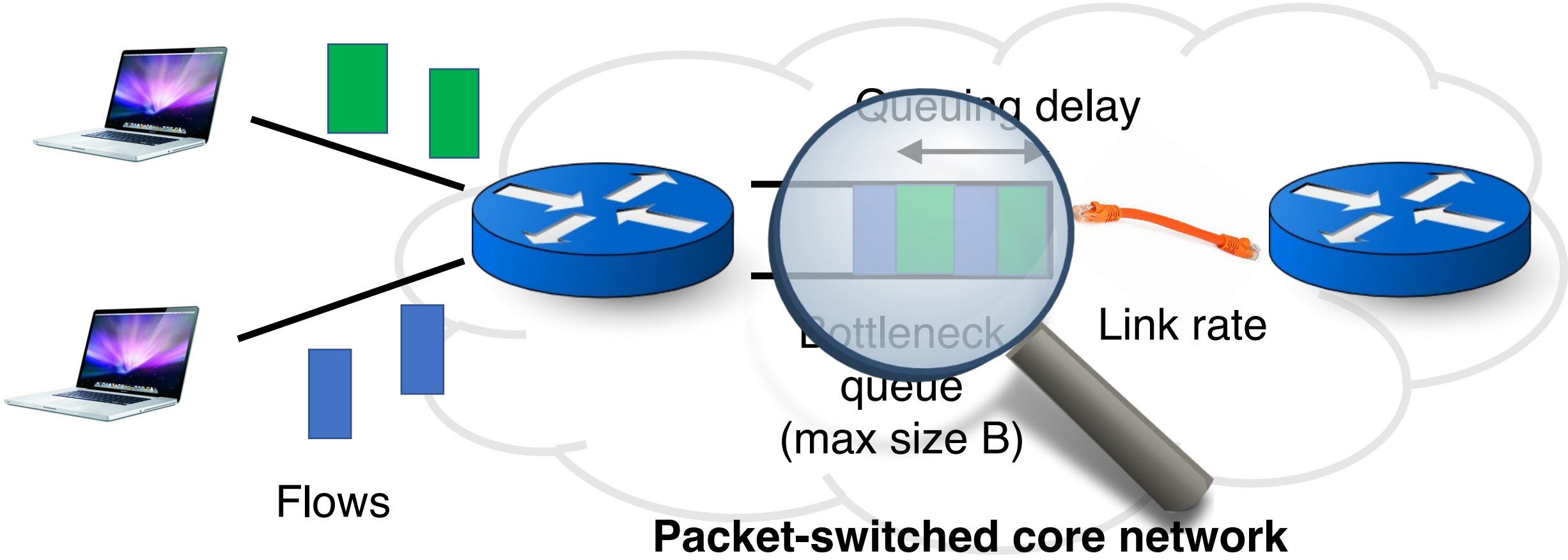
Transport

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Packet scheduling



Packet Scheduling Algorithms

Which packet to send next? (order)

When to send the next packet? (timing)

Providing Isolation through Rate Limiting

Used to isolate flows from each other by giving each a fixed data rate through a link

Three commonly used terms:

- *(long term) average rate:* how many pkts can be sent per unit time (in the long run)
  - crucial question: what is the interval length? 100 packets per sec or 6000 packets per min have same average, but instantaneous behaviors can be very different
- *peak rate:* e.g., 6000 pkts per min (ppm) avg.; 1500 ppm peak rate
- *(max.) burst size:* max number of pkts sent consecutively (with no intervening idle)

Shaping and Policing

Policing	Enforces rate by *dropping* excess packets immediately – Can result in high loss rates + Does not require memory buffer + No RTT inflation
Shaping	Enforces rate by *queueing* excess packets + Only drops packets when buffer is full – Requires memory to buffer packets – Can inflate RTTs due to high queueing delay

Token Bucket

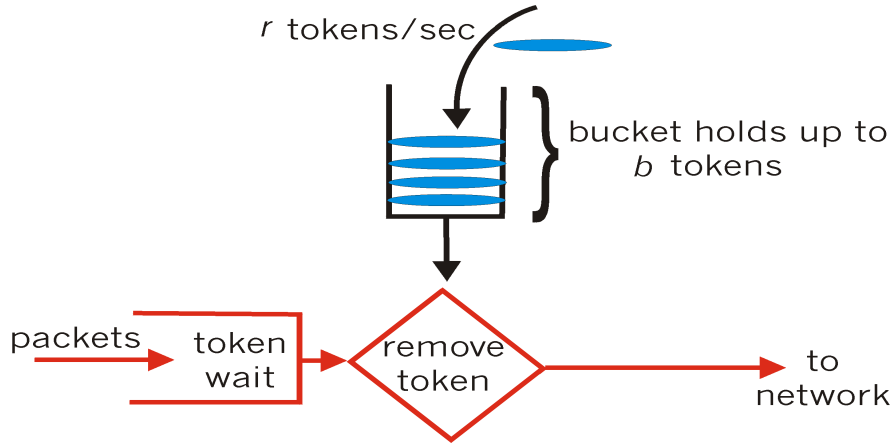
token bucket: limit input flow to specified *burst size* and *average rate*

