks
   – What hard problems must they solve?
   – What design strategies have proven valuable?
Why learn the Fundamentals?

1. Apply to all computer networks
2. Intellectual interest
3. Change / reinvention
Fundamentals – Intellectual Interest

• Example key problem: Reliability!
  – Any part of the Internet might fail
  – Messages might be corrupted
  – So how do we provide reliability?
## Fundamentals – Intellectual Interest (2)

<table>
<thead>
<tr>
<th>Key problem</th>
<th>Example solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability despite failures</td>
<td>Codes for error detection/correction (§3.2, 3.3)</td>
</tr>
<tr>
<td></td>
<td>Routing around failures (§5.2)</td>
</tr>
<tr>
<td>Network growth and evolution</td>
<td>Addressing (§5.6) and naming (§7.1)</td>
</tr>
<tr>
<td></td>
<td>Protocol layering (§1.3)</td>
</tr>
<tr>
<td>Allocation of resources like bandwidth</td>
<td>Multiple access (§4.2)</td>
</tr>
<tr>
<td></td>
<td>Congestion control (§5.3, 6.3)</td>
</tr>
<tr>
<td>Security against various threats</td>
<td>Confidentiality of messages (§8.2, 8.6)</td>
</tr>
<tr>
<td></td>
<td>Authentication of communicating parties (§8.7)</td>
</tr>
</tbody>
</table>
Fundamentals – Reinvention

• The Internet is constantly being re-invented!
  – Growth over time and technology trends drive upheavals in Internet design and usage »

• Today’s Internet is different from yesterday’s
  – And tomorrow’s will be different again
  – But the fundamentals remain the same
Fundamentals – Reinvention (2)

- At least a billion Internet hosts and growing …

Source: Internet Systems Consortium (www.isc.org)
Fundamentals – Reinvention (3)

• Examples of upheavals in the past 1-2 decades

<table>
<thead>
<tr>
<th>Growth / Tech Driver</th>
<th>Upheaval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emergence of the web</td>
<td>Content Distribution Networks</td>
</tr>
<tr>
<td>Digital songs/videos</td>
<td>Peer-to-peer file sharing</td>
</tr>
<tr>
<td>Falling cost/bit</td>
<td>Voice-over-IP calling</td>
</tr>
<tr>
<td>Many Internet hosts</td>
<td>IPv6</td>
</tr>
<tr>
<td>Wireless advances</td>
<td>Mobile devices</td>
</tr>
</tbody>
</table>
Not a Course Goal

• To learn IT job skills
  – How to configure equipment
    • e.g., Cisco certifications
  – But course material is relevant, and we use hands-on tools
Course Mechanics

• Course Administration
  – Everything you need to know will be on the course web page:
    http://www.cs.washington.edu/461/

• Teaching Assistants:
  – Qiao Zhang
  – Danyang Zhuo
Course Logistics

1. Reading
2. Projects/Homeworks: 55%
3. Mid-term/final: 45%
Introduction to Computer Networks

Uses of Networks (§1.1)
Example Uses of Networks

• Work:
  – Email, file sharing, printing, ...

• Home:
  – Movies / songs, news, calls / video / messaging, e-commerce, ...

• Mobile:
  – Calls / texts, games, videos, maps, information access ...
Example Uses of Networks

• Work:
  – Email, file sharing, printing, ...

• Home:
  – Movies / songs, news, calls / video / messaging, e-commerce, ...

• Mobile:
  – Calls / texts, games, videos, maps, information access ...

