CS/ECE 4457

Computer Networks:
Architecture and Protocols

Lecture 25

Beyond The Internet

Qizhe Cai

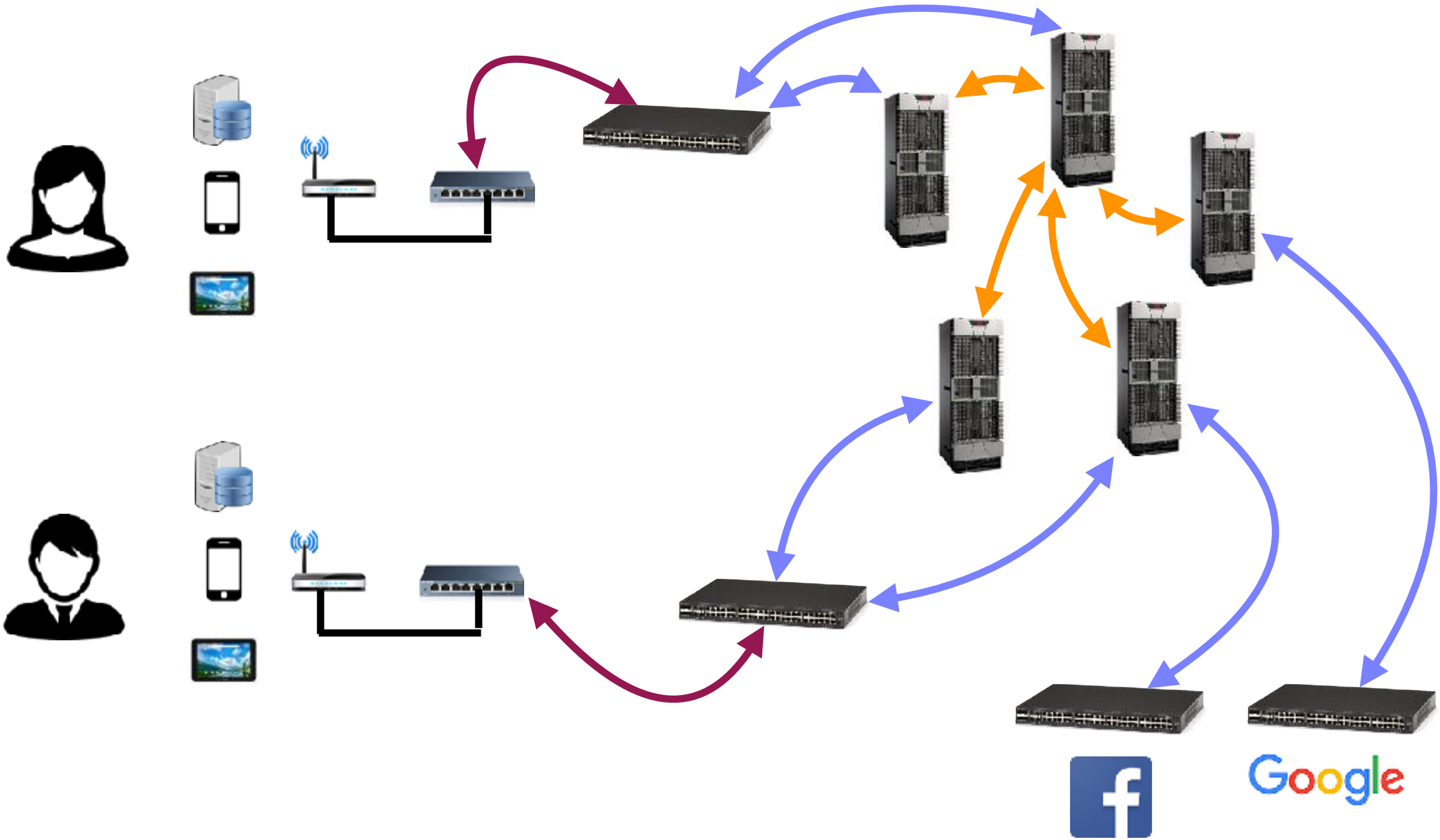
Announcements

- **Final: 04/28**
- **The solution of practice exam will be posted**

Taking 25 steps back!

