1DT052
Computer Networks I

Summary
Chapter 1
Introduction
Chapter 1: Overview of the Internet

Our goal:
- get context, overview, “feel” of networking
- more depth, detail later in course
- approach:
  o descriptive
  o use Internet as example

Overview:
- what’s the Internet
- what’s a protocol?
- network edge
- network core
- access net, physical media
- Internet/ISP structure
- performance: loss, delay
- protocol layers, service models
- history
The network edge:

- **Q: Which is better?**

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

- **peer-peer model:**
  - minimal (or no) use of dedicated servers
  - e.g. Skype, BitTorrent, eMule
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: FDMA and TDMA

FDMA

Example:
4 users

TDMA

Introduction
Packet Switching: Statistical Multiplexing

Sequence of A & B packets does not have fixed pattern, shared on demand \( \Rightarrow \) statistical multiplexing.

TDM: each host gets same slot in revolving TDM frame.
Internet protocol stack

- application: supporting network applications
  - FTP, SMTP, STTP
- transport: host-host 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”
Protocol layering and data

Each layer takes data from above
- adds header information to create new data unit
- passes new data unit to layer below

source

```
  M
 /  \
H_t M
 /  \
H_n H_t M
 /  \
H_l H_n H_t M
```

destination

```
  M
 /  \
H_t M
 /  \
H_n H_t M
 /  \
H_l H_n H_t M
```

message
segment
datagram
frame
Chapter 2
Application Layer
Chapter 2: Application Layer

Our goals:
- conceptual, implementation aspects of network application protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm
- learn about protocols by examining popular application-level protocols
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- programming network applications
  - socket API
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
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>
Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client
DNS: Domain Name System

People: many identifiers:
- SSN, name, passport #

Internet hosts, routers:
- IP address (32 bit) - used for addressing datagrams
- "name", e.g., ww.yahoo.com - used by humans

Q: map between IP addresses and name?

Domain Name System:
- distributed database implemented in hierarchy of many name servers
- application-layer protocol host, routers, name servers to communicate to resolve names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network’s "edge"


**DNS name resolution example**

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

  **iterated query:**
  - contacted server replies with name of server to contact
  - “I don’t know this name, but ask this server”

---

2: Application Layer

17
Pure P2P architecture

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

Three topics:
- File distribution
- Searching for information
- Case Study: Skype
Chapter 3 outline

- 3.1 Transport-layer services
- 3.2 Multiplexing and demultiplexing
- 3.3 Connectionless transport: UDP
- 3.4 Principles of reliable data transfer
- 3.5 Connection-oriented transport: TCP
  - segment structure
  - reliable data transfer
  - flow control
  - connection management
- 3.6 Principles of congestion control
- 3.7 TCP congestion control
UDP: User Datagram Protocol [RFC 768]

- “no frills,” “bare bones” Internet transport protocol
- “best effort” service, UDP segments may be:
  - lost
  - delivered out of order to app
- connectionless:
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

Why is there a UDP?
- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small segment header
- no congestion control: UDP can blast away as fast as desired
Internet Checksum Example

- **Note**
  - When adding numbers, a carryout from the most significant bit needs to be added to the result

- **Example:** add two 16-bit integers

```
1 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
-----------------------------
wraparound 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1
sum 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0
checksum 1 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
```
Principles of Reliable data transfer

- important in app., transport, link layers
- top-10 list of important networking topics!

- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)
Reliable Data Transfer

(a) operation with no loss

(b) lost packet
Pipelining Protocols

Go-back-N: overview
- **sender**: up to N unACKed pkts in pipeline
- **receiver**: only sends cumulative ACKs
  - doesn’t ACK pkt if there’s a gap
- **sender**: has timer for oldest unACKed pkt
  - if timer expires: retransmit all unACKed packets

Selective Repeat: overview
- **sender**: up to N unACKed packets in pipeline
- **receiver**: ACKs individual pkts
- **sender**: maintains timer for each unACKed pkt
  - if timer expires: retransmit only unACKed packet
Go-Back-N

Sender:
- k-bit seq # in pkt header
- “window” of up to \( N \), consecutive unACKed pkts allowed

\[
\begin{array}{c}
\text{send_base} \quad \text{nextseqnum} \\
\text{window size} \quad N
\end{array}
\]