What do these uses tell us about why we build networks?
For User Communication

- From the telephone onwards:
  - VoIP (voice-over-IP)
  - Video conferencing
  - Instant messaging
  - Social networking

What is the metric we need to be optimizing for these uses?
For Resource Sharing

• Many users may access the same underlying resource
  – E.g., 3D printer, search index, machines in the cloud

→ More cost effective than dedicated resources per user
  – Even network links are shared via statistical multiplexing »
Statistical Multiplexing

- Sharing of network bandwidth between users according to the statistics of their demand
  - (Multiplexing just means sharing)
  - Useful because users are mostly idle and their traffic is bursty

- Key question:
  - How much does it help?
Statistical Multiplexing (2)

• Example: Users in an ISP network
  – Network has 100 Mbps (units of bandwidth)
  – Each user subscribes to 5 Mbps, for videos
  – But a user is active only 50% of the time …

• How many users can the ISP support?
  – With dedicated bandwidth for each user:
    – Probability all bandwidth is used:
Statistical Multiplexing (3)

- With 30 users, still unlikely (2% chance) to need more than 100 Mbps!
  - Binomial probabilities

→ Can serve more users with the same size network
  - Statistical multiplexing gain is 30/20 or 1.5X
  - But may get unlucky; users will have degraded service

![Binomial Calculator](image)
For Content Delivery

- Same content is delivered to many users
  - Videos (large), songs, apps and upgrades, web pages, ...

→ What is the metric that we want to optimize in such cases?
Content Delivery (2)

• Sending content from the source to 4 users takes $4 \times 3 = 12$ “network hops” in the example
Content Delivery (3)

- But sending content via replicas takes only 4 + 2 = 6 “network hops”
Introduction to Computer Networks

Network Components (§1.2)
Parts of a Network

- Host
- Router
- Link
- Application (app)
## Component Names

<table>
<thead>
<tr>
<th>Component</th>
<th>Function</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application, or app, user</td>
<td>Uses the network</td>
<td>Skype, iTunes, Amazon</td>
</tr>
<tr>
<td>Host, or end-system, edge device, node, source, sink</td>
<td>Supports apps</td>
<td>Laptop, mobile, desktop</td>
</tr>
<tr>
<td>Router, or switch, node, intermediate system, ...</td>
<td>Relays messages between links</td>
<td>Access point, cable/DSL modem</td>
</tr>
<tr>
<td>Link, or channel</td>
<td>Connects nodes</td>
<td>Wires, wireless</td>
</tr>
</tbody>
</table>
Types of Links

• Full-duplex
  – Bidirectional

• Half-duplex
  – Bidirectional

• Simplex
  – unidirectional
Wireless Links

• Message is broadcast
  – Received by all nodes in range
  – Not a good fit with our model
Example Networks

- WiFi (802.11)
- Enterprise / Ethernet
- ISP (Internet Service Provider)
- Cable / DSL
- Mobile phone / cellular (2G, 3G, 4G)
- Bluetooth
- Telephone
- Satellite ...
## Network names by scale

<table>
<thead>
<tr>
<th>Scale</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vicinity</td>
<td>PAN (Personal Area Network)</td>
<td>Bluetooth (e.g., headset)</td>
</tr>
<tr>
<td>Building</td>
<td>LAN (Local Area Network)</td>
<td>WiFi, Ethernet</td>
</tr>
<tr>
<td>City</td>
<td>MAN (Metropolitan Area Network)</td>
<td>Cable, DSL</td>
</tr>
<tr>
<td>Country</td>
<td>WAN (Wide Area Network)</td>
<td>Large ISP</td>
</tr>
<tr>
<td>Planet</td>
<td>The Internet (network of all networks)</td>
<td>The Internet!</td>
</tr>
</tbody>
</table>
Internetworks

• An internetwork, or internet, is what you get when you join networks together
  – Just another network

• The Internet (capital “I”) is the internet we all use
Key Interfaces

• Between (1) apps and network, and (2) network components
  – More formal treatment later on
Key Interfaces (2)

1. Network-application interfaces define how apps use the network
   - Sockets are widely used in practice
Key Interfaces (3)

2. Network-network interfaces define how nodes work together
   – **Traceroute** can peek in the network
Introduction to Computer Networks

Peeking inside the Network with Traceroute

Computer Science & Engineering
UNIVERSITY of WASHINGTON
Network Service API Hides Details

