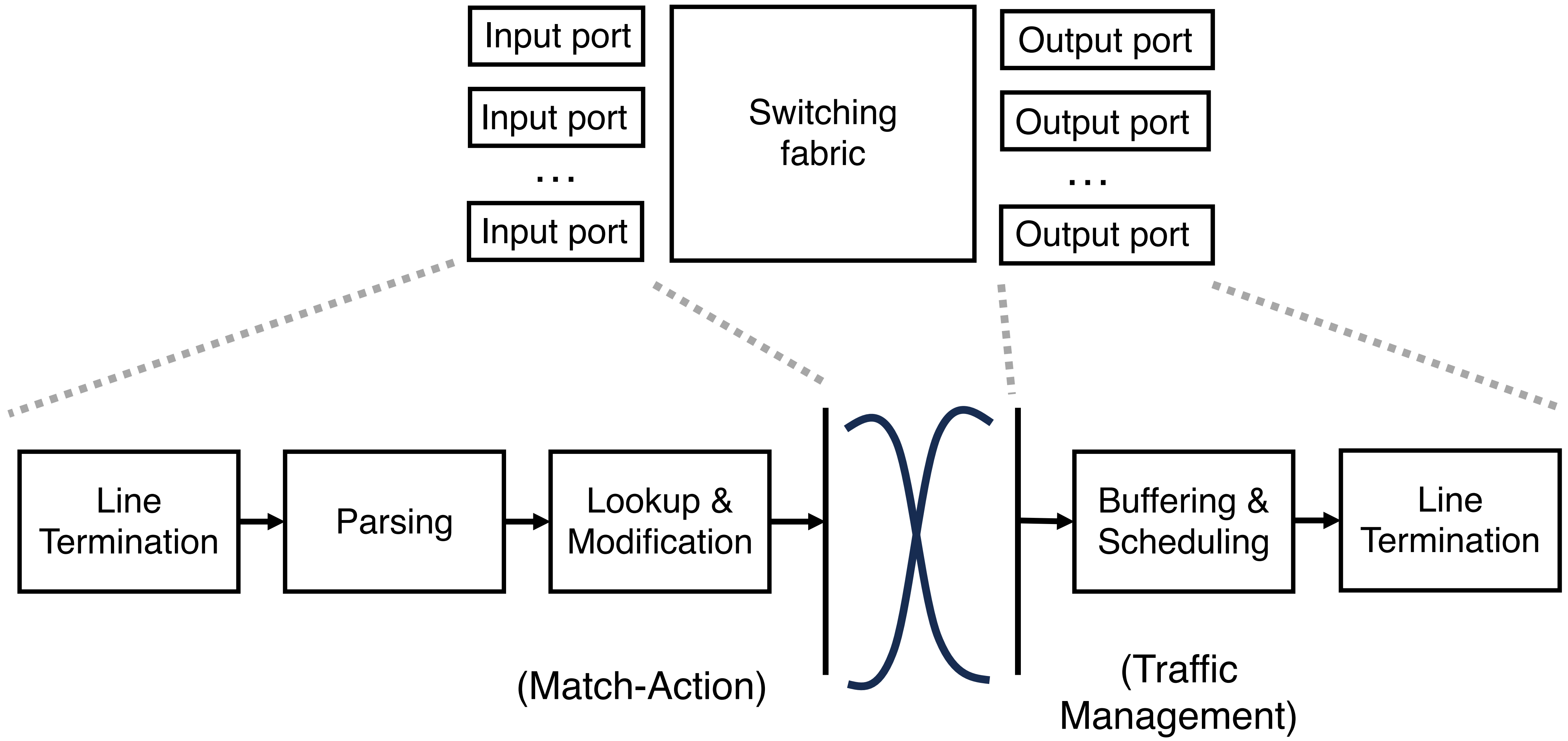
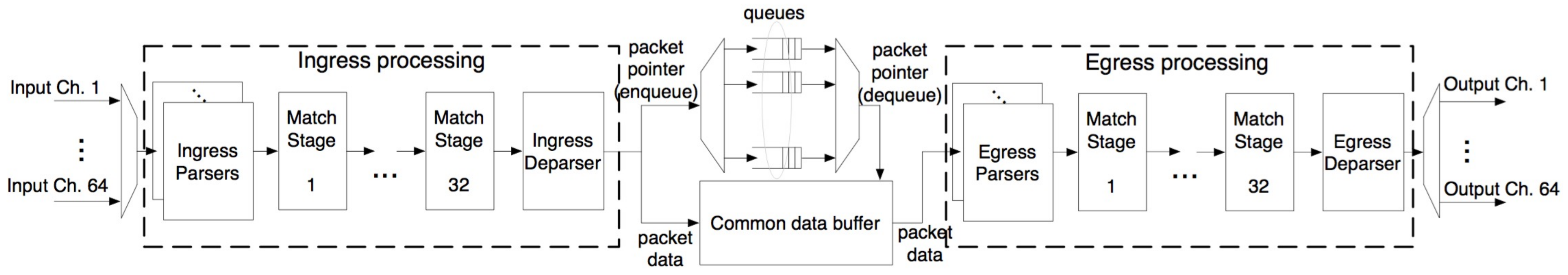
Network

