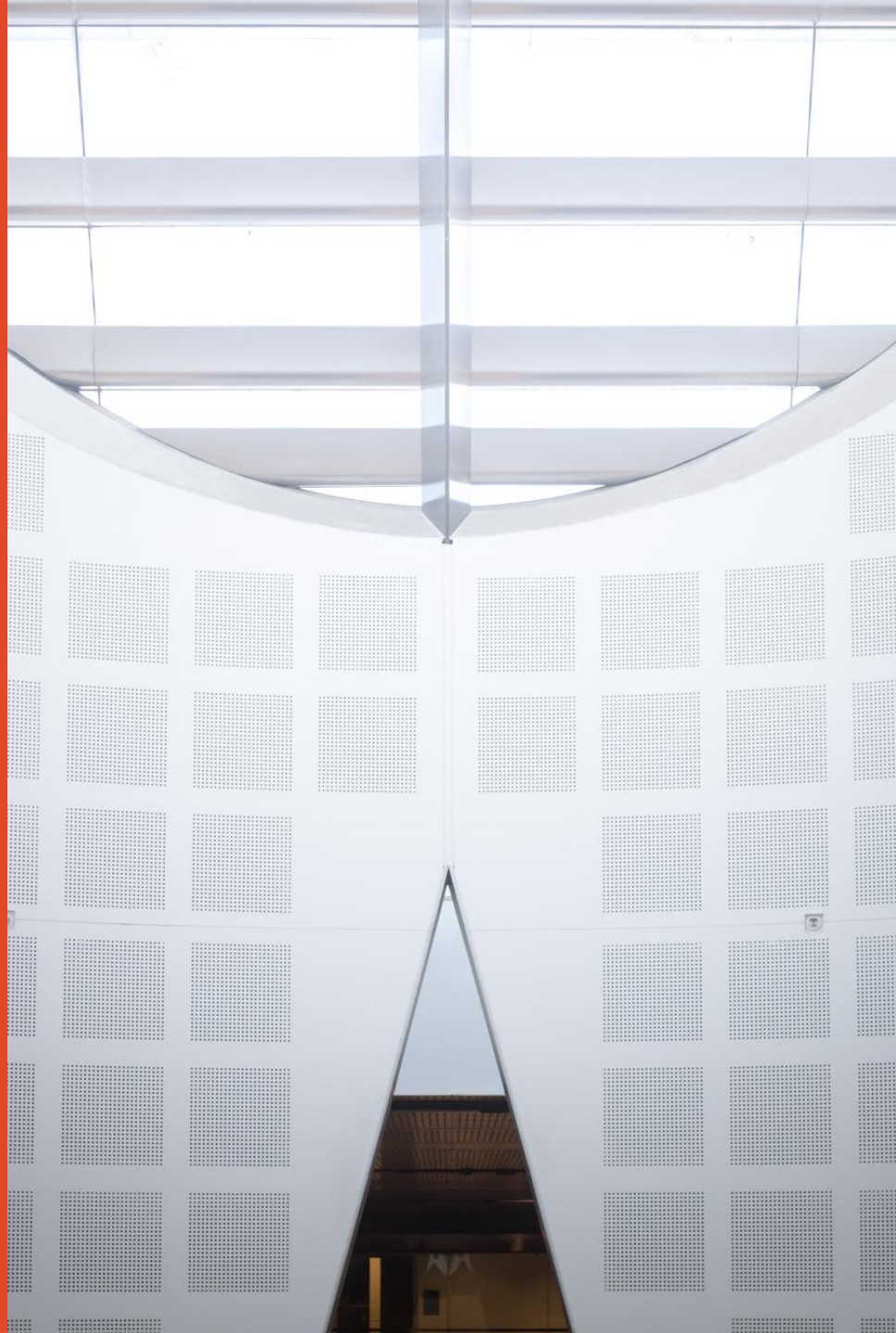
Topic 7: Programmable Data Plane

Presented by
Dong YUAN

School of Electrical and Information Engineering

dong.yuan@sydney.edu.au

THE UNIVERSITY OF SYDNEY



Overview of Data Plane

Overview of Data Plane

- What does data plane do?
  - Router gets packet
  - Looks at packet header for destination
  - Looks up forwarding table for output interface
  - Modifies header (TTL, IP header, etc.)
  - Passes packet to appropriate output interface

Key functions of data plane

- Streaming algorithms that act on packets
  - Matching on some bits (header), taking simple actions
  - Follow the order of control and management plane
- Wide range of functions
  - Forwarding
  - Access control
  - Mapping header fields
  - Traffic monitoring
  - Buffering and marking
  - Shaping and scheduling
  - Deep packet inspection

The Need of Software/Programmable Data Plane

- Network devices are diverse
  - Must do more than just forward/route packets
  - Adding functions is difficult
  - Match/action is only one type function of data plane
- Data plane design goals
  - Flexible
  - Extensible
  - Clean interfaces

A trade-off of software and hardware

- Software – flexible
- Hardware – fast

- To get the best of both world – design a programmable data plane
  - Make the software faster
  - Make the hardware more programmable

Motivation

- SDN protocols require data-plane changes
- Performance requirement
  - Protocols must forward packets at acceptable speeds.
- Support different protocols
  - Run in parallel with existing protocols

- Requirement of SDN data plane – a platform that
  - Forwards packets at high speed
  - Run multiple protocols in parallel

Existing Approaches

- Develop Custom Software
  - Flexible, easy to program
  - Slow forwarding speeds
- Develop Custom Hardware
  - Excellent performance
  - Long development cycles, rigid
- Develop programmable hardware
  - Flexible and fast
  - Programming is difficult
- https://rg0now.github.io/prog_dataplane_reading_list/README.html

Contents

- Software Router
- Programmable Hardware Data Plane
- Network Assembly Language
- High Level Language

Click: A Software Data Plane

- Princeton project (early)

