

Lecture 8:

Packet Switching

CSE 123: Computer Networks
Alex C. Snoeren



Lecture 8 overview



- Finish TCP congestion control
- Packet switching



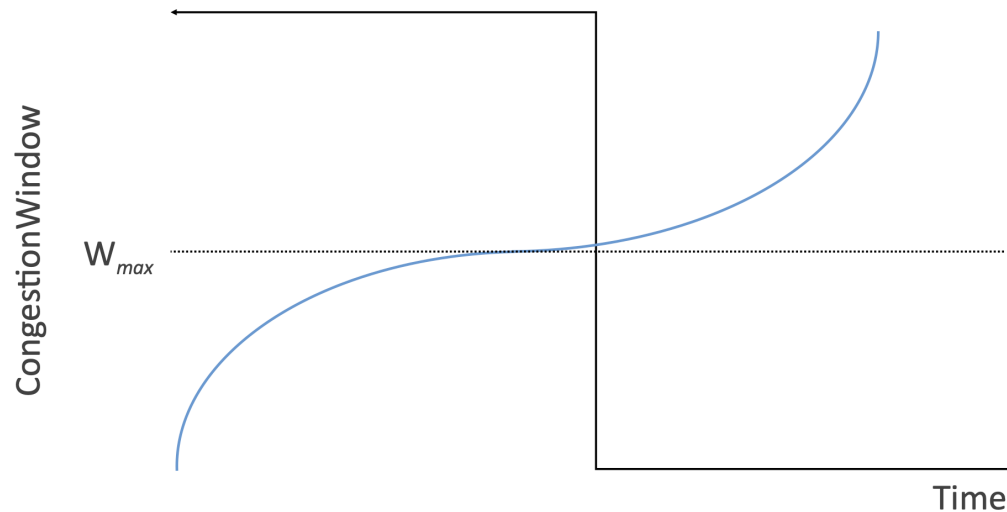
Short connections

- Short connection: only contains a few packets
- How do short connections and Slow Start interact?
 - ◆ What happens when a packet is lost during Slow-Start?
 - ◆ What happens when the SYN is dropped?
- Bottom line: Which packet gets dropped matters a lot
 - ◆ SYN
 - ◆ Slow Start
 - ◆ Congestion avoidance

TCP Cubic



- What happens when new bandwidth becomes available?
 - ♦ AIMD will take a very long time to make use of it
 - ♦ TCP CUBIC effectively “returns to Slow Start” after a while

