06 - Switching

MAC Address

- `ipconfig` on windows, `ifconfig` on mac → `ethernet/eth` address
- 6 bytes = 48 bits, fixed to device, (as hex): `00-FF:00-FF:00-FF:00-FF:00-FF:00-FF`
- each endpoint on the device has a different MAC address → this means each device can have multiple MAC addresses
- IPv4 addr is 32 bits = 4 bytes, (as ints): `0-255.0-255.0-255.0-255`
- MAC/Eth header updated hop to hop

Structure

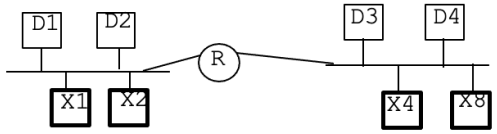
- assigned by 802 committee in DB
- first 3 bytes are vendor identifiers
- The last 3 bytes can be a destination that the vendor can assign per device

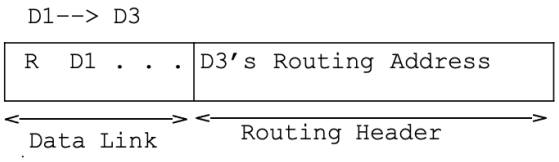
From Hubs to Switches/Bridges

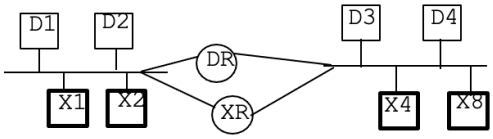
- Hubs are physical layer repeaters that broadcast the input to all endpoints on the signal → can lead to collisions
- high bandwidth cables resulted in CD support for small distances only bc of pipe size limit of 64 bytes
- so, change to switches: point-to-point, multi-input, multi-output endpoint, but only 1-to-1 signals, without collisions → separate bridge for each connection
- switches store and buffer frames and relay entire frames, not bytes

Ethernet hub limitation on bandwidth and distance

- suppose multiple stations and 2 separate LANs/ethernets, how to interconnect ethernets







and double bandwidth, distance, and stations

- cant use routers because routers at the time (1980s) were dependent on protocol: Xerox or Dack?
- using a brige/switch that buffers data, if the hub doesn't know the dest mac, just **flood** it and send to all stations on eth2 → doubles distance and stations but not bandwith

- instead, have a DB for mac addrs and switch port ids (eth1 or eth2 line in/out) → build the DB on the switch using frames src and dest macs and eth line id then you can know where the data is going and whether its going up or down (like defaultdict)
  - keys → eth line/id, values are mac addrs
  - flood if addr not in db and update db
  - remove least recently used (or some other metric) → rarely required as db is large enough
  - only condition is non-cyclic topology → any tree topology is ok
    - concurrency is not an issue because frames are buffered, so switches wait if there is data on the line with the same mac dest as the buffered data (e.g., multiple switches on a shared ether)
  - timer for buffering, if timer expires, flood → something on the order of 10 mins
  - code

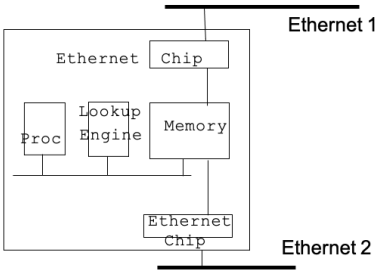
```
ReceiveFrame F on port X
    AddTable(F.Source, X) (* learn source, refresh timer *)
    Y = Lookup (F.Dest) (* lookup destination *)
    If (Y = Nil) then (* unknown destination *)
        Forward Frame F on All Ports Y!= X; (*flood*)
    Else if (Y != X) then
        Forward Frame F on Port Y
    Else if Y = X (* filtering *)
        Drop frame F
    (* received on same port as source *)

TimerExpiry (E)          (* timer for entry E fires *)
    E.port = Nil;                (* reenable flooding *)
```

Terms

- transparency - switches must be transparent to stations, stations should not care whether the switch exists or have knowledge abt it
- promiscuous receive - switches buffer from all stations regardless of src
- flooding - packets forwarded to all stations on the line and picked up by correct mac
- filtering - switches can filter packets based on headers to filter forwarding or buffering

Design



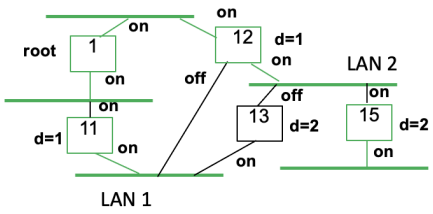
1) 3 bus design to avoid congestion on processor bus

2)4 ported memory design.

3)Hardware binary search lookup engine. Takes
log(8000) = 13 memory accesses of 100 nsec each = 1.3 usec

4)Processor stays in loop after a packet interrupt servicing as many packets as arriive to reduce context switching overhead.

Routing - Spanning Tree



- Root is Min ID node (in this case Bridge 1)
- Other bridges finds Min port, port through which it has shortest path to root (parent), For 11 it is upper port.
- Each bridge also finds the ports for which this bridge is on the shortest path between root and corresponding LAN: Designated Ports. For example, 11 and 12 have d= 2 for LAN 1, so we pick shorter ID as tiebreaker, Bridge 11 is designated bridge for LAN 1, 12 for LAN 2
- Each bridge turns ON Min port and all Designated Ports. ON,OFF are software states: always receive hello and management messages on all ports. Drop data packets to/from OFF port.

13

- every bridge enables parent port and

Autoconfiguration (Multicast)

- based on multi-cast addressing
- if MSB of MAC addr is 1 → multi-cast addr (if multiple 1s in MSBs → broadcast addr)
- multi-cast - sender broadcasts signal to a subset of stations (instead of single)
  - multiple like-kind servers (e.g., file servers) and 1 responds (note requires src mac addr in eth header)

- vendors get 2^{24} unicast and multicast addresses (they get assigned 2, 3 byte venddor identifiers, one with leading 0 bit for unicast and one with leading 1 for multicast addrs)
- switches just flood if dest is multicast addr