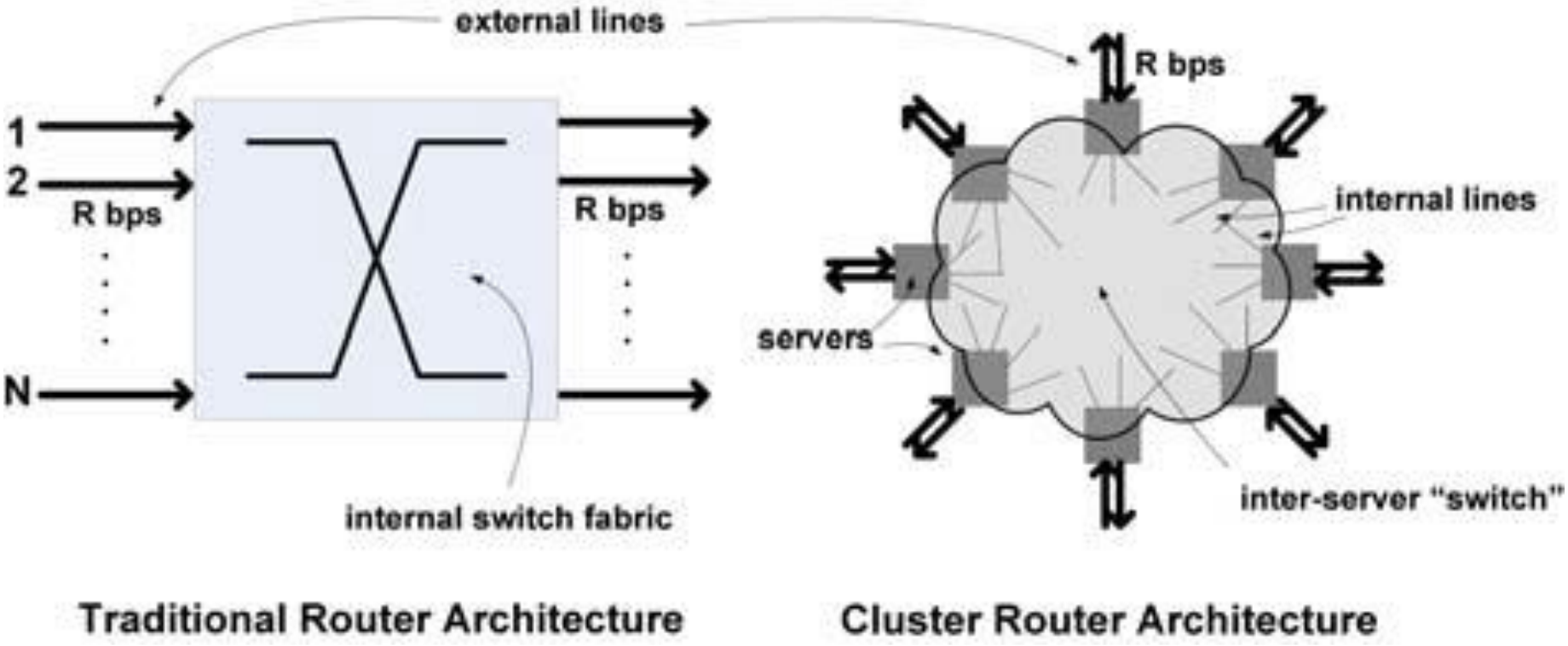
- Elements – building blocks
- Each element provides unique function
  - Packet switching
  - Lookup and Classification
  - Dropping

- Implement functions: assemble building blocks

Software router

- Dobrescu, Mihai, et al. "RouteBricks: exploiting parallelism to scale software routers." *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009.

