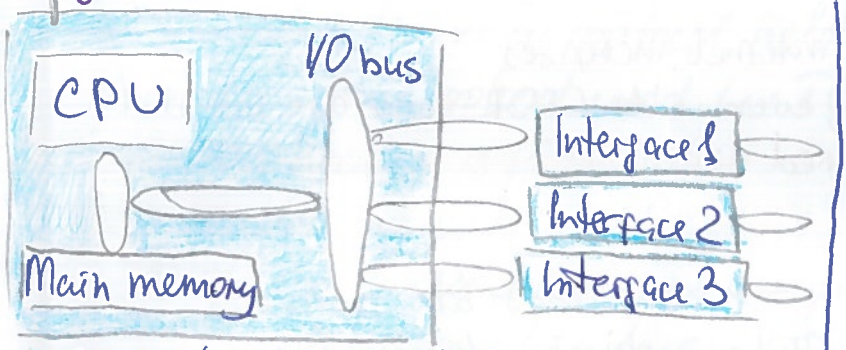
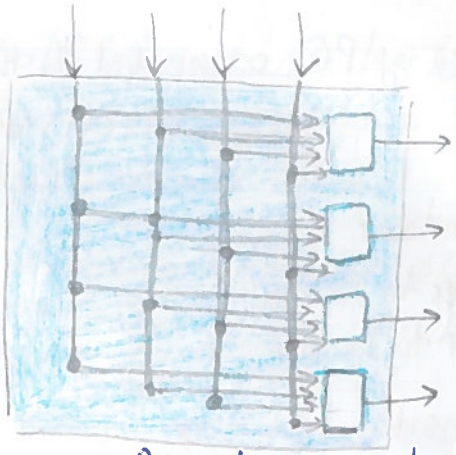
Lecture 10 — Network Layer

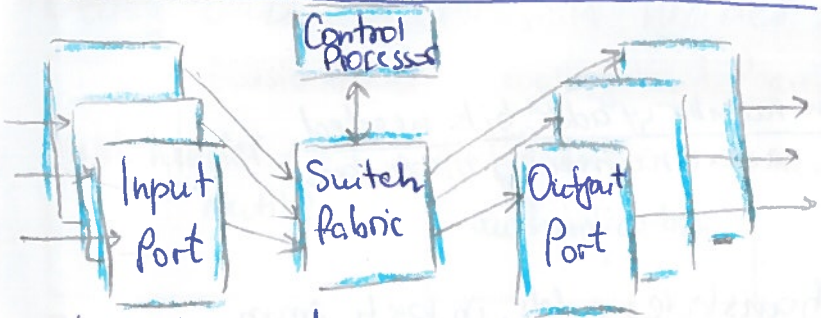
- Figures



Router/switch on PC

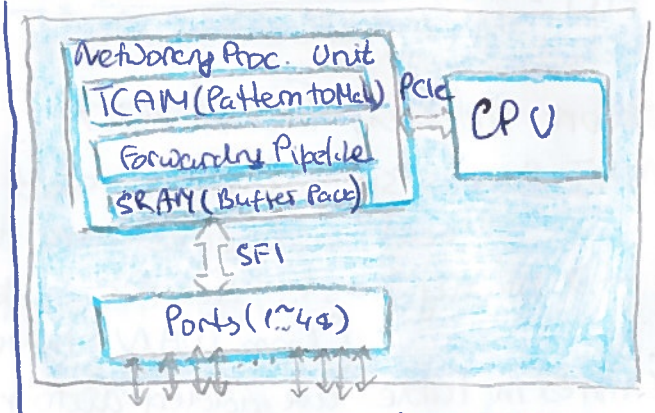


Crossbar switch

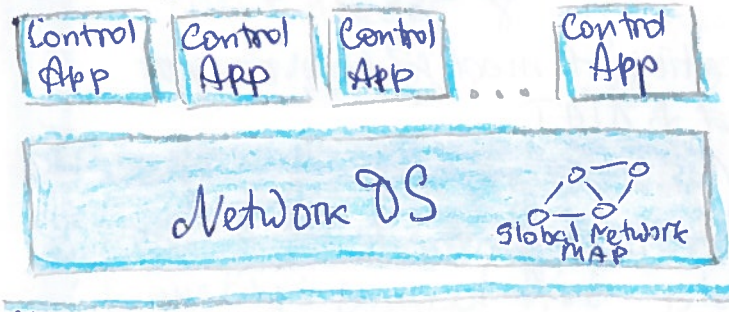


- How to build switch fabric?
  - shared bus
  - shared memory
  - crossbar
  - self-routing fabric

