UNIVERSITY OF WATERLOO

CS 856: Programmable Networks

Mina Tahmasbi Arashloo

Winter 2025

How can we make it better?

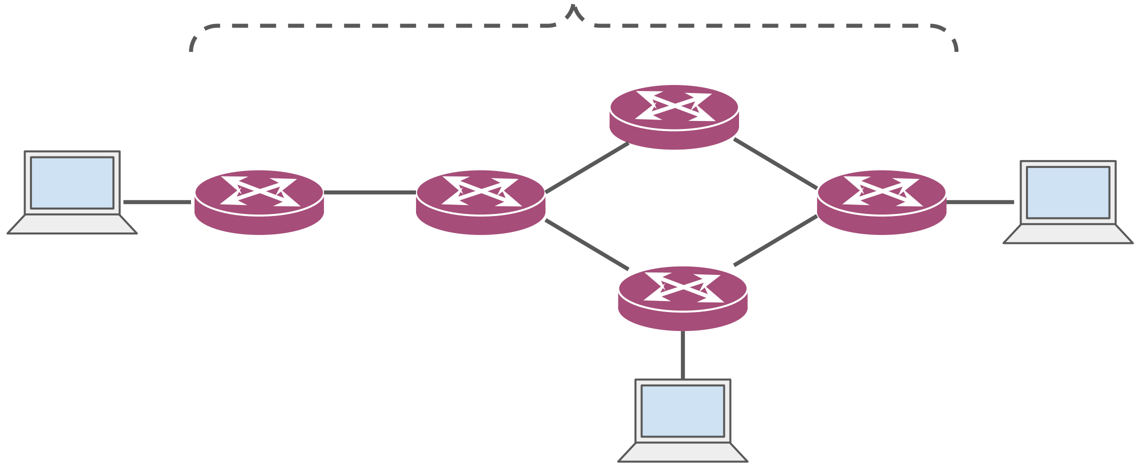
Recurring theme in this course

Separate *what* you want the network to do from *how* it is implemented → **Abstraction**

Don't implement in *manually* 🙂 → **Automation**

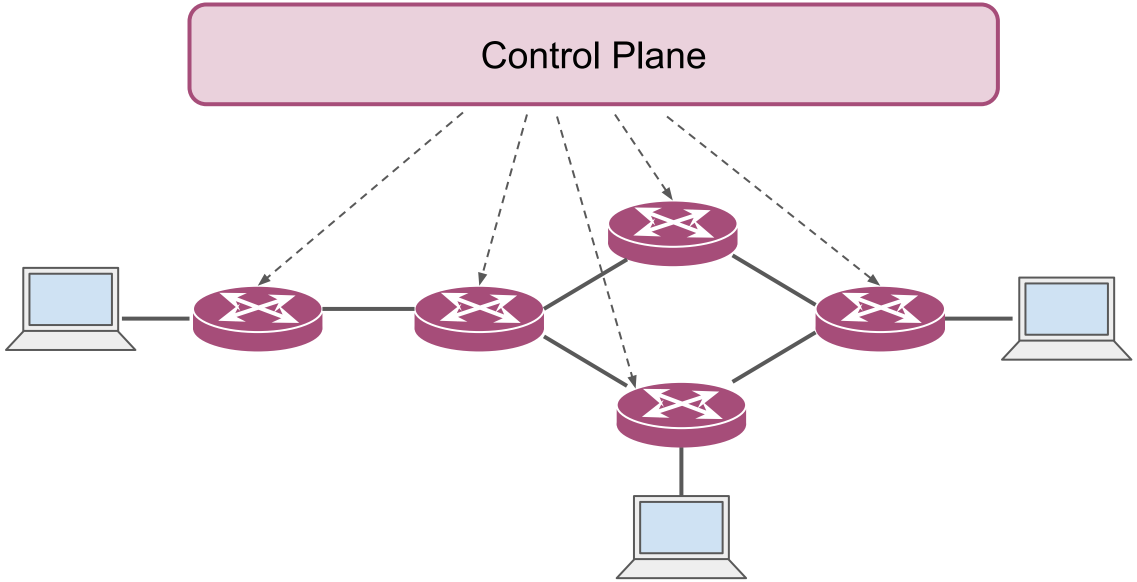
Here are some examples…

Configure a pre-defined set of distributed protocols (e.g., OSPF, BGP, etc.) to pick your desired forwarding paths.



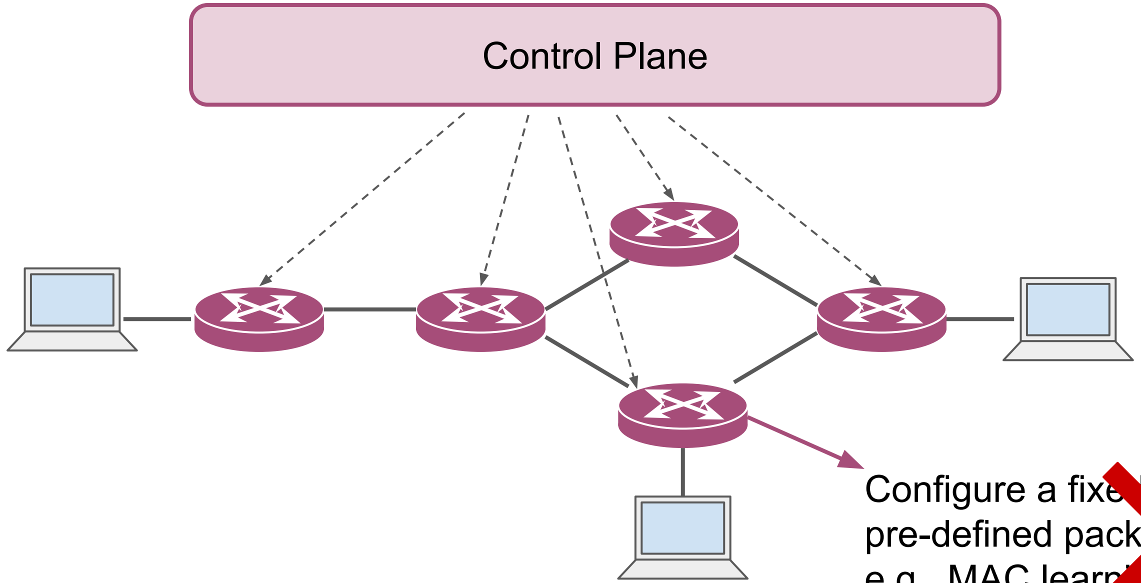
Here are some examples…

- Write a program that decides the forwarding paths.
- Have a runtime compute and communicate proper configurations to network devices.