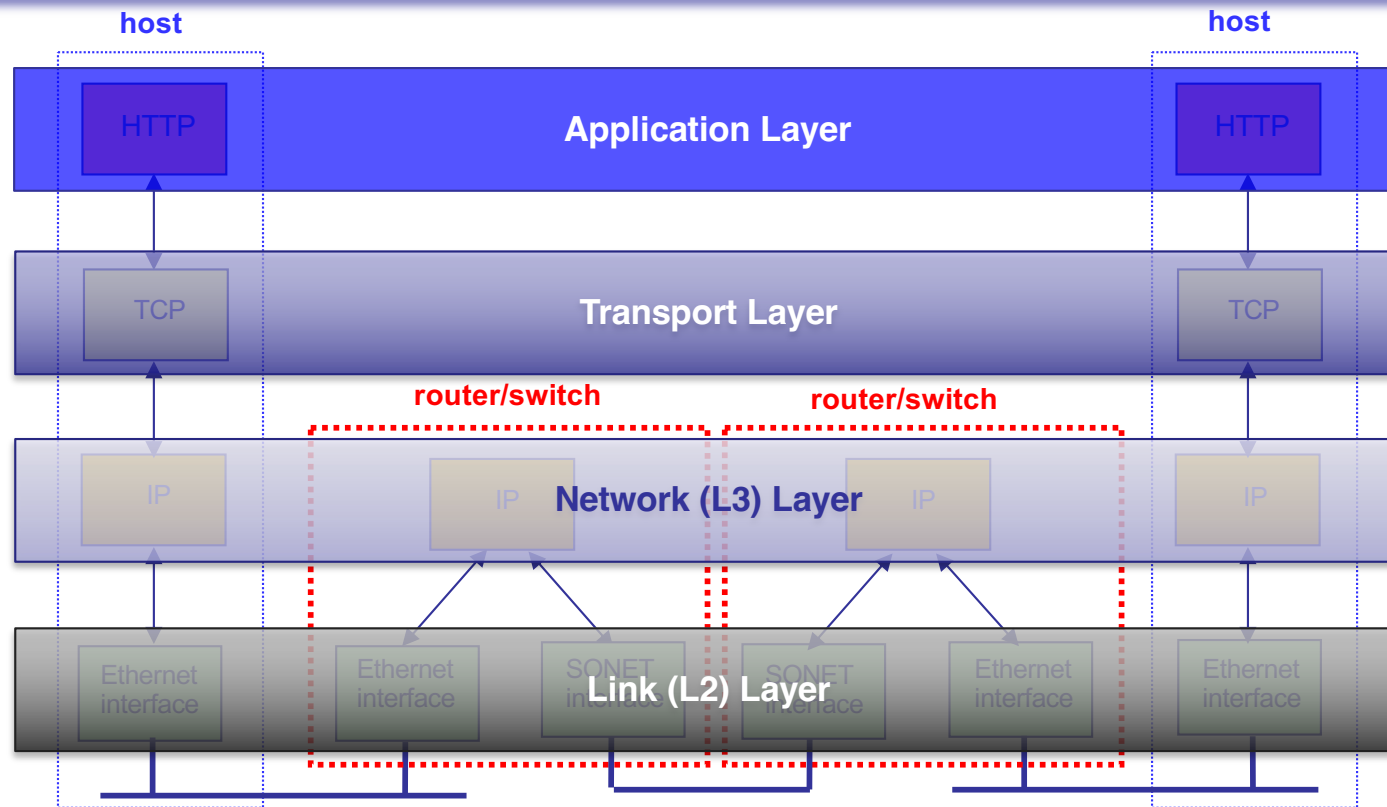




TCP CC summary

- TCP Probes the network for bandwidth, assuming that loss signals congestion
- The congestion window is managed with an additive increase/multiplicative decrease policy
 - ◆ It took fast retransmit and fast recovery to get there
 - ◆ Fast recovery keeps pipe “full” while recovering from a loss
- Slow start is used to avoid lengthy initial delays
 - ◆ Ramp up to near target rate, then switch to AIMD

