CSC2229 – Computer Networks for Machine Learning

Handout # 2:
Course Logistics and Introduction

Professor Yashar Ganjali
Department of Computer Science
University of Toronto

ganjali7@cs.toronto.edu
http://www.cs.toronto.edu/~yganjali

Today

- Outline
  - What this course is about
- Logistics
  - Course structure, evaluation
  - What is expected from you
  - What you want to know
- Introduction
  - Packet switching systems
  - Data-Center Networks
  - Networks and Machine Learning

CSC229: Computer Networks for ML

- Graduate level course
  - Core challenges of interconnection networks
    - Emphasis on machine learning applications
  - + new opportunities

- Course Structure
  - Introductory Lectures: 3 weeks
  - Research Papers: 7-8 weeks
  - Final Project Presentations: 1-2 weeks

- Theory + Practice
  - Switching systems are simple enough for us to prove something about them
  - Yet they are complex enough to work in practice

Outline – Part I

- Packet switch systems
  - Basic architectural components
  - Examples
  - Evolution of the Internet, and Internet routers
- Data Center Networks
  - Design Decisions
  - Topology
  - Transport
- Network Programming
  - Software-Defined Networking
  - Programmable Switches
- Network-Application Integration

Outline – Part II

- Networks for Machine Learning: Challenges and Opportunities
  - Hyperscale data center networking
  - Switch and controller design
  - Reliability, monitoring, and fault tolerance
  - Network optimization
  - High-performance transport in data centers,
    - Congestion control,
    - Flow control,
    - Flow scheduling, and prioritization
  - Network-Application Integration, reconfigurable data center networks

- Machine learning for computer networks
  - Traffic prediction & forecasting
  - Congestion control
  - Routing and load balancing
  - Automation
  - Management and troubleshooting

Logistics

- **Class time**: Tuesday 1 PM – 3 PM
  - **Location**: BA 2145
- **Office hours**: Tuesday 3 PM - 4 PM
  - Or by appointment
  - **Location**: BA 5238
- Course web page
  - http://www.cs.toronto.edu/~yganjali/teaching/csc2229-winter-2026/
  - Please check regularly for updates/announcements.
- Teaching Assistant
  - Parsa Pazhooheshy <parsap @ cs . toronto . edu >

Mailing List, Bulletin Board

- Bulletin board
  - We will use Piazza for announcements and Q&A
    - https://piazza.com/utoronto.ca/winter2026/csc2229
    - Sign up link on class web site
  - Post any questions related to the course here.
  - Check previous posts before asking a question.

- Class mailing list
  - Based on e-mail address you have defined on ACORN.
  - The TA and I will use this list for announcements only.
  - Do not send e-mails to this list!

Prerequisites & Papers

- Prerequisites
  - Any introductory course on networking (e.g., CSC458/2209)
  - Algorithms, probability theory, etc.

- Papers
  - Will be listed on class web page
  - Please read suggested papers BEFORE the class

Grading & Deadlines

- Grading
  - Paper presentations: 20%
  - Final project: 70%
    - Proposal: 5% - 1-2 pages – Due: Feb. 13
    - Intermediate report: 10% - 3-4 pages – Due: Mar. 13
    - Presentation: 20% - 25 minutes – Last two weeks of classes
    - Final report: 35% - 6-8 pages – Due: Apr 3.
  - Active participation in class and discussions: 10%

- Deadlines
  - Free late submission: two days
    - Use on your proposal, intermediate report, or final report.
    - Notify the TA before the deadline.
  - 10% deduction for each day of delay, up to 2 days.

Paper Presentations

- We will read and discuss papers from the top-tier conferences in the area.
- Each student will present one paper throughout the semester
  - + final project presentation which is in groups
- All students are expected to read the paper beforehand.
- The presentation will be followed by discussion (all of us).

Final Projects

- Final Project
  - Groups of 2-3 students

- Final Project Topic
  - Consult with the instructor to choose a problem
    - Choose 1-2 problems from the offered list
  - Replicate & Improve
    - Choose a paper related to the course topic, replicate existing solution, and improve (bonus)
  - Survey of existing solutions
    - Create new insights by looking at certain problems/solutions from different perspectives
  - Apply your background/ideas
    - Identify challenges and opportunities in the areas covered in the course to apply your own ideas.

- Please discuss your ideas with me before submitting a proposal.

Academic Integrity

- All submissions must present original, independent work.
- We take academic offenses very seriously.
- Please read
  - Handout # 1 (course information sheet)
  - "Guideline for avoiding plagiarism"
  - http://www.cs.toronto.edu/~fpitt/documents/plagiarism.html
  - "Advice about academic offenses"
  - http://www.cs.toronto.edu/~clarke/acoffences/
- Use of AI tools: OK to use for learning. Cannot be submitted as your original work.
  - Please see Handout #1 for more information.
  - Need to clearly state what/how you have used it.

Accessibility

- Accessibility Needs
  - The University of Toronto is committed to accessibility. If you require accommodations or have any accessibility concerns, please visit http://studentlife.utoronto.ca/accessibility as soon as possible.

Acknowledgements

- Special thanks to:
  - Prof. Nick McKeown, Stanford University
  - Prof. Balaji Prabhakar, Stanford University
  - Prof. Jennifer Rexford, Princeton University
  - Prof. Nick Feamster, University of Chicago

Questions?

What else do you like to know about this course?

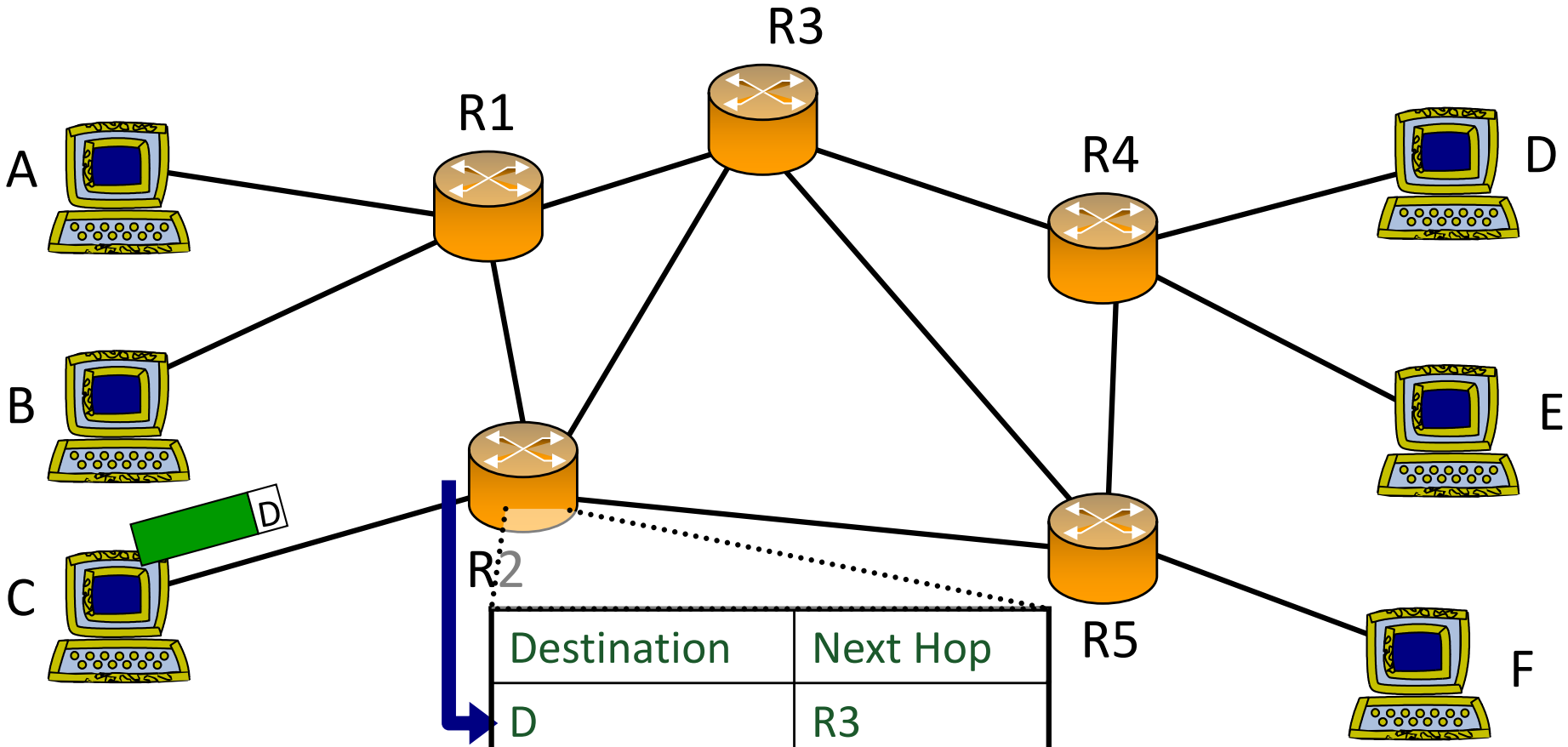
Before We Start …

- Need volunteers for paper presentations
  - Presentation in week 4 (Feb. 3rd)
  - Bonus for people who volunteer to present first

Introduction

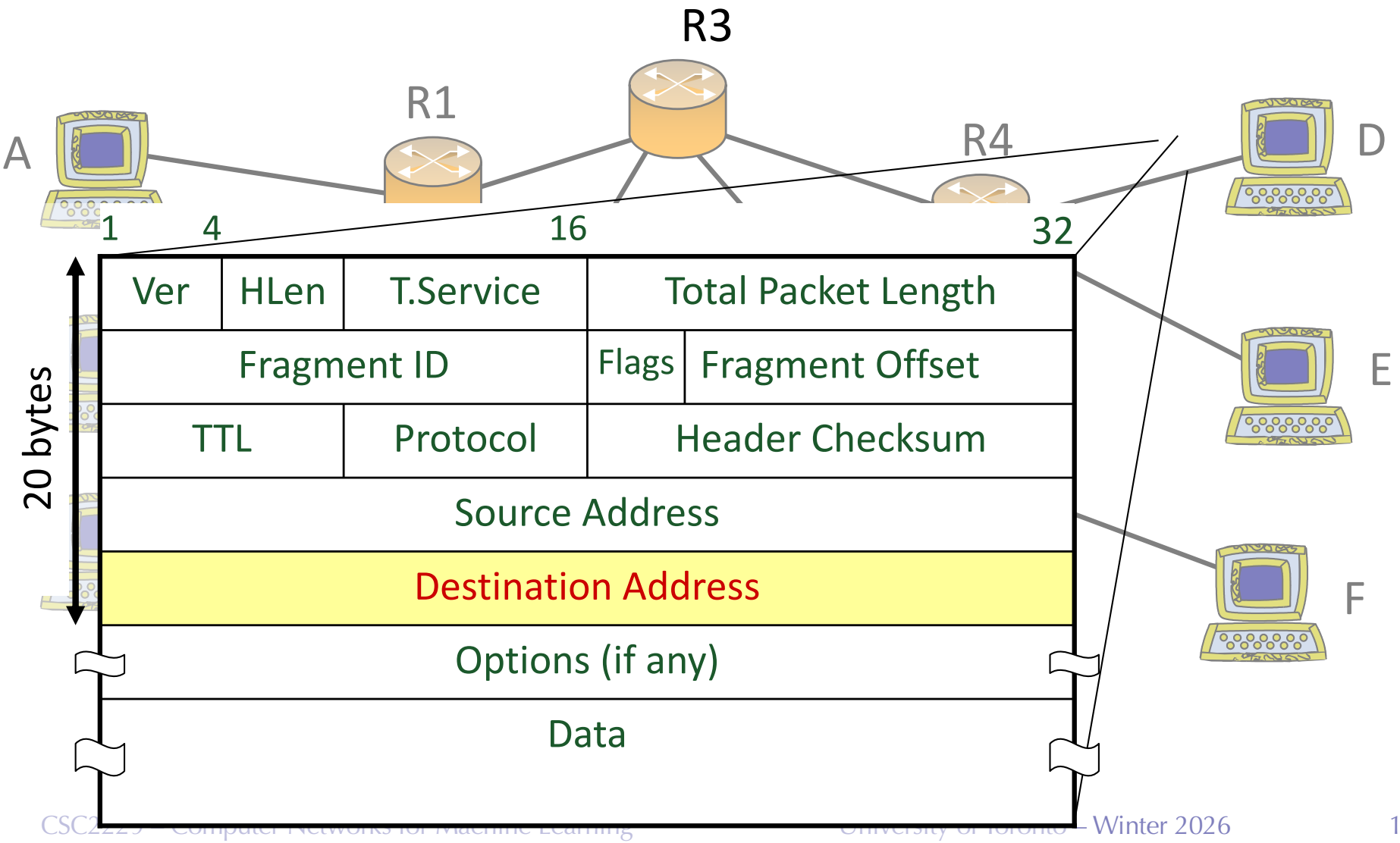
- Background
  - What is a router?
  - Basic router architecture
- Basic packet processing in routers
  - IP address lookup
  - Packet buffering
  - Switching
- Data Center Networking
  - Design Dimensions
  - Topology
  - Transport
- Machine Learning and Computer Networks

What is Routing?

