Chapter 4
Network Layer

Slides adopted from original ones provided by the textbook authors.
Chapter 4: outline

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
Key Network-Layer Functions

- **forwarding**: move packets from router’s input to appropriate router output

- **routing**: determine route taken by packets from source to dest.

- **connection management**: for connection based network-layer protocols
Chapter 4: outline

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
Virtual circuits vs. datagram networks

- **Virtual circuits**: *connection* service
  - call setup, teardown for each call *before* data can flow
  - each packet carries VC identifier
  - link, router resources (bandwidth, buffers) *allocated* to VC

- **Datagram networks**: *connectionless* service
  - no call setup at network layer
  - routers: no state about end-to-end connections
  - packets forwarded using destination host address
Longest prefix matching

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

- Given address range, calculate prefix.
- Given prefix, calculate address range.
- Given address, find outgoing interface.
Chapter 4: outline

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
Router architecture overview

two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link
Switching fabrics

- transfer packet from input buffer to appropriate output buffer
- three types of switching fabrics
  - memory: two memory accesses per packet
  - bus: exclusive access by single sender
  - crossbar: supporting multiple transmission pairs
Chapter 4: outline

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
# IP datagram format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP protocol version</td>
<td>Number of the IP protocol version</td>
</tr>
<tr>
<td>header length (bytes)</td>
<td>Total length of the IP header (in bytes)</td>
</tr>
<tr>
<td>“type” of data</td>
<td>Type of data packet (e.g., TCP, UDP, ICMP, etc.)</td>
</tr>
<tr>
<td>max number remaining</td>
<td>The maximum number of hops the packet can be sent (decremented at each router)</td>
</tr>
<tr>
<td>upper layer protocol</td>
<td>Protocol used to deliver the payload to the upper layer</td>
</tr>
<tr>
<td>options (if any)</td>
<td>Options included in the datagram</td>
</tr>
<tr>
<td>data</td>
<td>Payload data (variable length, typically a TCP or UDP segment)</td>
</tr>
</tbody>
</table>

**Header fields:***

- **ver**: Version number
- **head. len**: Header length
- **type of service**: Type of service
- **flgs**: Flags
- **fragment offset**: Fragment offset
- **time to live**: Time to live
- **upper layer**: Upper layer
- **header checksum**: Header checksum
- **32 bit source IP address**: Source IP address
- **32 bit destination IP address**: Destination IP address
- **e.g. timestamp, record route taken, specify list of routers to visit.**
IP fragmentation, reassembly

- Network links have MTU (max.transfer size) - largest possible link-level frame
  - Different link types, different MTUs
- Large IP datagram divided ("fragmented") within net
  - One datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments

*fragmentation:*

- **In:** one large datagram
- **Out:** 3 smaller datagrams

*reassembly*
IPv4 Addressing

- **Classful addressing**
  - Fixed subnet/host address length
  - Inflexible

- **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
DHCP: Dynamic Host Configuration Protocol

**goal:** allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/“on”)
- support for mobile users who want to join network (more shortly)

**DHCP overview:**

- host broadcasts “DHCP discover” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg
NAT: network address translation

**implementation:** NAT router must:

- **outgoing datagrams: replace** (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

- **remember (in NAT translation table)** every (source IP address, port #) to (NAT IP address, new port #) translation pair

- **incoming datagrams: replace** (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table
ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer “above” IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error
- traceroute case study

<table>
<thead>
<tr>
<th>Type</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>echo reply (ping)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>dest. network unreachable</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>dest host unreachable</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>dest protocol unreachable</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>dest port unreachable</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>dest network unknown</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>dest host unknown</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>source quench (congestion control - not used)</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>echo request (ping)</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>route advertisement</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>router discovery</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>TTL expired</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>bad IP header</td>
</tr>
</tbody>
</table>
IPv6 datagram format

- **initial motivation:** 32-bit address space soon to be completely allocated.
- **additional motivation:**
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