What is a computer network?

A set of network elements connected together, that implement a set of protocols for the purpose of sharing resources at the end hosts



Sharing networks

- **Two approaches**
    - Reservation (circuit switching)
    - Statistical multiplexing (packet switching)

- **Motivation for WHY modern networks use “packets”**

- **How to implement this?**

The end-to-end story

- Application opens a **socket** that allows it to connect to the **network stack**
- Maps **name** of the web site to its **address** using **DNS**
- The network stack at the source embeds the address and **port** for both the source and the destination in **packet header**
- Each **router** constructs a **routing table** using a distributed algorithm
- Each router uses destination address in the packet header to look up the **outgoing link** in the routing table
    - And when the link is free, forwards the packet
- When a packet arrives the destination:
    - The network stack at the destination uses the port to forward the packet to the right application

Realizing end-to-end design: Three Principles

- How to break system into modules
  - **Layering**
- Where are modules implemented
  - **End-to-End Principle**
- Where is state stored?
  - **Fate-Sharing**

Five Layers (Top - Down)

- **Application:** Providing network support for apps
- **Transport (L4):** (Reliable) end-to-end delivery
- **Network (L3):** Global best-effort delivery
- **Datalink (L2):** Local best-effort delivery
- **Physical (L1):** Bits on wire

Link Layer (L2)

- **Broadcast medium:** Ethernet and CSMA/CD
- **We studied that Broadcast Ethernet does not scale to large networks**
    - Motivation for switched Ethernet
- **Broadcast storm:** if using broadcast on switched Ethernet
    - Motivation for Spanning Tree Protocol
- **Limitations of Spanning Tree Protocol:**
    - Low bandwidth utilization, high latency, unnecessary processing
    - Does not scale to the entire Internet
    - Motivation for **routing protocols** in the Internet

Network Layer (L3)

- **Internet Protocol:**
    - Addressing, packet header as an interface, routing
- **Routing tables:**
    - Correctness and validity: Dead ends, loops
    - A collection of spanning trees, one per destination
- **Constructing valid routing tables (within an ISP)**
    - Link-state and distance-vector protocols
    - Focused a lot on learning via examples
    - Can still have loops: failures remain to be a pain
- **How to use routing tables**
    - **Packet header as an interface**
    - Learnt why packet headers look like the way they do

Network Layer (L3), Cont.

- **Internet Protocol:**
    - Addressing, packet header as an interface, routing
- **Addressing:**
    - Link layer uses “flat” addresses
    - **Does not scale to Internet**: motivation for IP addresses
    - **Scalability challenges**: Routing table sizes, #updates
    - Solution: **Hierarchical addressing**
- **Forwarding**
    - **Switch architecture**
    - Longest Prefix matching for forwarding at line rate
    - Scheduling using priorities

Network Layer (L3), Cont.

- **Internet Protocol:**
    - Addressing, packet header as an interface, routing
- **Limitations of link-state and distance-vector routing:**
    - Require visibility of the entire Internet
    - **ISPs do not like that:** motivation for Inter-domain routing
    - **Border Gateway Protocol**
        - A simple modification of distance-vector protocol
- **Routing with policies**
    - **Customer-provider-peer relationships**
    - Gao-Rexford policies
- **Completes the network layer: provides connectivity**

Details for complete picture

- **DHCP: Dynamic Host Configuration Protocol**
    - For each host to figure out its IP address, local DNS, first-hop router
- **ARP: Address Resolution Protocol**
    - For finding other servers on the same local area network (L2)
    - Mapping from IP addresses to names (MAC addresses)
- **Domain Name System**
    - **Mapping Human readable destination names to IP addresses**
    - Hierarchical structure

Transport Layer

- **Goals of reliable transport**
    - **Correctness condition**
    - Why do we need ACKs, timers, window-based design
- **One realization of reliable transport: TCP**
    - Mostly implementation details following the above design
    - For **max-min fairness**, flow performance and utilization
    - **Flow control**
        - Ensuring the sender does not overwhelm the **receiver**
        - Via receiver advertised window size
    - **Congestion control**
        - Ensuring the sender does not overwhelm the **network**
        - Slow start, Additive-increase Multiplicative-decrease, timeouts