Here are some examples…

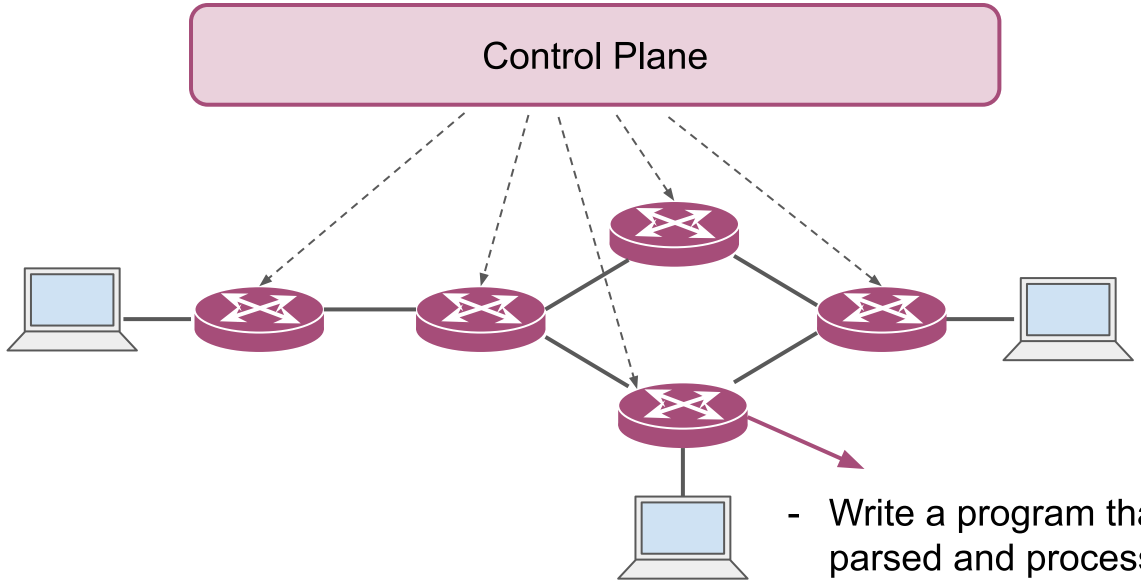
- Write a program that decides the forwarding paths.
- Have a runtime compute and communicate proper configurations to network devices.



~~Configure a fixed-function hardware with pre-defined packet processing steps, e.g., MAC learning → GRE-Tunnel Processing → IP forwarding~~

Here are some examples…

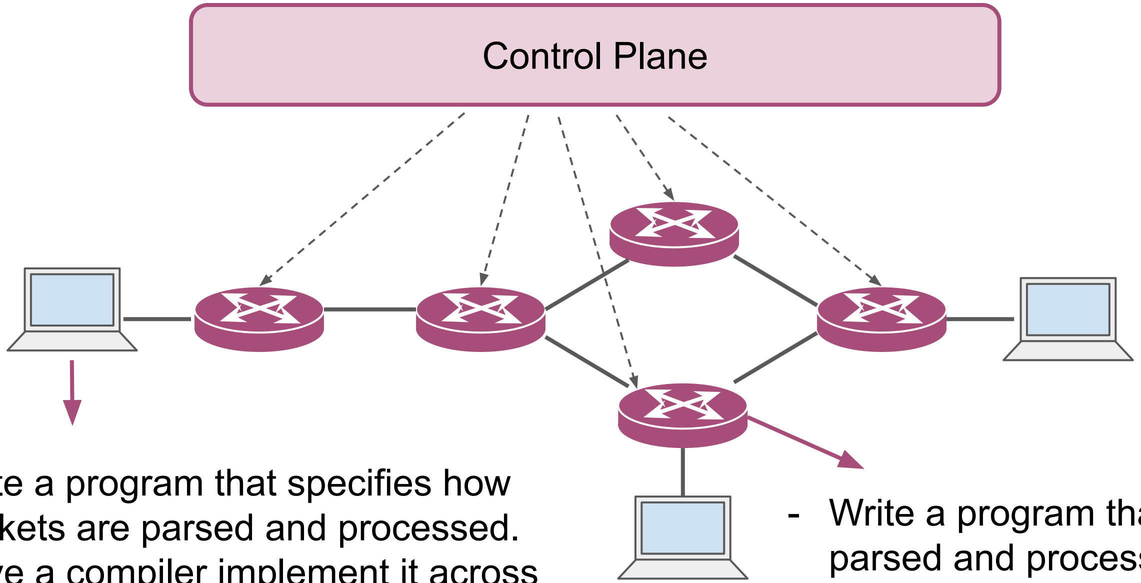
- Write a program that decides the forwarding paths.
- Have a runtime compute and communicate proper configurations to network devices.



- Write a program that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

Here are some examples…

- Write a program that decides the forwarding paths.
- Have a runtime compute and communicate proper configurations to network devices.

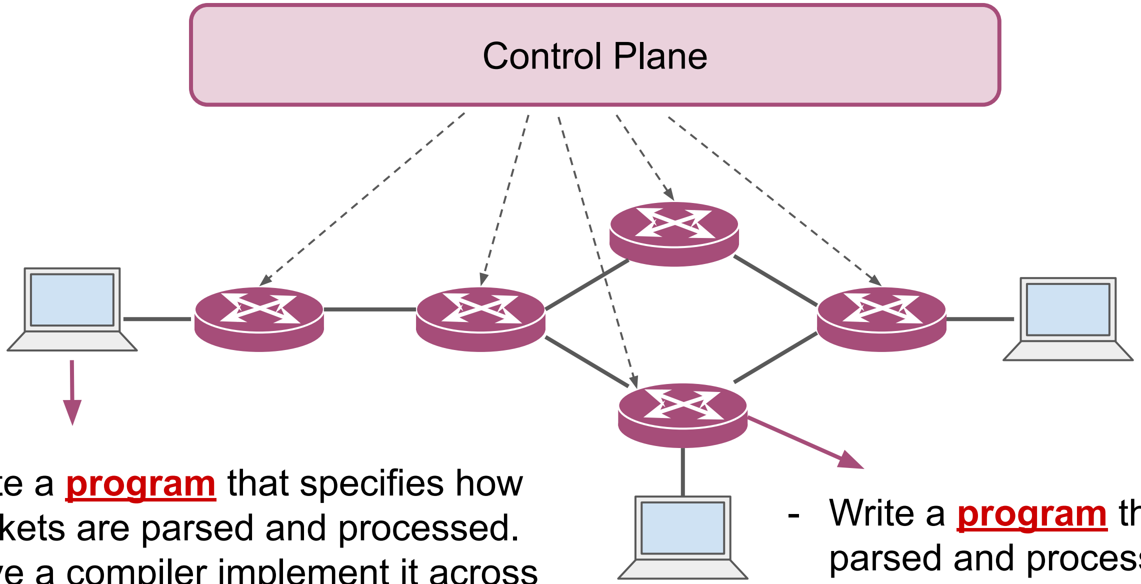


- Write a program that specifies how packets are parsed and processed.
- Have a compiler implement it across user-space, the Kernel, and hardware accelerators.

- Write a program that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

Here are some examples…

- Write a **program** that decides the forwarding paths.
- Have a runtime compute and communicate proper configurations to network devices.

