Technology Deep Dives

An Experiment

Deep dives?

- A discussion on a particular wide topic.
- Focus on concepts, not necessary on the details.
- More on architecture and design.
- Less on implementation and deployment.
- Network engineering minus vendor marketing.

This is an experiment

- We need your feedback.
- Hallway discussions indicated that this may be interesting.
- Community education.

Modern Router Architectures

A fairy tale of spherical routers in vacuum

Credits: John Scudder, Frank Brockners, Toerless Eckert, Brian Petersen, Alvaro Retana, Spencer Dawkins, Warren Kumari

What?

- A brief look at how a (modern) router operates.
- Intentionally substantially simplified view.
- Focus on core principles and not on details.
- This is not a tutorial on how to build a router.
- This is not a tutorial on how to operate a router.

Why?

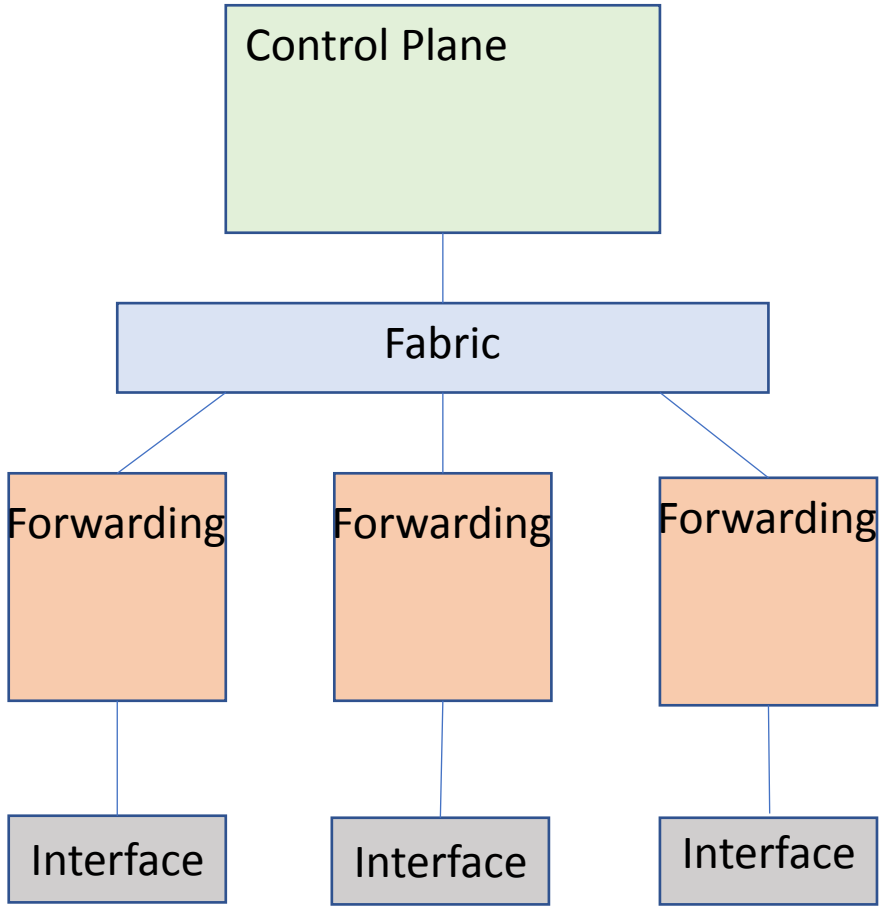
- Misconceptions of what a router does.
- Misconceptions of how a router does it.
- Misconceptions related to Software Defined <whatever>.

- Similar content was presented at IETF 104 – with a quite positive feedback.

- Community education.

Conceptual components

- Control plane complex.
- Forwarding complex.
- Interconnection fabric.



Control Plane Complex

- Control plane complex.

  Control plane protocols and exception path processing.

- Also can be used for forwarding – at a huge cost and performance/functionality hit.

