Link Layer
(continued)
Topics

1. Framing
   • Delimiting start/end of frames
2. Error detection and correction
   • Handling errors
3. Retransmissions
   • Handling loss
4. Multiple Access
   • 802.11, classic Ethernet
5. Switching
   • Modern Ethernet
Retransmissions
Context on Reliability

- Where in the stack should we place reliability functions?

<table>
<thead>
<tr>
<th>Application</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td></td>
</tr>
<tr>
<td>Link</td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td></td>
</tr>
</tbody>
</table>
Context on Reliability

• Everywhere? It is a key issue
  • Different layers contribute differently

- Application
- Transport
- Network
- Link
- Physical

Recover actions
(correctness)

Mask errors
(performance optimization)
ARQ (Automatic Repeat reQuest)

• ARQ often used when errors are common or must be corrected
  • E.g., WiFi, and TCP

• Rules at sender and receiver:
  • Receiver automatically acknowledges correct frames with an ACK
  • Sender automatically resends after a timeout, until an ACK is received
So What’s Tricky About ARQ?

• Two non-trivial issues:
  • How long to set the timeout?
  • How to avoid accepting duplicate frames as new frames

• Want performance in the common case and correctness always
Timeouts

• Timeout should be:
  • Not too big (link goes idle)
  • Not too small (spurious resend)

• Fairly easy on a LAN
  • Clear worst case, little variation

• Fairly difficult over the Internet
  • Much variation, no obvious bound
  • We’ll revisit this with TCP (later)
Detecting Duplicates

• Frames and ACKs must both carry UIDs for correctness

• Sequence numbers are a handy form of UID that also allow receiver to detect missing frames
  • Useful for sliding window

• Do we need sliding window on a LAN?
Link Layer Retransmission Summary

• Should retransmissions occur at link layer
  • Depends on expected error rate
  • Think of them as a performance optimization (relative to just leaving it to TCP) when they’re implemented

• Because latencies are typically small(ish) and tightly bounded on a single link
  • Timeout estimation is simpler
  • Less motivation to use sliding window, rather than stop-and-wait
Multiple Access
Topic

- **Multiplexing** is the network word for the sharing of a resource

- Classic scenario is sharing a link among different users
  - Time Division Multiplexing (TDM)
  - Frequency Division Multiplexing (FDM)
Time Division Multiplexing (TDM)

• Users take turns on a fixed schedule
Frequency Division Multiplexing (FDM)

• Put different users on different frequency bands

Overall FDM channel
TDM versus FDM (2)

• In TDM a user sends at a high rate a fraction of the time; in FDM, a user sends at a low rate all the time
TDM/FDM Usage

• Statically divide a resource
  • Suited for continuous traffic, fixed number of users

• Widely used in telecommunications
  • TV and radio stations (FDM)
  • GSM (2G cellular) allocates calls using TDM within FDM
Multiplexing Network Traffic

• **Network traffic is bursty**
  - ON/OFF sources
  - Load varies greatly over time

![Graph showing bursty network traffic](image)

CSE 461 University of Washington
Multiplexing Network Traffic (2)

- **Network traffic is bursty**
  - Inefficient to always allocate user their ON needs with TDM/FDM
Multiplexing Network Traffic (3)

- Multiple access schemes multiplex users according to demands – for gains of statistical multiplexing

![Diagram showing rate and time with two users, each needing R, and combined needing R’ < 2R]
Random Access

• How do nodes share a single link? Who sends when, e.g., in WiFi?
  • Explore with a simple model

• Assume no-one is in charge
  • Distributed system
Random Access

• We will explore random multiple access control (MAC) protocols
  • This is the basis for classic Ethernet
  • Remember: data traffic is bursty
ALOHA Network

• Seminal computer network connecting the Hawaiian islands in the late 1960s
  • When should nodes send?
  • A new protocol was devised by Norm Abramson ...
ALOHA Protocol

• Simple idea:
  • Node just sends when it has traffic.
  • If there was a collision (no ACK received) then wait a random time and resend

• That’s it!
ALOHA Protocol

• Some frames will be lost, but many may get through...

• Good idea?
ALOHA Protocol

• Simple, decentralized protocol that works well under low load!

• Not efficient under high load
  • Analysis shows at most 18% efficiency
  • Improvement: divide time into slots and efficiency goes up to 36%

• We’ll look at other improvements
Classic Ethernet

• ALOHA inspired Bob Metcalfe to invent Ethernet for LANs in 1973
  • Nodes share 10 Mbps coaxial cable
  • Hugely popular in 1980s, 1990s
CSMA (Carrier Sense Multiple Access)

• Improve ALOHA by listening for activity before we send (Doh!)
  • Can do easily with wires, not wireless

• So does this eliminate collisions?
  • Why or why not?
CSMA

• Still possible to listen and hear nothing when another node is sending because of delay
CSMA/CD (with Collision Detection)

• Can reduce the cost of collisions by detecting them and aborting (Jam) the rest of the frame time
  • Again, we can do this with wires
CSMA/CD Complications

- Everyone who collides needs to know it happened
  - Time window in which a node may hear of a collision is 2D seconds
CSMA/CD Complications

• Impose a minimum frame length of 2D seconds
  • So node can’t finish before collision
  • Ethernet minimum frame is 64 bytes
CSMA “Persistence”

• What should a node do if another node is sending?

• Idea: Wait until it is done, and send
CSMA “Persistence” (2)

• Problem is that multiple waiting nodes will queue up then collide
  • More load, more of a problem
CSMA “Persistence”

• Intuition for a better solution
  • If there are $N$ queued senders, we want each to send next with probability $1/N$