MODERN ROUTER ARCHITECTURE
FOR PROTOCOL DESIGNERS

BRIAN PETERSEN

JOHN SCUDDER

JUNIPER NETWORKS

NOVEMBER 1, 2015

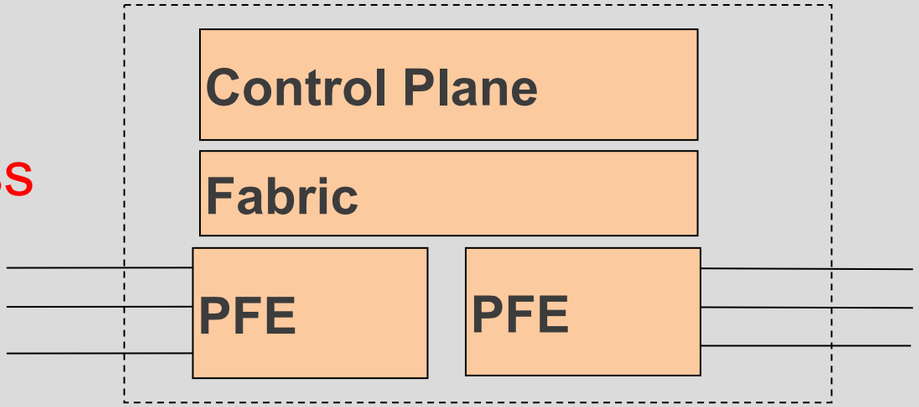
AGENDA

- Router Taxonomy
- Pipeline Characteristics
- Considerations for Protocol Designers