- Tokens generated at rate *r tokens/sec,* put into bucket
- Bucket can hold b tokens. Tokens are not added if bucket is full
- A packet can be transmitted successfully if a token is available
  - Packet "picks up" the token as it leaves the router
- Over interval of time t:

number of packets that leave the token bucket is less than or equal to (r * t + b)

Average rate r

Burst size b



Purpose of the bucket

- Suppose each flow starts with a “full” bucket (b tokens)
- Bucket of tokens enables small bursts to go through unscathed
- Short flows (bursts) can exceed rate limit r, since they can use up the reserve in the bucket
- Long flows will fit into the rate limit r over their lifetime, since they cannot exceed the average token-filling rate r after using up the reserve

Token Bucket Shaper vs. Policer

- Shaper: there is a bucket for tokens and a packet buffer hold packets waiting for tokens
  - Packets will be delayed if the buffer can hold the packet
  - Dropped when the buffer is full

- A token bucket policer doesn't contain the packet buffer:
  - a packet arriving while the bucket is empty is dropped right away

Token Buckets are simple to implement

- Assume byte-based rates. Each time a packet from a flow arrives, refill tokens according to gap in time and bucket size

```
tokens += r * time since last packet of flow
tokens = min(tokens, b)
```

- Need enough tokens to dequeue a packet: `tokens >= pkt_size`
- When a packet departs, remove tokens: `tokens -= pkt_size`

- Small per-flow state: r, b, timestamp, current # of tokens
  - Used to build more complex metering like TR-TCM (RFC 2698)

The Internet uses token bucket policers at several bottleneck links.

Impact of Token Bucket Policers

Region	Policed segments		Loss rate	
	(among lossy)	(overall)	(policed)	(non-pol.)
India	6.8%	1.4%	28.2%	3.9%
Africa	6.2%	1.3%	27.5%	4.1%
Asia (w/o India)	6.5%	1.2%	22.8%	2.3%
South America	4.1%	0.7%	22.8%	2.3%
Europe	5.0%	0.7%	20.4%	1.3%
Australia	2.0%	0.4%	21.0%	1.8%
North America	2.6%	0.2%	22.5%	1.0%

Table 2: % segments policed among lossy segments ($\geq$ 15 losses, the threshold to trigger the policing detector), and overall. Avg. loss rates for policed and unpoliced segments.

