Lecture 7 (Routing 4)

IP Routers

Slides credit: CS 168 @ UC Berkeley

IP Routers in Real Life

Lecture 7, Spring 2026

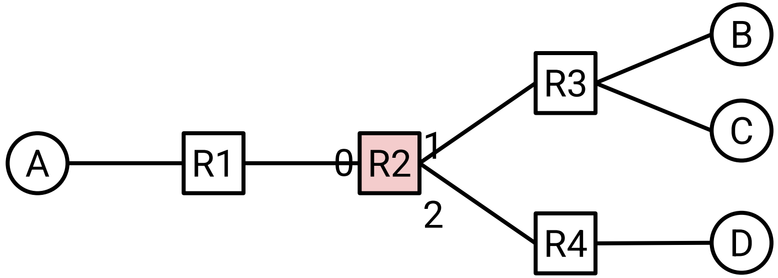
IP Routers

Routers – Conceptually

Recall:

- A router performs routing protocols to learn about routes.
- A router receives packets and forwards them according to the forwarding table.

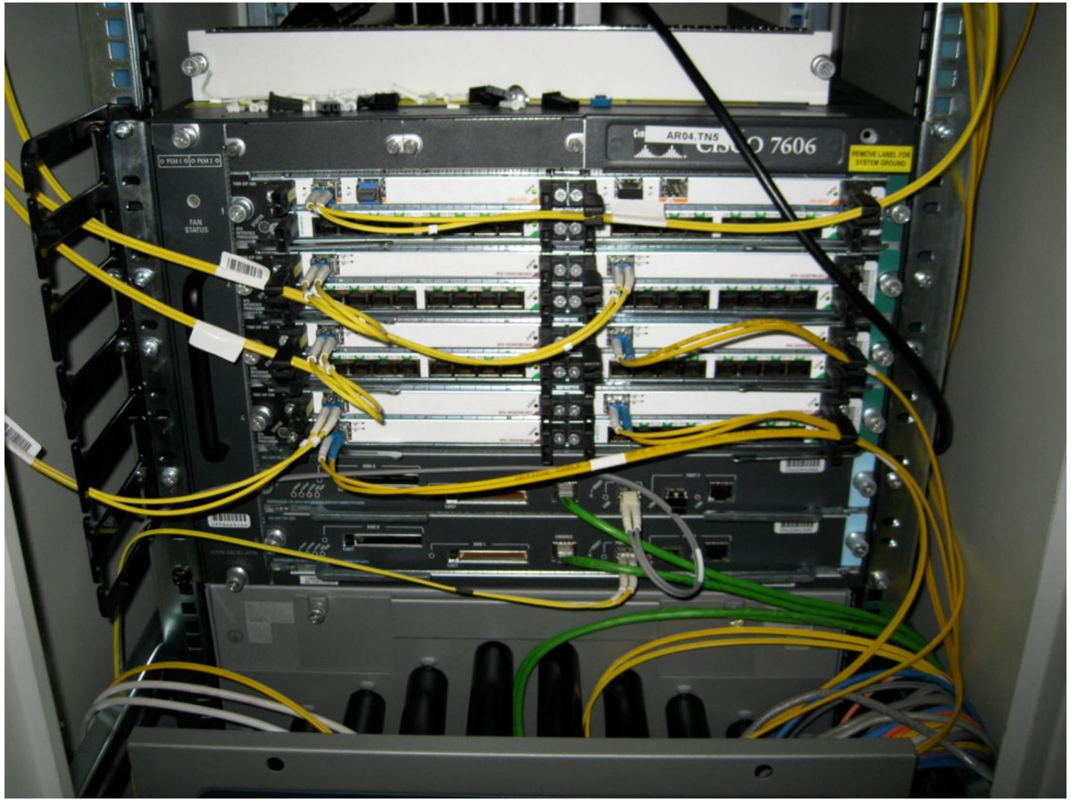
Today: What do routers actually look like in real life?

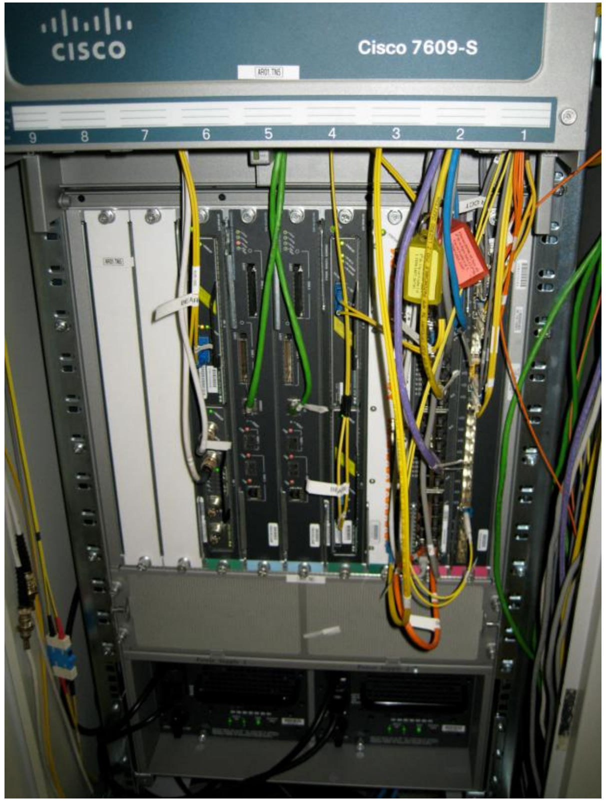


R2's Table	
Destination	Port
A	0
B	1
C	1
D	2

Routers in Real Life

A router is just a computer specialized for forwarding packets.





Measuring Router Size

Different dimensions for measuring the size of a router:

- Physical size, number of ports, bandwidth.
- Capacity = number of ports × speed of each port.
  - The speed of a port is sometimes called its **line rate**.

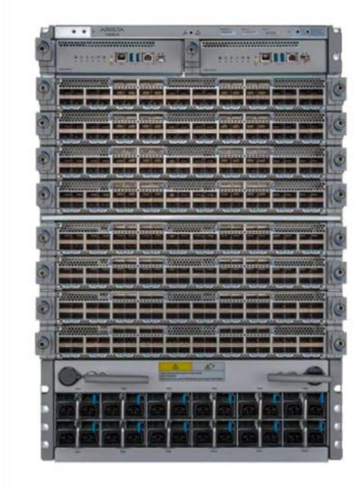
Example:

- A home router might have four 100 Mbps ports, and one 1 Gbps port.
- Total capacity: 1.4 Gbps.

Modern Router Capacity

Modern router:

- 288 ports.
- 400 Gbps line rate per port.
- Capacity = 115.2 Tbps.



Next-generation router:

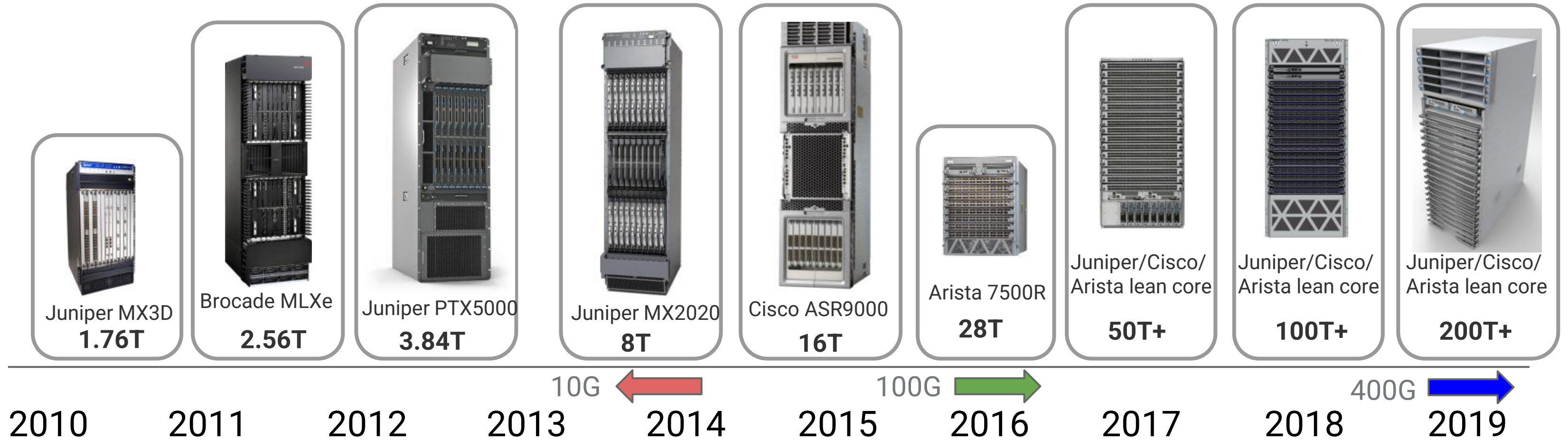
- 288 ports.
- 800 Gbps line rate per port.
- Capacity = 230 Tbps.

Innovation is focused on improving line rate, since we're running out of physical space to add more ports.

Evolution of Router Capacity

Modern routers are facing physical constraints: size, power, cooling, etc.

Rate of growth is slowing: 10G → 100G → 400G → 800G.