Forwarding Complex

- Actual packet forwarding – move from ingress to egress.
- Flexibility varies greatly (x86 to custom purpose-built)
- 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 buffer memory until there's room to transmit it
- Queuing & scheduling
- Decide which packets should go in what order to achieve fairness and delivery guarantees.
- Forwarding components may be micro-programmable, table-driven or hard-coded
- It's the old cost/performance/flexibility trade-off matrix.
- Forwarding components may be totally integrated (the features, buffering, and scheduling may all be on a single chip) or they may be separated into different physical devices

Interconnection Fabric

- Responsible for moving packets from one forwarding component to another.

- Fabric within a single entity (fabric on a chip).
- Fabric between different entities

    Between components on the same linecard.

    Between linecards.

    Between shelfs.

Hardware vs software way of thinking

- Software is sequential – lookup this, then lookup that, then write that.
- Hardware can be parallel – lookup this and lookup that and write that all at the same clock cycle.

Hardware vs software way of thinking

- Wait – processors have many cores and software has threads?

  True. This does not mean free speedup though.

- Parallel software is not easy.
- Parallel hardware is expensive.

Specialized vs general purpose processors

- Traditional computer architectures (e.g., x86) are “infinitely” flexible – but 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 some 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

HW and SW interaction

- Programmable processors need instructions -> memory access.
- Programmable processors need data to work on -> memory access.
- Programmable processors threat packet as a buffer -> easy (but not necessary cheap access).
- Fixed pipeline processors do not need instructions.
- They still need access to data.
- And are (severely) limited by what they can access.

HW and SW interaction

- Instructions live in memory 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 memory-based tables enables flexibility, but decreases efficiency

HW and SW interaction

- Pipelines break down processing into bite-sized chunks

  Parsing, receive context, destination lookup, etc.

  Each stage performs a specific operation and delivers results to the next stage

  Stages can be made more complex - at the expense of trading speed for complexity

  Ideally, the stages work on header data sequentially

  Out-of-order parsing or loops in the parse tree (e.g. overloaded next header type in MPLS) can pose significant challenges

HW and SW interaction

- Forwarding component design is always a set of tradeoffs between complexity and speed.
- Processing with a programmale instruction sets are really flexible, but also really slow.
- Processing everything in a tightly integrated pipeline is really fast, but much less flexible (or totally inflexible at the end-case)
- In reality, all NPU designs are a series of tradeoffs trying to optimize around some very precious resources.
- Transistor counts: Every gate used for processing a packet cannot be used for another interface (i.e. a port-density for features tradeoff)
- Memory depth and access rate – to process a billion packets per second where each requires 10 lookups means 10 billion memory lookups per second.

HW and SW interaction

- If dedicated processing pipelines are limited, why bother? In a word: efficiency
- Operations per packet per Watt is much higher.
- Throughput per unit volume is far higher
- It is not uncommon for a pipeline to sustain 500M–1B packets per second.
- Which is 50–100 times faster than a commodity x86 platforms can do.

Power concerns

- Power is a hard requirement.
- Power in (current) and power out (cooling).
- Physical limitations tend to be stronger than logical limitations.