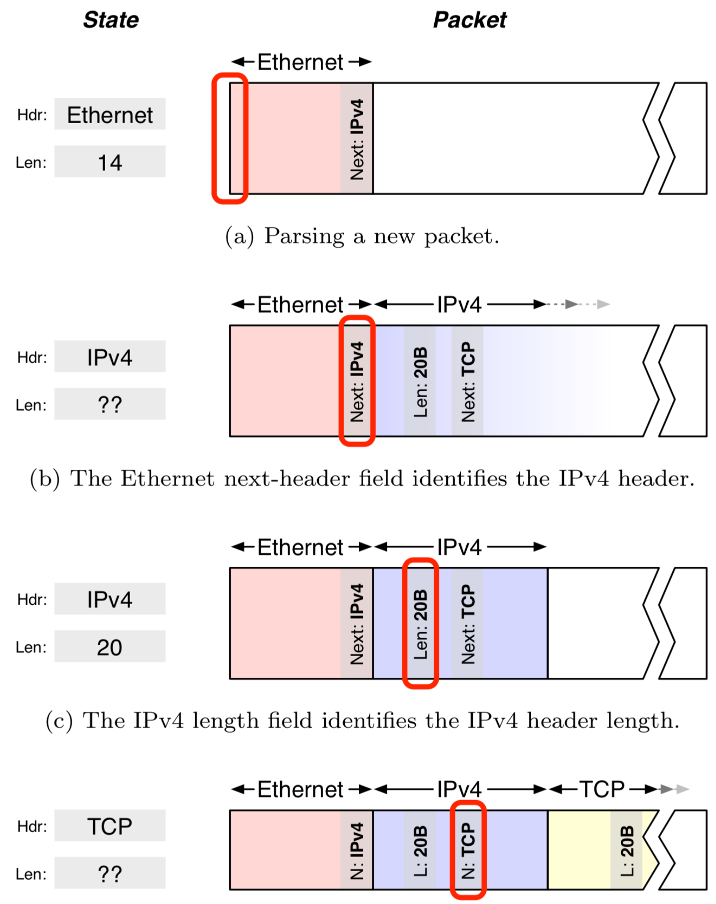
RUTGERS
UNIVERSITY | NEW BRUNSWICK

Life of a packet in a hardware router





Pipelined Parallelism



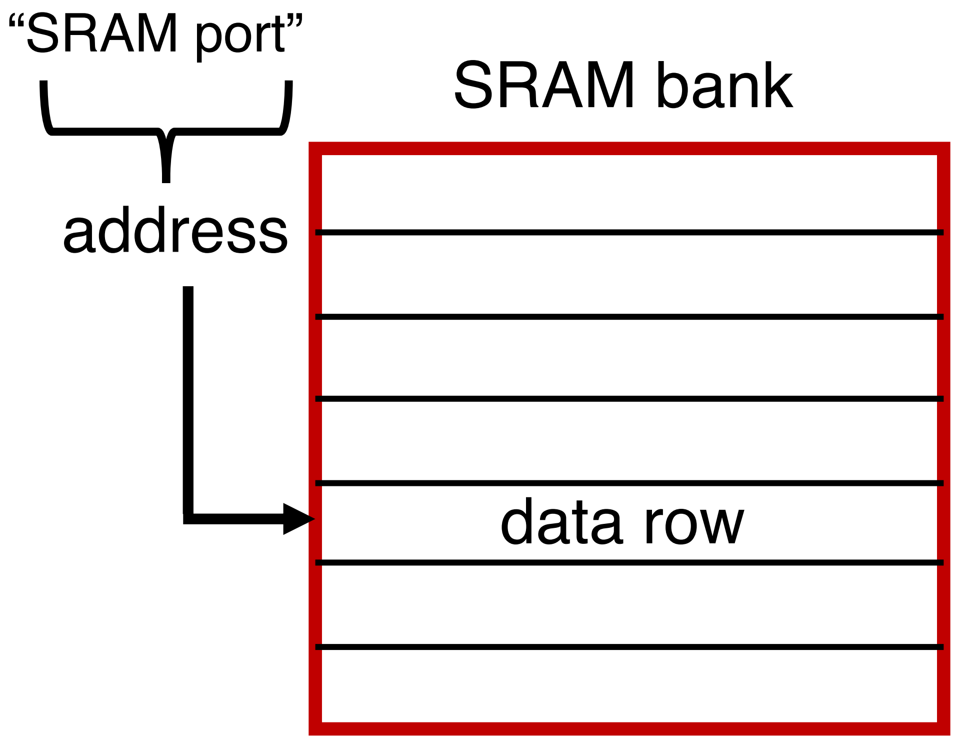
(a) Parsing a new packet.

(b) The Ethernet next-header field identifies the IPv4 header.

(c) The IPv4 length field identifies the IPv4 header length.

(d) The IPv4 next-header field identifies the TCP header.

Packet Header Vector

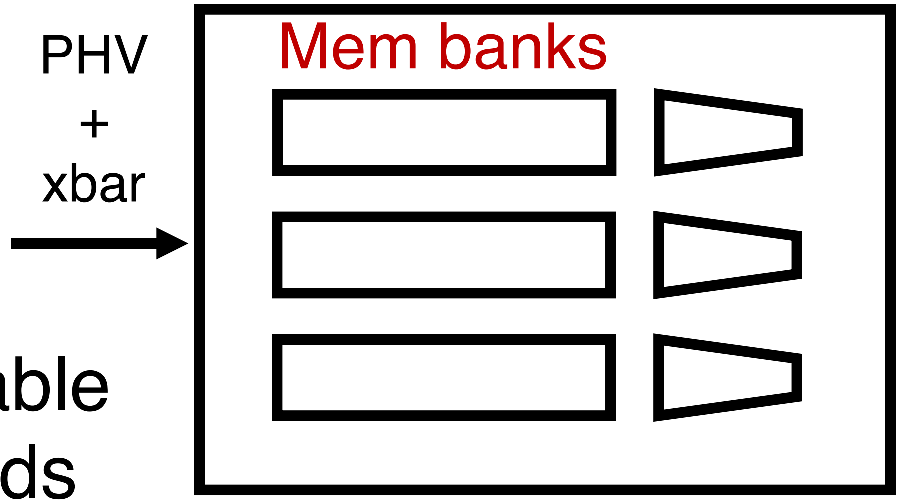


RMT: Match-Action Table memory design

- Match and Action units supplied with the Packet Header Vector
- Each stage accesses its own memory containing tables

- RMT uses a crossbar to pick PHV fields for matching against contents of SRAM/TCAM banks
  - Flexible key generation for lookup

- Distinction from fixed-function:
  - User-programmed fields stored in table
  - PHVs include user-programmed fields
  - Table match keys chosen by users

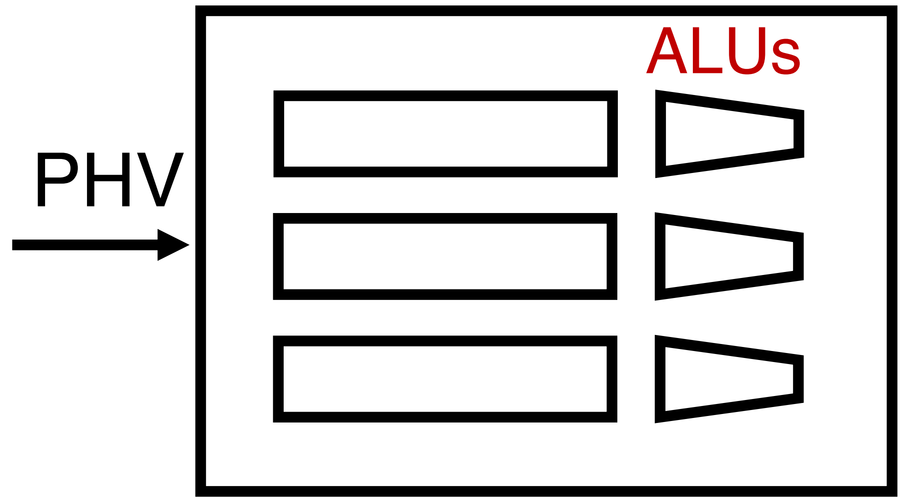


RMT: Match-Action Table memory design

- Entry looked up in the memory (SRAM/TCAM) contains a pointer to the instruction (action) for that entry

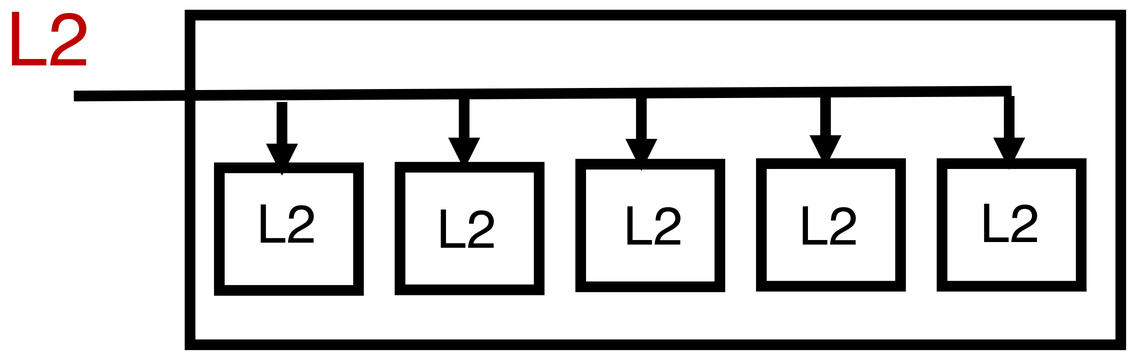
- Instructions implemented through programmable ALUs
  - More general ALUs than fixed-function hardware devices
- Feasible since compute components “cheap” inside a router
- VLIW: operate on multiple headers simultaneously

- Entry also contains pointers to:
  - Action data (e.g., output port),
  - statistics (counters), if needed

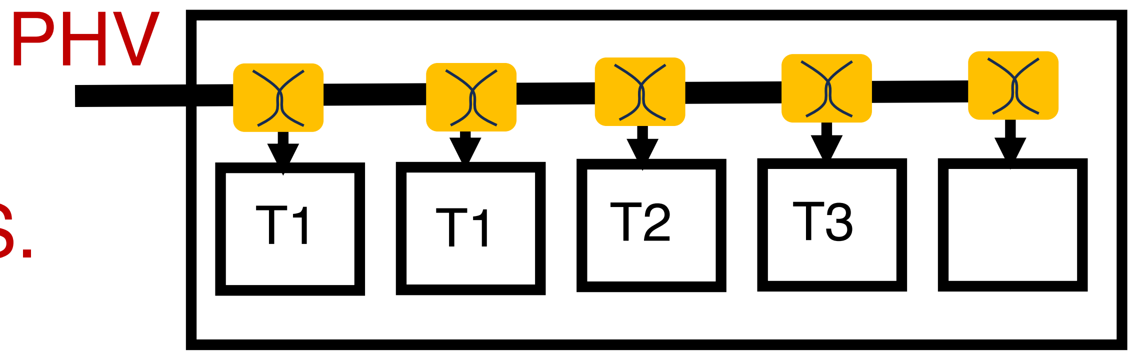


Achieving reconfigurable match-action

Separately configurable and addressable memory blocks is the key to using tables flexibly. Independent block level access



VS.