Router Components (Planes)

Lecture 7, Spring 2026

IP Routers

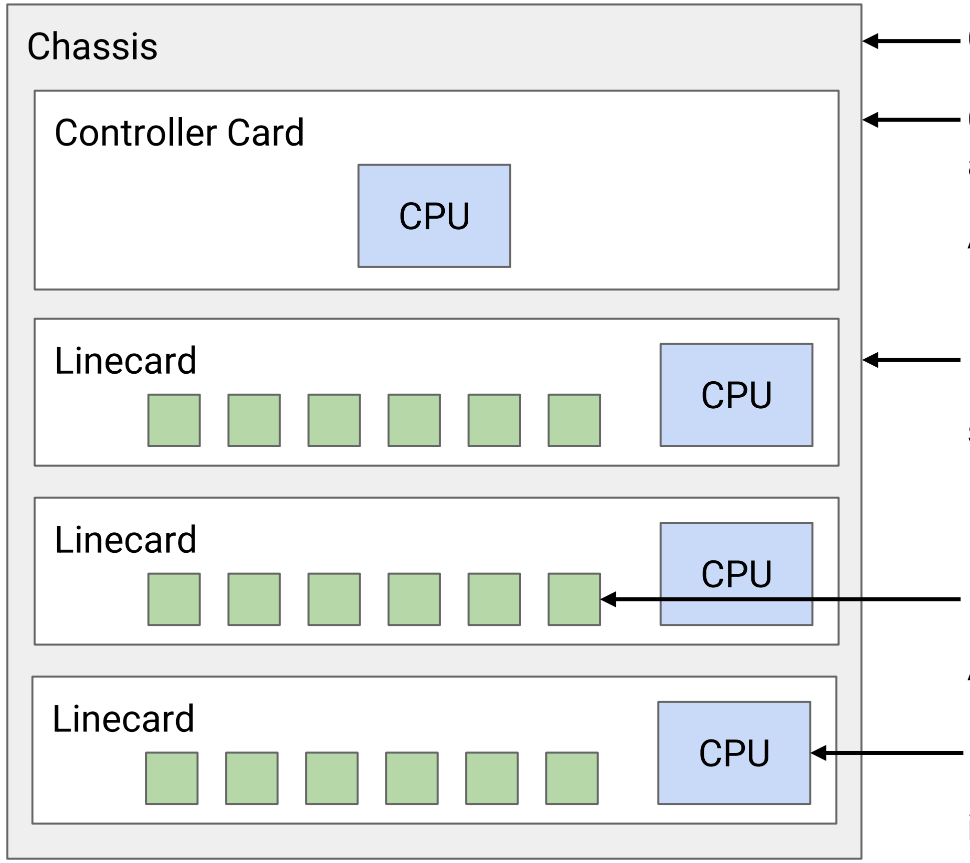
Router Planes

The components of a router can be split into 3 planes:

- The **data plane** handles forwarding packets.
  - Used every time a packet arrives: Nanosecond time scale.
  - Operates locally. No coordinating with other routers.
- The **control plane** performs routing protocols.
  - Used every time the network topology changes. Second time scale.
- The **management plane** lets the operator interact with the router.
  - Operator uses **network management system** (NMS) to interact with router.
  - Tell the router what to do, and monitor what the router is doing.
  - Time scale of seconds/minutes.
  - Example: Assigning costs to links.
  - Example: How much traffic is being sent on each link?

Each plane is optimized for its tasks and its time scale.

What's Inside a Router? – View of Components



Chassis: The metal box holding the hardware.

Control processor: Runs routing protocols and puts forwarding tables on linecards.

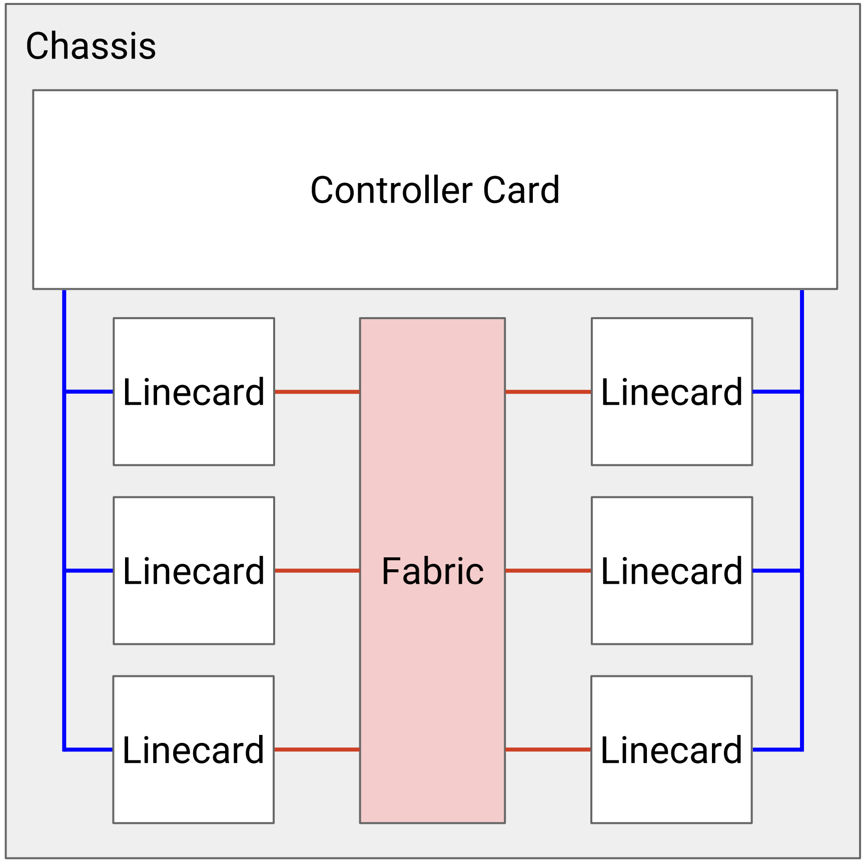
Also handles management plane.

Removable linecards gives us flexibility in scaling the router. 1–20 linecards per chassis.

Each linecard has several ports.
A port can be used for both input/output.

Each linecard has some hardware to control its functionality.

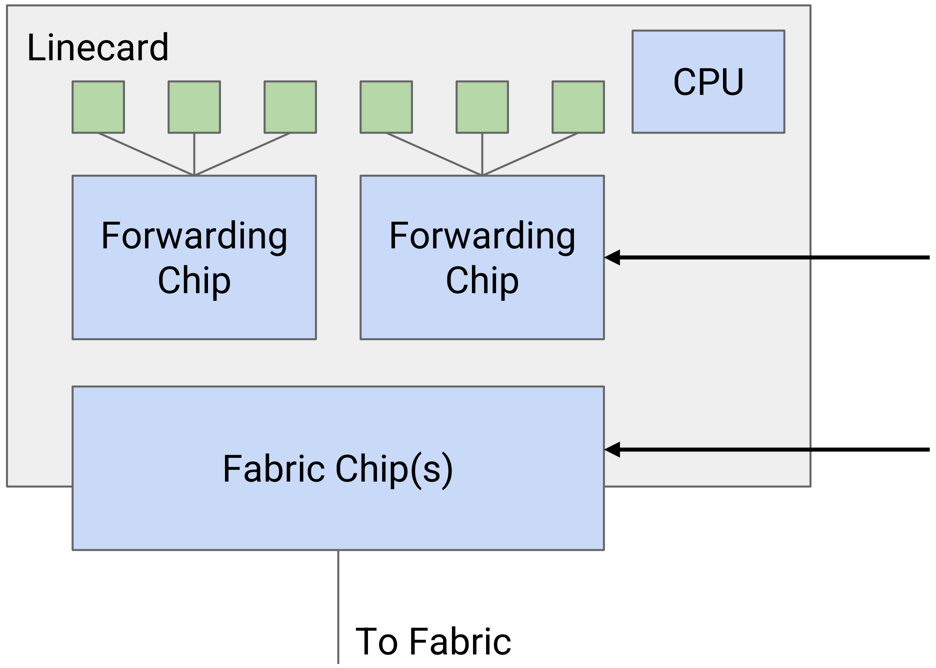
What's Inside a Router? – View of Links



Controller card is connected to the linecards.

Each linecard is connected to the **fabric**:
A bunch of wires providing high-bandwidth, fault-tolerant interconnection between linecards.