Flach et al., An Internet-Wide Analysis of Traffic Policing, SIGCOMM 2016

Impact on TCP

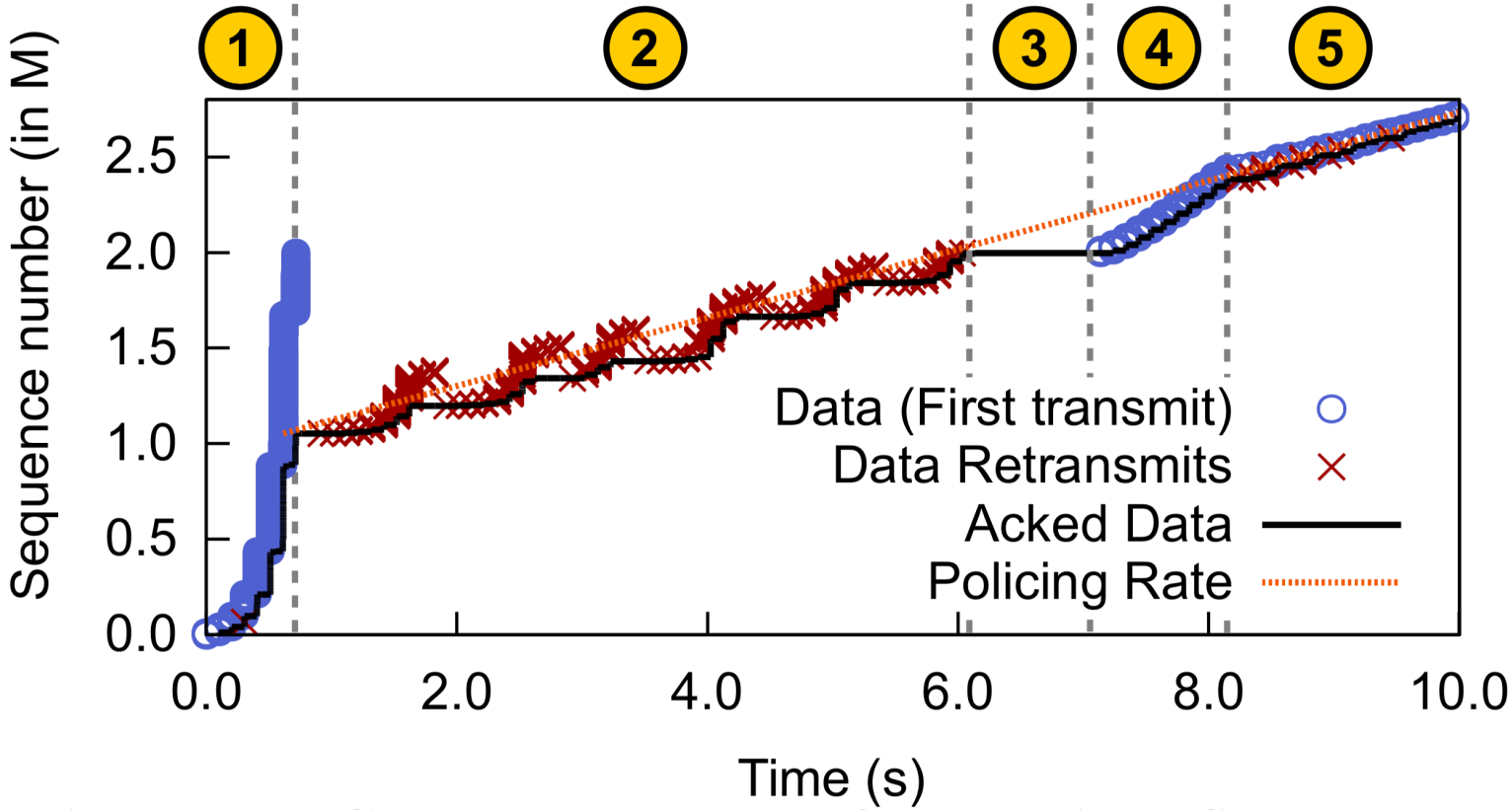


Figure 1: TCP sequence graph for a policed flow: (1 and 4) high throughput until token bucket empties, (2 and 5) multiple rounds of retransmissions to adjust to the policing rate, (3) idle period between chunks pushed by the application.

Policing losses impact applications

- Video rebuffer rate: rebuffer time / overall watch time

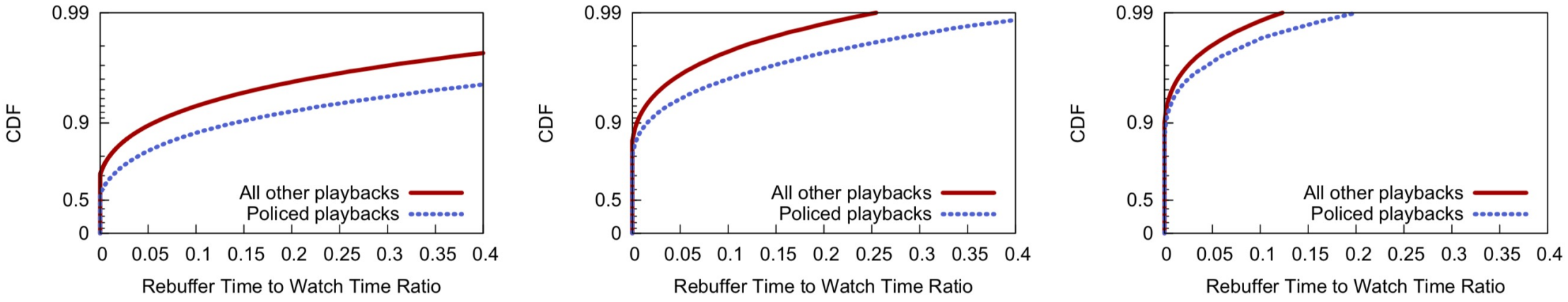


Figure 9: Rebuffer to watch time ratios for video playbacks. Each had at least one chunk with a goodput of 300 kbps, 1.5 Mbps, or 3 Mbps (±15%).

