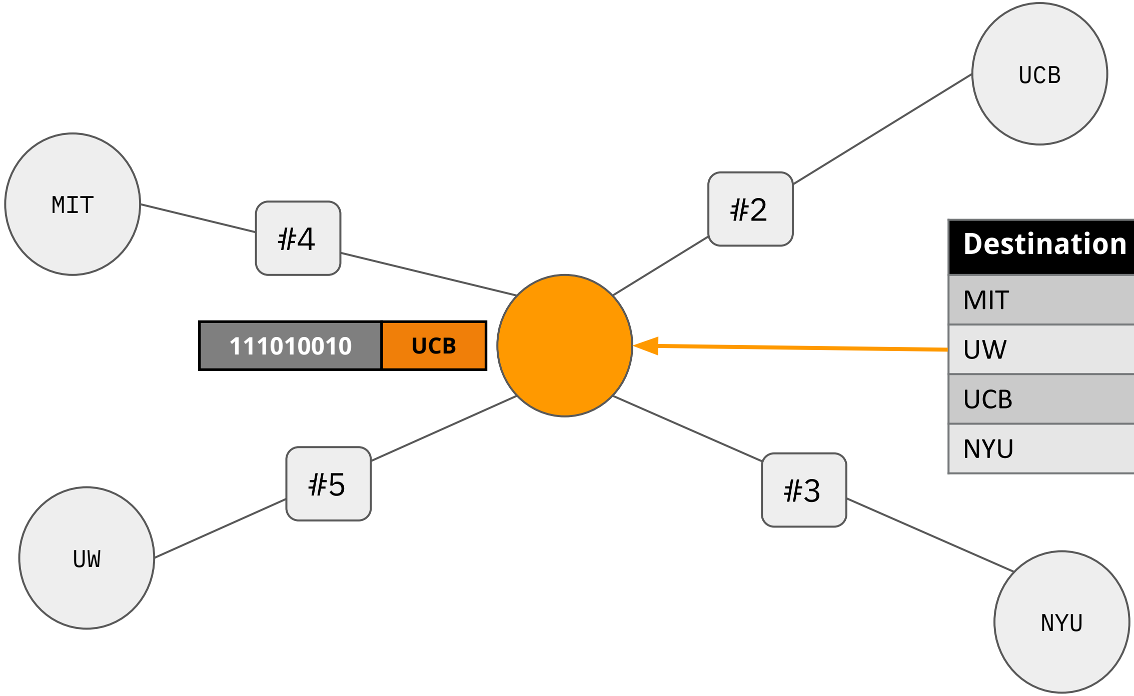
IP Routers

Rob Shakir (robjs@google.com)
Autumn 2022
cs168.io

Recall: IP router purpose.



Destination	Next Hop Link
MIT	Link#4
UW	Link#5
UCB	Link#2
NYU	Link#3

Recall

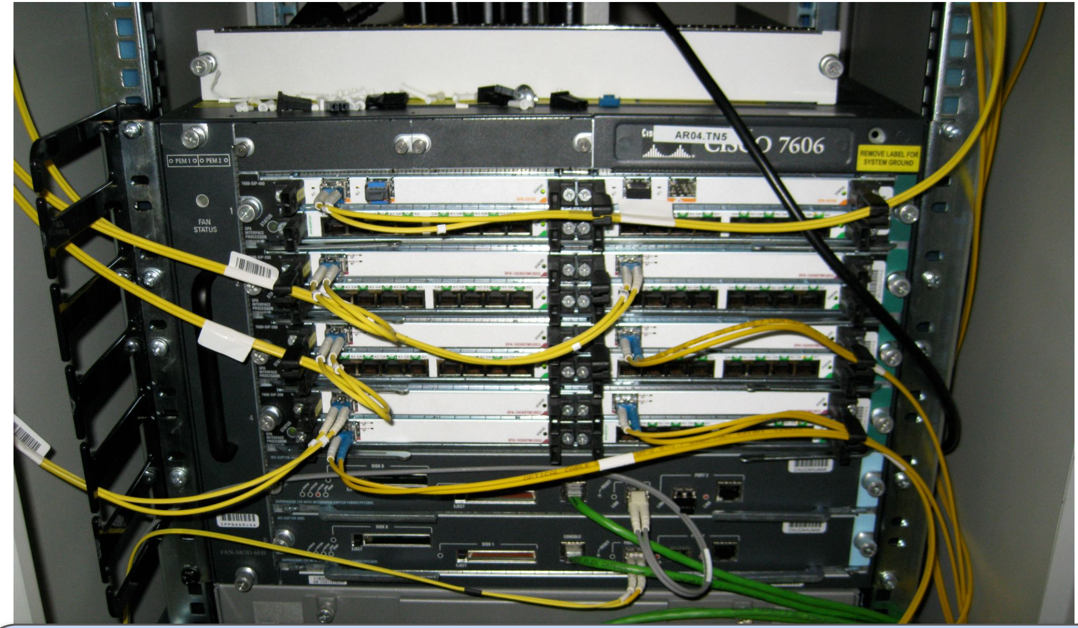
- A router performs IPv4/IPv6 lookup against the destination IP of a packet.
- Routers run *routing protocols* to learn about routes.
- “Routes” are sets of destination IP addresses.

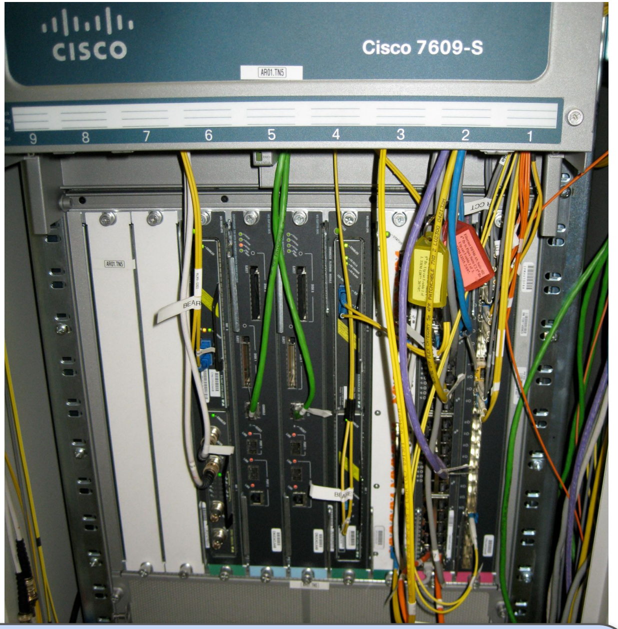
- **Today: What *is* a router?**

What makes up the Internet?

Colocation facilities: Datacenters housing multiple Internet Service Providers.
Many routers from different companies!

IP routers?





Computers specialised for forwarding packets.
Different sizes and configurations depending on requirements.

Different Sizes of IP Router.

Dimensions:
- Physical size
- Number of ports
- Bandwidth

Router Definitions

- N = number of external ports.
- R = speed (“line rate”) of a port
- Router Capacity = N x R

- N = 4, R = 100Mbps
- N = 1, R = 1Gbps
- Total: 0.4+1 = 1.4Gbps.

Today's capacity.

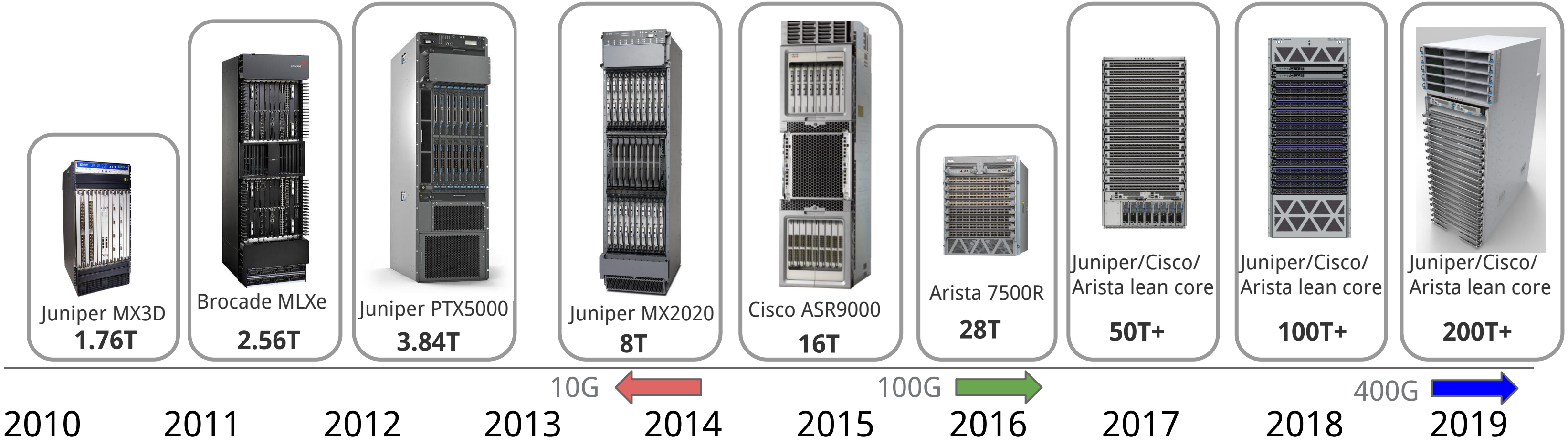
Today (400G linerate)

- 8 linecards, 36 ports each.
- N = 8 x 36 = 288
- R = 400Gbps
- Router Capacity = 288 * 400G = 115.2Tbps

Next Gen (800G linerate)

- 8 linecards, 36 ports each.
- N = 8 x 36 = 288
- R = 800Gbps
- Router Capacity = 288 * 800G = 230Tbps