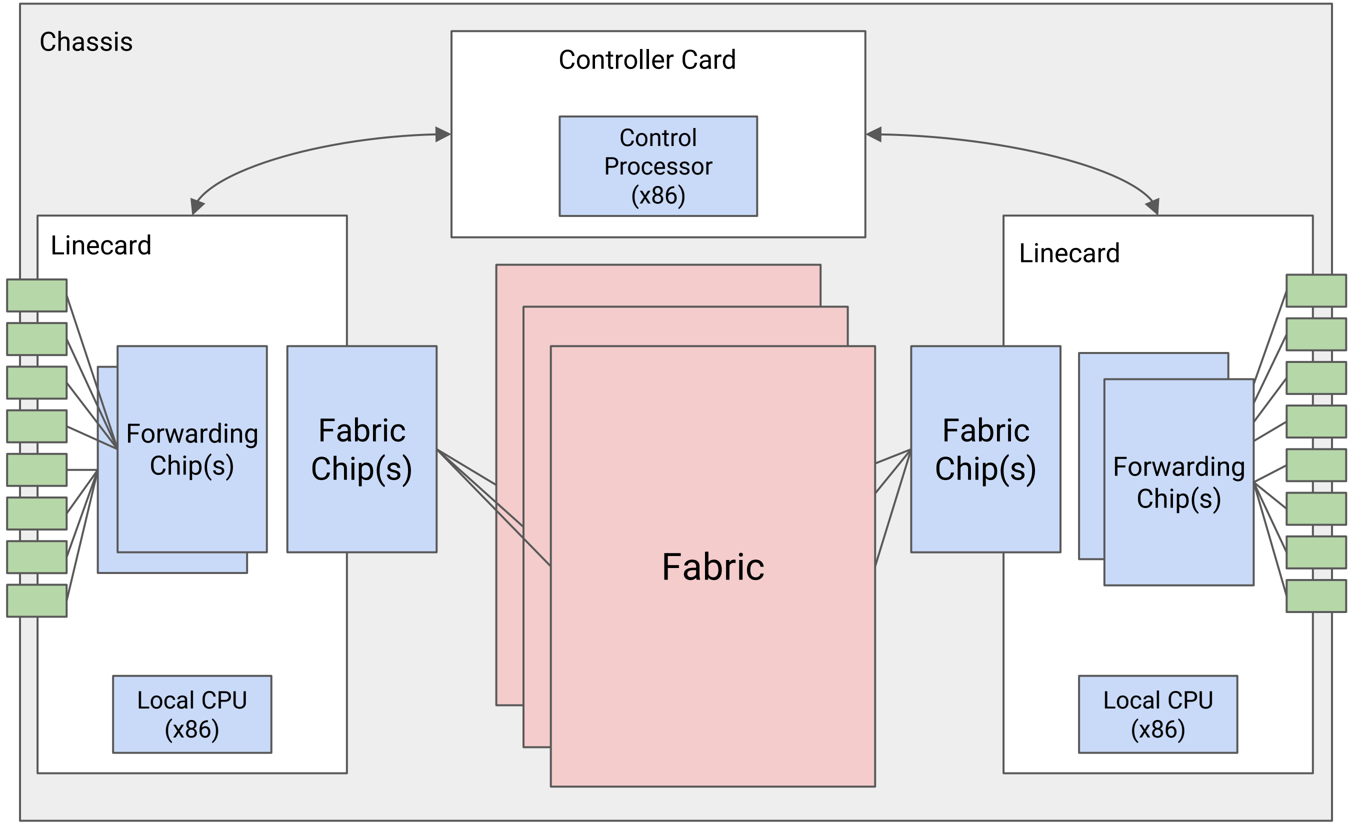
What's Inside a Router? – View of a Linecard



Specialized hardware optimized to forward packets.

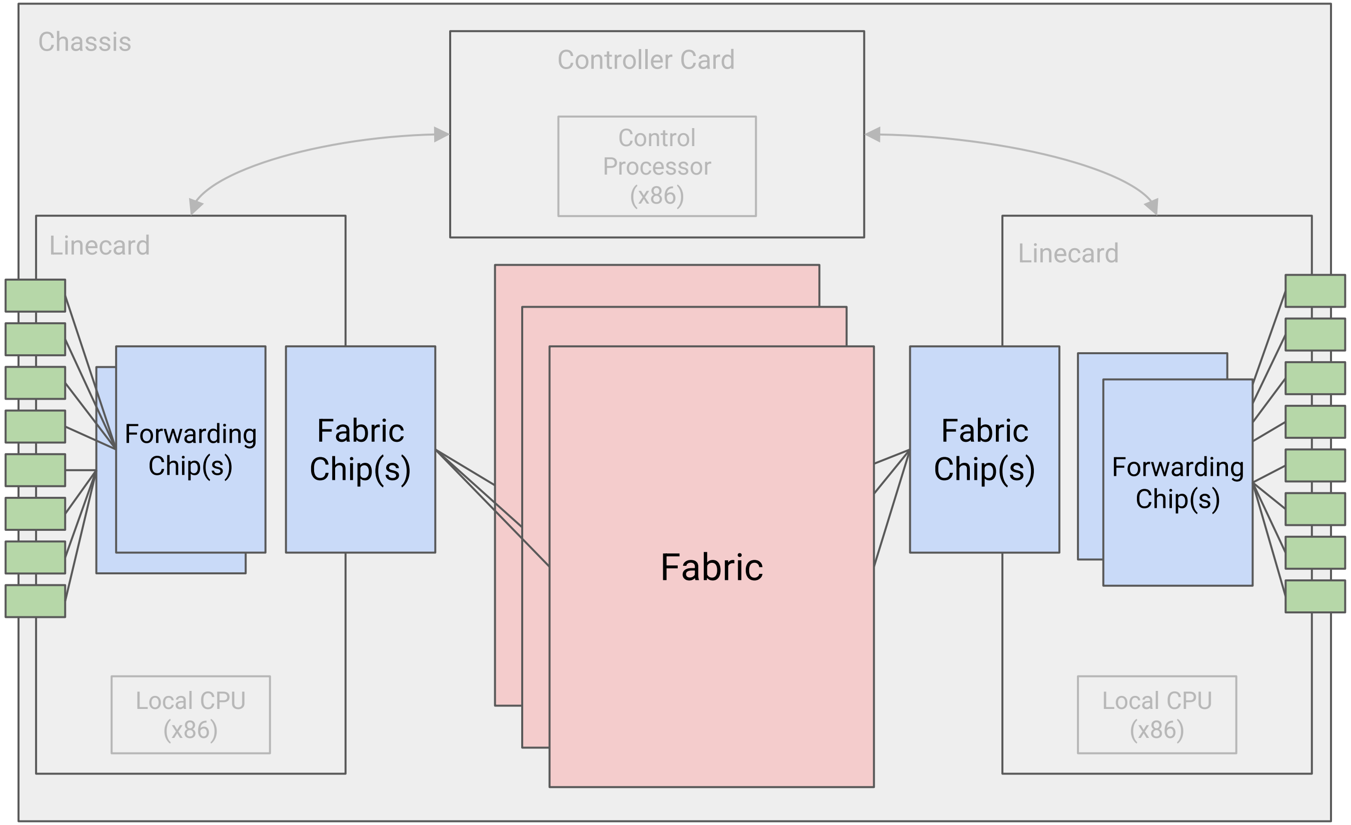
Sends packets into the fabric, to other ports on the router.

What's Inside a Router? – Full View



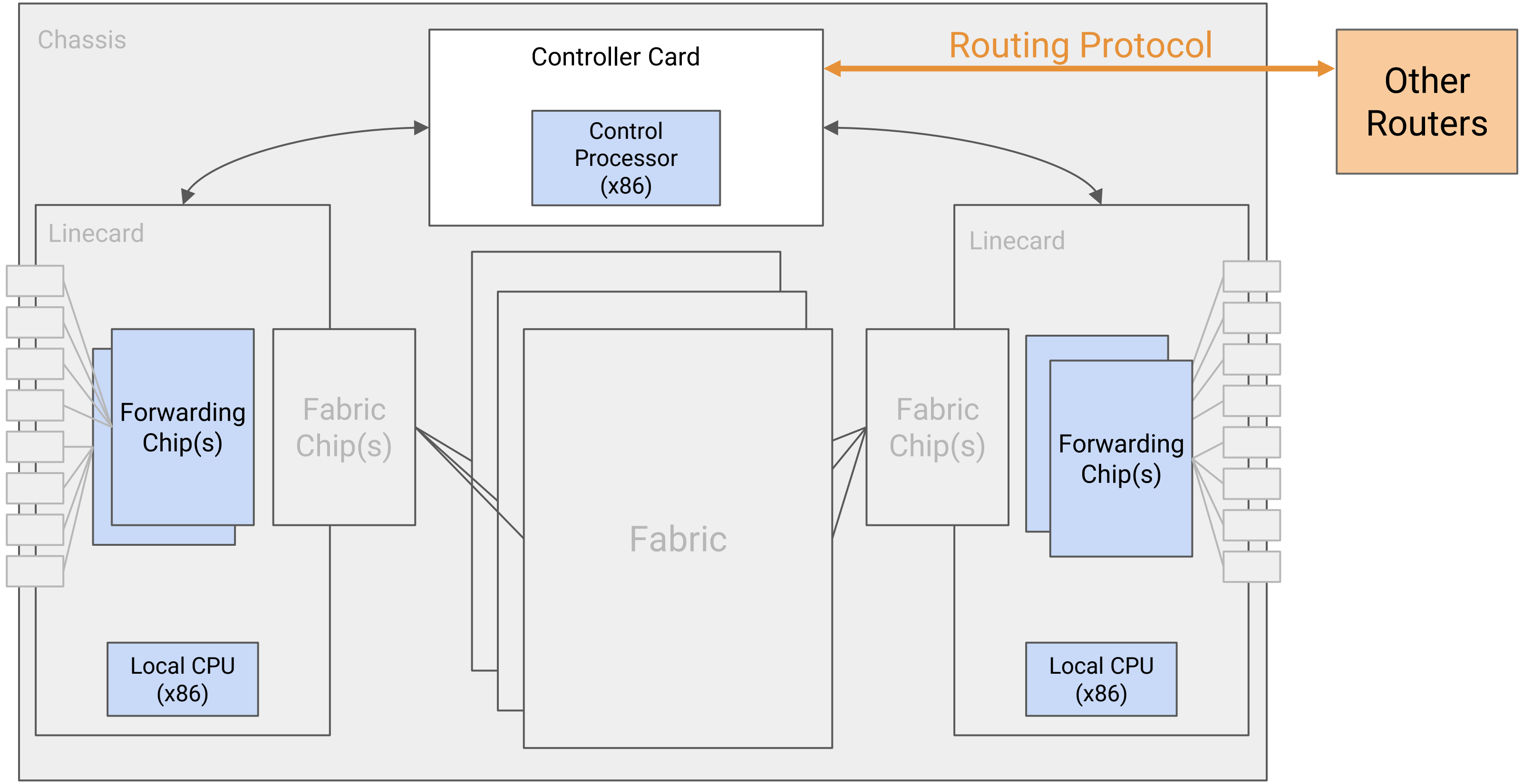
What's Inside a Router? – Data Plane

Data: Packet travels from port to port, via forwarding chips and fabric.