- ACK\((n)\): ACKs all pkts up to, including seq # n - “cumulative ACK”
  - may receive duplicate ACKs (see receiver)
- timer for each in-flight pkt
- \( \text{timeout}(n) \): retransmit pkt n and all higher seq # pkts in window
Selective repeat: sender, receiver windows

(a) sender view of sequence numbers

(b) receiver view of sequence numbers
TCP Flow Control

- receive side of TCP connection has a receive buffer:

  ![Diagram showing IP datagrams mapping to unused buffer space to TCP data (in buffer) to application process]

- speed-matching service: matching send rate to receiving application’s drain rate

- app process may be slow at reading from buffer

- flow control: sender won’t overflow receiver’s buffer by transmitting too much, too fast
Principles of Congestion Control

Congestion:
- informally: “too many sources sending too much data too fast for network to handle”
- different from flow control!
- manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)
- a top-10 problem!
TCP congestion control: bandwidth probing

- "probing for bandwidth": increase transmission rate on receipt of ACK, until eventually loss occurs, then decrease transmission rate
  - continue to increase on ACK, decrease on loss (since available bandwidth is changing, depending on other connections in network)

ACKs being received, so increase rate

X loss, so decrease rate

TCP’s “sawtooth” behavior
Chapter 4
Network Layer
Chapter 4: Network Layer

- 4.1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What’s inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing
Interplay between routing and forwarding

value in arriving packet’s header

<table>
<thead>
<tr>
<th>header value</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

routing algorithm

local forwarding table
Longest prefix matching

<table>
<thead>
<tr>
<th>Prefix Match</th>
<th>Link Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

Examples

DA: 11001000 00010111 00010110 10101010
Which interface?

DA: 11001000 00010111 00011000 10101010
Which interface?
IP Addressing: introduction

- **IP address:** 32-bit identifier for host, router *interface*

- **interface:** connection between host/router and physical link
  - router’s typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface

223.1.1.1 = 11011111 00000001 00000001 00000001

223.1.1.2

223.1.1.3

223.1.1.4

223.1.2.1

223.1.2.2

223.1.2.9

223.1.3.1

223.1.3.2

223.1.3.27
IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address

```
11001000  00010111  00010000  00000000

subnet part  host part
200.23.16.0/23
```
IP addresses: how to get one?

Q: How does a host get IP address?

- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config

- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - “plug-and-play”
Hierarchical addressing: more specific routes

**ISPs-R-Us has a more specific route to Organization 1**

- Organization 0
  - 200.23.16.0/23
- Organization 2
  - 200.23.20.0/23
- Organization 7
  - 200.23.30.0/23
- Organization 1
  - 200.23.18.0/23

Fly-By-Night-ISP

- “Send me anything with addresses beginning 200.23.16.0/20”

ISPs-R-Us

- “Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23”

Internet
NAT: Network Address Translation

All datagrams leaving local network have the same single source NAT IP address: 138.76.29.7, different source port numbers.

Datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual).

local network (e.g., home network) 10.0.0/24

rest of Internet

138.76.29.7

10.0.0.1

10.0.0.2

10.0.0.3

10.0.0.4
IPv6

- Initial motivation: 32-bit address space soon to be completely allocated.
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS
IPv6 datagram format:
  - fixed-length 40 byte header
  - no fragmentation allowed
A Link-State Routing Algorithm

Dijkstra’s algorithm

- net topology, link costs known to all nodes
  - accomplished via “link state broadcast”
  - all nodes have same info
- computes least cost paths from one node (‘source’) to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.’s

Notation:

- \( c(x,y) \): link cost from node \( x \) to \( y \); = \( \infty \) if not direct neighbors
- \( D(v) \): current value of cost of path from source to dest. \( v \)
- \( p(v) \): predecessor node along path from source to \( v \)
- \( N' \): set of nodes whose least cost path definitively known
Distance vector algorithm

Basic idea:
- From time-to-time, each node sends its own distance vector estimate to neighbors
- Asynchronous
- When a node \( x \) receives new DV estimate from neighbor, it updates its own DV using B-F equation:
  \[
  D_x(y) \leftarrow \min_v \{c(x, v) + D_v(y)\} \quad \text{for each node } y \in N
  \]
- Under minor, natural conditions, the estimate \( D_x(y) \) converge to the actual least cost \( d_x(y) \)
Chapter 5
Link Layer and LANs
Link Layer

- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3 Multiple access protocols
- 5.4 Link-layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches
Where is the link layer implemented?