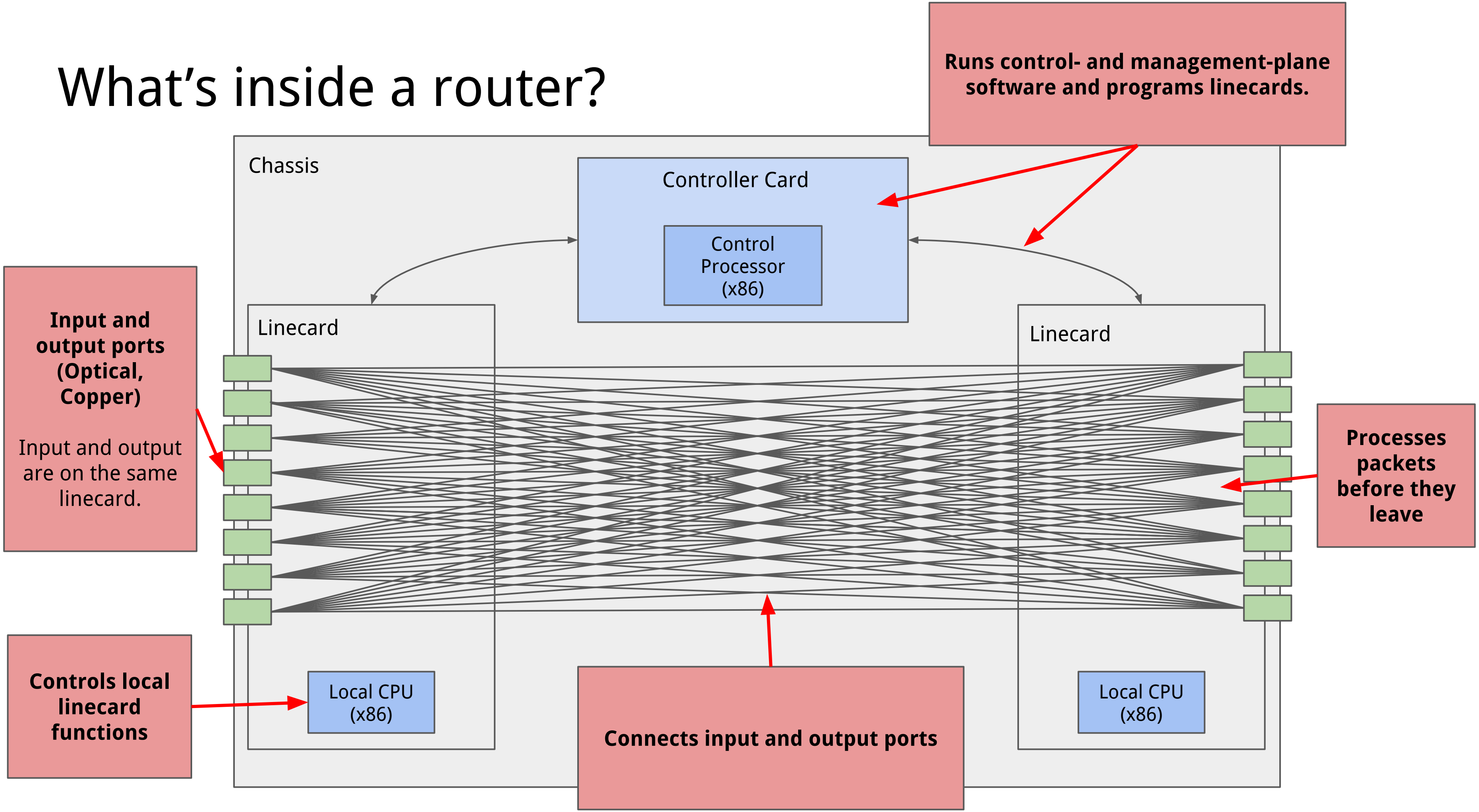
Evolution of Capacity…



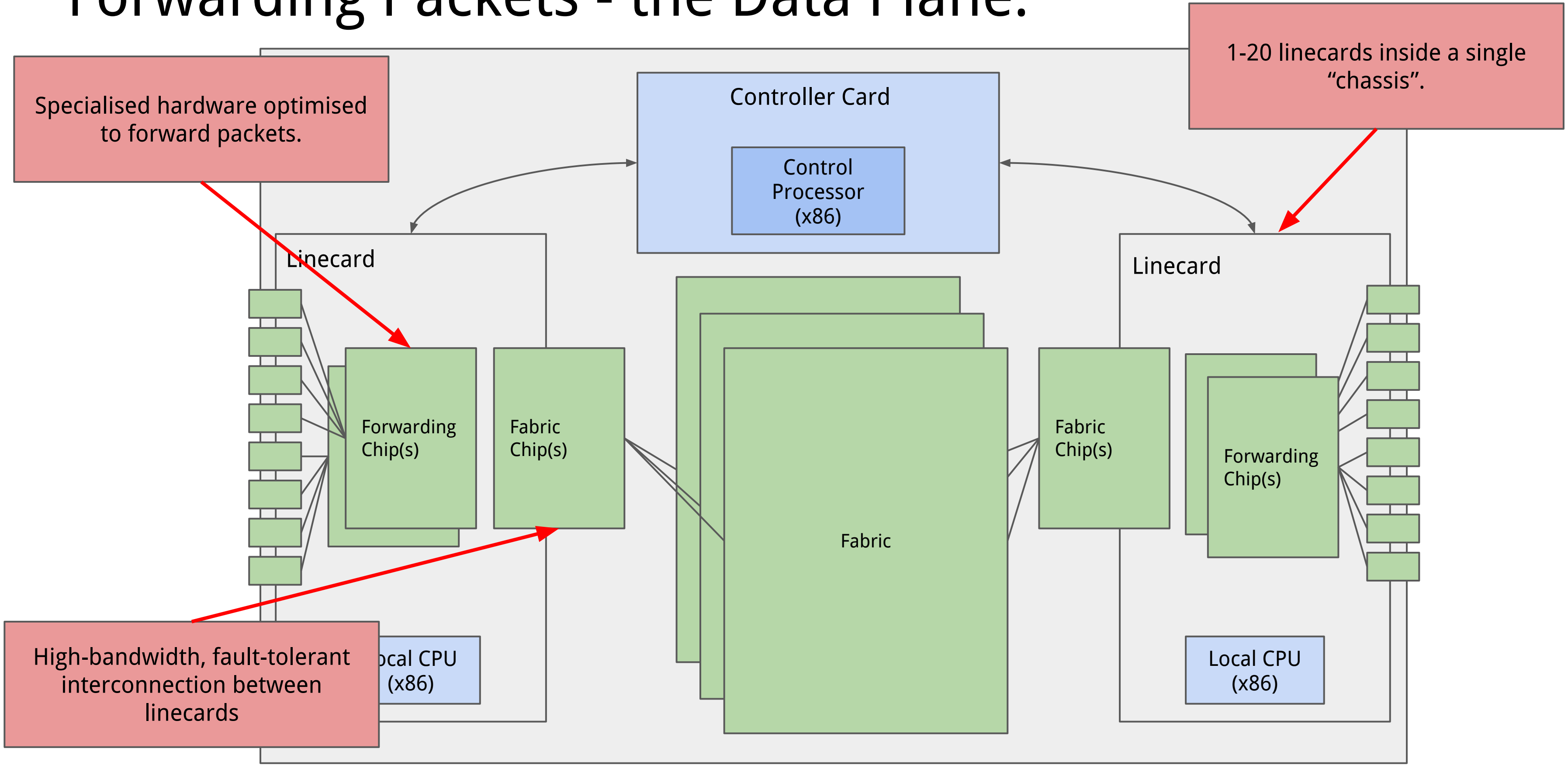
Note:

- Physical size (constrained by racks!)
- Impact of link speed (10G → 100G → 400G)

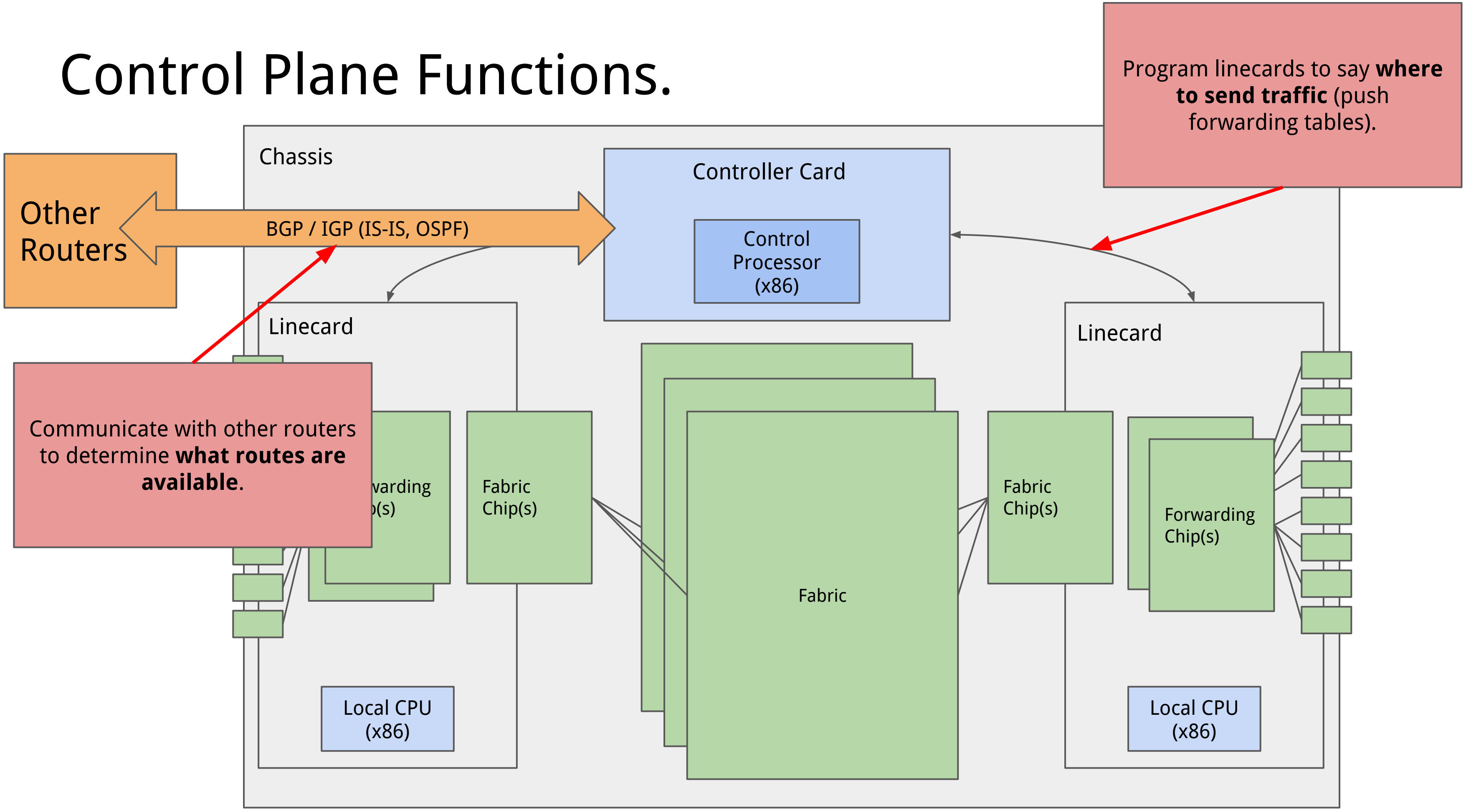
What's inside a router?



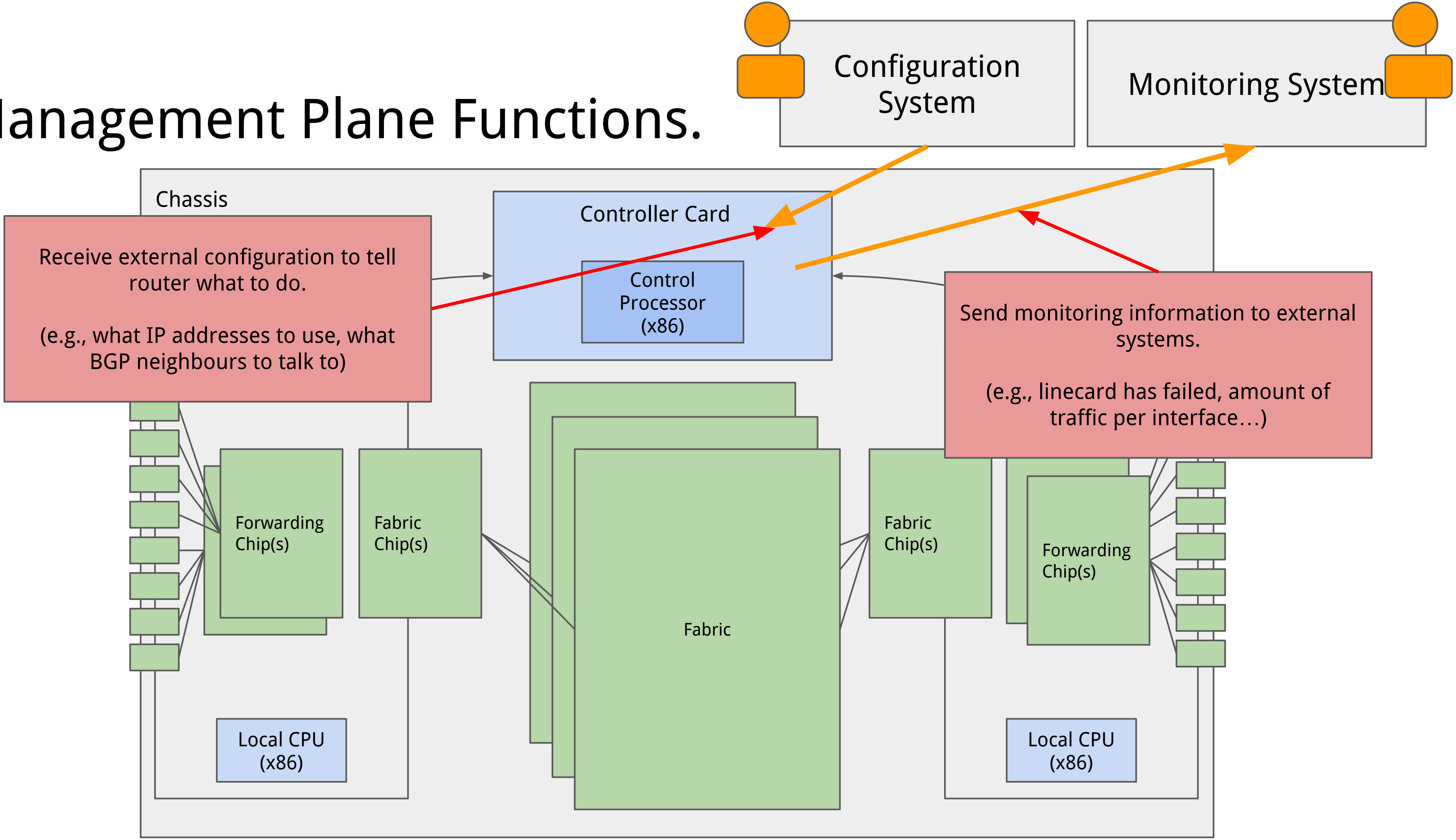
Forwarding Packets - the Data Plane.



Control Plane Functions.



Management Plane Functions.



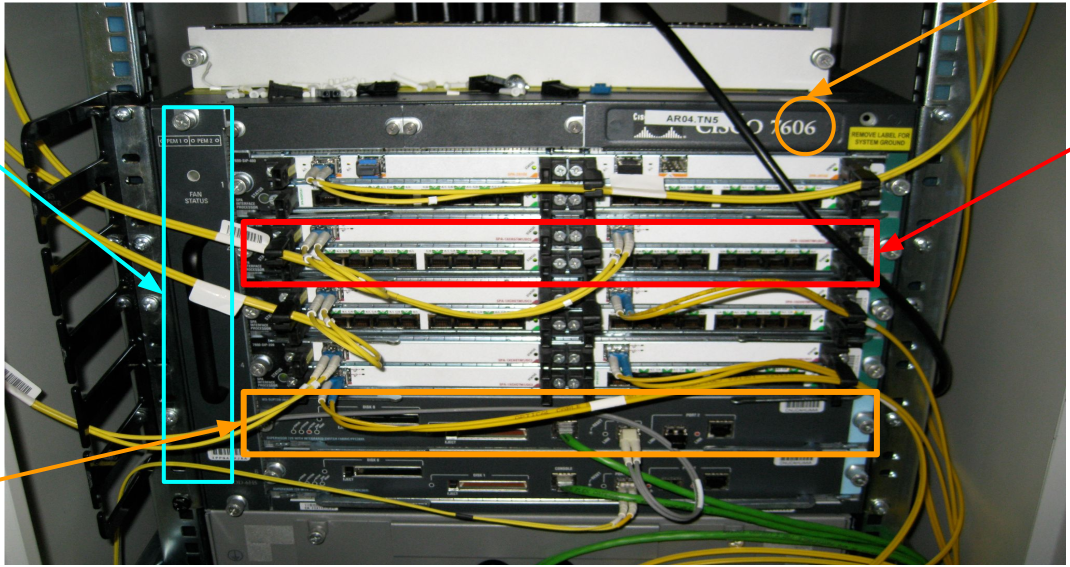
Looking at a real router.

Fan Tray

Controller Card

"06" - 6 slots, 2 controllers + 4 linecards

Linecard



A small cluster of computers ~~Computers~~ specialised for forwarding packets.

Definitions

Control Plane

- Runs routing protocols to allow router to understand *where* to route packets.