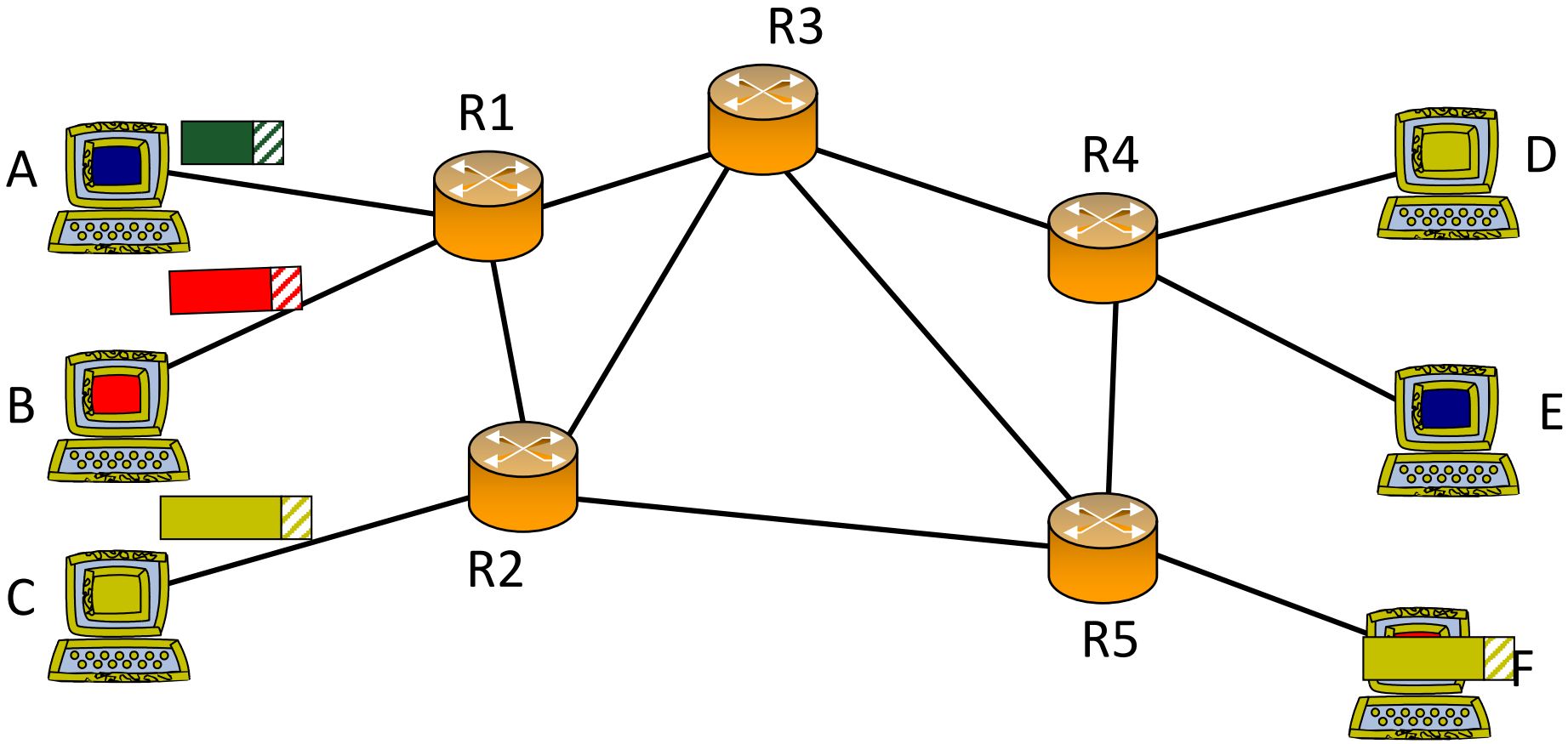


Destination	Next Hop
D	R3
E	R3
F	R5

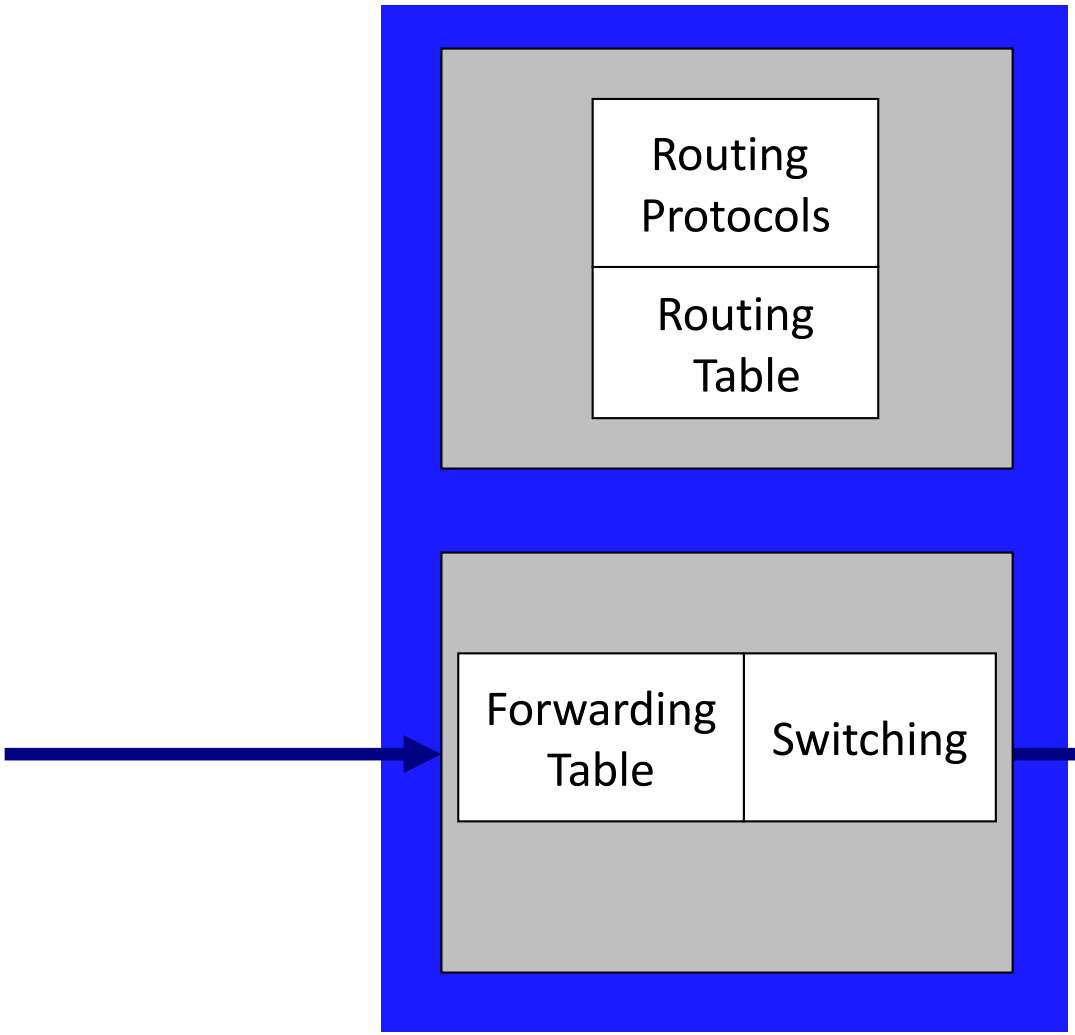
What is Routing?



What is Routing?



Basic Architectural Components of a Router

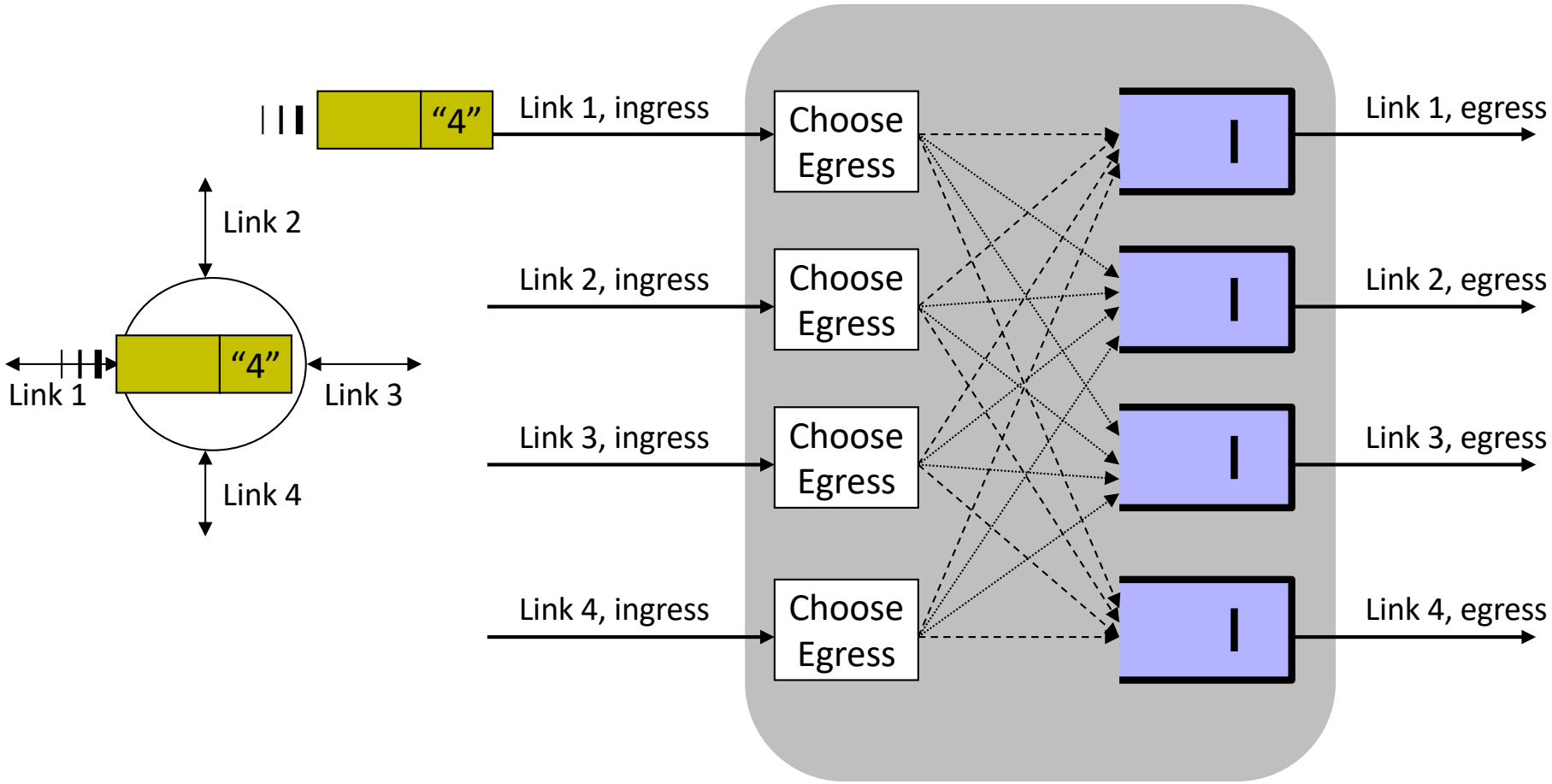


Control Plane

Data-path
per-packet processing

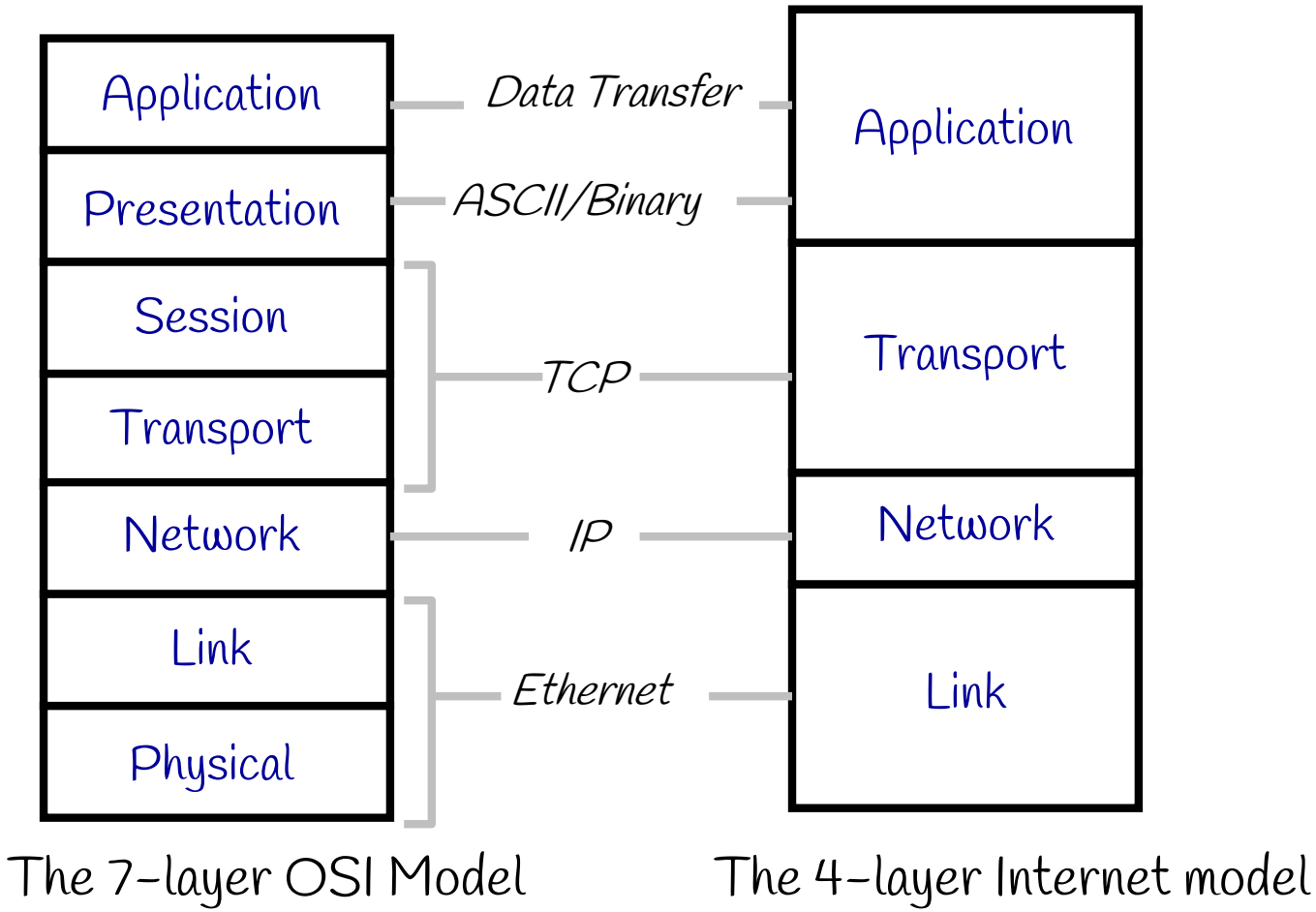
More advanced functionalities are added to this basic design. We will talk about those later.

Packet Switching – *Simple Router Model*

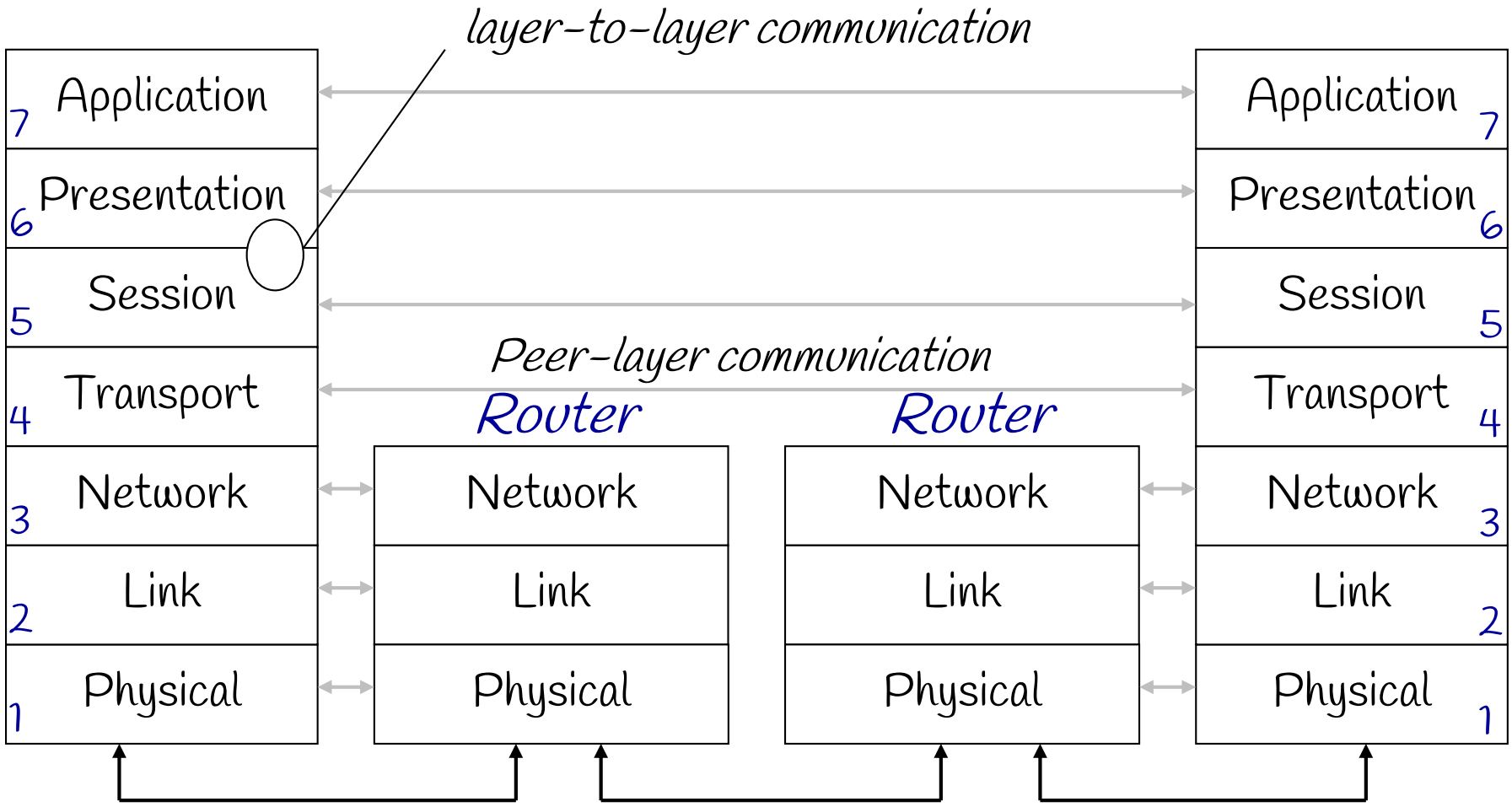


Routing vs. Switching

- Routing: network layer based on IP address
- Switching: link layer based on MAC address



Layering – *The OSI Model*



Introduction

- Background
  - What is a router?
  - Basic router architecture

- Basic packet processing in routers
  - IP address lookup
  - Packet buffering
  - Switching