- Apps talk to other apps with no real idea of what is inside the network
  - This is good! But you may be curious
Traceroute

- Widely used command-line tool to let hosts peek inside the network
  - On all OSes (tracert on Windows)
  - Developed by Van Jacobson ~1987
  - Uses a network-network interface (IP) in ways we will explain later
Traceroute (2)

• Probes successive hops to find network path
Traceroute (3)
Using Traceroute

```
C:\Users\djw>tracert www.uw.edu
Tracing route to www.washington.edu [128.95.155.134] over a maximum of 30 hops:
1  1 ms  <1 ms  2 ms  192.168.1.1
2  8 ms  8 ms  9 ms  88.Red-80-58-67.staticIP.rima-tde.net [80.58.67.88]
3 16 ms  5 ms  11 ms 169.Red-80-58-78.staticIP.rima-tde.net [80.58.78.169]
4 12 ms 12 ms  13 ms 217.Red-80-58-87.staticIP.rima-tde.net [80.58.87.217]
5  5 ms 11 ms  6 ms  et-1-0-0-1-101-GRIBONES1.red.telefonica-wholesale.net [94.142.103.20]
51  40 ms 38 ms  38 ms 176.52.250.226
7 108 ms 106 ms 136 ms xe-6-0-2-0-grtnycpt2.red.telefonica-wholesale.net [213.140.43.9]
8 108 ms 179 ms 182 ms xe9-2-0-0-grtpapx2.red.telefonica-wholesale.net [94.142.118.178]
9 170 ms 175 ms 176 ms te-4-2.car1.SanJose2.Level3.net [4.59.0.225]
10 190 ms 186 ms 187 ms vian80.csw1.SanJose1.Level3.net [4.69.152.190]
11 185 ms 185 ms 187 ms ae-82-82.ebr2.SanJose1.Level3.net [4.69.153.25]
12 268 ms 205 ms 207 ms ae-7-7.ebr1.Seattle1.Level3.net [4.69.132.50]
13 334 ms 202 ms 195 ms ae-12-51.car2.Seattle1.Level3.net [4.69.147.132]
14 195 ms 196 ms 195 ms PACIFIC-NOR.car2.Seattle1.Level3.net [4.53.146.142]
15 197 ms 195 ms 196 ms ae0--4000.iccr-sttwa01-02.infra.pnw-gigapop.net [209.124.188.132]
16 196 ms 196 ms 195 ms ui4000.uwb-ads-01.infra.washington.edu [209.124.188.133]
17   *    *    *  Request tine out.
18 201 ms 194 ms 196 ms ae4---583.uwar-ads-1.infra.washington.edu [128.95.155.131]
19 197 ms 196 ms 195 ms www1.cac.washington.edu [128.95.155.134]
```

Trace complete.
Using Traceroute (2)

- ISP names and places are educated guesses

![Diagram of traceroute results showing hops and delays between locations such as Home, tde, Telefonica, Level3, pnw-gigapop, UW, NYC, San Jose, Seattle, and www.uw.edu (www1.cac.washington.edu).]
Traceroute to another commercial webserver

-bash-3.1$ traceroute www.nyse.com
traceroute to www.nyse.com (209.124.184.150), 30 hops max, 40 byte packets
 1 acar-hsh-01-vlan75.cac.washington.edu (128.208.2.100) 0.327 ms 0.353 ms 0.392 ms
 2 uwcr-hsh-01-vlan3904.cac.washington.edu (205.175.110.17) 0.374 ms 0.412 ms 0.443 ms
 3 uwcr-hsh-01-vlan1901.cac.washington.edu (205.175.103.5) 0.595 ms 0.628 ms 0.659 ms
 4 uwbr-ads-01-vlan1902.cac.washington.edu (205.175.103.10) 0.445 ms 0.472 ms 0.501 ms
 5 ccar1-ads-ge-0-0-0-0.pnw-gigapop.net (209.124.176.32) 0.679 ms 0.747 ms 0.775 ms
 6 a209.124.184.150.deploy.akamaitechnologies.com.184.124.209.in-addr.arpa (209.124.184.150) 0.621 ms 0.456 ms 0.419 ms

What is going on?