Management Plane

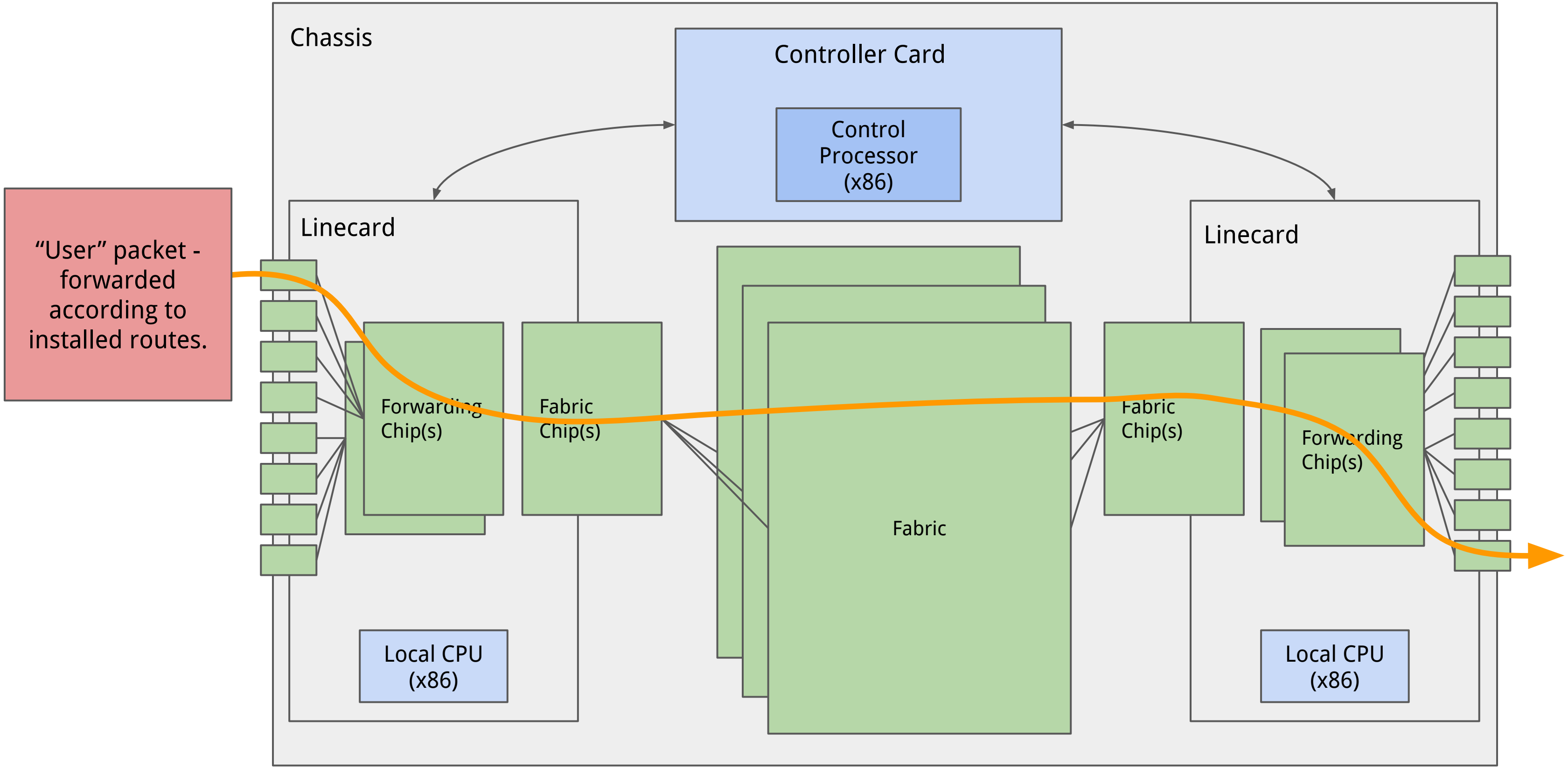
- Interacts with systems and humans to configure and monitor the device.

Data Plane

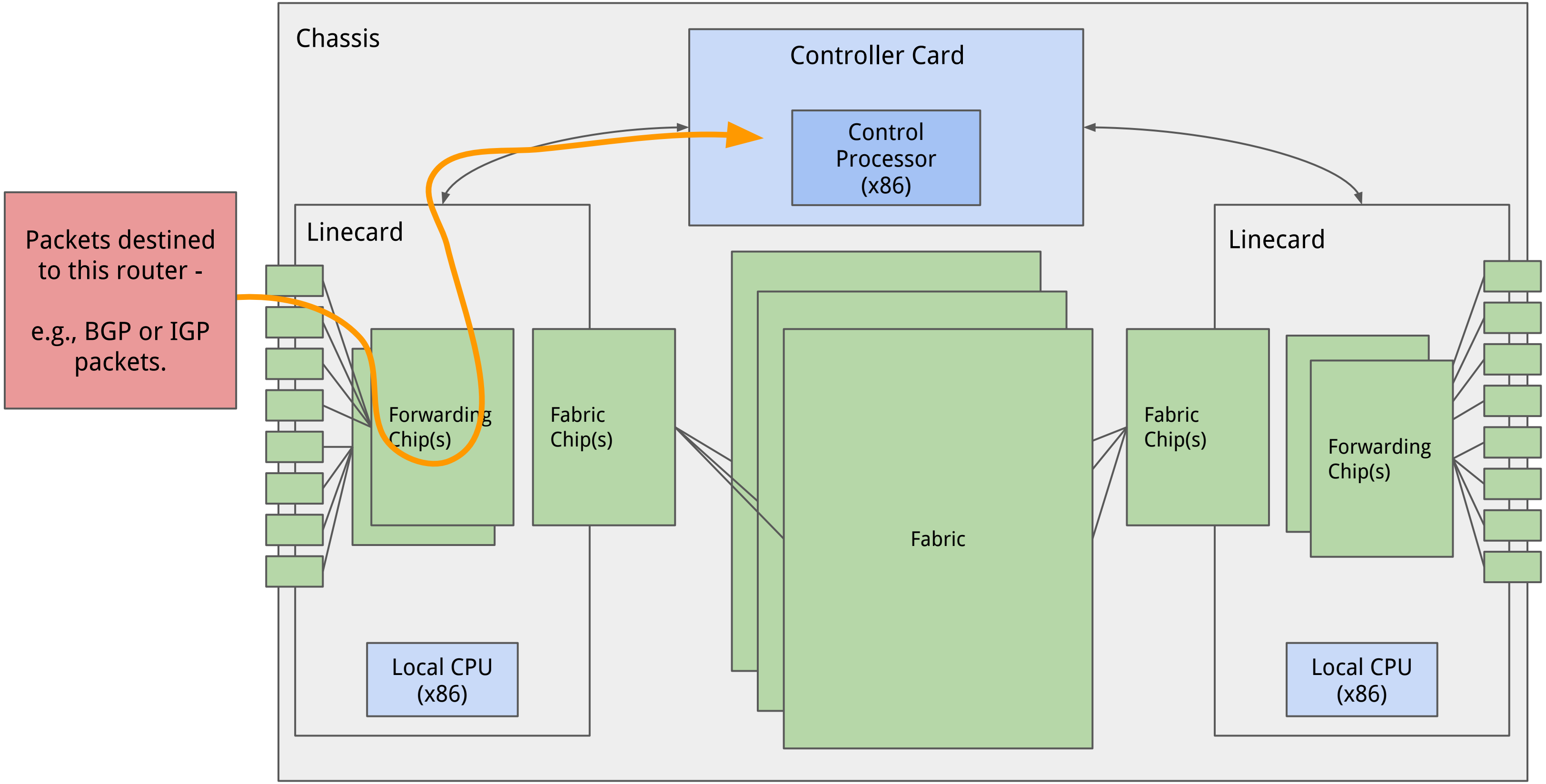
- Forwards packets.

We need all these to run a router in a real network!

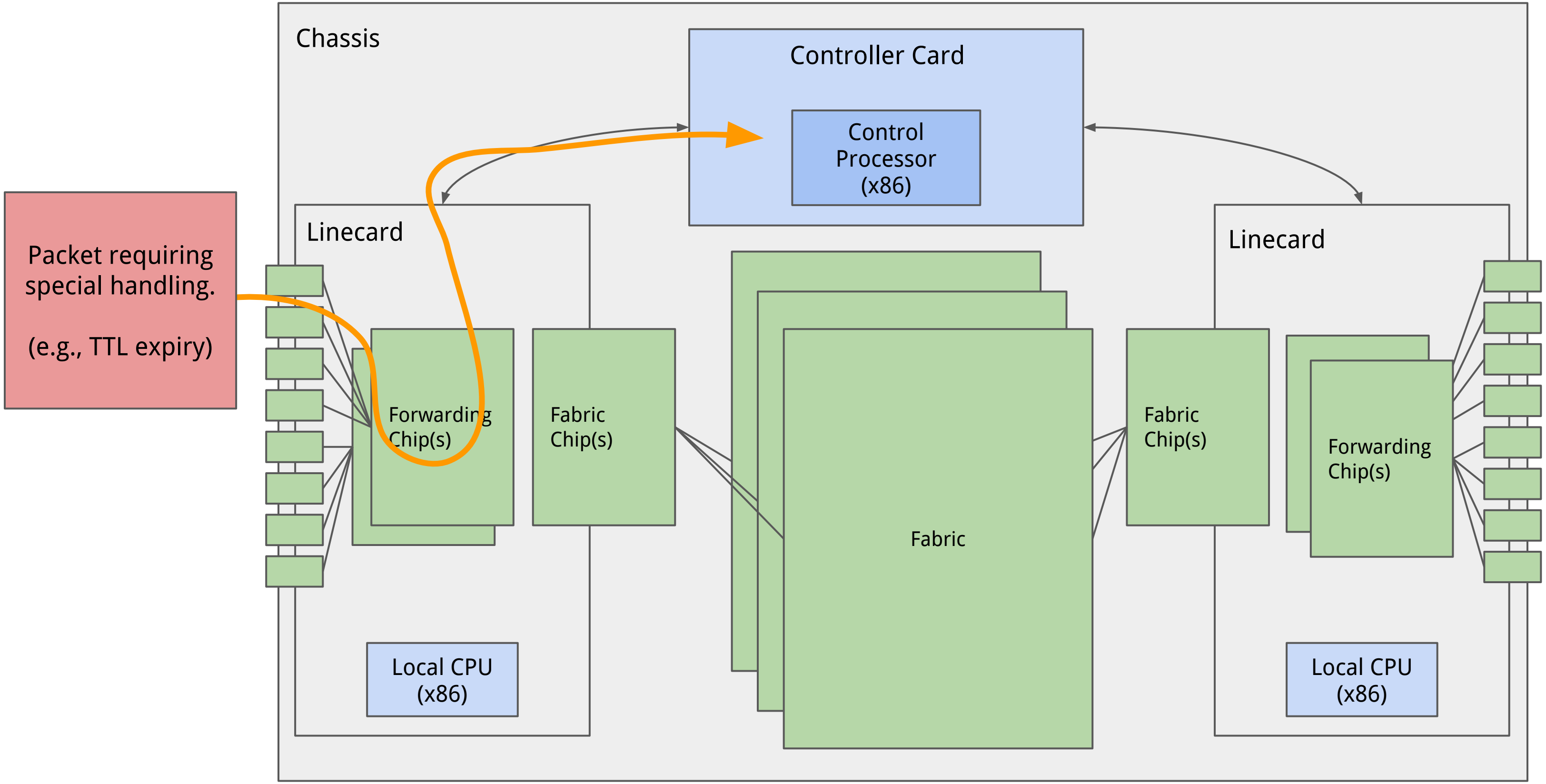
Types of Packets - “User” Traffic.



Types of Packets - Control Plane Traffic.



Types of Packets - “Punt” Traffic.

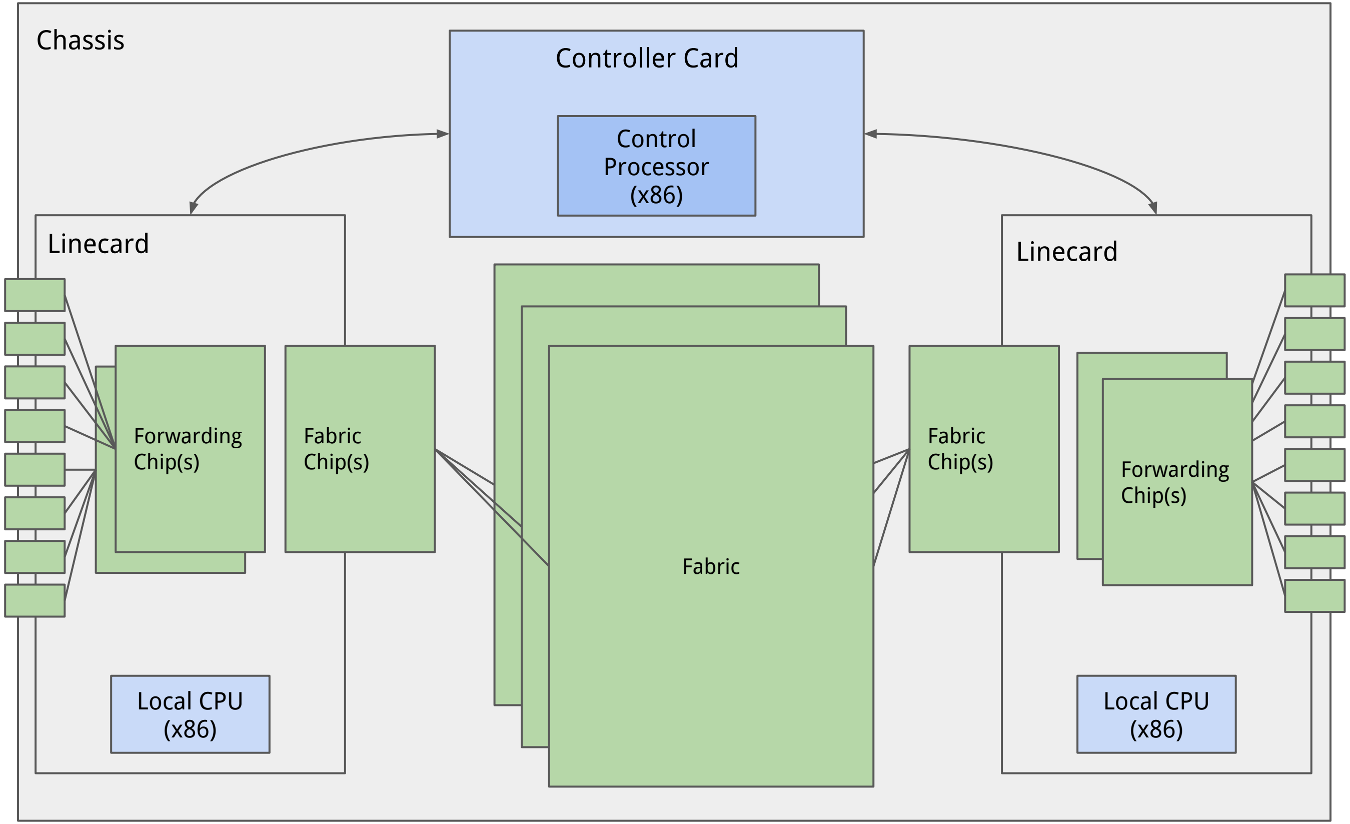


Why this architecture?