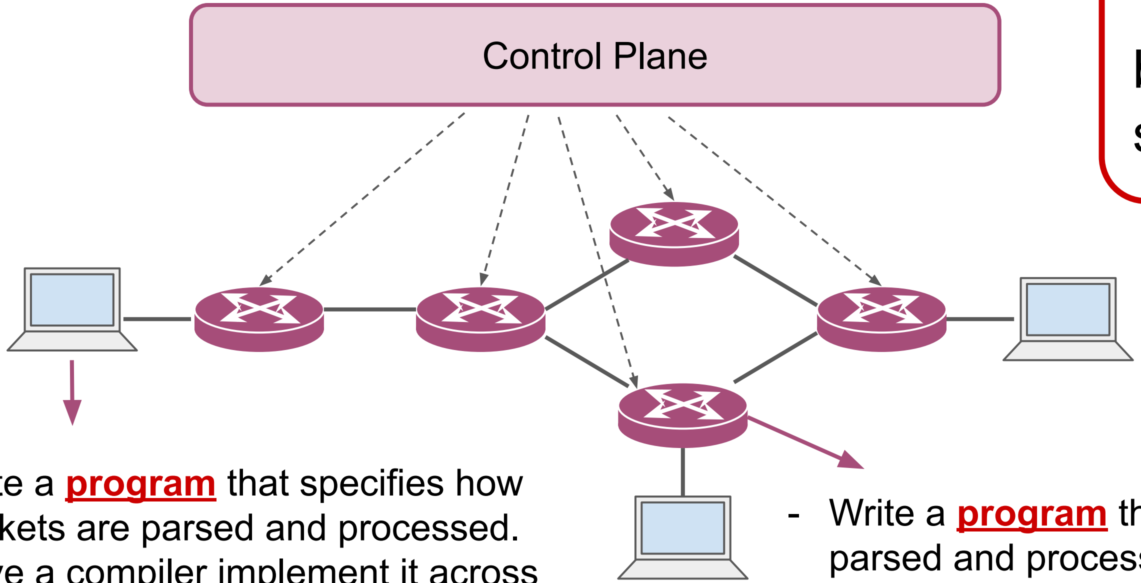


- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler implement it across user-space, the Kernel, and hardware accelerators.

- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

Here are some examples…

- Write a **program** that decides the forwarding paths.
- Have a runtime compute and communicate proper configurations to network devices.



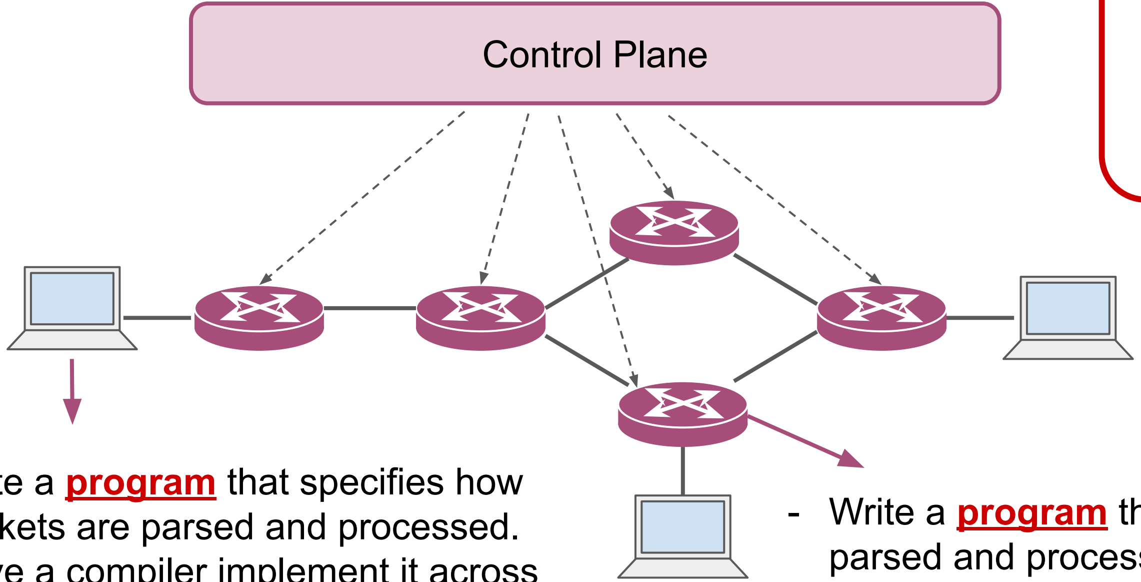
Treat the network as a big, distributed, and specialized computer

- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler implement it across user-space, the Kernel, and hardware accelerators.

- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

Here are some examples…

- Write a **program** that decides the forwarding paths.
- Have a runtime compute and communicate proper configurations to network devices.



Programmable Networks

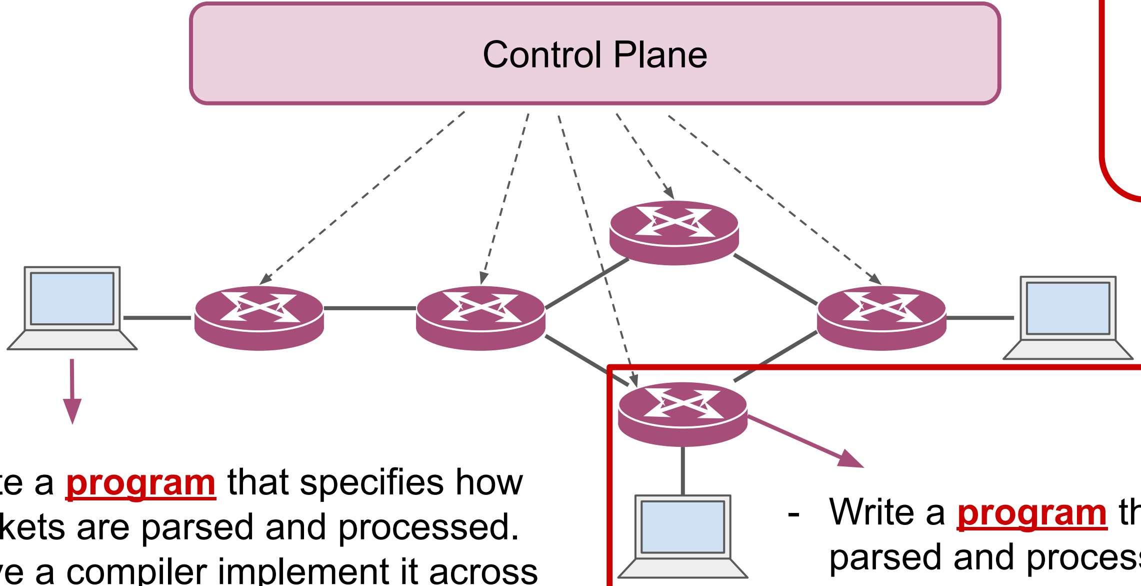
- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler implement it across user-space, the Kernel, and hardware accelerators.

- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

Here are some examples…

- Write a **program** that decides the forwarding paths.
- Have a runtime compute and communicate proper configurations to network devices.