Goals for Today's Lecture

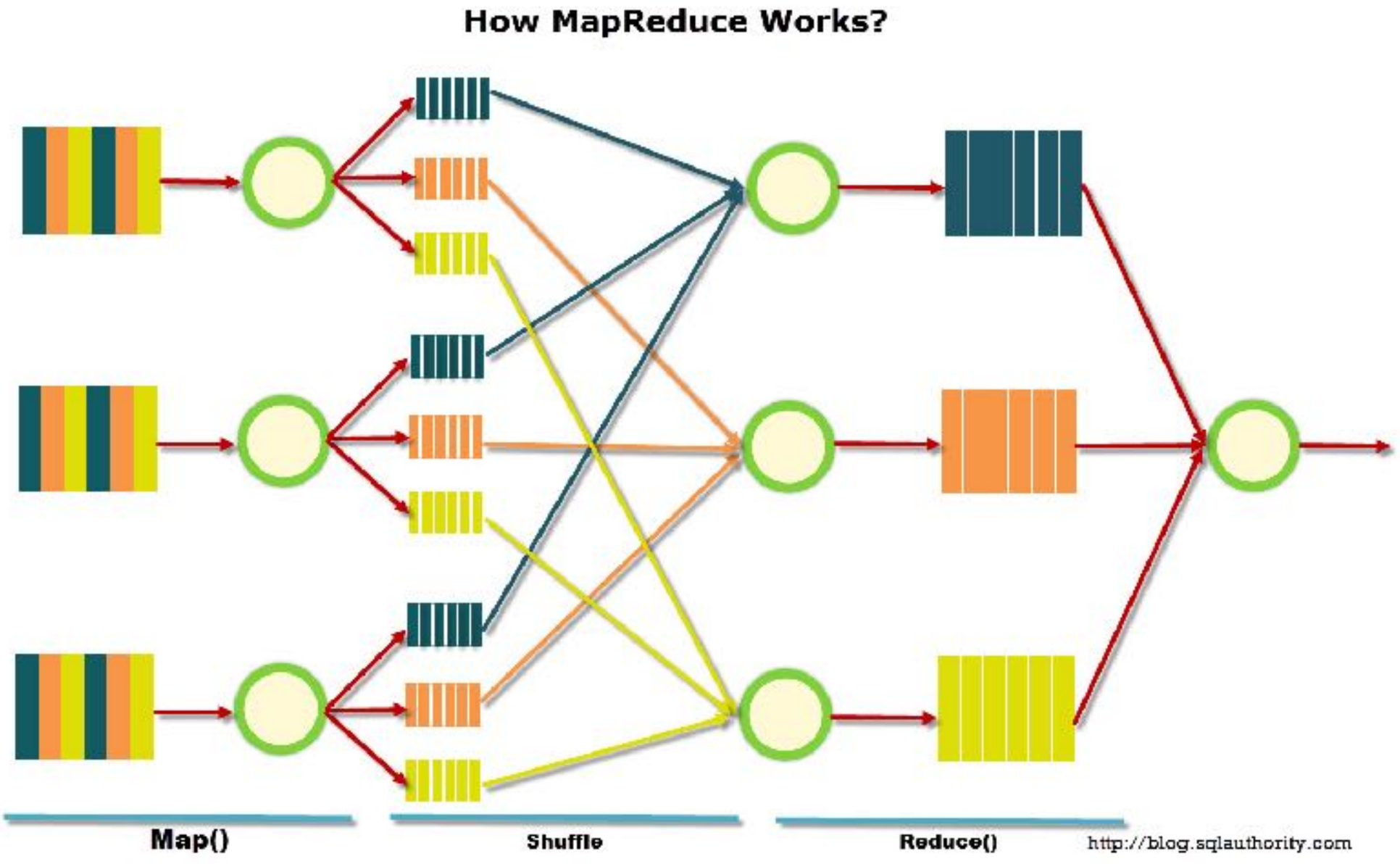
- **Understand how a new environment may lead to new design decisions**