- Smallest Ethernet packet = 64 bytes.
- Current interface speed = 400 gigabits per second.
- $4x10^{11}$ / 64*8 = $781.25x10^{6}$ packets per second per direction.
- $1.5625x10^{9}$ packets per second x 36 ports = $56.25x10^{9}$ pps.
  - In practice a little lower… but a lot!

- Not achievable on a general purpose CPU.
  - ~millions of packets per second are.
  - “Slow path” used only when necessary.
- Forwarding hardware is the “fast path”.
  - Much more efficient (power, cost).

Any questions?



What does the input linecard do?

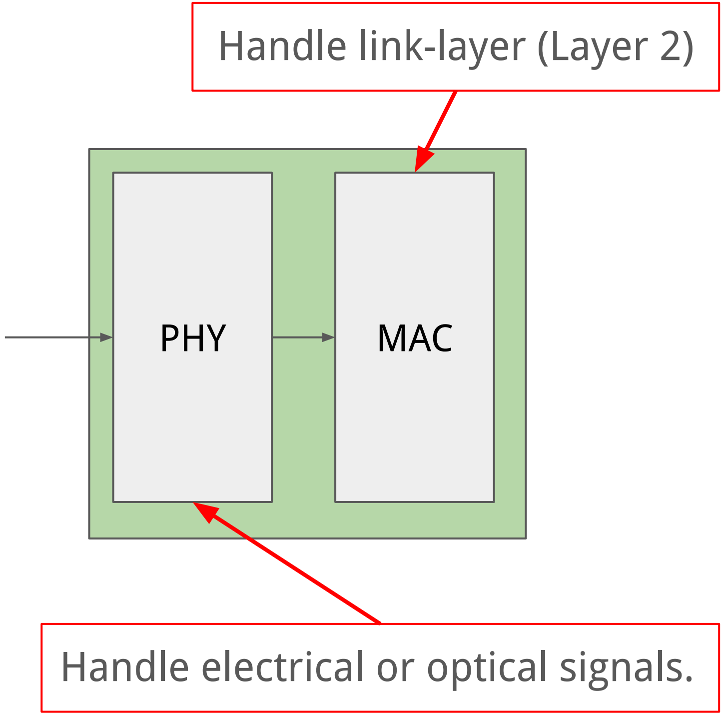
- Tasks
  - Receive incoming packets from other systems
    - Handle the physical layer (electrical, or optical) **- PHY**
    - On-the-wire encoding (Ethernet) - **MAC**
  - Update the IP header
    - TTL, checksum, options, fragment
  - Perform lookup for forwarding.

Challenges for Input Linecards - Speed!

- We talked about packet rate - even with 250 byte packets - 1 packet per 5 nanoseconds.

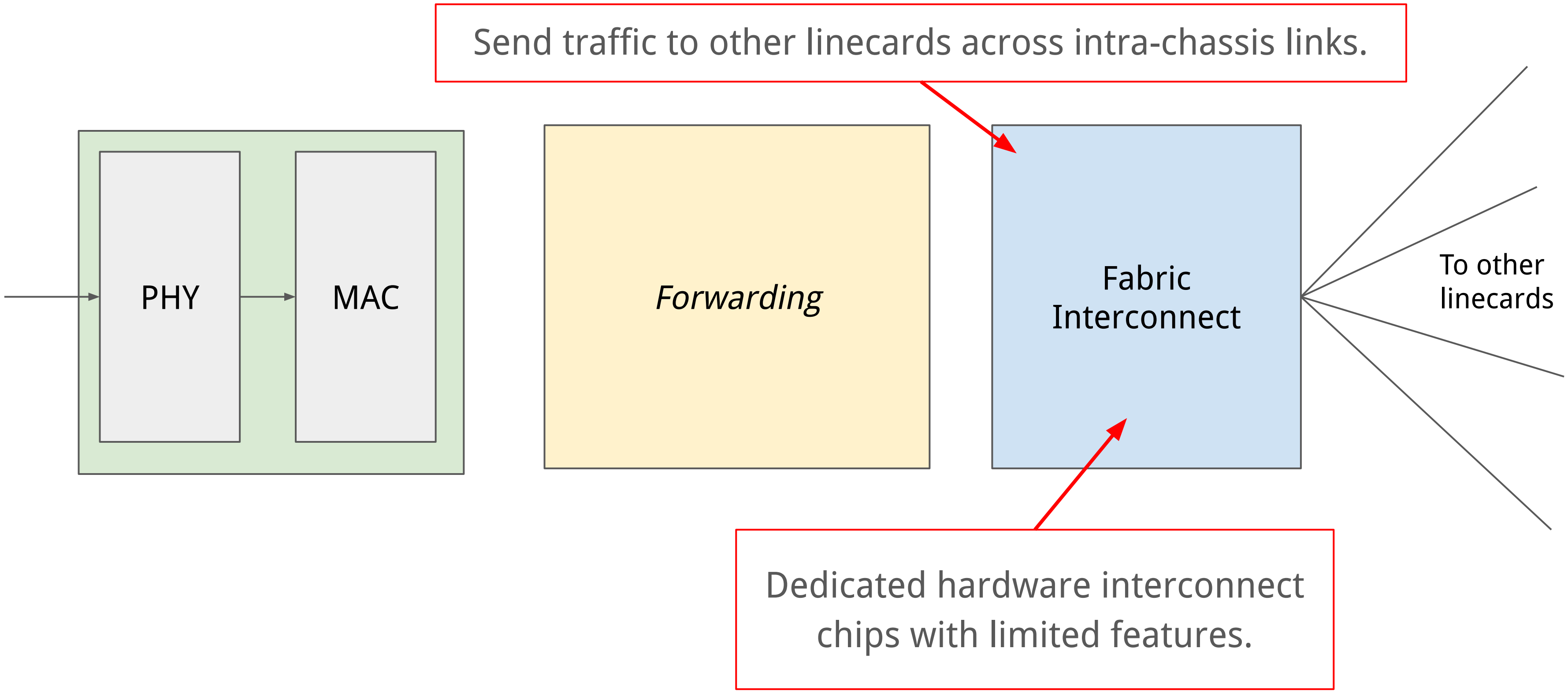
- Need to run at 0.2GHz <u>for each port</u> even with (the ideal of) one cycle per packet - but, we need to do multiple operations on each packet, and have many ports per chip.
  - Could we parallelise? Lots of CPU == lots of power.
  - Typically have specialised **network processors** - with some programmability, but with <u>limited functions</u>.
  - Special processing that can’t be done there done at control processor (per linecard or central).

“Pipeline” For Packet Forwarding: Layer 1 + 2

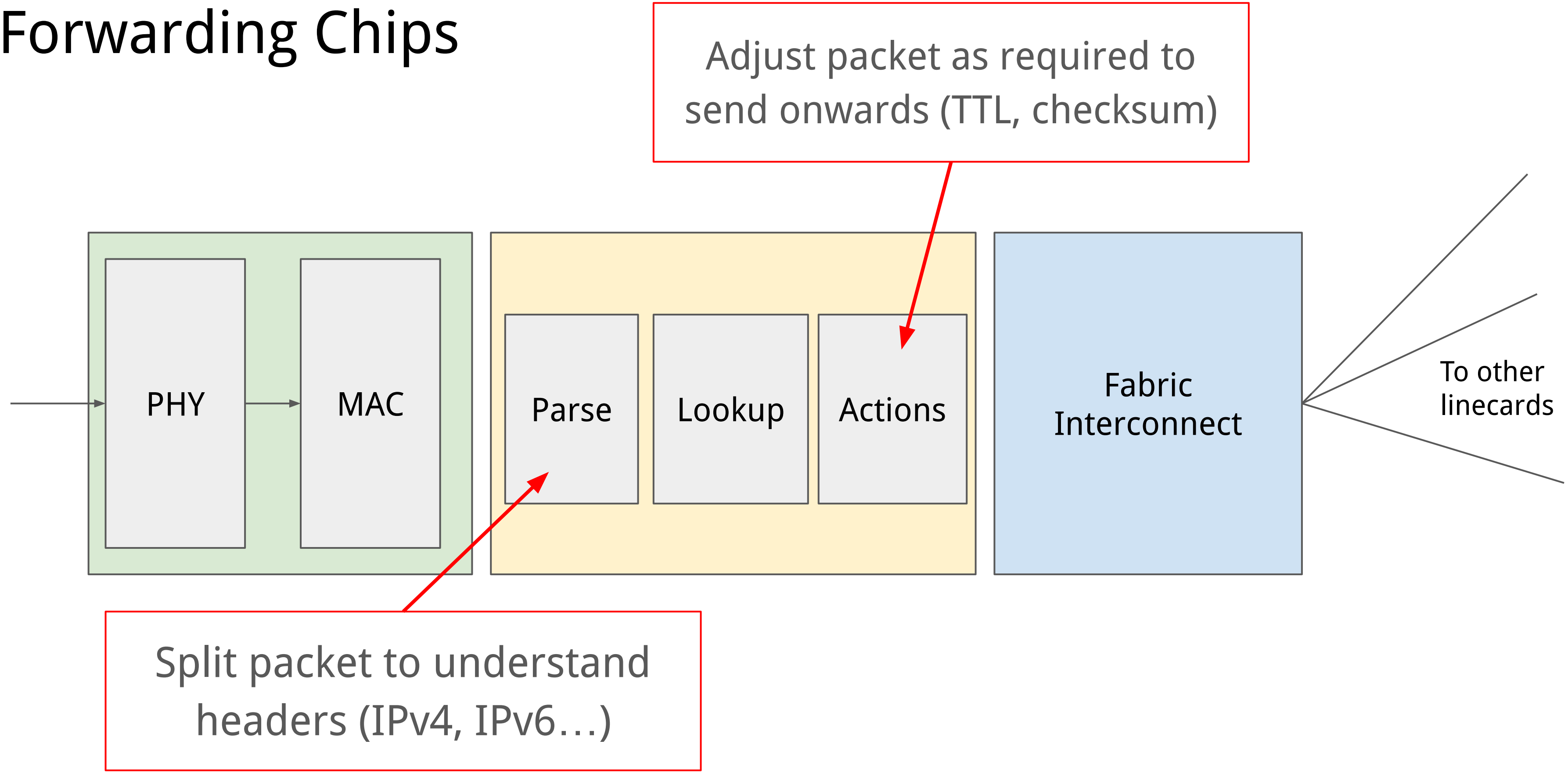


Implemented in hardware.

“Pipeline” For Packet Forwarding: Fabric



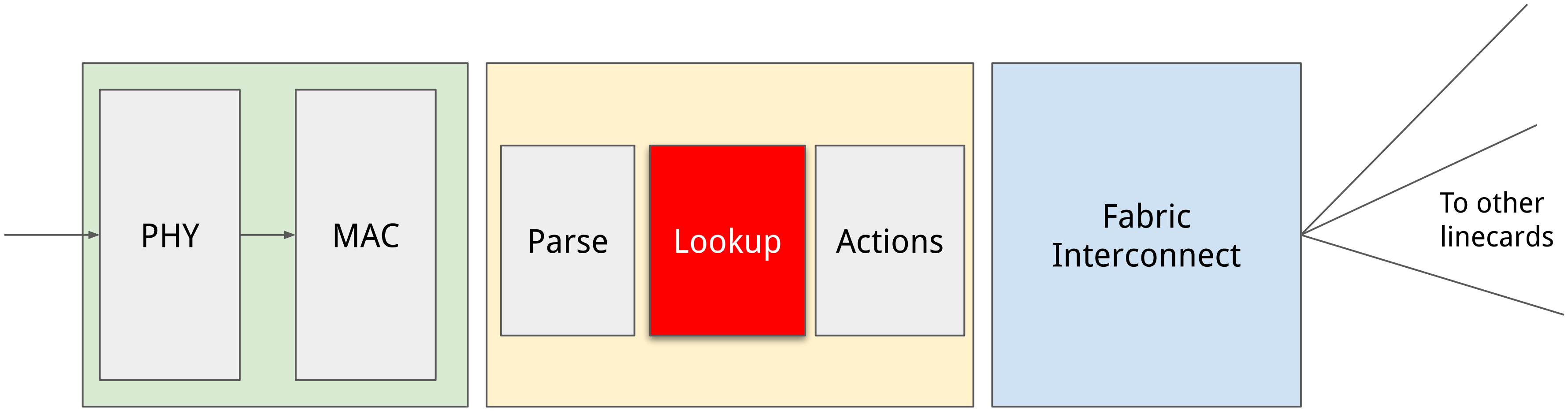
Forwarding Chips



Actions in Hardware

- **Easy to achieve:**
  - Checksum
  - Decrement TTL
- **More difficult:**
  - Options
    - Small number of cycles per packet on dedicated forwarding chips!
    - **Generally don't use/allow options!**
  - Fragmentation
    - Achievable in hardware with some overhead.
    - **Typically avoided** (Internet MTU is 1500-bytes).

