Error control

3 steps:
1. **Protection**: Add redundancy to the payload.
   - Transmission
   - Reception (probably – detect loss by using sequence numbers)
2. **Detection** of errors
   - Generally based on checking redundancy added during the protection step.
3. **Correction** of errors
   - May involve retransmission by the source, or use of redundancy transmitted with the payload.

Parity checking

Add an extra bit whose value is such that the total number of 1s in the payload+extra_bit is either
- **even**: Even parity
- **odd**: Odd parity
  (Generally prefer odd parity s.t. we get some signal variations when transmitting information, e.g. 0V=0, 1V=1 don’t transmit 0s indefinitely.)

Properties:
- Can detect any single error
- Can detect any odd number of errors
- Can’t detect even number of errors

Basic idea of CRCs

Rather than use addition as the operator, use division: Ensure that payload+CRC is divisible by some known value, without remainder.

- e.g. Knowing that computer memory sizes are generally powers of 2, e.g. 128MB, 512MB → A memory specification of "1 38MB" is likely to contain an error.
- Want to transmit unmodified payload + redundancy → don’t directly multiply payload, but extend it, then ensure that extended number is a multiple of the known value.

Outline
RDT2.0: channel with bit errors

Underlying channel may flip bits in packet
- recall: CRCs & checksums to detect bit errors

The question: how to recover from errors:
- acknowledgements (ACKs): receiver explicitly tells sender that packet received OK
- negative acknowledgements (NAKs): receiver explicitly tells sender that packet had errors
- sender retransmits packet on receipt of NAK
- human scenarios using ACKs, NAKs?

New mechanisms in RDT2.0 (beyond RDT1.0):
- error detection
- Feedback (receiver->sender): control msgs (ACK, NAK)

† Assumes that receiver can tell that a packet was sent in the first place.
  e.g. if packets have sequence numbers, may detect loss by virtue of next received packet not having expected sequence number.

Stop-and-wait

RDT 2.0 is an example of the “Stop-and-wait” family of protocols:
1. Source sends a frame.
2. Source stops sending, and waits to receive feedback, before sending again.

Pipelining: increased utilization

First packet bit transmitted, \( t = 0 \)
Last bit transmitted, \( t = L / R \)

<table>
<thead>
<tr>
<th>Event</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTT</td>
<td>( 0.24 ) ms</td>
</tr>
<tr>
<td>L / R</td>
<td>( 0.0008 ) microsec</td>
</tr>
</tbody>
</table>

Increase utilization by a factor of 3!

Go-Back-N

**Sender:**
- \( k \)-bit seq # in packet header
- “window” of up to \( N \), consecutive unack’ed packets allowed
- ACK(n): ACKs all packets up to, including seq # \( n \)-“cumulative ACK”.
- timer for each in-flight packet
- timeout(n): retransmit packet n and all higher seq # packets in window

In the context of TCP, we’ll see what info can be gleaned from duplicate acks.
Selective Repeat

Receiver *individually* acknowledges all correctly received packets
- buffers packets, as needed, for eventual in-order delivery to upper layer
Sender only resends packets for which ACK not received
- sender timer for each unACKed packet
Sender window
- $N$ consecutive seq #'s
- again limits seq #s of sent, unACKed packets

† Strictly, it need only be able to ack individual packets. When consecutive packets are received, it may be more efficient to ack these as a group. Send individual acks for packets received before predecessors.

Outline

Comparison of techniques

Static channel partitioning:
- share channel efficiently and fairly at high load†
- inefficient at low load: delay in channel access, $1/N$ bandwidth allocated even if only 1 active node!

Contestion-based protocols
- efficient at low load: single node can fully utilize channel
- high load: collision overhead

Contention-free “taking turns” protocols
- look for best of both worlds!
- Require ordering of nodes or central failure point.

† "load" here = number of active stations

Contention-free protocols

Reservation protocols
Desire to transmit is broadcast before the actual transmission.
Displaces contention to the reservation channel, e.g.:
- Bit map
- Binary countdown

Turn taking protocols
- One entity coordinates access by all stations
- The "entity" may be:
- a station ("master"; others are "slaves"), or
- a piece of information ("token")
- → Token bus, ring, FDDI
Aloha

“Aloha” = Hawaiian greeting
Protocol developed at U. Hawaii to connect computers on separate islands by radio

Repeat
1. Transmit frame
2. Check for collision when transmitting frame
   - Listen to see how clear transmitted signal is
     - May not work if transmission power >> receive power
     - May not work when propagation delays cause another transmission (colliding at some point) to only be heard after this transmission has finished.
   - Wait to receive acknowledgement from destination
3. If collision, backoff for a random period
   - If multiple stations deterministically chose the same backoff period, then they would collide when they try again.
   - Random backoff makes repeated collision improbable.
Until transmitted without collision

2 variants: Pure and Slotted Aloha

Photo from corbis.com

Binary exponential backoff

How to choose the random backoff period?
Range should increase with amount of contention.