- **Case study: Datacenter networks**

Lets start with an application - MapReduce

- **Large scale data analytics**

- **Ex: Google “crawls” the web, and creates search indexes**



Performance of distributed systems

depends heavily on the

datacenter interconnect

Evaluation metrics for datacenter interconnects

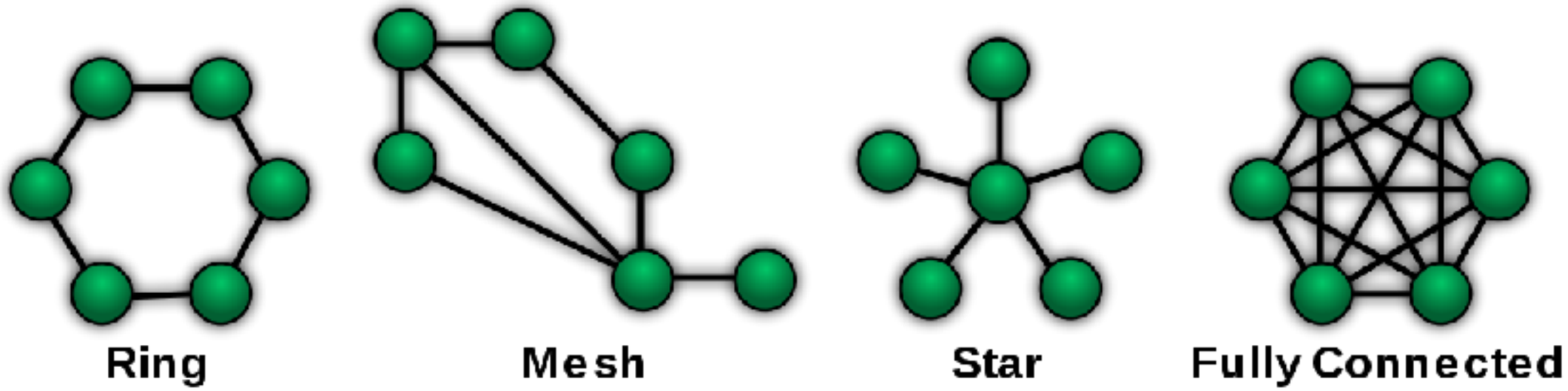
- **Diameter** –
    - Definition: max #hops between any 2 nodes
    - Importance: Speed-of-light latency (why not observed latency?)
        - Answer: queueing delays dependent on traffic as well

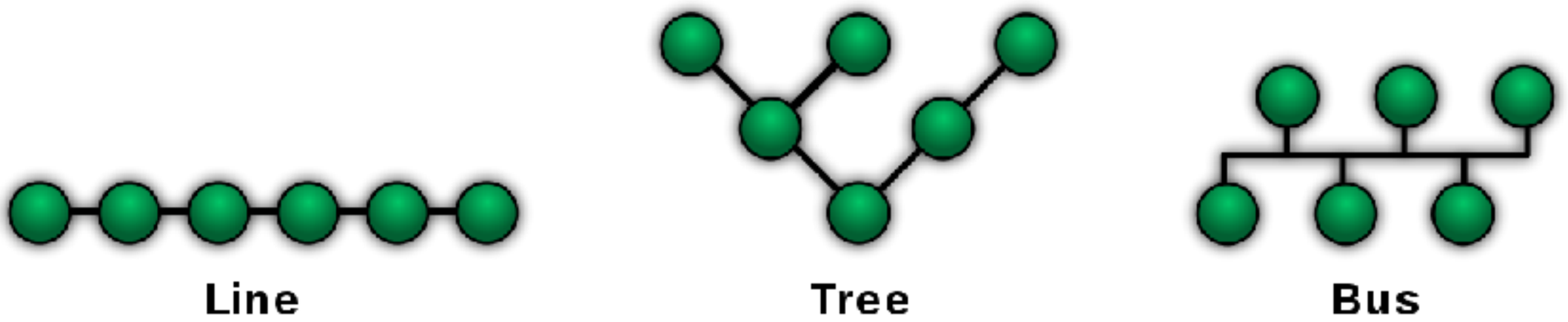
- **Bisection Width** –
    - Definition: min #links cut to partition network into 2 equal halves
    - Importance: Fault tolerance