Flach et al., An Internet-Wide Analysis of Traffic Policing, SIGCOMM 2016

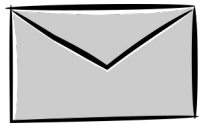
Transport topics we didn’t consider

- Multipath transport
- Network utility maximization

Network Data Plane

RUTGERS
UNIVERSITY | NEW BRUNSWICK

The Internet



Applications

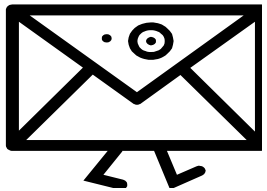
Transport

Network

Link layer

Packet starts as an app “payload”

Packet takes on headers (metadata) at each layer



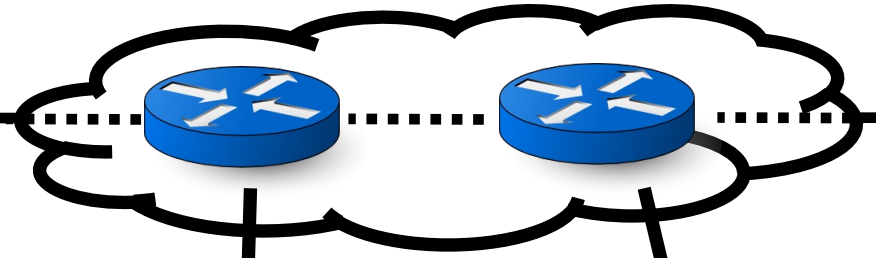
The Internet

Applications

Transport

Network

Link layer



Applications

Transport

Network

Link layer

Network

Link layer

Network

Link layer

Applications

Transport

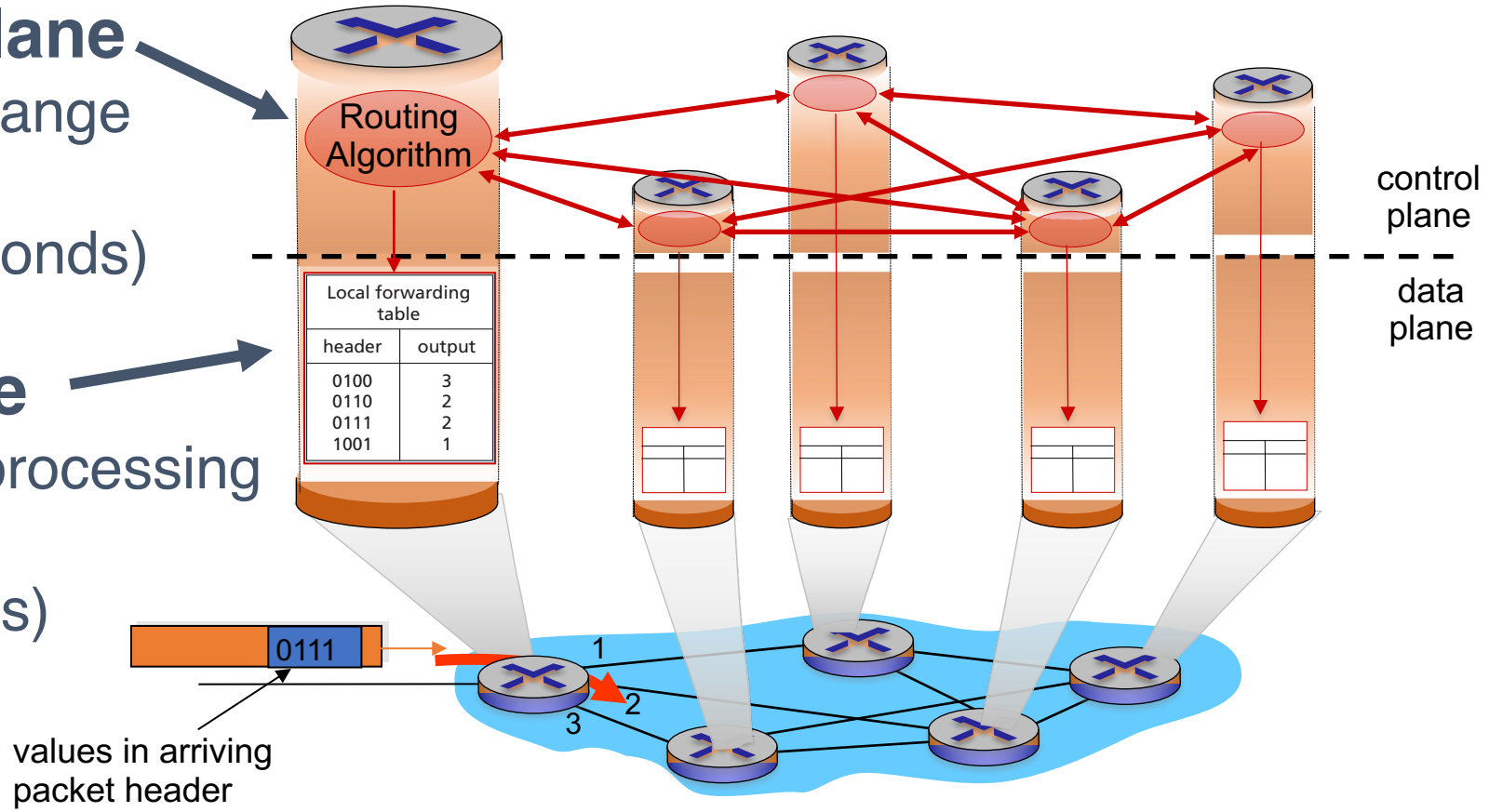
Network

Link layer

Control & Data Planes inside a router

Control plane
per route-change processing
(~ a few seconds)

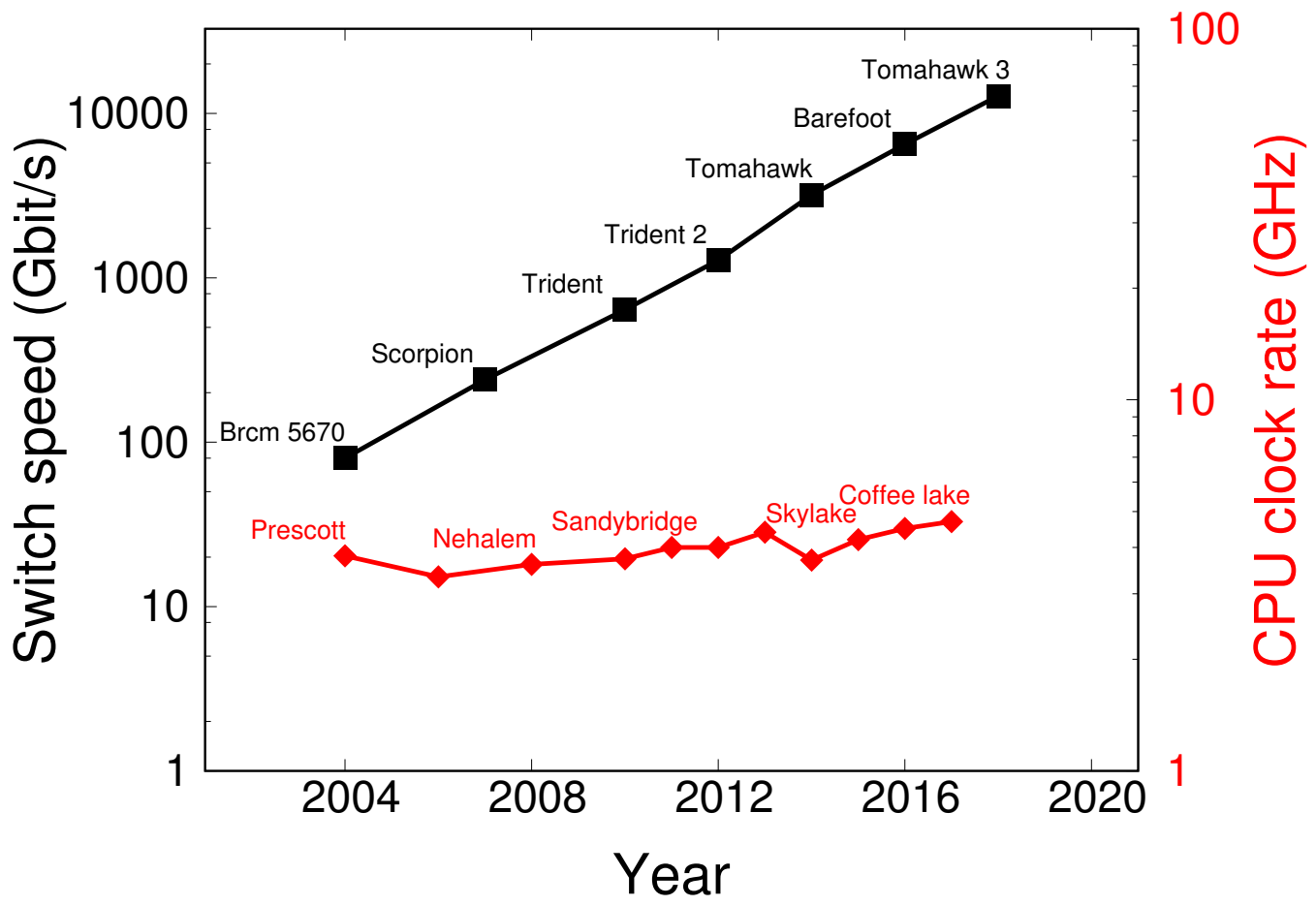
Data plane
per-packet processing
(~ tens of nanoseconds)



