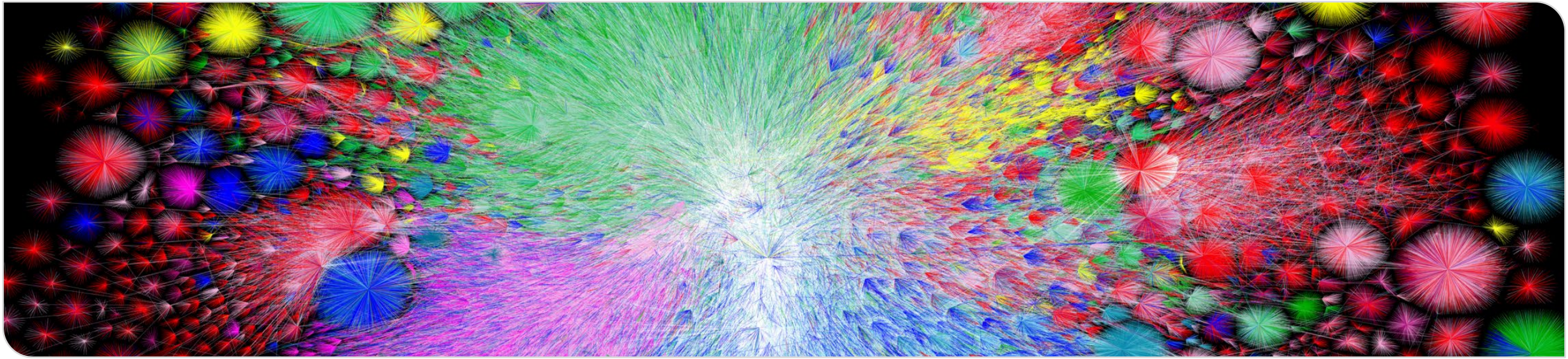
Karlsruher Institut für Technologie

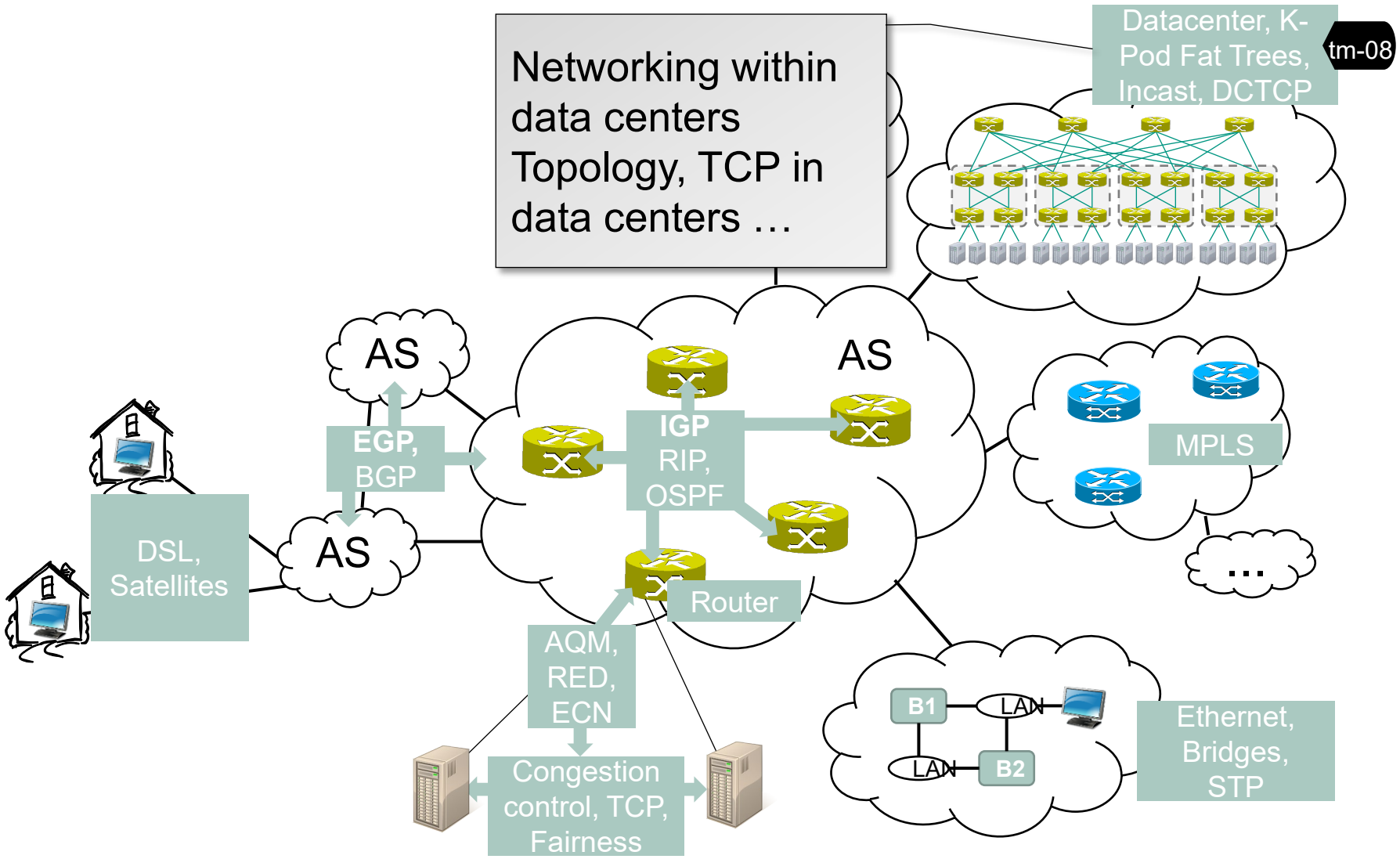
8. Data Center Networking

Prof. Dr. Martina Zitterbart
Institute of Telematics



Karlsruher Institut für Technologie

Networking within data centers Topology, TCP in data centers …



8

Data Center Networking

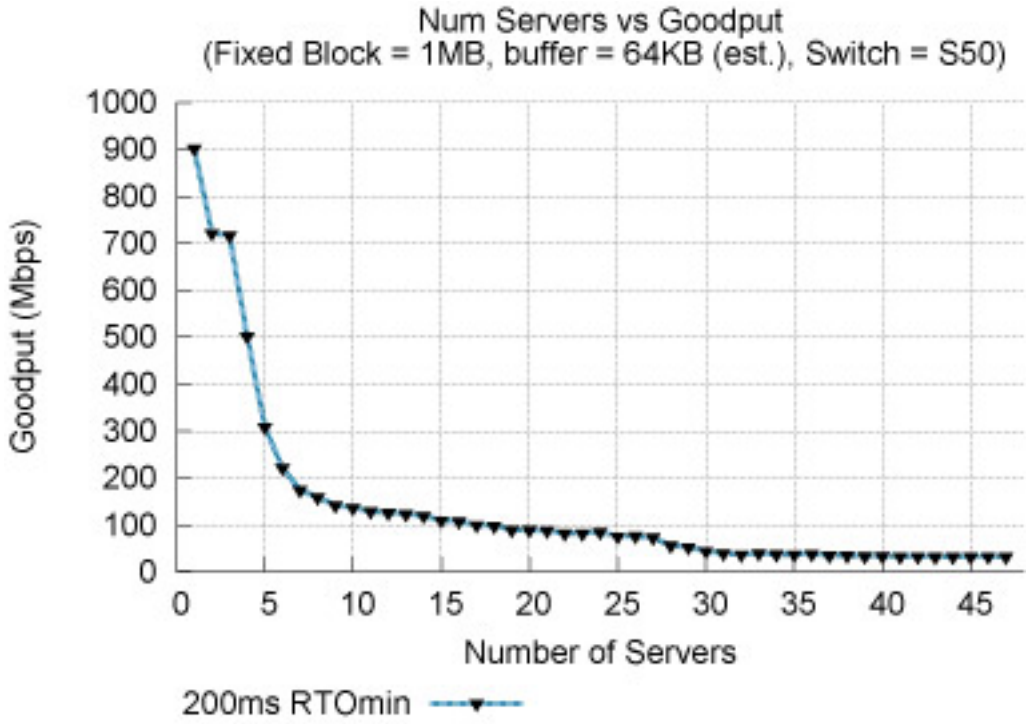
8.0 Communication Matters

Goodput at Application Layer

- Synchronized test application
  - Requests data block of 1 MByte which is evenly distributed over N servers
  - Each server responds with $^1/_N \ of \ 1 \ MByte$
  - Next data block can only be requested if actual data block is received completely

- Problem: incast collapse
  - Goodput drops dramatically as number of servers increases
    - With 47 servers only 3% goodput observed

- Achieved goodput

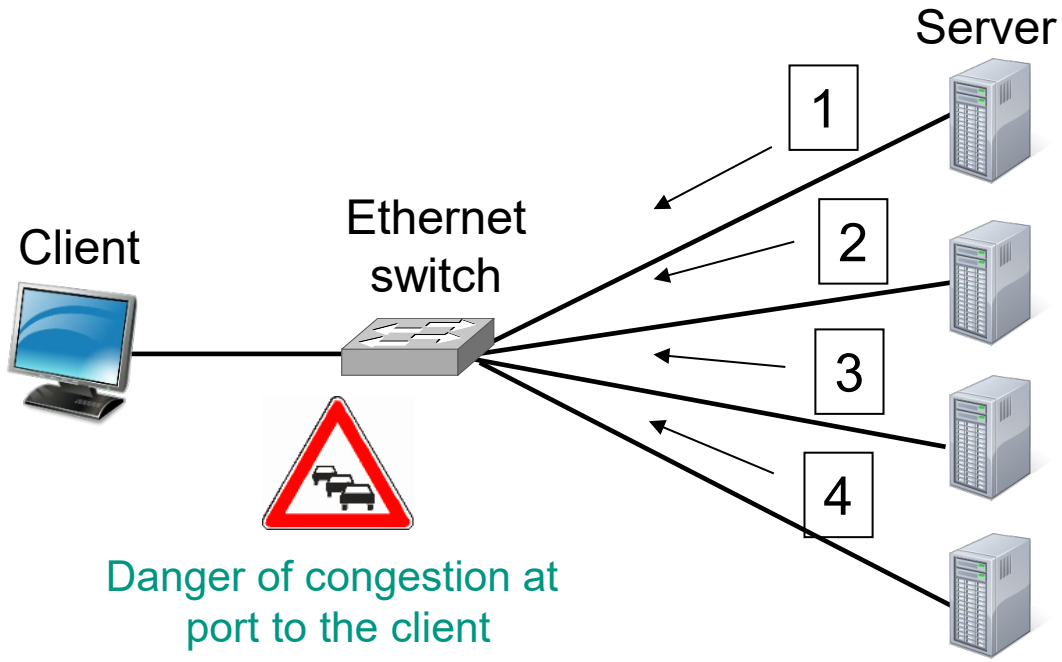


[Vasu09]

Incast Communication Pattern

- Incast: many-to-one communication
  - Common in applications such as web search, analytics …
  - Request is distributed to multiple servers
  - Servers respond almost synchronously
    - Total number of responses can cause overflows in small switch buffers
    - Often, applications can not continue until all responses are received or provide less valuable result

- Scenario



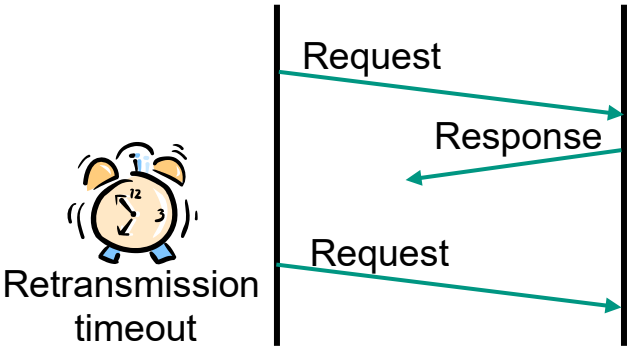
Packet Loss in Ethernet Switch

- Situation
  - Ports often share buffers
  - Individual response may be small (a few kilobytes)

- Packet losses in switch possible because
  - Large number of responses can overload a port,
  - High background traffic shares same port with incast traffic or
  - High background traffic on a different port as incast traffic

- Packet loss causes TCP retransmission timeout

  → no further data is received,
  so no duplicate acks can be generated



Consequence

- Slowest TCP connection determines application performance
- TCP instance affected by congestion (packet loss) must wait for retransmission timeout
  - No further data can be sent on this TCP connection during that time

  → Application blocked, i.e, response time increases or result gets imprecise

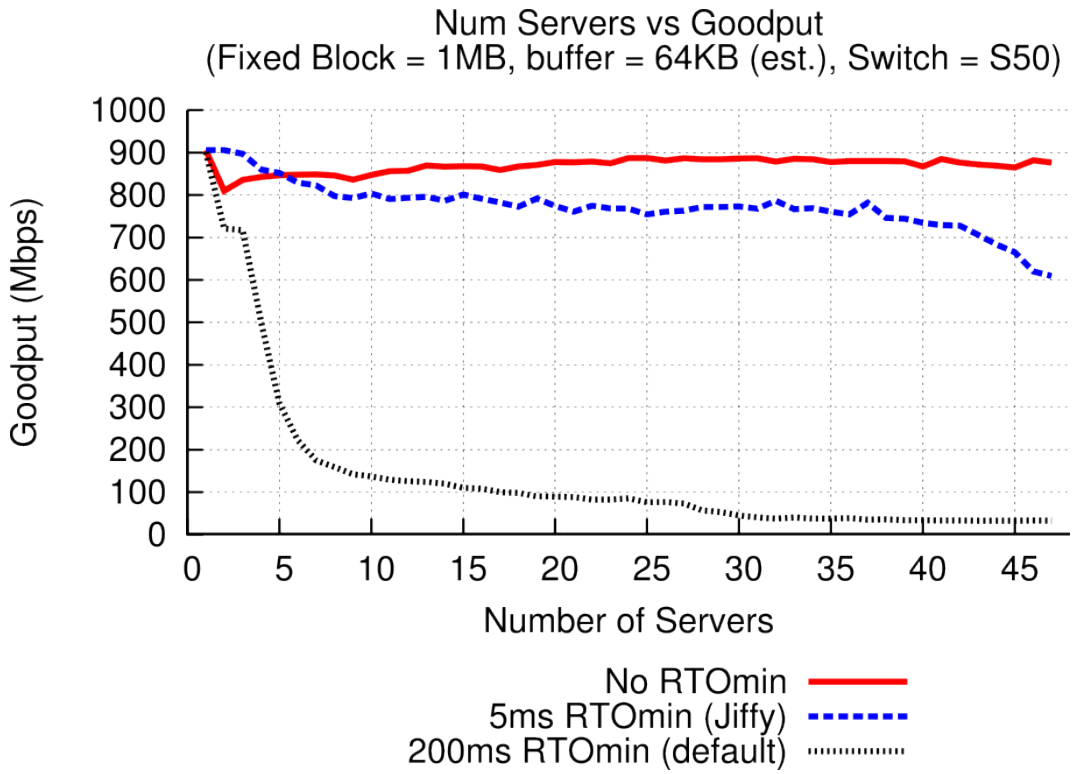
- Problem
  - There is a minimum value for the retransmission timer RTO_{min}
  - According to RFC 6289: $RTO_{min} = 1s$
    - RTTs in data centers are typically much shorter: $RTO_{datacenter} < 1ms$
    - → RTO_{min} highly inefficient: leads to long waiting times and, thus, increased latencies

Improvements

- Smaller minimum retransmission timeout: same order of magnitude as RTT
  - Increase timer granularity from milliseconds to microseconds
    - Implementation: Linux High Resolution Timers
- Desynchronization
  - Exponential backoff of retransmission timer
    - ... leads to longer response times
  - Randomization
    - Desynchronize retransmissions
    - $timeout = \left(RTO + \left(rand(0{,}5) \times RTO\right)\right) \times 2^{backoff}$
    - TCP connections with subsequent timeouts become more desynchronized

[Vasu09]

Impact on Goodput



- With high resolution timers and retransmission timeouts of 5 ms or below, incast collapse is avoided
  - Relatively stable for up to 40 servers

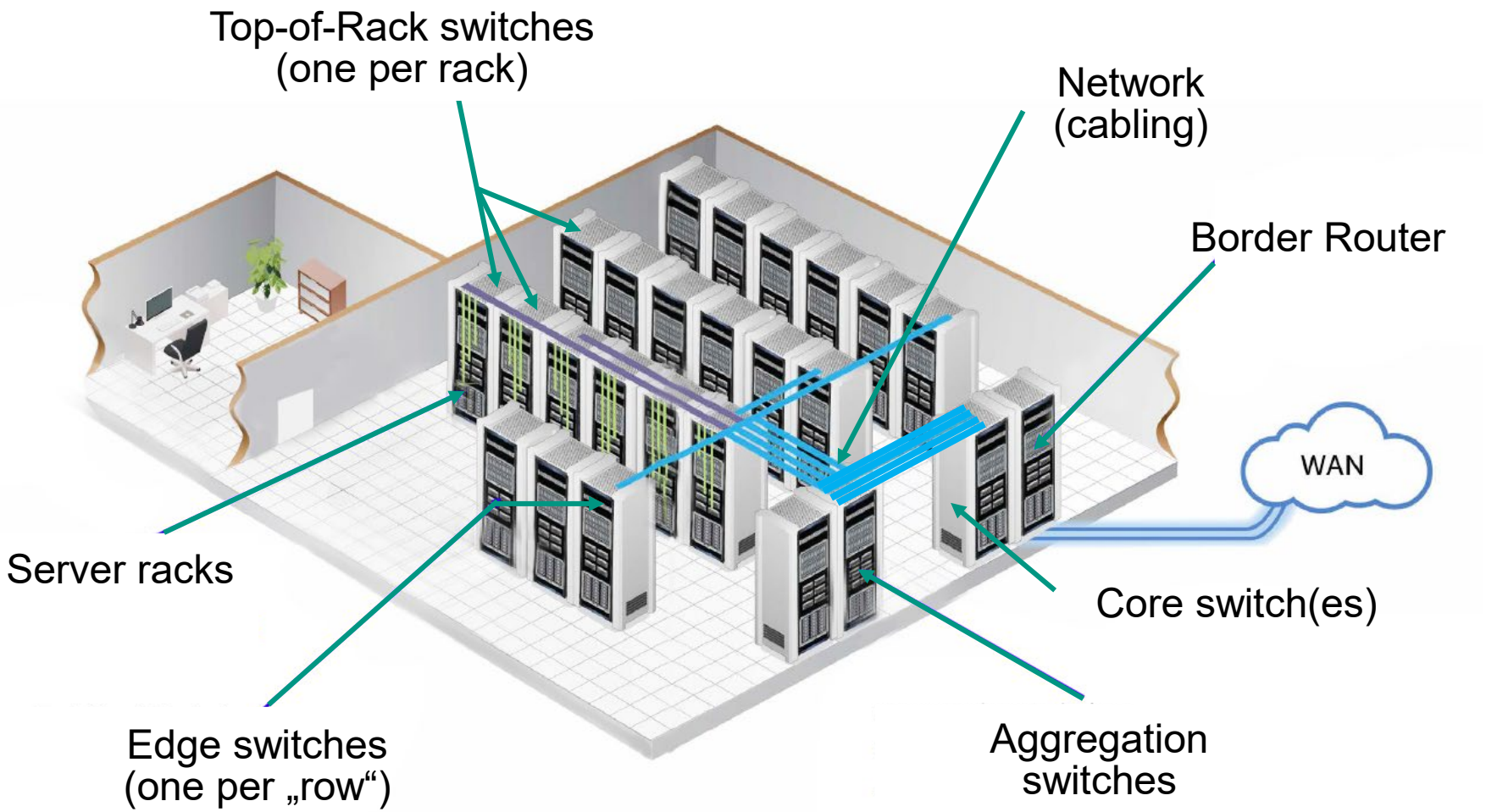
[Vasu09]

