Chapter 1: introduction

**our goal:**
- get "feel" and terminology
- more depth, detail later in course
- approach:
  - use Internet as example

**overview:**
- what’s the Internet?
- what’s a protocol?
- network edge: hosts, access net, physical media
- network core: packet/circuit switching, Internet structure
- performance: loss, delay, throughput
- security
- protocol layers, service models
- history

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

- **Internet:** “network of networks”
  - Interconnected ISPs
- **protocols** control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, 802.11
- **Internet standards**
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force

What’s the Internet: a service view

- **Infrastructure that provides services to applications:**
  - Web, VoIP, email, games, e-commerce, social nets, …
- **provides programming interface to apps**
  - hooks that allow sending and receiving app programs to "connect" to Internet
  - provides service options, analogous to postal service
A closer look at network structure:

- **network edge:**
  - hosts: clients and servers
  - servers often in data centers

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

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

Four sources of packet delay

- **d\(_{\text{proc}}\):** processing delay
  - check bit errors
  - determine output link
  - typically < msec

- **d\(_{\text{queue}}\):** queueing delay
  - time waiting at output link for transmission
  - depends on congestion level of router

How do loss and delay occur?

- packets queue in router buffers
  - packet arrival rate to link (temporarily) exceeds output link capacity

- packets queue, wait for turn

\[
d_{\text{total}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}
\]
**Router architecture overview**

Two key router functions:
- Run routing algorithms/protocol (RIP, OSPF, BGP)
- **Forwarding** datagrams from incoming to outgoing link

**Input port functions**

- **Routing processor**
- High-speed switching fabric
- **Routing, management control plane** (software)
- **Forwarding data plane** (hardware)
- **Routing, management control plane**
- **Forwarding data plane**

**Output ports**

- **Buffering** required when datagrams arrive from fabric faster than the transmission rate
- **Scheduling discipline** chooses among queued datagrams for transmission

**Encapsulation**

- **Message**
- **Segment**
- **Datagram**
- **Frame**
- **Source**
- **Destination**

- **Physical layer**
- **Bit-level reception**
- **Decentralized switching**
  - Given datagram dest., lookup output port using forwarding table in input port memory ("match plus action")
  - Goal: complete input port processing at 'line speed'
  - Queuing if datagrams arrive faster than forwarding rate into switch fabric

**Introduction**

- **Network Layer**
- **Physical Layer**
- **Data Link Layer**
- **Network Layer**
- **Transport Layer**
- **Application Layer**

**Datagram**

- **Interface**
- **Header**
- **Payload**

- **Switch fabric**
- **Line termination**
- **Link layer protocol** (receive)
- **Queueing**
- **Switch fabric**
- **Line termination**
- **Link layer protocol** (send)
- **Queueing**

**Network Layer**

- **Input port functions**
- **Decentralized switching**

**Introduction**

- **Transport layer**
- **Network layer**
- **Physical layer**

**Encapsulation**

- **Source**
- **Destination**
- **Physical layer**

- **Switch**
- **Router**
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
- Creating network applications
  - Socket API

Processes Communicating
- Process: Program running within a host
- Within the same host, two processes communicate using **interprocess communication** (defined by OS)
- Processes in different hosts communicate by exchanging messages
- Aside: Applications with P2P architectures have client processes & server processes

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 allow for rapid app development, propagation

Sockets
- Process sends/receives messages to/from its **socket**
  - Socket analogous to door
  - Sending process shoves message out door
  - Sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process

Aside: Applications with P2P architectures have client processes & server processes
Addressing processes

- to receive messages, process must have **identifier**
- host device has unique 32-bit IP address
- **Q**: does IP address of host on which process runs suffice for identifying the process?
  - **A**: no, many processes can be running on same host

**identifier** includes both IP address and port numbers associated with process on host.

- example port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to gaia.cs.umass.edu web server:
  - IP address: 128.119.245.12
  - port number: 80

Transport services and protocols

- provide **logical communication** between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into **segments**, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP

Transport vs. Network Layer

- **network layer**: logical communication between hosts
- **transport layer**: logical communication between processes
  - relies on, enhances, network layer services

Internet transport-layer protocols

- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup
- unreliable, unordered delivery: UDP
  - no-frills extension of ”best-effort” IP
- services not available:
  - delay guarantees
  - bandwidth guarantees
Multiplexing/demultiplexing

Multiplexing at sender:
- handle data from multiple sockets, add transport header (later used for demultiplexing)

Demultiplexing at receiver:
- use header info to deliver received segments to correct socket

TCP: Overview

- point-to-point:
  - one sender, one receiver
- reliable, in-order byte stream:
  - no "message boundaries"
- pipelined:
  - TCP congestion and flow control set window size
- full duplex data:
  - bi-directional data flow in same connection
- MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) initiates sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver
- congestion control:
  - benefits entire network and all users

TCP seq. numbers, ACKs

Connection-oriented demux: example

Three segments, all destined to IP address B, dest port: 80 are demultiplexed to different sockets
TCP congestion control: additive increase multiplicative decrease

- **approach**: sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
  - **additive increase**: increase \( cwnd \) by 1 MSS every RTT until loss detected
  - **multiplicative decrease**: cut \( cwnd \) in half after loss

AIMD saw tooth behavior: probing for bandwidth

- additively increase window size ...
- until loss occurs (then cut window in half)

**TCP: switching from slow start to CA**

Q: when should the exponential increase switch to linear?
A: when \( cwnd \) gets to 1/2 of its value before timeout.