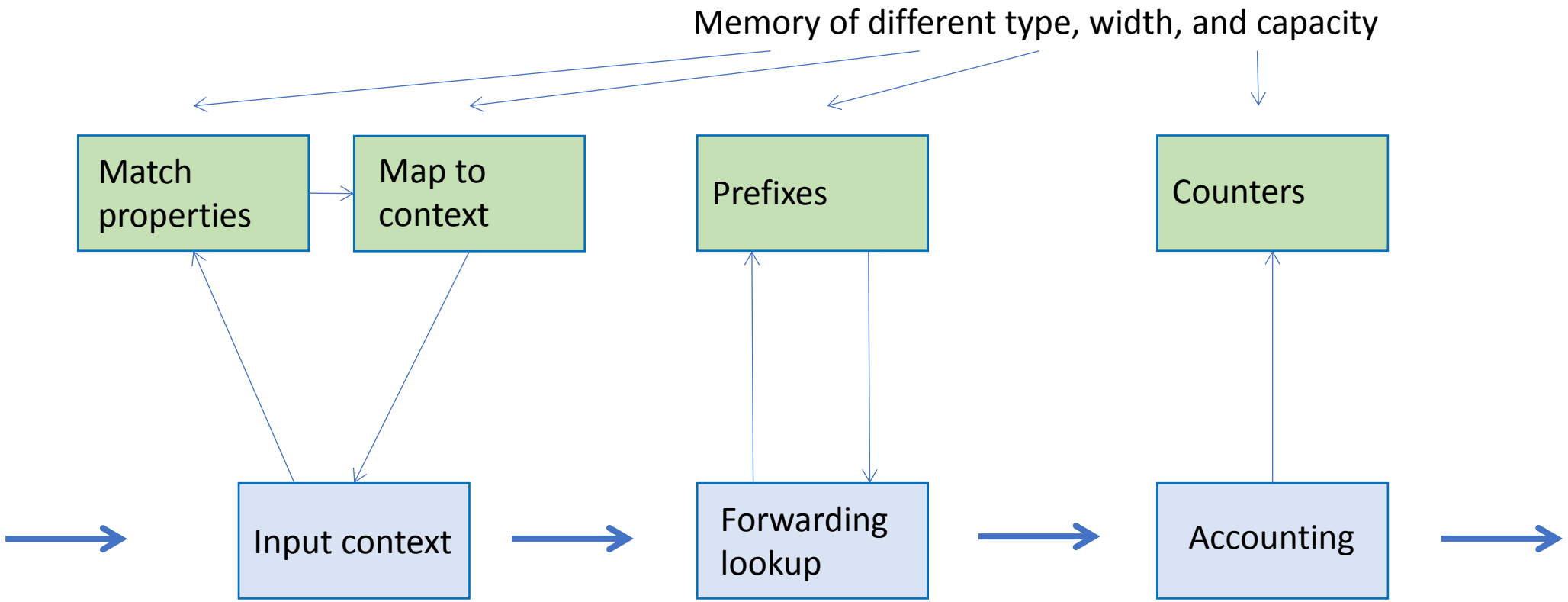
Gates

- Any hardware functionality requires gates.
- Laws of physics apply to gates.
- Throughput vs logical complexity.

Memory

- Memory has finite access speed.
- Random vs local vs sequential access.
- Memory size vs access speed.
- MPLS direct lookup is practical. Hierarchical MPLS lookup is tricky though.
- IPv4 direct lookup is feasible, but not practical.
- IPv6 direct lookup is just not feasible at all.
- Memory power considerations.
- Memory width vs memory speed.
- Different memory types for different tasks.
- Control plane data input and output.

Memory subsystem



Electrical Interface

- Memory and fabric interfaces, host CPU interface, assorted support circuitry – all that are electrical signals.
- Several thousand of signal pins per component package are not uncommon.
- There are practical limitations related to power feed, signal integrity, and package physical dimensions.
- Single component platforms have rather low physical scalability limits.

NPUs

- A rather abused term.
- Ranging from rebranded generic processors to highly customized and highly specialized non-programmable processing components.
- NPU does not equal to unlimited packet processing performance.

Spherical Routers

Open Discussion

Technology Deep Dives

Open Discussion

Technology tutorials

- Focus on specific area of technology.
- Focus on architectural aspects and protocol mechanics – not vendor implementation aspects.

- Possible examples:
- How BGP and global routing works.
- How L3VPN works.
- How DNS works.

Discussion

- What do you think?
  - Of the concept
  - Of the topic

- Did you see any value?
- Wishlist for other topics?
- Wishlist on format?
- Is RIPE meeting a right venue for this?