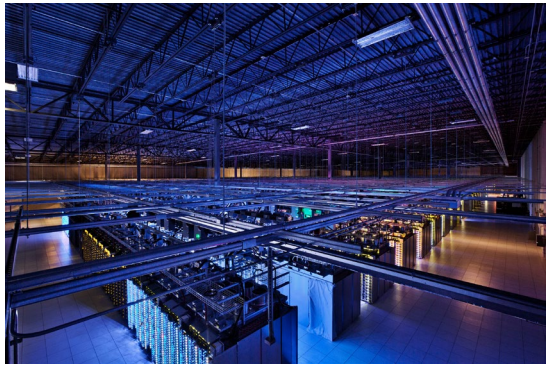
8.1 Introduction

Data Centers

- Internet companies (e.g., Google, Amazon) build (distributed) data centers of increasing size
  - They host tens to hundreds of thousands of servers
  - They are the engines behind Internet applications that we use on a daily basis
- Servers are the worker bees in a data center
  - Commodity servers
  - Stacked in racks
- Interconnection
  - (Complex) internal network topologies that efficiently and reliably interconnect the servers
    - Provide redundant paths
    - Typically based on Ethernet (as „fabric“) and commodity switches with shallow shared buffers
  - Connected to the Internet through border routers

Simplified Sketch of a Data Center



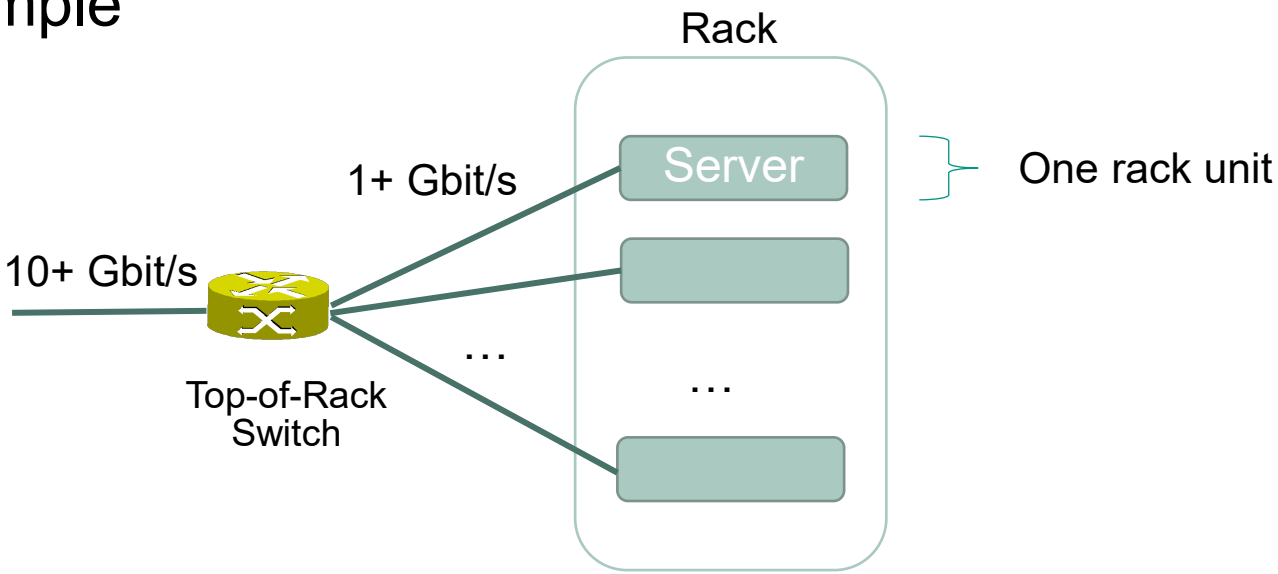


https://www.coherent.com/resources/brochures/communication-cables/pluggable-finisar-transceivers-for-the-data-center-br.pdf

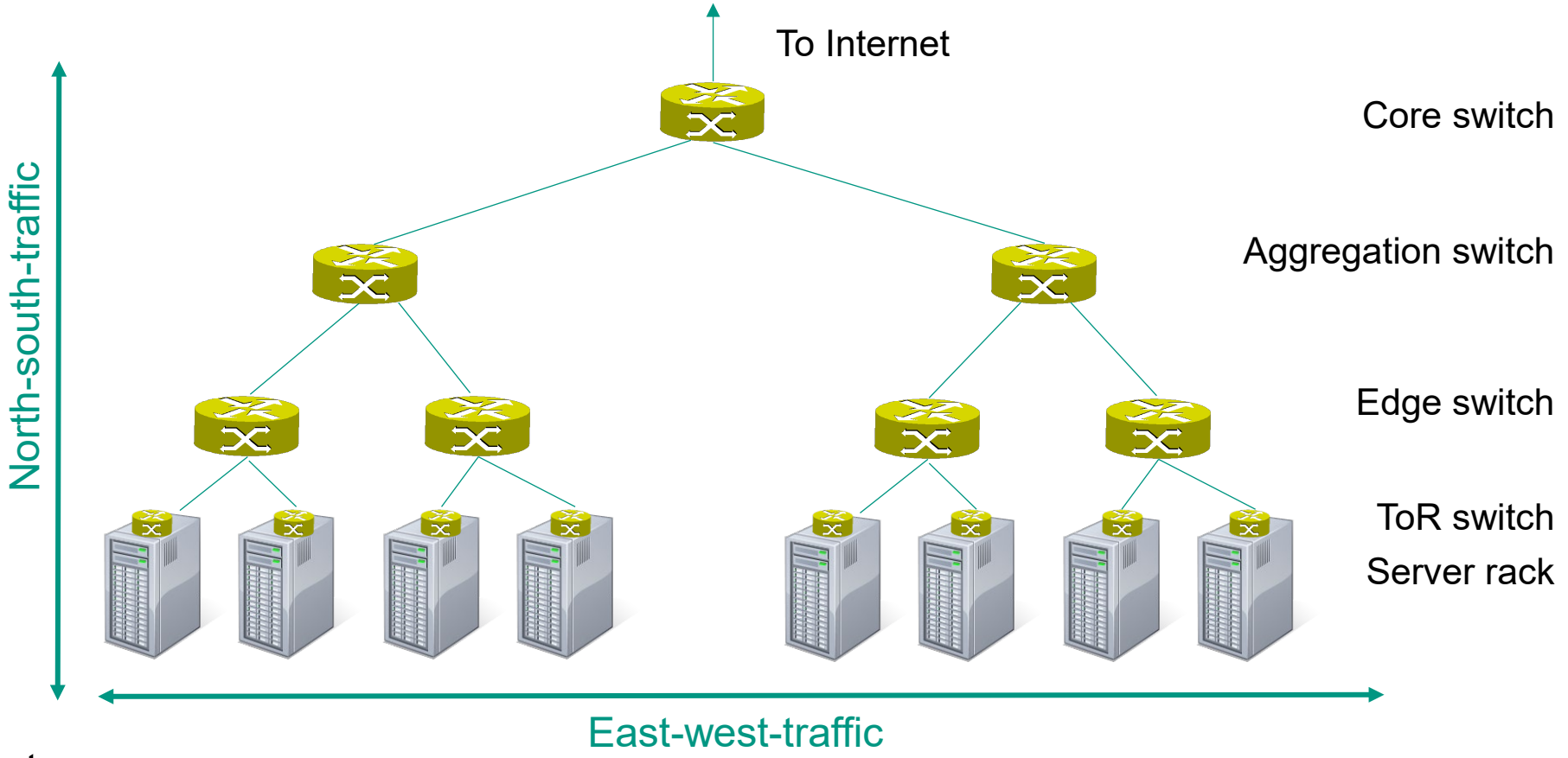
https://www.google.com/about/datacenters/gallery/#/tech

Top-of-Rack Switches

- Top-of-Rack (ToR) Ethernet switches connect servers within a rack
  - E.g., 48 port Gigabit Ethernet switch with 10+ Gbit/s uplink
    - Switches typically have small buffers
  - Can be placed directly at the „top“ of the rack
    - Can also be placed outside of the rack, e.g., at the end of a row of racks
  - Typical data center rack has 42-48 rack units per rack
- Example



Tree-Based Data Center Network Topology



- Note
  - Tree-based topologies are typical in simple, smaller data center networks
  - Single LAN, e.g., not really suitable for larger data center networks
    - Addressing, scalability (broadcast traffic) …

Two Types of Traffic

- North-south traffic
  - Result of external request from the public Internet
  - Between external clients and internal servers
  - Predominant traffic in early data centers with mostly simple distributed applications

- East-west traffic
  - Between internal servers and server racks
    - Intra data center traffic
  - Result of internal applications, e.g.,
    - MapReduce
    - Storage data movement between servers
  - Predominant traffic in many large-scale data centers with distributed data analytics and machine learning


[HoHR22]

North-south traffic may cause east-west traffic.

Ratio between east-west traffic and north-south traffic may vary widely.

Large-scale distributed data center also face high inter-data center traffic between multiple locations. Different traffic pattern than traditional north-south traffic

Data Center Characteristics

- Scalability and flexibility
  - Incrementally expanded and updated
    - Ability to grow a data center without major disruption and re-organization
      - E.g., addressing and topology
      - Operators cannot afford extensive downtime for reconfiguration
    - Ability to deal with heterogeneous components
- Efficient routing/forwarding
  - Several distinct shortest paths between any two servers
    - High fault tolerance, high efficiency
    - Utilize network efficiently
    - (ultra) low latencies, very high data rates
- (Very) low round trip times (RTT)
  - Server in a single data center typically in close geographical proximity
  - Values in the range of microseconds instead of milliseconds

  →RTT is orders of magnitude smaller than in traditional Internet
- Reduced statistical multiplexing
  - Compared to traditional Internet – fewer flows
  - Individual flows can dominate buffer occupancy

Data Center Characteristics

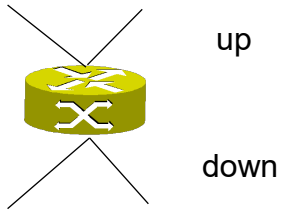
- Very different requirements need to be served concurrently
  - (ultra) high latency sensitivity
  - Very high data rates
- Traffic flows
  - (very) many short-lived flows
    - E.g., small search requests
    - Flow completion time matters (→ latency)
  - Few long-lived flows with high amounts of data
    - E.g., backup
- Incast communication (many-to-one)
  - Roughly synchronized traffic
    - Multiple servers transmit data to one target
    - Application examples: MapReduce, web search, advertising, distributed storage systems, ...
    - Design Pattern: partition - aggregate
  - Asynchronous traffic
    - Many flows start asynchronously but are destined for same target
    - Application example: distributed machine learning
- Leads to micro bursts
  - Bursty traffic with microscopic time scale
    - Hundreds of microseconds
    - Cannot be handled by end-to-end congestion control

8.2 Fat-Tree Networks

8.2.1 Basics

Fat-Tree Networks

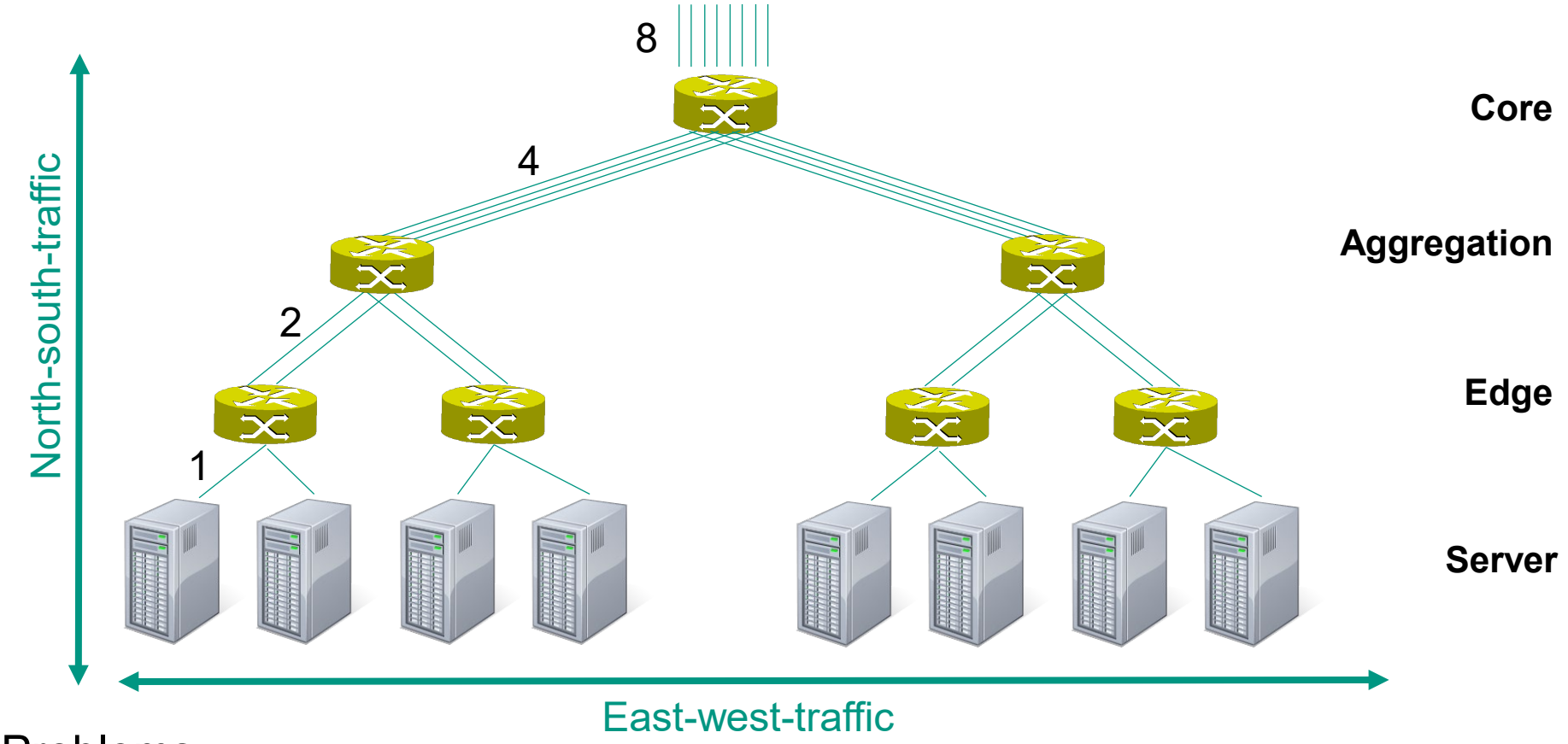
- General problem
  - Connect large number of servers by using switches that only have a limited number of ports
- Characteristics
  - For any switch
    - Number of links going down to its children is equal to number of links going up to its parents
  - The links get „fatter“ towards the top of the tree



- Adapted for use in data center networks
  - K-pod fat trees

[AILV08]

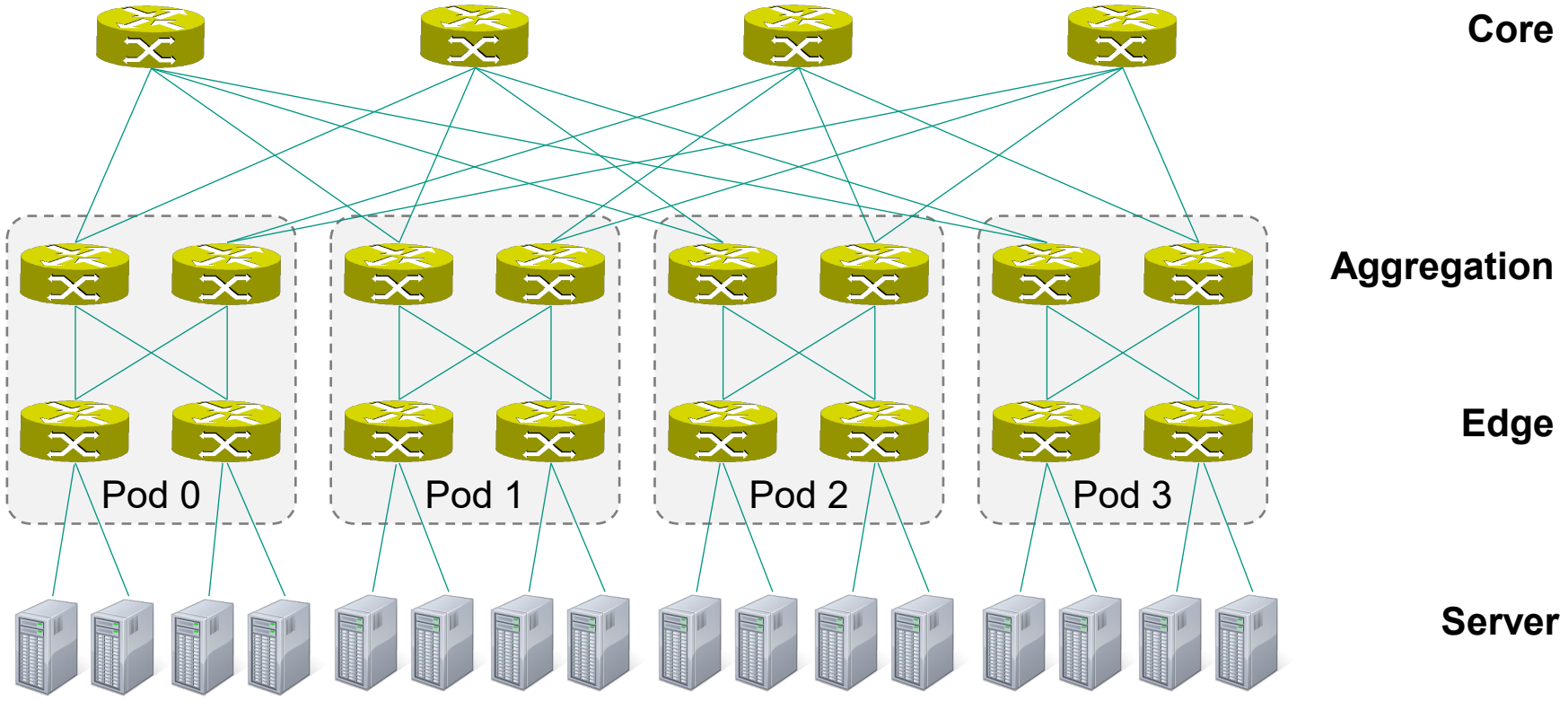
Fat-Tree Network: Example



- Problems
  - Switches need different numbers of ports
  - Number of ports increases towards core → high performance needed
    - Switches with high number of ports are (more) expensive

8.2.2 K-Pod Fat Tree

K(=4)-Pod Fat-Tree



K-Pod Fat-Tree: Characteristics

- Each switch: k ports

- Edge and aggregation switch arranged in k pods
  - $\frac{k}{2}$ edge switches and $\frac{k}{2}$ aggregation switches per pod