- in each and every host
- link layer implemented in “adaptor” (aka network interface card NIC)
  - Ethernet card, PCMCI card, 802.11 card
  - implements link, physical layer
- attaches into host’s system buses
- combination of hardware, software, firmware
Parity Checking

**Single Bit Parity:**
Detect single bit errors

**Two Dimensional Bit Parity:**
Detect and correct single bit errors

```
0111000110101011 0
```

```
\[
\begin{array}{ccccccc}
\text{d}_{1,1} & \ldots & \text{d}_{1,i} & \ldots & \text{d}_{1,j} \\
\text{d}_{2,1} & \ldots & \text{d}_{2,i} & \ldots & \text{d}_{2,j} \\
\vdots & \ldots & \vdots & \ldots & \vdots \\
\text{d}_{i,1} & \ldots & \text{d}_{i,j} \\
\text{d}_{i+1,1} & \ldots & \text{d}_{i+1,j} \\
\end{array}
\]
```

```
1010111 1010111
111100 101100
011101 011101
001010 001010
\text{no errors}
```

```
1010111
111100
011101
001010
\text{parity error}
```

```
\text{correctable single bit error}
```

5: DataLink Layer
Checksumming: Cyclic Redundancy Check

- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)

\[ D \times 2^r \oplus R \]

\( D: \text{data bits to be sent} \]
\( R: \text{CRC bits} \]

bit pattern

mathematical formula
Random Access Protocols

- When node has packet to send
  - transmit at full channel data rate R.
  - no a priori coordination among nodes
- two or more transmitting nodes $\rightarrow$ “collision”,
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- Examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA
CSMA/CD (Collision Detection)

CSMA/CD: carrier sensing, deferral as in CSMA
  - collisions detected within short time
  - colliding transmissions aborted, reducing channel wastage

Collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength
MAC Addresses and ARP

- **32-bit IP address:**
  - *network-layer address*
  - Used to get datagram to destination IP subnet

- **MAC (or LAN or physical or Ethernet) address:**
  - Function: *get frame from one interface to another physically-connected interface (same network)*
  - 48 bit MAC address (for most LANs)
    - Burned in NIC ROM, also sometimes software settable
ARP: Address Resolution Protocol

**Question:** how to determine MAC address of B knowing B’s IP address?

- Each IP node (host, router) on LAN has ARP table
- ARP table: IP/MAC address mappings for some LAN nodes
  - `<IP address; MAC address; TTL>`
  - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)
Chapter 8
Network Security
Chapter 8: Network Security

Chapter goals:

- understand principles of network security:
  - cryptography and its many uses beyond “confidentiality”
  - authentication
  - message integrity

- security in practice:
  - firewalls and intrusion detection systems
  - security in application, transport, network, link layers
The language of cryptography

symmetric key crypto: sender, receiver keys *identical*
public-key crypto: encryption key *public*, decryption key *secret* (private)
Public key cryptography

**symmetric key crypto**
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never “met”)?

**public key cryptography**
- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver
Public key cryptography

plaintext message, m

encryption algorithm

Bob's public key

Bob's private key

K^+(m)

ciphertext

decryption algorithm

plaintext message

m = K^-B(K^+(m))
Message Integrity

Bob receives msg from Alice, wants to ensure:
- message originally came from Alice
- message not changed since sent by Alice

Cryptographic Hash:
- takes input $m$, produces fixed length value, $H(m)$
  - e.g., as in Internet checksum
- computationally infeasible to find two different messages, $x$, $y$ such that $H(x) = H(y)$
  - equivalently: given $m = H(x)$, ($x$ unknown), can not determine $x$.
  - note: Internet checksum fails this requirement!
Digital Signatures

simple digital signature for message m:
- Bob “signs” m by encrypting with his private key $K_B$, creating “signed” message, $K_B(m)$
Certification Authorities

- Certification Authority (CA): binds public key to particular entity, E.
- E registers its public key with CA.
  - E provides “proof of identity” to CA.
  - CA creates certificate binding E to its public key.
  - Certificate containing E’s public key digitally signed by CA: CA says “This is E’s public key.”
Firewalls

A firewall isolates an organization’s internal network from the larger Internet, allowing some packets to pass, blocking others.
Intrusion detection systems

- multiple IDSs: different types of checking at different locations