-bash-3.1$ nslookup www.nyse.com
Name:  a789.g.akamai.net
Address: 209.124.184.137
Introduction to Computer Networks

The Socket API
(§1.3.4, 6.1.2-6.1.4)
Network-Application Interface

- Defines how apps use the network
  - Lets apps talk to each other via hosts; hides the details of the network
Motivating Application

• Simple client-server setup
Motivating Application (2)

• Simple client-server setup
  – Client app sends a request to server app
  – Server app returns a (longer) reply

• This is the basis for many apps!
  – File transfer: send name, get file (§6.1.4)
  – Web browsing: send URL, get page
  – Echo: send message, get it back

• Let’s see how to write this app …
Socket API

• Simple abstraction to use the network
  – The network service API used to write all Internet applications
  – Part of all major OSes and languages; originally Berkeley (Unix) ~1983

• Supports two kinds of network services
  – Streams: reliably send a stream of bytes »
  – Datagrams: unreliably send separate messages. (Ignore for now.)
  – Question: when would you use streams vs. datagrams?
Socket API (2)

- Sockets let apps attach to the local network at different ports

![Diagram showing sockets and ports](image)
## Socket API (3)

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCKET</td>
<td>Create a new communication endpoint</td>
</tr>
<tr>
<td>BIND</td>
<td>Associate a local address with a socket</td>
</tr>
<tr>
<td>LISTEN</td>
<td>Announce willingness to accept connections; give queue size</td>
</tr>
<tr>
<td>ACCEPT</td>
<td>Passively establish an incoming connection</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Actively attempt to establish a connection</td>
</tr>
<tr>
<td>SEND</td>
<td>Send some data over the connection</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>Receive some data from the connection</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Release the connection</td>
</tr>
</tbody>
</table>
Using Sockets

Client (host 1)  Time  Server (host 2)
Using Sockets (2)

Client (host 1) \(\downarrow\) Time \(\uparrow\) Server (host 2)

1: socket

5: connect*

7: send

8: recv*

10: close

1: socket

2: bind

3: listen

4: accept*

6: recv*

9: send

10: close

*= call blocks
Client Program (outline)

socket()    // make socket
getaddrinfo()  // server and port name
               // www.example.com:80
connect()    // connect to server [block]
...
send()      // send request
recv()     // await reply [block]
...        // do something with data!
close()    // done, disconnect
Server Program (outline)

socket()     // make socket
getaddrinfo() // for port on this host
bind()       // associate port with socket
listen()     // prepare to accept connections
accept()     // wait for a connection [block]
...
recv()       // wait for request
...
send()       // send the reply
close()      // eventually disconnect
Sample Code

- Python client code:

- Python server non-threaded code:

- Python server threaded code:
Introduction to Computer Networks

Protocols and Layering (§1.3)
Networks Need Modularity

- The network does much for apps:
  - Make and break connections
  - Find a path through the network
  - Transfers information reliably
  - Transfers arbitrary length information
  - Send as fast as the network allows
  - Shares bandwidth among users
  - Secures information in transit
  - Lets many new hosts be added
  - ...
Networks Need Modularity

- The network does much for apps:
  - Make and break connections
  - Find a path through the network
  - Transfers information reliably
  - Transfers arbitrary length information
  - Sends as fast as the network allows
  - Shares bandwidth among users
  - Secures information in transit
  - Lets many new hosts be added
  - …

We need a form of modularity, to help manage complexity and support reuse
Protocols and Layers

• Protocols and layering is the main structuring method used to divide up network functionality
  – Each instance of a protocol talks virtually to its peer using the protocol
  – Each instance of a protocol uses only the services of the lower layer
Protocols and Layers (2)