    → overall $\frac{k^2}{2}$ edge and $\frac{k^2}{2}$ aggregation switches → k^2 switches in pods

- $\left(\frac{k}{2}\right)^2$ core switches, each of which connects to k pods

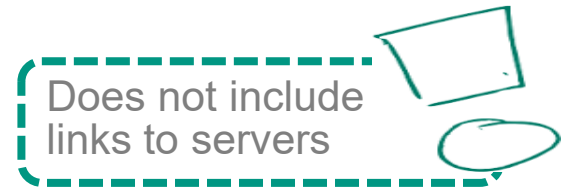
→ Overall $k^2 + \left(\frac{k}{2}\right)^2 = \frac{5}{4}k^2$ switches

- Each edge switch connected to $\frac{k}{2}$ servers

→ Overall $\frac{k^2}{2} \cdot \frac{k}{2} = \frac{k^3}{4}$ servers can be connected

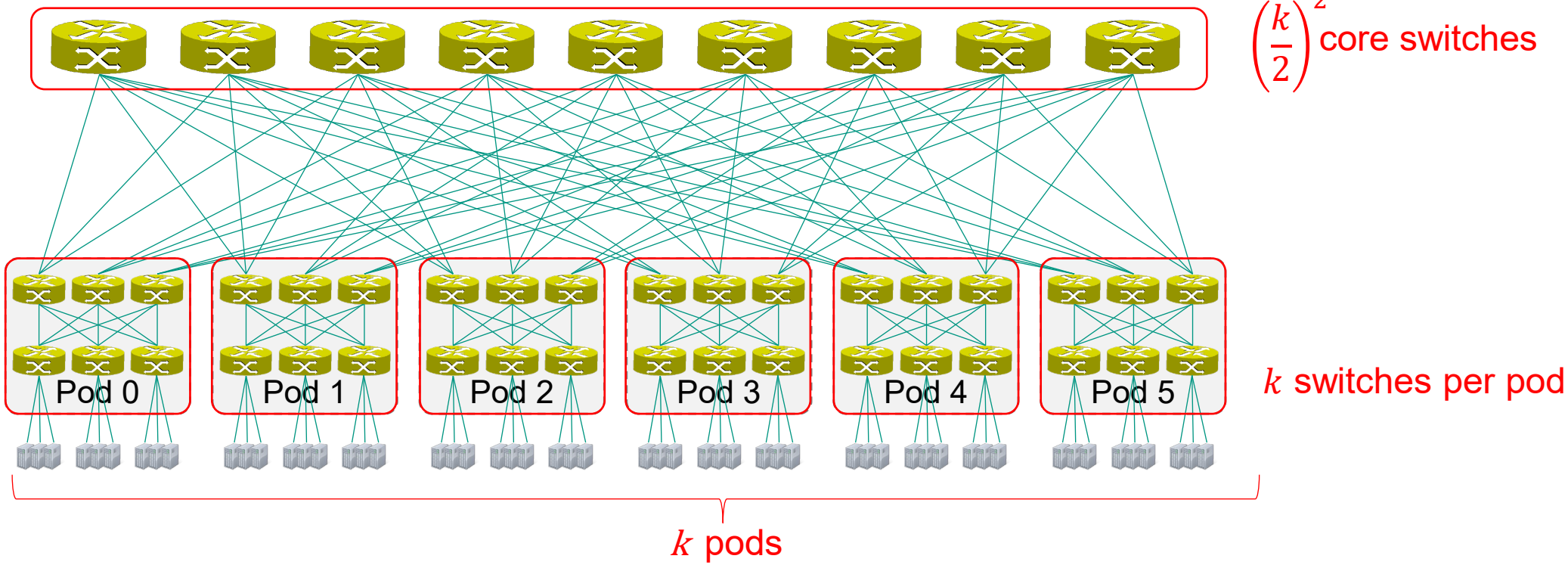
- Each aggregation switch connected to $\frac{k}{2}$ edge and $\frac{k}{2}$ core switches

→ Overall $2k \cdot \left(\frac{k}{2}\right)^2 = \frac{k^3}{2}$ links



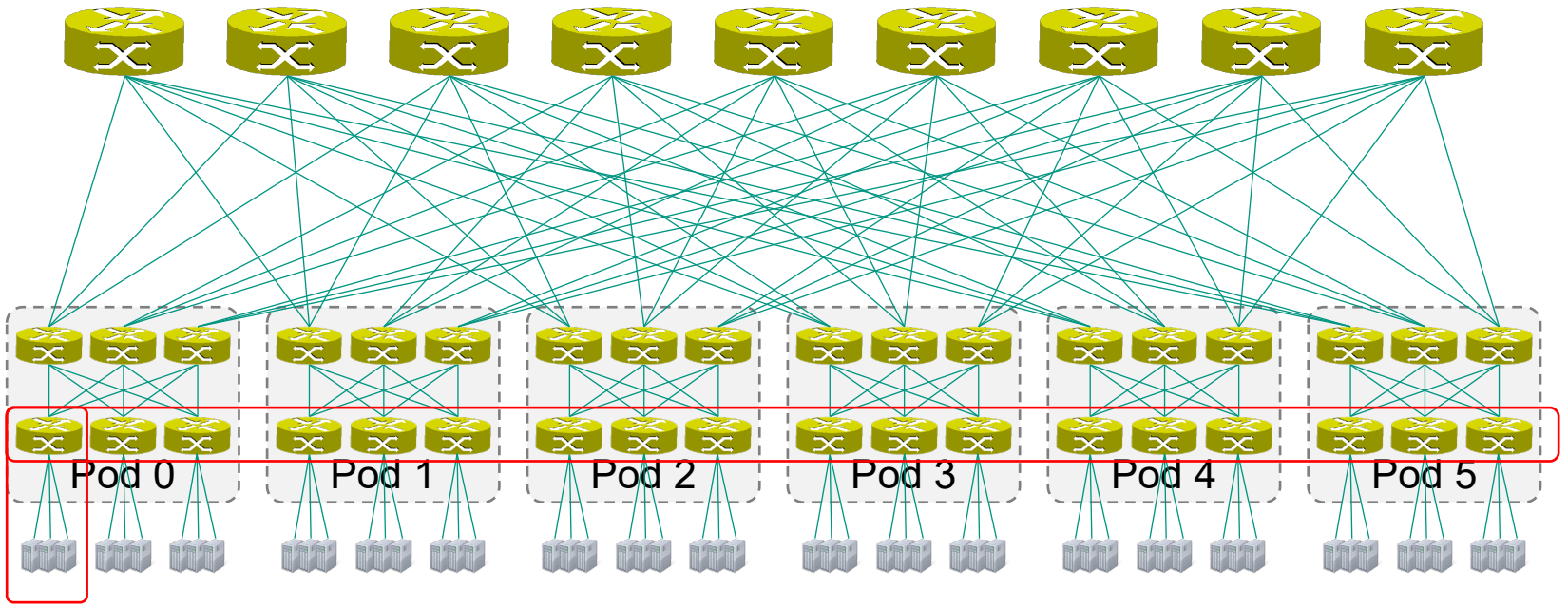
Characteristics

- How many switches?
  - $k^2 + \left(\frac{k}{2}\right)^2 = \frac{5}{4}k^2$ switches
  - Example k=6: 45 switches



Characteristics

- How many servers?
  - $\frac{k}{2} \cdot (k \cdot \frac{k}{2}) = \frac{k^3}{4}$ servers
  - Example k=6: 54 servers

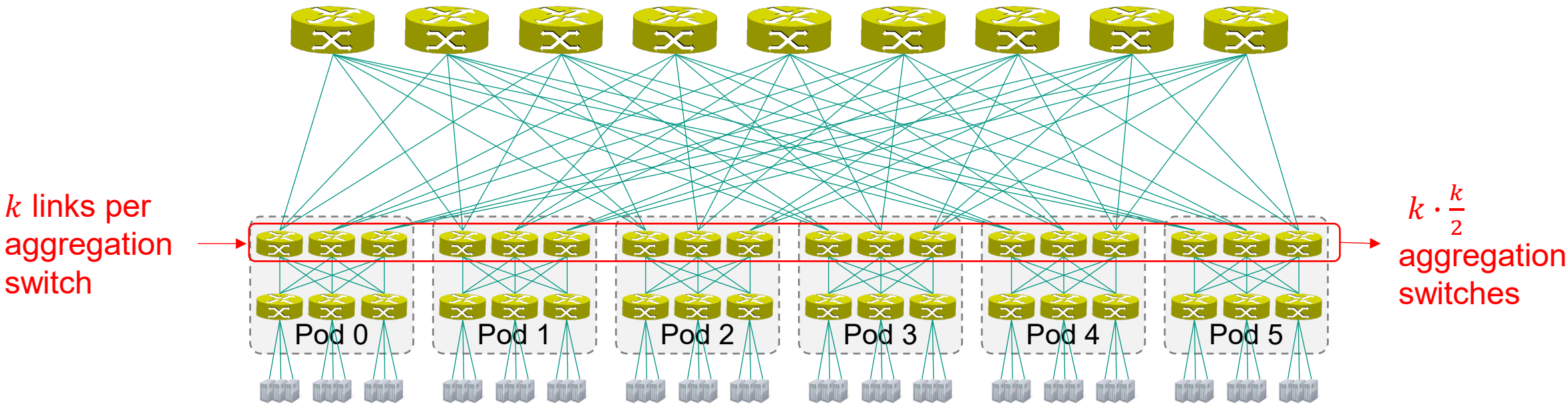


$k \cdot \frac{k}{2}$ edge switches

$\frac{k}{2}$ servers per edge switch

Characteristics

- How many links (without links to servers)?
  - Observation: all links must be connected to an aggregation switch
  - $k \cdot (k \cdot \frac{k}{2}) = \frac{k^3}{2}$ links
  - Example k=6: 108 links



Example: 48-Pod Fat Tree

- $k = 48$
  - Edge layer
    - $\frac{k}{2} = 24$ switches with $\frac{k}{2} = 24$ server per switch
  - Aggregation layer
    - $\frac{k}{2} = 24$ switches

- Overall
  - $k \cdot \frac{k}{2} \cdot \frac{k}{2} = \frac{k^3}{4} = 27648$ servers
  - $k \cdot \frac{k}{2} = 1152$ subnets with $\frac{k}{2} = 24$ servers each
  - $k^2 \cdot \left(\frac{k}{2}\right)^2 = 2880$ switches

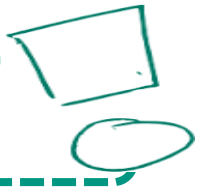
- Between any given pair of servers in different pods
  - $\left(\frac{k}{2}\right)^2 = 576$ equal-cost paths

K-Pod Fat-Tree: Numbers for Different K

- Number of servers, switches and links for different values of k

k	Server	Switches	Links
4	16	20	32
6	54	45	108
8	128	80	256
16	1024	320	2048
24	3456	720	6912
32	8192	1280	16384
48	27648	2880	55296

Every link is in fact a physical cable
→ high cabling complexity

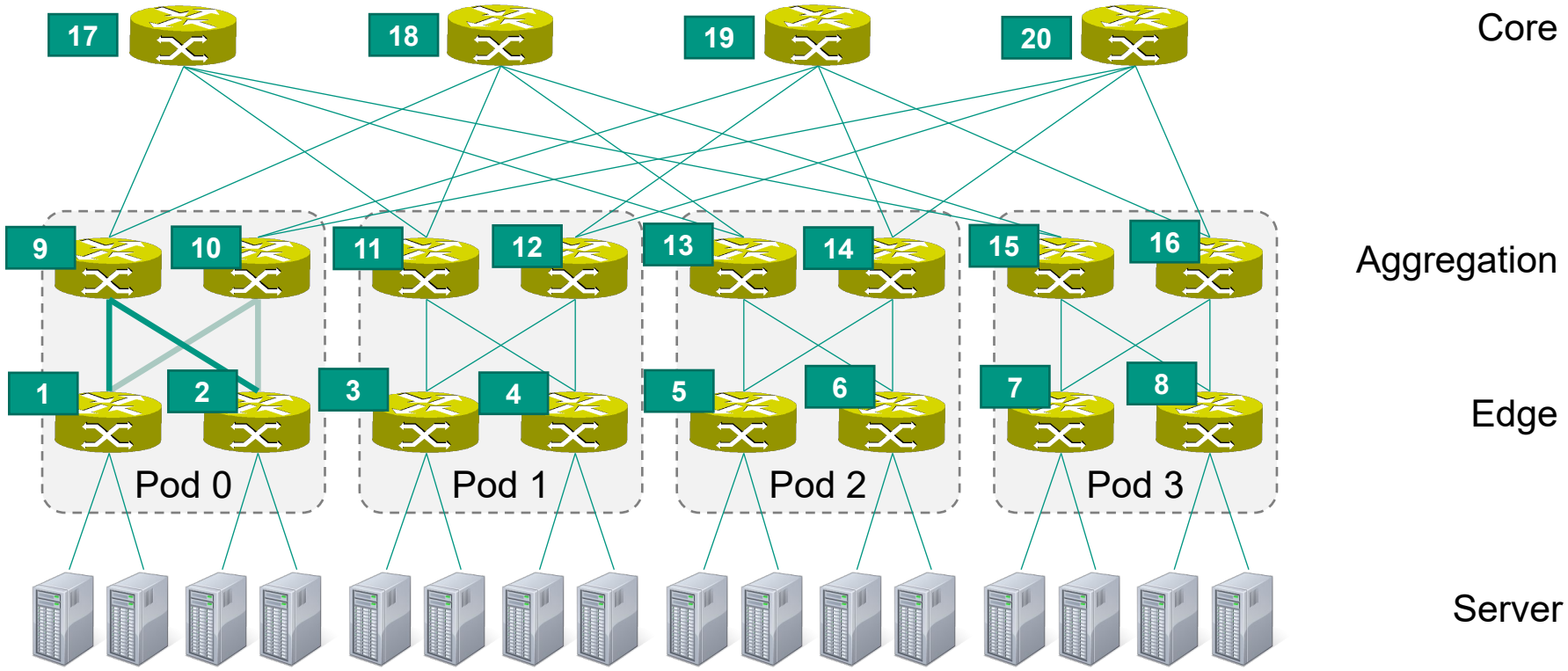


Some Advantages and Disadvantages

- Advantages
  - All switches are identical
  - Cheap commodity switches can be used
  - Multiple equal cost paths between any servers

- Disadvantages
  - High cabling complexity

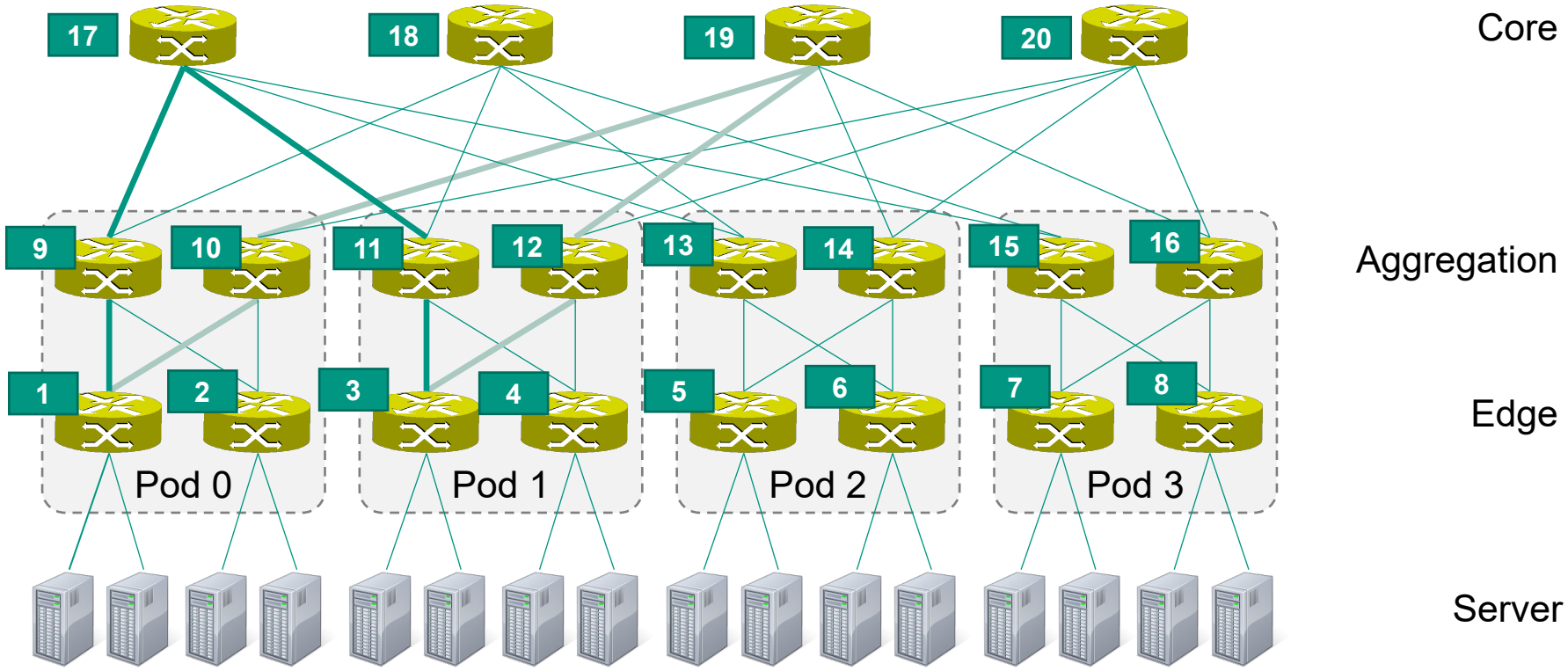
Routing Paths (Intra-Pod)



- Valid paths, e.g. in pod 0
  - Child server of switch 1 wants to send data to child server of switch 2
  - Possible paths: 1-9-2 and 1-10-2

  → $\frac{k}{2}$ paths from source to destination within a pod

Routing Paths (Inter-Pod)



- Traffic between servers in different pods
  - E.g., between switch 1 and switch 3
    - 1-9-17-11-3, 1-9-18-11-3, 1-10-19-12-3, 1-10-20-12-3
  - → $\left(\frac{k}{2}\right)^2$ paths from source to destination between different pods

Routing/Forwarding in K-Pod Fat-Tree

- Traditional IGPs, such as OSPF select a single shortest path
  - K-pod fat-tree: $(k/2)^2$ shortest paths exist among any two servers on different pods
    - Does not utilize redundant paths
    - May lead to bottlenecks at servers
  - Use ECMP supported by OSPF
    - High number of prefixes needed
    - Edge switch needs $\frac{k}{2}$ prefixes for every other subnet
      - One prefix per possible aggregation switch
    - k pods, $k/2$ subnets per pod $\rightarrow \frac{k^2}{2}$ subnets
    - $k \cdot \left(\frac{k}{2}\right)^2$ total prefixes per edge switch

- … make use of the structure of fat-trees
  - The following address assignment and routing provide an example

[AILV08]