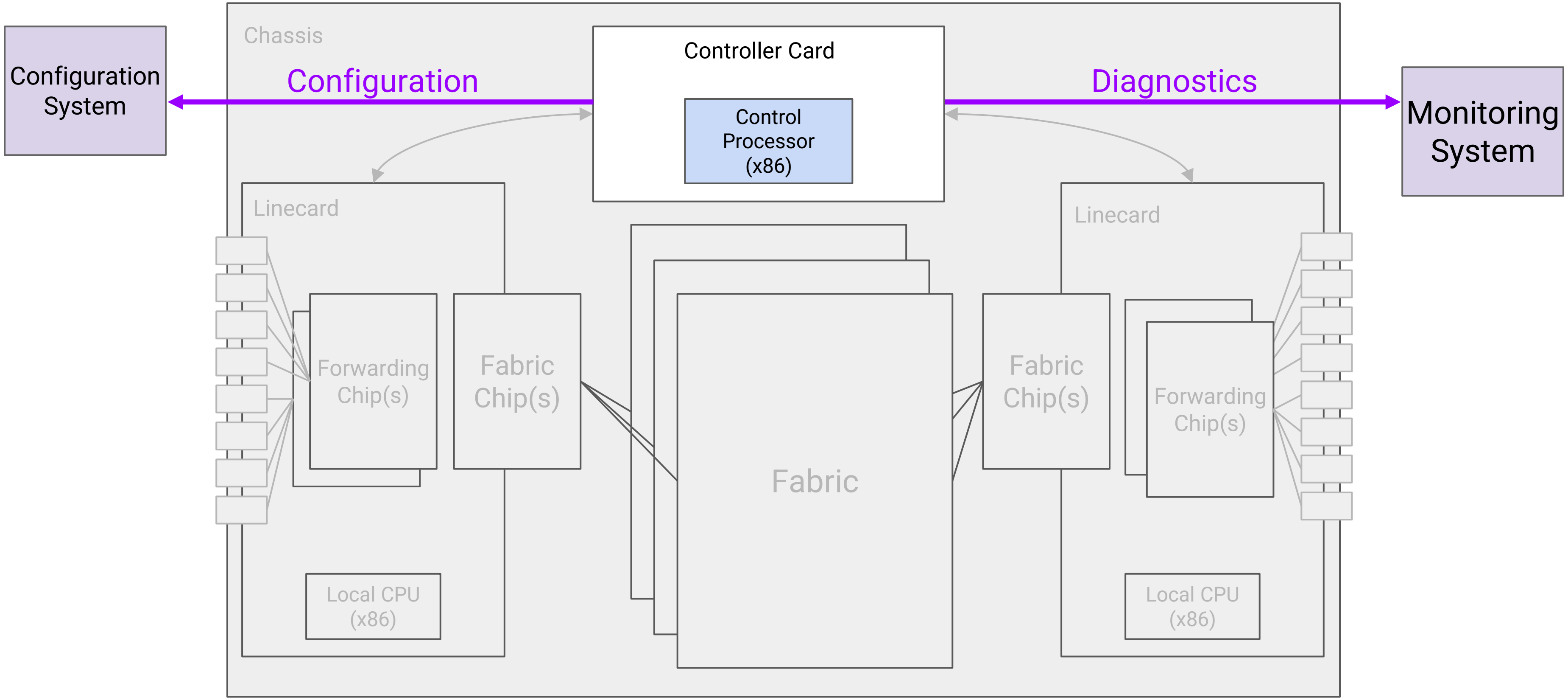
What's Inside a Router? – Control Plane

Control: Controller card talks with other routers, and programs linecards with routes.



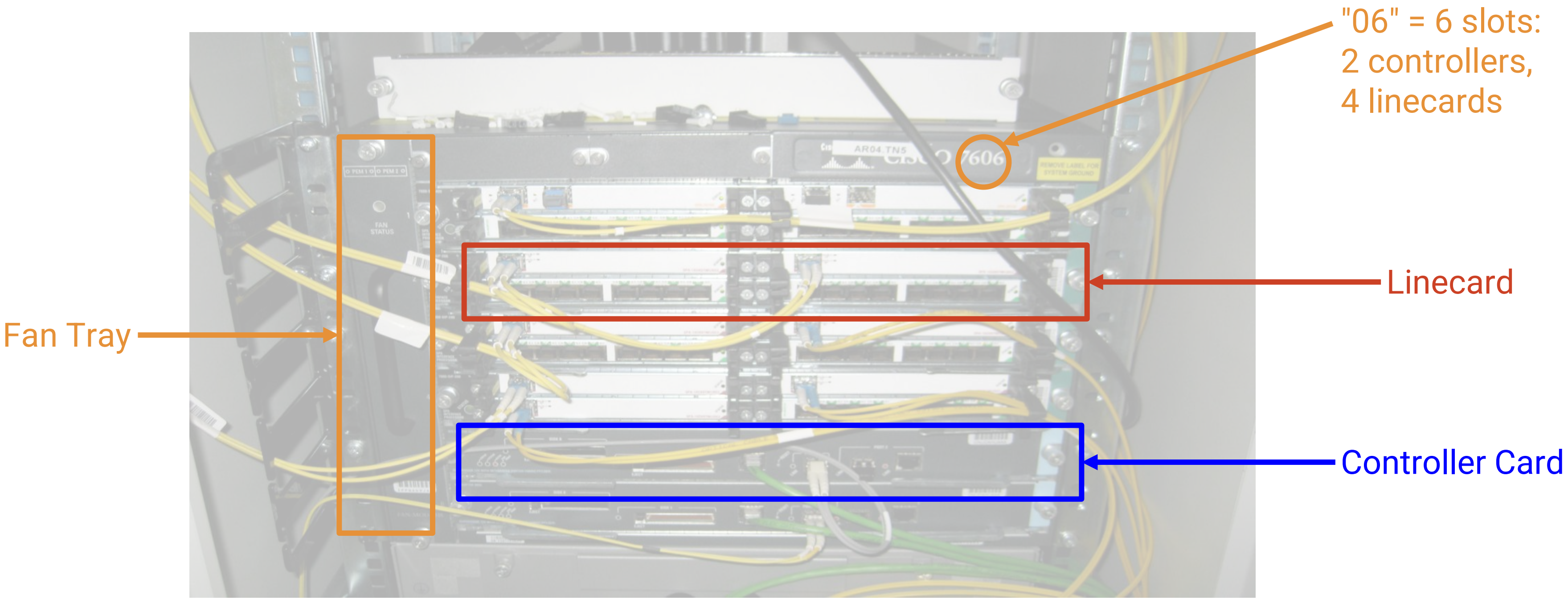
What's Inside a Router? – Management Plane

Management: Controller card talks with operator.



What's in a Router?

A router is really a *cluster* of computers specialized for forwarding packets.