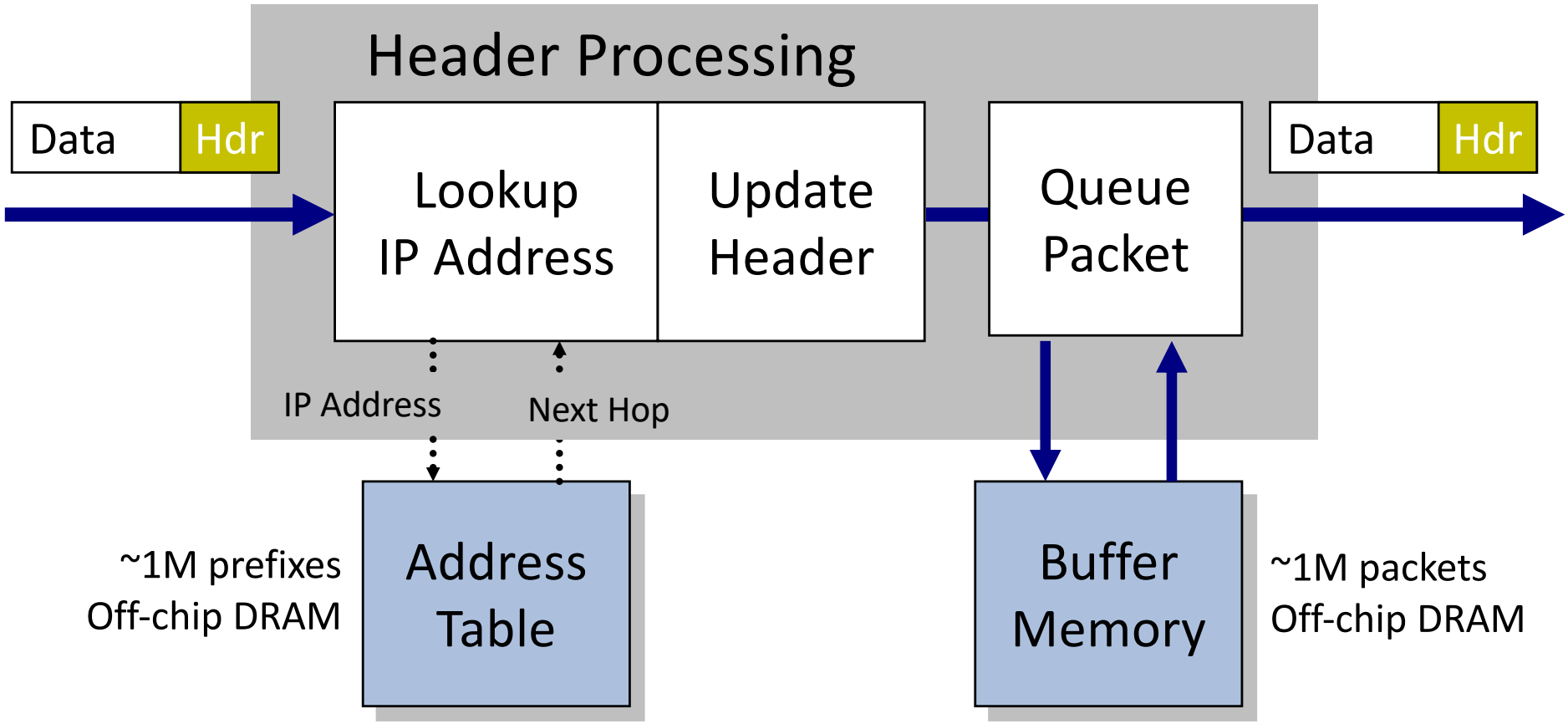
- Data Center Networking
  - Design Dimensions
  - Topology
  - Transport

- Machine Learning and Computer Networks

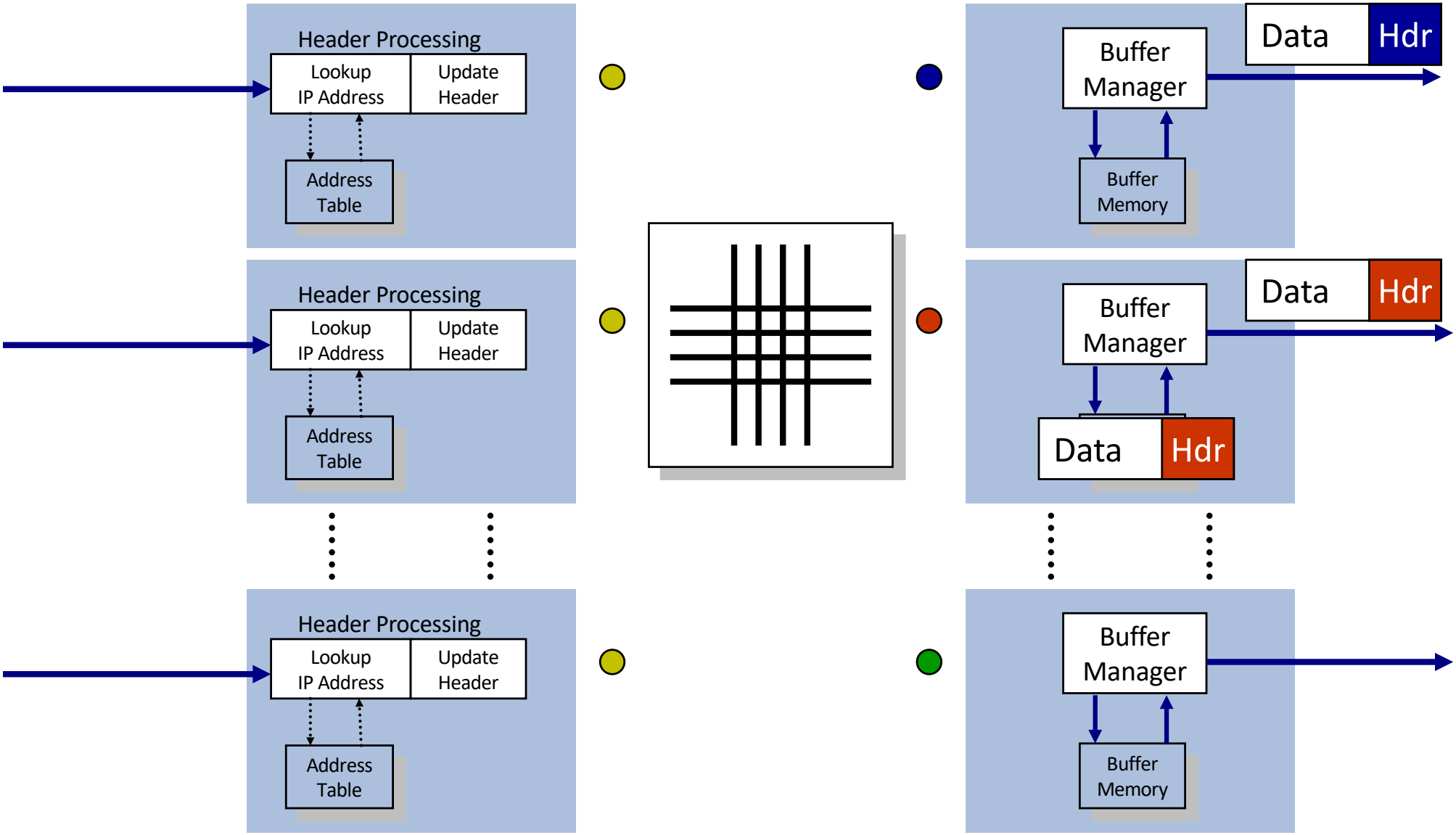
Per-packet Processing in an IP Router

1. Accept packet arriving on an incoming link.
2. Lookup packet destination address in the forwarding table, to identify outgoing port(s).
3. Manipulate packet header: e.g., decrement TTL, update header checksum.
4. Send packet to the outgoing port(s).
5. Buffer packet in the queue.
6. Transmit packet onto outgoing link.

Generic Router Architecture



Generic Router Architecture



Why are Fast Routers Difficult to Make?

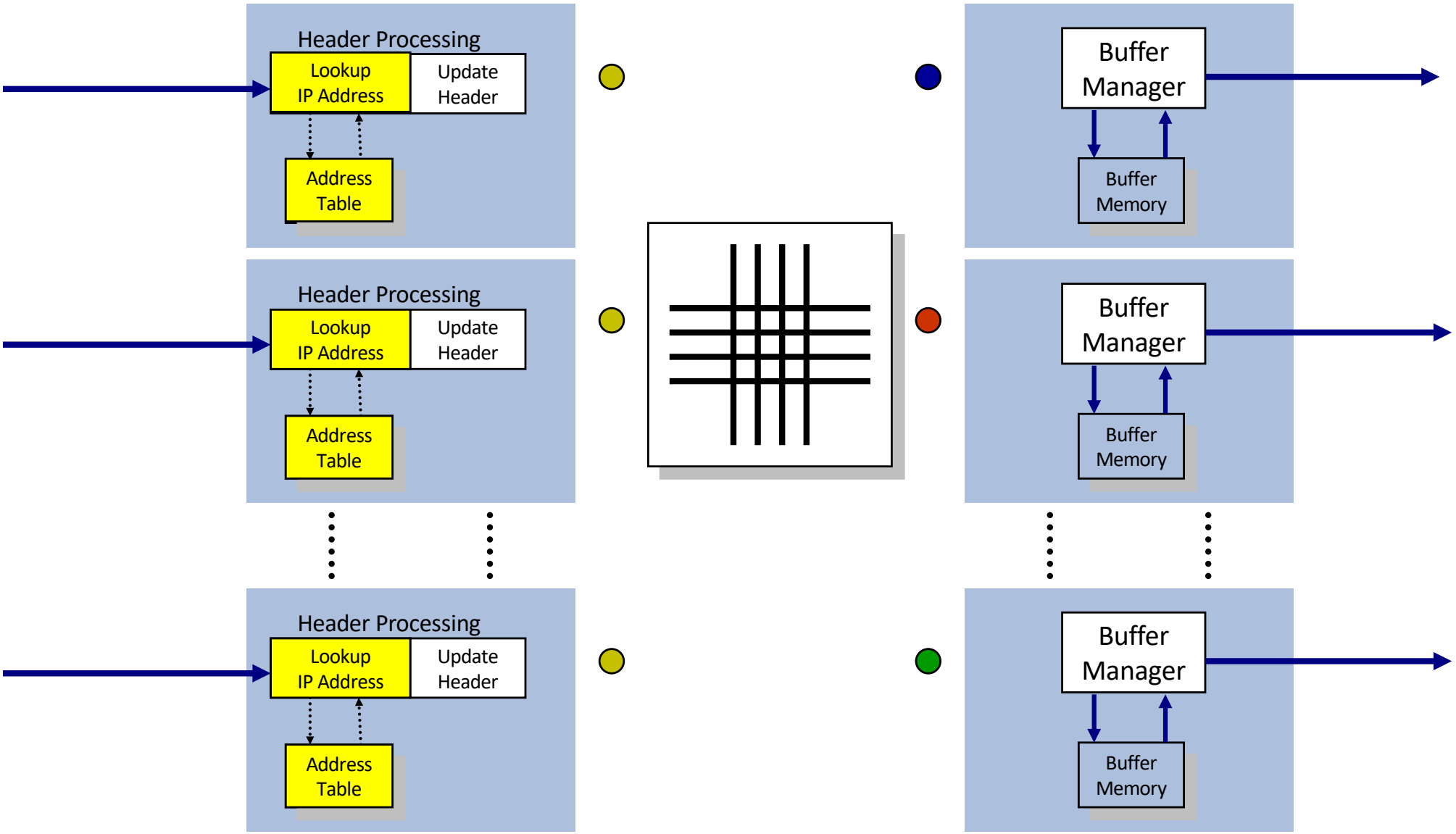
- It's hard to keep up with Moore's Law:
  - The bottleneck is memory speed.
  - Memory speed is not keeping up with Moore's Law.
- Moore's Law is too slow:
  - Routers need to improve faster than Moore's Law.

- The growth in this gap is due to three major challenges …
  - Address lookup
  - Packet buffering
  - Switching

Introduction

- Background
  - What is a router?
  - Basic router architecture
- Basic packet processing in routers
  - IP address lookup
  - Packet buffering
  - Switching
- Data Center Networking
  - Design Dimensions
  - Topology
  - Transport
- Machine Learning and Computer Networks

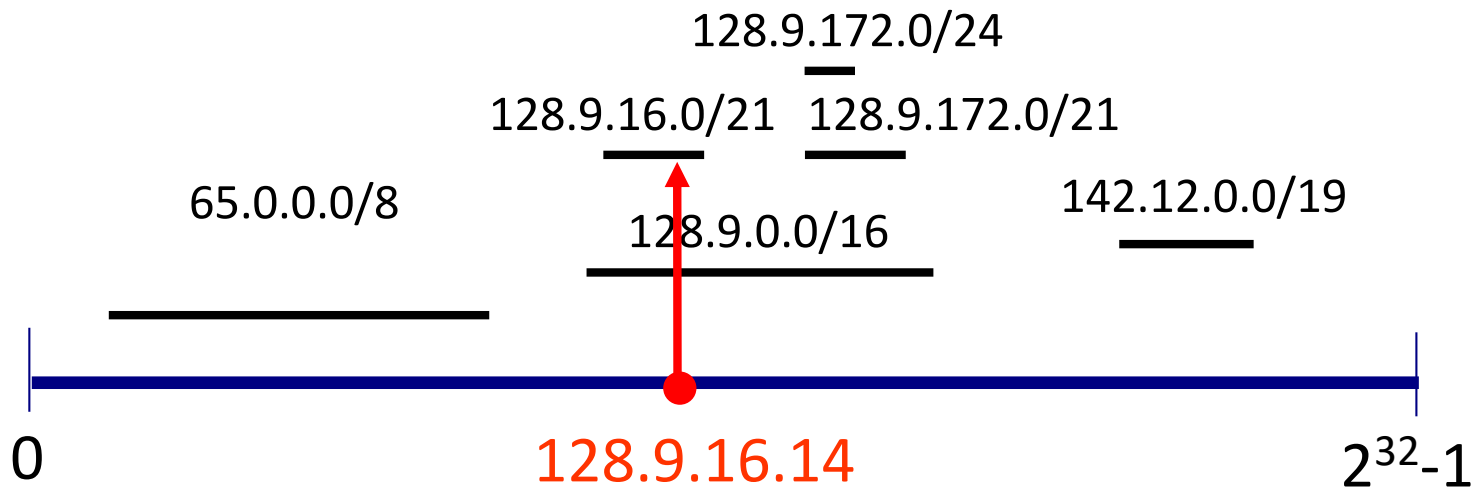
Generic Router Architecture



IP Address Lookup

- Why it's thought to be hard:
  1. It's not an exact match: it's a longest prefix match.
  2. The table is large: hundreds of thousands to millions of entries and growing
  3. The lookup must be fast: about 4-5ns for a 100Gb/s line.

IP Lookups find Longest Prefixes

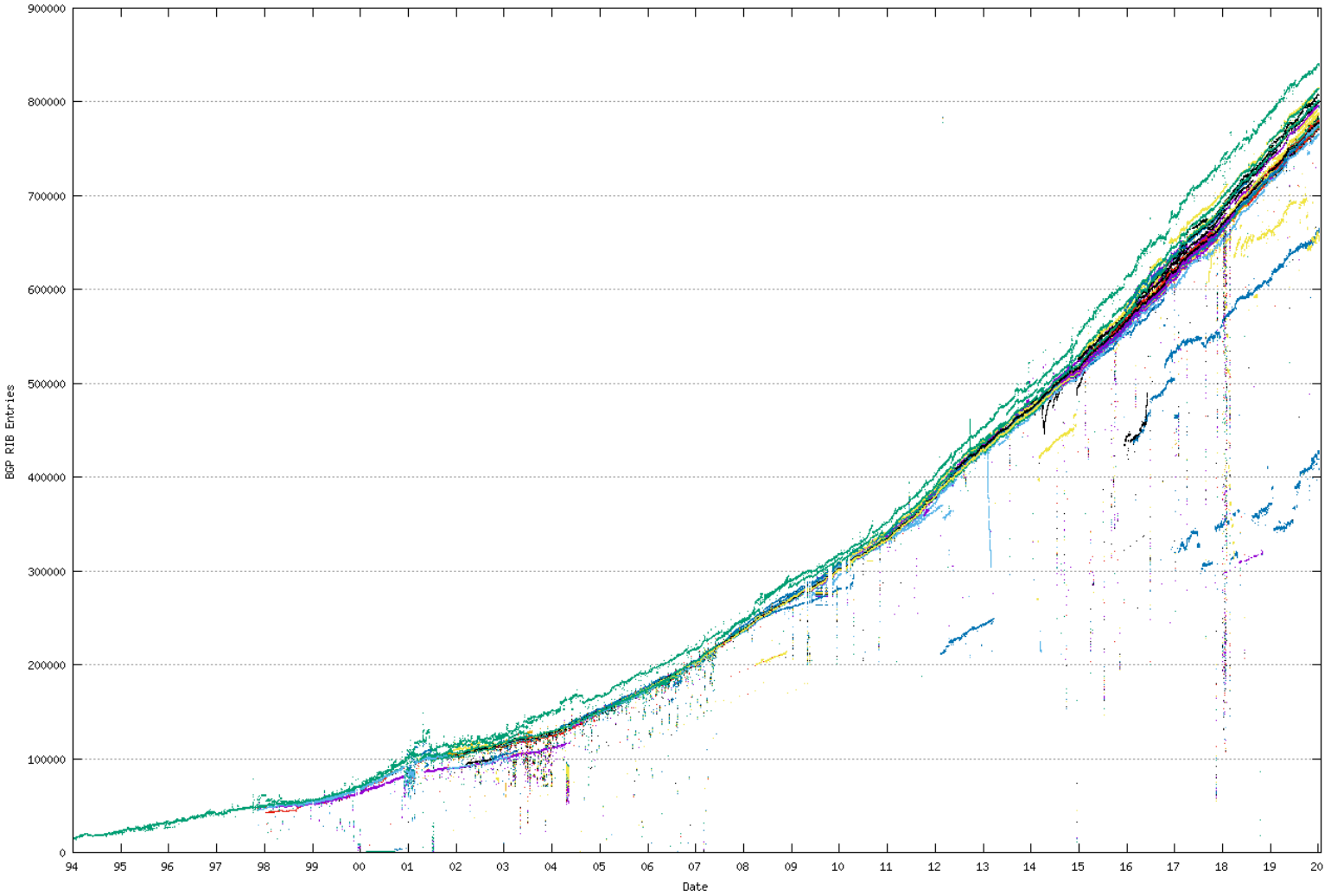


Routing lookup: Find the longest matching prefix (aka the most specific route) among all prefixes that match the destination address.