Cost: 14% extra area (& power) for fatter wires

Fixed-function matching

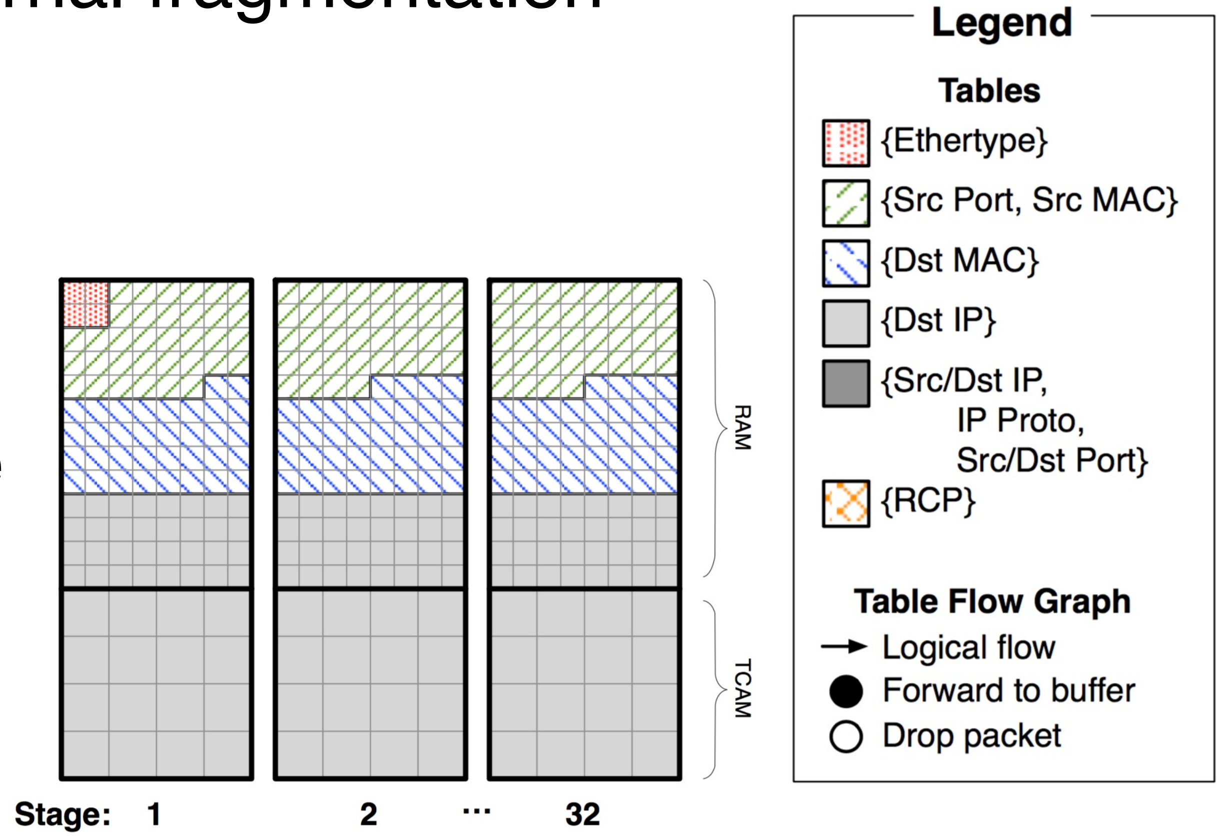
Flexible match function

Each table in a match-action stage may use a different crossbar configuration to extract a different set of fields from the PHV

Different memory banks can contain entries for different tables (e.g., IPv4 matching, virtualization, …)

Memory layout and use matters

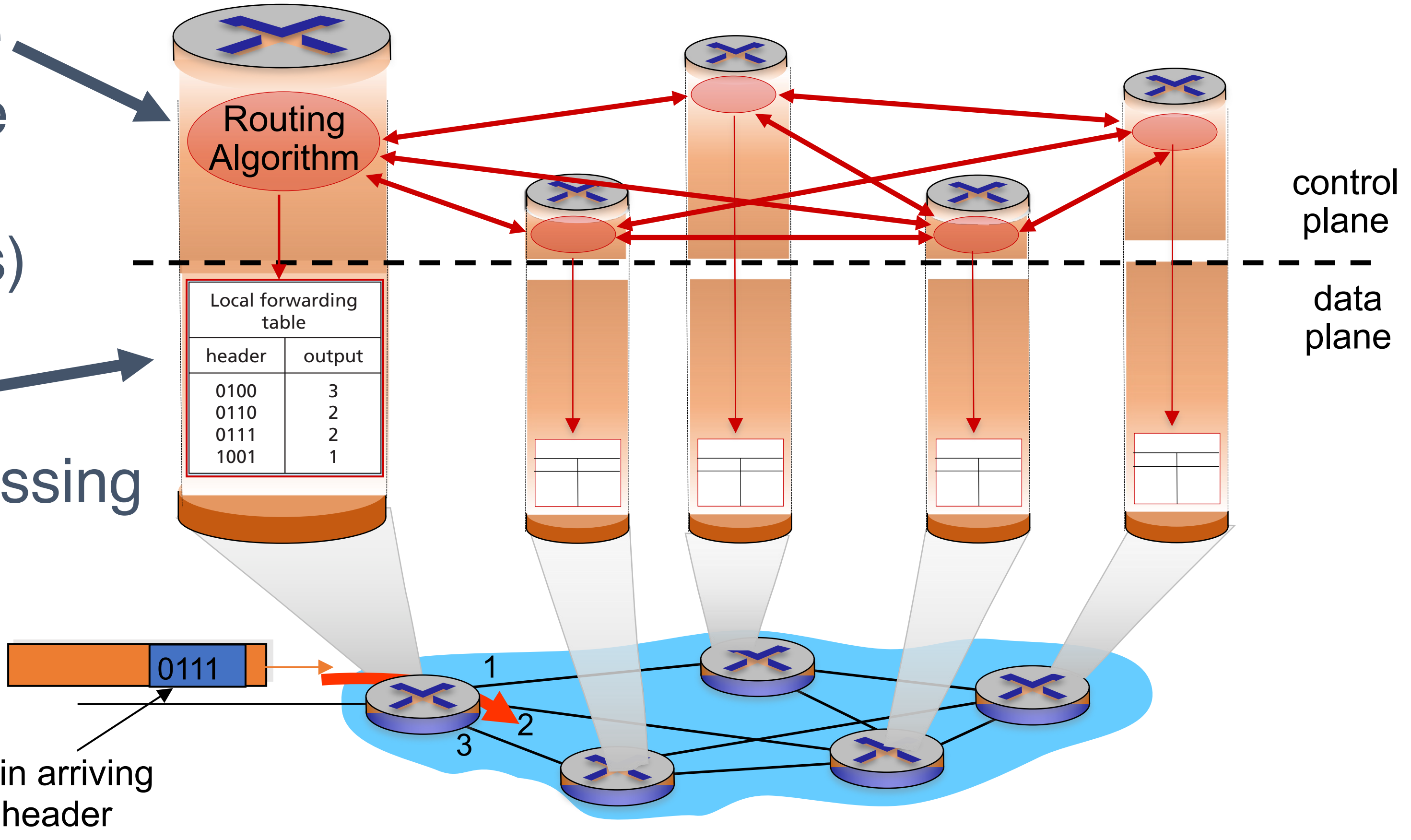
- Flexible partitioning of memory across SRAM and TCAM
- Fixed size memory blocks: internal fragmentation
- Deterministic access times
  - All of it is SRAM or TCAM
- Interesting compiler concerns
  - “Packing” tables
  - Within & across pipeline stage