Focusing on lookups!



Core router functionality! This is our challenge!

Where should we send a packet?

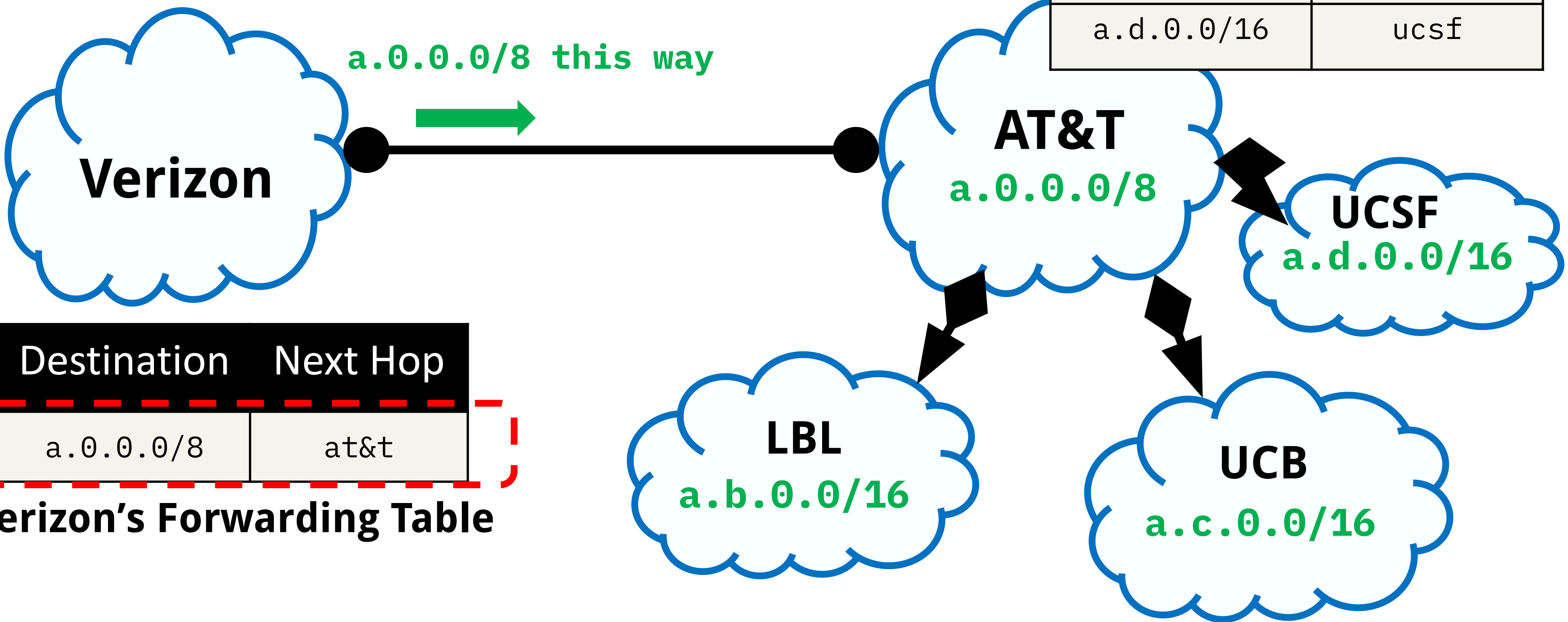
- Output lookups!

- Ideal? One lookup → forwarding entry.
  - Exact-match on destination IP - for O(1) lookups.
  - Forwarding table size?
  - Updating these tables - lots of entries to update!

- IP prefixes tend to be hierarchical.
  - Assigned IP addresses in a block to some ISP, and assigned to “downstream” networks.
  - Practically: /24 (256 address blocks) are the smallest we have on the Internet.
  - /32 (1 address) is the smallest internally though!
  - We can use compact tables that exploit this hierarchy – but lookups are more complex.

Find the *prefix* that matches

Where do we send a packet to a.b.x.y?



AT&T's Forwarding Table

Destination	Next Hop
a.b.0.0/16	lbl
a.c.0.0/16	ucb
a.d.0.0/16	ucsf

Verizon's Forwarding Table

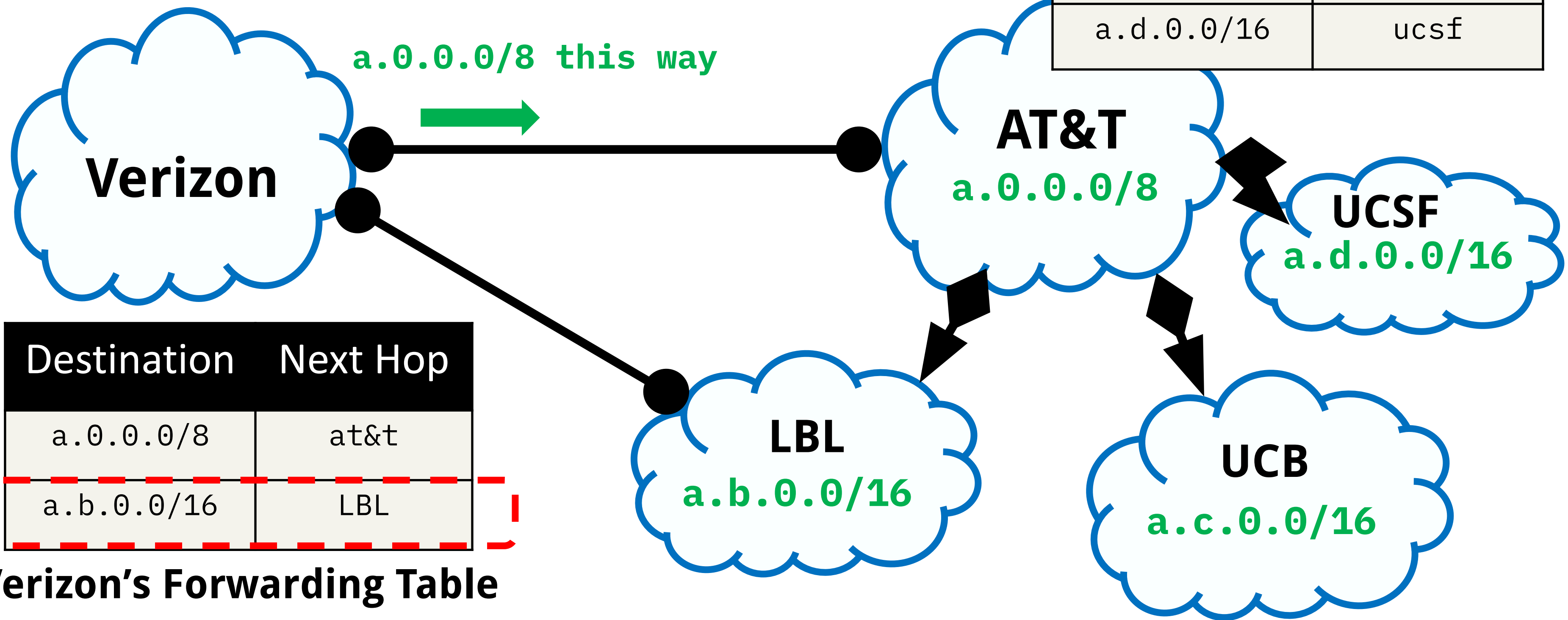
Destination	Next Hop
a.0.0.0/8	at&t

Find the **longest** *prefix* that matches

Where do we send a packet to a.b.x.y?

AT&T's Forwarding Table

Destination	Next Hop
a.b.0.0/16	lbl
a.c.0.0/16	ucb
a.d.0.0/16	ucsf



Destination	Next Hop
a.0.0.0/8	at&t
a.b.0.0/16	LBL

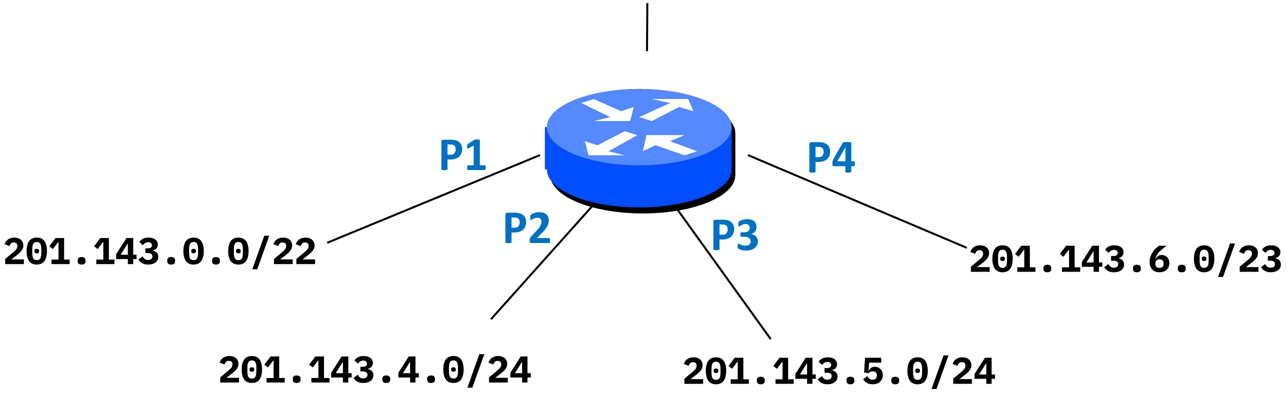
Verizon's Forwarding Table

Longest Prefix Match (LPM)

Take the **most specific** route that matches.

- If address matches multiple prefixes, then take the **longest** match.
- If the address matches no prefixes, take the default route.
- If there's no default route, drop the packet!

Example #1: 4 prefixes, 4 ports



Prefix	Port
201.143.0.0/22	P1
201.143.4.0.0/24	P2
201.143.5.0.0/24	P3
201.143.6.0/23	P4

Finding a Matching Route

- Incoming packet destination: 201.143.7.210

Prefix	Port
201.143.0.0/22	P1
201.143.4.0.0/24	P2
201.143.5.0.0/24	P3
201.143.6.0/23	P4

Finding a Matching Route: Convert to Binary

- Incoming packet destination: 201.143.7.210

11001001	10001111	00000111	11010010

Routing table

201.143.0.0/22	11001001	10001111	000000--	--------
201.143.4.0/24	11001001	10001111	00000100	--------
201.143.5.0/24	11001001	10001111	00000101	--------
201.143.6.0/23	11001001	10001111	0000011-	--------

Finding a Matching Route: Convert to Binary

- Incoming packet destination: 201.143.7.210

11001001	10001111	00000**111**	11010010

Routing table

~~201.143.0.0/22~~	~~11001001~~	~~10001111~~	~~00000**0**--~~	~~--------~~
201.143.4.0/24	11001001	10001111	00000**100**	--------
201.143.5.0/24	11001001	10001111	00000**101**	--------
201.143.6.0/23	11001001	10001111	00000**11**-	--------

Finding a Matching Route: Convert to Binary

- Incoming packet destination: 201.143.7.210

11001001	10001111	00000111	11010010

Routing table

~~**201.143.0.0/22**~~	~~11001001~~	~~10001111~~	~~000000--~~	~~--------~~
~~**201.143.4.0/24**~~	~~11001001~~	~~10001111~~	~~00000100~~	
~~**201.143.5.0/24**~~	~~11001001~~	~~10001111~~	~~00000101~~	
201.143.6.0/23	11001001	10001111	0000011-	--------

Longest Prefix Match

- Incoming packet destination: 201.143.7.210

Check an address against all prefixes and select the longest prefix it matches with

Routing table

~~201.143.0.0/22~~	~~11001001~~	~~10001111~~	~~000000--~~	~~--------~~
~~201.143.4.0/24~~	~~11001001~~	~~10001111~~	~~00000100~~	
~~20~~				
20				

NOT Check an address against all prefixes and find the one it matches *most bits* with

Finding a Match Efficiently

- Looking up against each entry scales poorly.
  - On average O(number of entries)
  - IPv4 Internet is ~1M prefixes!
- We can leverage the tree structure of binary strings.

