Lecture 8: Overview of Computer Networking

Slides adapted from those of

Jim Kurose, Keith Ross, Addison-Wesley, April 2009.

Roadmap

- what’s the Internet?
- network edge: hosts, access net
- network core: packet/circuit switching, Internet structure
- performance: loss, delay, throughput
- media distribution: UDP, TCP/IP
What’s the Internet: “nuts and bolts” view

- millions of connected computing devices: 
  - hosts = end systems
  - running network apps

- communication links
  - fiber, copper, radio, satellite
  - transmission rate = bandwidth

- routers: forward packets (chunks of data)

What’s the Internet: “nuts and bolts” view

- protocols control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, Ethernet

- Internet: “network of networks”
  - loosely hierarchical
  - public Internet versus private intranet

- Internet standards
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force
A closer look at network structure:

- **network edge**: applications and hosts

- **access networks, physical media**: wired, wireless communication links

- **network core**: interconnected routers, network of networks

The network edge:

- **end systems (hosts)**:
  - run application programs
  - e.g. Web, email
  - at “edge of network”

- **client/server model**
  - client host requests, receives service from always-on server
  - e.g. Web browser/server, email client/server

- **peer-peer model**:
  - minimal (or no) use of dedicated servers
  - e.g. Skype, BitTorrent
Access networks

Q: How to connect end systems to edge router?
- residential access nets
- institutional access networks (school, company)
- mobile access networks

Keep in mind:
- bandwidth (bits per second) of access network?
- shared or dedicated?

Home networks

Typical home network components:
- DSL or cable modem
- router/firewall/NAT
- Ethernet
- wireless access point
The Network Core

- mesh of interconnected routers
- the fundamental question: how is data transferred through net?
  - circuit switching: dedicated circuit per call: telephone net
  - packet-switching: data sent thru net in discrete “chunks”

Network Core: Circuit Switching

End-end resources reserved for “call”
- link bandwidth, switch capacity
- dedicated resources: no sharing
- circuit-like (guaranteed) performance
- call setup required
**Network Core: Circuit Switching**

Network resources (e.g., bandwidth) divided into "pieces"
- pieces allocated to calls
- resource piece *idle* if not used by owning call (*no sharing*)
- dividing link bandwidth into "pieces"
  - frequency division
  - time division

**Circuit Switching: FDM and TDM**

**FDM**
- Example: 4 users

**TDM**
Network Core: Packet Switching

Each end-end data stream divided into packets:
- User A, B packets share network resources.
- Each packet uses full link bandwidth.
- Resources used as needed.

Resource contention:
- Aggregate resource demand can exceed amount available.
- Congestion: packets queue, wait for link use.
- Store and forward: packets move one hop at a time.
  - Node receives complete packet before forwarding.

Bandwidth division into "pieces":
- Dedicated allocation
- Resource reservation

Packet Switching: Statistical Multiplexing

Sequence of A & B packets does not have fixed pattern, bandwidth shared on demand ➔ statistical multiplexing.

TDM: each host gets same slot in revolving TDM frame.
Packet-switching: store-and-forward

- Takes $\frac{L}{R}$ seconds to transmit (push out) packet of $L$ bits on to link at $R$ bps
- **Store and forward**: entire packet must arrive at router before it can be transmitted on next link
- Delay = $\frac{3L}{R}$ (assuming zero propagation delay)

**Example:**
- $L = 7.5$ Mbits
- $R = 1.5$ Mbps
- Transmission delay = 15 sec

Packet switching versus circuit switching

*Packet switching allows more users to use network!*

- 1 Mb/s link
- Each user:
  - 100 kb/s when “active”
  - Active 10% of time
- **Circuit-switching**:
  - 10 users
- **Packet switching**:
  - With 35 users, probability > 10 active at same time is less than .0004

Q: How did we get value .0004?
Packet switching versus circuit switching

Is packet switching a “slam dunk winner?”

- great for bursty data
  - resource sharing
  - simpler, no call setup
- excessive congestion: packet delay and loss
  - protocols needed for reliable data transfer, congestion control
- Q: How to provide circuit-like behavior?
  - bandwidth guarantees needed for audio/video apps
  - still an unsolved problem (chapter 7)

Internet structure: network of networks

- a packet passes through many networks!
Roadmap

- what’s the Internet?
- network edge; hosts, access net
- network core: packet/circuit switching, Internet structure
- performance: loss, delay, throughput in packet switching networks
- TCP/IP

How do loss and delay occur?

packets queue in router buffers

- packet arrival rate to link exceeds output link capacity
- packets queue, wait for turn

![Diagram showing packet transmission and queueing delays](image)
Four sources of packet delay

1. nodal processing:
   - check bit errors
   - determine output link

2. queueing
   - time waiting at output link for transmission
   - depends on congestion level of router

3. Transmission delay:
   - \( R = \text{link bandwidth (bps)} \)
   - \( L = \text{packet length (bits)} \)
   - time to send bits into link = \( \frac{L}{R} \)