Who programs the rules? Control plane

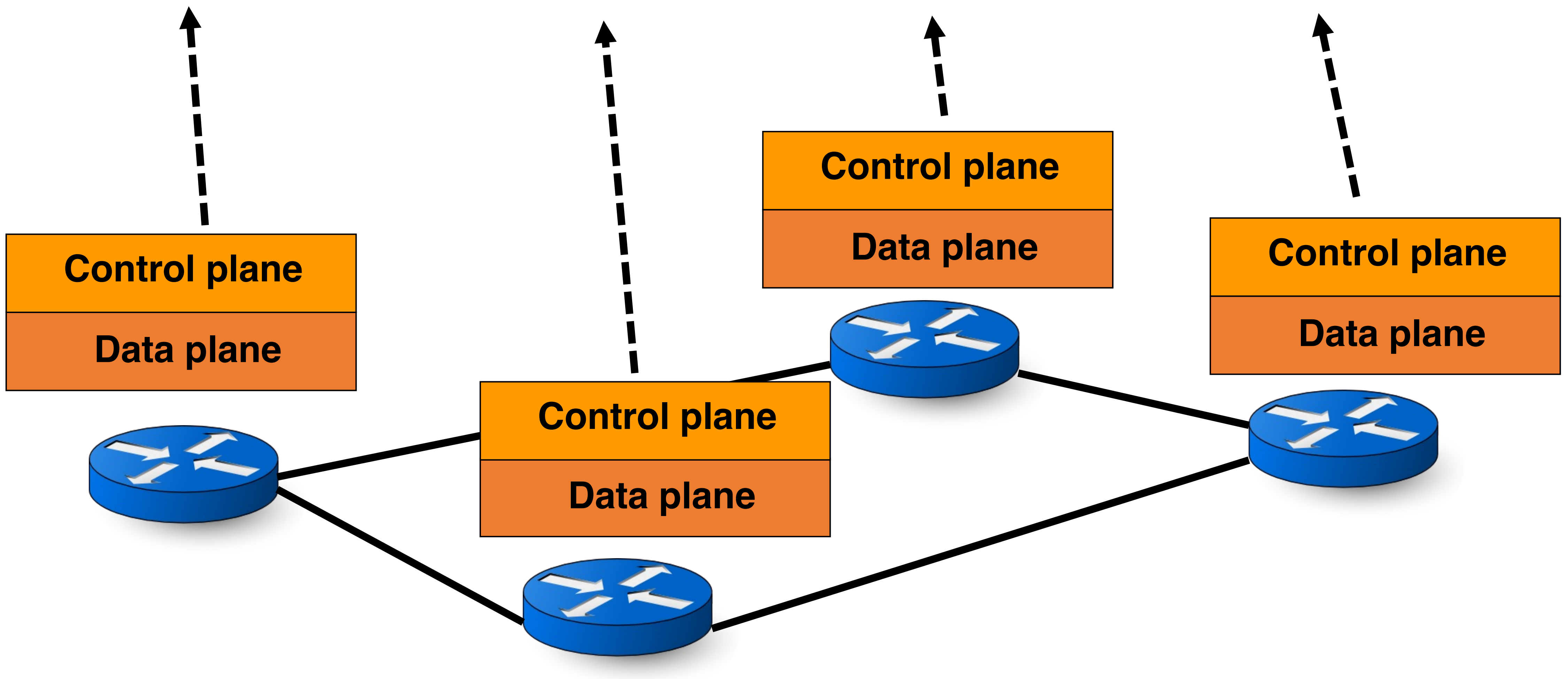
Control plane
per route-change processing
(~ a few seconds)

Data plane
per-packet processing
(~ tens of nanoseconds)