IP Address Lookup

- Why it's thought to be hard:
  1. It's not an exact match: it's a longest prefix match.
  2. The table is large: hundreds of thousands to millions of entries and growing.
  3. The lookup must be fast: about 4-5ns for a 100Gb/s line.

Address Tables are Large



Source: http://www.cidr-report.org/

IP Address Lookup

- Why it's thought to be hard:
  1. It's not an exact match: it's a longest prefix match.
  2. The table is large: hundreds of thousands to millions of entries and growing
  3. The lookup must be fast: about 4-5ns for a 100Gb/s line.

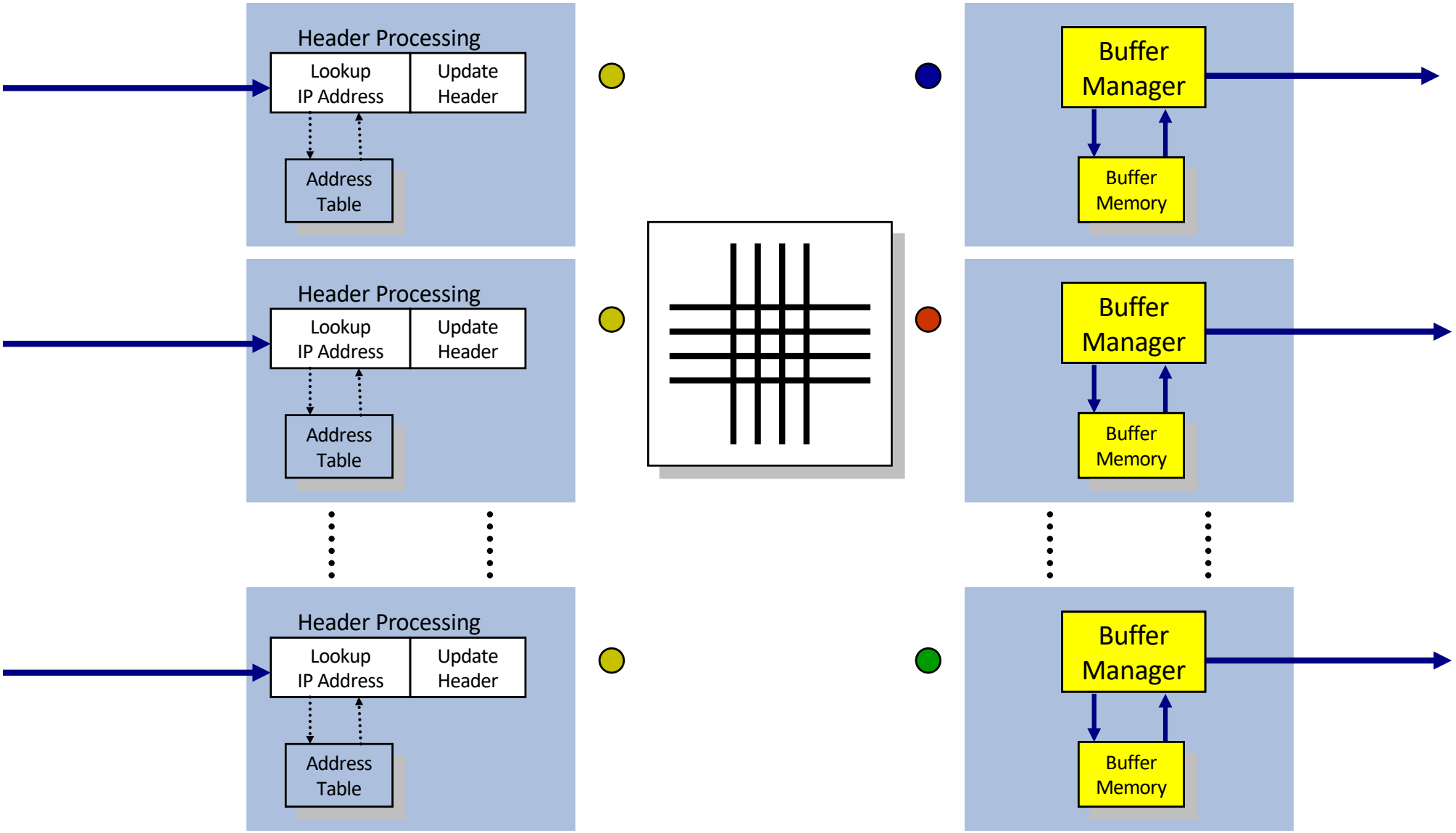
Lookups Must be Fast

- Line rates have been constantly growing over time.
  - 100Gb/s to 500Gb/s
- Packet sizes have not changed
  - Max: 1500 bytes (9000 jumbo packets)
  - Average: around 250 bytes
  - Min: 40 bytes
- Time to process each packet around a few nano-seconds

Introduction

- Background
  - What is a router?
  - Basic router architecture
- Basic packet processing in routers
  - IP address lookup
  - Packet buffering
  - Switching
- Data Center Networking
  - Design Dimensions
  - Topology
  - Transport
- Machine Learning and Computer Networks

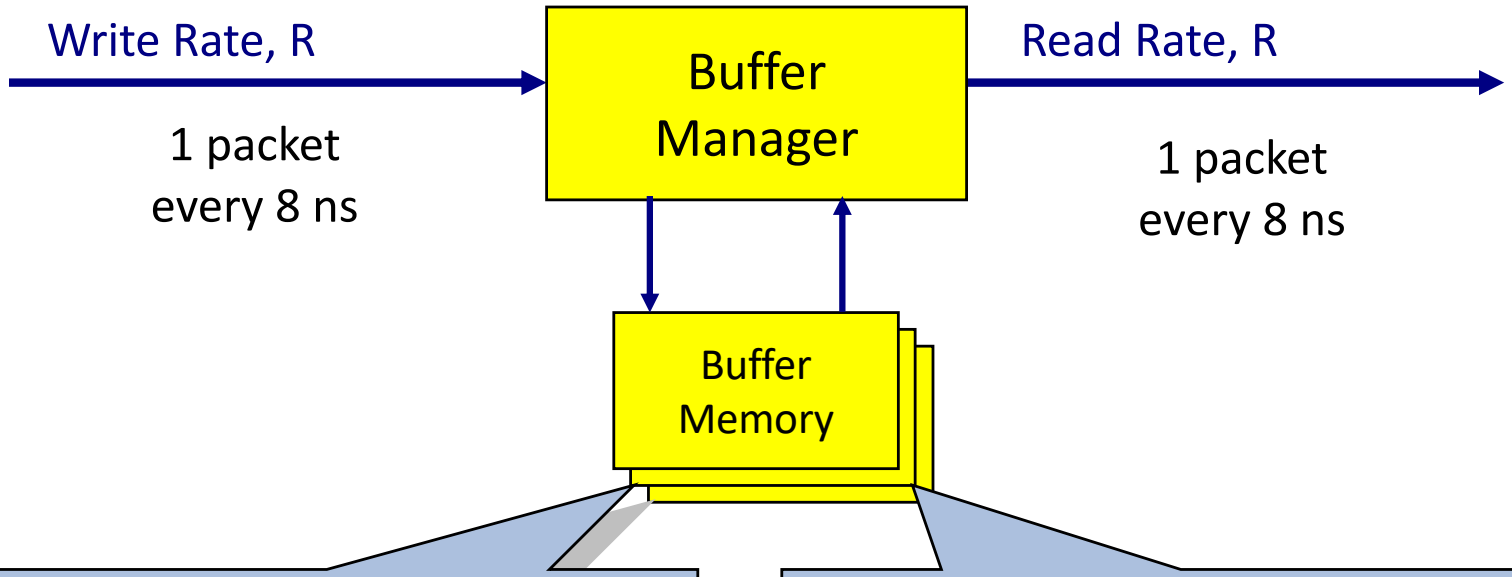
Generic Router Architecture



Fast Packet Buffers

Example: 40Gb/s packet buffer

Size = RTT*BW = 10Gb; 40 byte packets



Use SRAM?

+ fast enough random access time, but

- too low density to store 10Gb of data.

Use DRAM?

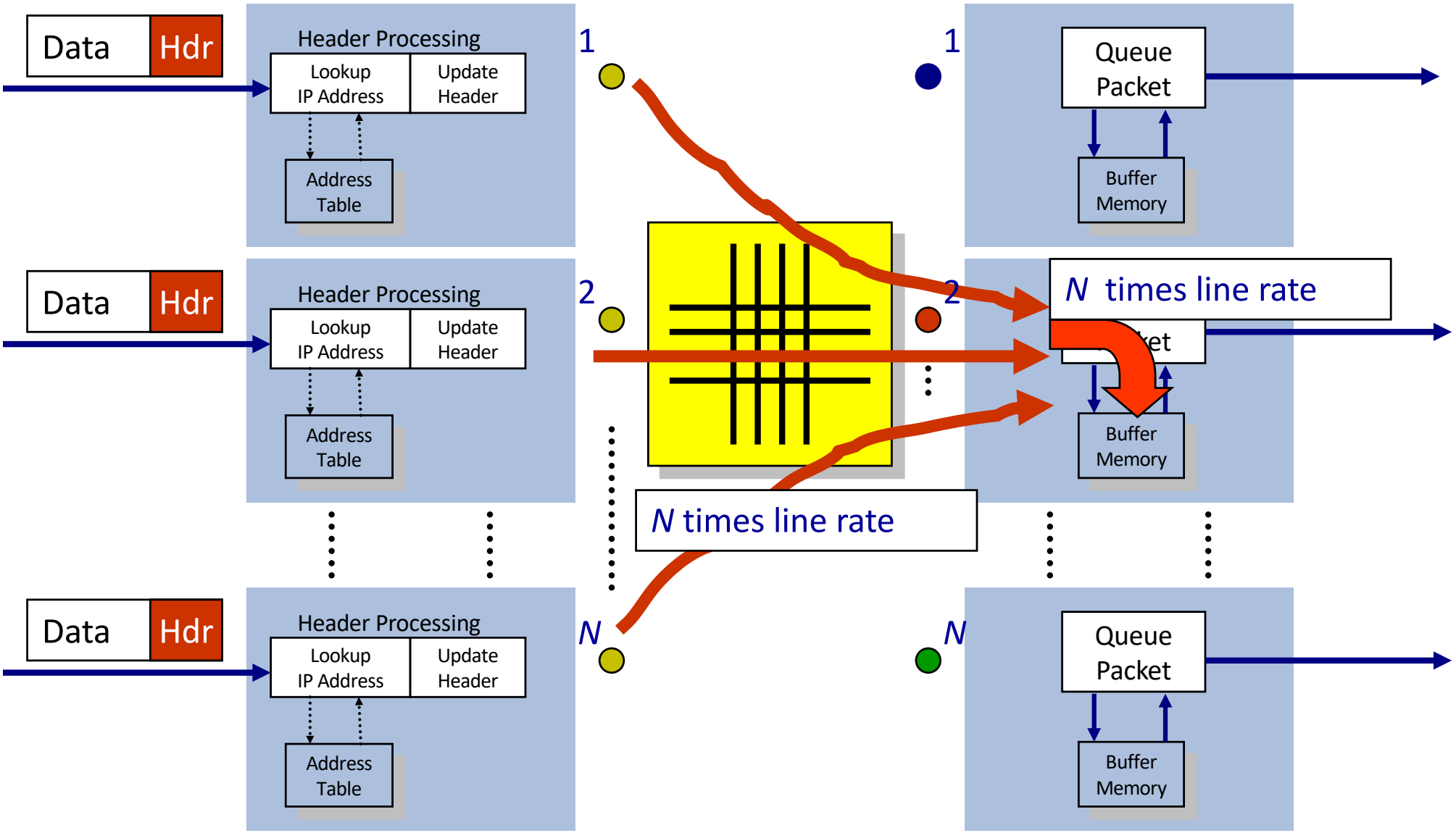
+ high density means we can store data, but

- too slow (10s of nano sec random access time).

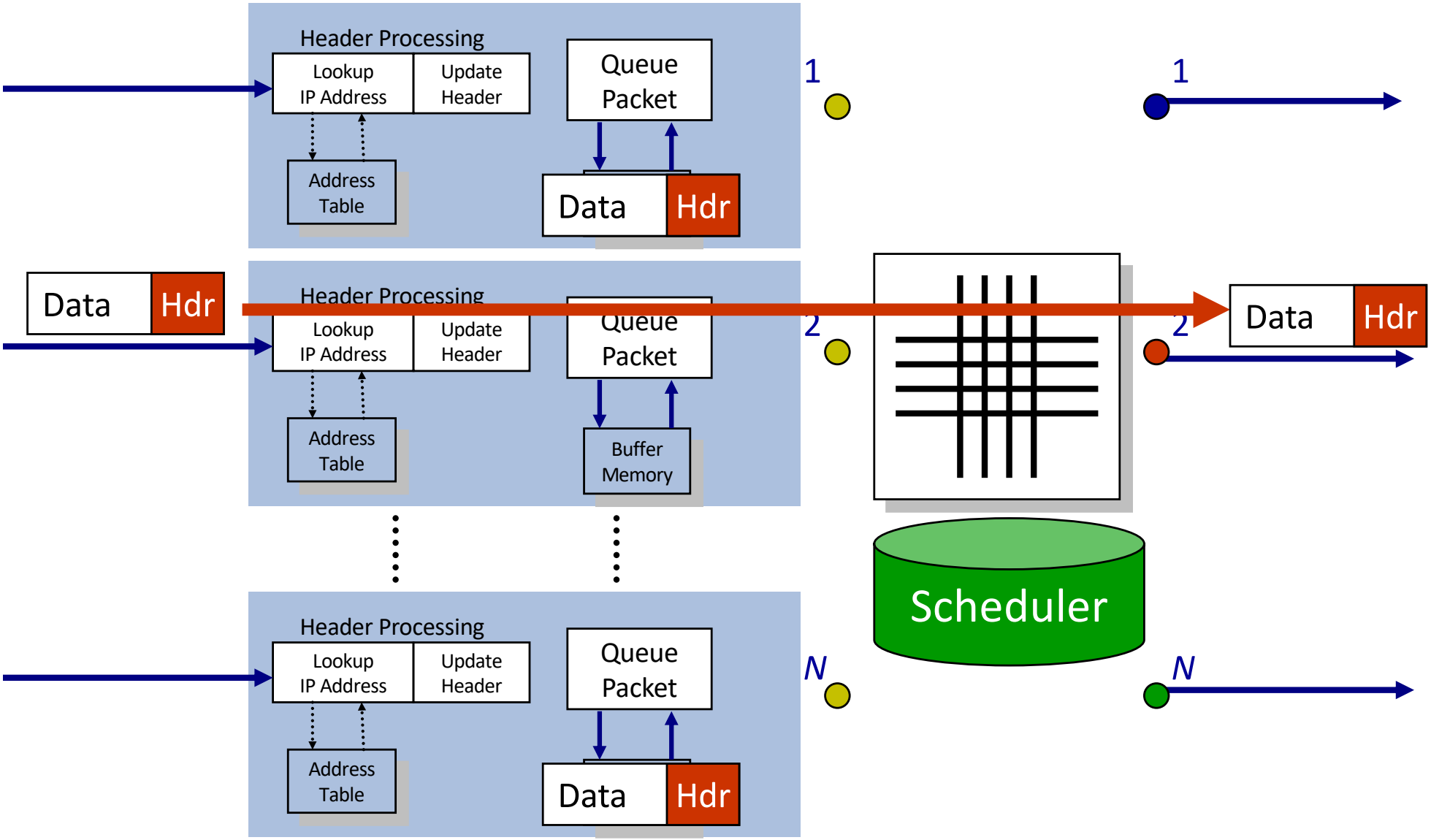
Introduction

- Background
  - What is a router?
  - Basic router architecture
- Basic packet processing in routers
  - IP address lookup
  - Packet buffering
  - Switching
- Data Center Networking
  - Design Dimensions
  - Topology
  - Transport
- Machine Learning and Computer Networks