Programmable Networks



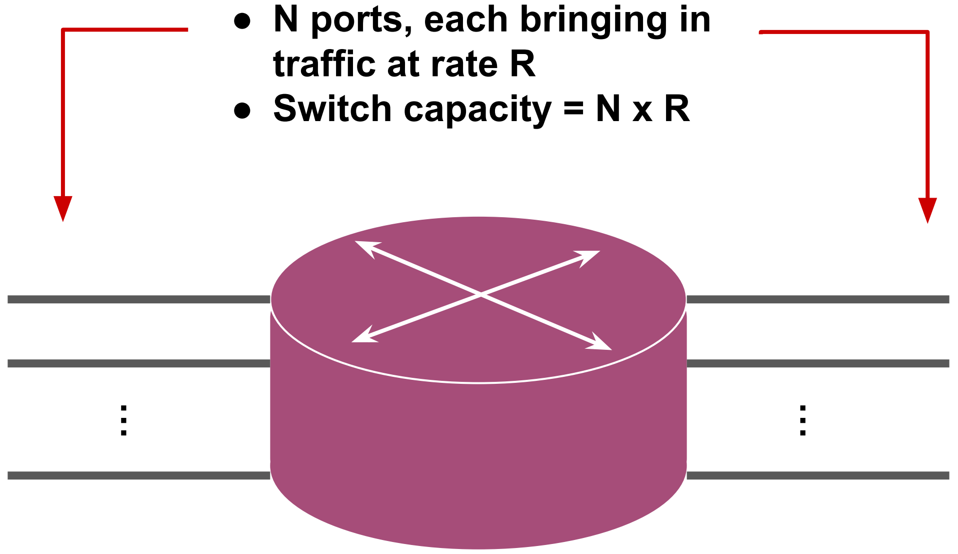
- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler implement it across user-space, the Kernel, and hardware accelerators.

- Write a **program** that specifies how packets are parsed and processed.
- Have a compiler translate that into instructions for switch hardware.

Programmable Switches

Challenge: High-Speed Reconfigurable Data Plane

- Switch data planes need to process packets very fast


● N ports, each bringing in traffic at rate R
● Switch capacity = N x R
⋮
⋮

Challenge: High-Speed Reconfigurable Data Plane

- Switch data planes need to process packets very fast

R = 100 Gbps

For back-to-back 64B packets, we have a packet every ~5ns.

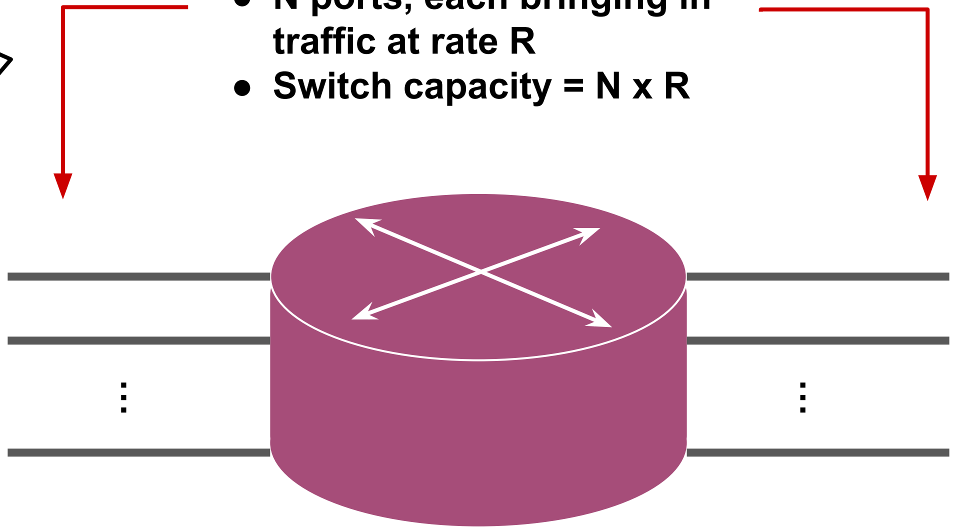
For back-to-back 1500B packets, we have a packet every ~120ns.

N = 16

This happens concurrently on 16 ports…

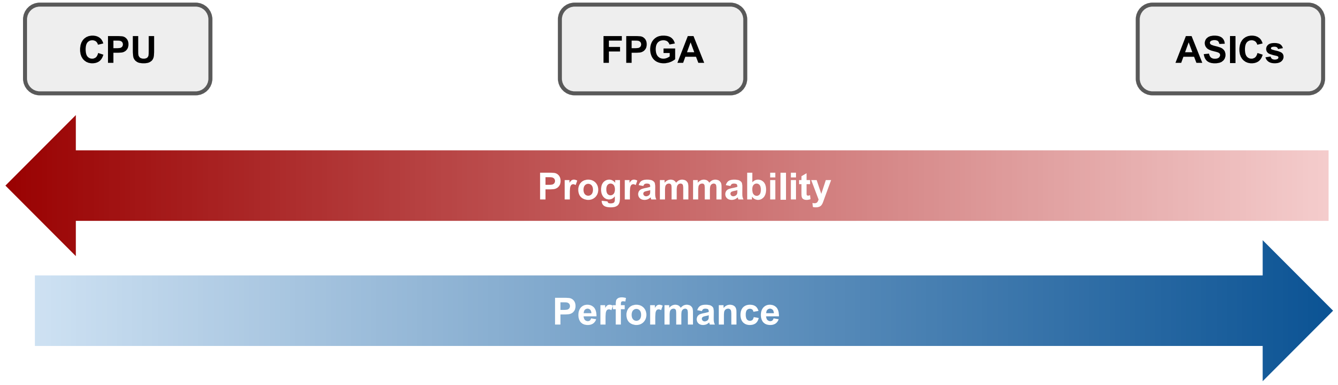
N x R = 1.6 Tbps!

- **N ports, each bringing in traffic at rate R**
- **Switch capacity = N x R**



Challenge: High-Speed Reconfigurable Data Plane

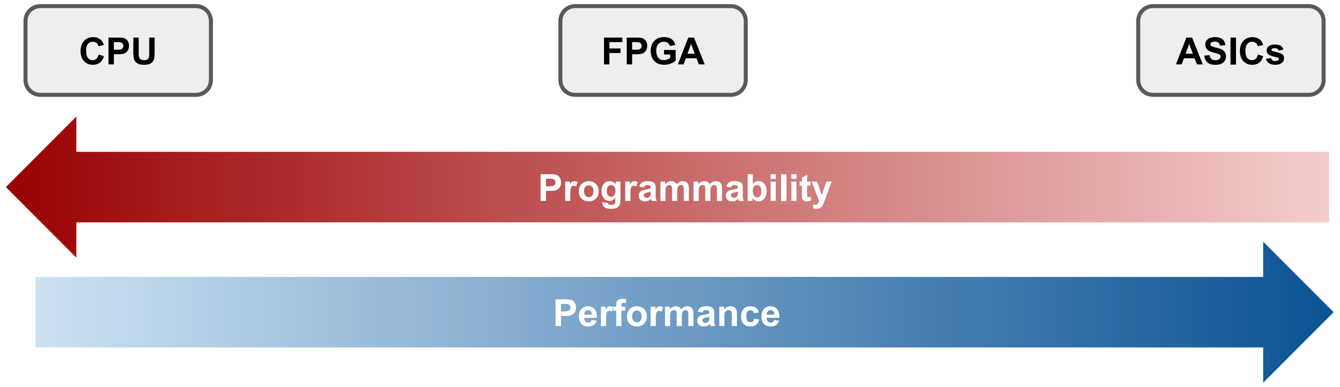
- There is a trade-off between programmability and performance



Challenge: High-Speed Reconfigurable Data Plane

- programmability and performance

General-purpose processors like CPUs can be programmed to execute any logic.