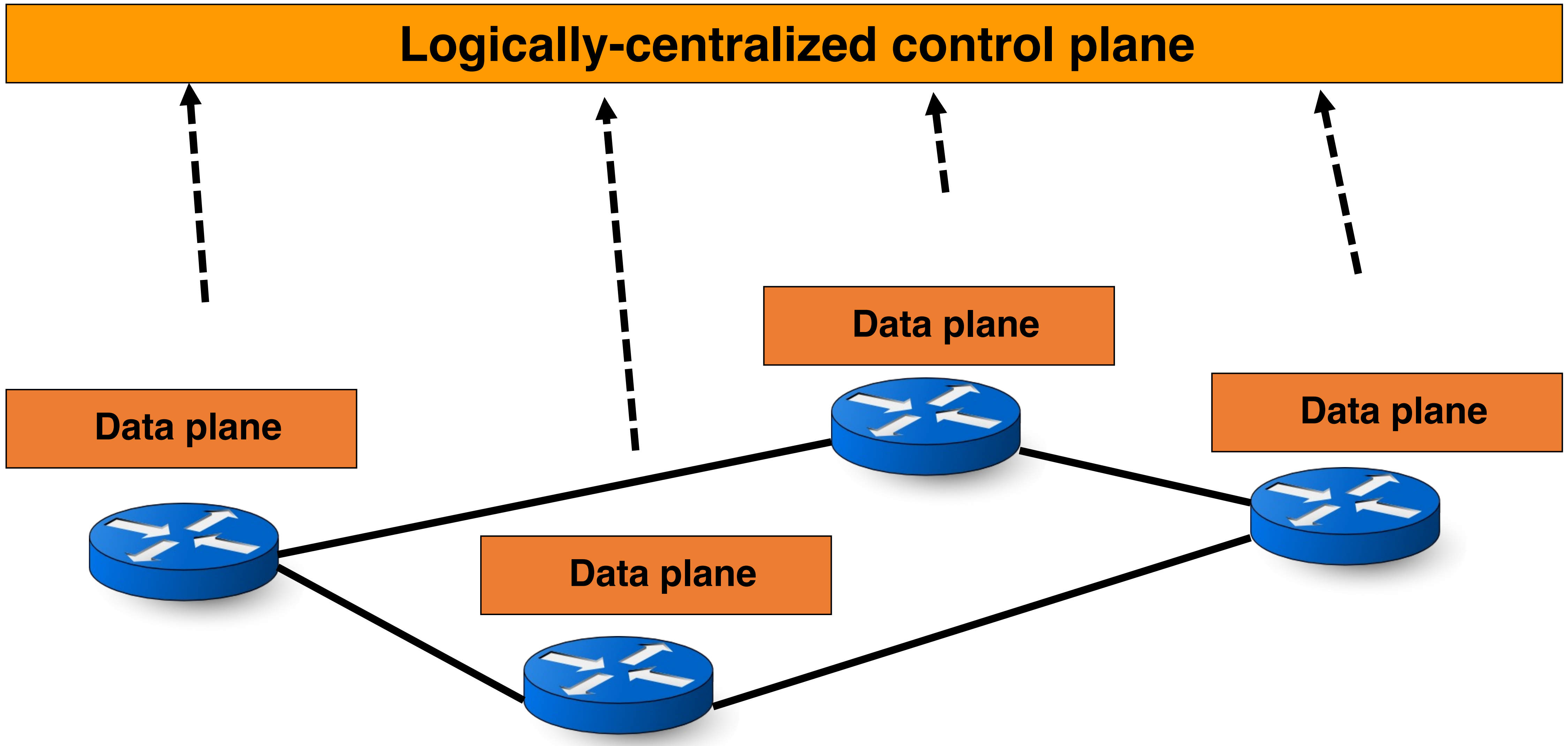


header	output
0100	3
0110	2
0111	2
1001	1

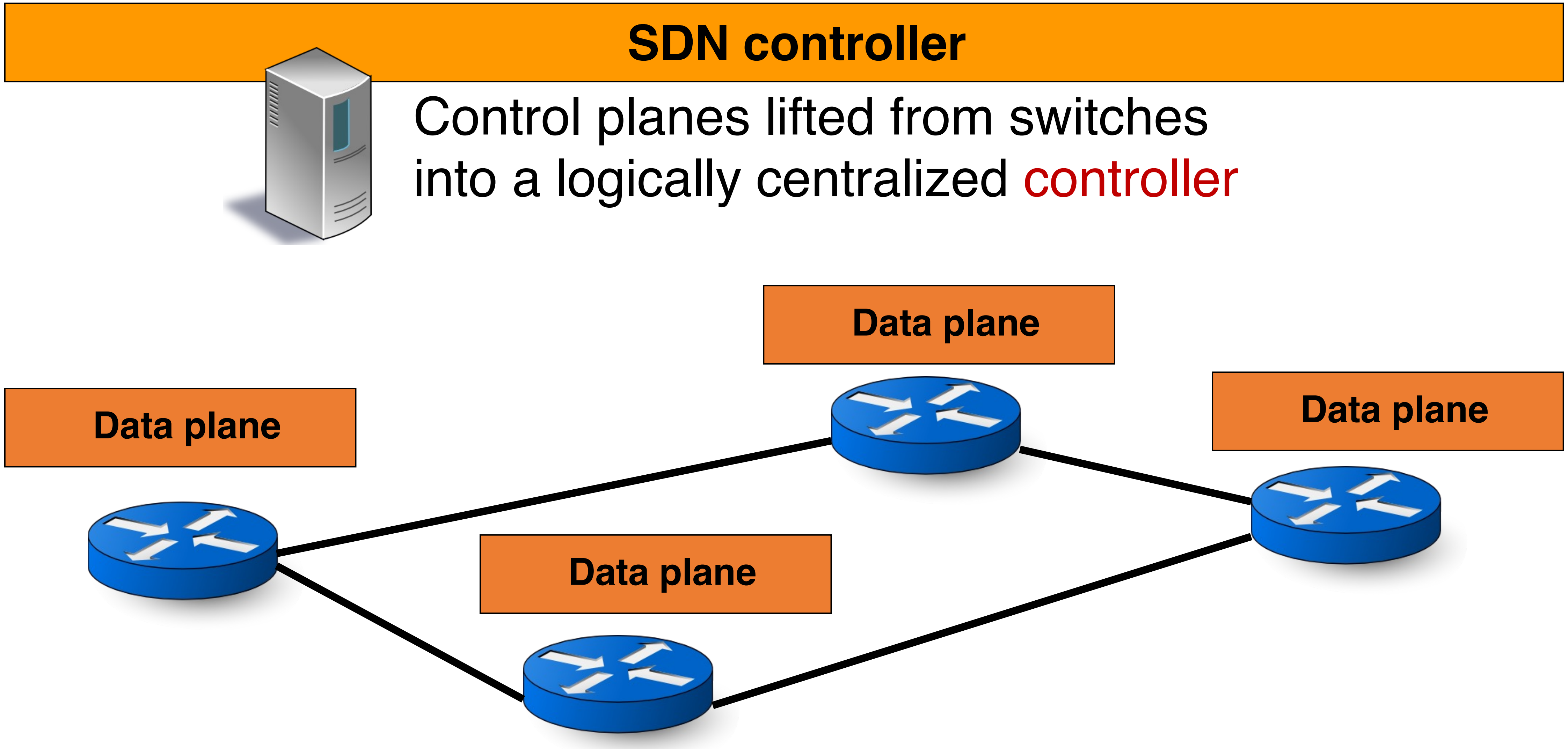
Distributed control planes



Software-defined network (SDN)

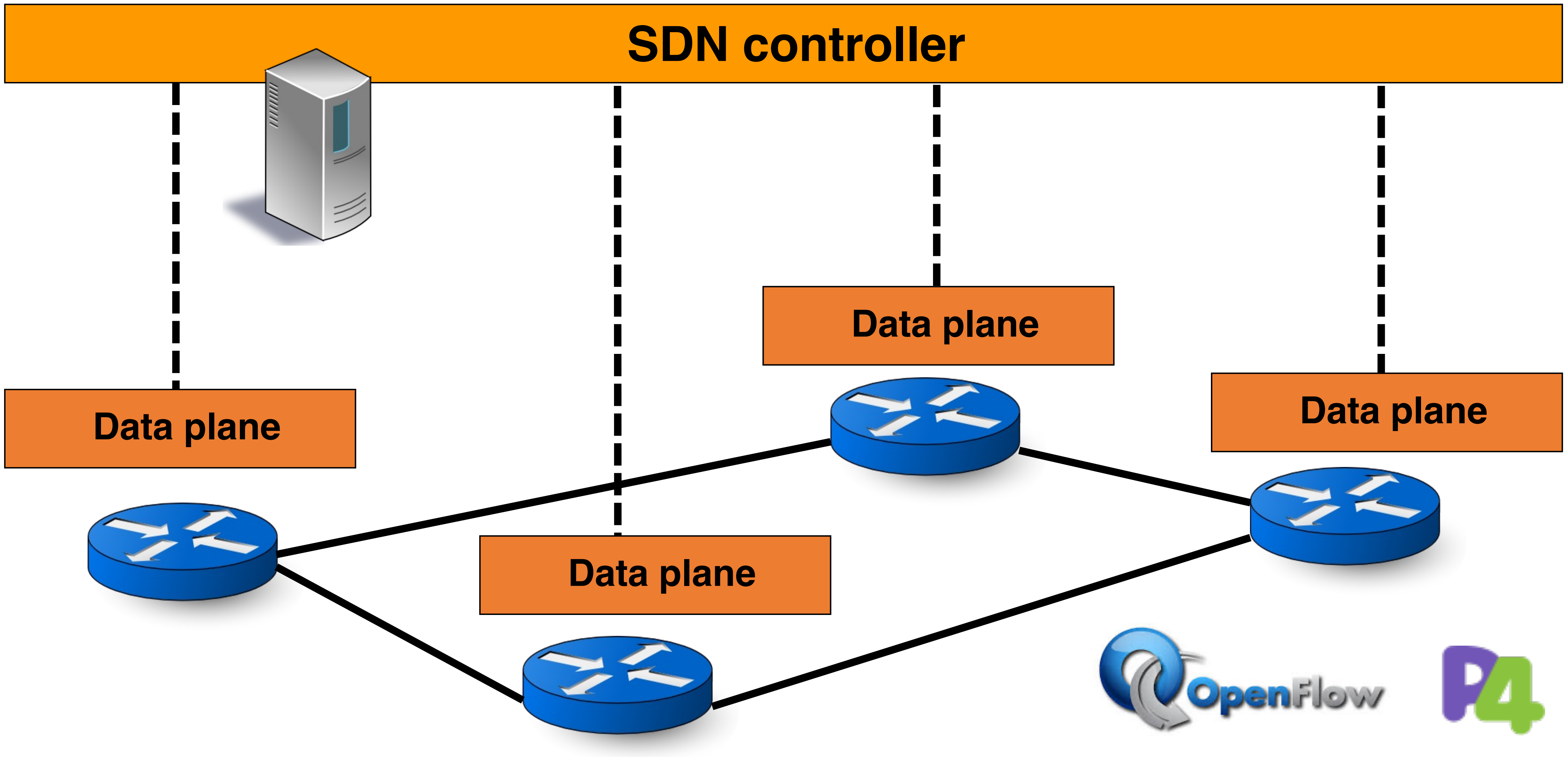


SDN (1/2): Centralized control plane



Control planes lifted from switches
into a logically centralized controller

SDN (2/2): Open interface to data plane



Packet after route lookup

- What we have after lookup and modifications:
  - One or more output ports, or a decision to drop
  - packet headers, possibly modified from ingress