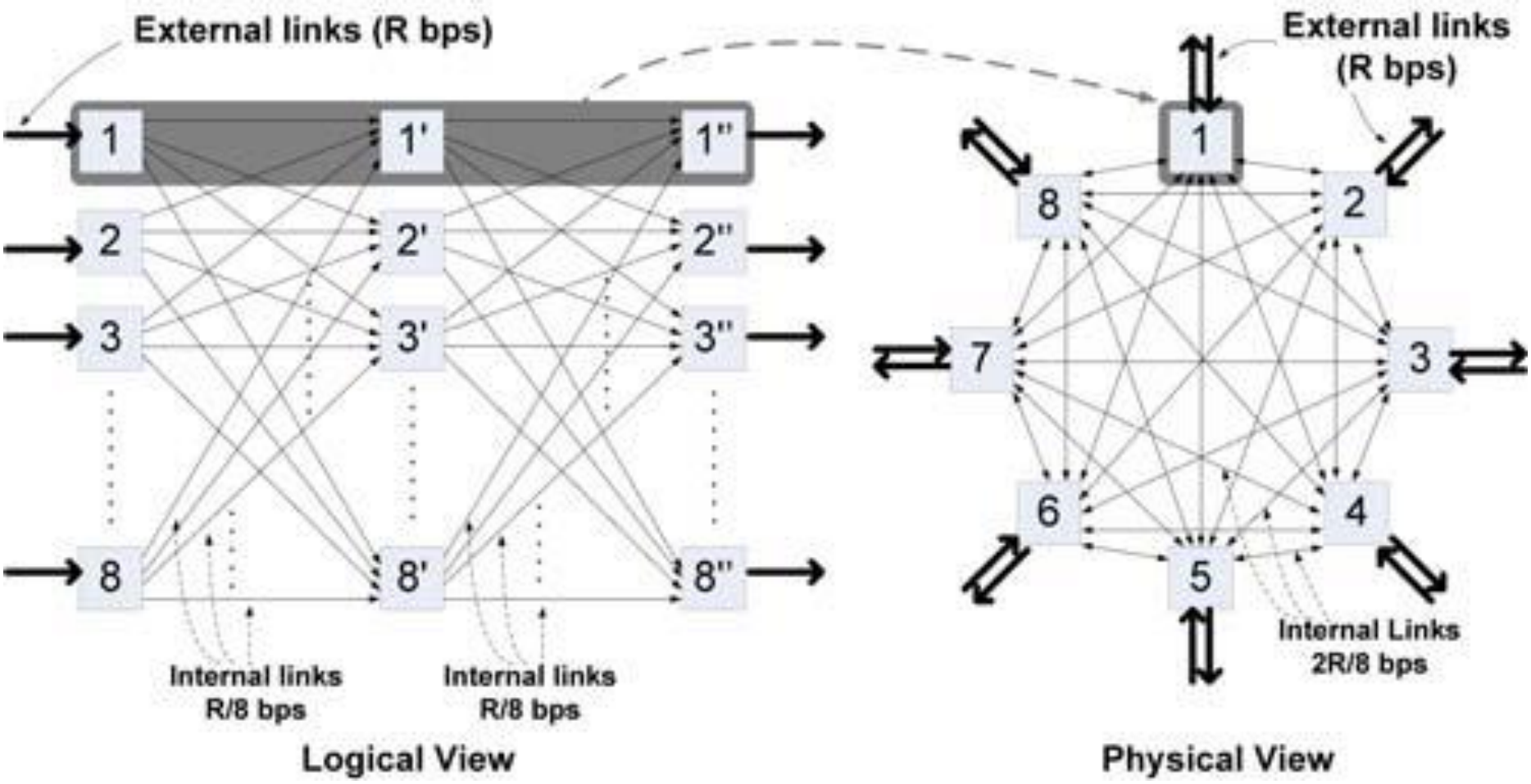
Hardware Router



- Processing at rate R bps per line card
- Switching at rate N*R by switch fabric

- N*N internal links of capacity R

RouteBricks



- **Internal link need the capacity of R/N**
- **Per-server processing rate: 3R**

Bookkeeping problem

- Managing too many packet descriptors
  - Moving between NIC and memory
- Solution: batch packet operations
  - NIC batches multiple packet descriptors
  - CPU polls for multiple packets
  - Cost: increased latency

Parallel processing with Multi-core CPU

- 1 core per queue (avoid locking memory)
- 1 core per packet (faster)

- Large packet buffers to hold multiple packets
- Use GPU to accelerate

- Many related papers…

Other approaches to fast software forwarding

- Large packet buffers to hold more packets
- Batch processing
- Avoid lookups on bridge between virtual interfaces and physical interfaces

- DPDK
  - Data Plane Development Kit that consists of libraries to accelerate packet processing workloads running on a wide variety of CPU architectures.
  - It offloading TCP packet processing from the operating system kernel to processes running in user space.
  - Frist introduced by Intel, now is an open source software project managed by the Linux Foundation

Today's hardware constraints

- Openflow is protocol dependent because of constrains of traditional switching chips.
- Mapping to existing switching chips enabled quick adoption, but it also limited the control of Openflow.
- What if we could re-design the data plane?

Few Data Plane Primitives

- The set of functions that we want to perform on packets are quite limited
  - Bit shifting, Parsing and rewriting header fields…
- Build a flexible data plane by developing modules + ways to integrate them.
- Basic idea to build programmable data plane

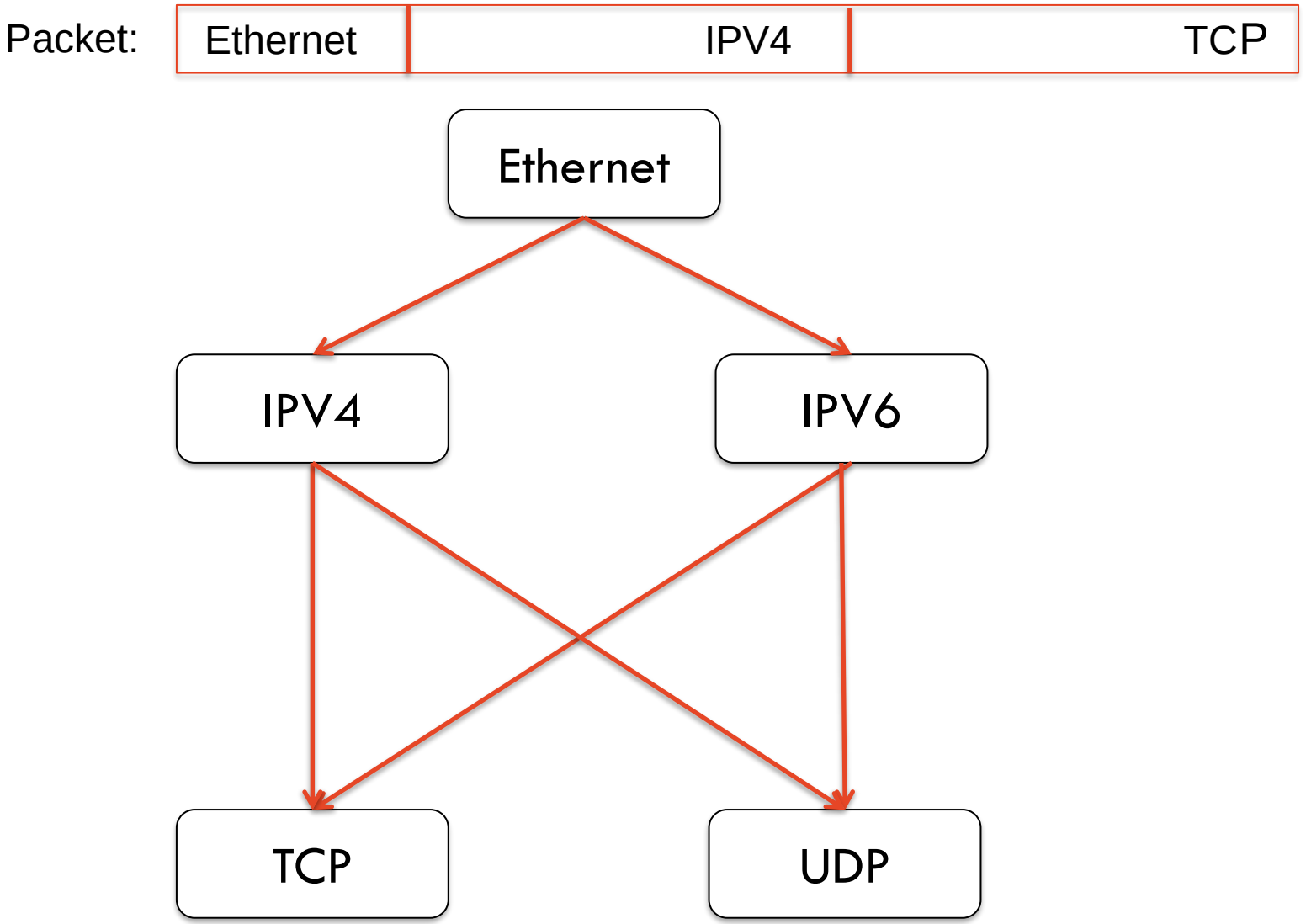
An OpenFlow Chip

- Bosshart, Pat, et al. "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN." *ACM SIGCOMM Computer Communication Review*. Vol. 43. No. 4. ACM, 2013.