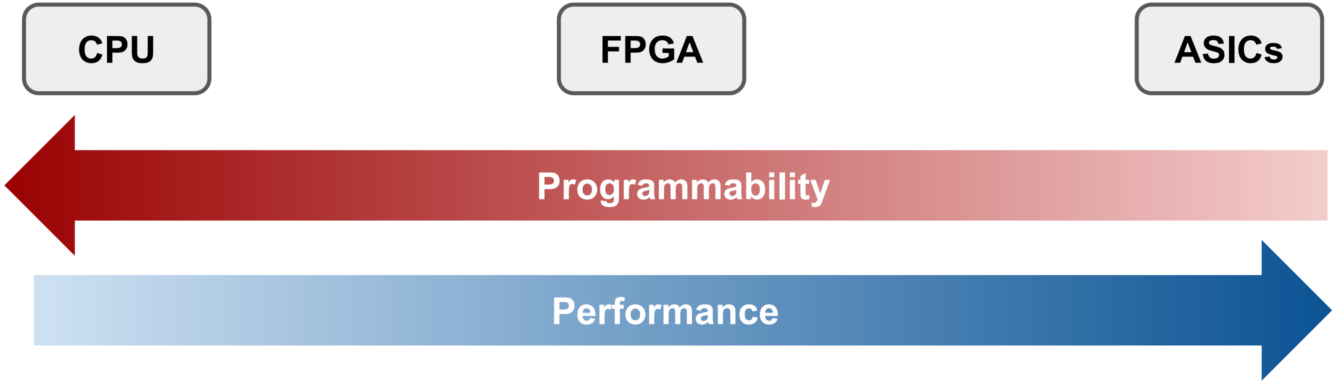


Challenge: High-Speed Reconfigurable Data Plane

- programmal

General-purpose processors like CPUs can be programmed to execute any logic.

Fixed-function ASICs are customized and optimized to for a certain kind of computation.

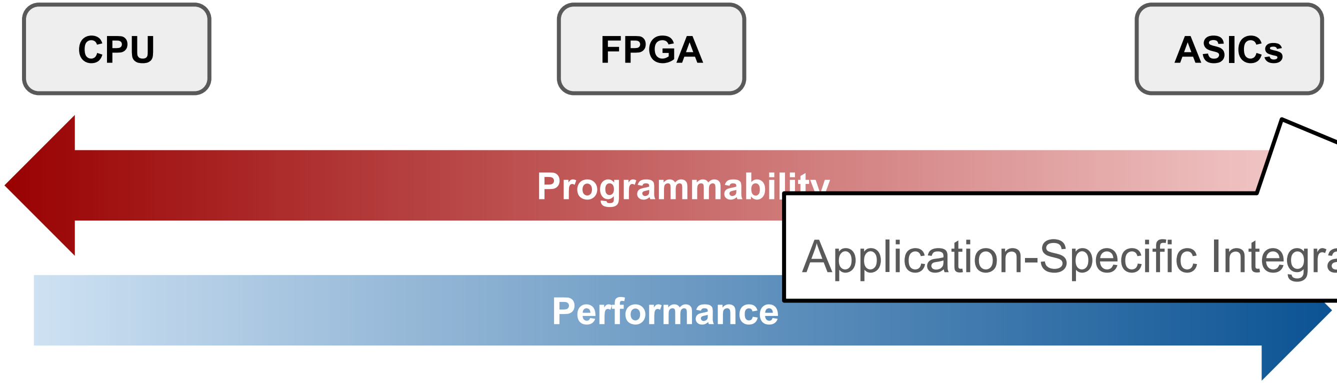


Challenge: High-Speed Reconfigurable Data Plane

- programmal

General-purpose processors like CPUs can be programmed to execute any logic.

Fixed-function ASICs are customized and optimized to for a certain kind of computation.