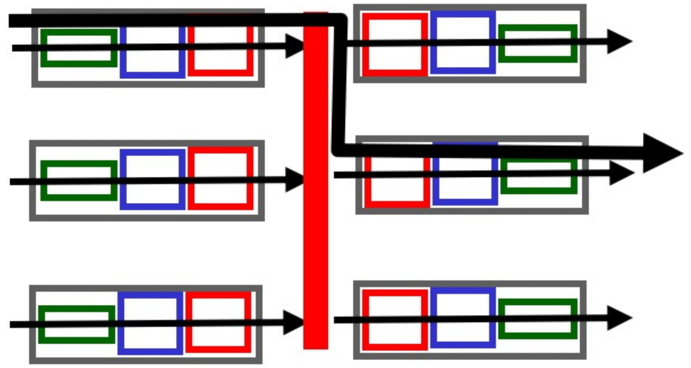
- Goal: reconstitute the headers with payload and send the packet out of one or more output ports

- Move the packet (header) to output through the switching fabric

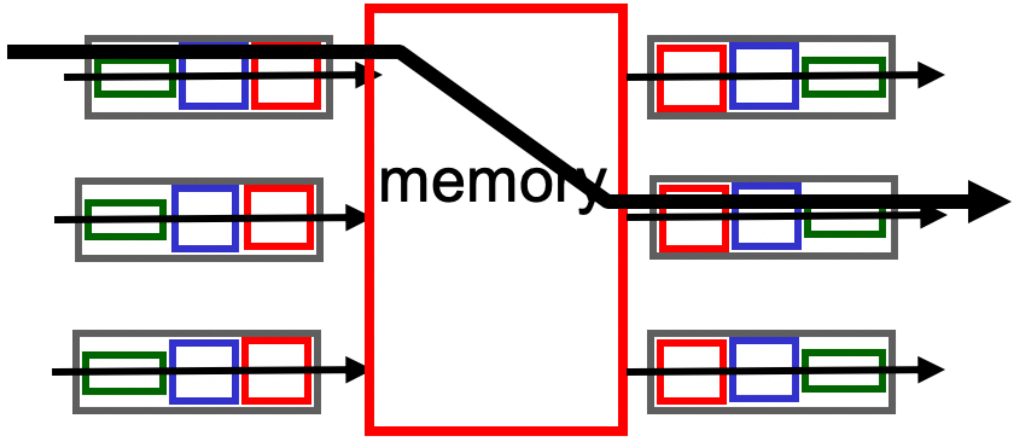
Building a high-speed switching fabric

A single (n X m)-port switching fabric

- Different designs of switching fabric possible
- Assume n ingress ports and m egress ports, half duplex links