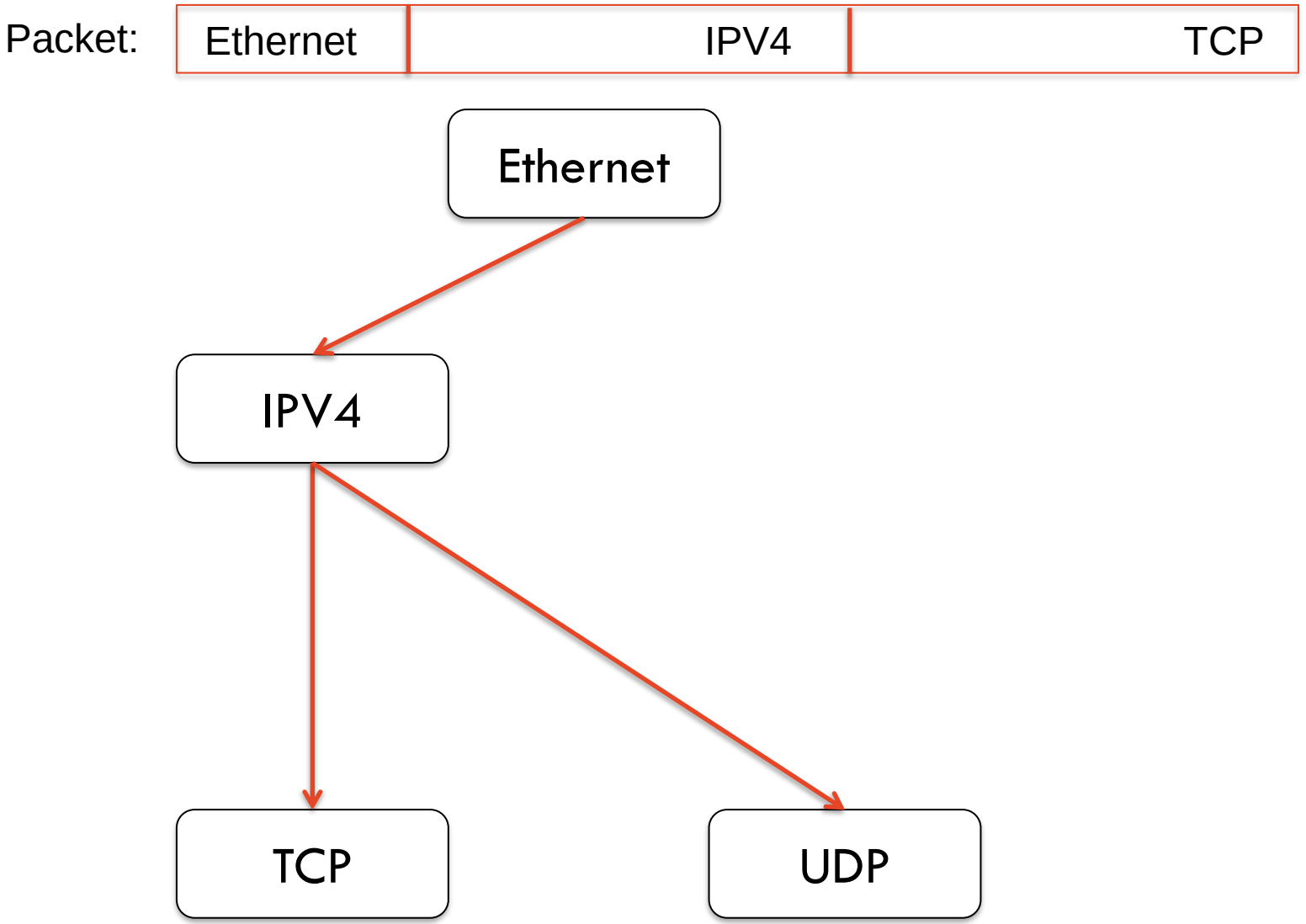
- Generalise, programmable match-action primitives