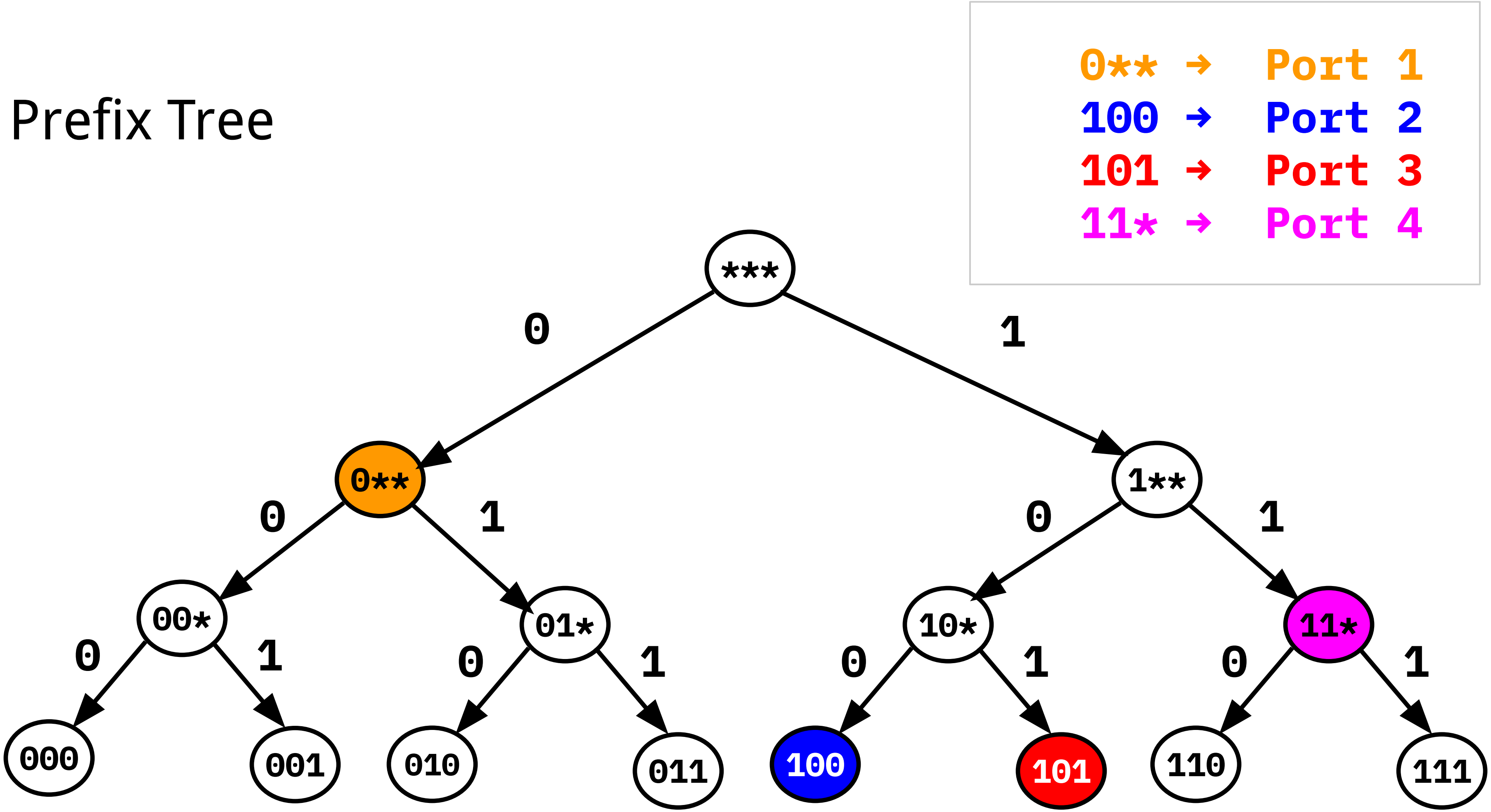
Prefix	Port
1100100110001111000000** ********	1
110010011000111100000100********	2
110010011000111100000101********	3
11001001100011110000011*********	4

Considering the 3-bit prefixes…

- (We'll focus on where the differences are)

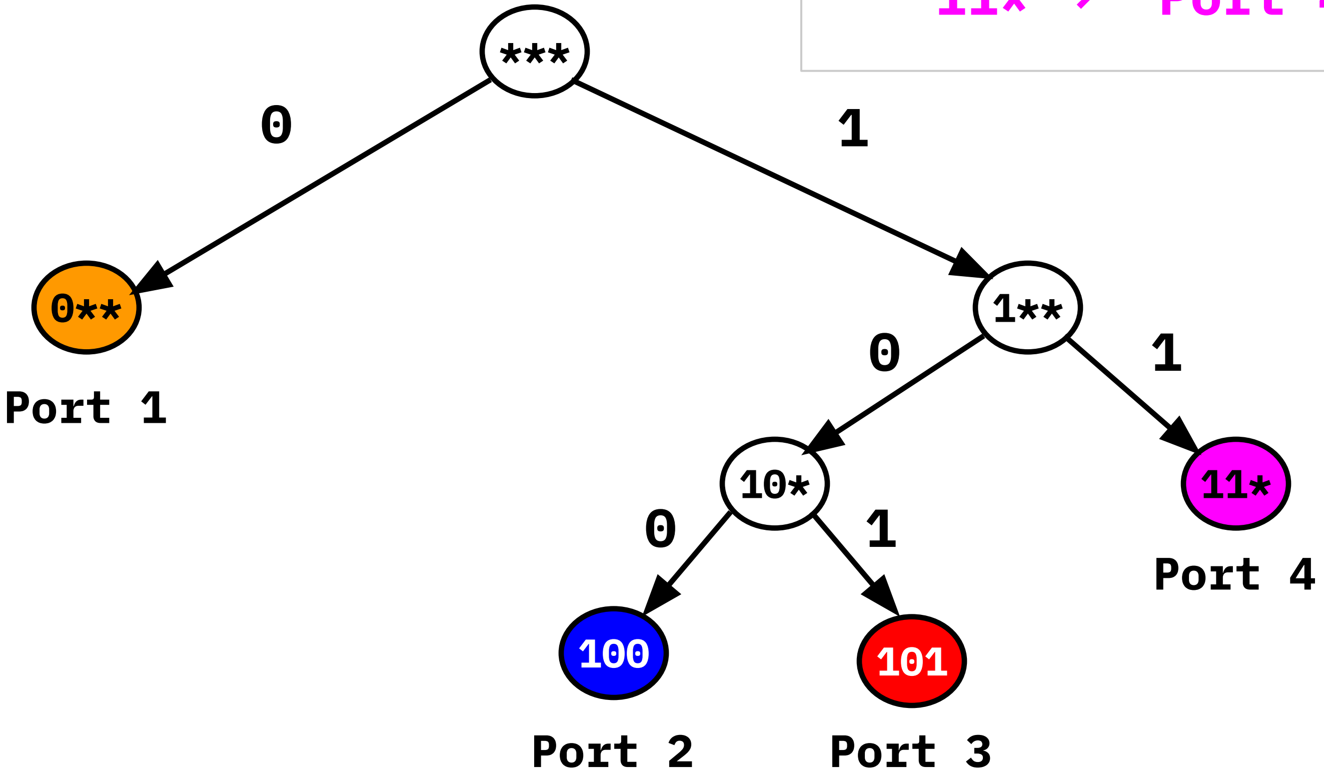
- `0** → Port 1`
- `100 → Port 2`
- `101 → Port 3`
- `11* → Port 4`

Prefix Tree


0** → Port 1
100 → Port 2
101 → Port 3
11* → Port 4
***
0
1
0**
1**
0
1
0
1
00*
01*
10*
11*
0
1
0
1
0
1
0
1
000
001
010
011
100
101
110
111