<table>
<thead>
<tr>
<th>ver</th>
<th>pri</th>
<th>flow label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>32 bits</td>
</tr>
<tr>
<td>payload len</td>
<td>next hdr</td>
<td>hop limit</td>
</tr>
<tr>
<td>source address</td>
<td>(128 bits)</td>
<td></td>
</tr>
<tr>
<td>destination address</td>
<td>(128 bits)</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 4: outline

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
Dijsktra’s Algorithm

1 *Initialization:*
2 \( N' = \{u\} \)
3 for all nodes \( v \)
4 if \( v \) adjacent to \( u \)
5 then \( D(v) = c(u,v) \)
6 else \( D(v) = \infty \)

8 *Loop*
9 find \( w \) not in \( N' \) such that \( D(w) \) is a minimum
10 add \( w \) to \( N' \)
11 update \( D(v) \) for all \( v \) adjacent to \( w \) and not in \( N' \): 
12 \( D(v) = \min( D(v), D(w) + c(w,v) ) \)
13 /* new cost to \( v \) is either old cost to \( v \) or known 
14 shortest path cost to \( w \) plus cost from \( w \) to \( v \) */ 
15 *until all nodes in \( N' \)*
Distance vector algorithm

**Bellman-Ford equation**
\[ d_x(y) = \min_v \{ c(x,v) + d_v(y) \} \]

**iterative, asynchronous:** each local iteration caused by:
- local link cost change
- DV update message from neighbor

**distributed:**
- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

**each node:**
- *wait* for (change in local link cost or msg from neighbor)
- *recompute* estimates
- if DV to any dest has changed, *notify* neighbors
Hierarchical Routing

- aggregate routers into regions: “autonomous systems” (AS)
- “intra-AS” routing
  - routers in same AS run same routing protocol
  - routers in different AS can run different intra-AS routing protocol
- “inter-AS” routing
  - Gateway router connecting another AS
Chapter 4: outline

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
RIP (Routing Information Protocol)

- distance vector algorithm, based on # of hops
- distance vectors: exchanged among neighbors every 30 sec
- poison reverse used to prevent ping-pong loops
- application-level process implemented on UDP
OSPF (Open Shortest Path First)

- open protocol, based on Link State algorithm
- advertisements disseminated to entire AS (via flooding)
- password based authentication
- ECMP: equal-cost multiple paths
- hierarchical OSPF in large domains
BGP (Border Gateway Protocol)

- two types of BGP (TCP) sessions
  - eBGP: between neighboring ASs
  - iBGP: inside same AS

- router may learn about more than 1 route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria
Chapter 4: outline

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
Approaches for building mcast trees

approaches:

- **source-based tree**: one tree per source
  - shortest path trees
  - reverse path forwarding
- **group-shared tree**: group uses one tree
  - minimal spanning (Steiner)
  - center-based trees
Shortest path tree

- mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra’s algorithm

LEGEND

- router with attached group member
- router with no attached group member
- link used for forwarding, i indicates order link added by algorithm

s: source

Network Layer 4-28
Reverse path forwarding

LEGEND

- router with attached group member
- router with no attached group member
- datagram will be forwarded
- datagram will not be forwarded

**if** (mcast datagram received on incoming link on shortest path back to center)

**then** flood datagram onto all outgoing links

**else** ignore datagram
Reverse path forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - “prune” msgs sent upstream by router with no downstream group members

s: source

LEGEND
- router with attached group member
- router with no attached group member
- prune message
- links with multicast forwarding
Center-based trees

- single delivery tree shared by all
- one router identified as “center” of tree

To join:
- edge router sends unicast join-msg addressed to center router
- join-msg “processed” by intermediate routers and forwarded towards center
- join-msg either hits existing tree branch for this center, or arrives at center
- path taken by join-msg becomes new branch of tree for this router
Chapter 5
Link Layer