- RISC-like architecture

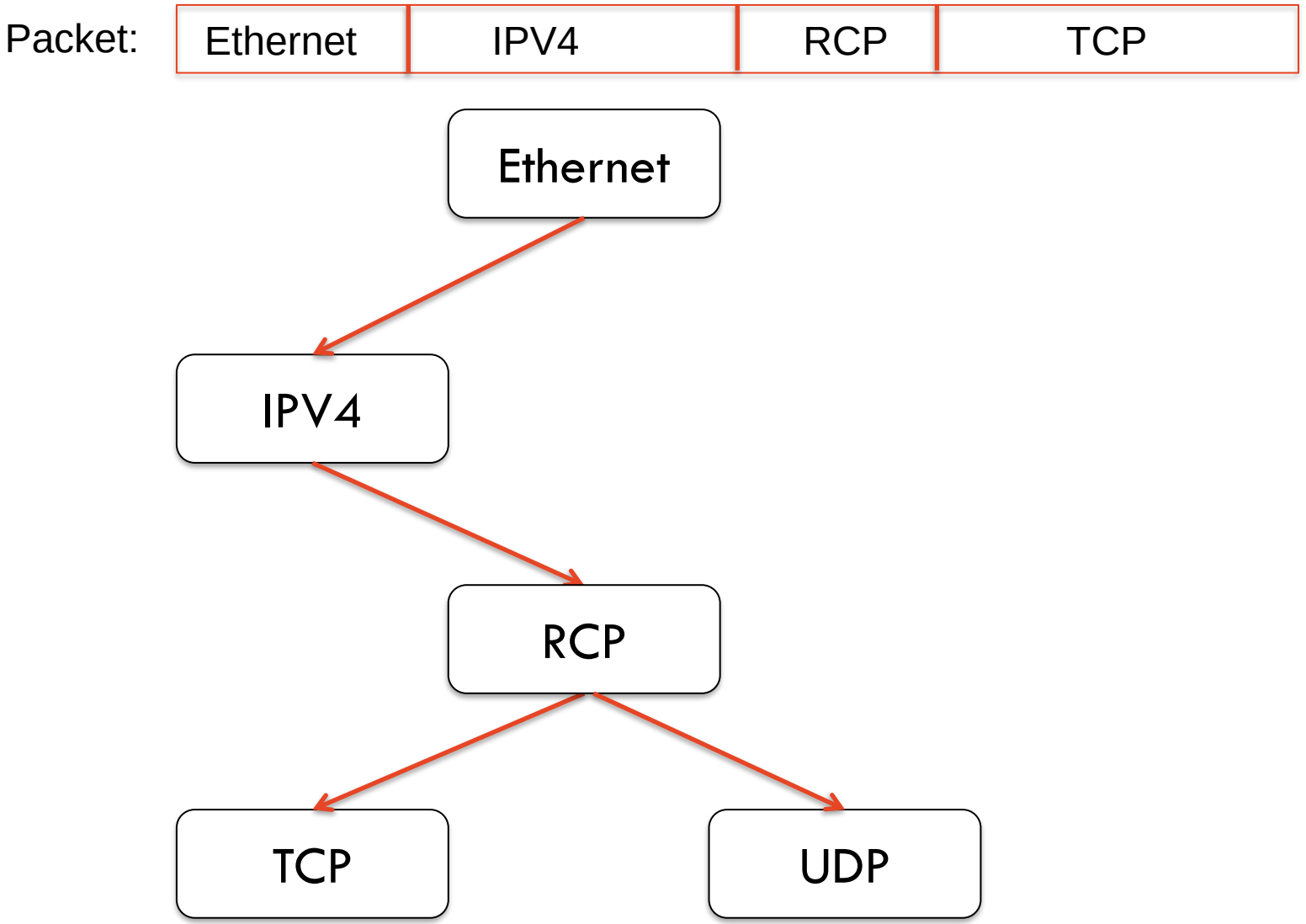
Arbitrary Fields: The Parse Graph



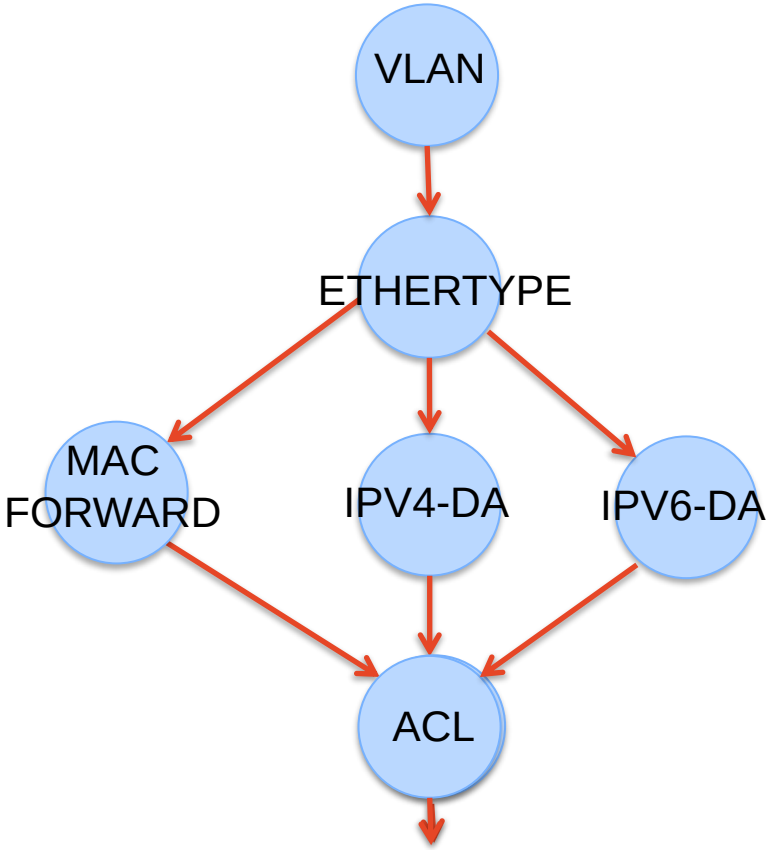
Arbitrary Fields: The Parse Graph



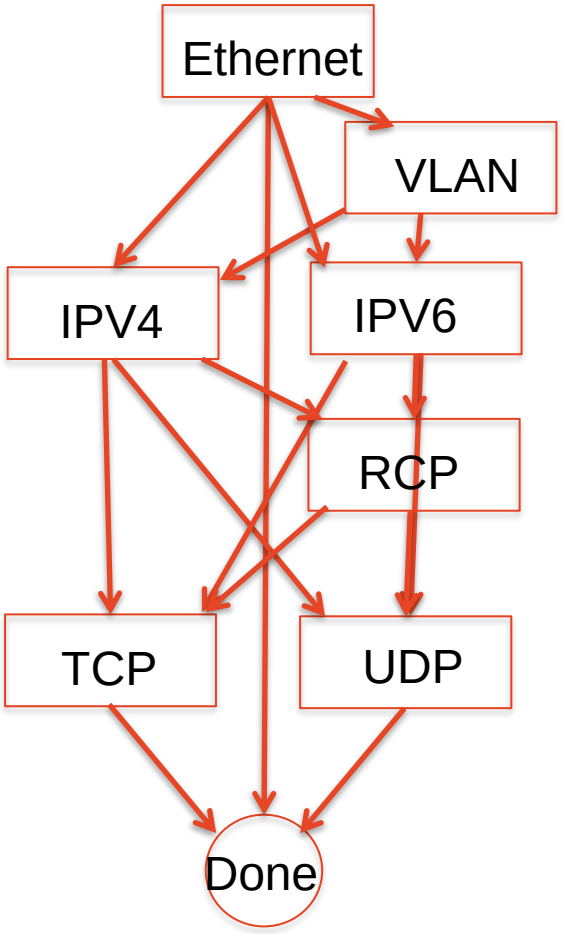