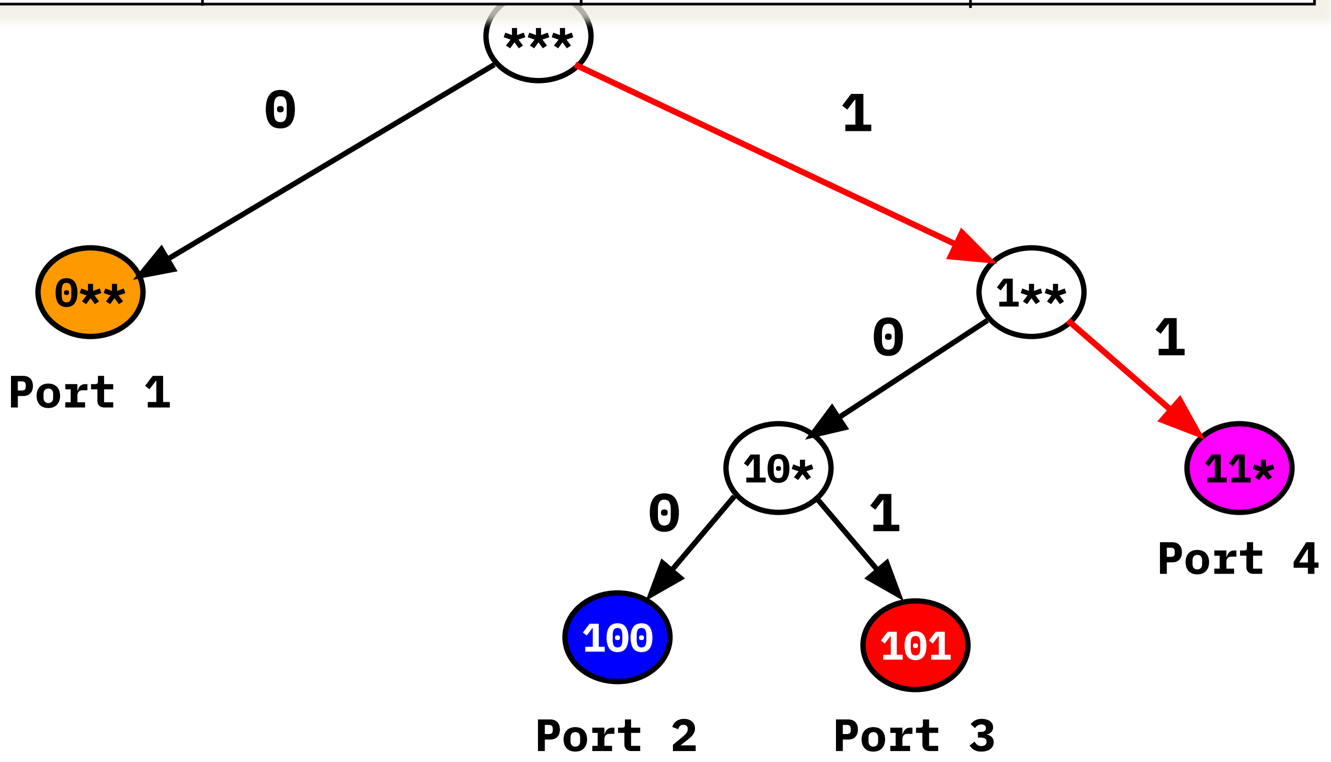
Prefix Tree

0** → Port 1
100 → Port 2
101 → Port 3
11* → Port 4

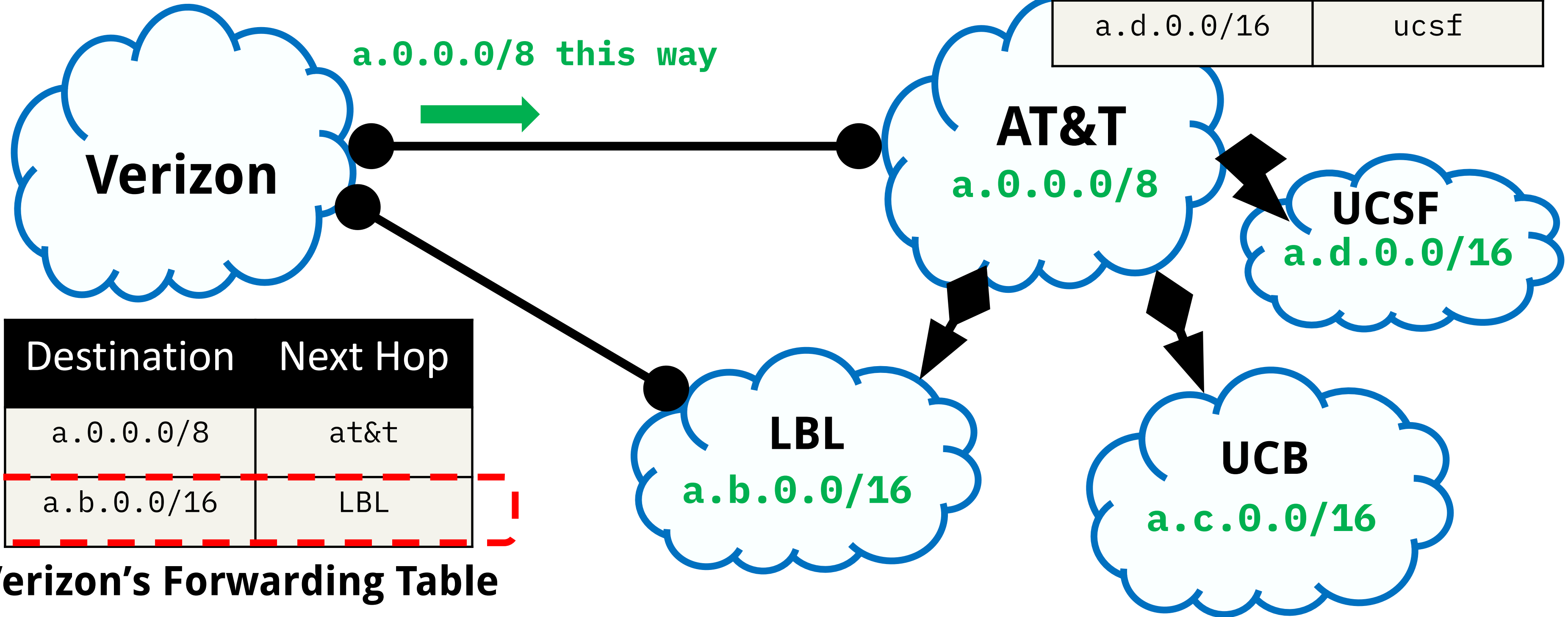


Prefix Match in the Tree

11001001	10001111	00000**111**	11010010



What about **multi-homing?**



AT&T's Forwarding Table

Destination	Next Hop
a.b.0.0/16	lbl
a.c.0.0/16	ucb
a.d.0.0/16	ucsf

Destination	Next Hop
a.0.0.0/8	at&t
a.b.0.0/16	LBL

Verizon's Forwarding Table

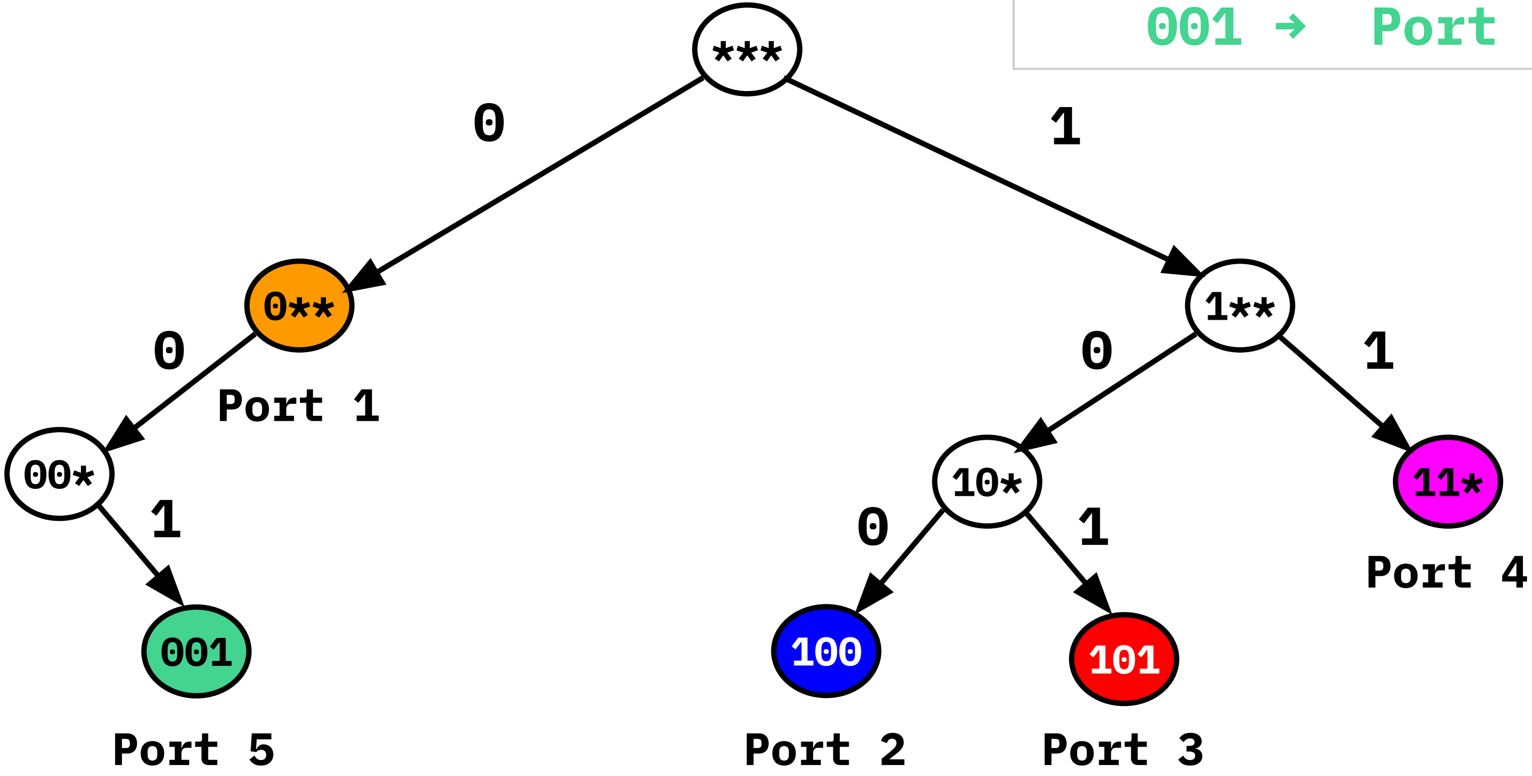
Some prefixes overlap

- Slightly different example - like Verizon.

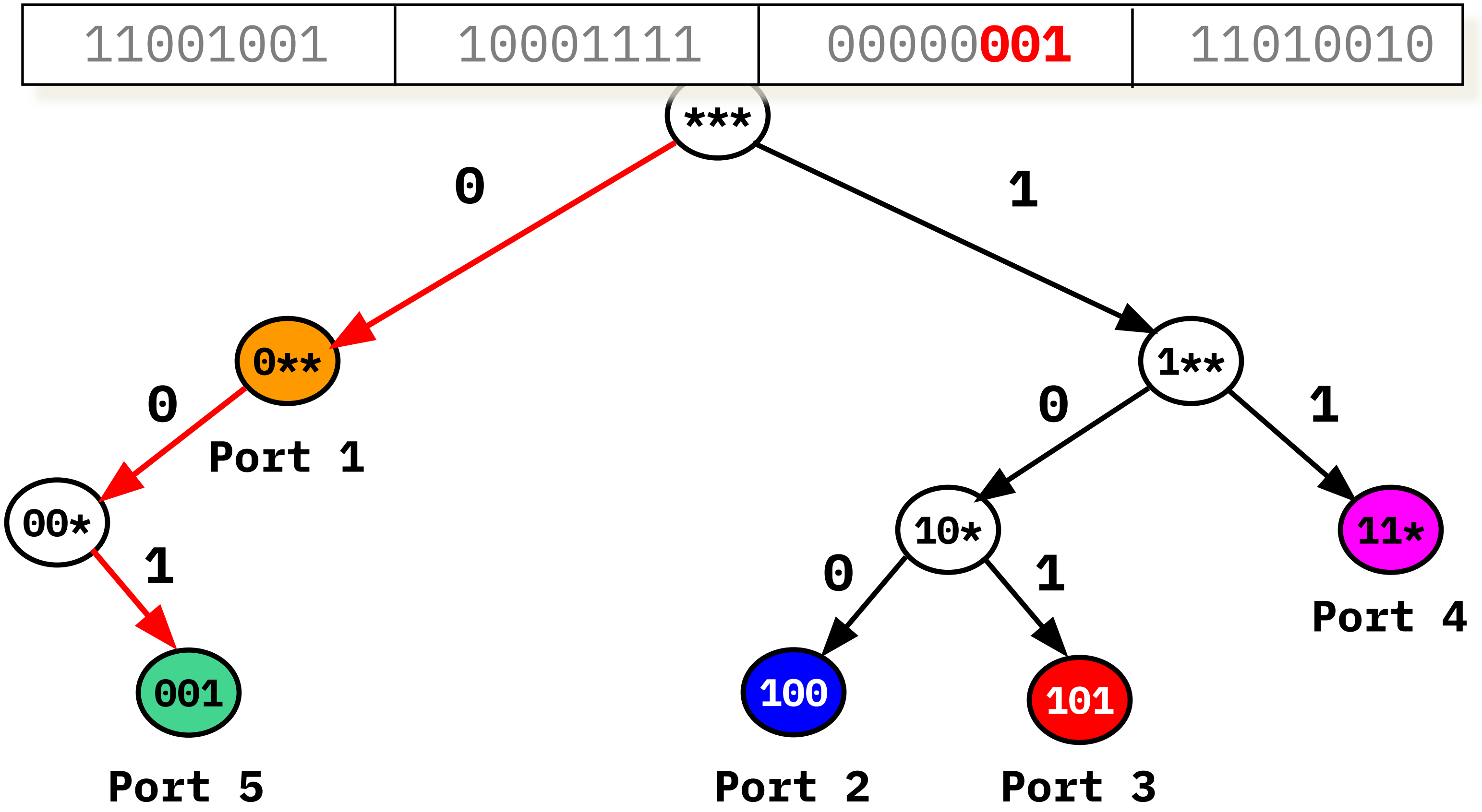
- `0** → Port 1`
- `100 → Port 2`
- `101 → Port 3`
- `11* → Port 4`
- `001 → Port 5`

Tree with overlapping entries.

0** → Port 1
100 → Port 2
101 → Port 3
11* → Port 4
001 → Port 5

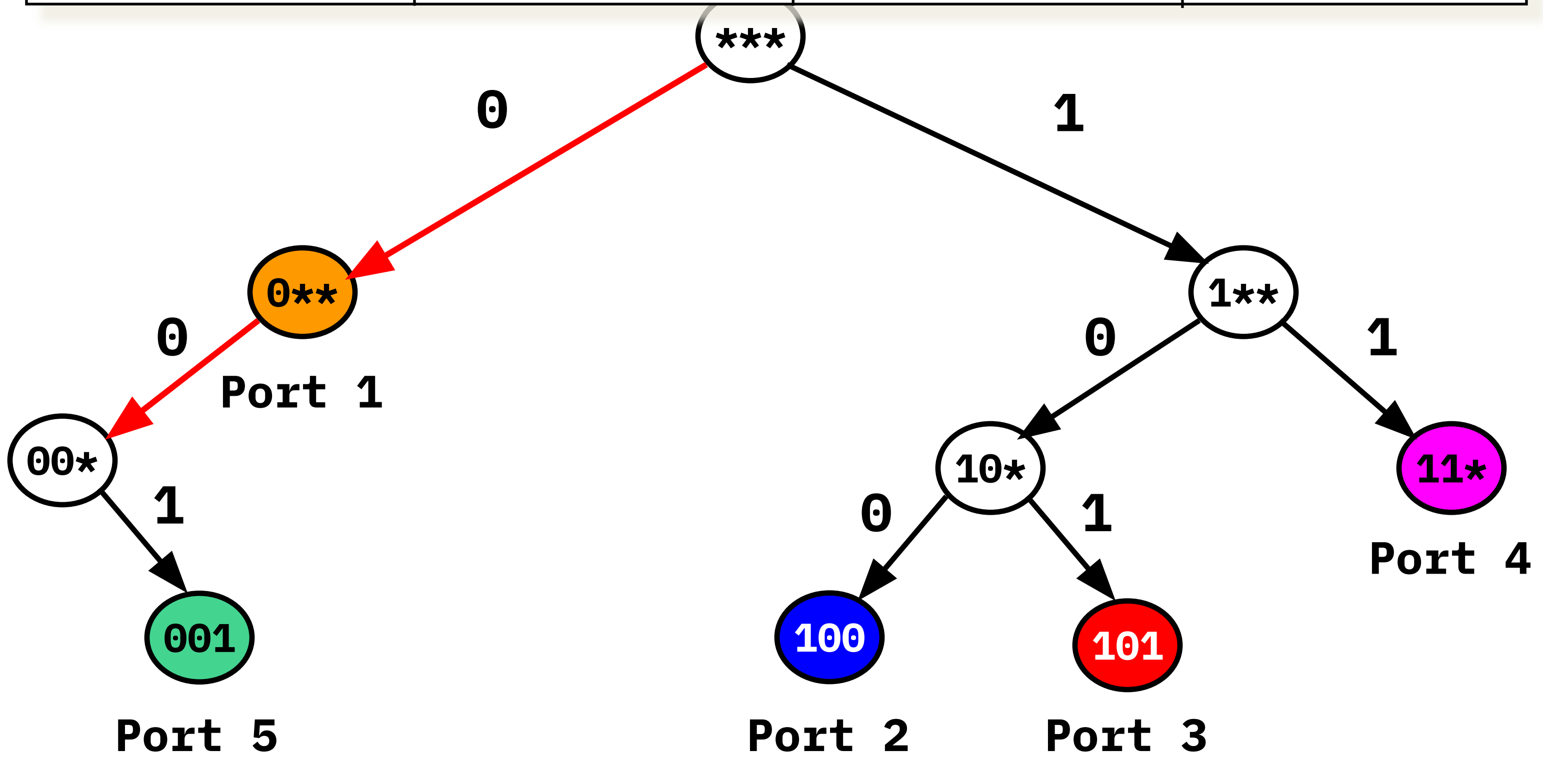


Example 1

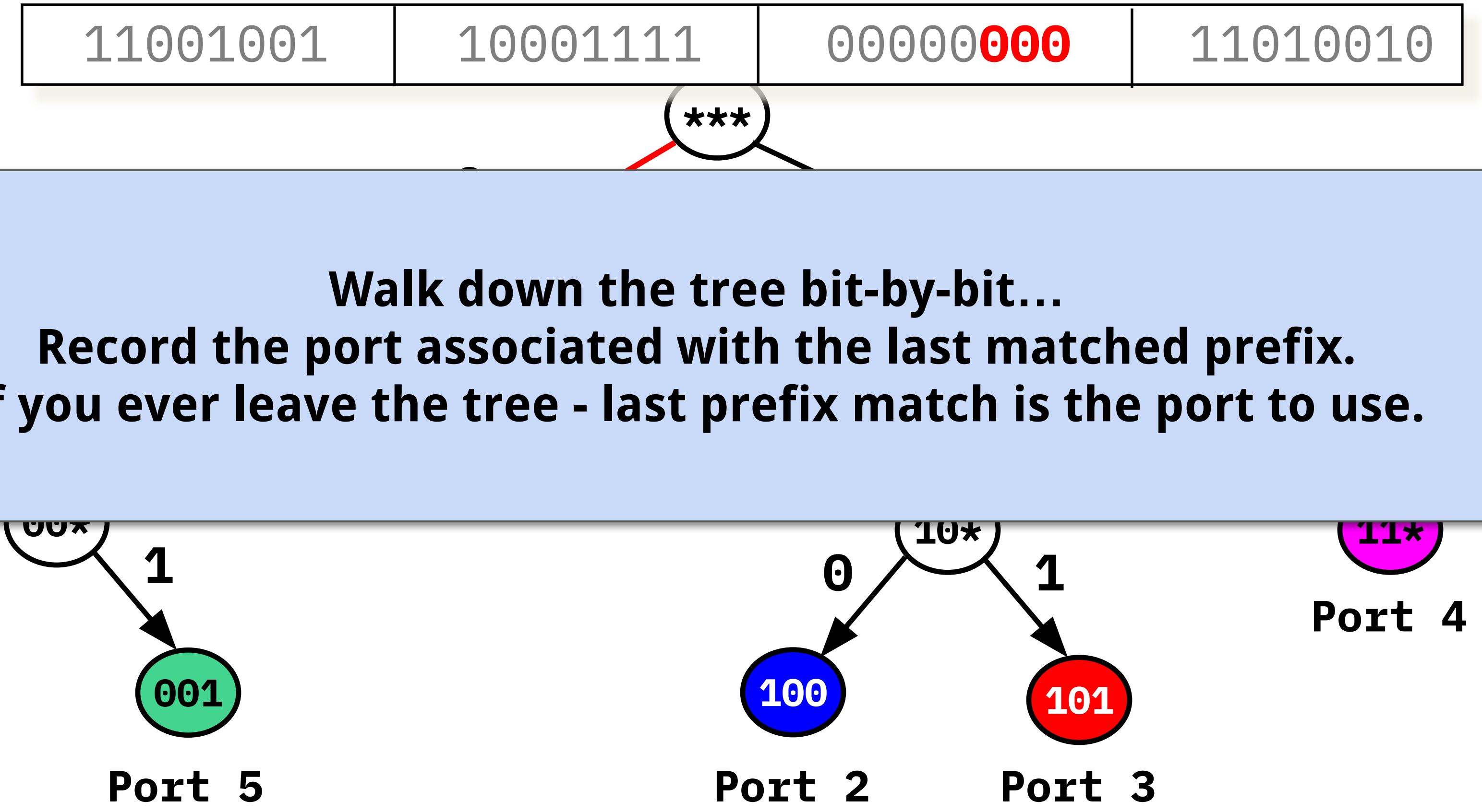

11001001
10001111
00000001
11010010
***
0
1
0**
Port 1
1**
0
1
00*
10*
11*
Port 4
1
0
1
001
100
101
Port 5
Port 2
Port 3