4. Propagation delay:
   - \( d = \text{length of physical link} \)
   - \( s = \text{propagation speed in medium (~} \times 10^8 \text{ m/sec)} \)
   - propagation delay = \( \frac{d}{s} \)

Delay in packet-switched networks

Note: \( s \) and \( R \) are very different quantities!
Nodal delay

\[ d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}} \]

- \( d_{\text{proc}} \) = processing delay
  - typically a few microsecs or less
- \( d_{\text{queue}} \) = queuing delay
  - depends on congestion
- \( d_{\text{trans}} \) = transmission delay
  - \( \approx \) L/R, significant for low-speed links
- \( d_{\text{prop}} \) = propagation delay
  - a few microsecs to hundreds of msecs

Queueing delay (revisited)

- \( R \) = link bandwidth (bps)
- \( L \) = packet length (bits)
- \( a \) = average packet arrival rate

traffic intensity = \( La/R \)

- \( La/R \approx 0 \): average queueing delay small
- \( La/R \to 1 \): delays become large
- \( La/R > 1 \): more "work" arriving than can be serviced, average delay infinite!
“Real” Internet delays and routes

- What do “real” Internet delay & loss look like?
- **Traceroute program**: provides delay measurement from source to router along end-end Internet path towards destination. For all i:
  - sends three packets that will reach router i on path towards destination
  - router i will return packets to sender
  - sender times interval between transmission and reply.

```
traceroute: gaia.cs.umass.edu to www.eurecom.fr
```

<table>
<thead>
<tr>
<th>Router</th>
<th>Delay 1</th>
<th>Delay 2</th>
<th>Delay 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs-gw (128.119.240.254)</td>
<td>1 ms</td>
<td>1 ms</td>
<td>2 ms</td>
</tr>
<tr>
<td>border1-ft-fa5-1-0.gw.umass.edu (128.119.3.145)</td>
<td>1 ms</td>
<td>2 ms</td>
<td></td>
</tr>
<tr>
<td>cs-tvbns.gw.umass.edu (128.119.3.130)</td>
<td>6 ms</td>
<td>5 ms</td>
<td>5 ms</td>
</tr>
<tr>
<td>jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)</td>
<td>16 ms</td>
<td>11 ms</td>
<td>13 ms</td>
</tr>
<tr>
<td>jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)</td>
<td>21 ms</td>
<td>18 ms</td>
<td>18 ms</td>
</tr>
<tr>
<td>abilene-vbns.abilene.ucaid.edu (198.32.11.9)</td>
<td>22 ms</td>
<td>18 ms</td>
<td>22 ms</td>
</tr>
<tr>
<td>nycm-wash.abilene.ucaid.edu (198.32.8.46)</td>
<td>22 ms</td>
<td>22 ms</td>
<td>22 ms</td>
</tr>
<tr>
<td>62.40.103.253</td>
<td>104 ms</td>
<td>109 ms</td>
<td>106 ms</td>
</tr>
<tr>
<td>62.40.96.129</td>
<td>109 ms</td>
<td>102 ms</td>
<td>104 ms</td>
</tr>
<tr>
<td>62.40.96.50</td>
<td>113 ms</td>
<td>121 ms</td>
<td>114 ms</td>
</tr>
<tr>
<td>62.40.103.54</td>
<td>112 ms</td>
<td>114 ms</td>
<td>112 ms</td>
</tr>
<tr>
<td>193.51.206.13</td>
<td>111 ms</td>
<td>114 ms</td>
<td>116 ms</td>
</tr>
<tr>
<td>195.220.98.102</td>
<td>123 ms</td>
<td>125 ms</td>
<td>124 ms</td>
</tr>
<tr>
<td>195.220.98.110</td>
<td>126 ms</td>
<td>126 ms</td>
<td>124 ms</td>
</tr>
<tr>
<td>193.48.50.54</td>
<td>135 ms</td>
<td>128 ms</td>
<td>133 ms</td>
</tr>
<tr>
<td>194.214.211.25</td>
<td>126 ms</td>
<td>128 ms</td>
<td>126 ms</td>
</tr>
<tr>
<td>fantasia.eurecom.fr (193.55.113.142)</td>
<td>132 ms</td>
<td>128 ms</td>
<td>136 ms</td>
</tr>
</tbody>
</table>

* means no response (probe lost, router not replying)
Packet loss

- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all

Throughput

- **throughput**: rate (bits/time unit) at which bits transferred between sender/receiver
  - *instantaneous*: rate at given point in time
  - *average*: rate over longer period of time
Throughput (more)

- $R_s < R_c$ What is average end-end throughput?

- $R_s > R_c$ What is average end-end throughput?