Arbitrary Fields: The Parse Graph



Reconfigurable Match Tables: The Table Graph



Changes to Parse Graph and Table Graph



Parse Graph

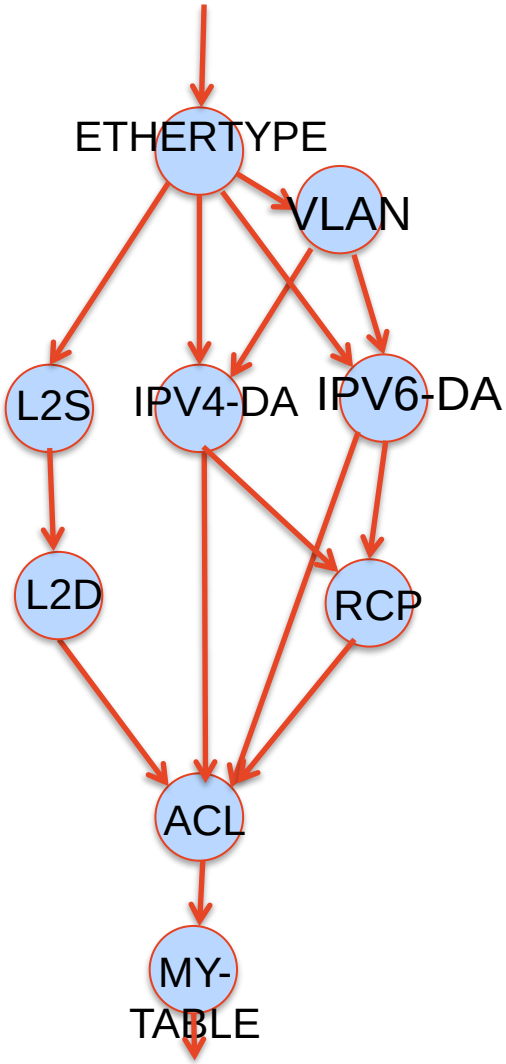
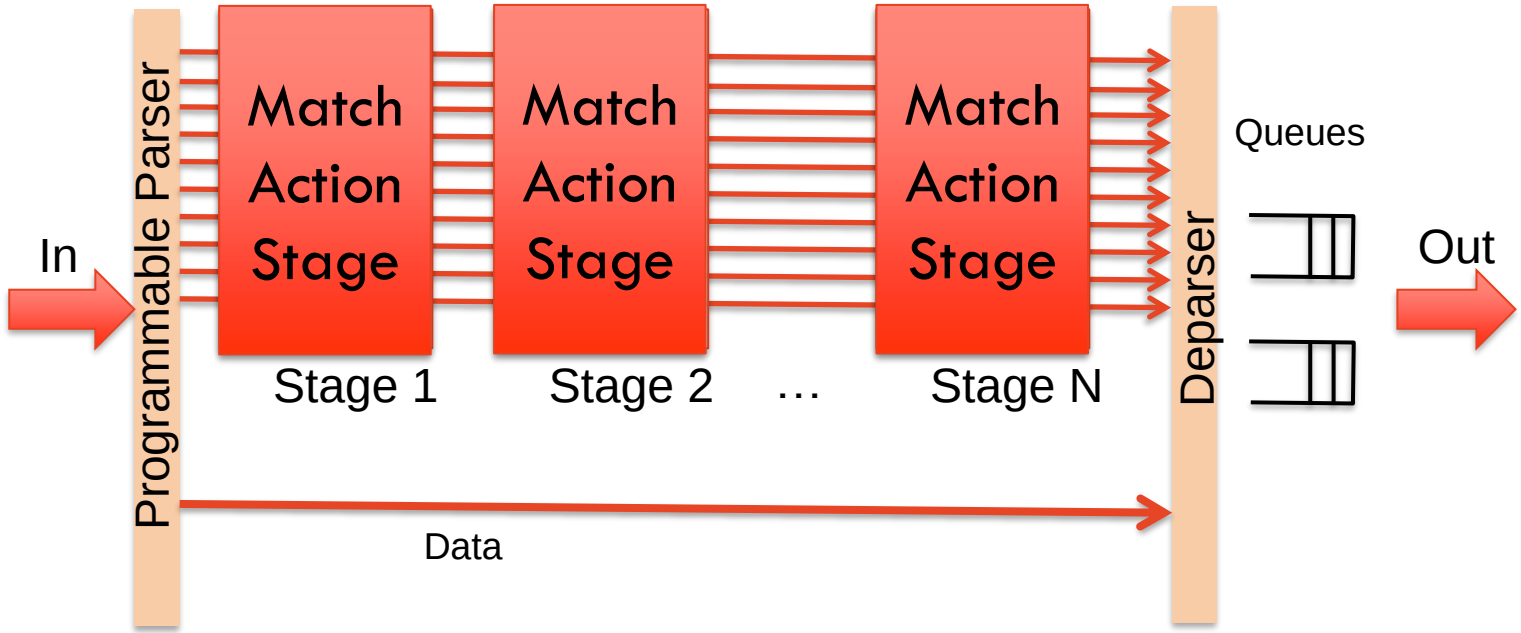