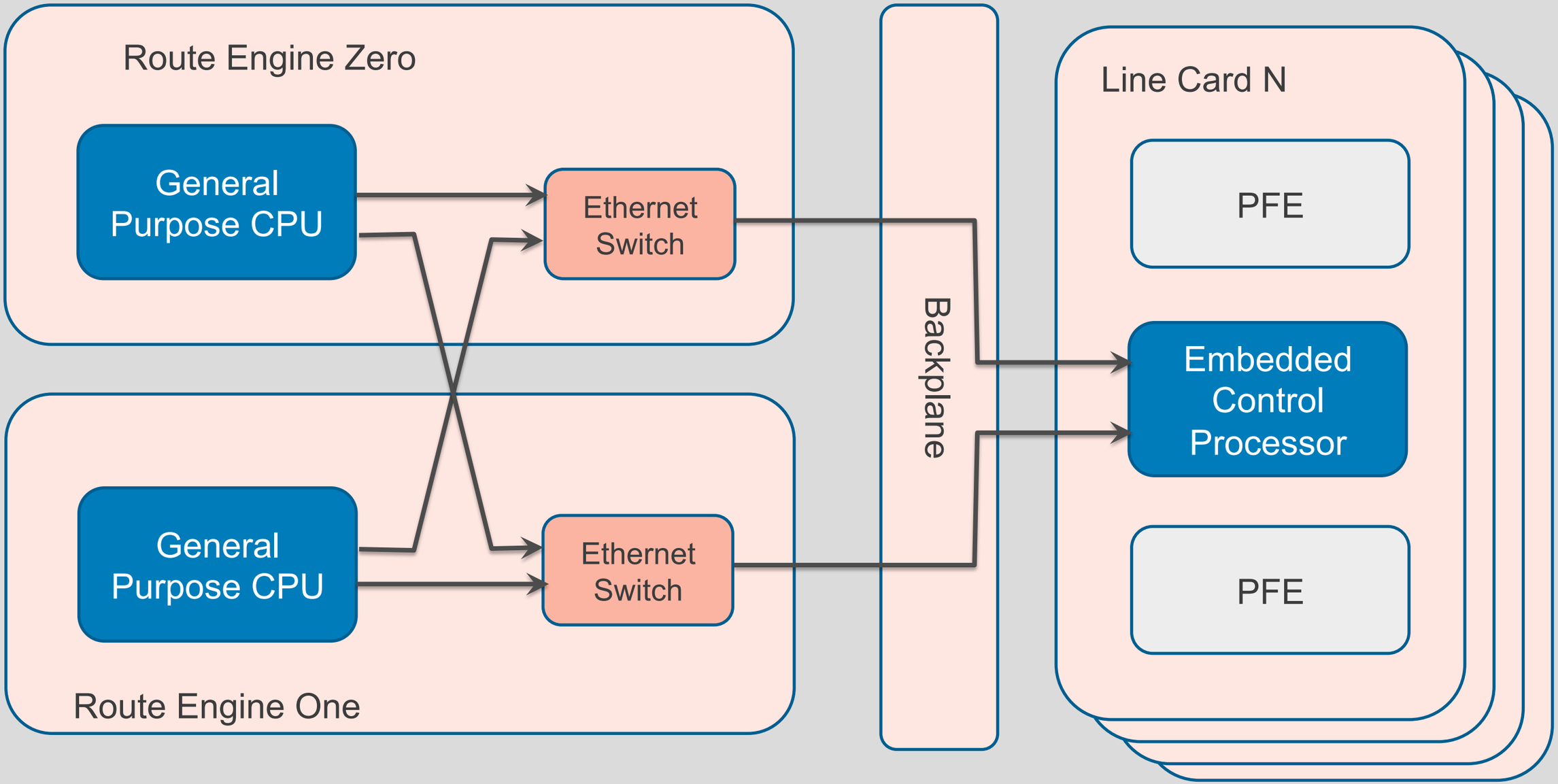
ROUTER TAXONOMY

KEY ELEMENTS

- High-scale routers comprise several key elements:
- Control Plane
  - Responsible for managing routing tables, authenticating subscribers, configuring interfaces
- Packet Forwarding Engine(s) (PFE)
  - Responsible for forwarding each packet (address lookup, queues, access lists, etc)
- Fabric
  - Responsible for moving packets from one line card to another inside the router