8.2.3 Address Assignment in K-Pod Fat Tree

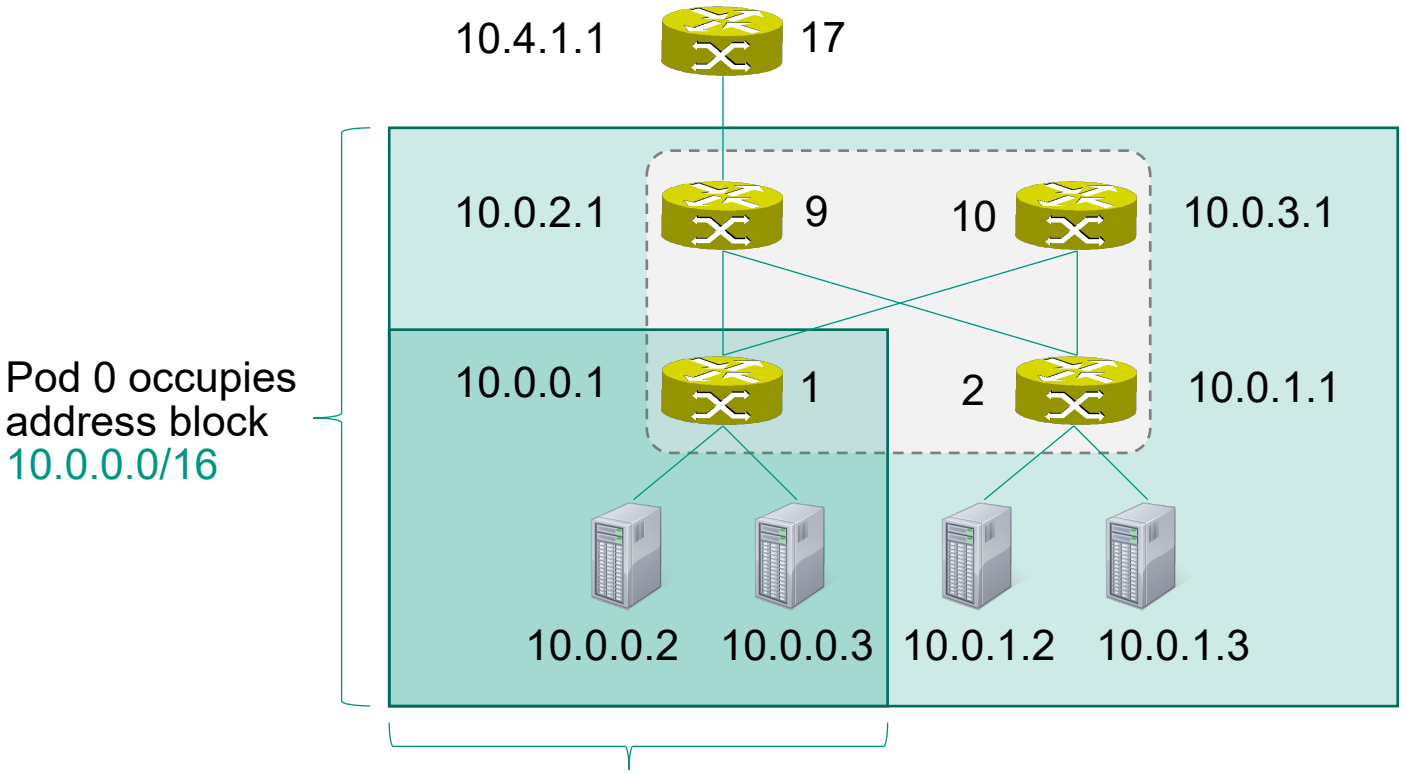
Address Assignment

- Assign private IPv4 address block 10.0.0.0/8 to data center network
  - Pods are enumerated from left to right: $[0, k-1]$
  - Switches in a pod: IP address 10.pod.switch.1
    - Edge switches are enumerated from left to right: $\left[0, \frac{k}{2}-1\right]$
    - Enumeration continues with aggregation switches from left to right: $\left[\frac{k}{2}, k-1\right]$
  - Servers: IP address 10.pod.switch.ID
    - Based on the IP address of the connected edge switch
    - IDs are assigned to servers from left to right starting with 2
  - Core switches: IP address $10.\ k.\ x.\ y$
    - x : starts at 1 and increments every $\frac{k}{2}$ core switches
    - y : enumerates each switch in a block of $\frac{k}{2}$ core switches from left to right, starting with 1

[AILV08]

Example: Address Assignment 4-Pod Fat-Tree

- IP address assignment for pod 0

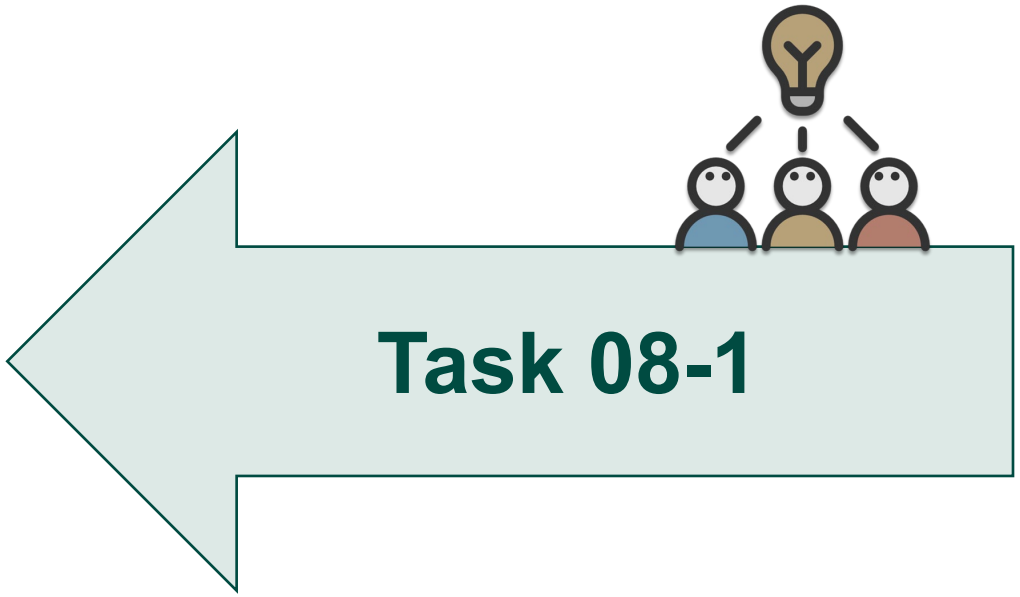


Switch 1 and its connected servers are in address block 10.0.0.0/24

→ /24 subnet with $\frac{k}{2}$ servers

Wasteful use of address space, but simplifies building routing tables


Task 08-1
Address Assignment


Task 08-1

8.2.4 Routing in K-Pod Fat Tree

Routing based on Two-level Prefix Lookup

- Routing table consists of two tables
  - Main routing table
    - Entries are „left-handed“: $/m$ prefix masks 1^m0^{32-m}
  - Secondary routing table
    - Entries are „right-handed“: $/m$ suffix masks $0^{32-m}1^m$
- Terminating prefixes
  - Destination subnet belongs to this pod, i.e., destination is located in „down direction“
- If longest-matching prefix in main routing table returns
  - Non-terminating prefix, then
    - Use longest matching suffix in secondary table
      - Forward to result of this matching
  - Terminating prefix, then
    - Forward to associated port and switch

[AILV08]

Routing Algorithm

- Intra-pod traffic
  - Switches in a pod have terminating prefixes to subnets in that pod
- Inter-pod traffic
  - For upward traffic, switches have default $/0$ prefix with secondary table matching ID
    - Least significant byte of destination IP address matters
  - For downward traffic, terminating prefixes are found in main table

- Core switches
  - Terminating prefixes are assigned for all network IDs
    - They point to the appropriate pod
    - Switch has terminating $/16$ prefix for packet $(10.pod.0.0\ /16, port)$
- Aggregation switch at destination pod
  - Has $(10.pod.switch.0\ /24, port)$ prefix to direct packet to destination edge switch

- Traffic diffusion occurs on the way to the core only (i.e., upward)

Routing Tables

- Pod switches include
  - Terminating prefixes for subnets in this pod
  - $/0$ prefix with secondary table matching server IDs for destinations outside this pod
  - Size of table
    - No more than $\frac{k}{2}$ prefixes and $\frac{k}{2}$ suffixes
- Core switches include
  - Terminating $/0$ prefixes to destination pods
    - Switch port i is connected to pod i
  - Size of table
    - No more than k prefixes

Inter-pod traffic

- $\frac{k}{2}$ entries in secondary routing table
  - Suffix 0.0.0.i/8
- Server-ID i: $[2, \frac{k}{2} + 1]$
- Switch z:
  - $\left[0, \frac{k}{2} - 1\right]$ for edge switches
  - $\left[\frac{k}{2}, k - 1\right]$ for aggregation switches
- Output port
  - $(i - 2 + z) mod \frac{k}{2} + \frac{k}{2}$
  - Traffic diffusion

$k = 4$

10.2.2.1 $z = 2$

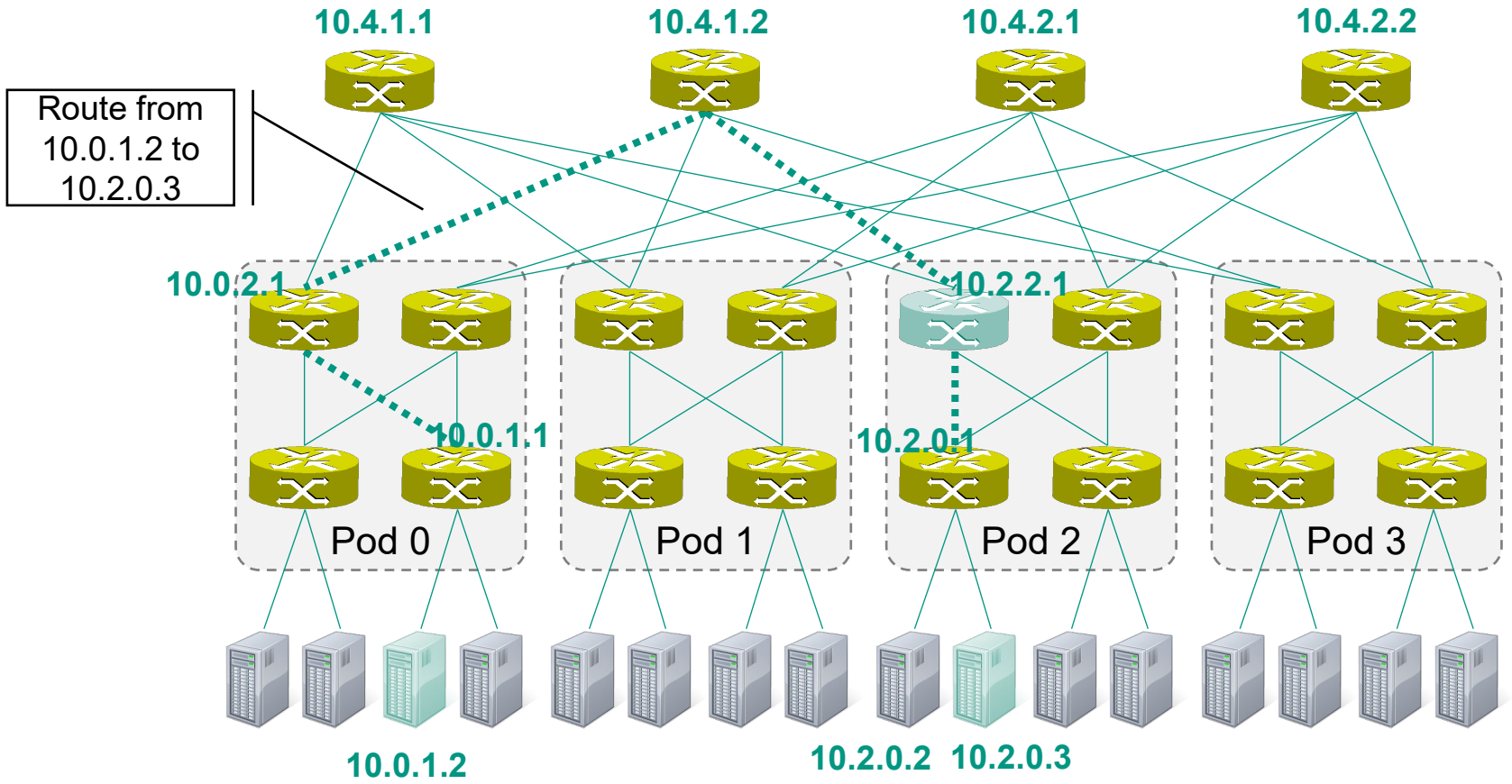
	Suffix	Output Port
$i = 2$	0.0.0.2/8	2
$i = 3$	0.0.0.3/8	3

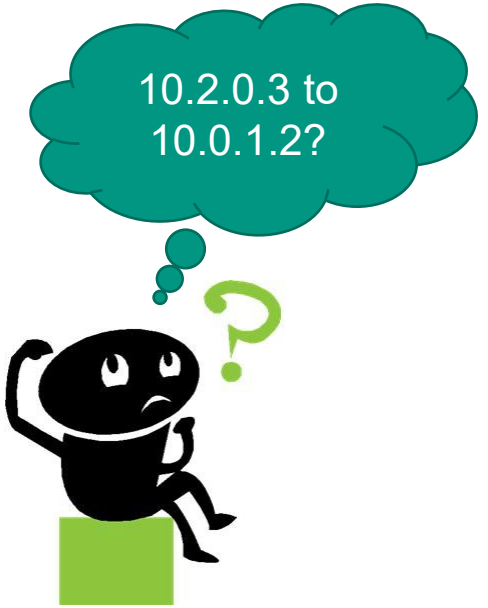
10.2.3.1 $z = 3$

	Suffix	Output Port
$i = 2$	0.0.0.2/8	3
$i = 3$	0.0.0.3/8	2

Ports flipped

Routing Example





- Table at switch 10.2.2.1

Prefix	Output Port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

Suffix	Output Port
0.0.0.2/8	2
0.0.0.3/8	3

[AILV08]

Routing Example

- Route from server 10.0.1.2 to server 10.2.0.3

1) Edge switch 10.0.1.1
   - Matches packet with /0 prefix
   - Forwards packet based on server ID byte (secondary table)
     - Matches 0.0.0.3 /8 suffix (points to port 2 and switch 10.0.2.1)
2) Aggregation switch 10.0.2.1
   - Same steps as edge switch before
   - Forwards on port 3 to switch 10.4.1.2
3) Core switch
   - Match to terminating 10.2.0.0 /16 prefix
     - Points to destination pod 2 on port 2 and switch 10.2.2.1
4) Aggregation switch 10.2.2.1
   - Match to terminating 10.2.0.0 /24 prefix
     - Point to switch responsible for subnet, port 0 and switch 10.2.0.1
5) Edge switch 10.2.0.1
   - Forwards packet to destination server

Routing Example

- Route from server 10.0.1.2 to server 10.2.0.2
  - Single-path IP routing would use same path
    - Since destinations are in the same subnet
    - Would eliminate fan-out possibilities of k-pod fat-tree
  - Two-level prefix routing
    - Allows edge switch 10.0.1.1 to forward data for server 10.2.0.2 to aggregation switch 10.0.3.1
      - Secondary table is applied

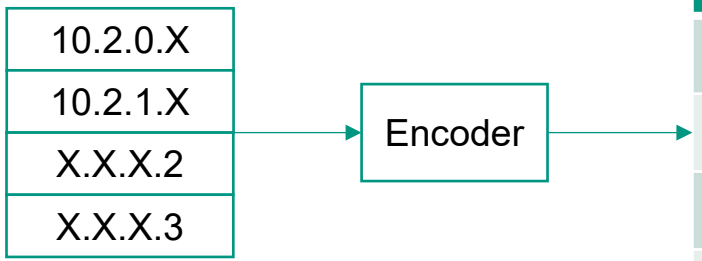
Utilize TCAM

- Example of two-level routing table

Prefix	Output Port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

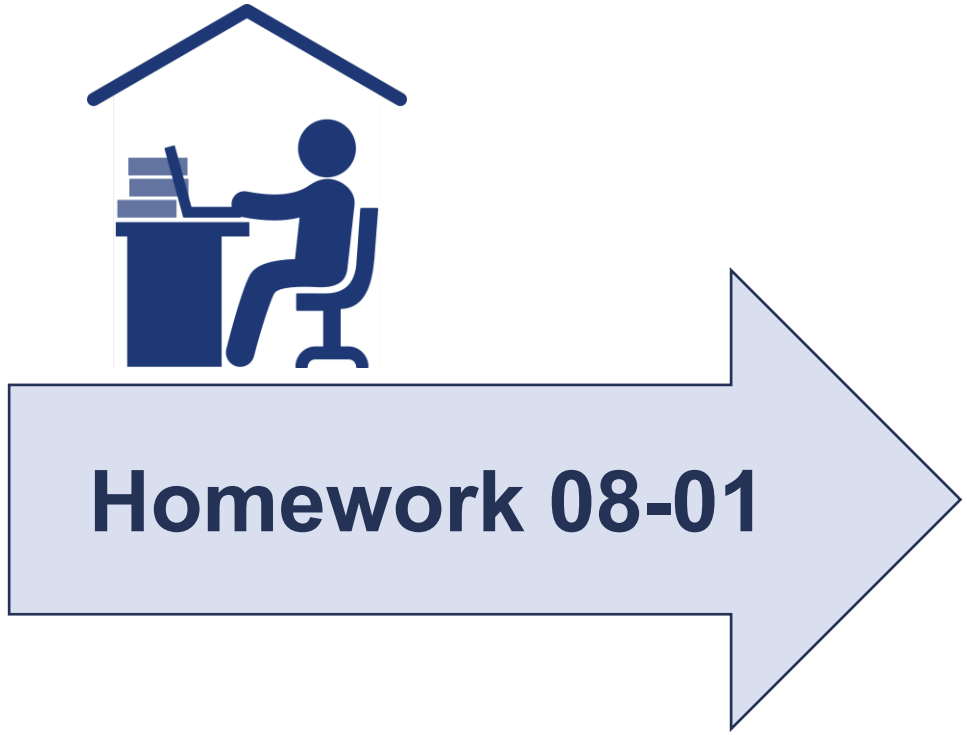
Suffix	Output Port
0.0.0.2/8	2
0.0.0.3/8	3

- Implementation in TCAM



Address	Next hop	Output port
00	10.2.0.1	0
01	10.2.1.1	1
10	10.4.1.1	2
11	10.4.1.2	3

Homework



Homework

Homework 08-02

8.3 Data Center Traffic Control

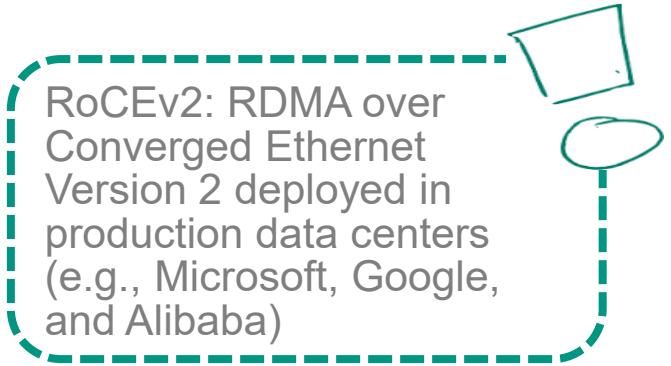
Motivation

- Use Ethernet as a "fabric" for data centers

- However
  - Ethernet offers best effort service only
    - No guaranteed delivery and delivery rates for packets
    - No guaranteed delays
      - At high load, full buffers → high delays, packet losses
    - No prioritization supported
      - May be needed for certain types of traffic