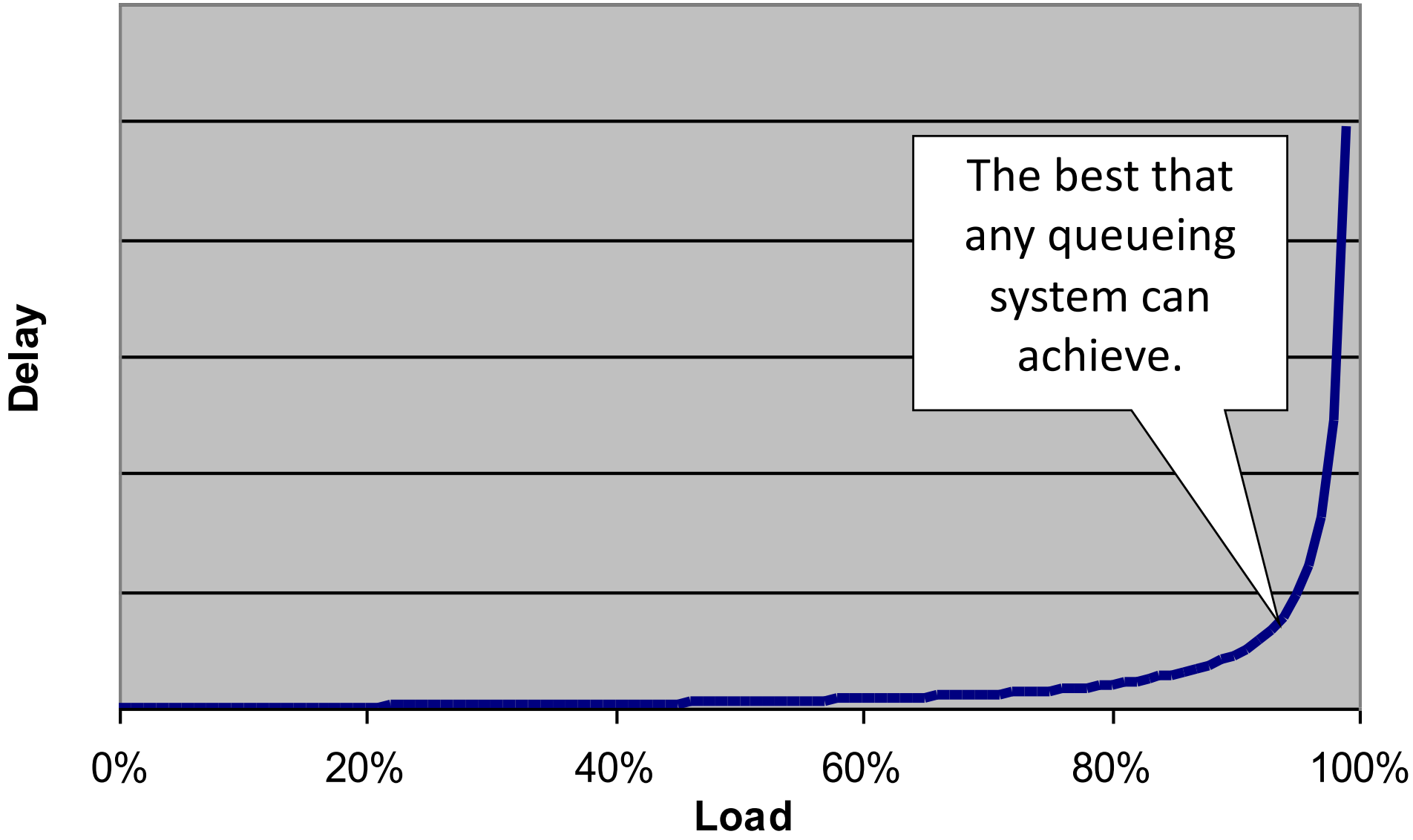
Generic Router Architecture



Generic Router Architecture

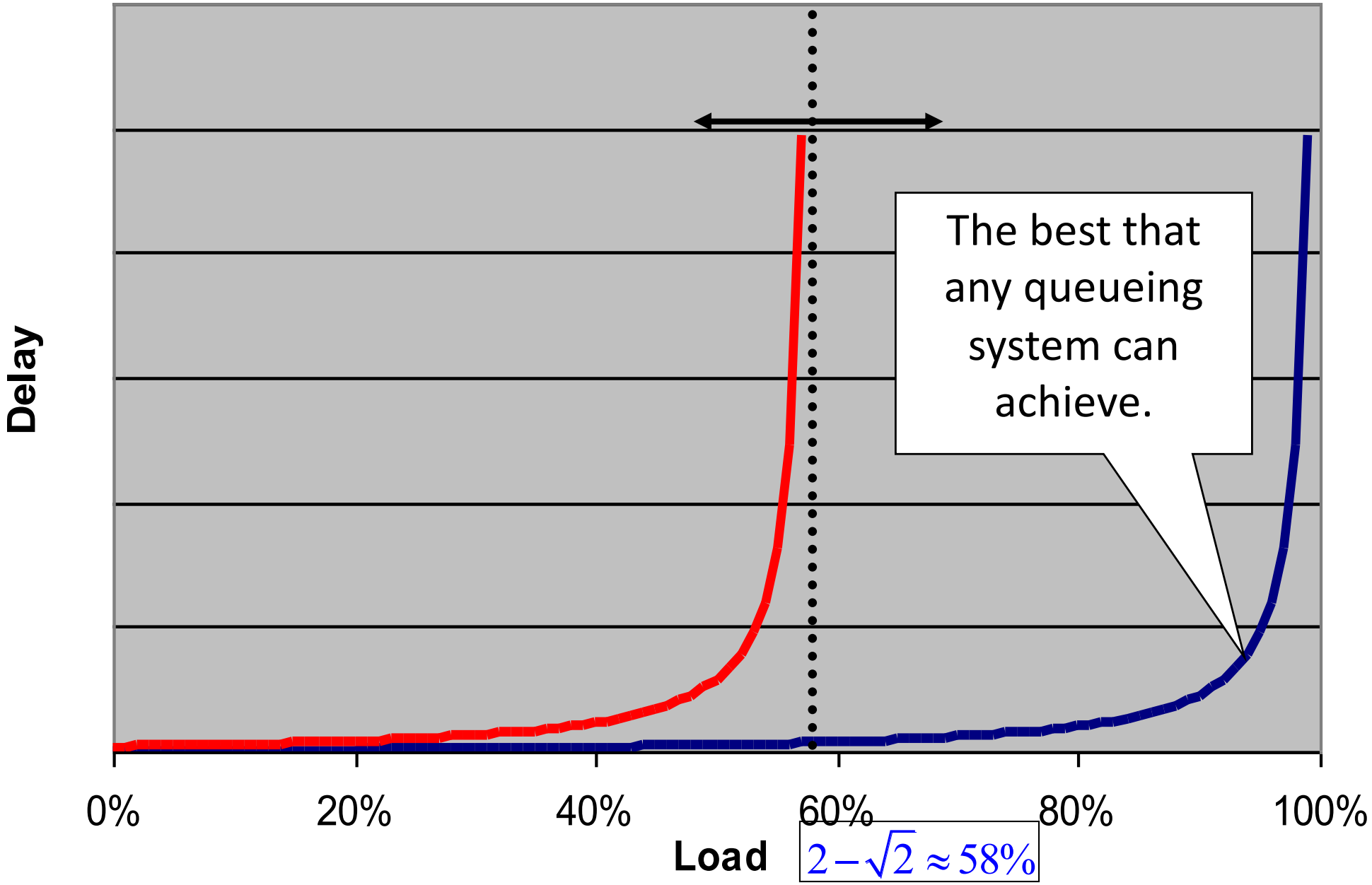


A Router with Output Queues

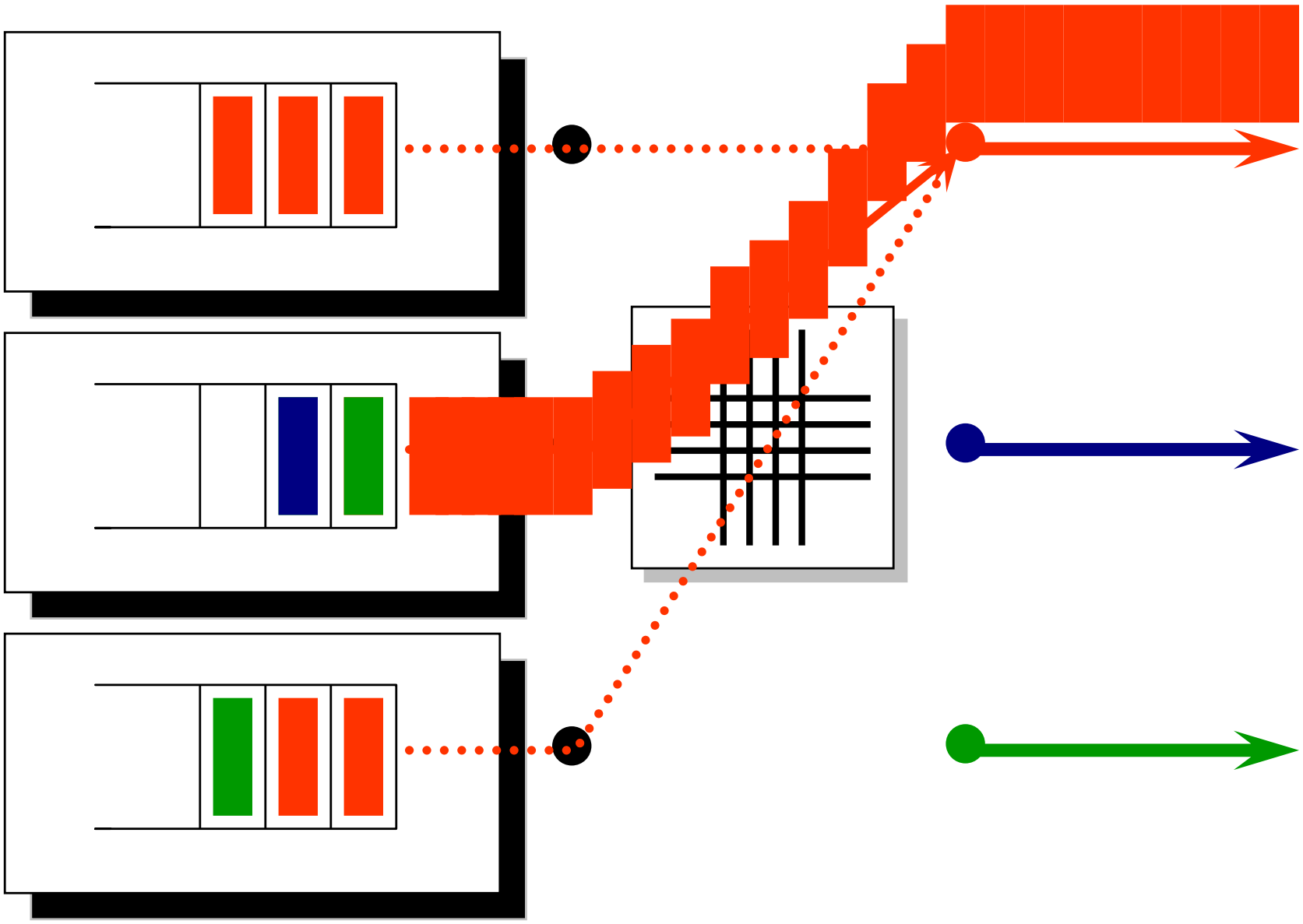


A Router with Input Queues

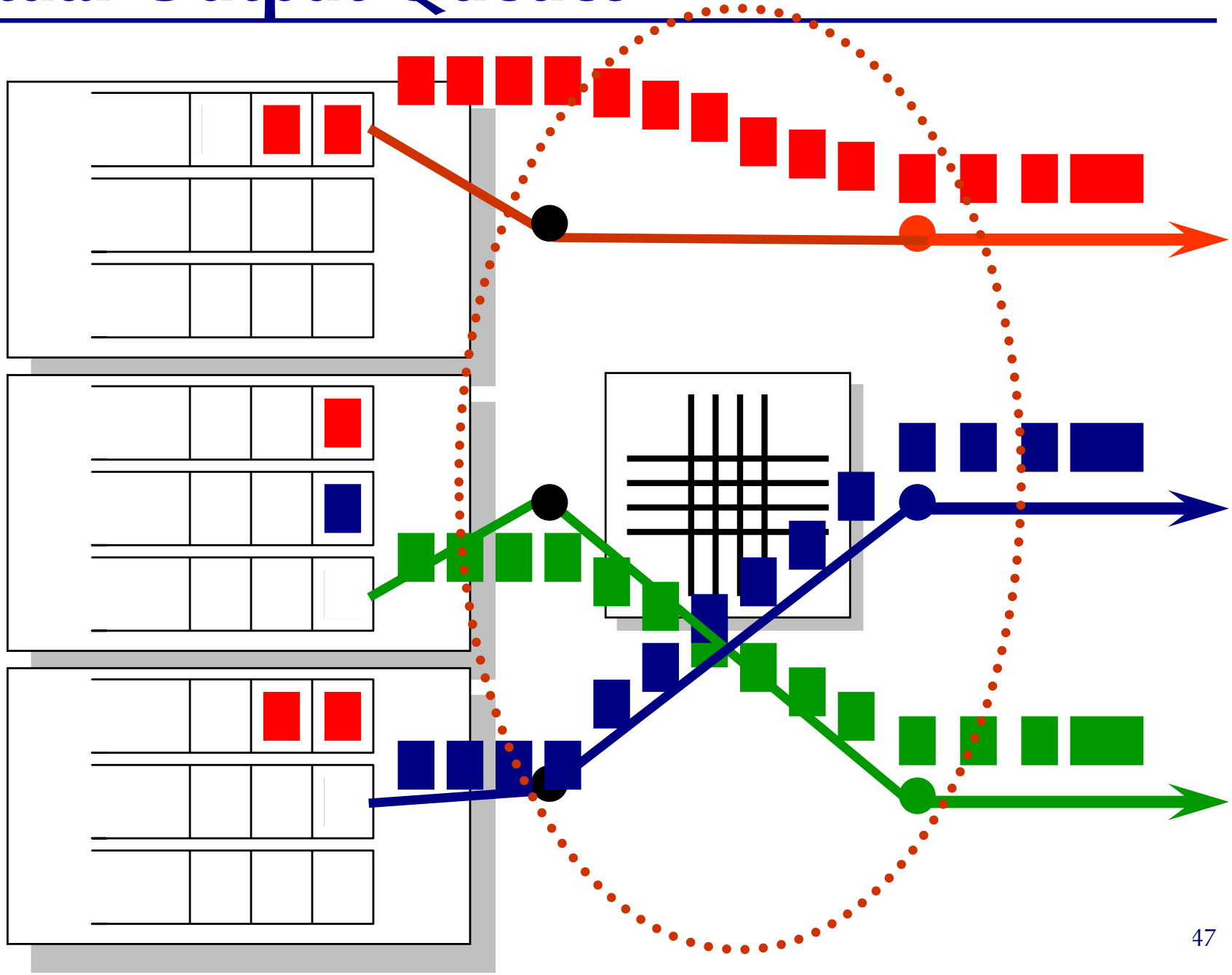
Head of Line Blocking



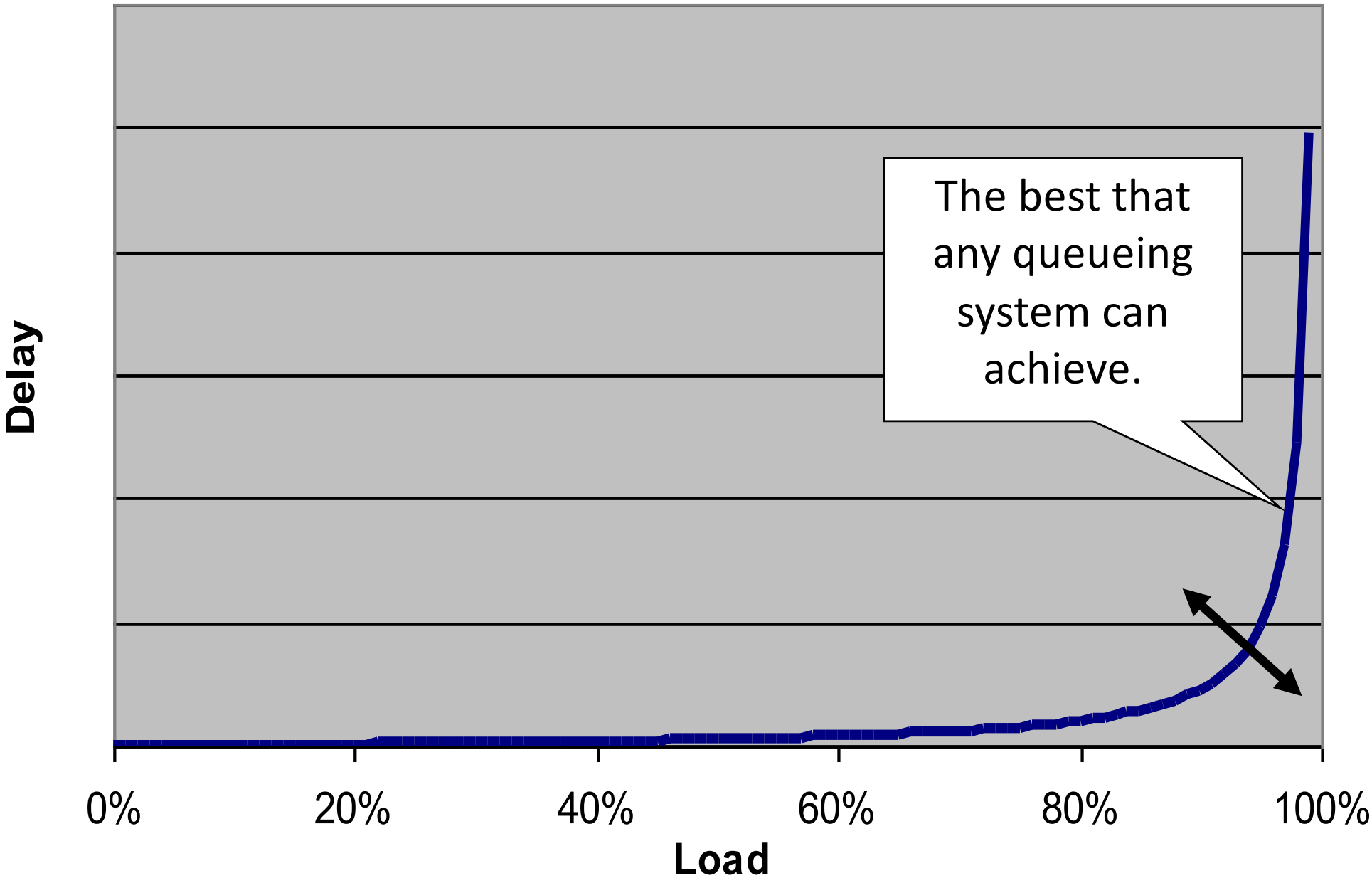
Head of Line Blocking



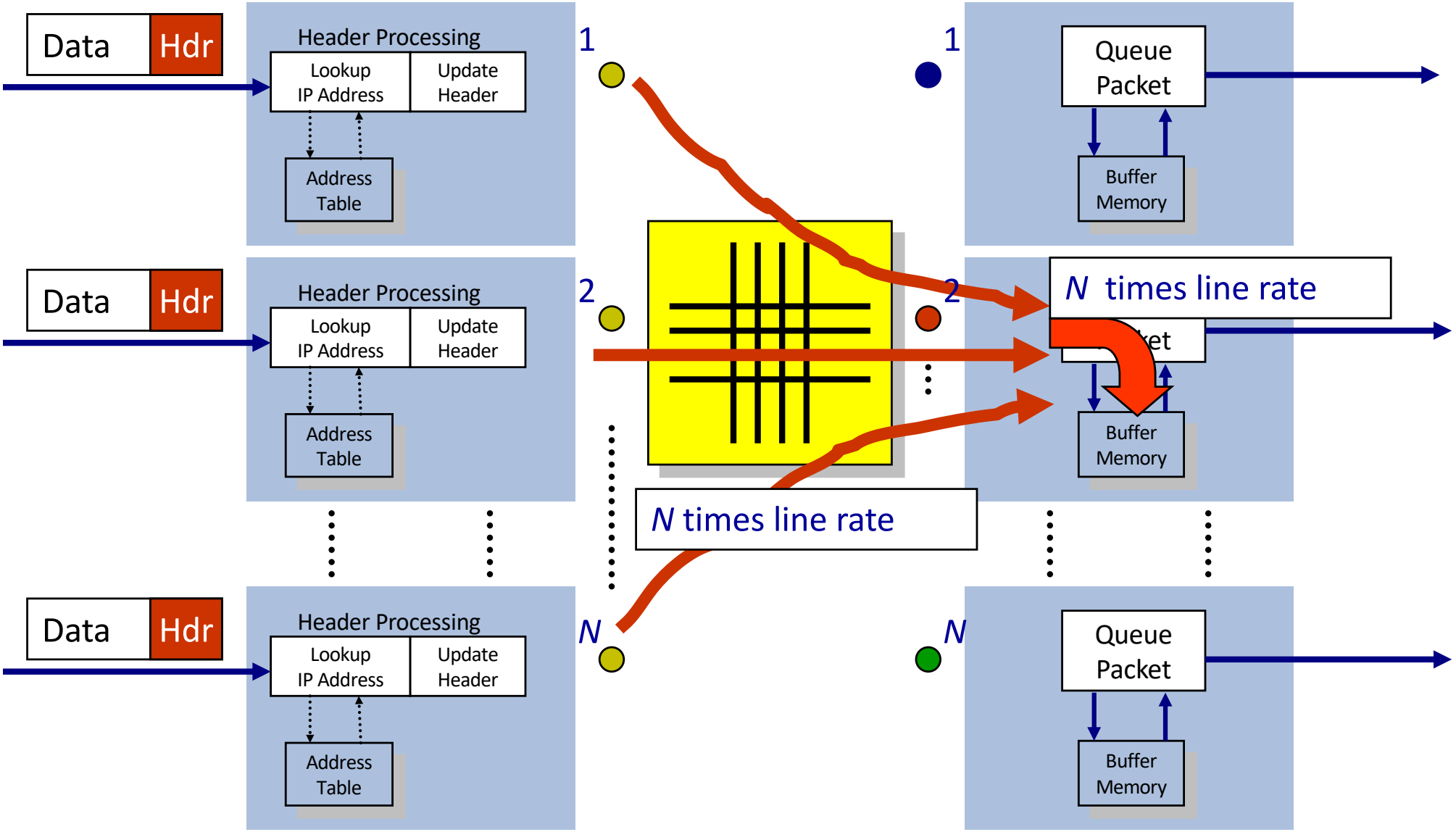
Virtual Output Queues



A Router with Virtual Output Queues

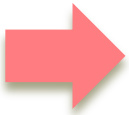


Generic Router Architecture



Introduction

- Background
  - What is a router?
  - Basic router architecture



- Basic packet processing in routers
  - IP address lookup
  - Packet buffering
  - Switching

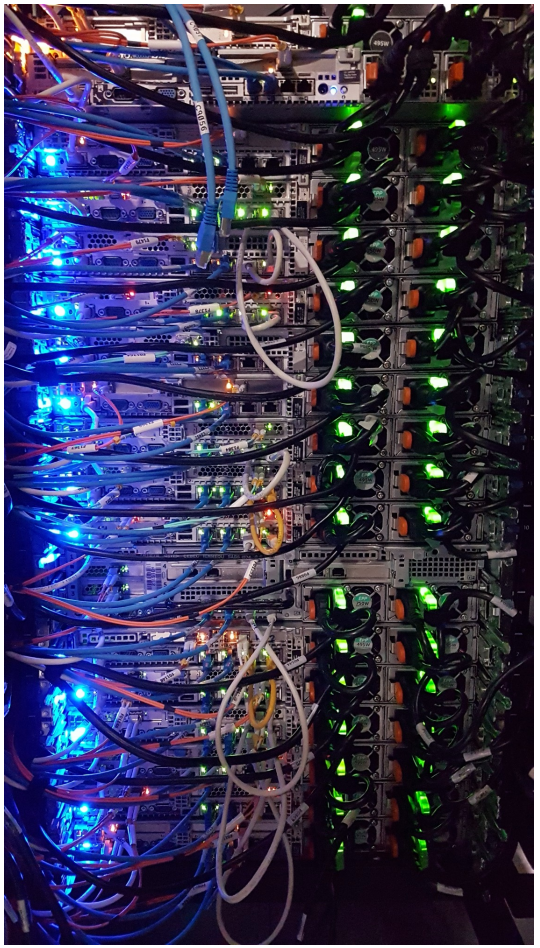