Table Graph

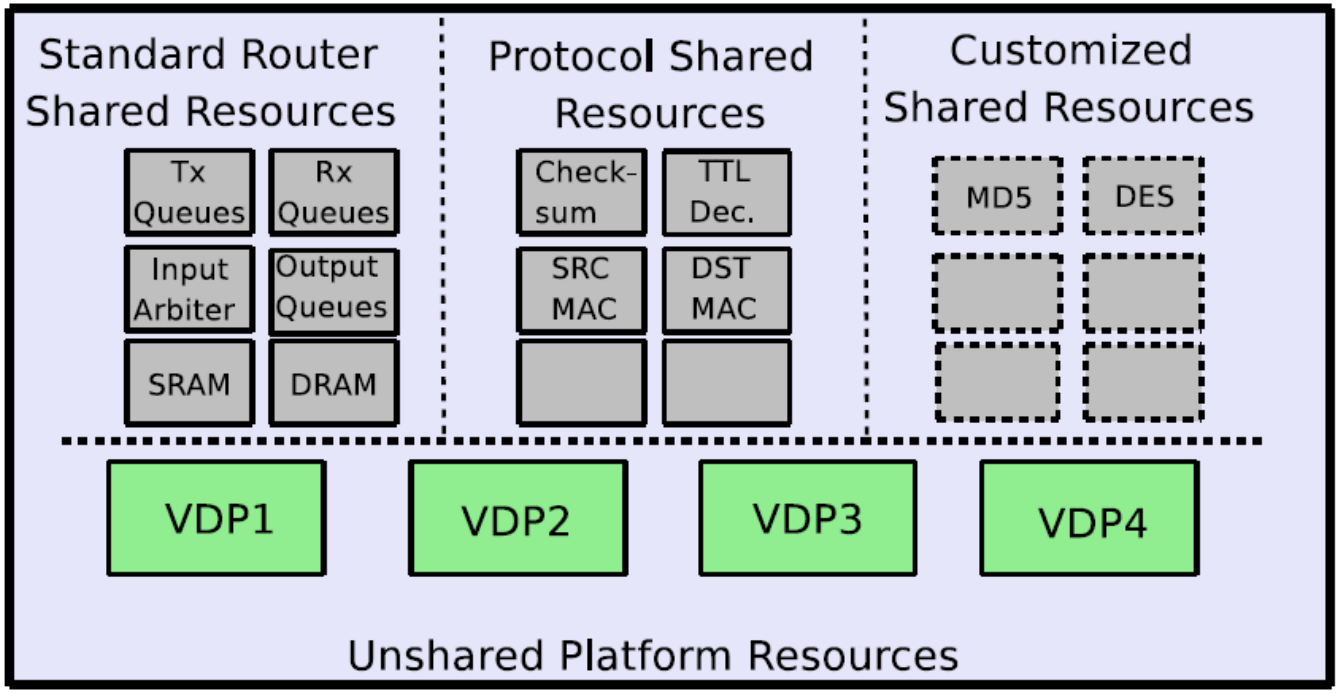
Match/Action Forwarding Model



SwitchBlade

- Anwer, Muhammad Bilal, et al. "Switchblade: a platform for rapid deployment of network protocols on programmable hardware." *ACM SIGCOMM Computer Communication Review*. Vol. 40. No. 4. ACM, 2010.

- Programmable, modularisable FPGA-based data plane



Main Idea

- Identify modular hardware building blocks that implement a variety of data plane functions
- Allow a developer to enable and connect various building blocks in a hardware pipeline from software
- Allow multiple custom data planes to operate in parallel on the same hardware

- Design the Virtual Data Planes with 4 stages
  - Virtual data plane: selection, shaping, processing, forwarding

Programming languages

- Network Assembly Language

- High Level Language

Network Assembly Language

- Openflow's design was motivated by the underlying device layout
  - Controller is limited in supporting new functions not supported by Openflow
- New Chipsets are adding data plane functions.
- New languages are specifying data plane at a high level – people like to use

- What's in between?