Exponential increase: After the $c$th collision, backoff for a period randomly chosen in the range $[0, (2^c-1)\times\text{slotTime}]$

- slotTime: Defined later – time to be sure transmission won’t experience collision = 512 bit times

- e.g.:
  1. after 1st collision backoff $[0, 1] \times \text{slotTimes}$
  2. after 2nd backoff randomly chosen interval in range $[0, 3]$
  3. after 3rd backoff randomly chosen interval in range $[0, 8]$

Exponential increase quickly leads to large (effectively infinite) range. Often saturate at some limit, and give up (& tell higher layer) after some number of attempts
- e.g. Ethernet parameters:
  - backoffLimit=10: range never exceeds $[0, 1023]$
  - attemptLimit=16: give up after 16 attempts

Carrier Sense Multiple Access (CSMA)

Transmitting when others transmit is mainly destructive:
- Destroy frame being sent by other station.
- Only “benefit” is that others become aware of own desire to transmit.

Idea: Listen before transmitting: Sense to detect any carriers from other transmissions. If none, then transmit.

Human analogy: Don’t interrupt others!

Can prioritise access by requiring low priority access to sense channel to be idle for longer. Not used by Ethernet, but used in wireless LANs (e.g. to ensure that receiver can send ack without having to contend for transmission medium).

How long should a station persist in sensing carrier?...

CSMA with collision detection (CD)

Note: If D transmitted for a shorter period of time, it might complete its transmission before the transmission from B arrived at D. D’s transmission would still interfere at B, but D might think that it had completed the transmission successfully. Rather than ask source to monitor for RTT after transmission, impose minimum frame size.
Outline

Wireless MAC Outline

1. Transmission overwhelms local receiver
2. Immediate ACKs for MAC
3. Capture Effect
4. Random delay before transmit
5. Hidden Terminal Problem
6. RTS/CTS handshaking
7. Exposed Terminal Problem

Numbers indicate the order in which we'll cover the topics
Arrows indicate causation

Hidden terminal problem†

Success of a transmission is determined by interference at
the receiver, but MAC decisions are made on the basis
of information available at the transmitters.

S₁ and S₂ are out of range of (hidden from) each other, but
their transmissions interfere at D. Because the sources can't detect the collision, they continue
transmitting, wasting transmission time.

†aka Hidden station problem

RTS/CTS solution for hidden terminals

Idea: Get the destination to transmit a signal informing
stations near it that it needs the channel for a certain
interval.

Implementation: Stations record when they shouldn't
transmit because it is needed for local reception.

They learn this information by:
1. Source sends Request To Send (RTS) to destination,
   indicating expected duration of transmission.
2. Destination sends Clear To Send (CTS) signal
   (indicating remaining duration) to its locality (&
   source).
3. Other recipients of RTS remain silent long enough for
   source to receive CTS.
4. Recipients of CTS suppress transmissions for indicated
duration.
5. If the source doesn't receive a CTS, it waits for a
   random backoff, and tries again.

† The RTS/CTS mechanism also provides a transmission path check.
Outline

- A bridge has a **bridge table**
  - entry in bridge table:
    - (Node LAN Address, Bridge Interface, Time Stamp)
    - stale entries in table dropped (TTL can be 60 min)
  - bridges **learn** which hosts can be reached through which interfaces
    - when frame received, bridge "learns" location of sender: incoming LAN segment
    - records sender/location pair in bridge table

Filtering/Forwarding

*When bridge receives a frame:*

index bridge table using MAC dest address
if entry found for destination
  then{
    if dest on segment from which frame arrived
      then drop the frame
    else forward the frame on interface indicated
  }
else flood

Review of weeks 7-10

**Network layer**
- Addressing
- Internetworking
- Principles of routing protocols
  - Distance-vector
  - Link-state
- Exemplary routing protocols
- Congestion & QoS

**Transport layer**
- Transport services
- UDP: Connectionless transport protocols
- TCP:
  - Connection management
  - Reliable transfer
  - TCP congestion control
IP Addresses

given notion of "network", let’s re-examine IP addresses: "class-full" addressing:

<table>
<thead>
<tr>
<th>Class</th>
<th>Network</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.0.0.0 to 127.255.255.255</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>128.0.0.0 to 191.255.255.255</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>192.0.0.0 to 223.255.255.255</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>224.0.0.0 to 239.255.255.255</td>
<td></td>
</tr>
</tbody>
</table>

32 bits

IP addressing: CIDR

- Classful addressing:
  - inefficient use of address space, address space exhaustion
  - e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network
- CIDR: Classless InterDomain Routing
  - network portion of address of arbitrary length
  - address format: a.b.c.d/x, where x is # bits in network portion of address

11001000  00010111  00010000  00000000

network part

host part

200.23.16.0/23

MAC vs IP addresses

- MAC addresses identify stations.
- IP addresses identify network+host → interface.
- IP addresses can be aggregated:
  - Packets destined to any host on network N follow same path → router only needs to consider most significant bits (network part, and possibly only some of those bits with CIDR)
- MAC addresses can’t be aggregated:
  - Adjacent stations may have completely independent addresses.
  - bridges have to record information for individual stations → scalability problems.