- Data Center Networking
  - Design Dimensions
  - Topology
  - Transport

- Machine Learning and Computer Networks

Data Center Network (DCN)

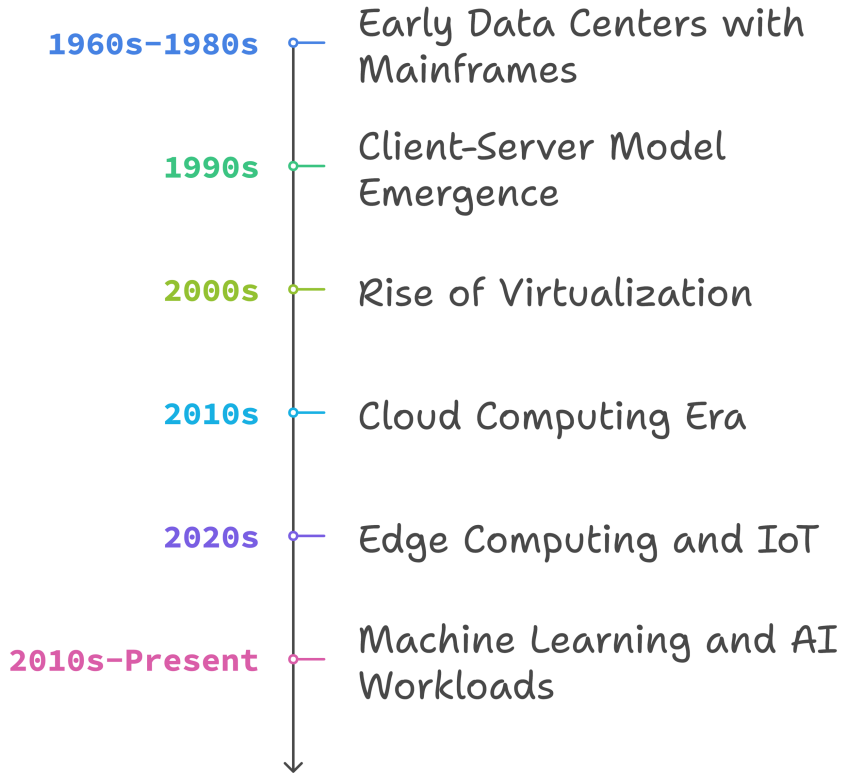
- A network of computing and storage resources
  - Proximity of components (within the data center) facilitates communication, i.e., high-performance
  - Can lead to reduced cost and overheads
    - Major cost upfront but less cost in long run.

- Functions
  - Data Storage and Management:
    - Security, efficiency, reliability
  - Application Hosting
    - End-users and business applications, cloud computing and SaaS models
  - Data Processing:
    - Large volumes of data for analytics and processing, big data and AI workloads
  - And more …