**Implementation:**
- variable sssthresh
- on loss event, sssthresh is set to 1/2 of \( cwnd \) just before loss event

Chapter 4: network layer

**chapter goals:**
- understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - routing (path selection)
  - broadcast, multicast

**Interplay between routing and forwarding**

- routing algorithm determines end-end-path through network
- forwarding table determines local forwarding at this router

<table>
<thead>
<tr>
<th>routing algorithm</th>
<th>local forwarding table</th>
<th>header value</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0110</td>
<td>1</td>
</tr>
</tbody>
</table>

Value in arriving packet’s header: 0111
Longest prefix matching

when looking for forwarding table entry for given destination address, use longest address prefix that matches destination address.

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010*** *******</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 000111000 *******</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 00011000 10101010
which interface?
DA: 11001000 00010111 00010110 10100001
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 or two interfaces (e.g., wired Ethernet, wireless 802.11)
- **IP addresses associated with each interface**

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; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- π(v): predecessor node along path from source to v
- N*: set of nodes whose least cost path definitively known
Dijkstra’s algorithm: another example

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$u$</td>
<td>$2.u$</td>
<td>$5.u$</td>
<td>$1.u$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>1</td>
<td>$ux$</td>
<td>$2.u$</td>
<td>$4.x$</td>
<td>$2.x$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>2</td>
<td>$uxy$</td>
<td>$2.u$</td>
<td>$3.y$</td>
<td>$4.y$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$uxxy$</td>
<td>$3.y$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$uxxyw$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$4.y$</td>
</tr>
<tr>
<td>5</td>
<td>$uxxywz$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

Let $d_x(y) := \text{cost of least-cost path from } x \text{ to } y$

then

$$d_x(y) = \min_v \{c(x,v) + d_v(y) \}$$

cost from neighbor $v$ to destination $y$

$\min$ taken over all neighbors $v$ of $x$

Distance vector algorithm

- $D_s(y) = \text{estimate of least cost from } x \text{ to } y$
  - $x$ maintains distance vector $D_x = [D_s(y) : y \in N]$
- node $x$:
  - knows cost to each neighbor $v$: $c(x,v)$
  - maintains its neighbors’ distance vectors. For each neighbor $v$, $x$ maintains
    $D_v = [D_s(y) : y \in N]$

Chapter 4 Summary

4.3 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
Chapter 5: Link layer

5.1 introduction, services
5.2 error detection, correction
5.3 multiple access protocols
5.4 LANs
  - addressing, ARP
  - Ethernet
  - switches

Link layer services

- framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses used in frame headers to identify source, dest
    - different from IP address!
- reliable delivery between adjacent nodes
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - Q: why both link-level and end-end reliability?

Link layer services (more)

- flow control:
  - pacing between adjacent sending and receiving nodes

- error detection:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame

- error correction:
  - receiver identifies and corrects bit error(s) without resorting to retransmission

- half-duplex and full-duplex
  - with half duplex, nodes at both ends of link can transmit, but not at same time

MAC protocols: taxonomy

three broad classes:

- channel partitioning
  - divide channel into smaller “pieces” (time slots, frequency, code)
  - allocate piece to node for exclusive use

- random access
  - channel not divided, allow collisions
  - “recover” from collisions

- “taking turns”
  - nodes take turns, but nodes with more to send can take longer turns
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
- human analogy: the polite conversationalist

### Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits.
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame!
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters **binary (exponential) backoff**:
   - after mth collision, NIC chooses K at random from \(\{0, 1, 2, ..., 2^m-1\}\). NIC waits \(K \cdot 512\) bit times, returns to Step 2
   - longer backoff interval with more collisions

### ARP: address resolution protocol

**Question**: how to determine interface’s MAC address, knowing its IP address?

- **ARP table**: each IP node (host, router) on LAN has 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)
Switches vs. routers
both are store-and-forward:
- routers: network-layer devices (examine network-layer headers)
- switches: link-layer devices (examine link-layer headers)
both have forwarding tables:
- routers: compute tables using routing algorithms, IP addresses
- switches: learn forwarding table using flooding, learning, MAC addresses

Wireless Link Characteristics
important differences from wired link ….
- decreased signal strength: radio signal attenuates as it propagates through matter (path loss)
- interference from other sources: standardized wireless network frequencies (e.g., 2.4 GHz) shared by other devices (e.g., phone); devices (motors) interfere as well
- multipath propagation: radio signal reflects off objects ground, arriving ad destination at slightly different times
… make communication across (even a point to point) wireless link much more “difficult”

Elements of a wireless network
base station
- typically connected to wired network
- relay - responsible for sending packets between wired network and wireless host(s) in its “area”
  - e.g., cell towers, 802.11 access points