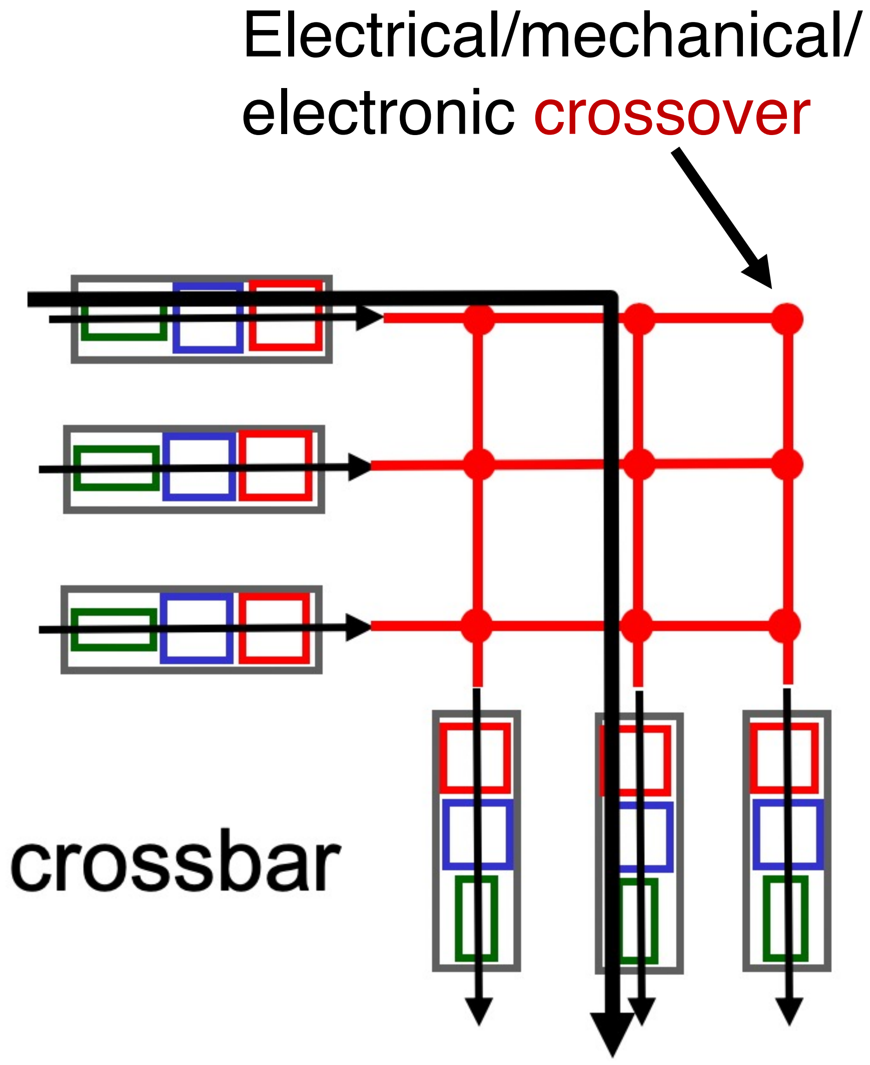
bus



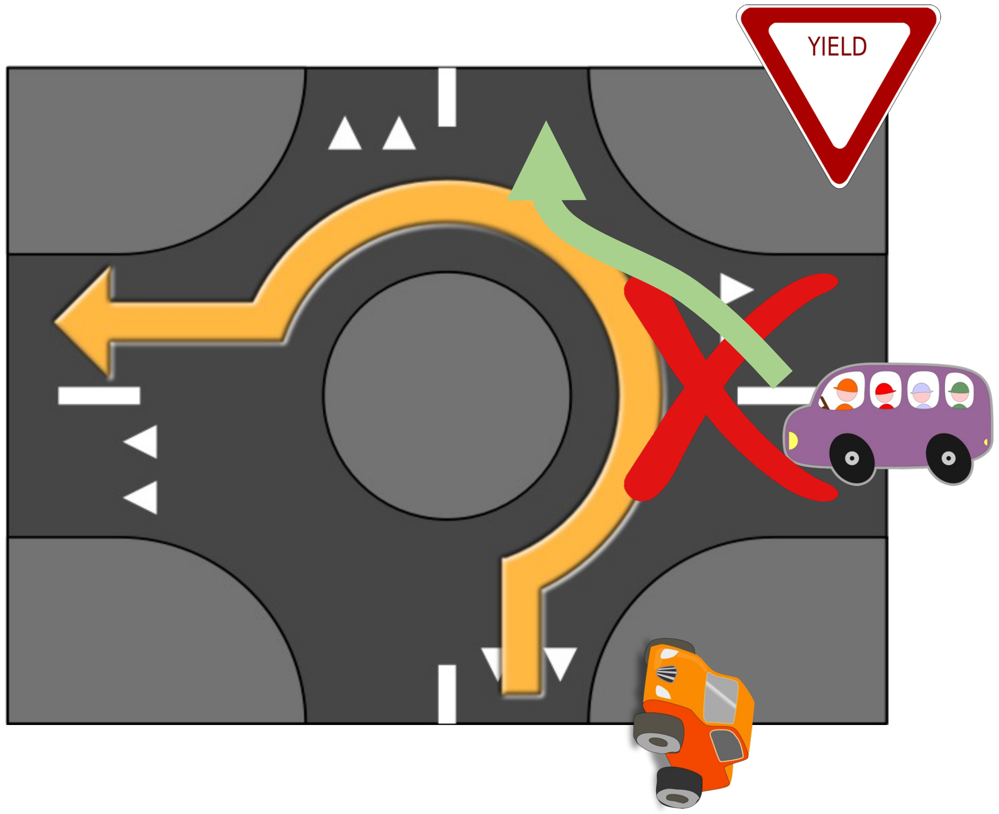
memory

A single (n X m)-port switching fabric

- We want a design such that:
- Any port can connect to any other directly if all other ports free
- Nonblocking: if input port x and output port y are both free, they should be able to connect
  - Regardless of other ports being connected.
  - If not satisfied, switch is *blocking.*



Nonblocking designs are nontrivial



Two aspects: topology and routing

High port density + nonblocking == hard!

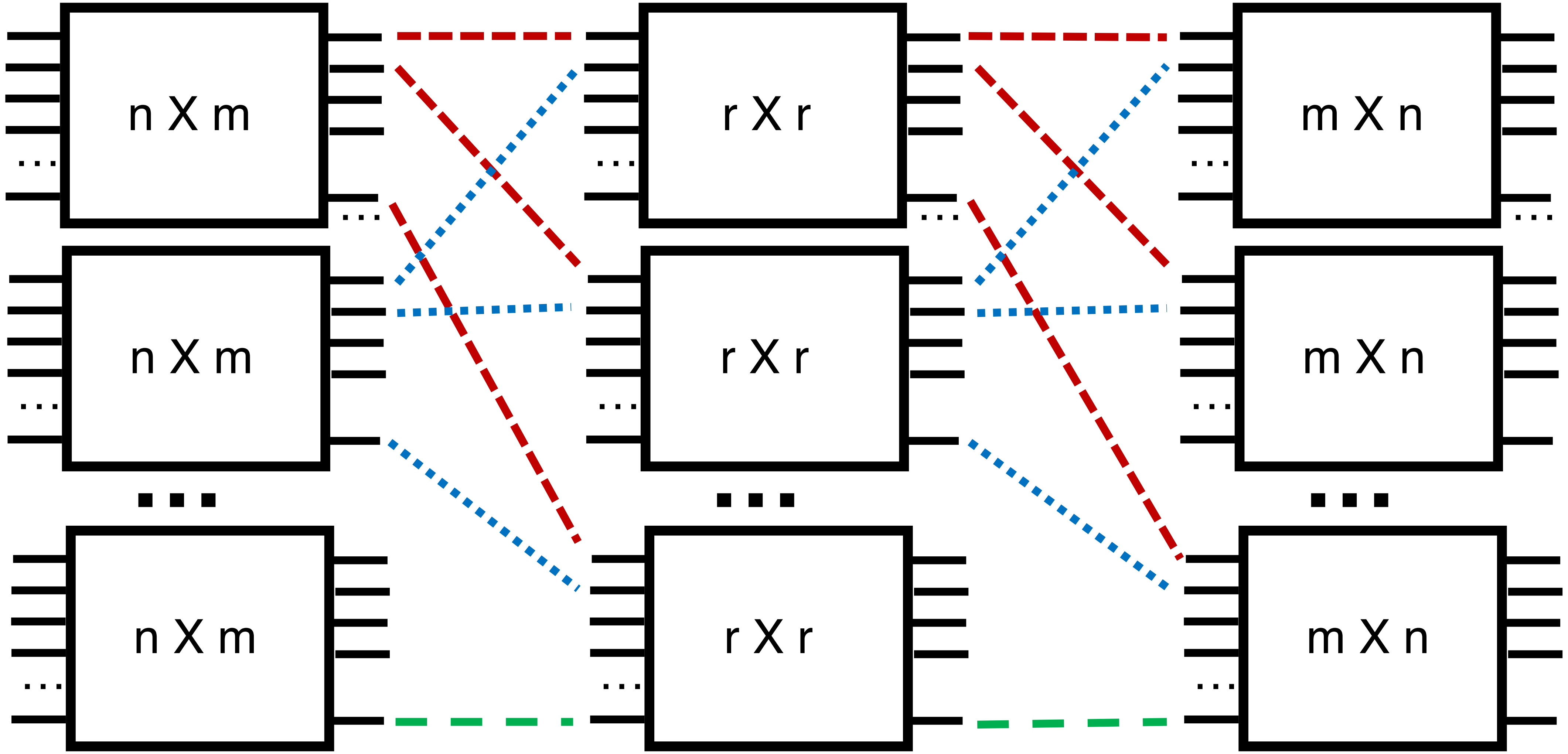
- Low-cost nonblocking crossbars are feasible for small # ports
- However, it is costly to be nonblocking with many ports
- If each crossover is as fast as each input port,
  - Number of crossover points == n * m
  - Cost grows quadratically on the number of input ports
- Else, crossover must transition faster than the port
  - … so that you can keep the number of crossovers small

Nonblocking switches with many ports

- Key principle: Build a fast, nonblocking switch with many ports using many fast, nonblocking switches with a small number of ports.

- How to build large nonblocking switches?
  - The subject of interconnection networks
  - https://en.wikipedia.org/wiki/Multistage_interconnection_networks

3-stage Clos network (r*n X r*n ports)



