Lecture 7: Transport Control Protocol

CSE 123: Computer Networks
Alex C. Snoeren
● TCP Connection Management
  ◆ Graceful, symmetric close
  ◆ The TIME_WAIT state

● Connecting together links to form networks
  ◆ Hubs/repeaters
  ◆ Switching
TCP State Transitions

CSE 123 – Lecture 7: TCP
Again, with States

Active participant (client)  Passive participant (server)

SYN_SENT

LISTEN

SYN_RCVD

ESTABLISHED

SYN, SequenceNum = x

ACK, Acknowledgment = x + 1

SYN, SequenceNum = y,

ACK, Acknowledgment = y + 1

+data

CSE 123 – Lecture 7: TCP
Connection Teardown

- Orderly release by sender and receiver when done
  - Delivers all pending data and “hangs up”

- Cleans up state in sender and receiver

- TCP provides a “symmetric” close
  - Both sides shutdown independently
TCP Connection Teardown

WEB SERVER

FIN_WAIT_1
FIN
ACK

FIN_WAIT_2
ACK
FIN
ACK

TIME_WAIT

CLOSED

WEB BROWSER

CLOSE_WAIT
LAST_ACK

CLOSED

CSE 123 – Lecture 7: TCP
The TIME_WAIT State

- We wait 2*MSL (maximum segment lifetime of 60 seconds) before completing the close
  - Why?

- ACK might have been lost and so FIN will be resent
  - Could interfere with a subsequent connection

- Real life: Abortive close
  - Don’t wait for 2*MSL, simply send Reset packet (RST)
  - Why?
Lots of Icky Details

- Window probes
- Silly Window Syndrome
- Nagle’s algorithm
- PAWS
- Etc…

- Steven’s books “TCP/IP Illustrated (vol 1,2)” is a great source of information on this
So far, just one link

- One shared link (a bus) limits scale in terms of:
  - Distance (e.g., 2500 m for Ethernet)
  - Number of nodes (1024 nodes)
  - Performance (Capacity shared across all nodes)

- Often called a Local Area Network (LAN)

- A better alternative is to have multiple busses
  - Each bus is of a limited size, scale, number of hosts, etc.

- Need the ability to connect multiple busses together
  - In other words move frames from one wire to another
Hubs/Repeaters

- Physical layer device
  - One “port” for each LAN
  - Repeat received *bits* on one port out *all* other ports
Hub Advantages

- Hubs can be arranged into hierarchies
  - Ethernet: up to four hubs between any pair of nodes

- Most of LAN continues to operate if “leaf” hub dies

- Simple, cheap
Still One Big Bus

- Single **collision domain**
  - No improvement in max throughput
  - Average throughput < as # of nodes increases
  - Why?

- Still limited in distance and number of hosts
  - Collision detection requirements
  - Synchronization requirements

- Requires performance homogeneity
  - Can’t connect 10 Mbps and 100 Mbps networks
Bridges

- **Store and forward** device
  - Data-link layer device
  - Buffers entire packet and *then* rebroadcasts it on other ports

- Creates **separate** collision domains
  - Uses CSMA/CD for access to each LAN (acts like a host)
  - Can accommodate different speed interfaces (issues?)
  - Separate CDs improves throughput (why?)

- Can significantly improve performance
  - Not all frames go everywhere. (Why did they with a hub?)
Selective Forwarding

- Only rebroadcast a frame to the LAN where its destination resides
  - If A sends packet to X, then bridge must forward frame
  - If A sends packet to B, then bridge shouldn’t

![Diagram of LANs and bridge](image-url)
Need to know “destination” of frame
  ◆ Destination address in frame header (48bit in Ethernet)

Need know which destinations are on which LANs
  ◆ One approach: statically configured by hand
    » Table, mapping address to output port (i.e. LAN)
  ◆ But we’d prefer something automatic and dynamic…

Simple algorithm:
Receive frame \( f \) on port \( q \)
Lookup \( f\).dest for output port /* know where to send it? */
If \( f\).dest found
  then if output port is \( q \) then drop /* already delivered */
    else forward \( f \) on output port;
else flood \( f \);
/* forward on all ports but the one where frame arrived*/
Learning Bridges

- Eliminate manual configuration by learning which addresses are on which LANs

- Basic approach
  - If a frame arrives on a port, then associate its source address with that port
  - As each host transmits, the table becomes accurate

- What if a node moves? Table aging
  - Associate a timestamp with each table entry
  - Refresh timestamp for each new packet with same source
  - If entry gets too stale, remove it
Learning Example

Suppose C sends frame to D and D replies back with frame to C

- C sends frame, bridge has no info about D, so floods to both LANs
  - bridge notes that C is on port 1
  - frame ignored on upper LAN
  - frame received by D
Learning Example

- \(D\) generates reply to \(C\), sends
  - bridge sees frame from \(D\)
  - bridge notes that \(D\) is on port 2
  - bridge knows \(C\) on port 1, so **selectively** forwards frame via port 1
For next time…

- Read P&D 3-3.1