Router/switch architecture



Generic modern hardware switch

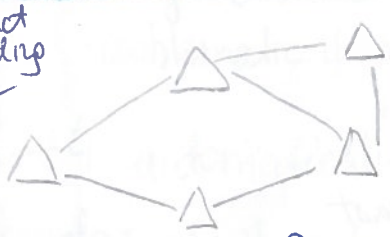


Control Plane

Data Plane

- Switch can't keep up making decisions if packets are small!
- "Difficult" cases are not handled by input port hardware but by CPU in software, which is much slower. (fragmentation, TTL, IPV6)

Abstract Forwarding Model



SDN: Software-Defined Networking

- Reasons
  - number of PCs connected to the internet increases
  - not all addresses can be used, even with CIDR some are wasted due to structured assignment
- Constraints:
  - in order to not run out of businesses → use almost all addresses
  - in order to keep forwarding tables small → structured assignment even if to waste many addresses

- Host-Density Ratio

$$HD = \frac{\log(\#\text{ allocated objects})}{\log(\#\text{ of allocatable objs})} = \frac{\text{min number of addr. bits needed}}{\text{actual number of addr. bits}}$$

- Network Address Translation
  - NAT Box is special router that translates addr. in pkts from "local" addr of network to "real" internet add, and back
    → many hosts share a single IPv4 addr (from WAN (Internet) to LAN and back)
  - Entries in table are added automatically based on outgoing connection requests
  - Disadvantages:
    - Breaks end-to-end transparency
    - NAT Box needs to know about applications, to modify pkts, or apps need to be adapted to NAT
    - Incoming connections are problematic (also security advantage!)
    - routers should not look at the higher-level protocols
- IPv6
  - larger addr space
  - reduce forwarding table
  - simplified header (faster processing)
  - better support for mobility, QoS, security, multicast
  - Header format:
    - Version 4
    - Traffic class 8
    - flow 20
    - payload length 16
    - next header 8
    - hop limit 8
    - src addr 32
    - dest addr 32
    - payload
  - Notation: 8 groups 16 bits, each as 4 hex.

if no extension headers:
- next header (same as protocol field in IPv4) indicats higher-level protocol used (as TCP/UDP/ICMP)

if extension header is used:
next header indicats what type it is. Extension header then contains a new Next header field, to specify more extension headers or higher layer protocol.

Daisy-chain of extension header:

basic header	routing header	authentication header	
next header = routing	next header = authentication	next header = TCP	TCP segment

Fragmentation
- no longer done by routers, but only by end-hosts.
- if pkt too long → router sends back ICMP msg.
  src host then resends the pkt in small fragments, using extension header:
  - next header 8
  - reserved 8
  - fragment offset 13
  - res 2
  - more 1
  - identification 32

Transition from IPv4 to IPv6
- Dual-attach: hosts have both IPv4/v6 capabilities (ISPs)
- Tunneling: connect IPv6 "islands" by encapsulating IPv6 pkt to IPv4 pkt
  - problematic if many island networks want to connect
- 6to4: automatically assign IPv6 addr to island networks and setup tunnel
- Teredo: similar, but works also behind NAT
- NAT64: allows IPv6 hosts to access IPv4 servers via special NAT

- How to proceed when IPv4 addr run out?
  - use regular NAT to share a single IPv4 addr among many users (carrier-grade NAT)
    - already exists (home ISP)
    - only partial solution
  - use something that provides IPv4 connectivity to IPv6 (NAT64)
    - only partial solution
  - try to reclaim inefficiently used IPv4 addr blocks
  - create a market place for IPv4 addr (scarce resources are worth money)