TCP/IP protocol stack

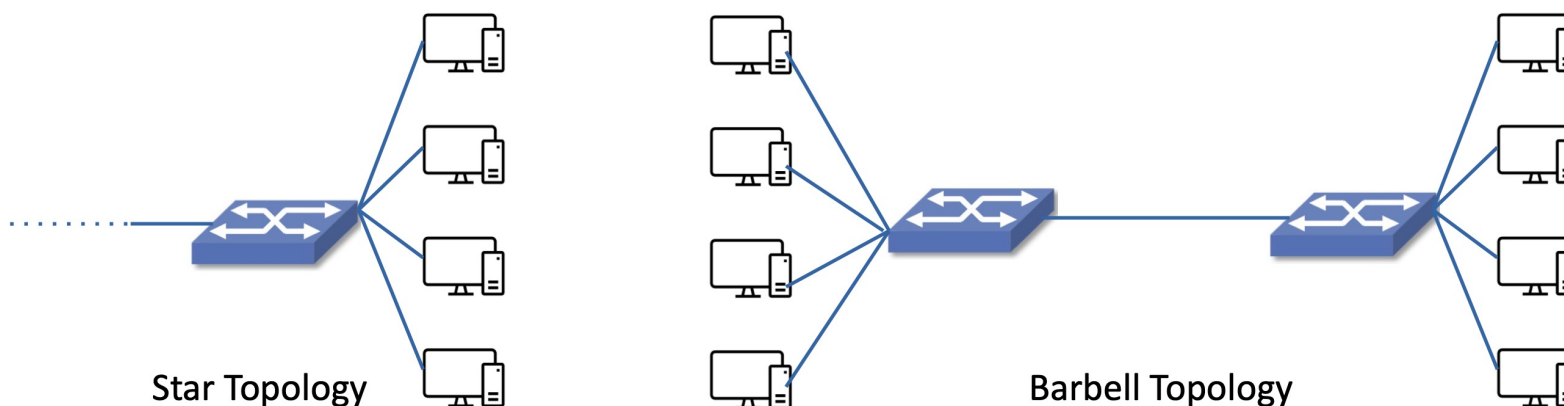


One link is not enough



- Sets of links **bridged** together into a **Local Area Network (LAN)**
 - ◆ Link is multiplexed across time
 - ◆ Frames have a link-layer header with addresses of source and destination of hosts on the LAN
- One shared link (a **bus**) limits scale in terms of:
 - ◆ Distance (e.g., 2500 m for Ethernet)
 - ◆ Performance (Capacity shared across all nodes)
- 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

Switched networks



- Connect together multiple links, either of the *same* link (L2) protocol
 - ◆ E.g., **L2 switches** typically forward Ethernet frames
- Or *different* link protocols
 - ◆ Device needs to "speak" multiple L2 protocols, one per interface/port
 - ◆ Such devices are known as **L3 switches** (aka **routers**) and forward IP **packets**



What's in a switch?

- Physical components
 - ♦ One or more **input interfaces (ports)** that receive packets
 - ♦ One or more **output interfaces** that transmit packets
 - ♦ A chassis (box + power) to hold it all
- Functions
 - ♦ **Forward** packets
 - ♦ **Drop** packets (congestion, security, ...)
 - ♦ **Delay** packets (QoS)
 - ♦ **Transform** packets? (NAT, encapsulation, tunneling, ...)



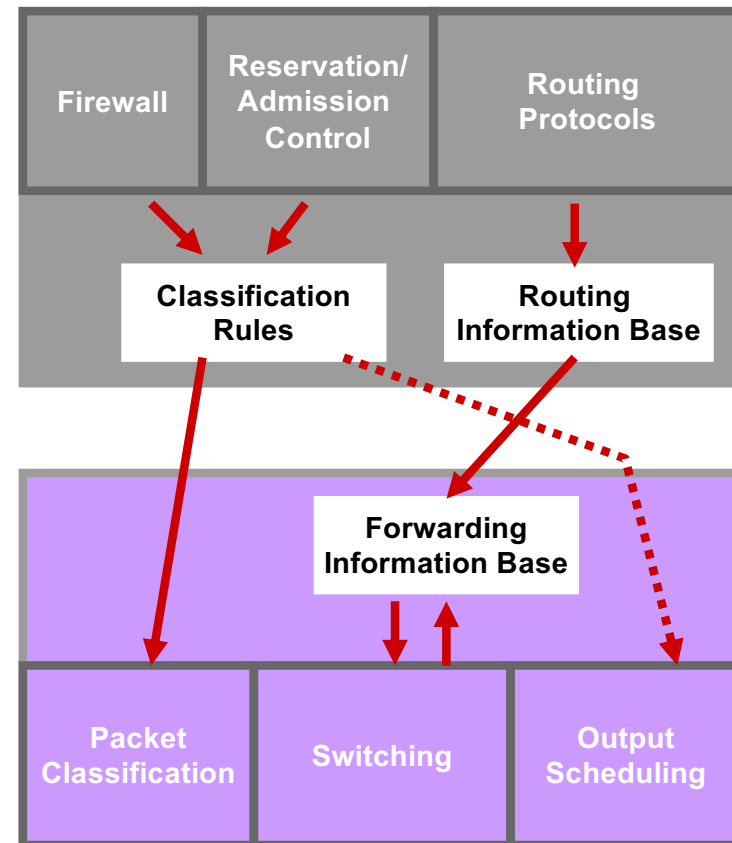
Switch functions

1. Receive incoming packet from link input interface
2. Validate checksum, buffer packet in input queue
3. Lookup packet destination in forwarding table (destination, output port(s))
4. Filter and/or update packet as necessary
5. Send packet to output interface (interfaces?)
6. Buffer packet in output queue
7. Send packet to output interface link

Functional architecture



- **Control Plane**
 - ◆ Complex
 - ◆ Per-control action
 - ◆ May be slow
- **Data plane**
 - ◆ Simple
 - ◆ Per-packet
 - ◆ Must be fast