Example 2

11001001	10001111	00000**000**	11010010



Longest Prefix Match



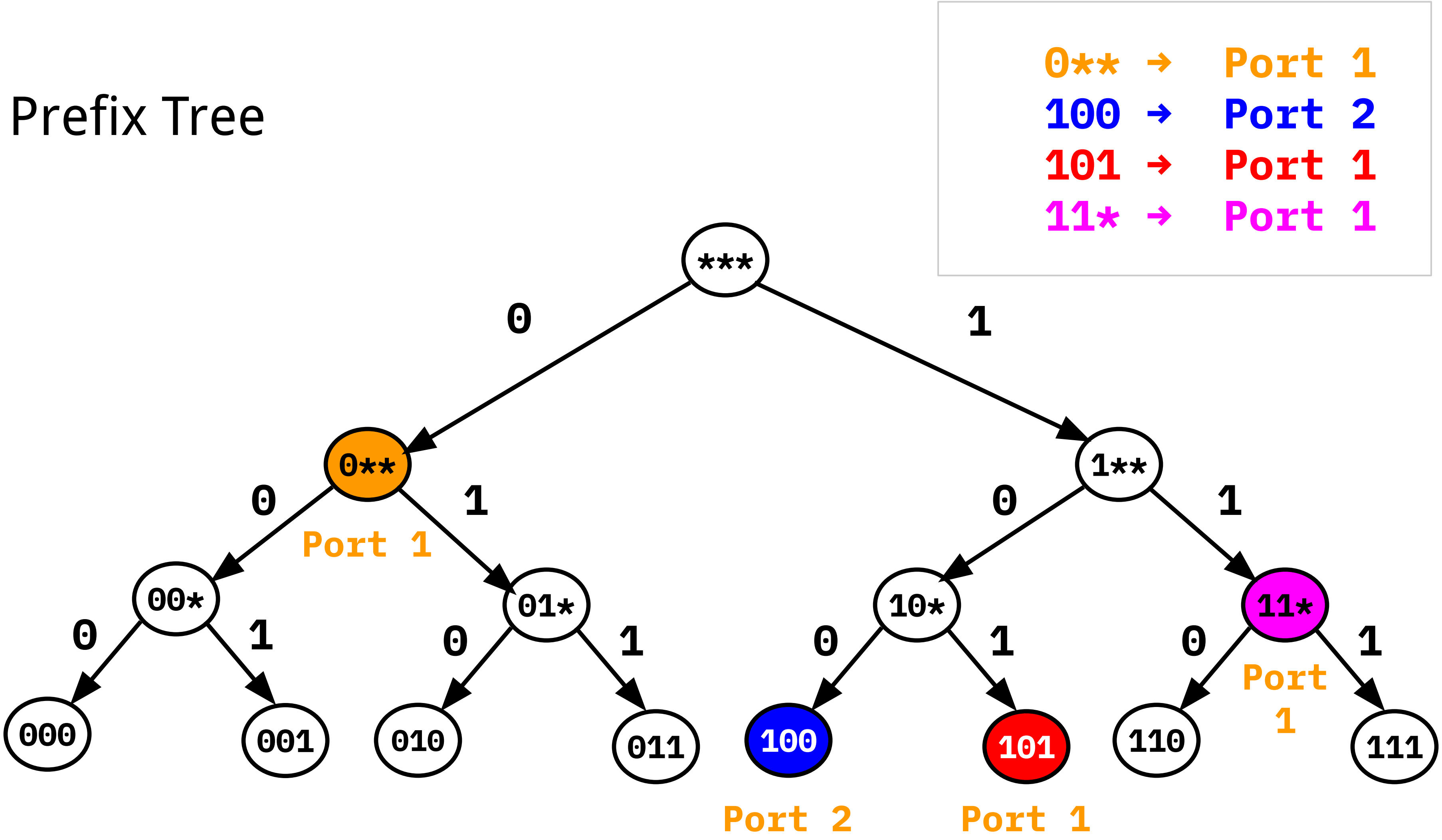
Walk down the tree bit-by-bit…
Record the port associated with the last matched prefix.
If you ever leave the tree - last prefix match is the port to use.

Several prefixes to the same port.

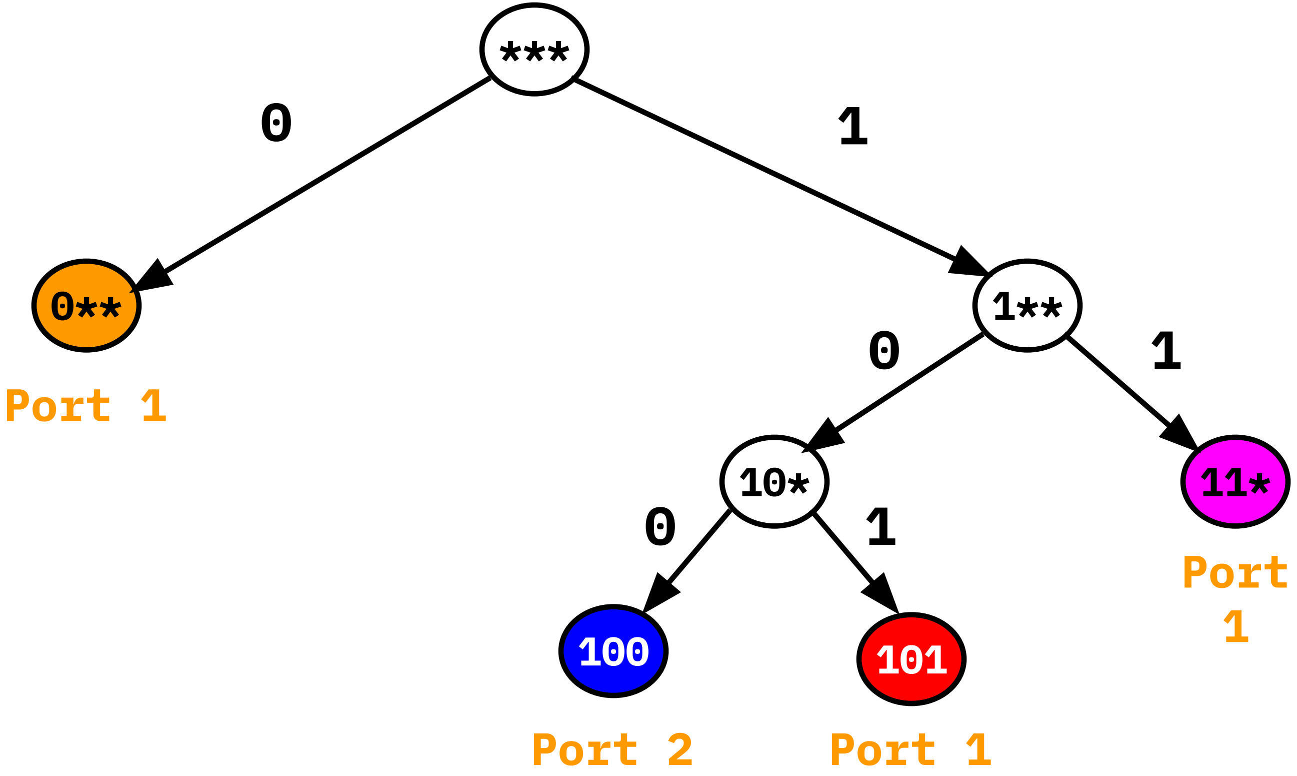
- More realistic Internet scenario.

- 0** → Port 1
- 100 → Port 2
- 101 → Port 1
- 11* → Port 1

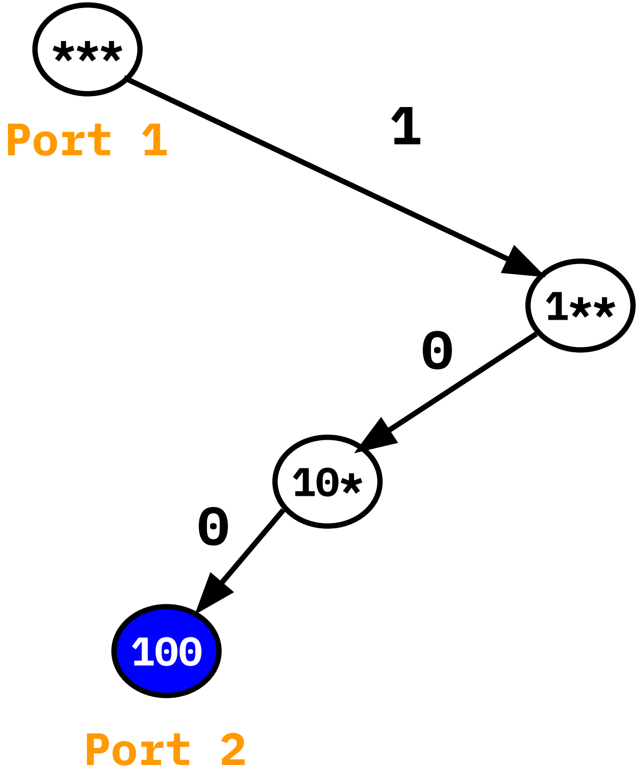
Prefix Tree


0** → Port 1
100 → Port 2
101 → Port 1
11* → Port 1
***
0
1
0**
1**
0
1
0
1
Port 1
00*
01*
10*
11*
0
1
0
1
0
1
0
1
Port 1
000
001
010
011
100
101
110
111
Port 2
Port 1

Normal Prefix Tree



A more compact representation.

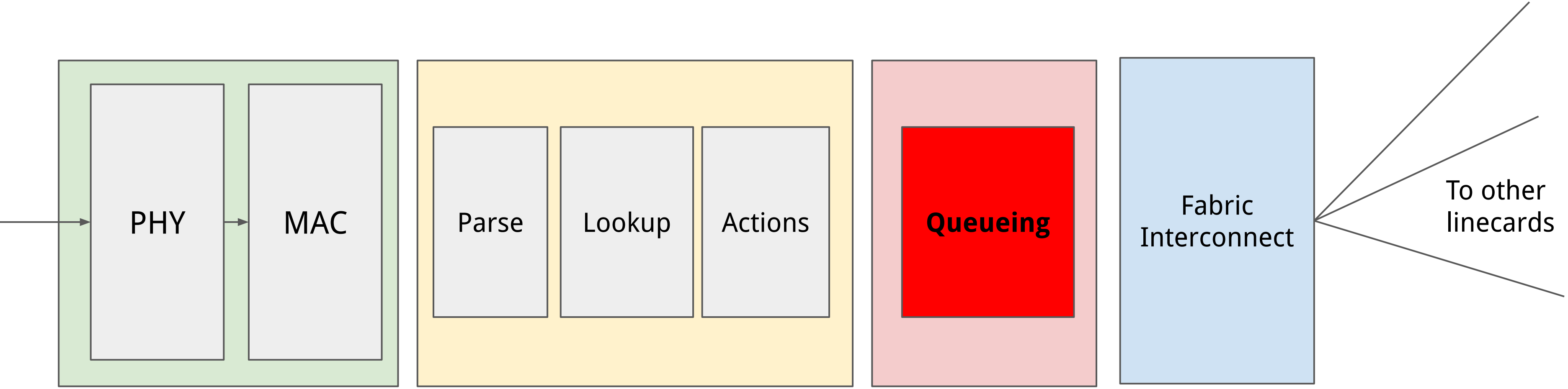


LPM in real routers

- All routers have this LPM functionality.
    - But use more advanced/complex solutions.

- Heuristics and optimisations can be made based on what is seen in the real internet.
    - Some destinations more popular than others.
    - Some ports have more destinations
    - Typical prefix sizes (recall: smallest IPv4 Internet prefix is /24).
    - Speed of update required.

Output Linecards: A wider picture



Packet Queueing

- Classification: what queue should this packet be put in to.
  - One queue per input port, one queue per marking on the packet (DSCP?)
- Buffer management.
  - Should we drop packets?
- Scheduling.
  - When should we transmit packets?

Traffic and queue management?

Our picture assumes the simplest possible!

- *No* classification
- *Drop-tail* buffer management: if the buffer is full, just drop the packet.
- *FIFO* scheduler - just send the packets in the order they arrive.

Many alternate (complex) scenarios - used to implement business objectives.

Recap: IP Routers

- Have different "planes":
  - Control plane - programming forwarding entries and exception packets.
  - Management plane - configure and monitor router functionality.
  - Data plane - packet forwarding!

- Data plane leverages tradeoffs in software vs hardware packet processing.
  - Software: flexible but slow
  - Hardware: inflexible but fast

- Data plane challenges: speed!
  - Update packet header (easy)
  - LPM lookup on destination address (harder).