Slides adopted from original ones provided by the textbook authors.
Link layer, LANs: outline

5.1 introduction, services
5.2 error detection, correction
5.3 multiple access protocols
5.4 LANs
  ▪ addressing, ARP
  ▪ Ethernet
  ▪ switches
  ▪ VLANS
5.5 link virtualization: MPLS
5.6 data center networking
5.7 a day in the life of a web request
Link layer services

- framing
- link access
- error detection and correction
Link layer, LANs: outline

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

5.5 link virtualization: MPLS
5.6 data center networking
5.7 a day in the life of a web request
**Parity checking**

**single bit parity:**
- detect single bit errors

![Example of single bit parity with bit positions and parity bit]

**two-dimensional bit parity:**
- detect and correct single bit errors

![Example of two-dimensional bit parity with data bits and parity bits]

Even parity: parity bit chosen for even # of 1s
Odd parity: parity bit chosen for odd # of 1s

```plaintext
Even parity: 1010111
Odd parity: 1010111
```

```plaintext
No errors
```

```
No errors
```
CRC calculation

want:
\[ D \cdot 2^r \text{ XOR } R = nG \]
equivalently:
\[ D \cdot 2^r = nG \text{ XOR } R \]
equivalently:
if we divide \( D \cdot 2^r \) by \( G \), want remainder \( R \) to satisfy:

\[ R = \text{remainder}[\frac{D \cdot 2^r}{G}] \]
Link layer, LANs: outline

5.1 introduction, services
5.2 error detection, correction
5.3 multiple access protocols
5.4 LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANS
5.5 link virtualization: MPLS
5.6 data center networking
5.7 a day in the life of a web request
Summary of MAC protocols

- **channel partitioning**, by time, frequency or code
  - Time Division
  - Frequency Division
  - Code Division

- **random access** (dynamic),
  - ALOHA, S-ALOHA
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet

- **taking turns**
  - polling from central site used in bluetooth
  - token passing used in FDDI, token ring
Link layer, LANs: outline

5.1 introduction, services
5.2 error detection, correction
5.3 multiple access protocols
5.4 LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANS
5.5 link virtualization: MPLS
5.6 data center networking
5.7 a day in the life of a web request
ARP: mapping IP to MAC

- A wants to send datagram to B
  - B’s MAC address not in A’s ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A’s MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is “plug-and-play”:
  - nodes create their ARP tables without intervention from net administrator
Addressing: routing to another LAN

walkthrough: send datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address
- assume A knows IP address of first hop router, R (how?)
- assume A knows R's MAC address (how?)

A
- 111.111.111.111
  74-29-9C-E8-FF-55
- 111.111.111.112
  CC-49-DE-D0-AB-7D

R
- 222.222.222.220
  1A-23-F9-CD-06-9B

B
- 222.222.222.222
  49-BD-D2-C7-56-2A
- 222.222.222.221
  88-B2-2F-54-1A-0F
Link layer, LANs: outline

5.1 introduction, services
5.2 error detection, correction
5.3 multiple access protocols
5.4 LANs
   - addressing, ARP
   - Ethernet
   - switches
   - VLANS
5.5 link virtualization: MPLS
5.6 data center networking
5.7 a day in the life of a web request
Ethernet: physical topology

- **bus**: popular through mid 90s
  - all nodes in same collision domain (can collide with each other)

- **star**: prevails today
  - active *switch* in center
  - each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)
frame structure: sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

features:
- **Connectionless**
- **Unreliable**
- MAC protocol: unslotted **CSMA/CD with binary backoff**
Link layer, LANs: outline

5.1 introduction, services
5.2 error detection, correction
5.3 multiple access protocols
5.4 LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANS
5.5 link virtualization: MPLS
5.6 data center networking
5.7 a day in the life of a web request
Ethernet switch

- link-layer device: takes an active role
  - store, forward Ethernet frames
  - examine incoming frame’s MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment

- transparent
  - hosts are unaware of presence of switches

- plug-and-play, self-learning
  - switches do not need to be configured
Switch: self-learning

- switch learns which hosts can be reached through which interfaces
  - when frame received, switch “learns” location of sender
  - records sender/location pair in switch table
- forwarding packet
  - frame destination unknown: flood
  - destination location known: selective send
VLANs: motivation

- Motivation
  - Lack of traffic isolation
  - Inefficient use of switches
  - Managing users

- Virtual Local Area
  - Switch(es) supporting VLAN capabilities can be configured to define multiple virtual LANS over single physical LAN infrastructure.
  - trunk port: carries frames between VLANS defined over multiple physical switches via the 802.1q protocol
## Link layer, LANs: outline

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>introduction, services</td>
</tr>
<tr>
<td>5.2</td>
<td>error detection, correction</td>
</tr>
<tr>
<td>5.3</td>
<td>multiple access protocols</td>
</tr>
<tr>
<td>5.4</td>
<td>LANs</td>
</tr>
<tr>
<td></td>
<td>- addressing, ARP</td>
</tr>
<tr>
<td></td>
<td>- Ethernet</td>
</tr>
<tr>
<td></td>
<td>- switches</td>
</tr>
<tr>
<td></td>
<td>- VLANS</td>
</tr>
<tr>
<td>5.5</td>
<td>link virtualization: MPLS</td>
</tr>
<tr>
<td>5.6</td>
<td>data center networking</td>
</tr>
<tr>
<td>5.7</td>
<td>a day in the life of a web request</td>
</tr>
</tbody>
</table>
Multiprotocol label switching (MPLS)

- initial goal: high-speed IP forwarding using fixed length label (instead of IP address)
  - fast lookup using fixed length identifier (rather than shortest prefix matching)
  - borrowing ideas from Virtual Circuit (VC) approach
  - but IP datagram still keeps IP address!

![MPLS header diagram]

<table>
<thead>
<tr>
<th>Label</th>
<th>Exp</th>
<th>S</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
MPLS versus IP paths

- **IP routing:** path to destination determined by destination address alone

- **MPLS routing:** path to destination can be based on source and dest. address
  - **fast reroute:** precompute backup routes in case of link failure
Chapter 7
Multimedia Networking

Slides adopted from original ones provided by the textbook authors.
Multimedia networking: outline

7.1 multimedia networking applications
7.2 streaming stored video
7.3 voice-over-IP
7.4 protocols for real-time conversational applications
7.5 network support for multimedia
Multimedia networking: 3 application types

- **streaming, stored** audio, video
  - *streaming*: can begin playout before downloading entire file
  - *stored (at server)*: can transmit faster than audio/video will be rendered (implies storing/buffering at client)
  - e.g., YouTube, Netflix, Hulu

- **conversational** voice/video over IP
  - interactive nature of human-to-human conversation limits delay tolerance
  - e.g., Skype

- **streaming live** audio, video
  - e.g., live sporting event
Multimedia networking: outline

7.1 multimedia networking applications
7.2 streaming stored video
7.3 voice-over-IP
7.4 protocols for real-time conversational applications
7.5 network support for multimedia
client-side buffering and playout delay: compensate for network-added delay, delay jitter
Streaming multimedia: UDP

- server sends at rate appropriate for client
  - often: send rate = encoding rate = constant rate
  - transmission rate can be oblivious to congestion levels
- short playout delay (2-5 seconds) to remove network jitter
- error recovery: application-level, timepermitting
Streaming multimedia: DASH

- **DASH**: Dynamic, Adaptive Streaming over HTTP

- **Server**:
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file*: provides URLs for different chunks

- **Client**:
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)
CDN: “simple” content access scenario

Bob (client) requests video http://netcinema.com/6Y7B23V
- video stored in CDN at http://KingCDN.com/NetC6y&B23V