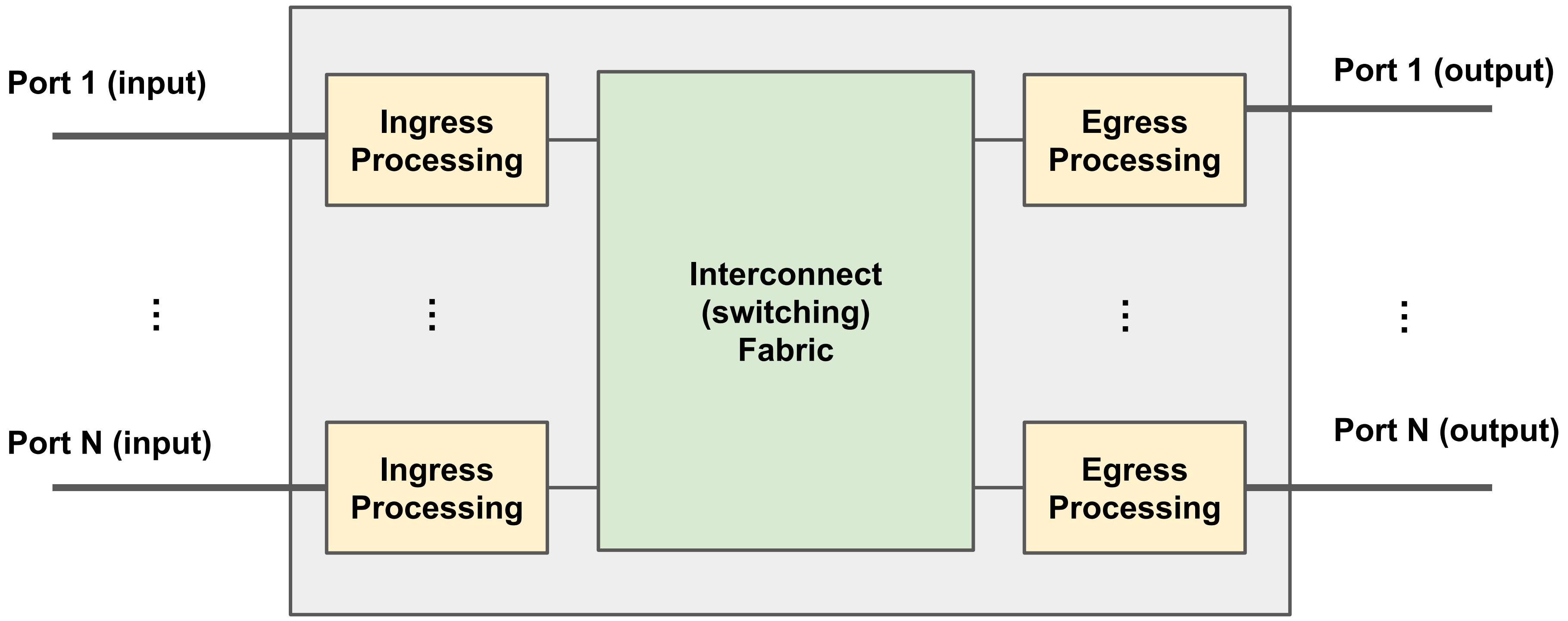


Application-Specific Integrated Circuit

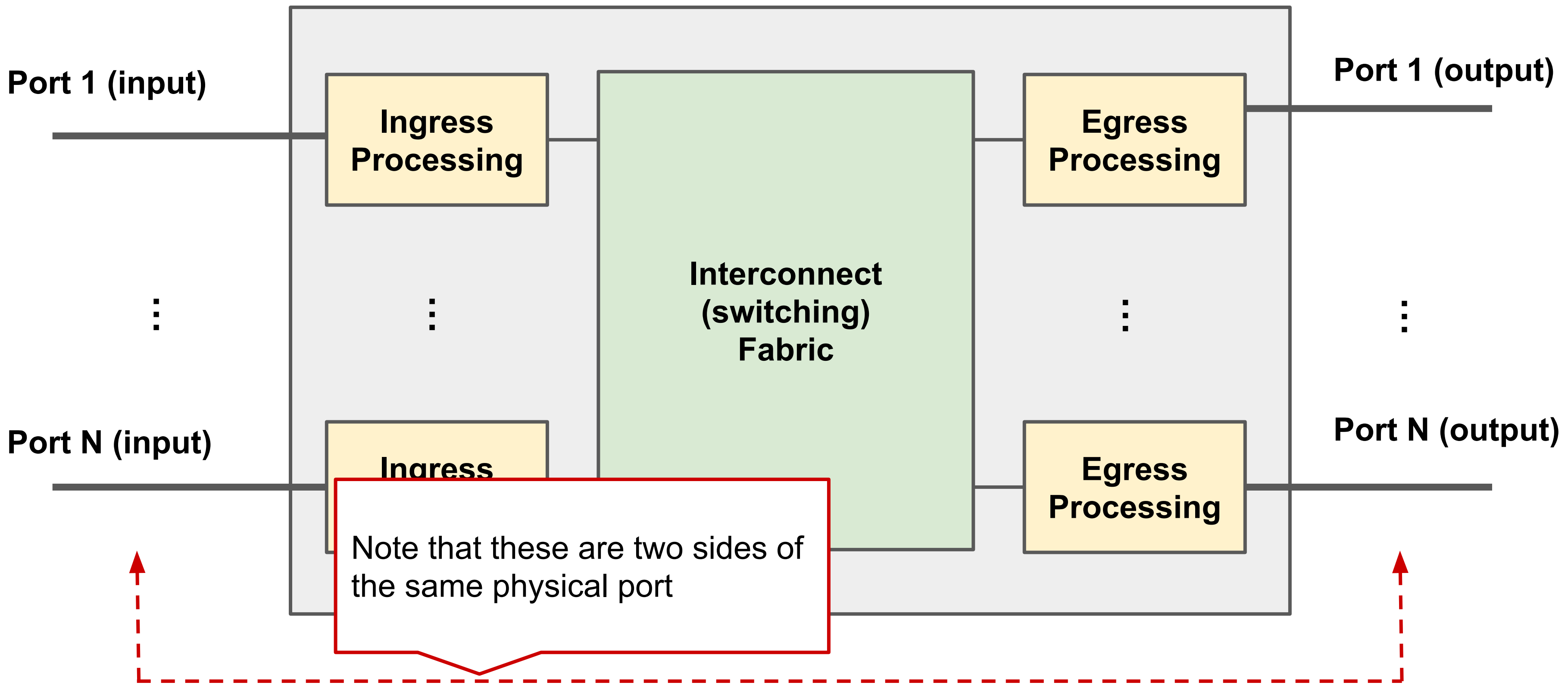
Challenge: High-Speed Reconfigurable Data Plane

- **Traditionally:** switching chips were ASICs
    - customized for packet processing, e.g., packet parsing, forwarding tables, etc.
- **The "programmability" trend:**
    - **Q1:** Is it possible to have a high-speed reconfigurable switch data plane?
    - **Q2:** How much reconfigurability can we add to the switch data plane and still be able to perform high-speed packet processing?