Traditionally, Distributed Control: Individual routing algorithm components *in every router* interact in the control plane

Router Design: Requirements

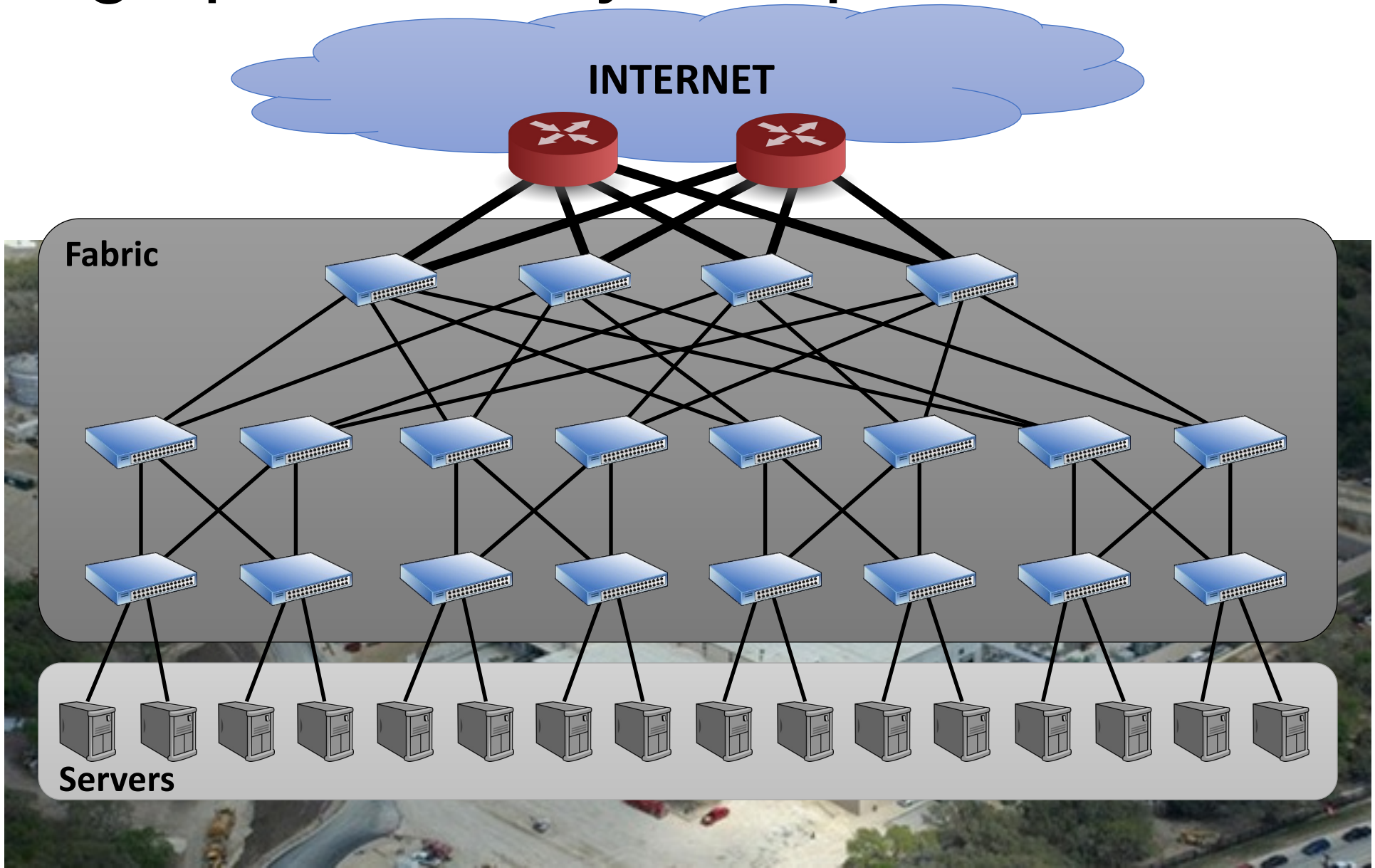
1. Speed: move packets quickly



(1) Inherently parallel workload; speed up through hardware parallelism

(2) Physical communication technologies (e.g., optical) not bound by semiconductor speed