Packet Types

Lecture 7, Spring 2026

IP Routers

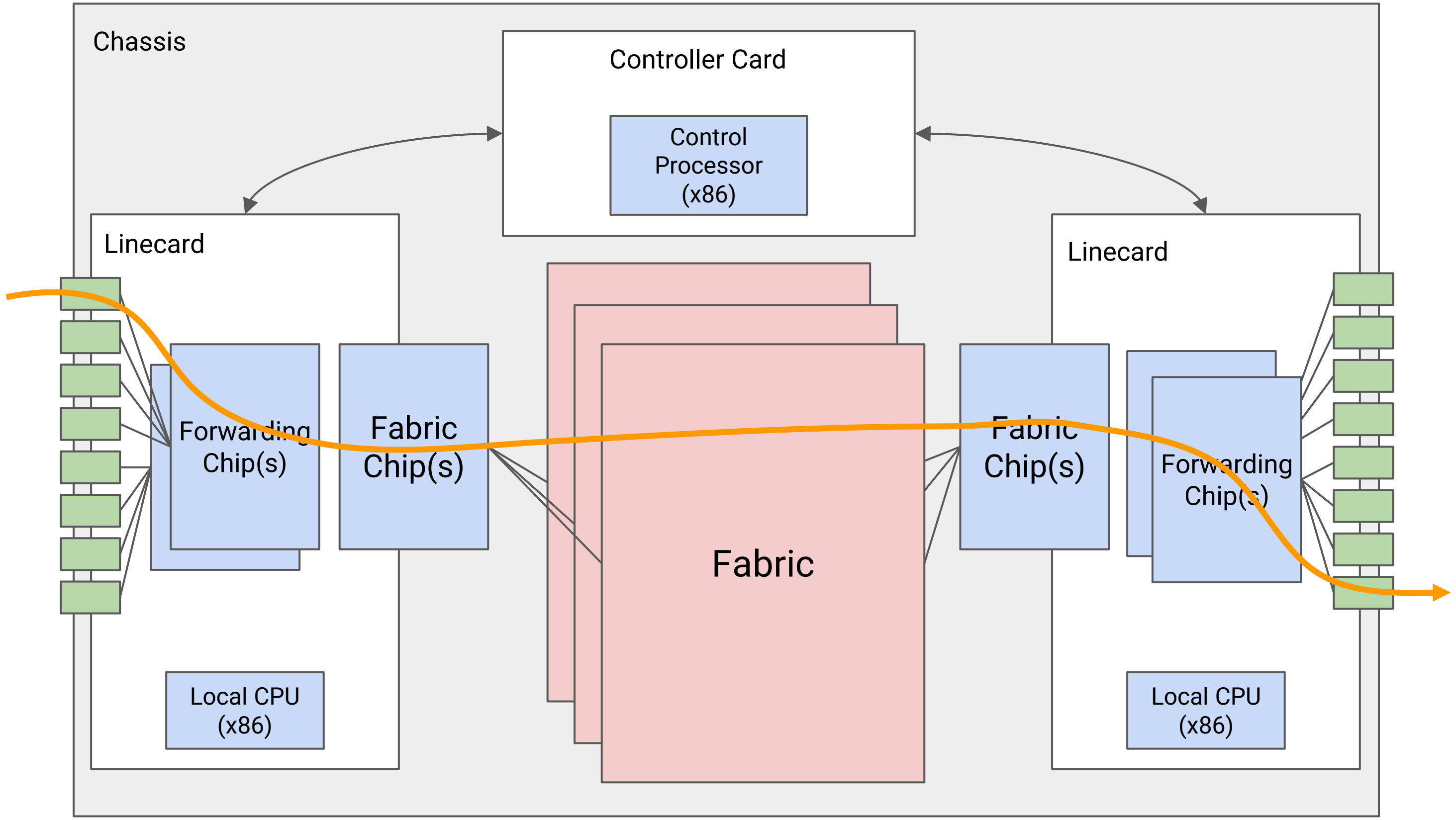
Types of Packets

3 types of packets can arrive at a router.

- **User packet**: The router needs to forward this packet toward its destination.
    - Most common type of packet.
    - Forwarding chip looks up which port to forward this packet.
    - If outgoing port is on a different linecard, send packet through fabric to that linecard.
- **Control plane traffic**: Packets intended for the router itself.
    - Example: Advertisements in routing protocols.
    - Forwarding chip sends the packet to the controller for processing.
- **Punt traffic**: Packets intended for the user, but requiring extra processing.
    - Example: Packet TTL has expired. Need to send back an error message.
    - Forwarding chip "punts" the packet to the controller for processing.

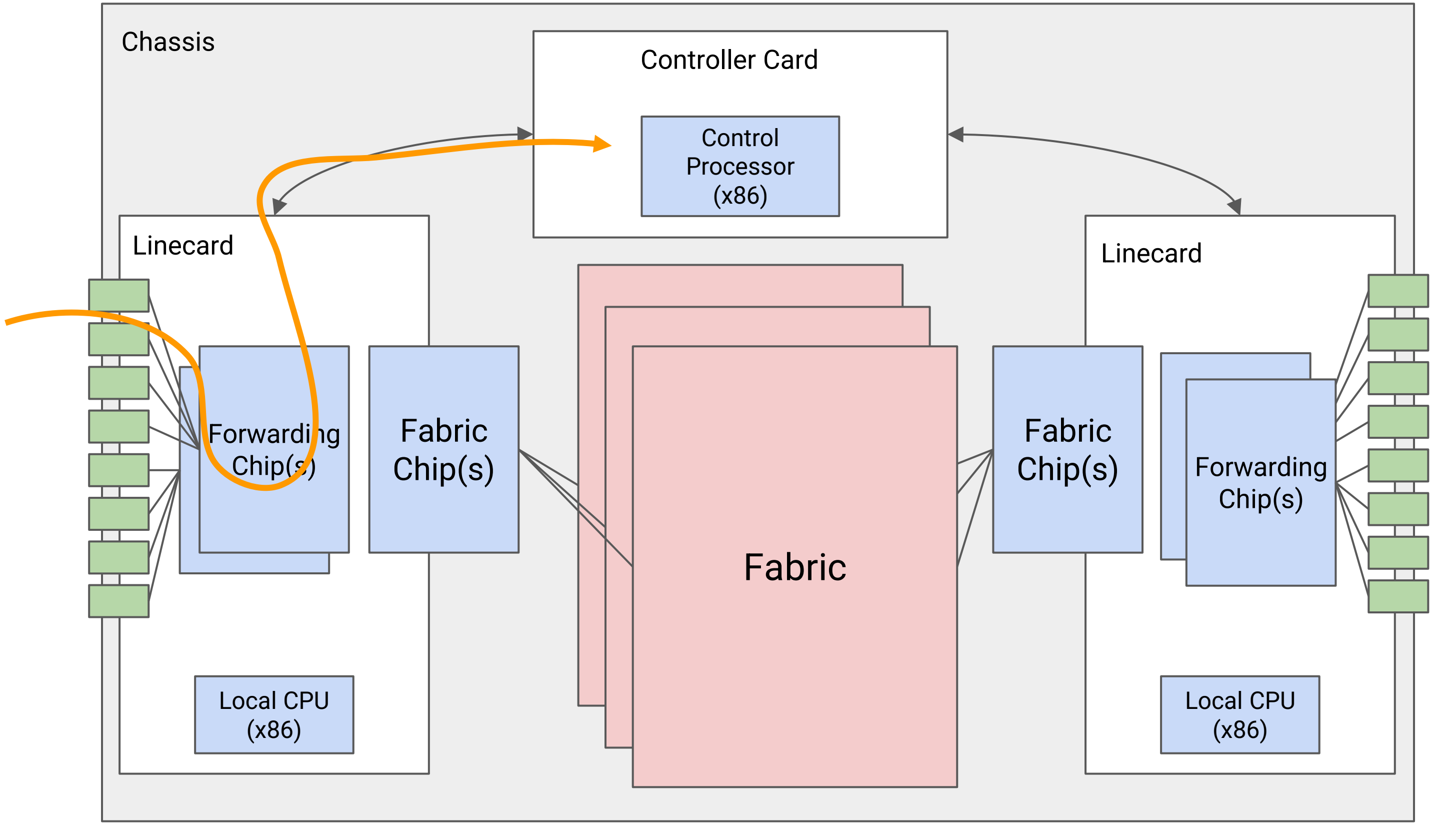
Types of Packets – User Traffic

User packet: Forward according to installed routes.



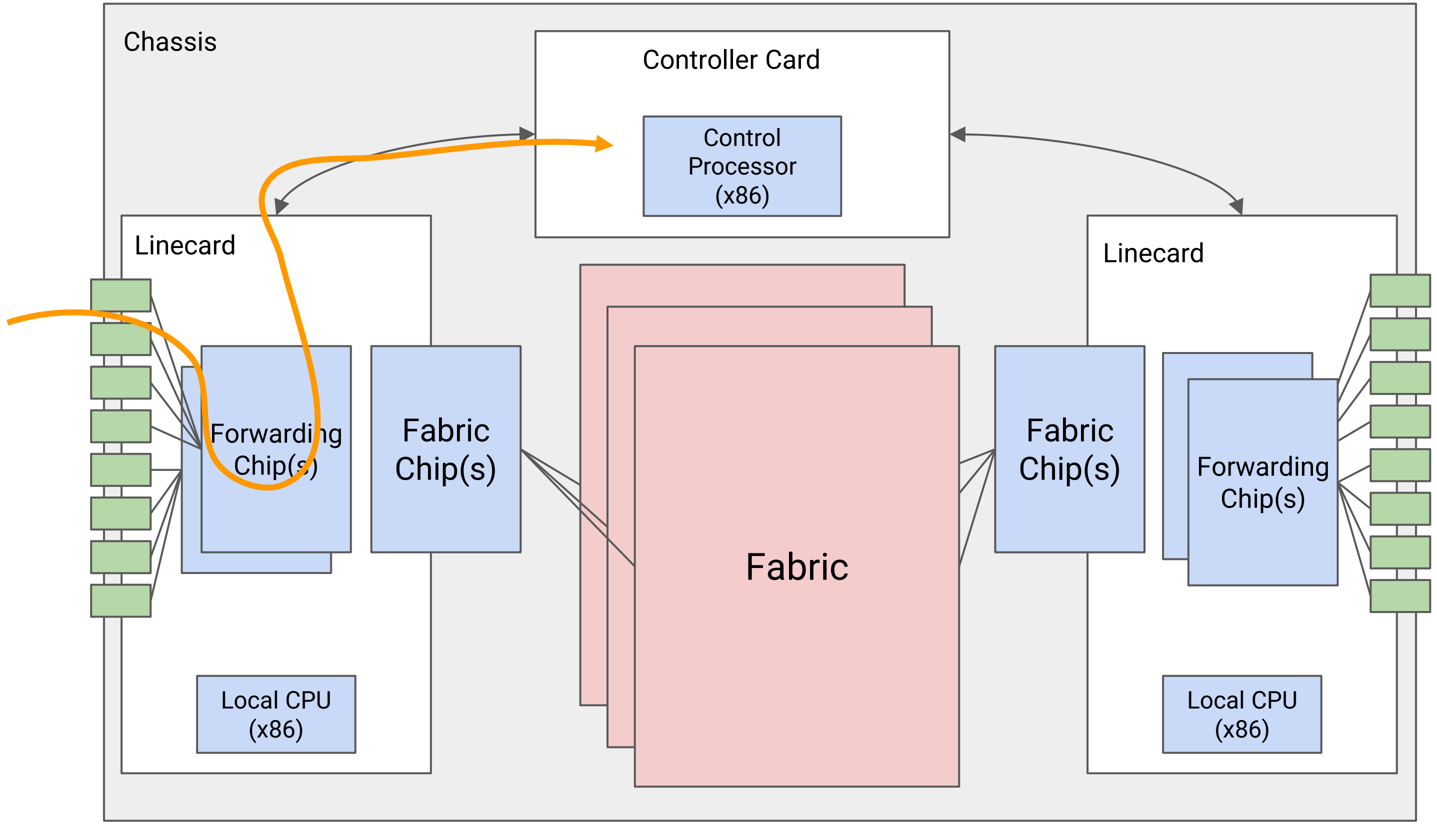
Types of Packets – Control Plane Traffic