Inside a (output-queued) switch

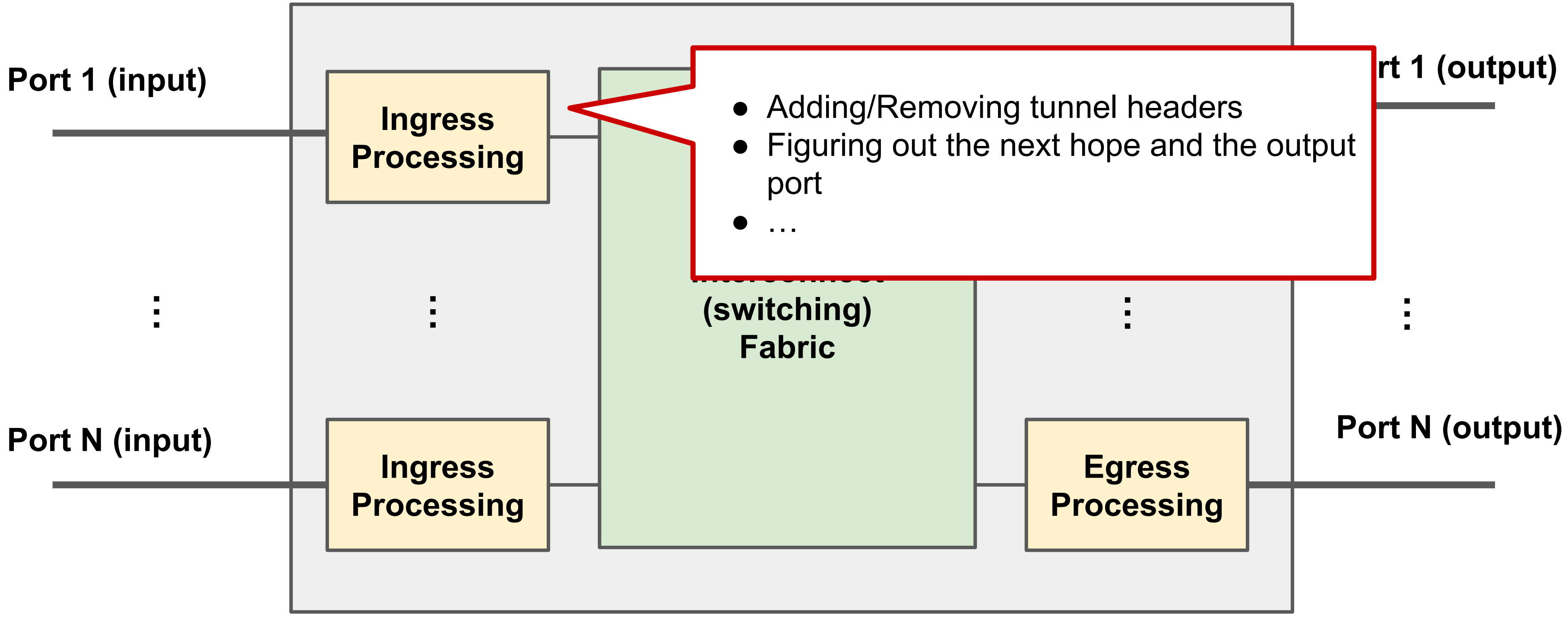


Inside a (output-queued) switch



Note that these are two sides of the same physical port

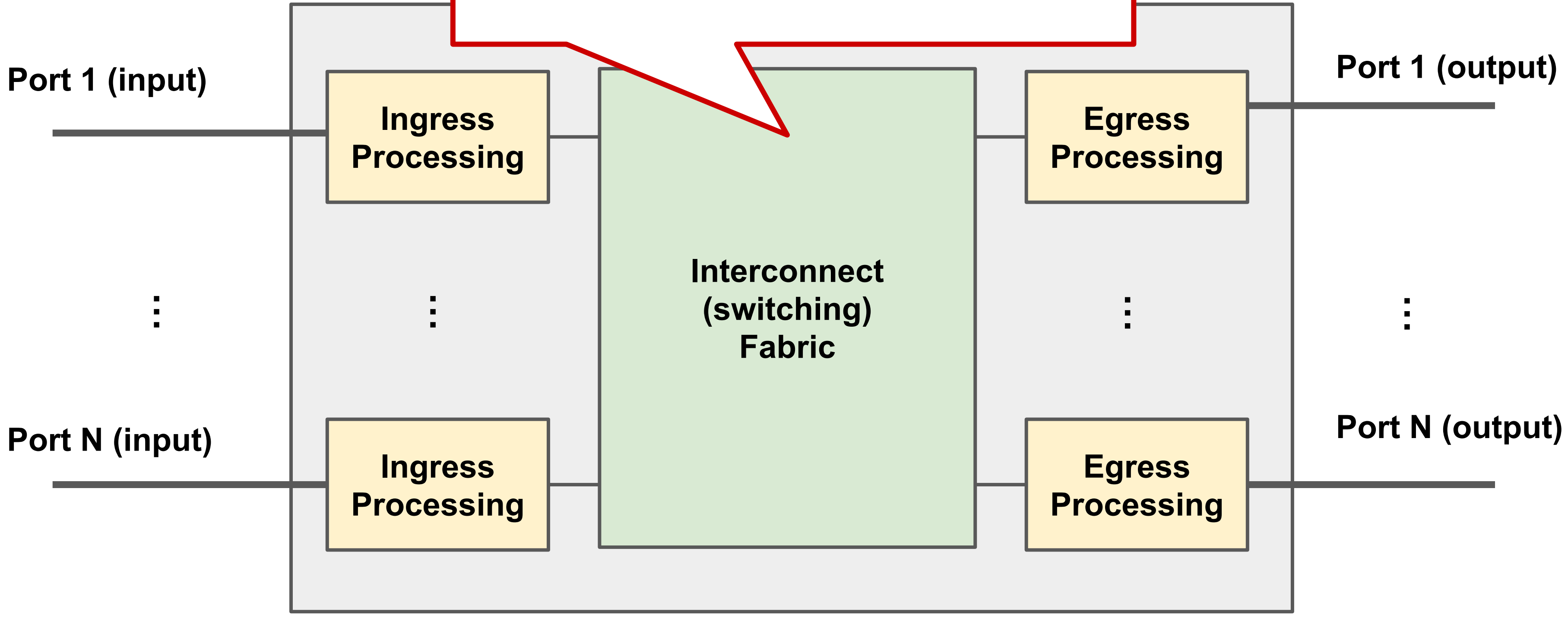
Inside a (output-queued) switch



- Adding/Removing tunnel headers
- Figuring out the next hope and the output port
- …

Inside a (output-c

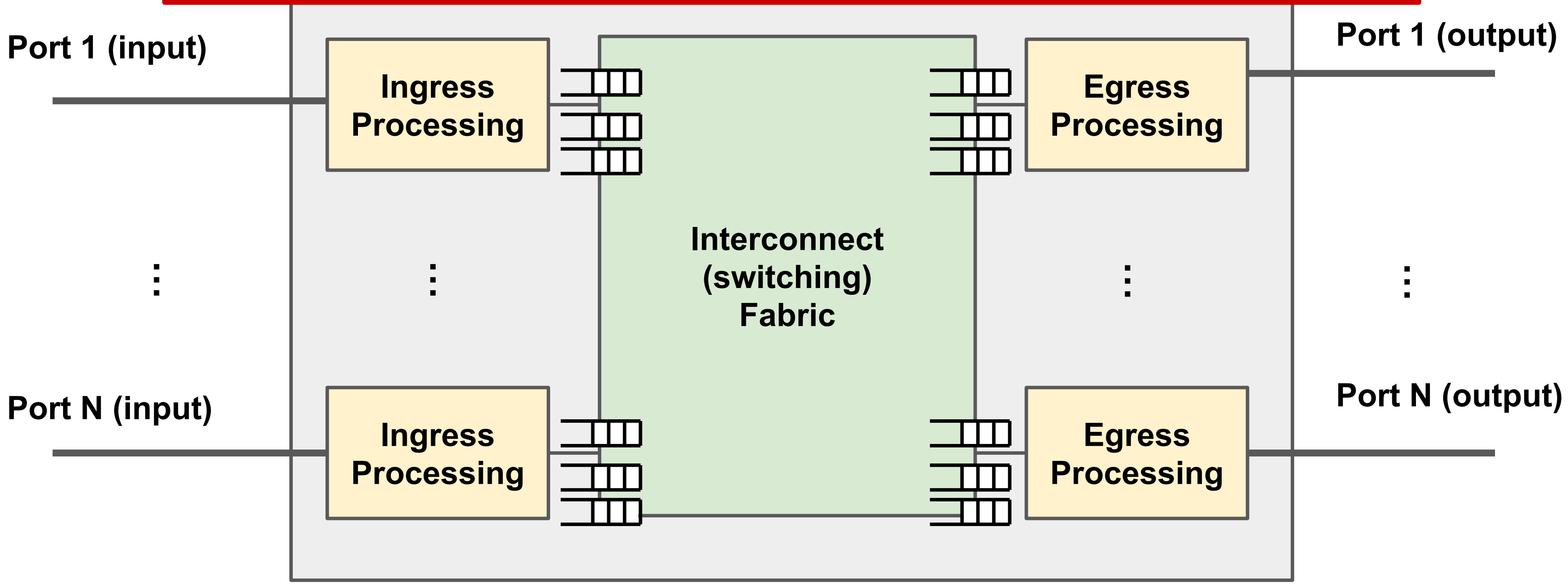
- Connects input ports to output ports
- Needs to operate at high speed (~ N times the speed of an individual port)



Inside

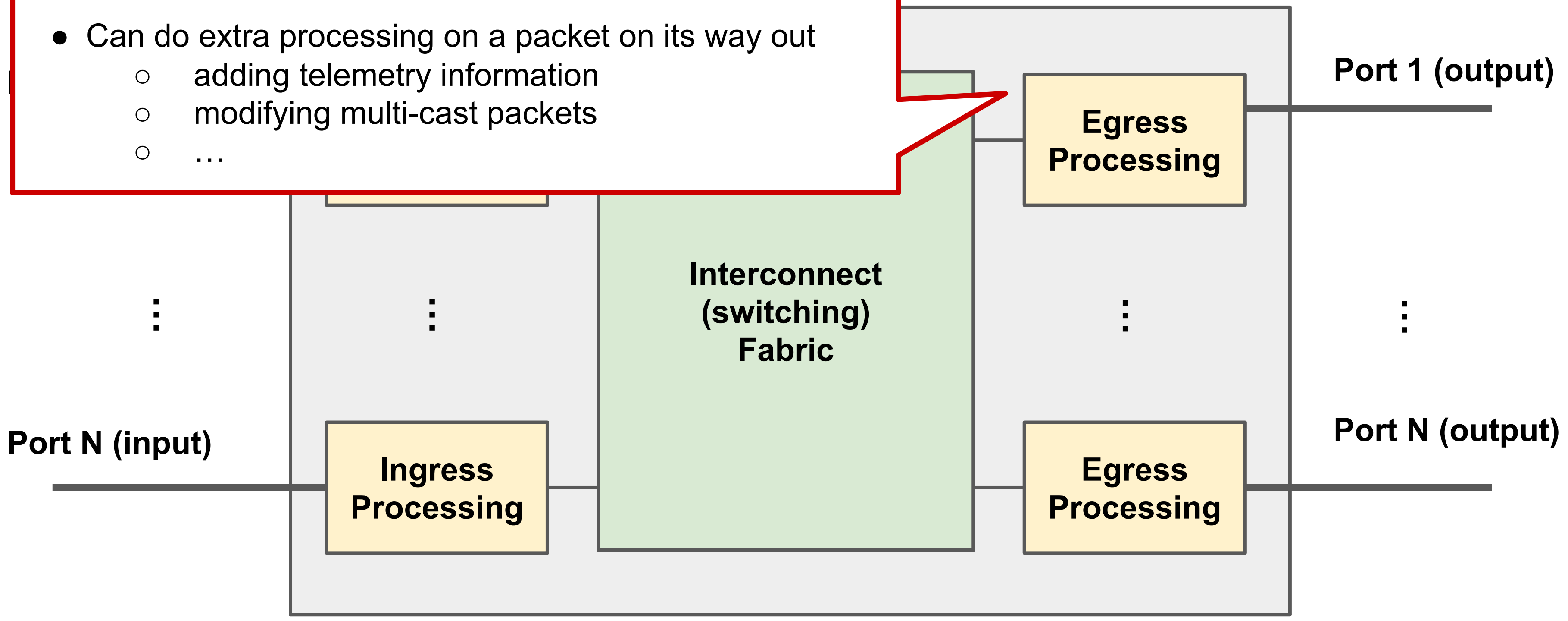
Traffic manager:

- Packets going to the same output will be buffered in a queue
  - In ingress and/or egress.
- Packet scheduling algorithms decide which packets will go out of that port next

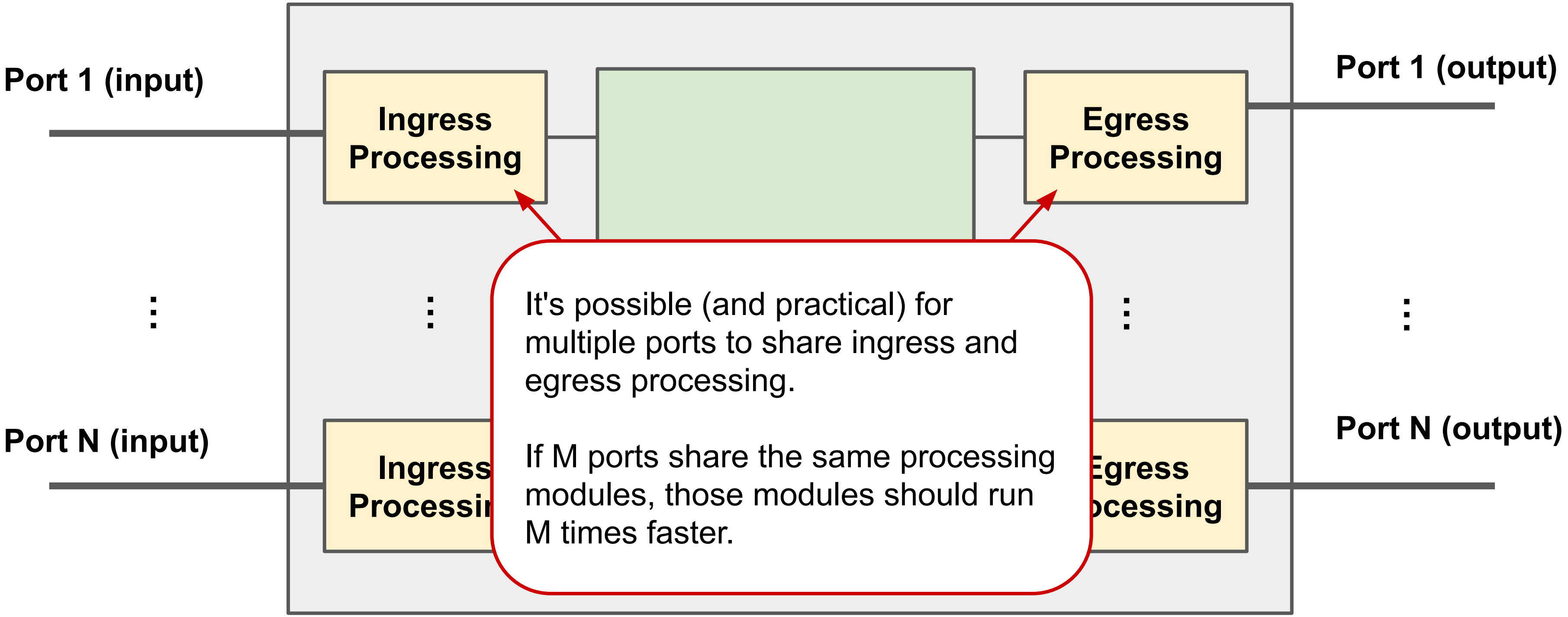


Inside a (output-queued) switch

- Can do extra processing on a packet on its way out
  - adding telemetry information
  - modifying multi-cast packets
  - …



Inside a (output-queued) switch



It's possible (and practical) for multiple ports to share ingress and egress processing.

If M ports share the same processing modules, those modules should run M times faster.

What should a "programmable" switch look like?

- We can't make everything programmable
  - the programmability-performance trade-off
- How do we decide what should be fixed and what programmable?
  - Which parts are subject to more innovation?
  - The logic of which part do we want to change more frequently?
  - Where can we afford to pay the overhead of programmability?

Proposals for programmable switch architectures (not exhaustive)

- 2013: Reconfigurable Match-Action Tables (RMT)
    - Evolved into Protocol-Independent Switch Architecture (PISA)
    - There was a successful startup (Barefoot Networks) and a commercial switching chip based on it (Tofino).
    - Acquire by Intel, and unfortunately discontinued ~2 years ago.
    - Why are we still talking about this then?
- 2017: dRMT = disaggregated RMT
- 2022: Trio by Juniper Networks
- 2022: FlexCore
- 2024: OptimusPrime

How do we program these switches?

- The P4 language is the de-facto at the moment
  - Came out of the research on RMT switches
  - Is the language used for programming Tofino chips
  - Has an active and large community (academic and industry)
  - checkout https://p4.org/
- Its benefits and use cases have extend beyond programmable switching chips
  - Programming other components of the network
  - Testing and verification of fixed-function switches (e.g., at Google)
  - …
- Other language/extensions have been proposed as well
  - NPL (Broadcom)
  - Domino, Mantis, MicroP4, P4All, …

What are some research questions to explore?

- What is the set of functionality that, if placed in the switch, will significantly benefit the network (and the applications using it) as a whole?
  - The answer could change from network to network
  - Are there some common sets of primitives?
- Can current switch architectures support them at high-speed?
  - If not, what changes are necessary?
- Do we have the right programming abstraction for implementing them?
- Heads-up: this has been studied quite a bit in the past ten years.
  - That doesn't mean all the problems are solved though.

What are some research questions to explore?

- Runtime programmability
    - Can you re-program the switch while it is still processing traffic?
    - Otherwise, you'll have to drain the switch, change the program, and put the switch back on the path.
- Has lead to re-thinking the hardware architecture and programming abstractions.