2. Resolve http://netcinema.com/6Y7B23V via Bob’s local DNS
3. netcinema’s authoritative DNS returns URL http://KingCDN.com/NetC6y&B23V
4&5. Resolve http://KingCDN.com/NetC6y&B23 via KingCDN’s authoritative DNS, which returns IP address of KingCDN server with video
6. Request video from KINGCDN server, streamed via HTTP
Multimedia networking: outline

7.1 multimedia networking applications
7.2 streaming stored video
7.3 voice-over-IP
7.4 protocols for real-time conversational applications
7.5 network support for multimedia
Adaptive playout delay

- **goal:** low playout delay, low late loss rate
- **approach:** adaptive playout delay adjustment:
  - estimate network delay, adjust playout delay at beginning of each talk spurt
  - silent periods compressed and elongated
  - chunks still played out every 20 msec during talk spurt
- adaptively estimate packet delay: (EWMA - exponentially weighted moving average, recall TCP RTT estimate):
  \[
  d_i = (1 - \alpha)d_{i-1} + \alpha (r_i - t_i)
  \]
  - delay estimate after ith packet
  - small constant, e.g. 0.1
  - time received - time sent (timestamp)
  - measured delay of ith packet
**VoIP: recovery from packet loss (1)**

**Challenge:** recover from packet loss given small tolerable delay between original transmission and playout
- each ACK/NAK takes $\sim$ one RTT
- alternative: *Forward Error Correction (FEC)*
  - send enough bits to allow recovery without retransmission (recall two-dimensional parity in Ch. 5)

**Simple FEC**
- for every group of $n$ chunks, create redundant chunk by exclusive OR-ing $n$ original chunks
- send $n+1$ chunks, increasing bandwidth by factor $1/n$
- can reconstruct original $n$ chunks if at most one lost chunk from $n+1$ chunks, with playout delay
another FEC scheme:

- “piggyback lower quality stream”
- send lower resolution audio stream as redundant information
- e.g., nominal stream PCM at 64 kbps and redundant stream GSM at 13 kbps
- non-consecutive loss: receiver can conceal loss
interleaving to conceal loss:

- audio chunks divided into smaller units, e.g. four 5 msec units per 20 msec audio chunk
- packet contains small units from different chunks
- if packet lost, still have most of every original chunk
- no redundancy overhead, but increases playout delay
Multimedia networking: outline

7.1 multimedia networking applications
7.2 streaming stored video
7.3 voice-over-IP
7.4 protocols for real-time conversational applications
7.5 network support for multimedia
Scheduling and policing mechanisms

- **scheduling**: choose next packet to send on link
- **FIFO (first in first out) scheduling**: send in order of arrival to queue
  - real-world example?
  - **discard policy**: if packet arrives to full queue: who to discard?
    - *tail drop*: drop arriving packet
    - *priority*: drop/remove on priority basis
    - *random*: drop/remove randomly

![Diagram of packet arrivals, queue, link, and packet departures]
Scheduling policies: priority

**priority scheduling**: send highest priority queued packet

- multiple classes, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
  - real world example?
Scheduling policies: still more

Round Robin (RR) scheduling:

- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)
- real world example?
Scheduling policies: still more

**Weighted Fair Queuing (WFQ):**
- generalized Round Robin
- each class gets weighted amount of service in each cycle
- real-world example?
Policing mechanisms: implementation

**token bucket**: limit input to specified *burst size* and *average rate*

- bucket can hold b tokens
- tokens generated at rate \( r \) token/sec unless bucket full
- **over interval of length \( t \): number of packets admitted less than or equal to \( (r \cdot t + b) \)**
Principles for QOS Guarantees

- resource reservation: routers maintain state info of allocated resources
- **RSVP: Resource Reservation Protocol**
  - allows applications to reserve bandwidth for their data flows.
- admit/deny new call setup requests