Control plane traffic: Packets destined for this router (e.g. routing advertisement).



Types of Packets – Punt Traffic

Punt traffic: User packets that need extra processing (e.g. TTL expired).



Scaling Routers

Why build routers like this? Why not just use a general-purpose CPU?

Reason: *Scale*.

- Assuming 64-byte packets and 400 Gbps, we have to process 781 million packets, per second, per port.
- Across 36 ports, the router has to process 28 billion packets per second.

We can't achieve this scale in software.

- You could write software to forward one packet per 10ms = 0.00001s.
- We need to forward one packet per 10ns = 0.00000001s.

Router functionality must be implemented directly on hardware.

- User packets take the "fast path" in hardware.
- Only use the "slow path" in software when necessary.

Forwarding in Hardware

Lecture 7, Spring 2026

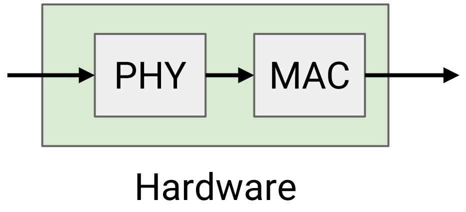
IP Routers

Forwarding Pipeline (1/3)

When a packet arrives, what does the input linecard need to do?

1. Receive the packet from other systems.

- PHY (physical layer): Decode the optical/electrical signal into 1s and 0s.
- MAC (link layer): Perform link-layer operations.
- These are implemented in hardware.

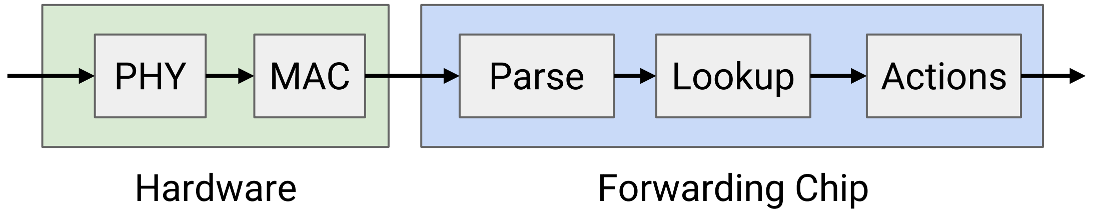


Forwarding Pipeline (2/3)

When a packet arrives, what does the input linecard need to do?

2. Process the packet.

- Parse the packet to understand its headers, e.g. IPv4 or IPv6.
- Look up the next hop in the forwarding table.
- Update the packet.
    - Decrement TTL, update checksum, fragment packet if it's too big, etc.

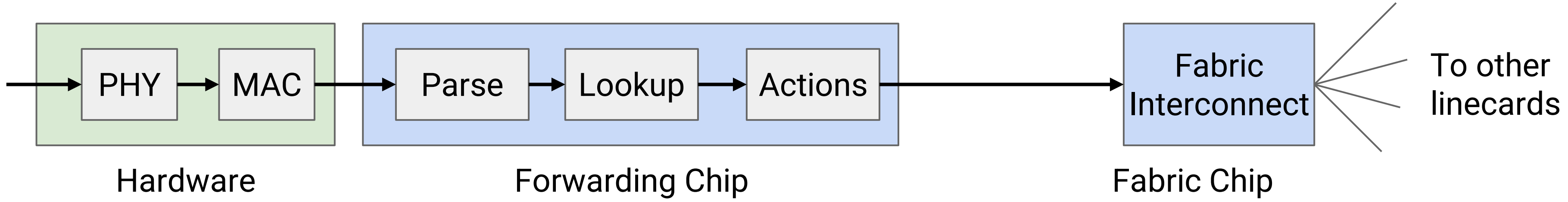


Forwarding Pipeline (3/3)

When a packet arrives, what does the input linecard need to do?

3. Send the packet onwards.

- Fabric interconnect chip sends packets to other linecards via inter-chassis links.

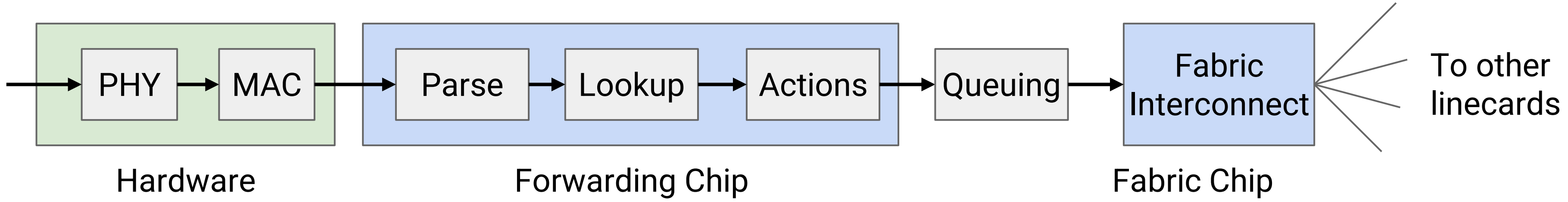


Forwarding Pipeline – Queuing

Many possible queuing approaches: We'll assume the simplest one.

- Classification: Which queue should this packet be put into?
  - One queue per input port? One queue per flow?
  - Assume no classification.
- Buffer management: Should we drop packets?
  - Assume tail drop: Drop packet if queue is full.
- Scheduling: What order do we send out packets?
  - Assume FIFO: Send packets in the order they arrive.

Alternate complex approaches can be used to implement business objectives.



Scaling Forwarding Hardware

Main challenge: *Speed*.

- We have to forward one packet every few nanoseconds.
- One chip is handling packets from many ports.
- We have to do multiple operations to forward a packet.

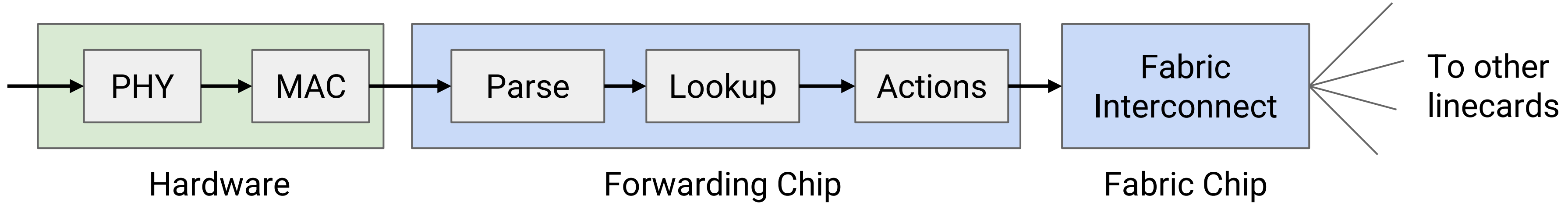
Network processors are specialized to perform forwarding quickly.

- Trade-off: The chip has limited functions. Can't run an arbitrary C program on it.
- Any special processing can be punted to the controller.

Scaling Forwarding Hardware

How hard is it to implement operations in hardware?

- Parse: Easy. Read specific bits of the packet.
- Lookup: ??? ← Doing efficient lookup is our challenge!
- Actions:
  - Some are easy: Update checksum, decrement TTL.
  - Some are harder: Special options, fragment packet if it's too big.
  - The Internet avoids using the harder ones. They usually require punting.
- Fabric interconnect: Dedicated chip with limited features ensures speed.