Basic Components

- RDMA (Remote Direct Memory Access)
  - Bypasses kernel networking stack
  - Zero memory copy techniques
  - Low CPU overhead
  - Needs lossless network (no packet loss due to buffering)
- Converged Ethernet
  - Provides PFC (Priority-based Flow Control)
  - Prevents buffer overflow
  - Coarse-grained control (on port level, not on flow level)
- Flow-based congestion control
  - Fine-grained control on flow level
  - E.g., DCTCP

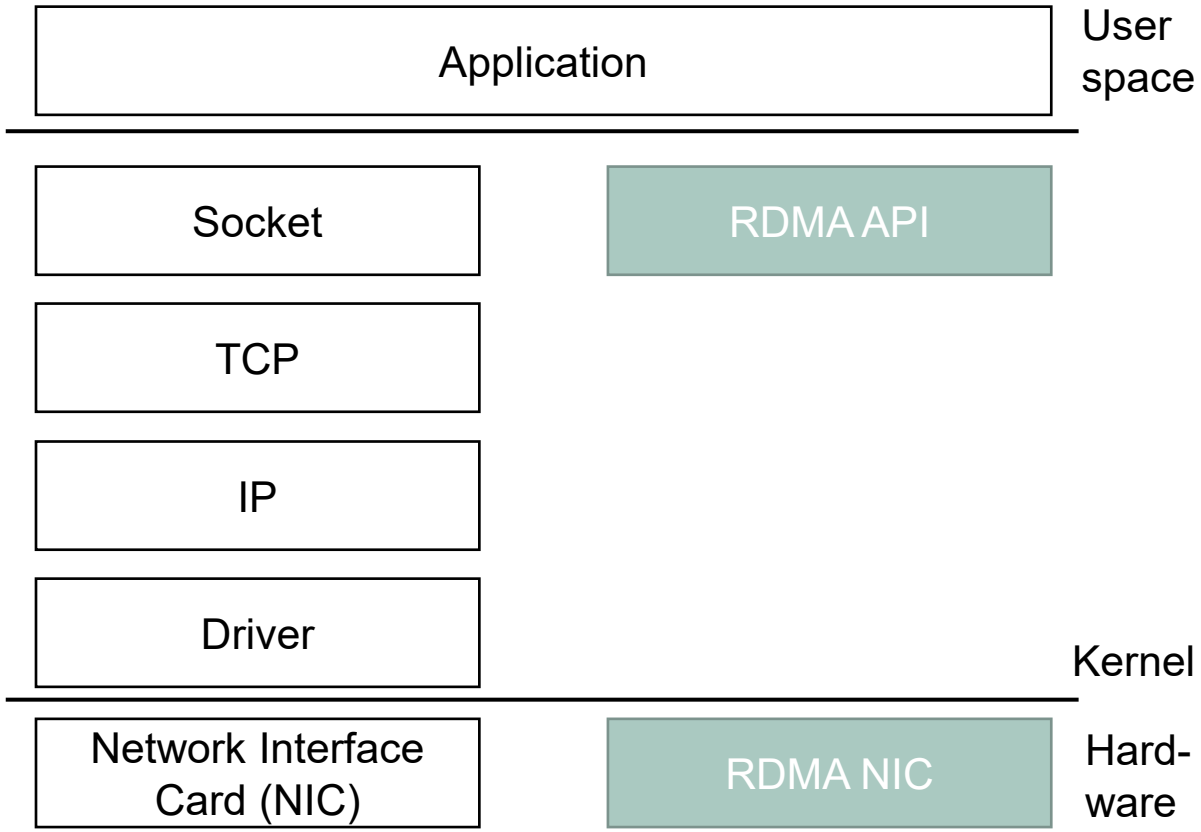


[HSLW24]

8.3.1 RDMA (Remote Direct Memory Access)

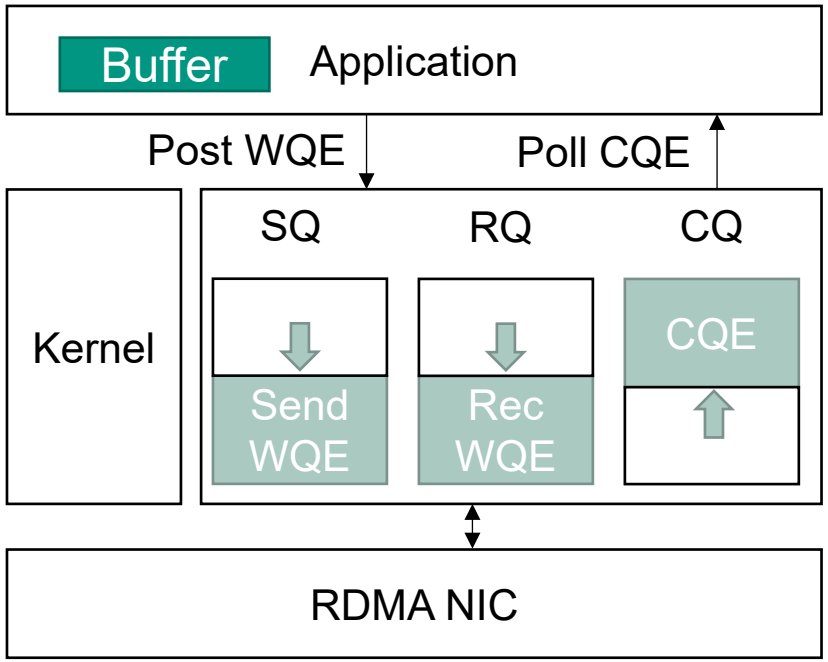
Socket vs. RDMA

- Sockets
  - TCP, IP, anddriver located in kernel space
  - Requires memory copies
- RDMA
  - Bypasses kernel processing
  - Uses directly RDMA API
  - Simple protocol processing on NIC
    - Simple Go-Back-N
      - → sensitive to packet losses
    - No congestion control provided
  - → reduced CPU overhead
  - → reduced context switch overhead
  - → reduced latency



RDMA Connection and Queue Pairs

- Establish RDMA connection
  - Three queues are created in RDMA NIC
    - Send Queue (SQ): tasks to be sent
    - Receive Queue (RQ): received tasks
    - Completion Queue (CQ): completed tasks
  - Register memory regions
- Application posts WQE to QP
  - It packages tasks into Work Queue Elements (WQE)
  - QP: Queue Pair (1 SQ and 1 RQ)
- RNIC completes work request, packages CQE, and puts CQE into CQ
- Application polls CQE



[HSLW24]

Example: App A Transmits Data to App B

1. RDMA NICs A and B create QPs, CQ, and register memory
   1. NIC A registers buffer for data of App A to be moved to App B
   2. NIC B allocates empty buffer for app B
2. App B creates WQE and posts it in RQ
   1. WQE contains pointer to empty buffer for the data
3. App A creates WQE and posts it in SQ
   1. WQE contains pointer to memory buffer to be moved to App B.
4. RDMA NIC A consumes WQE in SQ of App A.
   1. Data from memory region of App A begin to transmit to App B over a network
   2. Data arrives at RDMA NIC B. NIC B consumes WQE of App B´s RQ and learns memory location where to place the data
5. Data transmission completed
   1. CQE created in App B´s CQ. App B polls CQE from CQ
   2. … similar operation for App A

[GuLZ19]

RDMA over Converged Ethernet (RoCE)

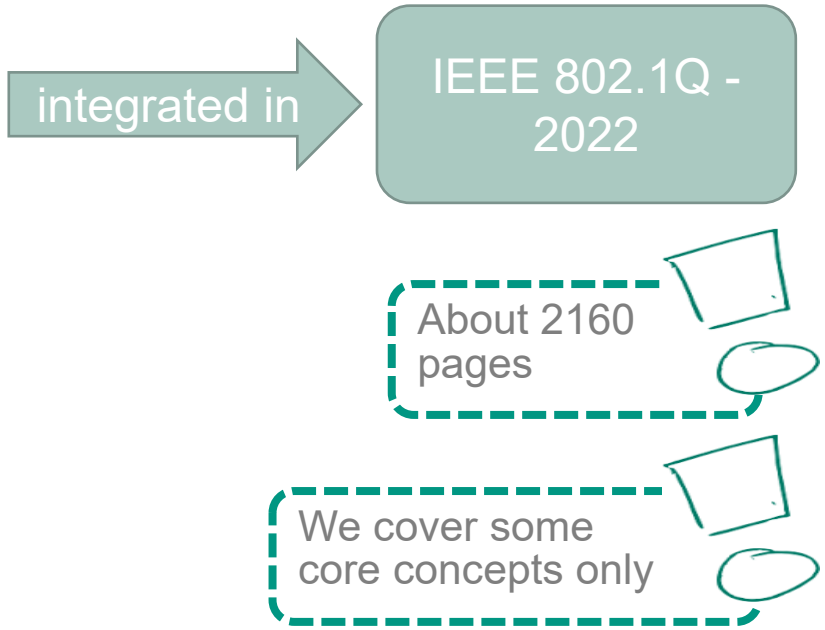
- Originally used in High Performance Computing
  - On lossless InfiniBand network

- In the context of data centers
  - Application of RDMA in Ethernet and IP networks → RoCE
    - Uses Ethernet, IP, and UDP
    - IP: enables layer-3-routing
    - UDP header used for ECMP routing (Equal-Cost-Multipath)
  - Applies Priority Flow Control (PFC) to achieve lossless Ethernet
    - Lossless: no packet losses due to buffer overflow

8.3.2 Converged Ethernet

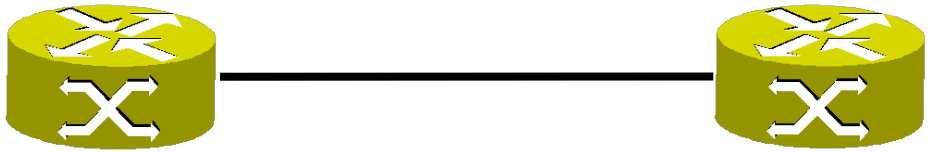
Converged Ethernet

- Extensions to Ethernet
  - Priority-based flow control (IEEE 802.1Qbb)
    - Link level flow control independent for each priority
  - Enhanced transmission selection (IEEE 802.1Qaz)
    - Assignment of bandwidth to traffic classes
  - …

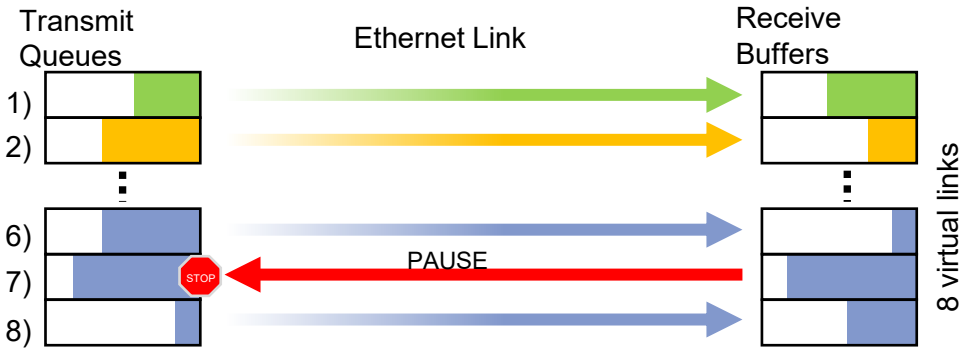


Priority-based Flow Control (PFC)

- Since Fast Ethernet: Simple flow control provided → PAUSE frame
  - All traffic on the corresponding port is paused
  - … and enabled back again



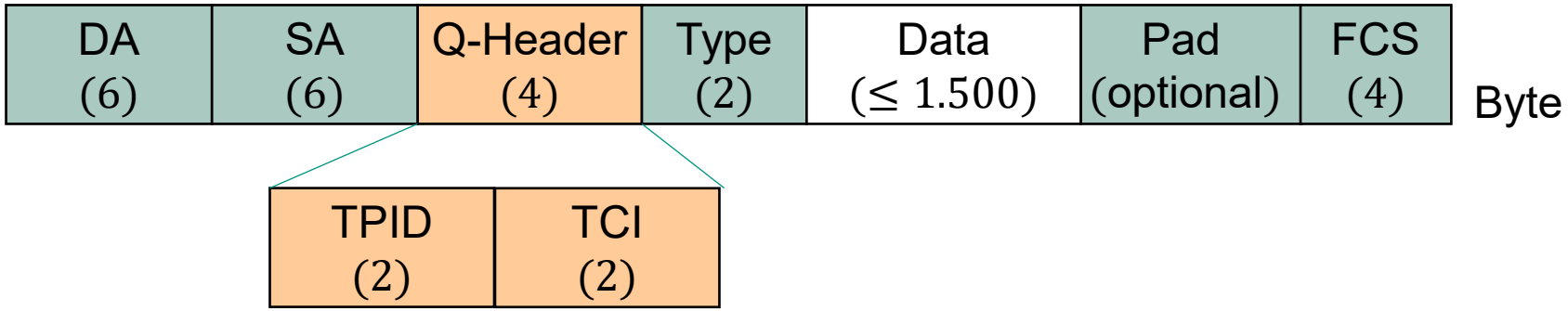
- With PFC: introduction of eight priority levels on one link
  - Use of VLAN identifier → Eight virtual links on a physical link
  - Strict priority scheduling for the queues
  - PFC works per port and priority
    - Individual flows are not addressed
  - PFC operates on a hop-by-hop basis



- Differentiated quality of service possible

Implementing Priorities with Ethernet

- Virtual LANs, standardized in IEEE 802.1Q
  - Introduction of a new field for VLAN tags: Q header



  - TPID: Tag Protocol Identifier
  - TCI: Tag Control Information
    - Priority Code Point (PCP) (3 bits): priority level of the frame
    - Drop Eligibile Indicator (1 bit)
    - VLAN Identifier (12 bits): VLAN to which the frame belongs … 4094 VLANs can be distinguished

- Differentiation of traffic according to priority chosen by Priority Code Point

PAUSE Frame

DA (6)	SA (6)	Type (2)	Opcode (2)	Parameters	Pad (optional)	FCS (4)

Byte

- MAC destination address: 01-80-C2-00-00-01
- EtherType: 88-08
  - MAC Control frame

- Opcode for PFC: 01-01
- 2 parameters
  - 2 Byte: priorities that should be paused
    - 0000 0000 XXXX XXXX
  - 8x 2 Byte pause duration
    - 2 Bytes per priority that should be paused
  - Pause time can be individually selected for each priority level

Pause Duration

- Duration given in “pause_quanta”
  - 2 Bytes → 0 to 65.535
- 1 pause quantum = 512 bit times
  - 1 bit time = time to transmit 1 bit
  - Pause duration dependent on line speed
- PFC only for short term pauses!
  - E.g. at 100 Gbit/s **at most** ~335 µs
- Pause until timer finishes
  - Or manual resume via PAUSE Frame with duration 0

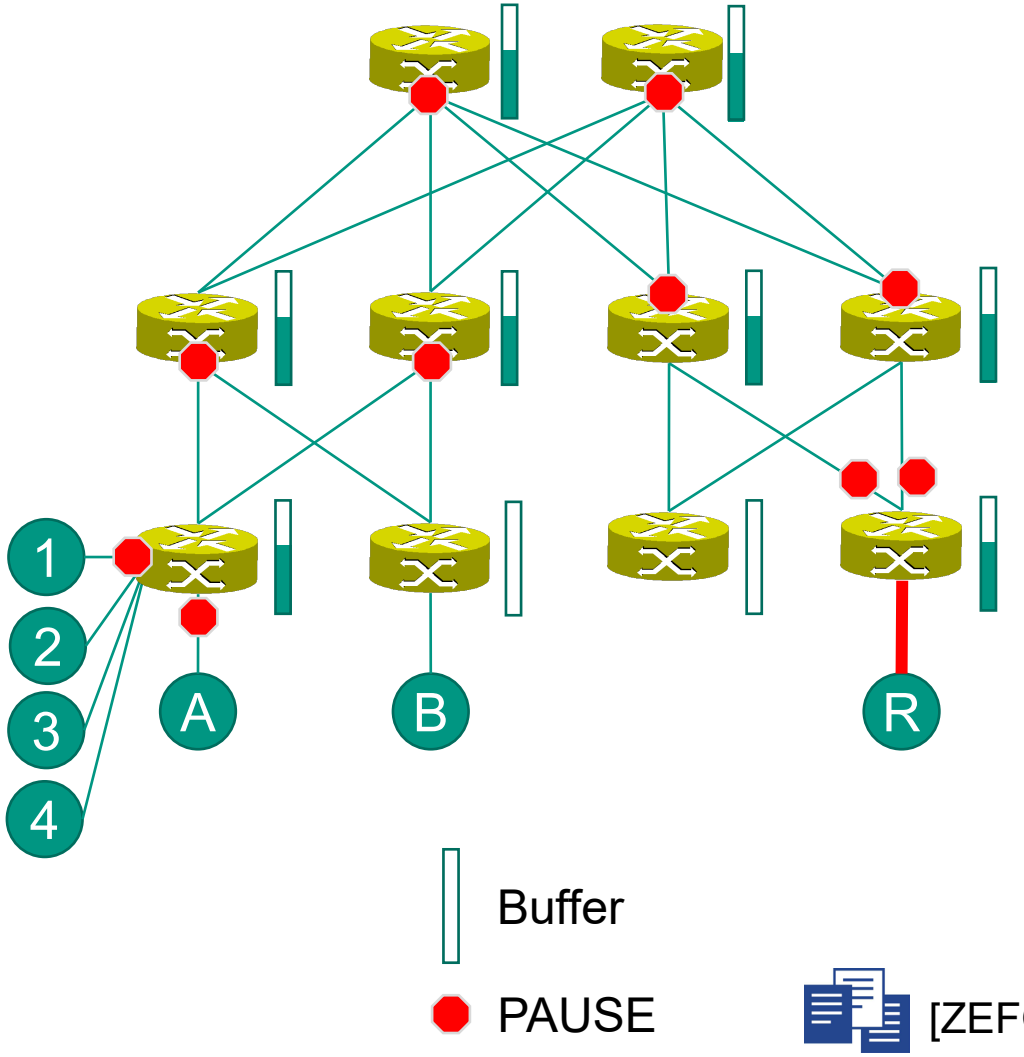
Speed	Pause quantum
100 Mbit/s	5,12 µs
1 Gbit/s	512 ns
100 Gbit/s	5,12 ns

Issues Related to PFC

- Recall: PFC works on ports and priorities
  - Not on individual flows
- PFC can cause
  - Head-of-Line (HoL) blocking
  - Victim flows
  - Cascading PAUSE frames
  - Deadlocks
- Memory requirements
  - In order to prevent packet losses free headroom in the buffer is needed
    - Buffer for packets that can be transmitted until PAUSE frame is received

Example: Victim Flows and Cascading PAUSEs

- Senders 1-4 send to receiver R
  - All senders are connected to same switch
  - Flows spread among multiple paths
    - Shared link of all paths: link to R
      - → Bottleneck at port to R
- Sender A sends to receiver B
  - Link to R not shared by flow A →B
  - However: Flow A→B inhibited by PAUSEs caused by other flows
  - Flow A→B is a victim flow



[ZEFG15]