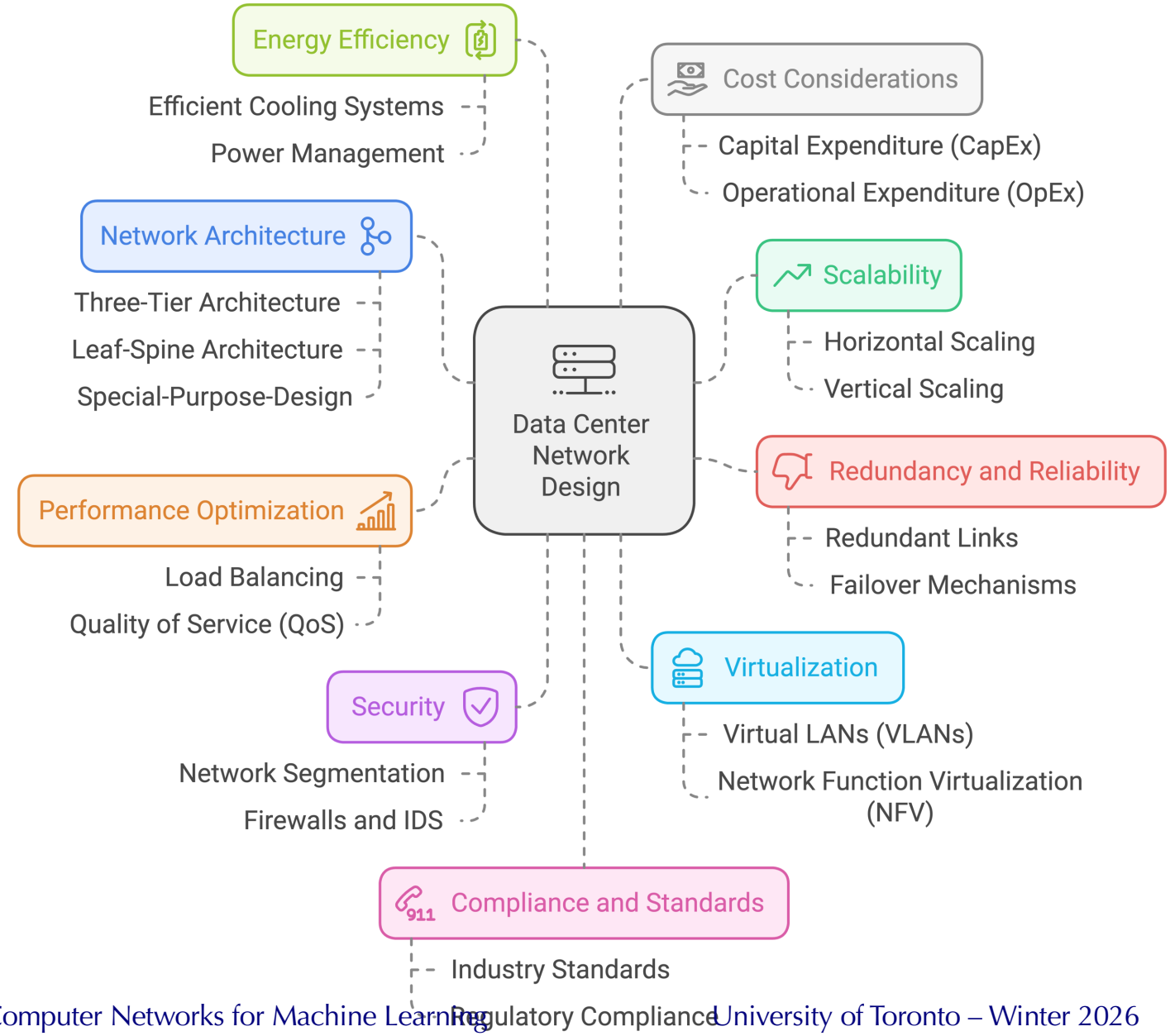


Evolution of DCN

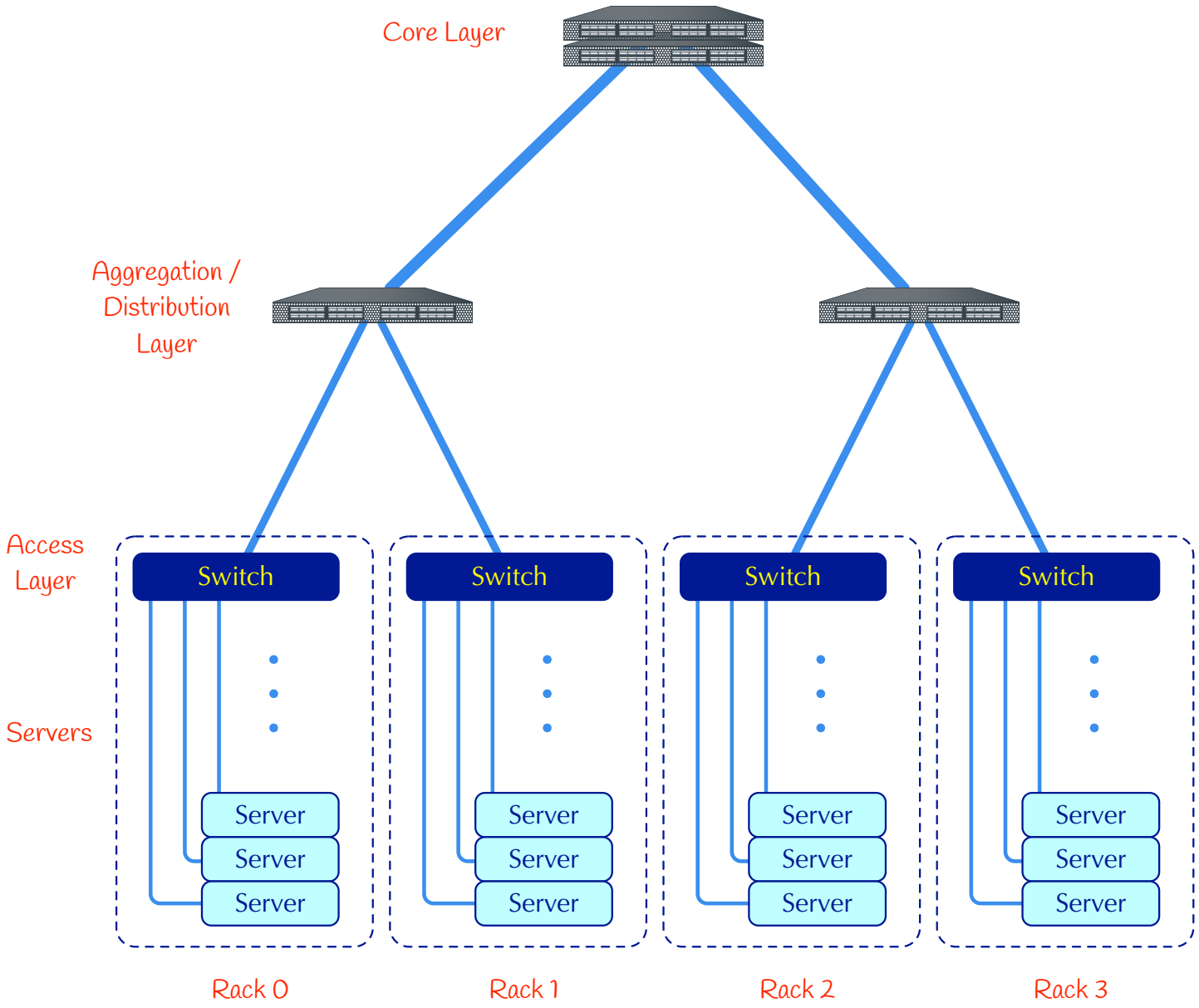
- Early Data Centers (1960s-1980s)
  - Mainframes
  - Centralized computing: limited networking
  - Point-to-point and proprietary connections
- Client-Server Model (1990s)
  - Distributed computing
  - Ethernet, TCP/IP
- Rise of Virtualization (2000s)
  - Virtual machines
  - Efficiency and scalability
  - VLANs, network segmentation
- Cloud Computing Era (2010s)
  - Cloud services
  - SDN: Scalability and automation
- Edge Computing and IoT (2020s)
  - Demand for low-latency, distributed network architectures
  - Micro data centers closer to data sources
- Machine Learning and AI (2010s-Present)
  - Exascale high-performance computing
  - Network as the bottleneck: stringent performance requirements



DCN Design Dimensions



Three-Tier Architecture



Hierarchical tree network topology

- Commonly used in traditional DCNs

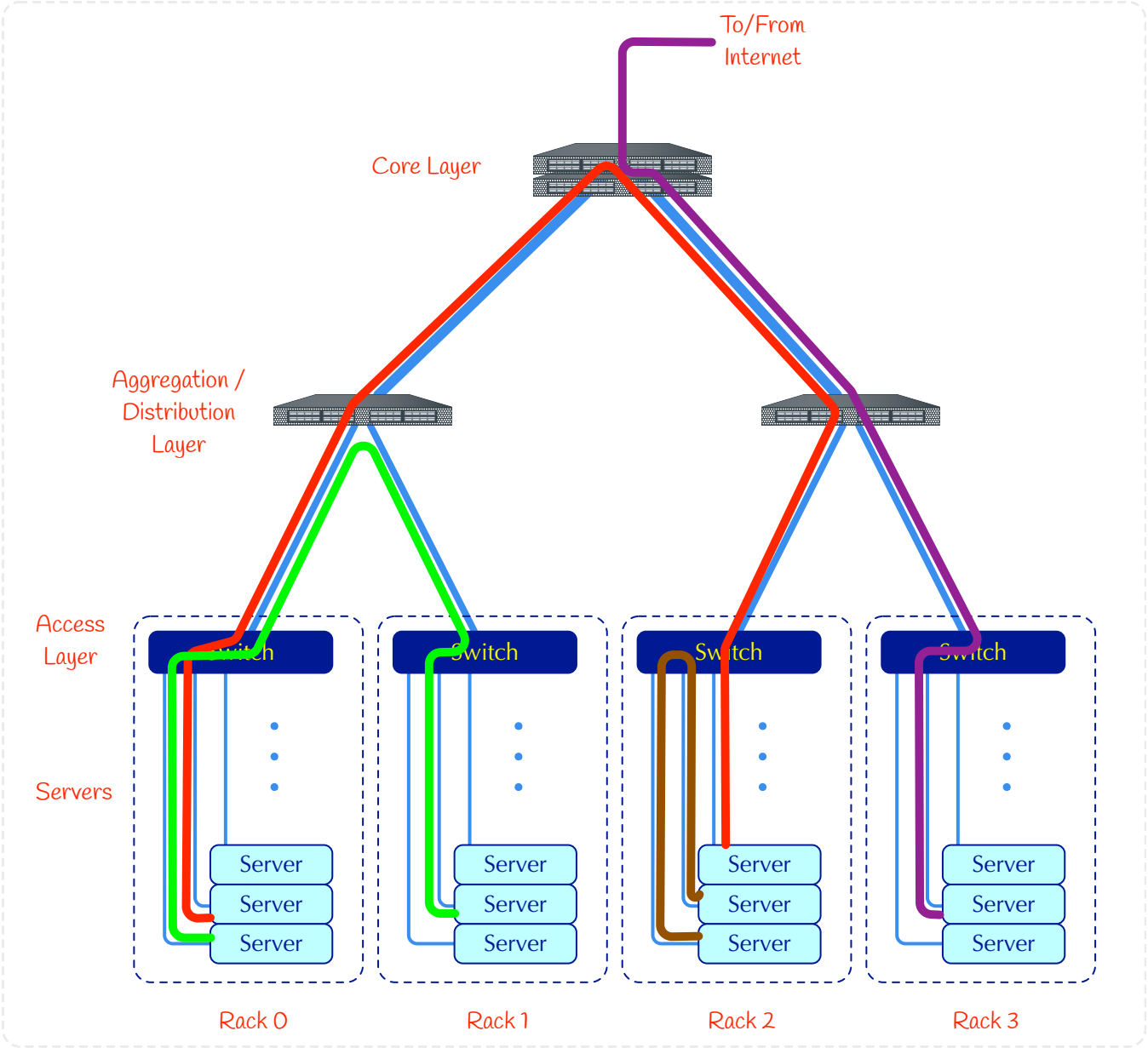
Three layers:

- Core:
  - Layer 3 (network layer)
  - Fully connected high-speed mesh of multiple routers
  - Connect to the external networks
- Aggregation:
  - Layer 3 and 2 (network and link layers)
  - Connect to core with few high-speed links (e.g., 100 Gb/s links), to access with many low-speed links (e.g., 10Gb/s) → simplify cabling
  - Middleboxes sit here (firewall, load balancer, …)
- Access: layer 2 (link)
  - Connect to each server in the rack (e.g., through one or two 10Gb/s links)
  - VLANs used to limit broadcast

Modular design

- Easy to expand

Traffic Direction in 3-Tier Architecture



Within a rack

Between racks

Between racks

To/from Internet

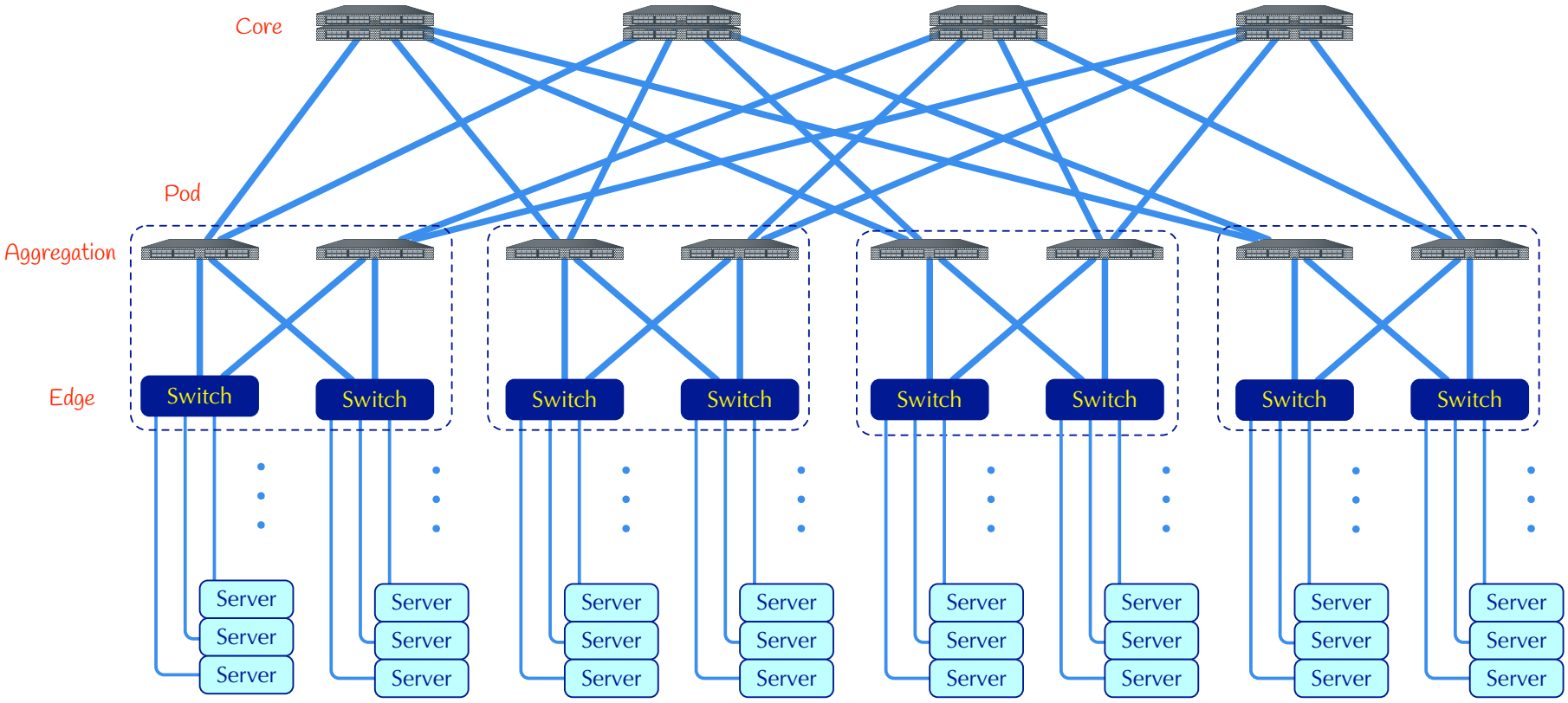
Switching and Routing:

- Communication within a rack or within the same aggregation switch happens at link layer (switching).
- Traffic going through core (between aggregation switches and to/from external networks/Internet) happens at network layer (routing).
- Different flows might have different RTTs (even within DCN)

Total Link Capacities at Each Layer Might Be:

- Equal to lower layers, or
- Less (over-subscription)
- Reason:
  - Cost-saving: less bandwidth → lower cost
  - Locality: most communication within the same rack, or within the same cluster.

Fat-Tree Architecture



Properties of Fat-Tree Architecture:

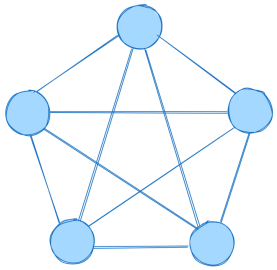
- Scalable: easily expandable
- Low latency, high throughput
- Cost-effective
  - Commodity hardware
  - Lower operational costs

Reliability and Improved Performance:

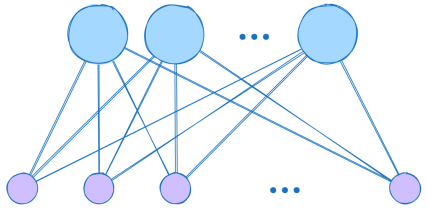
- Edge and aggregation switches are grouped into "pods".
  - Multiple-paths (choice of aggregation and core)
  - Automatic failover
- Leads to better load balance, redundancy, and thus
  - Reliability and high availability, and
  - Improved performance

Other DCN Architectures/Topologies

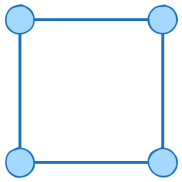
- Mesh: every node is connected to every other node
  - Direct communication, costly, but high performance
- Leaf-Spine topology: two-tier structure, servers and storage node connect directly to leaf switches
  - Switched environment, VLANs to limit broadcasts
- Hyper-cube: multi-dimensional cube structure
  - Used in high-performance computing and ML solutions
- Hybrid: combine two or more topologies
  - Tailored to specific requirements of DCNs
- And many more …
- Question: what are the properties of each of these topologies?
  - End-to-end latency?
  - Simplicity?
  - Scalability?
  - …



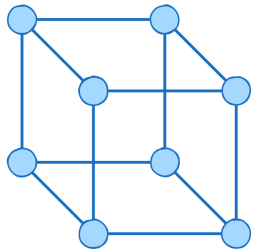
Mesh Topolog



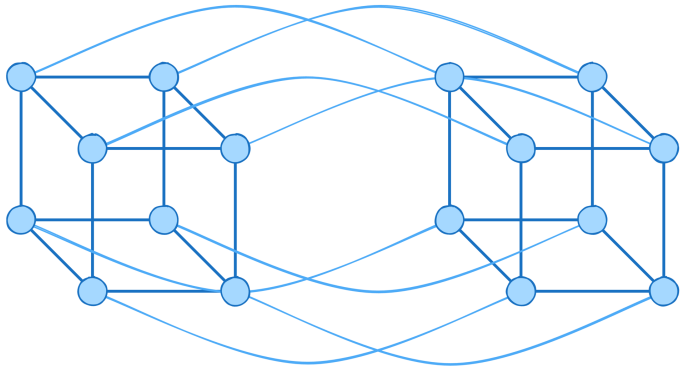
Leaf-Spine Topology



Two Dimensional Hypercube



Three Dimensional Hypercube



Four Dimensional Hypercube

Transport in Data Center Networks

- Data center network properties:
  - Extremely short RTTs
  - Extremely high bandwidth links
  - Extremely large transfer
  - Single authority (typically)

- Leads to new challenges and opportunities
  - Can you think of any challenges for providing good transport solutions?
  - How about opportunities?

ECN + DCTCP

- Easier to ensure ECN is enabled on all devices in DCN
  - Single authority

- DCTCP
  - A variant of TCP
  - Uses ECN as congestion signal Use → reduced packet loss

- How does DCTCP work?
  - Congestion measured based on fraction of packets marked with ECN (called α).
    - α is the moving average of the observed fractions (like estimatedRTT)
  - Adjust congestion window based on the extent of congestion: $cwnd \leftarrow cwnd \times (1-\alpha/2)$
    - Instead of halving the window in case of congestion.