Efficient Forwarding with Tries

Lecture 7, Spring 2026

IP Routers

Efficient Forwarding

The forwarding table is a map (key-value pairs).

How do we do fast lookups?

Challenges:

- Entries can contain a range of addresses, not just one.
- Ranges might overlap. A destination can match multiple entries.

Naive solution: Write out the whole range.

- Table gets really big.
- If a route changes, we have to update tons of entries.
- We need something smarter.

R2's Table	
Destination	Port
2.1.1.0/24	5

R2's Table	
Destination	Port
2.1.1.0	5
2.1.1.1	5
2.1.1.2	5
2.1.1.3	5
2.1.1.4	5
...	...
2.1.1.252	5
2.1.1.253	5
2.1.1.254	5
2.1.1.255	5

Longest Prefix Match

We want a fast implementation of **longest prefix match**.

- If the address matches multiple prefixes, take the most specific (longest) match.
- If the address matches no prefixes, take the default route.
- If there's no default route, drop the packet.

These two prefixes match. The first one is longer, so use it to forward to Port 5.

R2's Table	
Destination	Port
`11101000 01100101 111..... ........`	**5**
`11101000 01100... ........ ........`	9
`11101100 01100101 111..... ........`	7
`11111... ......... ........ ........`	2

Destination: `11101000 01100101 11101011 11000110`

Longest Prefix Match

Naive longest prefix match implementation:

- For each prefix, check if it fully matches. If yes, add to list.
- Pick the longest prefix in the list.
- If list is empty, use default route (or drop).

Requires scanning every entry. O(*N*) runtime for table with *N* entries.

These two prefixes match.
The first one is longer.

R2's Table	
Destination	Port
`11101000 01100101 111..... ........`	**5**
`11101000 01100... ........ ........`	9
`11101100 01100101 111..... ........`	7
`11111... ........ ........ ........`	2

Destination: `11101000 01100101 11101011 11000110`

Longest Prefix Match

Naive longest prefix match implementation:

- For each prefix, check if it fully matches. If yes, add to list.
- Pick the longest prefix in the list.
- If list is empty, use default route (or drop).

Requires scanning every entry. O(*N*) runtime for table with *N* entries.

These two prefixes match.
The first one is longer.

R2's Table	
Destination	Port
`11101000 01100101 111..... ........`	**5**
`11101000 01100... ........ ........`	9
`11101100 01100101 111..... ........`	7
`11111... ......... ........ ........`	2

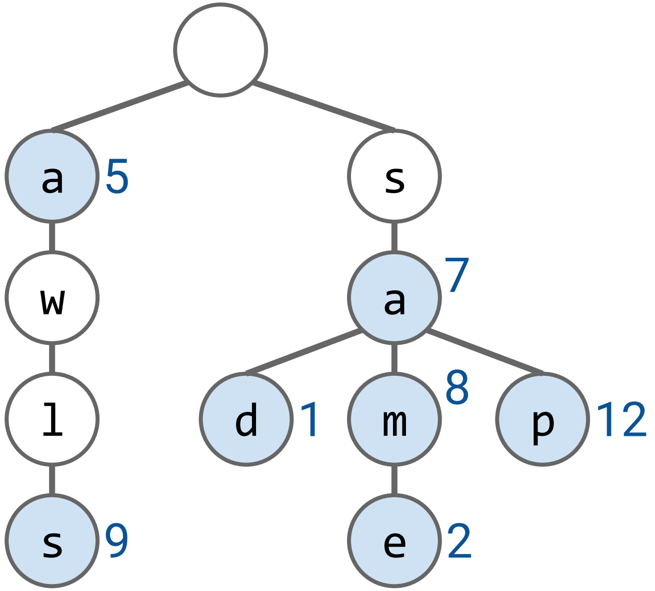
Destination: `11101000 01100101 11101011 11000110`

Tries – Conceptual

Is there a map data structure that supports efficient longest prefix match?

- In a **trie**, also known as a **prefix tree**, each node contains:
  - A “key”: a sequence of “characters” from the alphabet
  - A “value”: the usual Dictionary value
- A node is marked blue if walking from root to that node forms a valid key.

Key	Value
a	5
awls	9
sa	7
sad	1
sam	8
same	2
sap	12

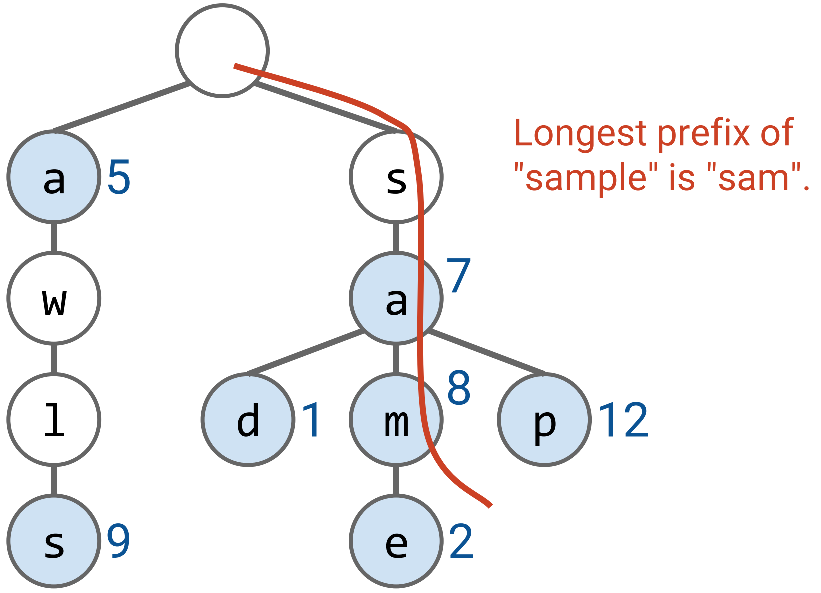


Tries – Conceptual

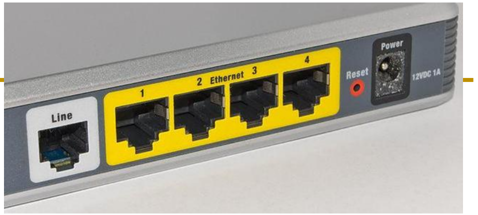
Longest prefix matching on a trie:

- Start at the root, and spell out the word.
- Remember most recent (longest) key (blue node) you see along the way.
- Stop when you're done, or fall off the tree. Return the longest key you saw.

Key	Value
a	5
awls	9
sa	7
sad	1
sam	8
same	2
sap	12



Efficient IP Lookup with Tries



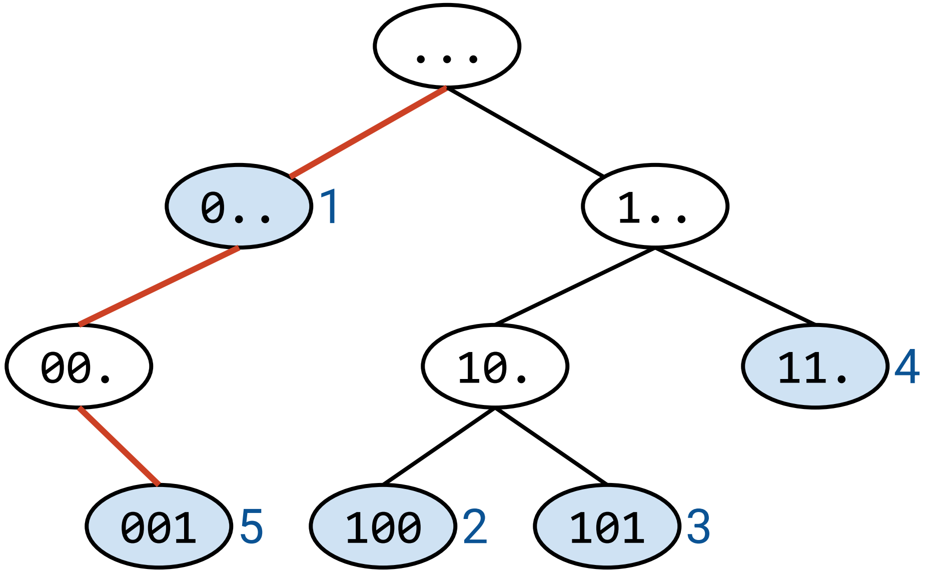
Physical port: The hole you plug a cable into. Exists in hardware.

Longest prefix matching on a trie:

- Start at the root, and spell out the word.
- Remember most recent (longest) key you see along the way.
- Stop when you're done, or fall off the tree. Return the longest key you saw.

Key	Value (Port #)
`0..`	1
`100`	2
`101`	3
`11.`	4
`001`	5

Longest prefix of `00100` is `001`.