ARP protocol

A wants to send datagram to B, and A knows B’s IP address.
Suppose B’s MAC address is not in A’s ARP table.
A broadcasts ARP query packet, containing B’s IP address
- all machines 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
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 and "on")
Support for mobile users who want to join network (more shortly)

DHCP overview:
- host broadcasts "DHCP discover" msg
- DHCP server responds with "DHCP offer" msg
- host requests IP addr.: "DHCP request" msg
- DHCP server sends addr.: "DHCP ack" msg

DHCP is an extension of BOOTP ⇒ You may see it called BOOTP by sniffers

NAT: Network Address Translation

Outline

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
ICMP: Internet Control Message Protocol

- used by hosts, routers, gateways 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
- Type Code description
  0 0 echo reply (ping)
  3 0 dest. network unreachable
  3 1 dest host unreachable
  3 2 dest protocol unreachable
  3 3 dest port unreachable
  3 6 dest network unknown
  3 7 dest host unknown
  4 0 source quench (congestion control - not used)
  8 0 echo request (ping)
  9 0 route advertisement
  10 0 router discovery
  11 0 TTL expired
  12 0 bad IP header

Outline

Dijkstra’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 \( D(v) = 8 \)
7
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\n14 shortest path cost to \( w \) plus cost from \( w \) to \( v \)
15 until all nodes in \( N' \)

Dijkstra's algorithm: example

<table>
<thead>
<tr>
<th>Step</th>
<th>( N' )</th>
<th>( D(v),p(v) )</th>
<th>( D(w),p(w) )</th>
<th>( D(x),p(x) )</th>
<th>( D(y),p(y) )</th>
<th>( D(z),p(z) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( u )</td>
<td>2, u</td>
<td>5, u</td>
<td>1, u</td>
<td>8, u</td>
<td>8, u</td>
</tr>
<tr>
<td>1</td>
<td>( x )</td>
<td>2, u</td>
<td>4, x</td>
<td>1, u</td>
<td>2, x</td>
<td>4, x</td>
</tr>
<tr>
<td>2</td>
<td>( x )</td>
<td>2, u</td>
<td>3, y</td>
<td>1, u</td>
<td>2, x</td>
<td>3, y</td>
</tr>
<tr>
<td>3</td>
<td>( u )</td>
<td>2, u</td>
<td>3, y</td>
<td>1, u</td>
<td>2, x</td>
<td>4, y</td>
</tr>
<tr>
<td>4</td>
<td>( y )</td>
<td>2, u</td>
<td>3, y</td>
<td>1, u</td>
<td>2, x</td>
<td>4, y</td>
</tr>
<tr>
<td>5</td>
<td>( v )</td>
<td>2, u</td>
<td>3, y</td>
<td>1, u</td>
<td>2, x</td>
<td>4, y</td>
</tr>
</tbody>
</table>
Distance-vector vs link-state

<table>
<thead>
<tr>
<th></th>
<th>Who is told?</th>
<th>What is told?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance-vector</td>
<td>neighbours</td>
<td>every node in network</td>
</tr>
<tr>
<td>Link-state</td>
<td>every node in network</td>
<td>neighbours</td>
</tr>
</tbody>
</table>

Distance Vector Routing Algorithm

iterative:
- continues until no nodes exchange info.
- self-terminating: no "signal" to stop

asynchronous:
- nodes need not exchange info/iterate in lock step!
- distributed:
  - each node communicates only with directly-attached neighbors

Distance Table data structure:
- each node has its own
- row for each possible destination
- column for each directly-attached neighbor to node
- example: in node X, for dest. Y via neighbor Z:

\[
D(X,Y) = \begin{cases} 
Y, & \text{via Z as next hop} \\
\infty, & \text{if no path} \\
c(X,Z) + \min_{W} \{ D(Y,W) \} & \text{if Z has a shorter path to Y} 
\end{cases}
\]

Outline

Intra-AS and Inter-AS routing

Gateways:
- perform inter-AS routing amongst themselves
- perform intra-AS routers with other routers in their AS

inter-AS, intra-AS routing in gateway A.c

network layer
link layer
physical layer

DL
DL
DL
DL
Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

Internet inter-AS routing: BGP

BGP (Border Gateway Protocol): the de facto standard

Path Vector protocol:
  - similar to Distance Vector protocol
  - each Border Gateway broadcast to neighbors (peers) entire path (i.e., sequence of AS's) to destination
  - BGP routes to networks (ASs), not individual hosts
  - E.g., Gateway X may send its path to dest. Z:
    \[ \text{Path} (X, Z) = X, Y_1, Y_2, Y_3, \ldots, Z \]