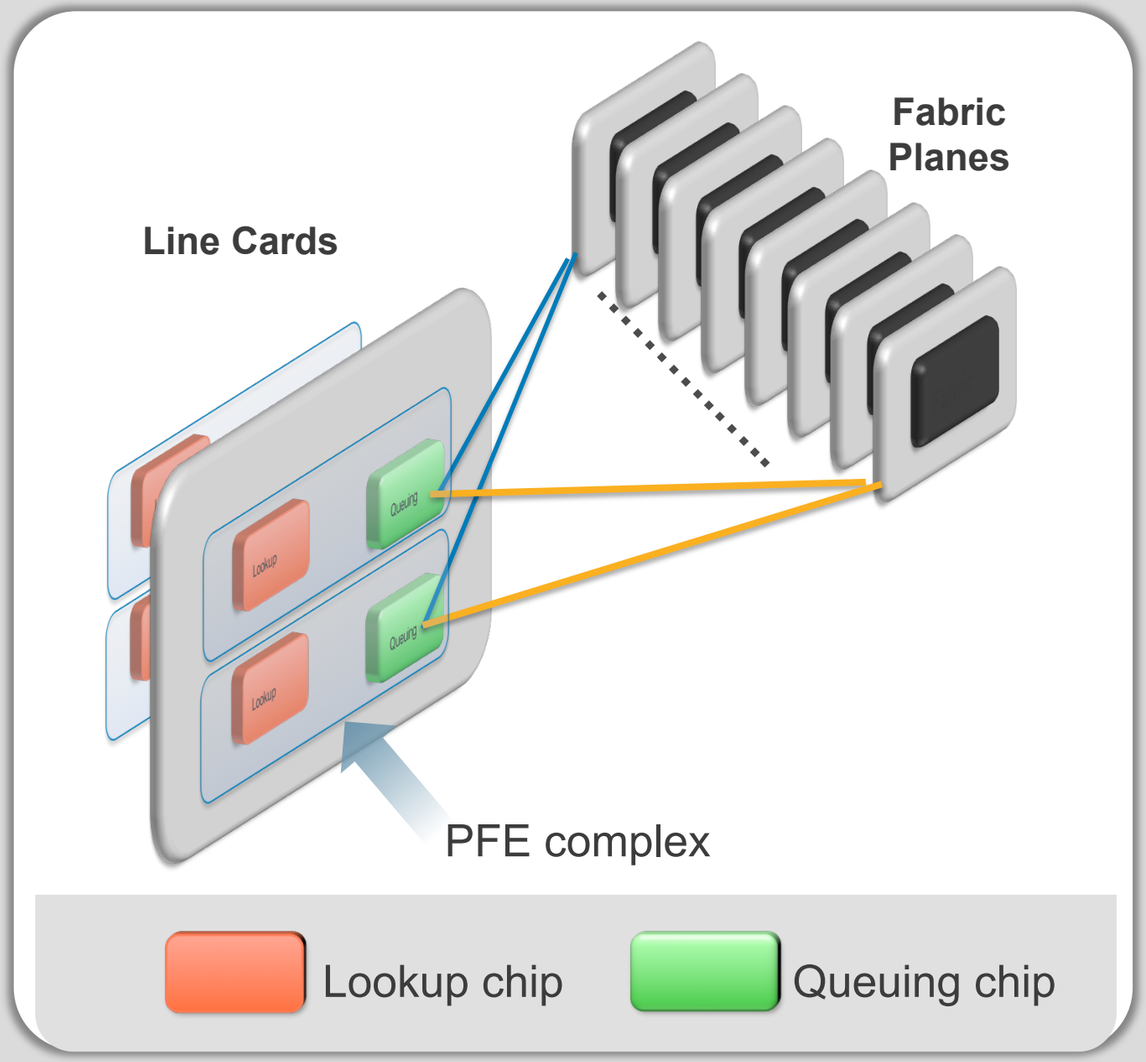


ROUTER CONTROL PLANE



FABRIC-BASED ARCHITECTURE

- Most high-scale routers are fabric-based
  - Multiple line cards, each containing PFEs
  - A chassis-wide interconnect fabric transfers traffic from ingress to egress line cards

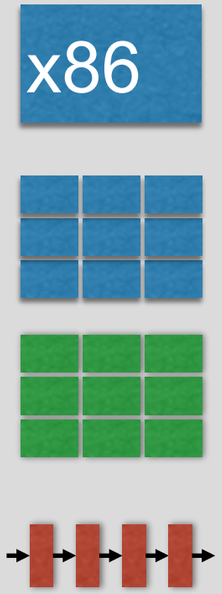


PACKET FORWARDING ENGINE (PFE)

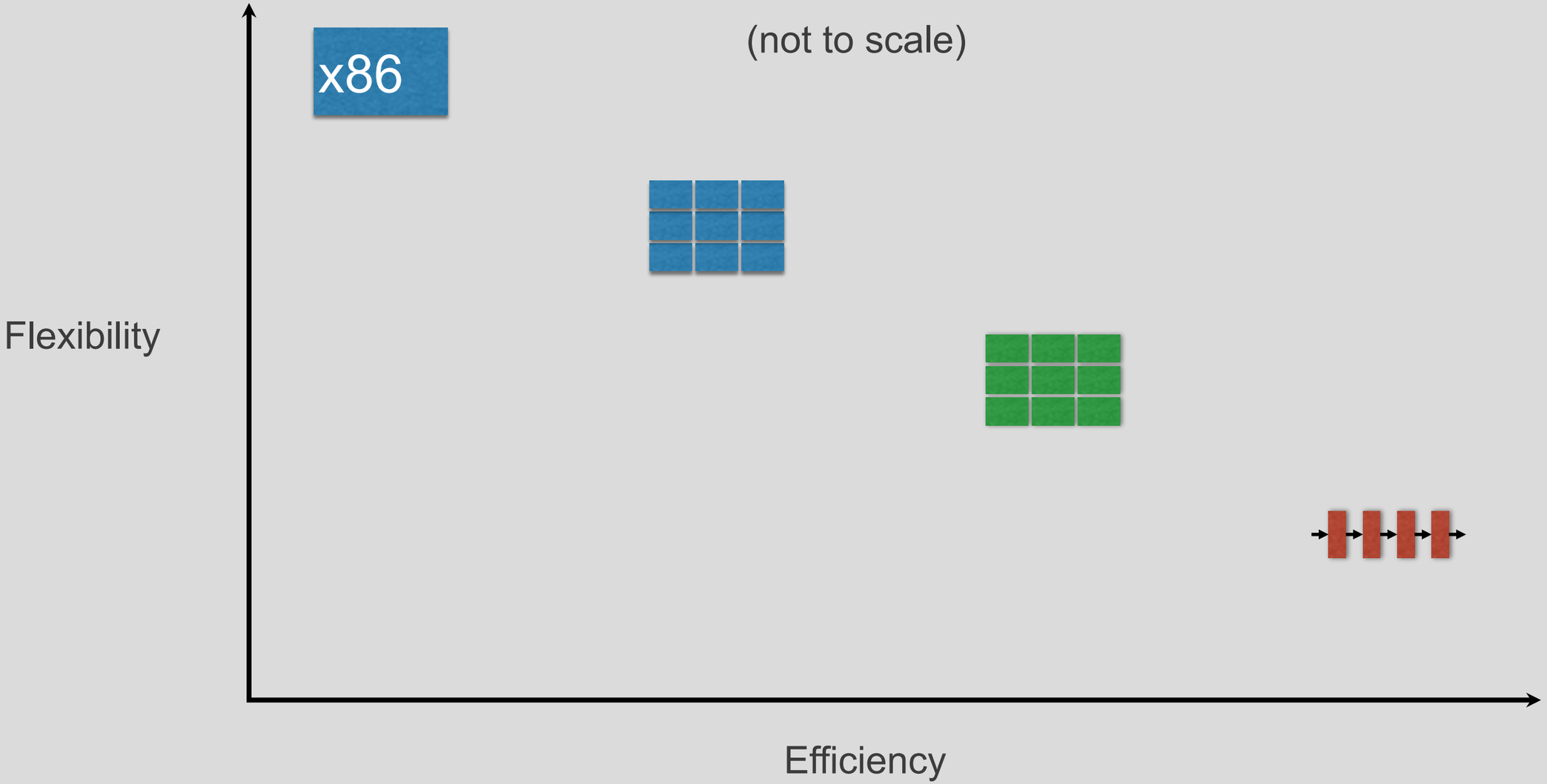
- PFEs do the work to move packets from ingress to egress
- Key functions:
  - L2 & L3 analysis & features
    - Figure out whose packet it is, what should happen to it, where it should go.
  - Packet buffering
    - Store the packet in DRAM until there's room to transmit it
  - Queuing & scheduling
    - Decide which packets should go in what order to achieve fairness and real-time delivery guarantees.
- PFEs may be micro-programmable, table-driven or hard-coded
  - It's the old cost/performance/flexibility tradeoff matrix…