- DCTCP Properties:
  - More responsive and less aggressive to network conditions compared to traditional TCP
  - Keeps queue lengths shorter (as reacts faster) → low latency

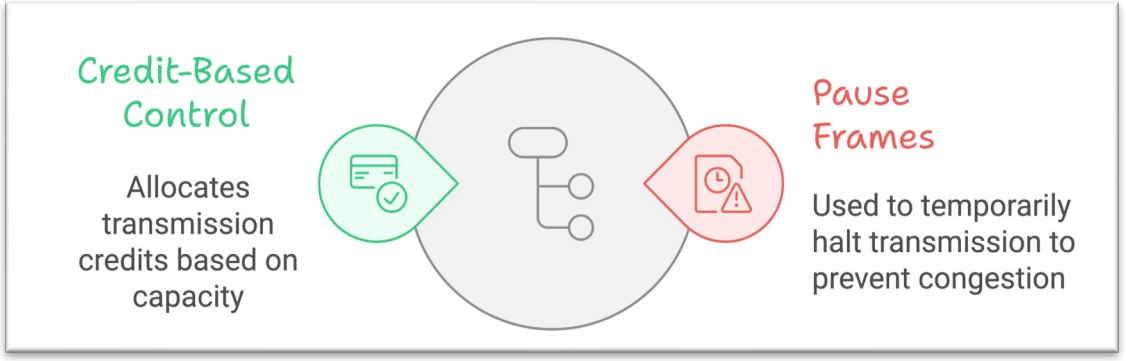
Link Layer Flow Control

Flow control mechanism used to create lossless networks.

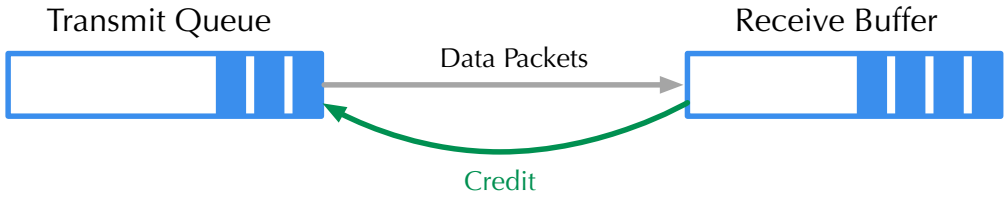
- Not to be confused with flow control in transport layer which is end-to-end.
- Setup: two nodes (end-hosts or switches) connected via a link.
  - Both have buffer (transmitter queue and receive buffer).
- Goal: ensure the receiver can handle the traffic injected on the link → no packet loss

Two prominent techniques:

- Credit-based
- Pause-based

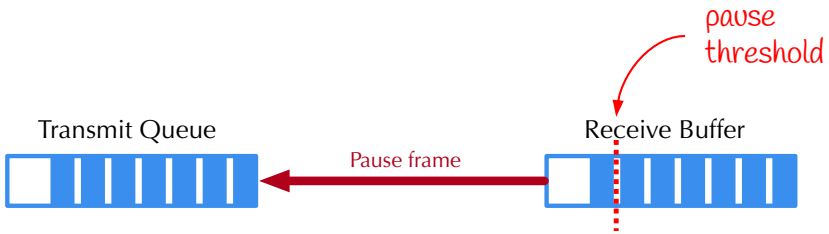


Credit-Based Link-Level Flow Control



- Receiver provides credits to the sender when it has room
  - Credit unit: bytes or packets
- Credit allocation:
  - At the beginning certain (fixed) credit is allocated.
    - Question: how much credit should be allocated initially?
  - Transmitter uses credit when transmitting.
    - Pauses if there is no more credit available.
  - Receiver replenishes transmitter credits as receiver buffer becomes available.
- Note: we also can have credit-based congestion control. This is not what we are covering here. The concepts are similar, but at different layers.

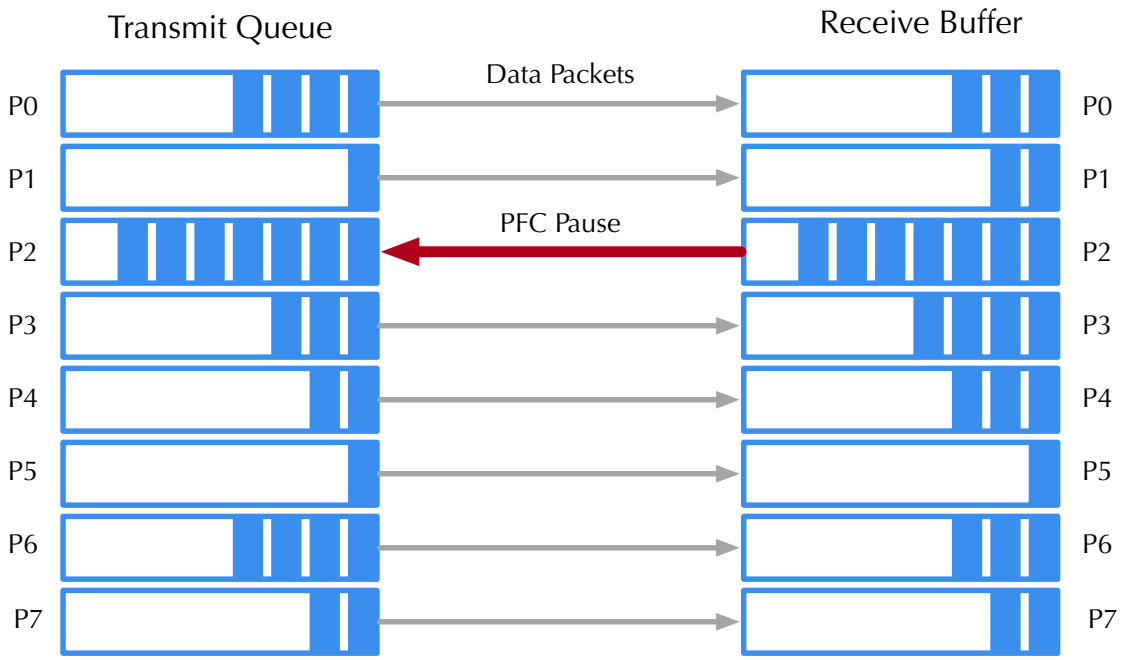
Pause-based Link-Level Flow Control



- Transmitter does not need permission to start.
- Receiver signals the transmitter when it is running out of buffer space.
  - When buffer occupancy goes above a fixed pause-threshold.
  - Pause-frame sent to transmitter
  - Transmitter halts transmission
- Once the receiver buffer has sufficient space …
  - Receiver sends a resume frame to the transmitter
  - The transmitter can resume sending.

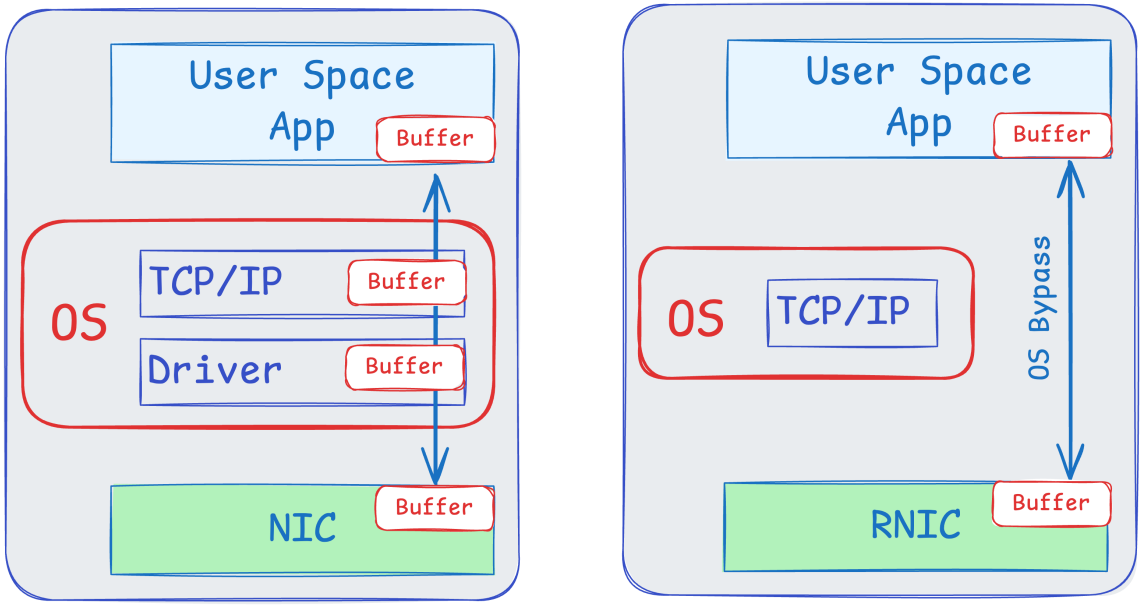
Priority-based Flow Control (PFC)

- Allows multiple priority-queues.
- Pause individual queues not all traffic
  - Allows other priority queues to continue transmitting even if a single queue is paused.
- Improves impact of pause on non-congested traffic to some extent
- Still, we might pause non-congested flows
  - Why?
  - Is there an easy way to solve this problem?
- Known issues: head-of-line blocking (deadlock), PFC storm



Remote Direct Memory Access

- Directly write to remote server's memory
  - Both sides register *memory regions* to give RDMA direct access permission and mapping
  - Need trust/cooperation between two ends
- RDMA-capable NIC (RNIC) handles data transfer entirely in hardware
  - No need to involve CPU for transfer
- Queue Pairs (QPs): a send queue and a receive queue.
  - Supports various operations like send, receive, read, and write.



Traditional Mode RDMA Mode

Benefits of RDMA

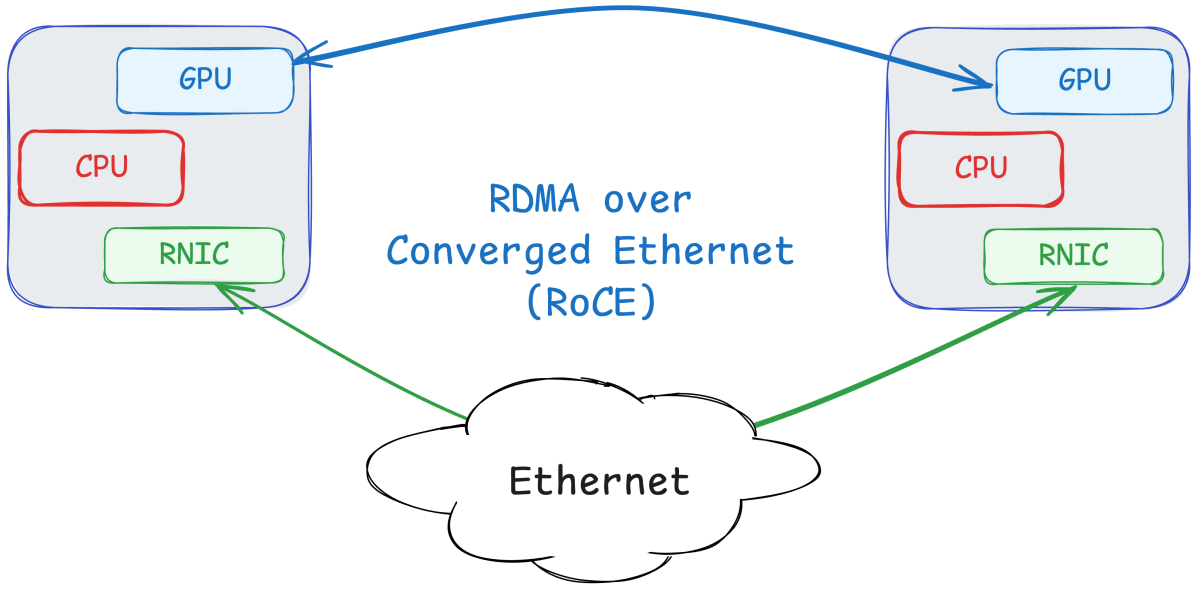
- Low Latency
  - Minimizes delays by avoiding CPU intervention.
  - Ideal for applications requiring real-time data processing.
- High Throughput
  - Enables faster data transfer rates.
  - Suitable for high-performance computing and large data sets.
- Reduced CPU Load
  - Frees up CPU resources for other tasks.
  - Improves overall system efficiency.
- Zero-copy data transfer.
  - Eliminate (or minimize data copies)
- Applications: High-Performance Computing (HPC), Storage, …

From Proprietary to Commodity

- Original technology InfiniBand
  - Touching physical layer, link layer, and transport layer in the stack.
  - Small number of vendors

- Later RDMA enabled over Ethernet
  - RoCE: RDMA over Converged Ethernet
  - With and without PFC support (RoCE v1 vs. v2)

- And even in WAN
  - iWARP (Internet Wide-Area RDMA Protocol)
  - Implemented over TCP/IP, no need for lossless network
  - Significant challenges here, especially over long distances



Introduction

- Background
  - What is a router?
  - Basic router architecture
- Basic packet processing in routers
  - IP address lookup
  - Packet buffering
  - Switching
- Data Center Networking
  - Design Dimensions
  - Topology
  - Transport
- Machine Learning and Computer Networks

Machine Learning and Computer Networks

- Machine Learning (ML):
  - Significant growth in the recent years
  - Lots of attention and impact

- How does it affect computer networking?
  - **Networks for ML**: can traditional networks handle ML requirements?
    - Our focus is on data center networks here.
  - **ML for Networks**: how can ML be used to enhance computer networks?
    - Networks in general.

Networks for Machine Learning

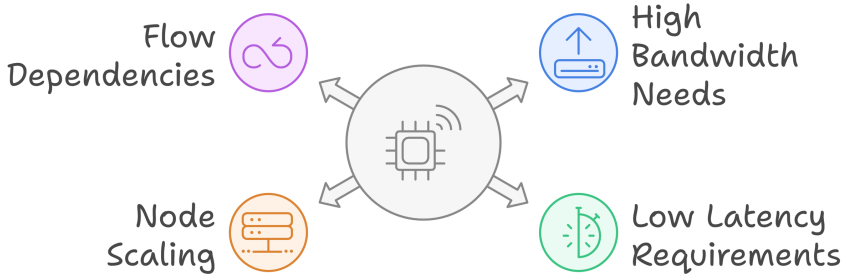
- Data Center Networks have evolved significantly
  - To accommodate demands for modern applications.
  - Example: many novel congestion control algorithms in recent years:
    - Swift, timely, HPCC, DCQCN, …
    - Enablers: more accurate information from network (exact queue occupancy), assumptions about start rate (start at line rate), etc.

- Machine learning applications have grown significantly as well.
  - Used more in various domains, solving a wide range of problems.
  - At the same time, ML applications have higher demands from the underlying network

- Question: are existing DCN solutions enough?
  - I.e., can they provide the high-performance connectivity needed for ML applications?

What Makes ML Different: Challenges

- ML workloads can be extremely large:
  - E.g., Training of Large-Language Models (LLMs)
  - Need various forms of parallelism
    - Data parallelism
    - Model parallelism
    - Hybrid



- ML workloads have extremely high requirements from the underlying network:
  - High bandwidth
  - Low latency
  - Low jitter (variations in delay)

Moore's Law: the number of transistors on a microchip will double approximately every two years.
- For years, Moore's law meant we could easily grow compute power according to growth in demand.

End of Moore's Law: recently, we have hit a wall and cannot continue growing compute per node as predicted by the Moore's Law.
- However, the demand keeps growing …
- For ML even faster than what Moore's law could handle.

We need to add more nodes to scale to the demands of ML means.

All of this leads to significant pressure on the network (throughput, latency, reliability, …)

What Makes ML Different: Challenges

- Handling many independent flows in a network makes many network problems easy (easier) to solve
  - Random arrivals, each flow has a small share of bandwidth
  - Why?
- In ML, we have few flows
  - Having few flows means each flow can have a large fraction of link bandwidth
    - ⇒ Any interaction between flows can lead to significant performance degradation
- In ML flows have direct/indirect dependencies
  - Dependence between compute and communication
    - ⇒ flows directly or indirectly depend on each other
    - ⇒ Performance degradation in one flow can impact the performance of the entire job
- Providing high-performance connectivity for ML workloads is extremely challenging.
  - Due to scale, high-performance requirements (bandwidth, latency, …), larger flows, dependencies, …
- Example: load balancing in ML
  - Even two flows sharing a path can significantly reduce the overall performance.

What Makes ML Different: Opportunities

- ML workloads are more more predictable
  - Repeating patterns of communication
  - Collective Communications: scatter, gather, all-reduce, …
- Knowing communication patterns ⇒ opportunities for …
  - Building specialized hardware
    - E.g., topology that matches the flow requirements
      - Today's most successful ML networking solutions
  - Build network solutions that adapt based on application requirements
    - Application-aware scheduling
    - Reconfigurable topology
    - Adaptive routing, …
- Even without prediction, access to "Collective Communication Libraries" can provide significant opportunities.
  - Examples to come.

Final Comments

- Largescale machine learning (training and inference) has led to significant pressure on computer networks.
- Major challenges for computer networks
  - Latency, throughput, loss, … requirements
- Opportunities to enhance networks
  - Take advantage of ML workload properties
  - Ability to integrate with existing solutions
- We will go through examples of these challenges and opportunities in the rest of this course.