PFC Deadlocks

- Normal behaviour
  - Packets from sender forwarded up the tree
  - Until common ancestor with receiver
  - Then back down to receiver
  - No cyclic dependency

- Interaction between PFC and flooding
  - Unknown destination MAC address
    - E.g. because server died
  - Packet gets flooded on all other ports
  - Can cause cyclic buffer dependency
    - Deadlock can occur

[Guo16]

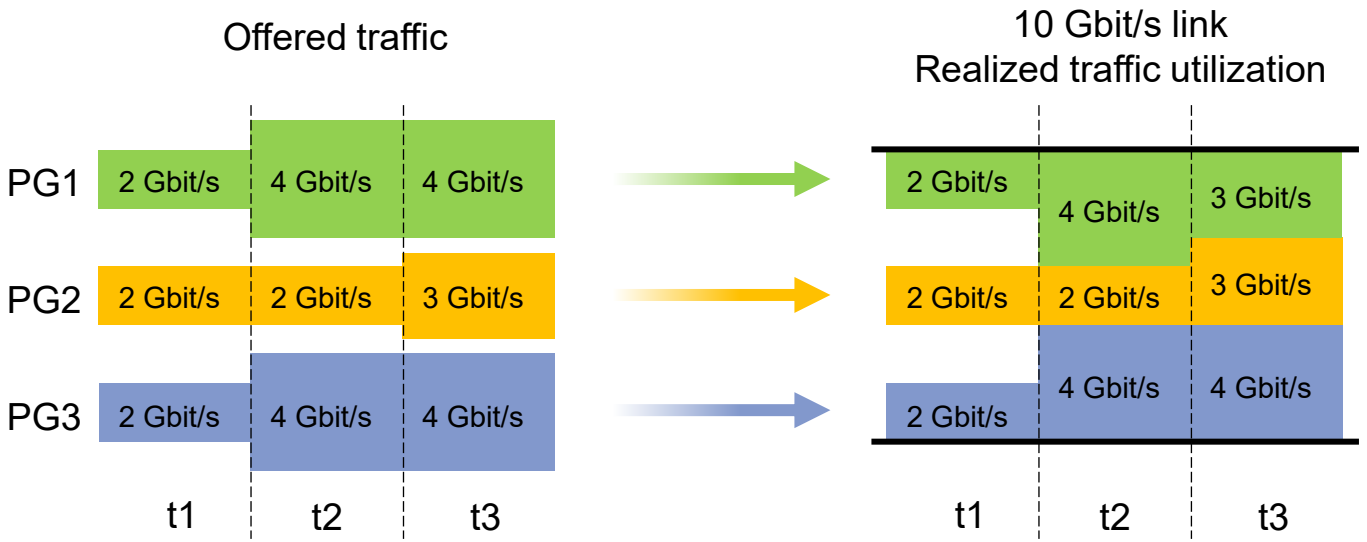
Enhanced Transmission Selection (ETS)

- Objective
  - Link bandwidth allocation for different types of traffic - differentiation
- Up to now
  - 8 priority levels
  - 8 queues within a switch – strict priority scheduling
- Introduction of priority groups (PGs)
  - Reservation of bandwidth for priority groups
    - Guarantees a minimum data rate per priority group
    - Unused capacity usable by other priority groups
  - Can contain one or more of the eight priority levels
    - Distributes assigned bandwidth among the priorities in the group
  - Different virtual queues in the network interface

[CYZY13]

Example

- Priority groups PG1, PG2, and PG3 use a single link with 10 Gbit/s capacity
- Reservations
  - PG1: 2 Gbit/s
  - PG2: 4 Gbit/s
  - PG3: 4 Gbit/s
- Time intervals t1 and t2
  - Sufficient capacity available to serve requests of all PGs
- Time interval t3
  - Total of 11 Gbit/s requested
    → PG1 is reduced to 3 Gbit/s, as only 2 Gbit/s was reserved



[CYZY13]

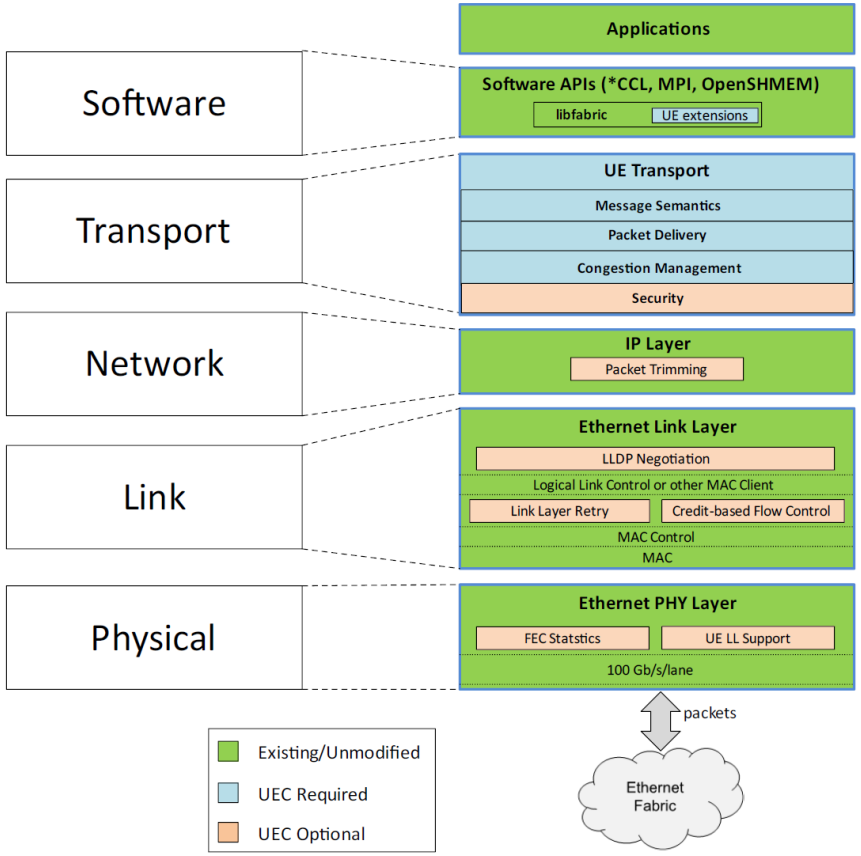
8.3.3 Ultra Ethernet

Ultra-Ethernet

- Designed by Ultra Ethernet Consortium
  - Specification 1.0 published June 2025
  - „As performant as a supercomputing interconnect
  - As ubiquitous and cost-effective as Ethernet
  - As scalable as a cloud data center“
- Mission
  - Deliver an Ethernet based open, interoperable, high performance, full-communications stack architecture to meet the growing network demands of AI & HPC at scale
- Core changes
  - Not in the „classical“ Ethernet protocol
  - But: Ultra Ethernet Transport (UET) – a new transport protocol
  - Improvements at network layer

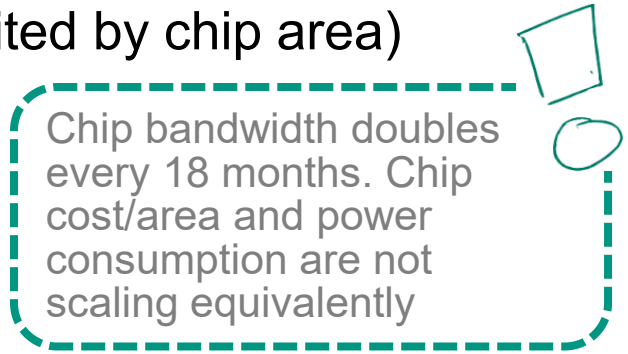
Ultra Ethernet Protocol Stack

- Advances on multiple layers



New Network Features

- Packet trimming
  - In case switch buffers are full (recall: buffers are in SRAM, size is limited by chip area)
    - Trim packet payload – signalled through DSCP bits in the IP header
    - Place remaining packet header in high priority queue
  - Enables rapid retransmissions
  - Congestin signal



- Link-layer retry
  - Retransmissions at link layer
  - Adds sequence number to packets, frames are acked (NACK)

Ultra Ethernet Transport

- Replacement for RDMA
- Packet spraying
  - Uses many/all paths to destination
    - More balanced use of all network paths (compared to flow-based multipathing)
    - Minimizes hotspots and congestion on links inside network
    - Incast-based congestion possible on last link to the receiver
      - Goal: fairly share final link without resulting in expensive packet loss, retransmission, or increased
- Trim-NACK, selective ACK, selective fast retransmission
- Congestion control mechanisms
  - Network Signal Congestion Control
    - Congestion window at source
      - Single window for multipath spraying
    - Relies on ACK-clocking
  - Receiver-Credit Congestion Control
    - Sender maintains pool of credit
    - Receives credit from destination
- Different packet delivery modes, e.g.
  - Reliable, ordered
    - Selective retransmissions
  - Reliable, unordered
    - Go-Back-N
- 0-RTT connection setup
- End to end encryption

Ultra Ethernet vs. RDMA

RDMA	Ultra Ethernet
Required in-order delivery, Go-Back-N recovery	Out-of-order packet delivery with in-order message completion
Security external to specification	Build-in high-scale modern security
Flow-level multi-pathing (ECMP)	Packet spraying (packet-level multi-pathing)
DCTCP or others	Sender- and receiver-based congestion control
Scale to low tens of thousands of simultaneous endpoints	Targeting scale of 1 million simultaneous endpoints

8.3.3 Congestion Control

Congestion Control in Data Centers

- Goal
  - Reduce tail latency caused by queuing
  - Avoid usage of PAUSE frames as much as possible
- Challenges
  - Timeliness
    - Many flows in data center finish in one RTT
  - Accuracy
    - Which congestion signals to choose?
    - How to adapt the rate?
- Constraint
  - Light-weight implementation on network interface

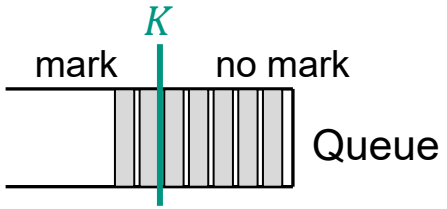
Data Center TCP (DCTCP)

- Goal
  - Achieve high burst tolerance, low latencies and high throughput with shallow-buffered commodity switches
- Property
  - DCTCP works with low utilization of queues without reducing throughput
- How does DCTCP achieve its goal?
  - Responds to strength of congestion and not to its presence
    - Response based on presence would be too strong in data centers
      → buffer underflow, low throughput
  - DCTCP
    - Modifies ECN
    - Estimates fraction of bytes that encountered congestion
    - Scales TCP congestion window based on this estimate

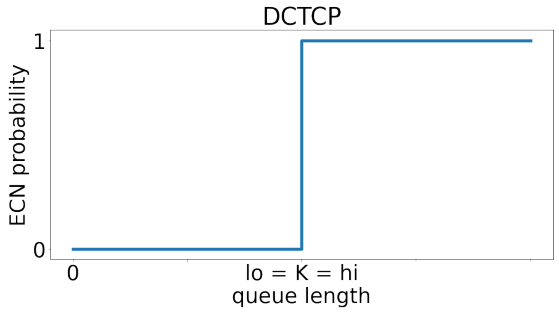

[Aliz10, RFC8257]

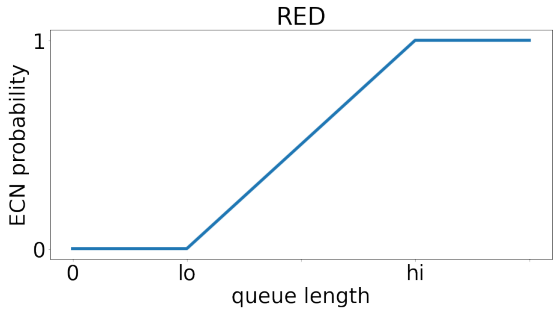
Modified explicit congestion notification (ECN)

- Very simple active queue management
  - A single parameter: threshold K
    - If $\#elements\ in\ queue > K$: set CE codepoint
    - Marking based on instantaneous rather than average queue length
  - Suggestion: $K > (RTT \cdot C)/7$, with C: data rate in packets/s



- ECN probability with DCTCP or classical RED





RED implementation in switches can be used. High and low thresholds are both set to K

Sender and receiver should not make any assumption about the marking algorithm in the switch since this is typically not known

[Aliz10]

ECN Echo at Receiver

- New boolean TCP state variable
  - DCTCP Congestion Encountered (DCTCP.CE)
- Receiving segments
  - If CE codepoint is set and DCTCP.CE is false
    - Set DCTCP.CE to true
    - Send an immediate ACK
  - If CE codepoint is not set and DCTCP.CE is true
    - Set DCTCP.CE to false
    - Send an immediate ACK
  - Otherwise
    - Ignore CE codepoint
- Immediate ACK if marked packets start/stop to appear
  - Cumulative ACKs otherwise
  - No individual ACKs needed in order to convey fraction of marked packets

[RFC8257]

Fraction of Bytes Encountered Congestion

- Controller at sender
  - Estimates fraction of bytes sent that encountered congestion (DCTCP.Alpha)
    - Initialized to 1, updated as follows
    - $DCTCP.Alpha = (1 - g) \cdot DCTCP.Alpha + g \cdot M$
      - g : estimation gain $(0 < g < 1)$
      - M : fraction of bytes sent that encountered congestion during previous observation window (approximately RTT)
      - $\mathrm{M} = \frac{\#\ \text{marked}\ bytes}{\#\ \text{Bytes acked (total)}}$

[RFC8257]

Scales Congestion Window Based on Estimate

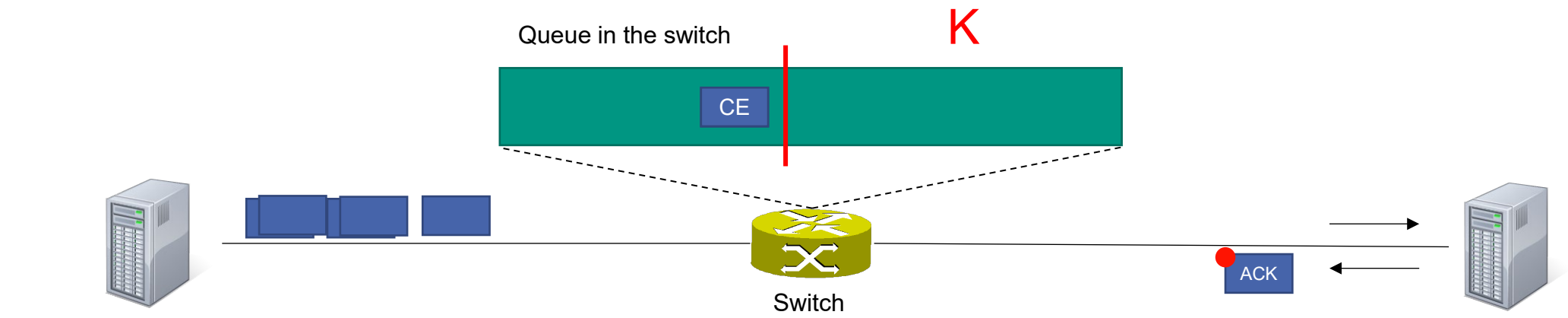
- Controller at sender
  - Updates congestion window in case of congestion
    - $CWnd = (1 - DCTCP.Alpha/2) \cdot CWnd$

    → if $DCTCP.Alpha$ close to 0, $CWnd$ is only slightly reduced

    → if $DCTCP.Alpha == 1$, $CWnd$ is reduced by factor 2
  - Handling of congestion window growth
    - As in conventional TCP
  - Apply as usual
    - Slow start, additive increase, recovery from lost packets

[RFC8257]

DCTCP



$\frac{\#\ Marked\ bytes}{\#\ Bytes\ acked}$ used for calculating new DCTCP.ALPHA

DCTCP.ALPHA used for calculating new CWnd

Check: CE == DCTCP.CE

false: Update DCTCP.CE to CE

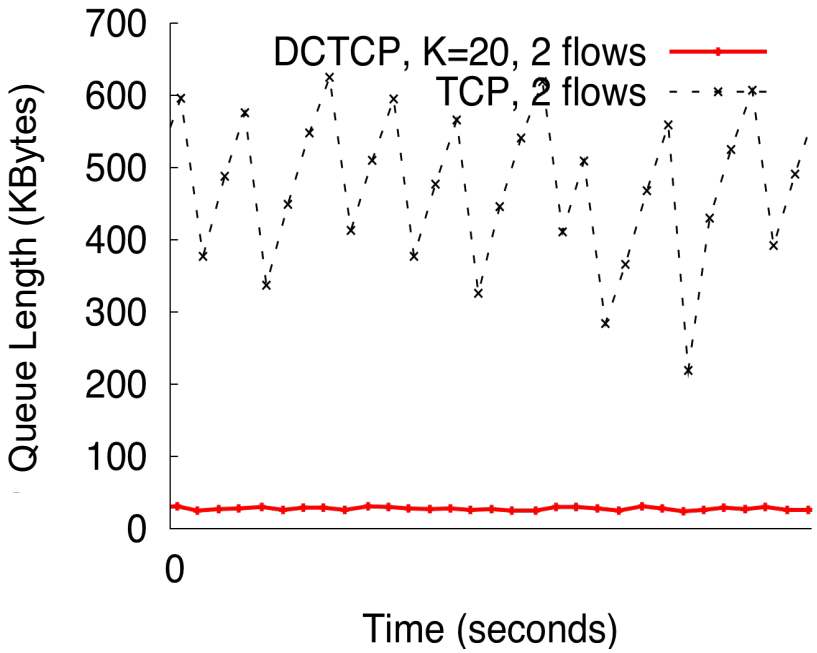
true: Done

Send immediate ACK

Set ● if CE

Queue Length

- Investigated scenario
  - Two data streams from different 1 Gbit/s ports to one 1 Gbit/s port
    - Queue can allocate up to 700 kByte buffer dynamically



→ Significantly reduced queue length in the switch

[Aliz10]

Configuration of DCTCP

- DCTCP only applicable for communication inside a data center
  - Both communication end points need to be located in same data center
- Usage of DCTCP needs to be configured
  - No negotiation mechanism provided within RFC 8257
  - Possible solutions
    - Based on IP address of remote end point
    - Dependend on estimated RTT

DCTCP Deployment?