SOME ROUTER ARCHITECTURE TYPES

- General-Purpose Processor
- Sea of General-Purpose Processors
- Sea of Special-Purpose Processors
- Pipeline



TRADEOFFS



HIGH-SCALE ROUTERS VS. GENERAL-PURPOSE COMPUTERS

- Traditional computer architectures (e.g., x86) are “infinitely” flexible
  - … at a cost
- High-performance routers trade flexibility for other important attributes
  - Example tradeoff: Access to packet Data
    - General-Purpose Processors are presented with a buffer containing an entire packet
    - Pipeline (et al) are presented with the first *n* bytes of a packet
- The trick is to only trade away flexibility you didn’t need anyway
  - But predicting the future is hard (“wait, you want to look *how* deep?”)
  - This is where protocol designers can help

WHY PIPELINES?

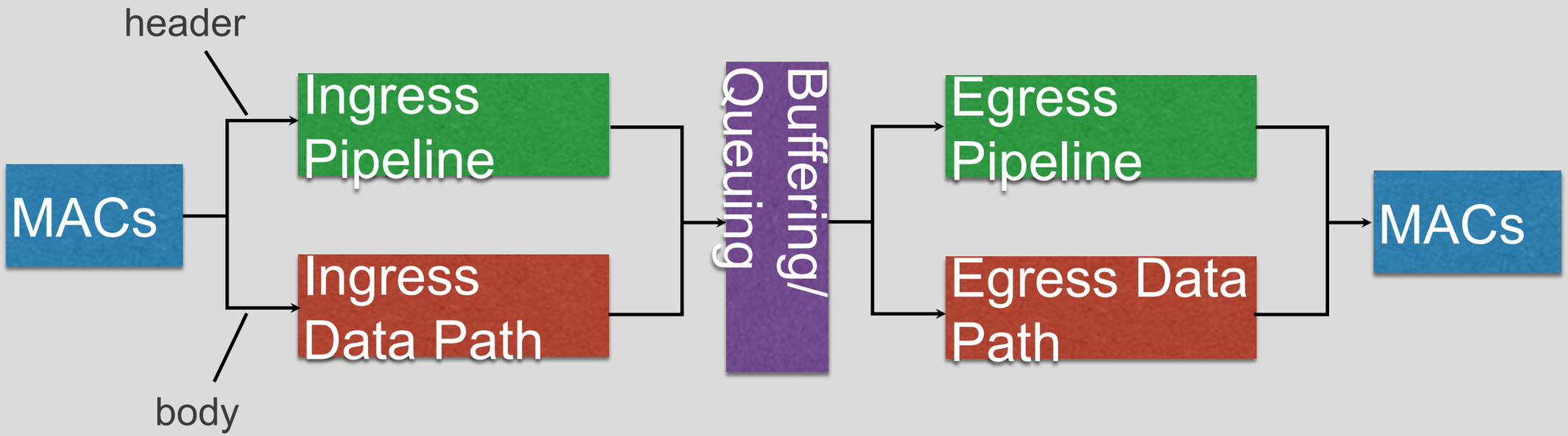
- If pipelines are limited, why bother?
- In a word: efficiency
  - Operations per packet per Joule is far higher
  - Throughput per unit volume is far higher
  - It is not uncommon for a pipeline to sustain 500M–1B packets per second
  - That’s 50–100 times faster than an x86