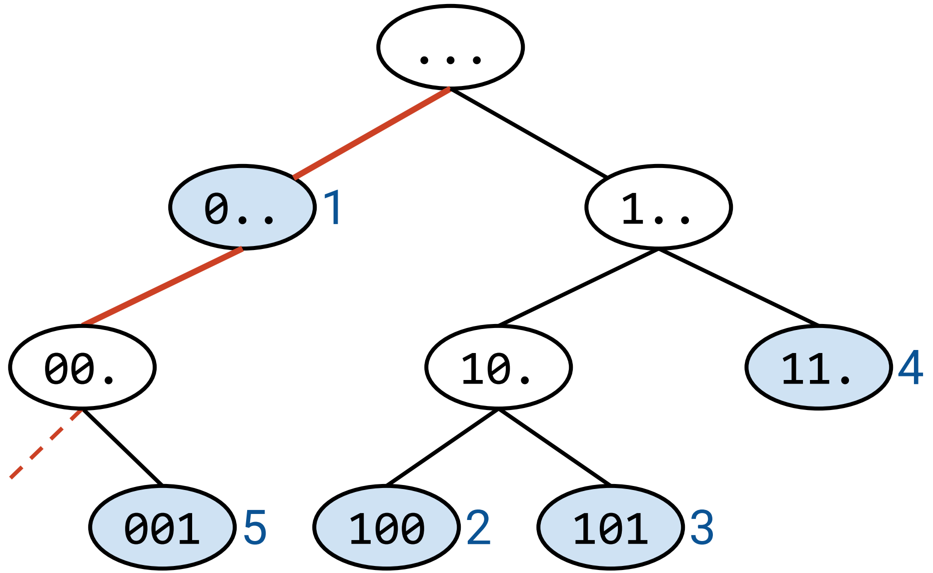
Efficient IP Lookup with Tries

Longest prefix matching on a trie:

- Start at the root, and spell out the word.
- Remember most recent (longest) key you see along the way.
- Stop when you're done, or fall off the tree. Return the longest key you saw.

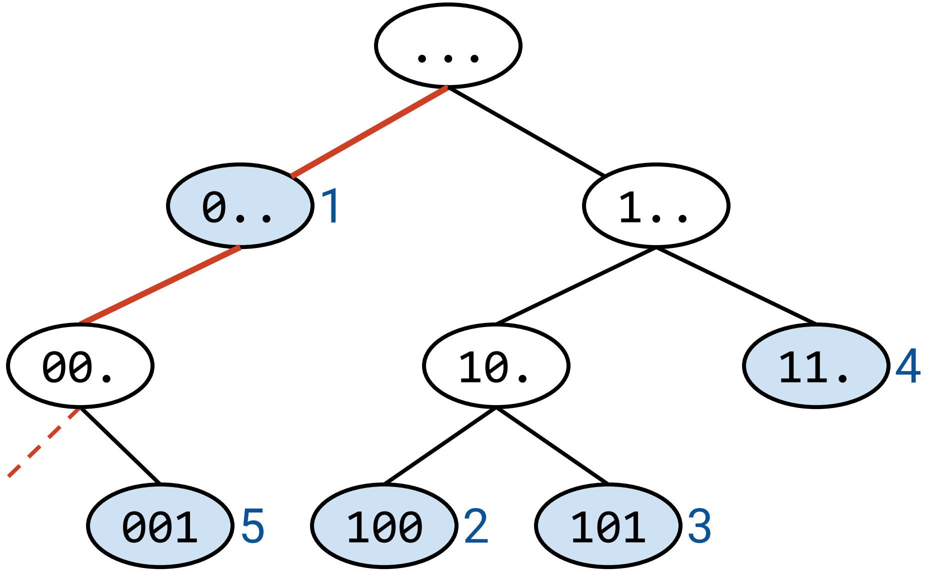
Key	Value (Port #)
`0..`	1
`100`	2
`101`	3
`11.`	4
`001`	5

Longest prefix of `00000` is `0`.



Efficient IP Lookup with Tries

- In the trie, a dot **means "don't care / anything can follow."**
- A node labeled with dots represents an **IP prefix**, not a full address.
- 0.. means:
  • the first bit is 0
  • the remaining bits can be **anything**
- In CIDR terminology
  - 0.. is a **/1 prefix,** and *every* IP address starting with 0 matches this prefix.
  - 00. is a **/2 prefix,** and *every* IP address starting with 00 matches this prefix.
  - Nodes without dots, like 001, 100, 101 are **exact prefixes** of that length.
- As you traverse the trie bit by bit:
  - Every node with dots is a **valid forwarding entry.** You keep track of the *last* such node you saw. When you fall off the tree or finish the address, you return the **longest (deepest) dotted prefix**

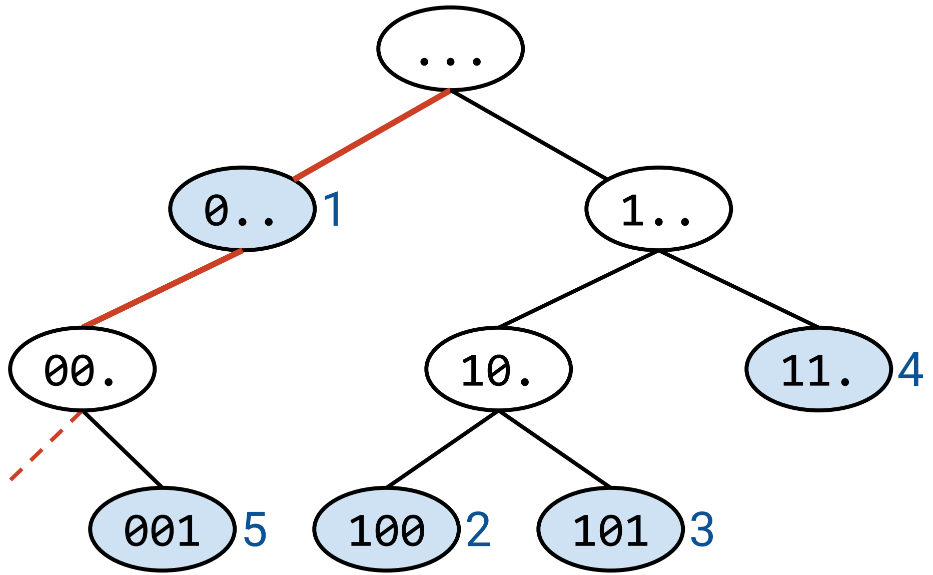


Efficient IP Lookup with Tries

Longest prefix matching on a trie:

- Start at the root, and spell out the word.
- Remember most recent (longest) key you see along the way.
- Stop when you're done, or fall off the tree. Return the longest key you saw.
- The longest prefix is the deepest node on the search path that has an associated value (a blue node).

Key	Value (Port #)
0	1
100	2
101	3
11	4
001	5



Longest prefix of 00000 is 0.

Even though 00 is a prefix of 00000, it is not stored as a key (it has no value in the trie).

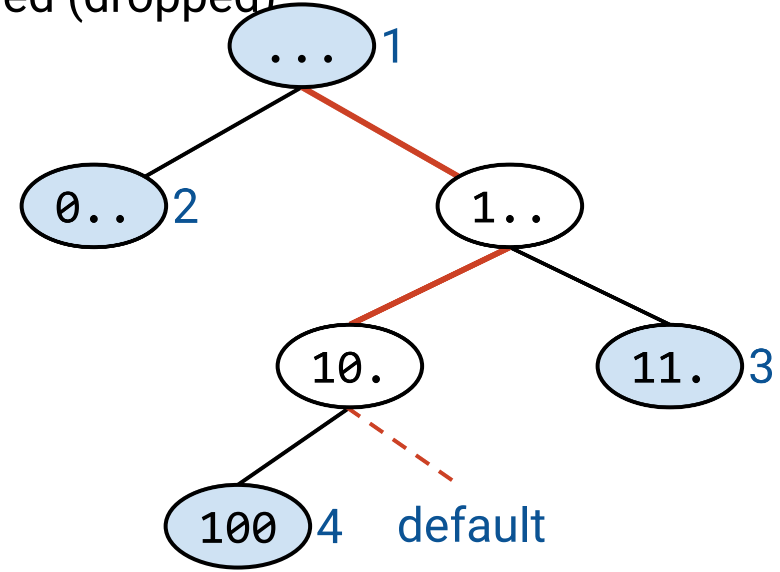
Efficient IP Lookup with Tries

Longest prefix matching also works with default route.

- If you see another prefix, it will be returned over the default route.
- If you finish spelling or fall off the tree without seeing any other prefix, the default is returned.
- Routers usually have a 'default route' (0.0.0.0/0) for unmatched traffic. If even that is missing, the packet is discarded (dropped).

Key	Value (Port #)
`...`	1
`0..`	2
`11.`	3
`100`	4

Longest prefix of `10100` is default.



Efficient IP Lookup with Tries

Runtime of longest prefix matching in a trie is *constant*: O(1).

- We'll only visit at most 32 nodes from spelling out the IP address.

All routers have efficient longest prefix matching functionality.

Some use more complex solutions with heuristics and optimizations.

- Some destinations are more popular than others.
- Some ports have more destinations.
- Longest prefix to external networks is /24 (e.g. you won't see a 29-bit prefix).
- We could optimize for fast trie/table updates too.

Summary: IP Routers

Routers have different planes.

- Data plane: Forward packets.
- Control plane: Programming forwarding entries and handling exception packets.
- Management plane: Configure and monitor router functionality.

Data plane leverages tradeoffs in software vs. hardware packet processing.

- Software: Flexible but slow.
- Hardware: Inflexible but fast.

Data plane challenges: Speed!

- Some operations are easy, e.g. update packet header.
- Use tries for efficient longest-prefix lookup on destination address.