- 2018: changed from CUBIC to DCTCP
  - Relatively simple, mature, and well-established protocol
    - Had been added to Linux Kernel since 2014
    - Wide-spread support for ECN marking
- … sounds easy, but paper reports on many practical issues
- Usage
  - DCTCP for in-region connections
  - CUBIC for inter-region connections
- Some interesting aspects mentioned
  - → how to deal with both concurrently?
  - → how to isolate them?
  - „Many connections in our networks are long, running for days or longer“
  - Limitation of DCTCP: short and heavy incast bursts
    - Heavy of incast means that a CWnd of 1, the lowest DCTCP can maintain, is too high

[A. Dhamija et al; A large-scale deployment of DCTCP;
21st USENIX Symposium on Networked Systems Design and Implementation;
Santa Clara, USA, 2024]

Overview of presented Congestion Controls

Chapter 7

TCP Tahoe
Historical algorithm, first congestion control

TCP Reno
Improvement of TCP Tahoe

Introduce general „mechanics“ behind Internet congestion control

Chapter 8

DCTCP
Congestion control for data centers

Adresses specifics of congestion control in data center networks

Chapter 9

CUBIC TCP
Improved congestion control for networks with high BDP

BBR TCP
Rate-based congestion control introduced by Google

8.5 Ultra Large Scale Data Centers

Ultra Large Scale Data Centers

- Typically have
  - Large number of compute servers with virtual machine support
  - Extensive storage facilities
  - High portion of east-west-traffic (server-to-server traffic)

- Typically use
  - Off-the-shelf commodity hardware devices
    - Huge amount: several 100.000 servers
    - Several 1000 switches
      - Switches with small buffers
  - Commodity protocols

- Need to be highly performant
  - Load of 100 Tbit/s and more
    - Support of multipath routing (east-west-traffic)
  - Very low latencies required (< 10 µs per hop)

Ultra Large Scale Data Centers

- Should be extensible without massive reorganization
  - Topology that scales horizontally
  - Possibility to add links and devices without need to upgrade switches
  - Stable addressing scheme
- Entire data center can not be upgraded at the same time
  - CapEx
  - Non-tolerable disruption
- Need to be reliable (at the application service level)
  - Requires adequate redundancy
  - Span of failure from protocol or equipment is contained
- Limited set of features and supported by multiple vendors
  - Routing protocol that is easily implementable
  - Traffic engineering supported by available protocols

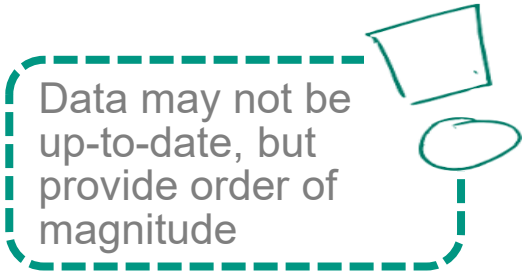
[RFC7938]

Some Numbers

- Amazon
  - 49 availability zones over 18 geographical regions (1-2 ms “distance”)
    - Clusters of data centers within a region (to avoid single point of failure)
  - 25-32 MW of power consumption per data center
    - Enough power for a city with a population of 10-20k
  - 50k – 80k physical servers per data center (typically)
  - Custom build routers and servers

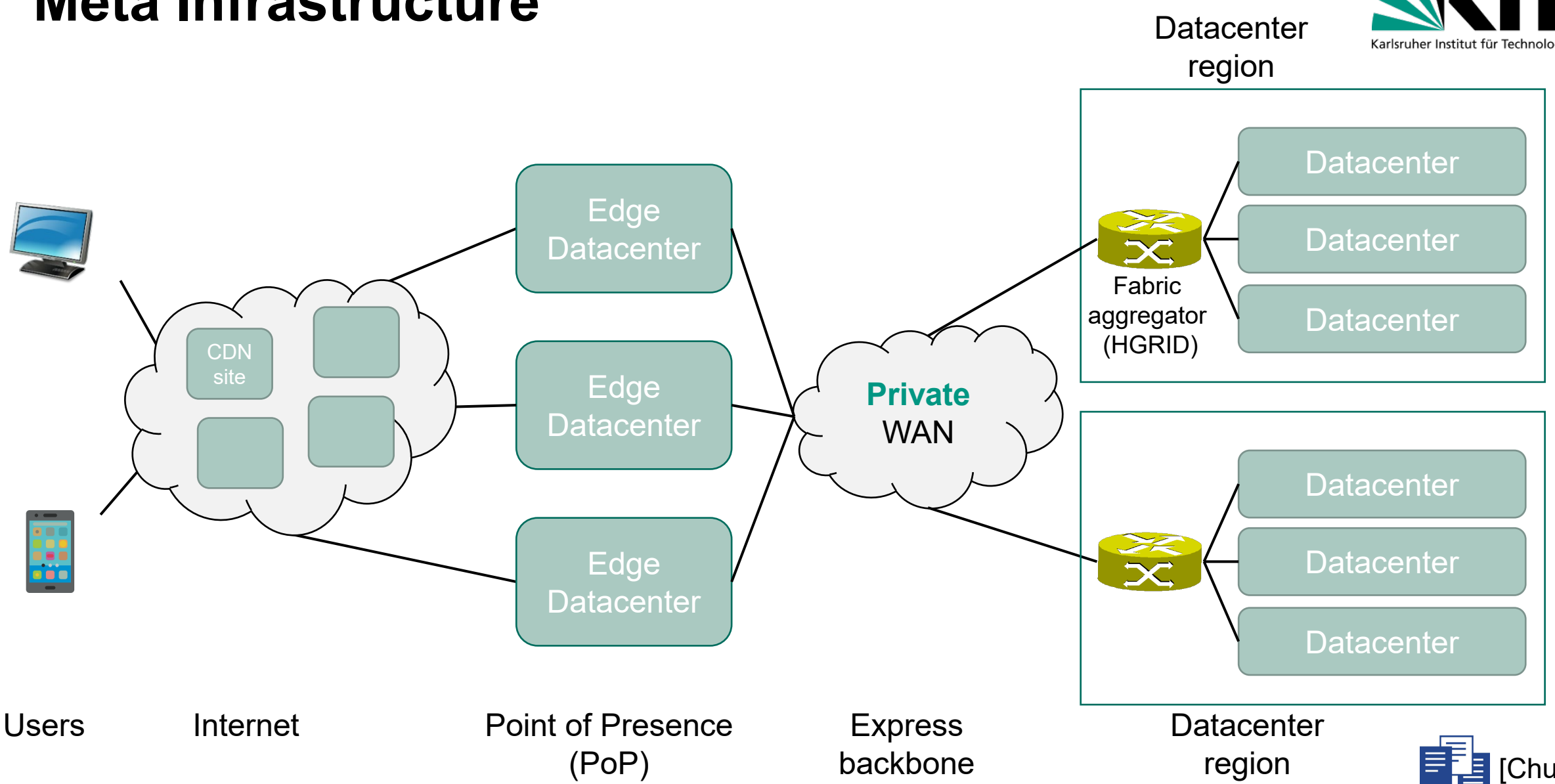
- Microsoft
  - 100+ data centers
  - More than 1 million servers overall
  - More than 1.5 million network requests per second

- Google cloud
  - 19 data centers on four continents
  - Several million servers



[KuRo22]

Meta Infrastructure



[Chun25]

Number and Sizes of Infrastructure Components

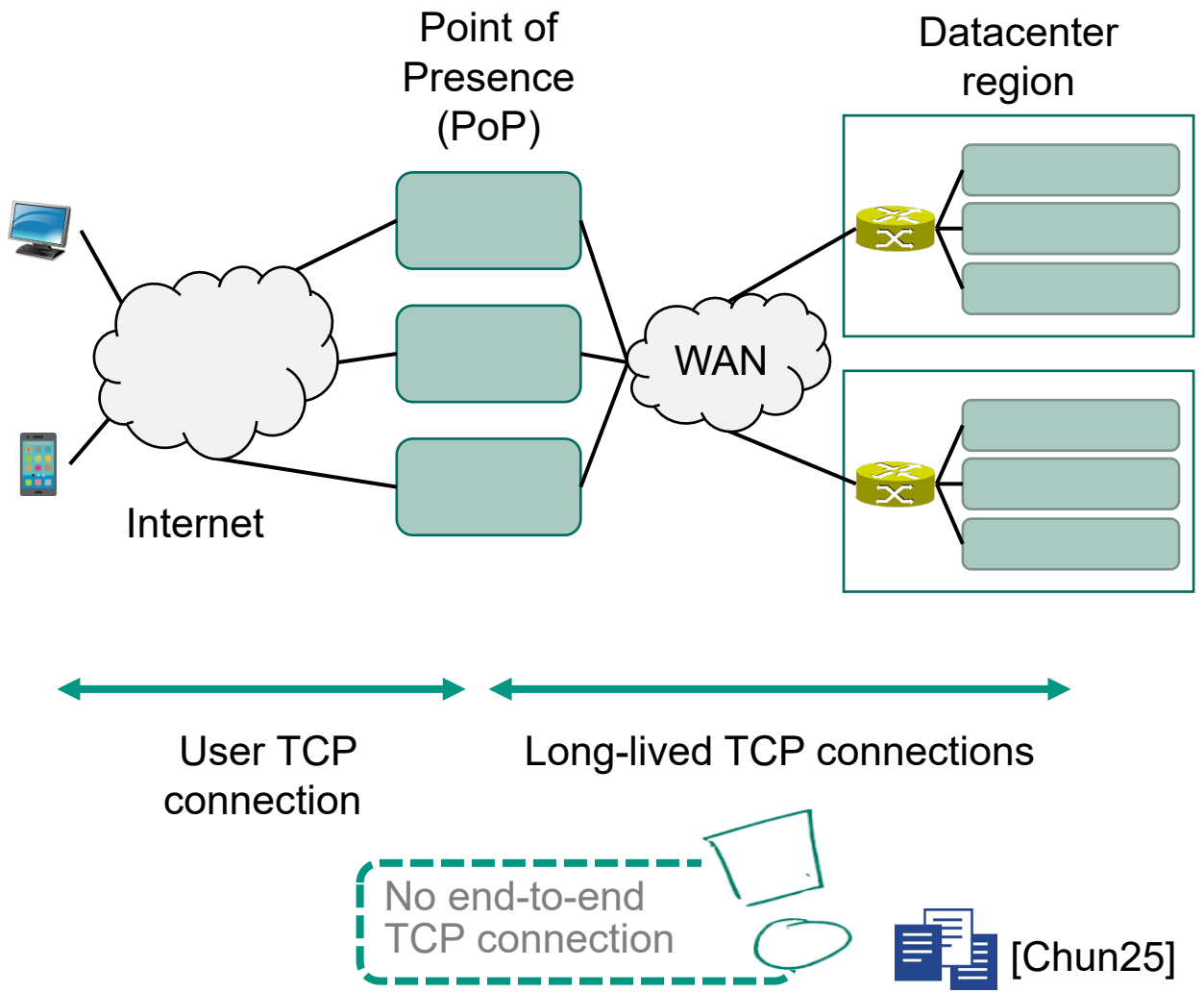
Entity type	Entity count	Servers in each entity
Region	O(10)	Up to one million
PoP	O(100)	Typically O(100) but up to O(1000)
CDN site	O(1000)	Typically O(10) but up to 100+
Datacenter	Multiple data centers per region	O(100000)

- Datacenter region
  - Multiple datacenters within radius of a few miles
- Datacenter
  - Servers are interconnected by a datacenter fabric
  - Switches form a three-level Clos topology
  - With sufficient top-level switches
    - Non-blocking and non-oversubscribed network can be provided
    - → enables communication between any two servers at their full NIC bandwidth
- Regional network
  - Fabric aggregator connects datacenters
  - … and connects them to private WAN
  - Topology of fabric aggregator
    - Similar to fat tree

[Chun25]

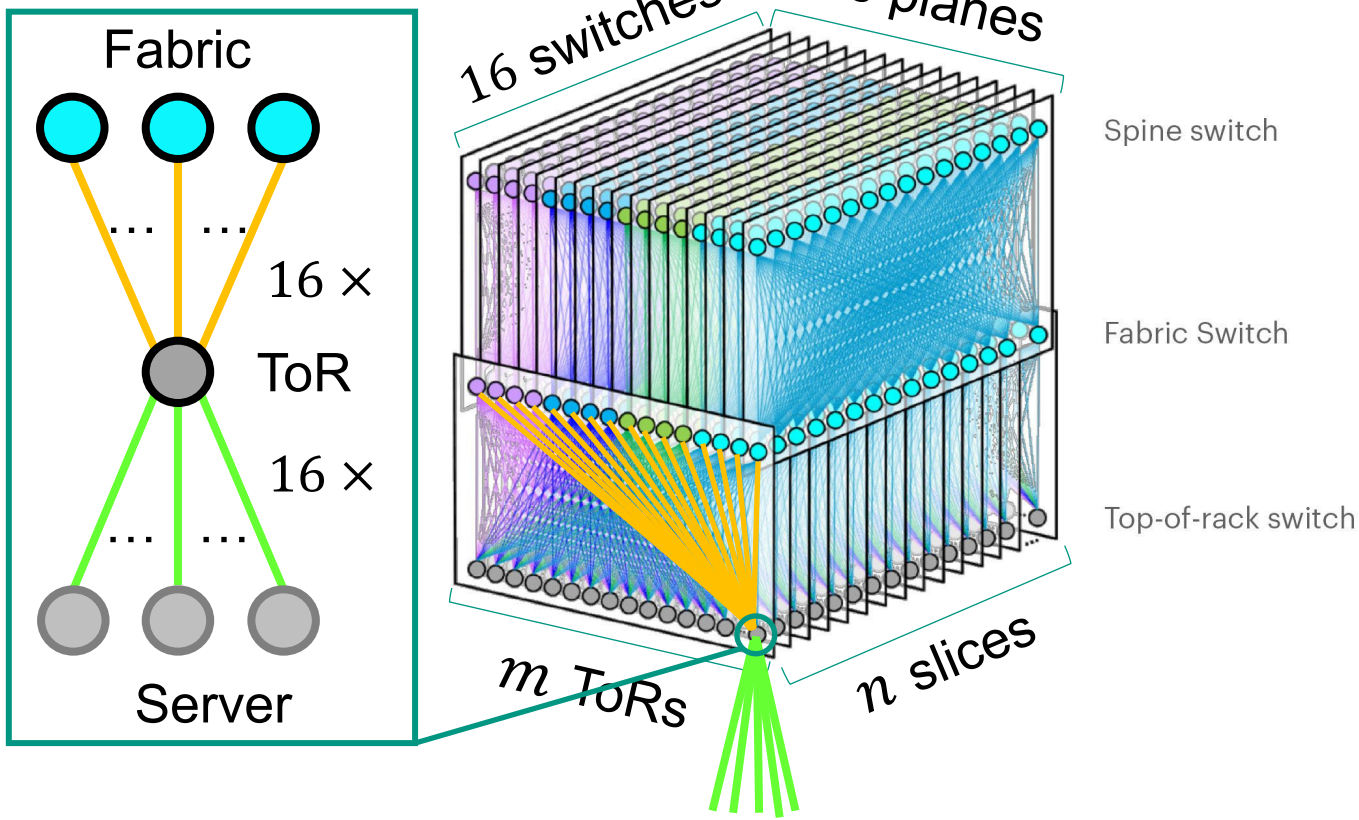
Some Facts

- Split-TCP
  - User establishes TCP connection to PoP
  - PoP uses already established TCP connection to datacenter
    - Reduces delay incurred by connection establishment
- Centralized network control in private WAN
  - Centralized routing
  - Centralized traffic engineering, no RSVP-TE
- Traffic among datacenters in private WAN
  - Orders of magnitude larger than external-facing traffic between users and PoPs



Facebook F16 Data Center Network Topology

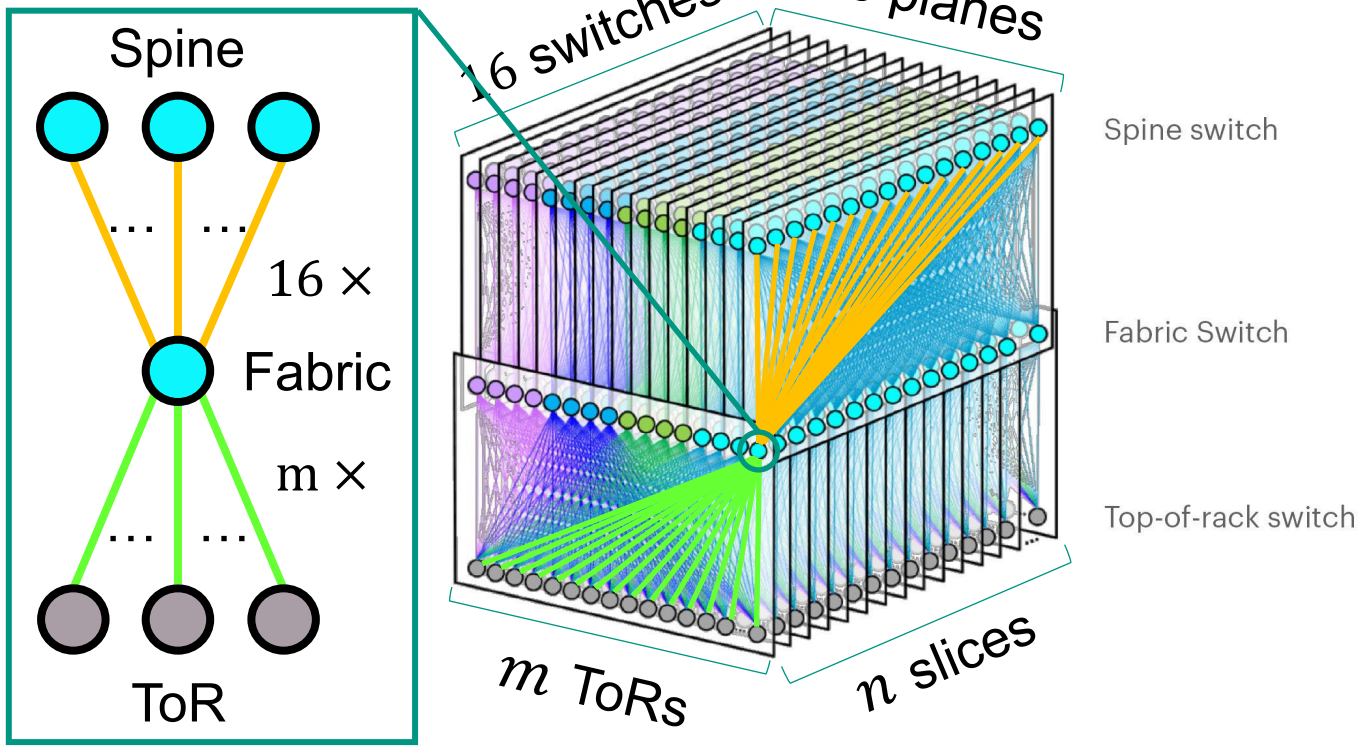
- Each ToR switch connected to 16 different planes (fabric switch)
  - 1.6 Tbit/s uplink and downlink capacity



[AnWE19]