SO WHAT?

- Okay, pipelines are a thing. Why should I care?
- Because of their efficiency, pipelines are in widespread use
- Pipelines have particular characteristics
- Certain protocol design characteristics are awkward fits

PIPELINE CHARACTERISTICS

PIPELINE ROUTER BLOCK DIAGRAM



HEADER VS. BODY

- Processing
  - Lots of stages
  - Variable delay, looping, out-of-order execution, etc.
  - Expensive (power, area, complexity) to transport long header chains
- Data Path
  - RAM-based
  - Optimized for temporary storage of variable length byte strings

PIPELINE STAGES

- Pipelines break down processing into bite-sized chunks
  - For example:
    - Parsing, receive context, destination lookup, etc.
- Each stage performs a specific operation and delivers results to the next stage
- Ideally, the stages work on header data sequentially

INSTRUCTIONS VS. GATES

- Instructions live in RAM and are infinitely flexible, but relatively slow
- Logic gates are fast and massively parallel
- Complex logic and math can operate at a blazing speeds
- But… gates are hard-wired and cannot be changed
- Replacing logic gates with RAM-based tables enables flexibility, but decreases efficiency

HEADER VOCABULARY

- Pipelines generally have a fixed header vocabulary
- Accommodating new ways of stacking and using headers is okay
- Accommodating entirely new headers may require new silicon

AGAIN, SO WHAT?

- Protocol and header design greatly impacts hardware design
- Certain protocol design choices increase hardware complexity and delay adoption
- Given the ubiquity of pipeline-based designs, it behooves the protocol designer to consider their characteristics to help foster widespread adoption and deployment.

CONSIDERATIONS

HEADER SIZE LIMITS

- Pipelines generally split header data from packet bodies
- A long series of large headers may not fit in the allotted space

HEADER SPACING

- Fixed header lengths make parsing easier
- Fields lengths aren’t an issue (up to a point), but making header lengths multiples of 32 bits improve hardware efficiency

RESPECT THE HIERARCHY

- Ideally, headers are processed in the order that they appear in the packet
- Out-of-order header processing adds significant hardware complexity
- Processing that spans multiple headers increases complexity and eliminates opportunities for optimizations
- Make headers self-contained units of information
- Specifically disallow using fields from other layers in processing the current layer

USE EXISTING HEADERS

- Whenever possible, use an existing header instead of inventing a new one
- However, avoid using well-defined fields in ways that are unrelated to the header’s original definition

ALLOW PARSING WITHOUT LOOKUPS

- It should not be necessary to perform a large-table lookup in order to fully parse a packet's headers
  - Of course lookup is still needed for forwarding!
- A next-header identifier value should be globally defined and not variable based on the forwarding context
- Provide reliable next-header type information in each header
- Disallow multiple definitions for the same constant value

FLOW IDENTIFICATION

- Flow identification is necessary for ECMP and LAG load balancing
- Make it clear which fields are reliable for flow identification
- Provide a robust means for carrying flow entropy fields to obviate deep parsing

CHECK VALUES

- Make checksums optional/experimental
- Don’t have header checksums span multiple headers
- Avoid checksums that span the entire packet
- Don’t require nested CRCs

IN CONCLUSION…

- Quite simply, don’t assume complete processing flexibility.