- Protocols are horizontal, layers are vertical

Instance of protocol X

Lower layer instance (of protocol Y)

Node 1

Service provided by Protocol Y

Node 2

Peer instance

Protocol X
Protocols and Layers (3)

- Set of protocols in use is called a protocol stack
Protocols and Layers (4)

• Protocols you’ve probably heard of:
  – TCP, IP, 802.11, Ethernet, HTTP, SSL, DNS, ... and many more

• An example protocol stack
  – Used by a web browser on a host that is wirelessly connected to the Internet
Encapsulation

- **Encapsulation** is the mechanism used to effect protocol layering
  - Lower layer wraps higher layer content, adding its own information to make a new message for delivery
  - Like sending a letter in an envelope; postal service doesn’t look inside
Encapsulation (2)

HTTP
TCP
IP
802.11
Encapsulation (3)
Encapsulation (4)

- Normally draw message like this:
  - Each layer adds its own header
    
    | 802.11 | IP  | TCP | HTTP |
    |--------|-----|-----|------|
    |        |     |     |      |

  - First bits on the wire
  - Last bits

- More involved in practice
  - Trailers as well as headers, encrypt/compress contents
  - Segmentation (divide long message) and reassembly
Demultiplexing

- Incoming message must be passed to the protocols that it uses

Diagram:
- SMTP
- HTTP
- DNS
  - TCP
  - UDP
  - IP
  - ARP
  - Ethernet

??
Demultiplexing (2)

- Done with demultiplexing keys in the headers
Advantage of Layering

• Information hiding and reuse
Advantage of Layering (2)

- Using information hiding to connect different systems
Disadvantage of Layering

• What are the undesirable aspects of layering?
Introduction to Computer Networks

Reference Models (§1.4, 1.6)
Guidance

• What functionality should we implement at which layer?
  – This is a key design question
  – Reference models provide frameworks that guide us »
OSI “7 layer” Reference Model

- A principled, international standard, to connect systems
  - Influential, but not used in practice. (Woops)

<table>
<thead>
<tr>
<th>Layer</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Application</td>
</tr>
<tr>
<td>6</td>
<td>Presentation</td>
</tr>
<tr>
<td>5</td>
<td>Session</td>
</tr>
<tr>
<td>4</td>
<td>Transport</td>
</tr>
<tr>
<td>3</td>
<td>Network</td>
</tr>
<tr>
<td>2</td>
<td>Data link</td>
</tr>
<tr>
<td>1</td>
<td>Physical</td>
</tr>
</tbody>
</table>

- Provides functions needed by users
- Converts different representations
- Manages task dialogs
- Provides end-to-end delivery
- Sends packets over multiple links
- Sends frames of information
- Sends bits as signals
## Internet Reference Model

- A four layer model based on experience; omits some OSI layers and uses the IP as the network layer.

| 1. Link | – Send frames over a link |
| 2. Internet | – Send packets over multiple networks |
| 3. Transport | – Provides end-to-end data delivery |
| 4. Application | – Programs that use network service |
Internet Reference Model (2)

- With examples of common protocols in each layer

1. Link
2. Internet
3. Transport
4. Application
Internet Reference Model (3)

- IP is the “narrow waist” of the Internet
  - Supports many different links below and apps above
Layer-based Names (2)

- For devices in the network:
Layer-based Names (3)

- For devices in the network:

```
+---+---+
<table>
<thead>
<tr>
<th>App</th>
<th>App</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport</td>
<td>Transport</td>
</tr>
<tr>
<td>Network</td>
<td>Network</td>
</tr>
<tr>
<td>Link</td>
<td>Link</td>
</tr>
</tbody>
</table>
```

Proxy or middlebox or gateway

But they all look like this!
A Note About Layers

- They are guidelines, not strict
  - May have multiple protocols working together in one layer
  - May be difficult to assign a specific protocol to a layer