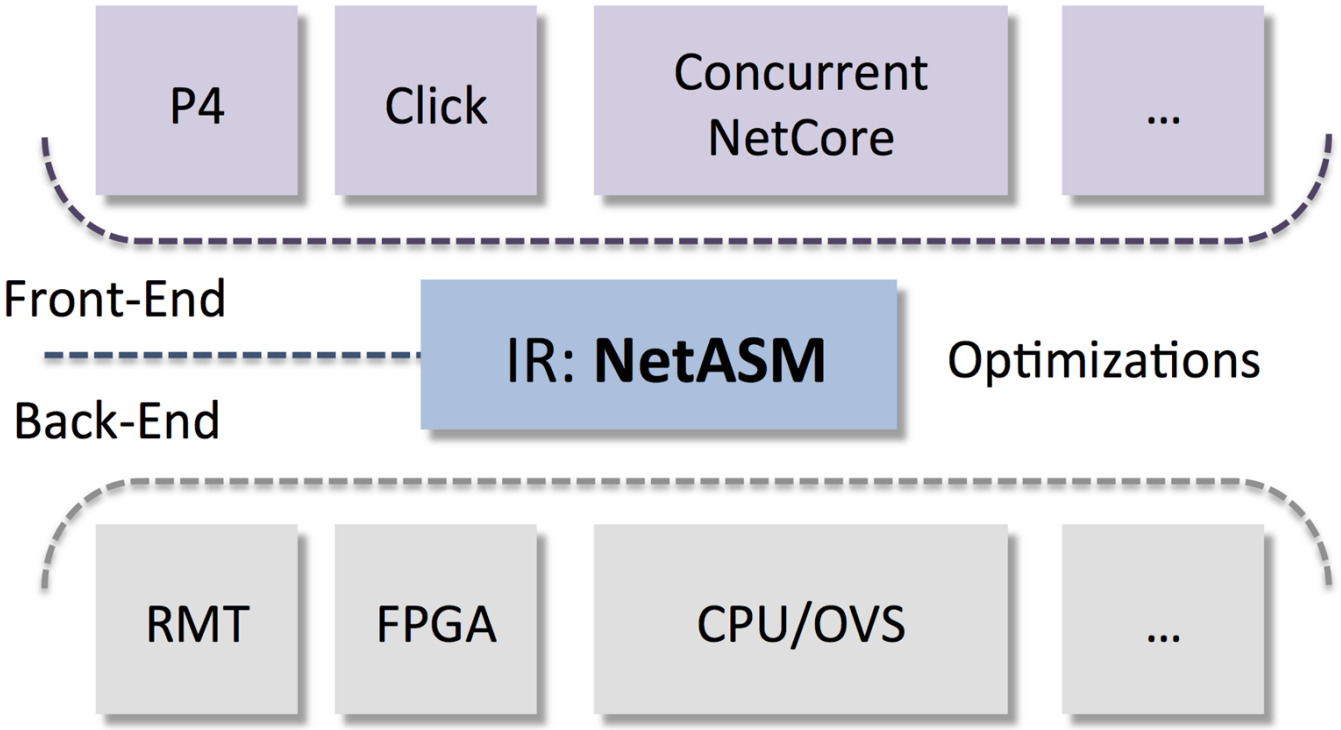
- http://netasm.cs.princeton.edu/

Need for network assembly

- A low-level programming language for programmable network devices
- Provides a one to one correspondence with the underlying hardware
- Uses well-defined constructs to define low-level packet operations
- Enables writing highly optimized network programs

NetASM: An Intermediate Representation

Enables a **common platform** for writing optimizations for programmable data planes



Protocol independence

- Compile from different languages
  - NetKat
  - P4
  - OpenState
  - OpenFlow
  - Flowlog
  - …

Target Independence

- Assemble for different Targets
  - FPGA
  - Click
  - GPU
  - Open vSwitch
  - …

High Level Language – P4

- Bosshart, Pat, et al. "P4: Programming protocol-independent packet processors." *ACM SIGCOMM Computer Communication Review* 44.3 (2014): 87-95.

- Protocol Independent packet processing

Desirable Features in SDN switches

- Configurable packet parser
  - Not tied to a specific header format
- Flexible match+action tables
  - Multiple tables (in series and/or parallel)
  - Able to match on all defined fields
- General packet-processing primitives
  - Copy, add, remove and modify
  - For both header fields and meta-data

New hardware makes this possible

- New generation of switch ASICs
  - Intel FlexPipe, FM Series
  - Broadcom BCM Series
  - RMT (SigComm'13)
  - Cisco Doppler

- But programming these chips is hard
  - Custom, vendor-specific interfaces
  - Low-level programming

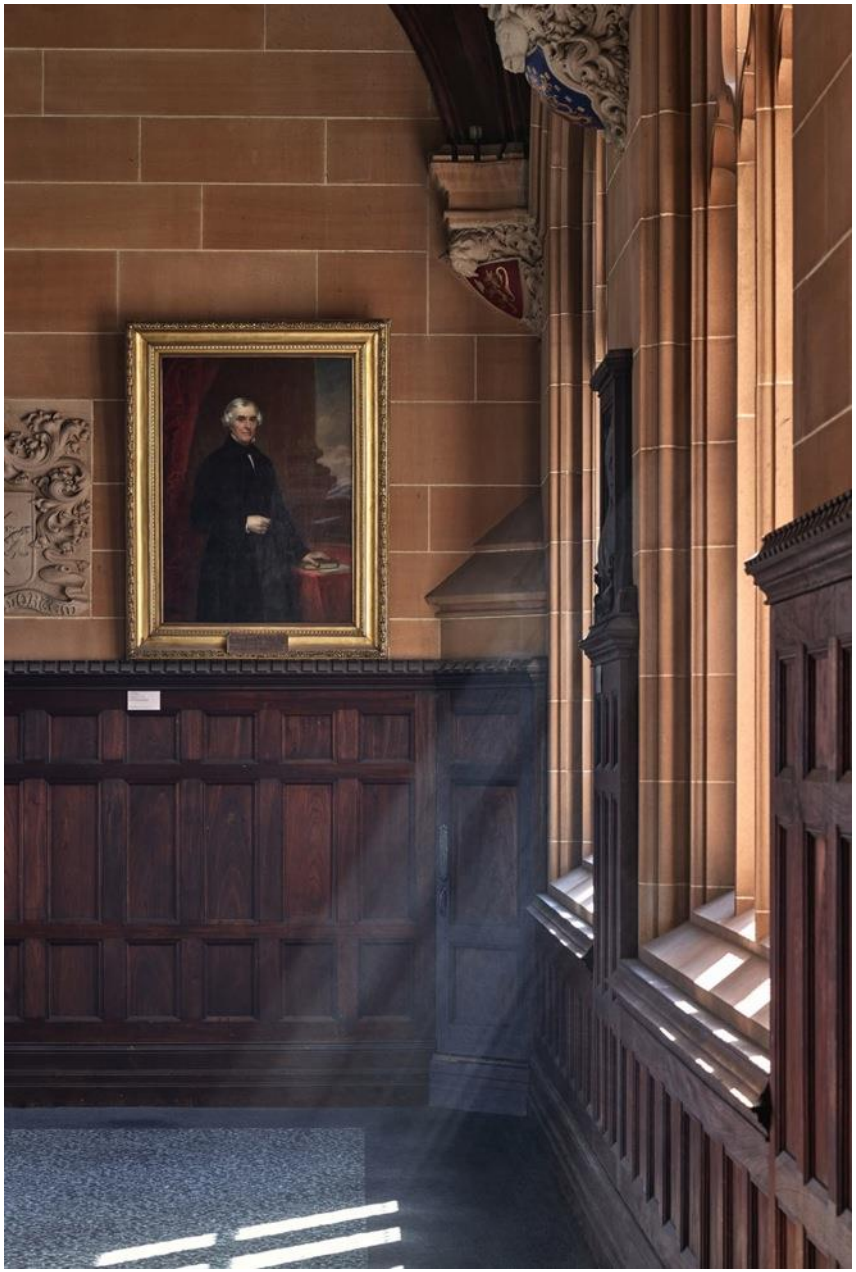
Thank you!

References:

https://www.scs.gatech.edu/news/195201/free-online-sdn-course

https://www.sdxcentral.com/sdn/?c_action=num_ball

https://www.opennetworking.org/



THE UNIVERSITY OF SYDNEY