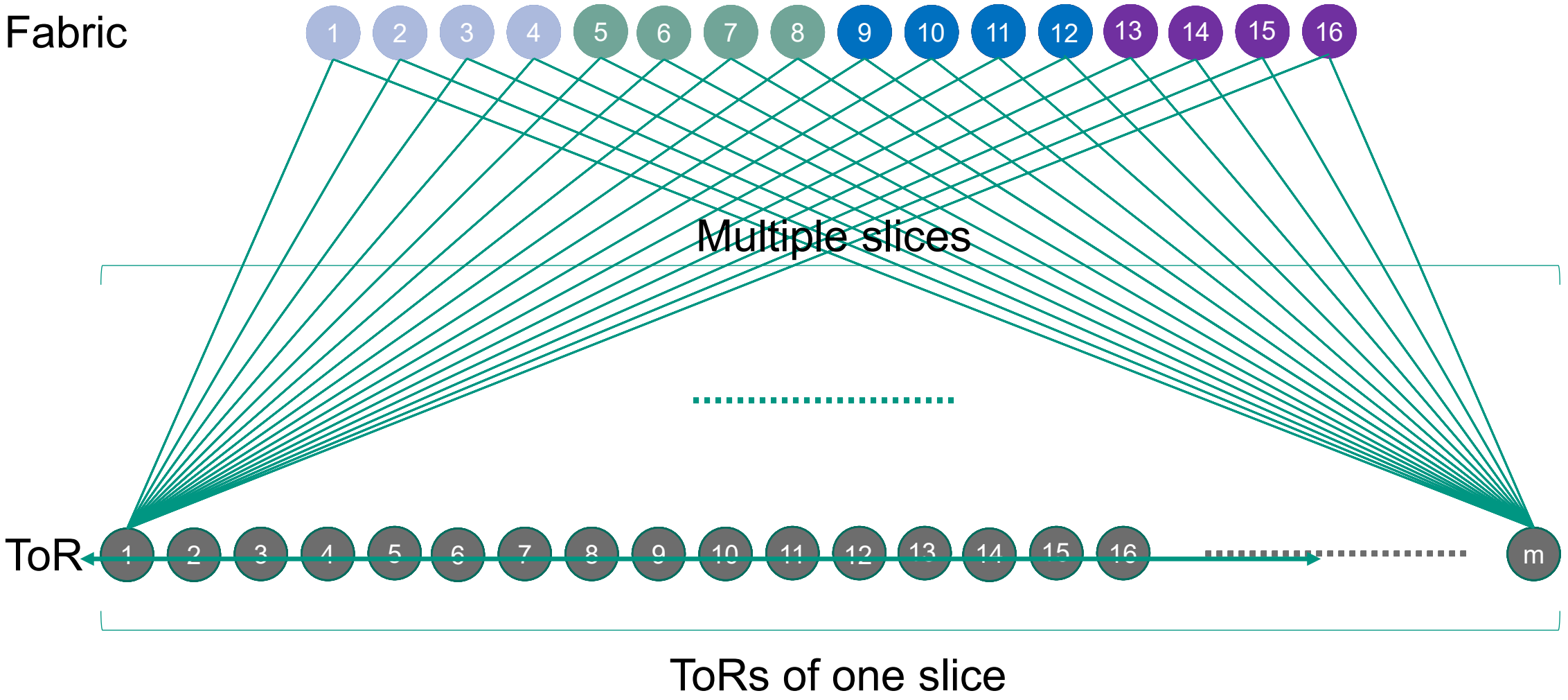
Facebook F16 Data Center Network Topology

- Each ToR switch connected to 16 different planes (fabric switch)
  - 1.6 Tbit/s uplink and downlink capacity
- Each Fabric switch connected to 16 Spine switches
- Fabric and Spine Switches have 128 ports
  - Up to 128−16=112 ToR switches per Fabric switch
  - Up to 112 Fabric switches per Spine switch

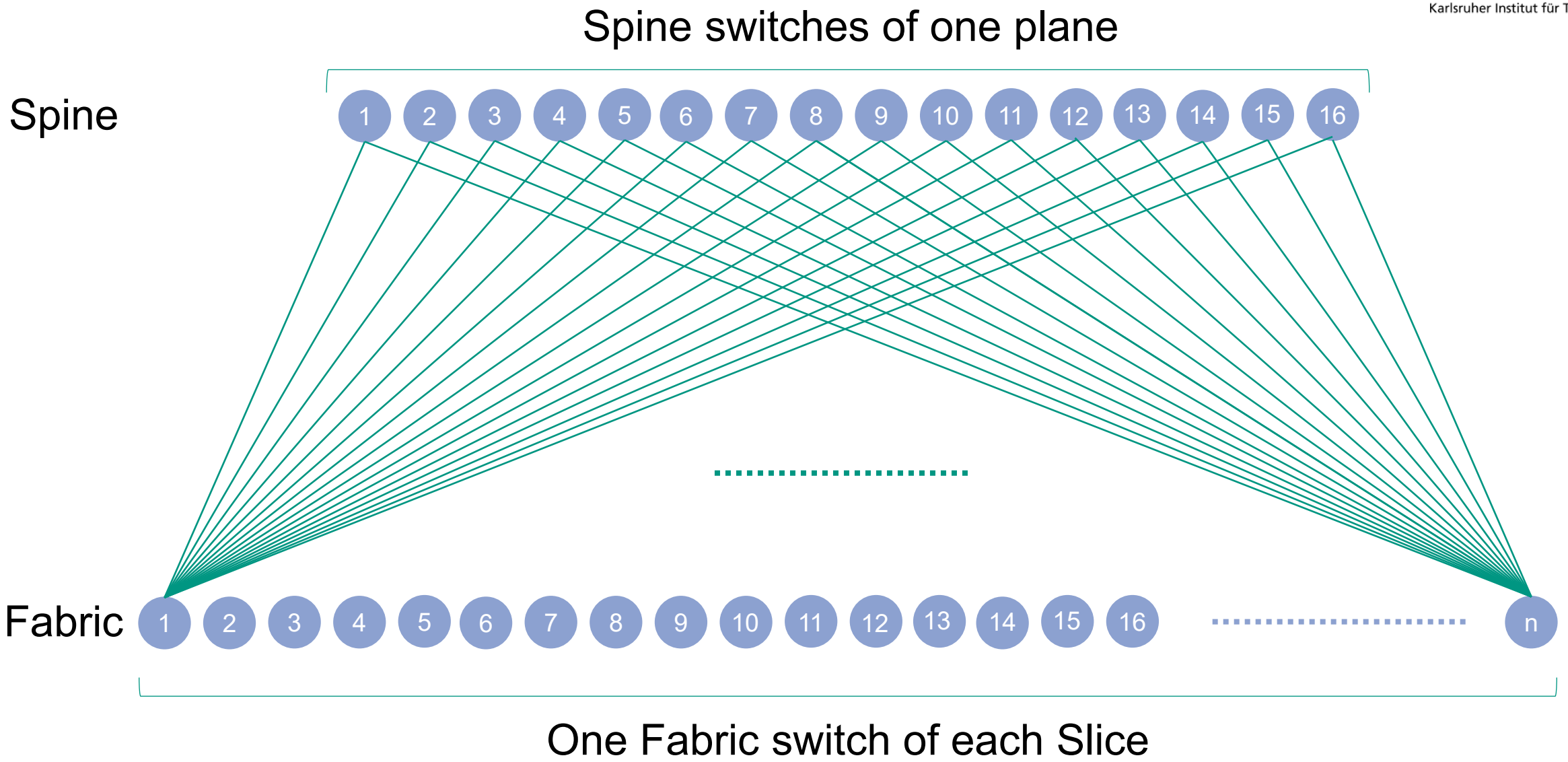


[AnWE19]

Facebook F16 Slice

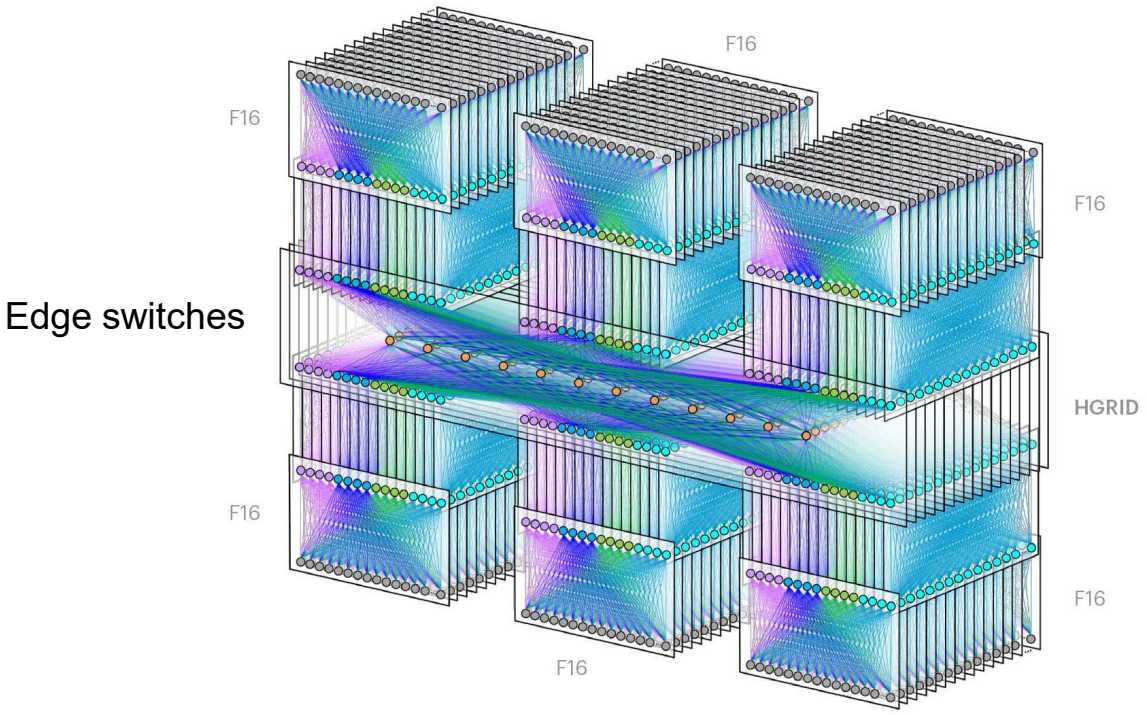


Facebook F16 Plane



HGRID: Fabric Aggregator

- Task
  - Interconnect datacenter region
  - Up to 6 buildings can be connected
- Interconnection among Spine switches
  - Each Spine switch connected to 12 Edge switches
- Path length
  - Network paths in same fabric have always 6 hops (best case)
  - Building to building path always take 8 hops



[AnWE19]

Network and Routing/Forwarding

- Different routing designs, e.g.,
  - Decentralized BGP-based, e.g.,
    - RFC 7938
    - Facebook
    - Microsoft
  - Centralized SDN-based, e.g.,
    - Google


[RFC7938]

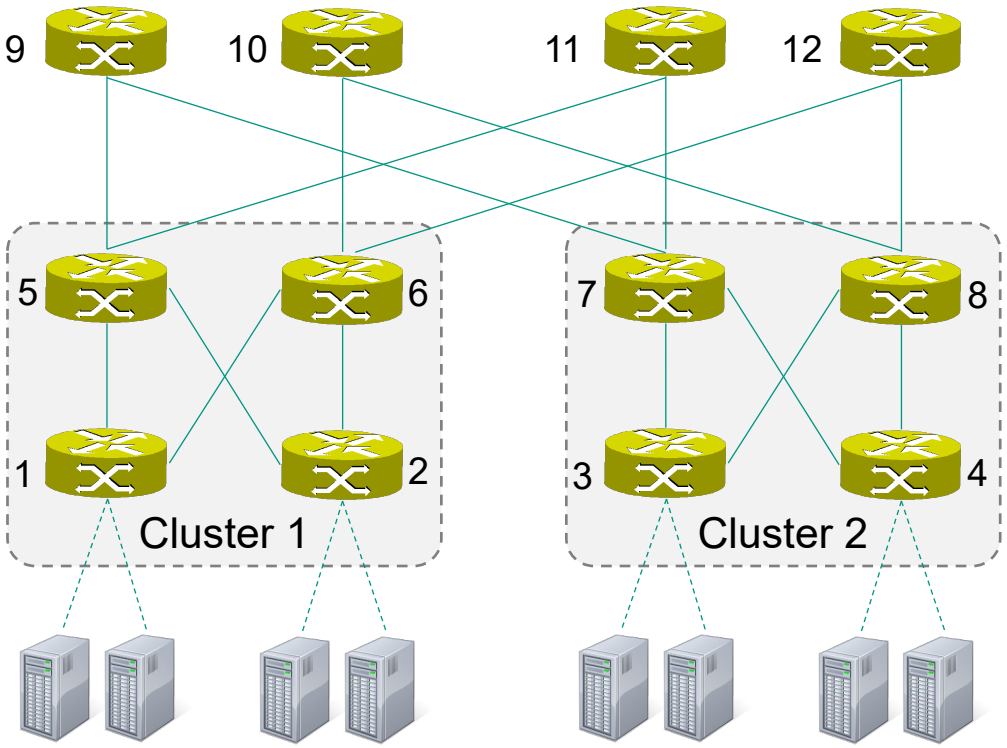

[Abha21]


[NkLK14]


[Pout22]

8.5.1 Sketch on BGP-based Design by Microsoft

5-Stage Clos Topology



Spine

Leaf

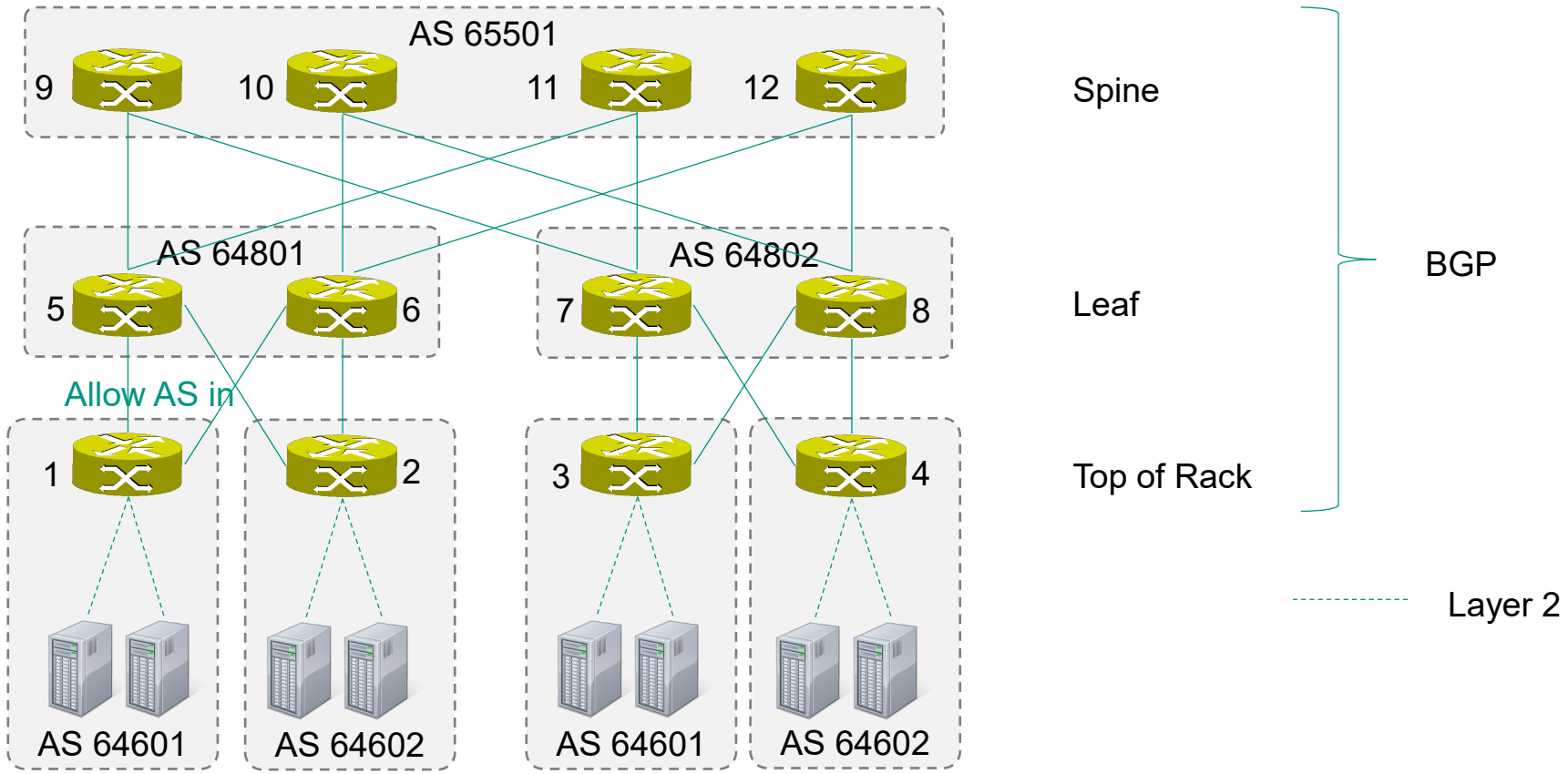
Top of Rack

Layer 3

Layer 2

- Three levels
  - Spline, leaf, top of rack
- Five stages
  - ToR – Leaf – Spline – Leaf - ToR

Routing Design with eBGP



- BGP used down to the ToR level
- Usage of 16 bit private AS numbers … but only 1022 available
- Same AS numbers are used in different data centers

Routing Design with eBGP

- All switches act as BGP speaker
  - BGP sessions have to be established between them
- Separate BGP ASN per ToR switch
- AS numbers are reused at ToR level
  - E.g., AS 64601 at switch 1 and switch 3
  - However, repeated AS numbers in BGP path indicate loops
  - Feature called „Allow AS in“ needed
- BGP announcements leaving the data center
  - Private AS number has to be replaced with public AS number of data center

LITERATURE

Literature

[Abha21] A. Abhashkumar et al.; Running BGP in Data Centers at Scale; 18th USENIX Symposium on Networked Systems Design and Implementation, April 2021

[Aliz10] M. Alizadeh et al.; Data Center TCP (DCTCP); New Delhi, ACM SIGCOMM, August 2010

[AlLV08] M. Al-Fares, A. Loukissas, A. Vahdat; A scalable, commodity data center network architecture; ACM SIGCOMM, 2008

[AnWE19] Alexey Andreyev, Xu Wang, Alex Eckert; Reinventing Facebook's data center network; https://engineering.fb.com/2019/03/14/data-center-engineering/f16-minipack/

[Chun25] Chunqiang Tang; Meta's Hyperscale Infrastructure: Overview and Insights; Communications of the ACM Volume 68 Issue 2; January 2025

[CYZY13] Y. Cai, Y. Yan, Z. Zhang, Y. Yang; Survey on Converged Data Center Networks with DCB and FCoE: Standards and Protocols; IEEE Network, Juli/August 2013, pp. 27-32

[GuLZ19] Z. Guo, S. Liu, Z. Zhang; Traffic Control for RDMA-Enabled Data Center Networks: A Survey; IEEE Systems Journal, 2019

[Guo16] C. Guo et al.; RDMA over Commodity Ethernet at Scale; ACM SIGCOMM, August 2016

[HoHR22] T. Hoefler, A. Hendel, D. Roweth; The Convergence of Hyperscale Datacenter and High-performance Computing Networks; IEEE Computer, July 2022; https://htor.inf.ethz.ch/publications/img/hpc-vs-dc-networking.pdf

Literature

[HSLW24] J. Hu, H. Shen, X. Liu, J. Wang; RDMA Transport in Datacenter Networks: Survey; IEEE Network, November/December 2024

[KuRo22] J.F. Kurose, K.W. Ross; Computer Networking – A Top-Down Approach; Addison Wesley, 8th Edition, 2022

- 1.2 Network Edge, 6.6 Data Center Networking

[NkLK14] E. Nkposong, T. LaBerge, N. Kitajima; Experiences with BGP in Large Scale Data Centers: Teaching an old protocol new tricks; https://www.janog.gr.jp/meeting/janog33/doc/janog33-bgp-nkposong-1.pdf

[Pout22] L. Poutievski et al.; Jupiter evolving: transforming google's datacenter network via optical circuit switches and software-defined networking; ACM SIGCOMM, August 2022

[RFC7938] P. Lapukhov et al.; Use of BGP for Routing in Large-Scale Date Centers; RFC 7938, August 2016

[RFC8257] S. Besley et al.; Data Center TCP (DCTCP): TCP Congestion Control for Data Centers; RFC 8257, October 2017

[Vasu09] V. Vasudevan et al.; Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communications; Barcelona, Aug. 2009, ACM SIGCOMM 09