2. High port density, low power & area



3. Make routers follow top-level intent

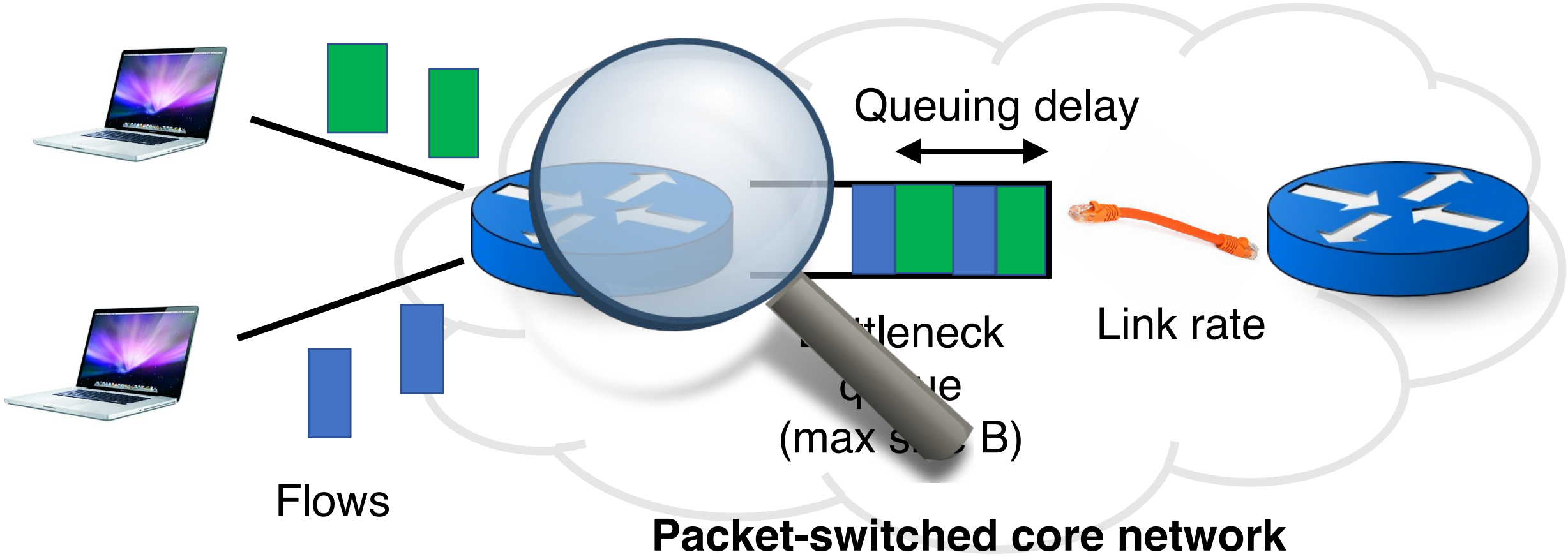
- Cloud era: special challenges
  - Scale: don’t want to configure manually
  - Multiple kinds of addressing
  - Telemetry
- Expressiveness:
  - Routing with global/centralized goals
  - Functions beyond simple forwarding:
    - access control, in-network computing, perf debugging, …
  - Scheduling policy
- Focus on programmability and observability

3.5 Standard router specifications

- RFC 1812: Forward pkts using a table lookup, but also …
- Update TTL (time-to-live IP header field): `ttl -= 1`
- Update IP checksum (TTL updated)
- IP to link-layer translation across networks (e.g., ARP for dst)
- Rewrite link layer source address
- Special processing (IP options): source route, record route, …
- Handle IPv4 packet fragmentation
- Handle multicast packets
- Maintain QoS assurances

Requirements for IP Version 4 Routers

What's in a router?