Forwarding Information Base (FIB)

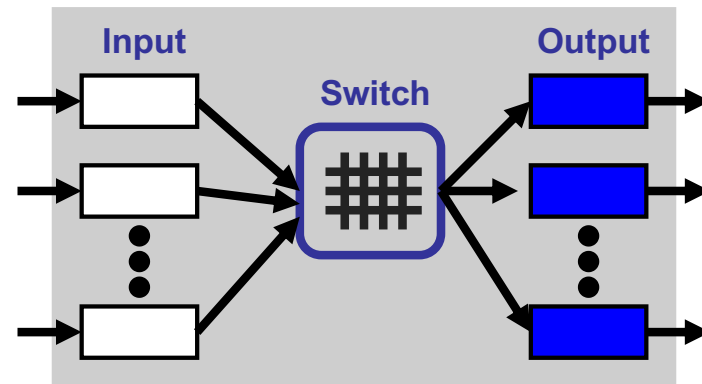
- Need to know “destination” of frame
 - ♦ Destination address in frame header (48bit in Ethernet)
- Need know which destinations are on which LANs
 - ♦ One approach: statically configured by hand
 - » Table, mapping address to output port (i.e. LAN)
 - ♦ But we’d prefer something automatic and dynamic...
- Simple learning 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*; update *f.src* entry in table to point to *q*
/ forward on all ports but the one where frame arrived*/*

Host	Port
A	1
B	1
C	1
D	1
W	2
X	2
Y	2
Z	2

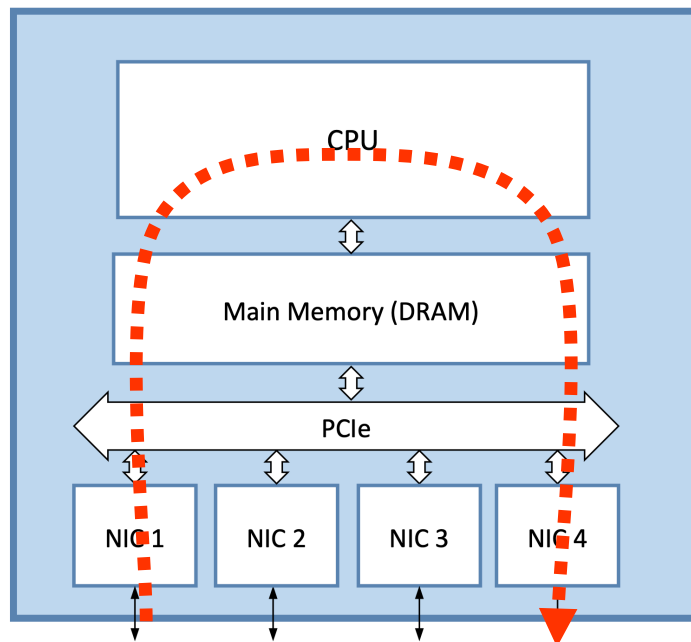
Interconnect architecture



- Input & output connected via switch fabric
- Kinds of switch fabric
 - ◆ Shared Memory
 - ◆ Bus
 - ◆ Crossbar
- How to deal with transient contention?
 - ◆ Buffering