![Diagram](image)

**bottleneck link**

Link on end-end path that constrains end-end throughput

Throughput: Internet scenario

- per-connection end-end throughput: $\min(R_c, R_s, R/10)$
- In practice: $R_c$ or $R_s$ is often bottleneck

![Diagram](image)

10 connections (fairly) share backbone bottleneck link $R$ bits/sec
Internet protocol stack

- **application**: supporting network applications
  - FTP, SMTP, HTTP
- **transport**: process-process data transfer
  - TCP, UDP
- **network**: routing of datagrams from source to destination
  - IP, routing protocols
- **link**: data transfer between neighboring network elements
  - PPP, Ethernet
- **physical**: bits “on the wire”

Encapsulation
Summary

Covered a “ton” of material!
- Internet overview
- network edge, core, access network
  - packet-switching versus circuit-switching
  - Internet structure
- performance: loss, delay, throughput
- layering, service models

Creating a network app

write programs that
- run on (different) end systems
- communicate over network
- e.g., web server software communicates with browser software

No need to write software for network-core devices
- Network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation
Application architectures

- Client-server
  - Including data centers / cloud computing
- Peer-to-peer (P2P)
- Hybrid of client-server and P2P

Client-server architecture

server:
- always-on host
- permanent IP address
- server farms for scaling

clients:
- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other
Google Data Centers

- Estimated cost of data center: $600M
- Google spent $2.4B in 2007 on new data centers

Pure P2P architecture

- no always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

Highly scalable but difficult to manage
Hybrid of client-server and P2P

Skype
- voice-over-IP P2P application
- centralized server: finding address of remote party:
- client-client connection: direct (not through server)

Instant messaging
- chatting between two users is P2P
- centralized service: client presence detection/ location
  - user registers its IP address with central server when it comes online
  - user contacts central server to find IP addresses of buddies

App-layer protocol defines
- Types of messages exchanged, e.g., request, response
- Message syntax: what fields in messages & how fields are delineated
- Message semantics: meaning of information in fields
- Rules for when and how processes send & respond to messages

Public-domain protocols:
- defined in RFCs
- allows for interoperability
  - e.g., HTTP, SMTP, BitTorrent

Proprietary protocols:
- e.g., Skype, ppstream
## What transport service does an app need?

### Data loss
- Some apps (e.g., audio) can tolerate some loss
- Other apps (e.g., file transfer, telnet) require 100% reliable data transfer

### Timing
- Some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

### Throughput
- Some apps (e.g., multimedia) require minimum amount of throughput to be “effective”
- Other apps (“elastic apps”) make use of whatever throughput they get

### Security
- Encryption, data integrity, …

## Transport service requirements of common apps

<table>
<thead>
<tr>
<th>Application</th>
<th>Data loss</th>
<th>Throughput</th>
<th>Time Sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>file transfer</td>
<td>no loss</td>
<td>elastic</td>
<td>no</td>
</tr>
<tr>
<td>e-mail</td>
<td>no loss</td>
<td>elastic</td>
<td>no</td>
</tr>
<tr>
<td>Web documents</td>
<td>no loss</td>
<td>elastic</td>
<td>no</td>
</tr>
<tr>
<td>Real-time audio/video</td>
<td>loss-tolerant</td>
<td>audio: 5kbps-1Mbps video:10kbps-5Mbps</td>
<td>yes, 100’s msec</td>
</tr>
<tr>
<td>Stored audio/video</td>
<td>loss-tolerant</td>
<td>same as above</td>
<td>yes, few secs</td>
</tr>
<tr>
<td>Interactive games</td>
<td>loss-tolerant</td>
<td>few kbps up</td>
<td>yes, 100’s msec</td>
</tr>
<tr>
<td>Instant messaging</td>
<td>no loss</td>
<td>elastic</td>
<td>yes and no</td>
</tr>
</tbody>
</table>
Internet transport protocols services

TCP service:
- connection-oriented: setup required between client and server processes
- reliable transport between sending and receiving process
- flow control: sender won’t overwhelm receiver
- congestion control: throttle sender when network overloaded
- does not provide: timing, minimum throughput guarantees, security

UDP service:
- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, throughput guarantee, or security

Q: why bother? Why is there a UDP?

Internet apps: application, transport protocols

<table>
<thead>
<tr>
<th>Application</th>
<th>Application layer protocol</th>
<th>Underlying transport protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>e-mail</td>
<td>SMTP [RFC 2821]</td>
<td>TCP</td>
</tr>
<tr>
<td>remote terminal access</td>
<td>Telnet [RFC 854]</td>
<td>TCP</td>
</tr>
<tr>
<td>Web</td>
<td>HTTP [RFC 2616]</td>
<td>TCP</td>
</tr>
<tr>
<td>file transfer</td>
<td>FTP [RFC 959]</td>
<td>TCP</td>
</tr>
<tr>
<td>streaming multimedia</td>
<td>HTTP (eg Youtube), RTP [RFC 1889]</td>
<td>TCP or UDP</td>
</tr>
<tr>
<td>Internet telephony</td>
<td>SIP, RTP, proprietary (e.g., Skype)</td>
<td>typically UDP</td>
</tr>
</tbody>
</table>
Next Lecture

- Media over IP
- Lab2 assigned later this week; due May 11th
  - Find your partner NOW (2-person group, email TA your group name + member names)
  - Build your own video compression/decompression system, use motion estimation
  - Apply different sets of packet loss & error patterns to your compressed video; observe the impact
  - Find ways to recover from errors and minimize media quality degradation
    - The top 5 groups will obtain extra 10% credit
  - Matlab or C/C++, must run on csil machines