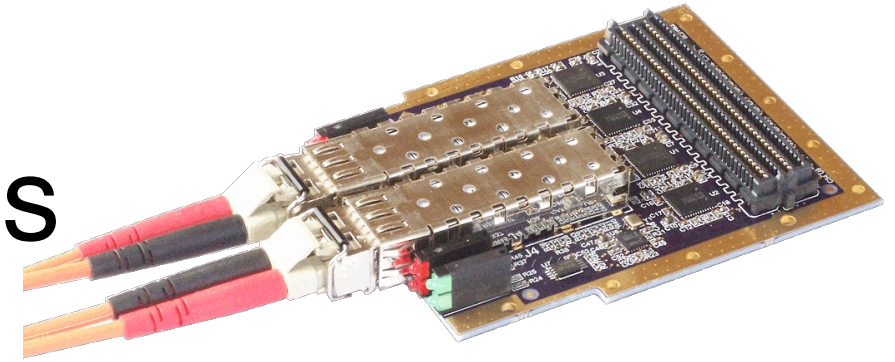


Overview of router functionality

- Historically evolving, multiple concurrent router designs
- 2 exemplars:
  - MGR: router from the late 1990s (50Gbit/s router, Partridge et al)
  - RMT: router from the late 2010s
- Many things in common. Some functions implemented:
  - Packet receive/transmit from/to physical interfaces
  - Packet and header parsing
  - Packet lookup and modification: ingress & egress processing
  - High-speed switching fabric to connect interfaces
  - Traffic management: Scheduling & Buffer Management

Life of a Packet

(1) Receive data at line cards



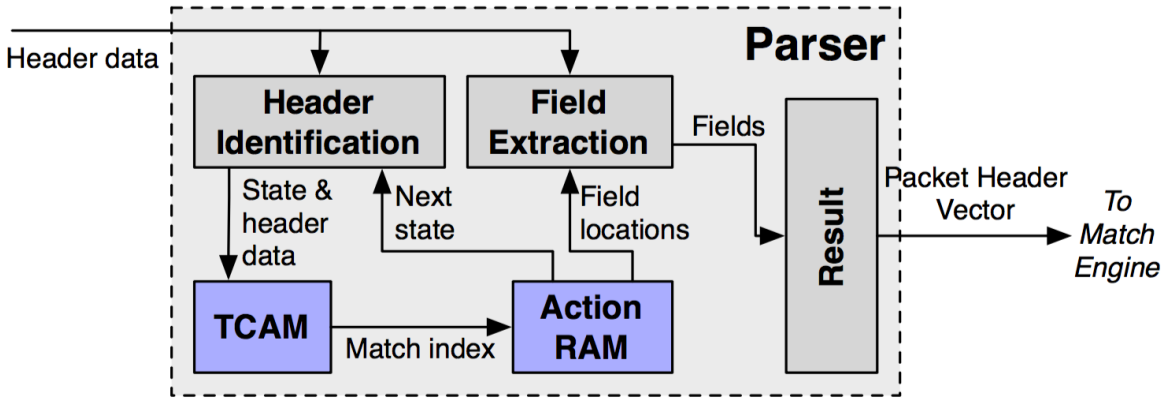
- Circuitry to interface with physical medium: copper, optical, …
  - SerDes/IO modules: serialize/deserialize data from the wire

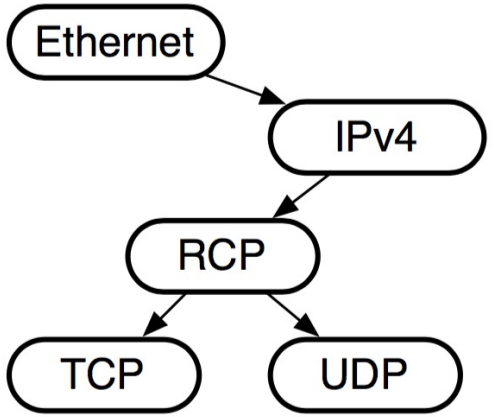
- Network interfaces keep getting faster
  - More parallelism, but stay the same size; Moore's law is alive for now
  - Link technologies still getting faster: optical & wireless

- Multiple network interfaces on a single "line card"
  - Component detachable from the rest of the switch
  - Ex: upgrade multiple 10 Gbit/s interfaces to 40 Gbit/s in one shot

(2) Packet parsing

- Extract header fields: extraction & branching, sometimes “loop”
- Example: extract Ethernet headers to know src/dst hw addresses
- Example: determine transport protocol based on IP proto field
- Example: encapsulation, e.g., IPv6 in IPv4
- Outcome: assignment of header fields to bits from packet
- MGR: software, RMT: hardware (programmable)





(2) Packet parsing

- A key principle: Separate header and payload data paths
  - Router functionality is “header-heavy” but “payload-light”
  - Don’t move the payload around too much
  - Conserve bandwidth & resources for data moved around inside the router
- Header goes on as input to route lookup/packet modification
- Payload sits on a buffer until router knows what to do with pkt
  - Buffer could be on the ingress line card (MGR) or a buffer shared between line cards (RMT)