r switches n X m

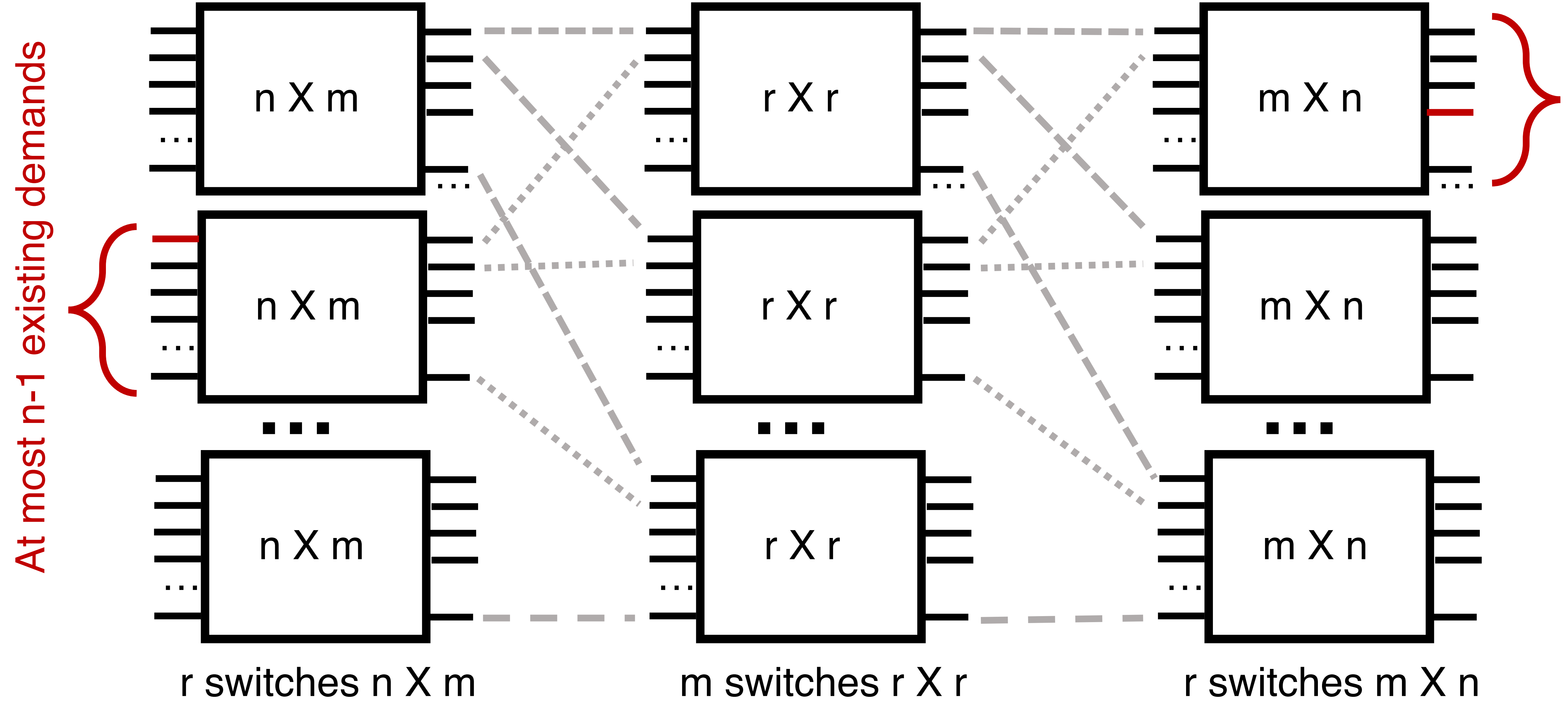
m switches r X r

r switches m X n

How Clos networks become nonblocking

- Adjacent layers of Clos are fully connected with each other
- if $m > 2n - 2$, then the Clos network is strict-sense nonblocking
  - Cost: you need more output ports than input ports
- That is, any new demand between any pair of free (input, output) ports can be satisfied without re-routing any of the existing demands.

Need at most (n-1)+(n-1) middle stage



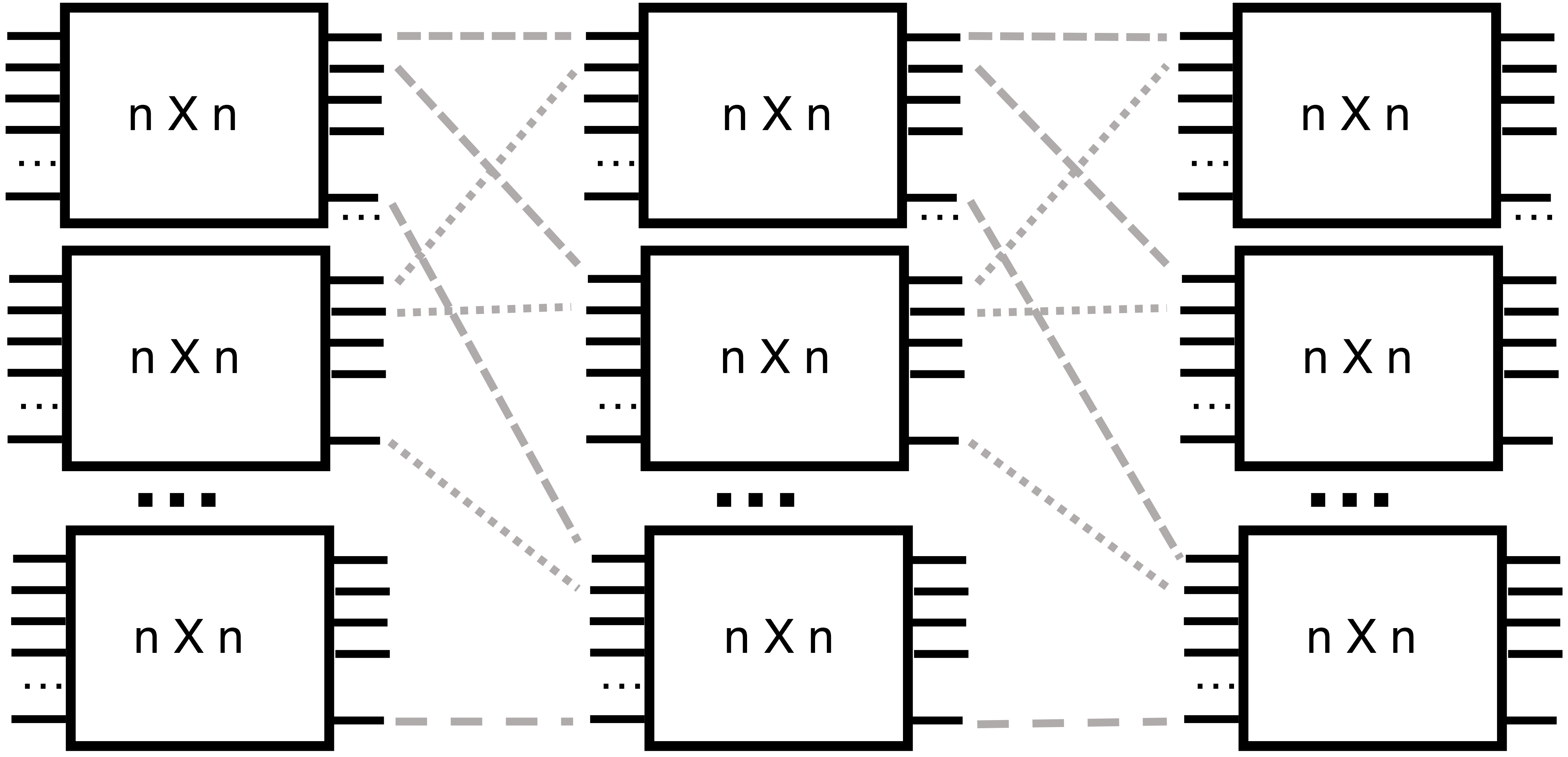
Surprising result about Clos networks

- if m >= n, then the Clos network is rearrangeably nonblocking

- Any new demand between a pair of free (input, output) ports can be satisfied by suitably re-arranging existing demands.
  - Re-arranging == re-routing

- It is easy to see that m >= n is necessary
  - The surprising part is that m >= n is sufficient
  - Intuition: matching in a fully connected bipartite graph

Rearrangeably nonblocking Clos built with identical switches: $n^2 X n^2$ using 3n nXn



n switches n X n

n switches n X n

n switches n X n

What about re-routing?

- How to rearrange existing demands when a new packet arrives, so that it can get across as quickly as possible?
- Ideally, do it without “interference” to (i.e.: rerouting) other pkts
  - In general, it is not possible. Packets must wait, adding to delays.
- But, doable with high probability based on the input traffic workload: if packets are “randomly shuffled” across output ports
- Valiant Load Balancing

Valiant Load Balancing (VLB)

- Suppose you had many paths to get to a destination
- Key idea: pick a random node to redirect a message to from the source, then follow a deterministically-chosen shortest path to the destination from there
- Guarantee: With high probability, the message reaches its destination “very quickly” without blocking (log n steps in a n-hypercube)
- Practically, this means close to zero queueing delays in switch
- Randomized algorithms can be powerful.

VLB and Clos

- VLB is more general than Clos
    - It is a form of oblivious routing
    - e.g., no need to measure traffic patterns before choosing routes
    - Extremely simple to implement: no global state
- VLB is handy in Clos topologies due to the numerous options to pick the intermediate layer switch
    - Balance load across many paths
- Very beneficial in practice
    - "Performance isolation:" other port – port flows don't matter
    - High capacity ("bisection bandwidth") between two ports