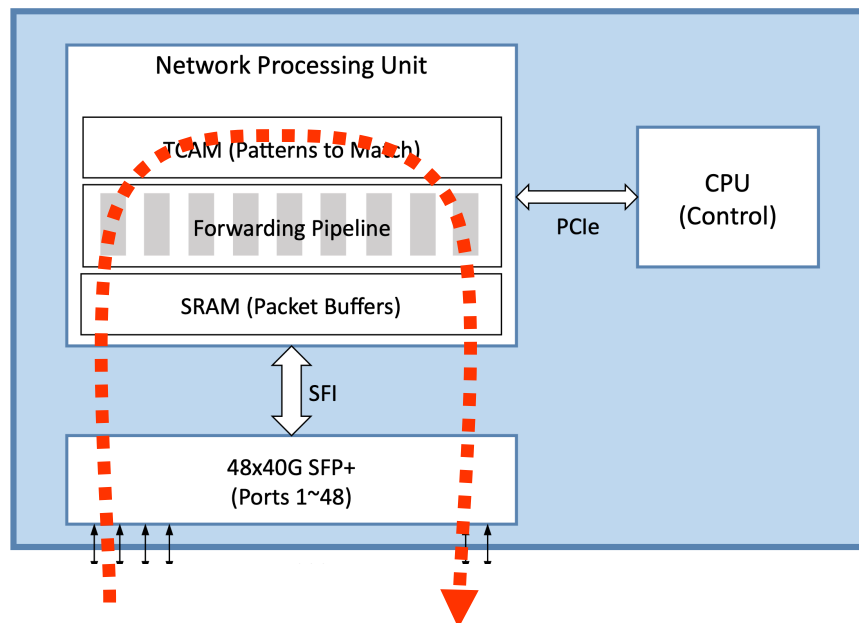


Software switch



- Single CPU and shared memory for all ports
- Classification and forwarding by main CPU
- Limited by CPU & DRAM speeds

Hardware switch

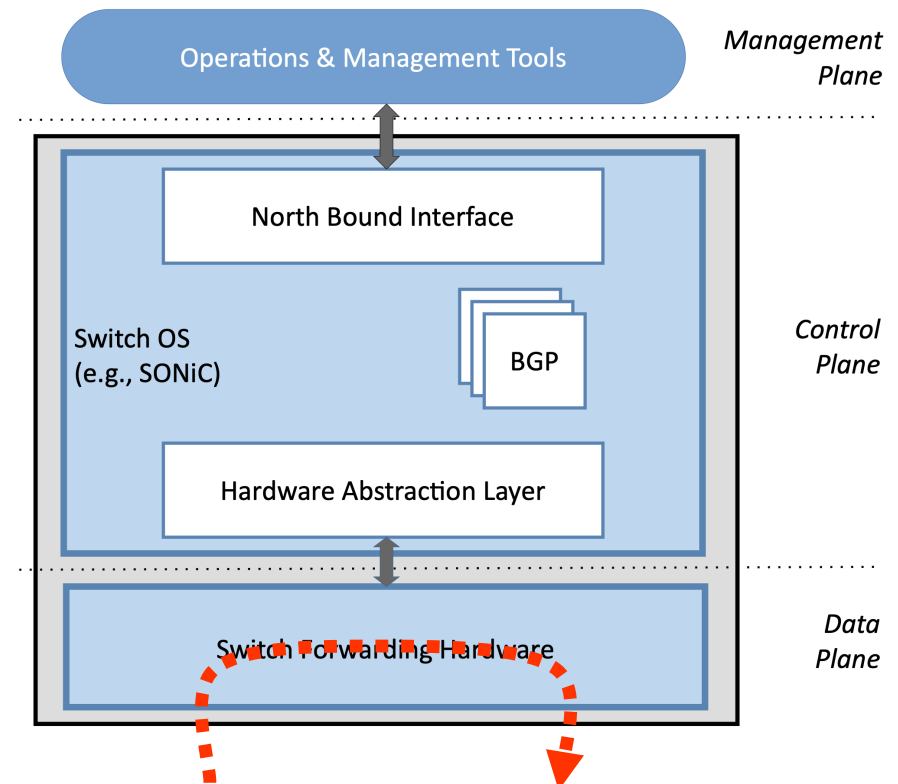


- Network Processing Unit (NPU)
 - ◆ domain-specific processor
 - ◆ optimized for processing packet headers
- Ternary CAMs
 - ◆ Content addressable memory
 - ◆ Hardware-assisted table lookups
- SRAM buffers
 - ◆ Static Random Access Memory
 - ◆ roughly an order of magnitude faster than the DRAM

Programmable switch



- Managed remotely
 - ◆ North bound interface (NBI)
- Control plan is software
 - ◆ Can be reconfigured online
 - ◆ Allows for custom features
- Forwarding still hardware
 - ◆ High-speed switching chips
 - ◆ Abstracted by vendor



For Next Time



- Discussion on P1b this afternoon
- Project 1a due tonight at 10PM
- Read P&D 6-6.2.1