Wireless network characteristics
Multiple wireless senders and receivers create additional problems (beyond multiple access):

Hidden terminal problem
- B, A hear each other
- B, C hear each other
- A, C can not hear each other means A, C unaware of their interference at B

Signal attenuation:
- B, A hear each other
- B, C hear each other
- A, C can not hear each other interfering at B
**802.11 LAN architecture**

- Wireless host communicates with base station
  - Base station = access point (AP)
- Basic Service Set (BSS) (aka "cell") in infrastructure mode contains:
  - Wireless hosts
  - Access point (AP): base station
  - Ad hoc mode: hosts only

**Collision Avoidance: RTS-CTS exchange**

**Mobility: vocabulary**

- **Home network**: Permanent "home" of mobile (e.g., 128.119.40/24)
- **Home agent**: Entity that will perform mobility functions on behalf of mobile, when mobile is remote
- **Permanent address**: Address in home network, can always be used to reach mobile (e.g., 128.119.40.186)
- **Care-of-address**: Address in visited network (e.g., 79.129.13.2)

**Mobility: more vocabulary**

- **Permanent address**: Remains constant (e.g., 128.119.40.186)
- **Visited network**: Network in which mobile currently resides (e.g., 79.129.13/24)
- **Foreign agent**: Entity in visited network that performs mobility functions on behalf of mobile
- **Correspondent**: Wants to communicate with mobile
What is network security?

**Confidentiality:** only sender, intended receiver should "understand" message contents
- sender encrypts message
- receiver decrypts message

**Authentication:** sender, receiver want to confirm identity of each other

**Message integrity:** sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

**Access and availability:** services must be accessible and available to users

Friends and enemies: Alice, Bob, Trudy
- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages

The language of cryptography

\[ m = K_B(K_A(m)) \]

**Symmetric key cryptography**
- Bob and Alice share same (symmetric) key: K
  - e.g., key is knowing substitution pattern in mono alphabetic substitution cipher
- how do Bob and Alice agree on key value?
Public key cryptography

RSA: encryption, decryption

0. given \((n,e)\) and \((n,d)\) as computed above

1. to encrypt message \(m < n\), compute
\[ c = m^e \mod n \]

2. to decrypt received bit pattern, \(c\), compute
\[ m = c^d \mod n \]

魔力发生了！
\[ m = (m^e \mod n)^d \mod n \]

Why \(K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))\)?

follows directly from modular arithmetic:

\[(m^e \mod n)^d \mod n = m^{ed} \mod n = m^{ed} \mod n = (m^e \mod n)^d \mod n\]

The following property will be very useful later:

\[ K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m)) \]

use public key first, followed by private key

use private key first, followed by public key

result is the same!
Digital signature = signed message digest

Bob sends digitally signed message:

Alice verifies signature, integrity of digitally signed message:

Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- KDC: server shares/​knows different secret key for each registered user (many users)
  - Alice, Bob know own symmetric keys, $K_{A,KDC}$, $K_{B,KDC}$, for communicating with KDC.
  - Permanent / static existence of these ‘identity’ keys
    - KDC creates a unique, single use “session key” for each new communication between Alice and Bob

KDC Session Key Distribution

Certification authorities

- **certification authority (CA):** binds public key to particular entity, E.
- E (person, router) 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”
email: fully secure

- Alice wants to provide secrecy, sender authentication & message integrity. How?

- Alice uses three keys: her private key, Bob’s public key, the newly created symmetric key.

- What does Bob do to retrieve the msg & be sure it came from Alice?

Pretty good privacy (PGP)

- Internet e-mail encryption scheme, de-facto standard.

- Uses
  - Symmetric key cryptography
  - Public key cryptography
  - Hash function
  - Digital signature

- Provides
  - Secrecy
  - Sender authentication
  - Integrity

Basic SSL: a simple secure channel

1. **Handshake**: Alice and Bob use their certificates and private keys to authenticate each other and exchange shared secret.

2. **Key Derivation**: Alice and Bob use shared secret to derive set of keys.

3. **Data Transfer**: Data to be transferred is broken up into a series of records.

4. **Connection Closure**: Special messages to securely close connection.

(1) SSL: **Handshake**

- Bob establishes TCP connection to Alice.

- Authenticates Alice via CA signed certificate.

- Creates, encrypts (using Alice’s public key), sends master secret key to Alice.

  - nonce exchange not shown.
(3) SSL: Data transfer

TCP byte stream

\[ b_1, b_2, ..., b_n \]

block \( n \) bytes together

\[ d \]

\[ H(d) \]

\[ M_a \]

compute MAC

\[ E_B \]

encrypted, MAC, SSL seq. #

SSL record format

Type | Var | Len

unencrypted | encrypted using \( E_B \)

Summary

- Chapter 1 – General overview of internet
  - Delay introduced
- Chapter 2
  - Apps, client-server
  - Socket programming
- Chapter 3 – TCP and UDP
- Chapter 4 – Network: IP, Routing algorithms
- Chapter 5 – Link layer: Ethernet, CSMA/CD and … /CA
- Chapter 6 – Wireless and Mobility
- Chapter 8 – Security