- **Bisection Bandwidth** –
    - Definition: min bandwidth between any 2 equal network halves
    - Importance: Bandwidth bottleneck

- **Oversubscription** –
    - Definition: ratio of worst-case achievable aggregate bandwidth between end-hosts to total bisection bandwidth

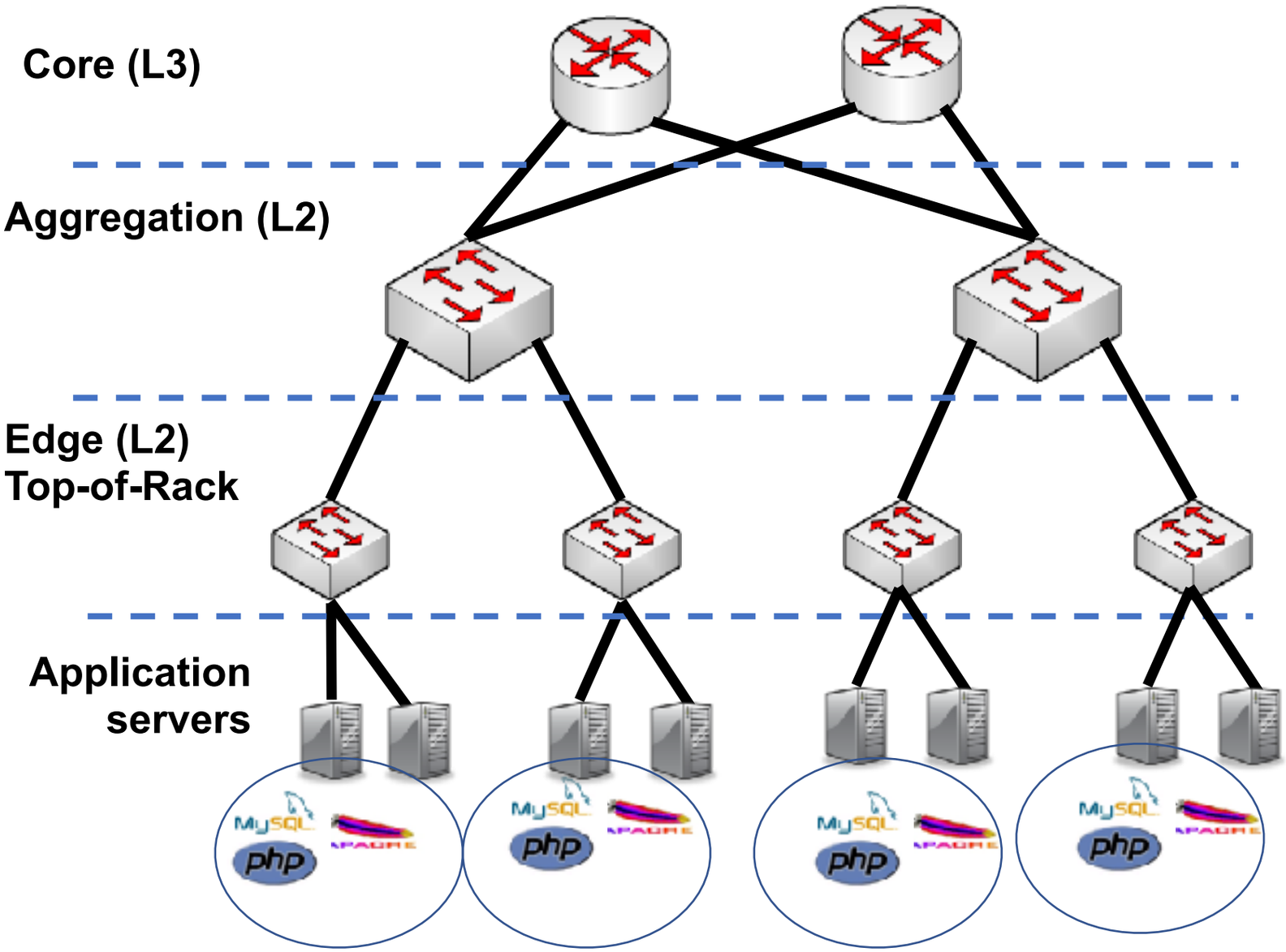
Legacy Interconnects





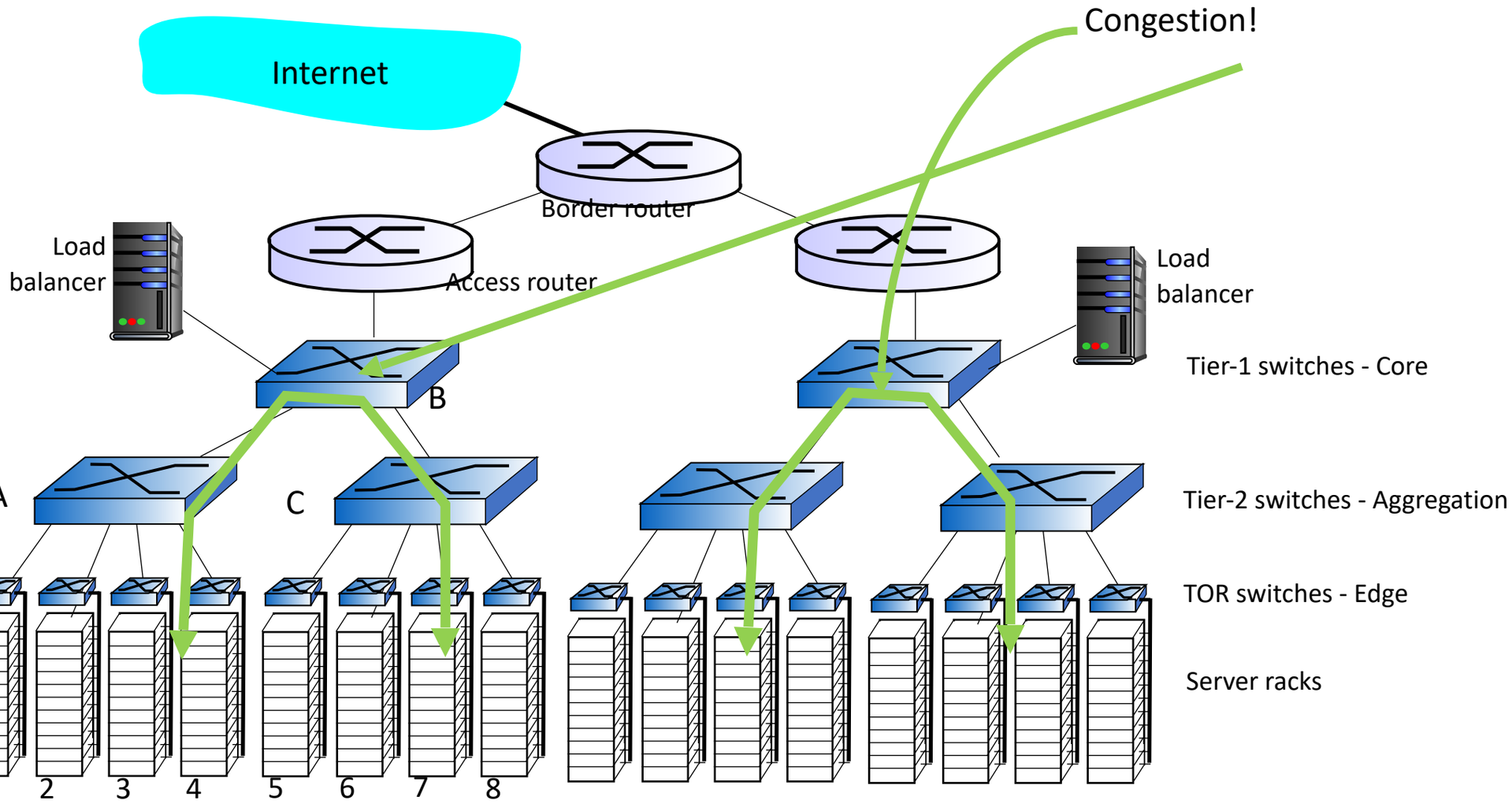
Diameter, Bisection Width, Bisection Bandwidth, Oversubscription

Canonical Datacenter Interconnect



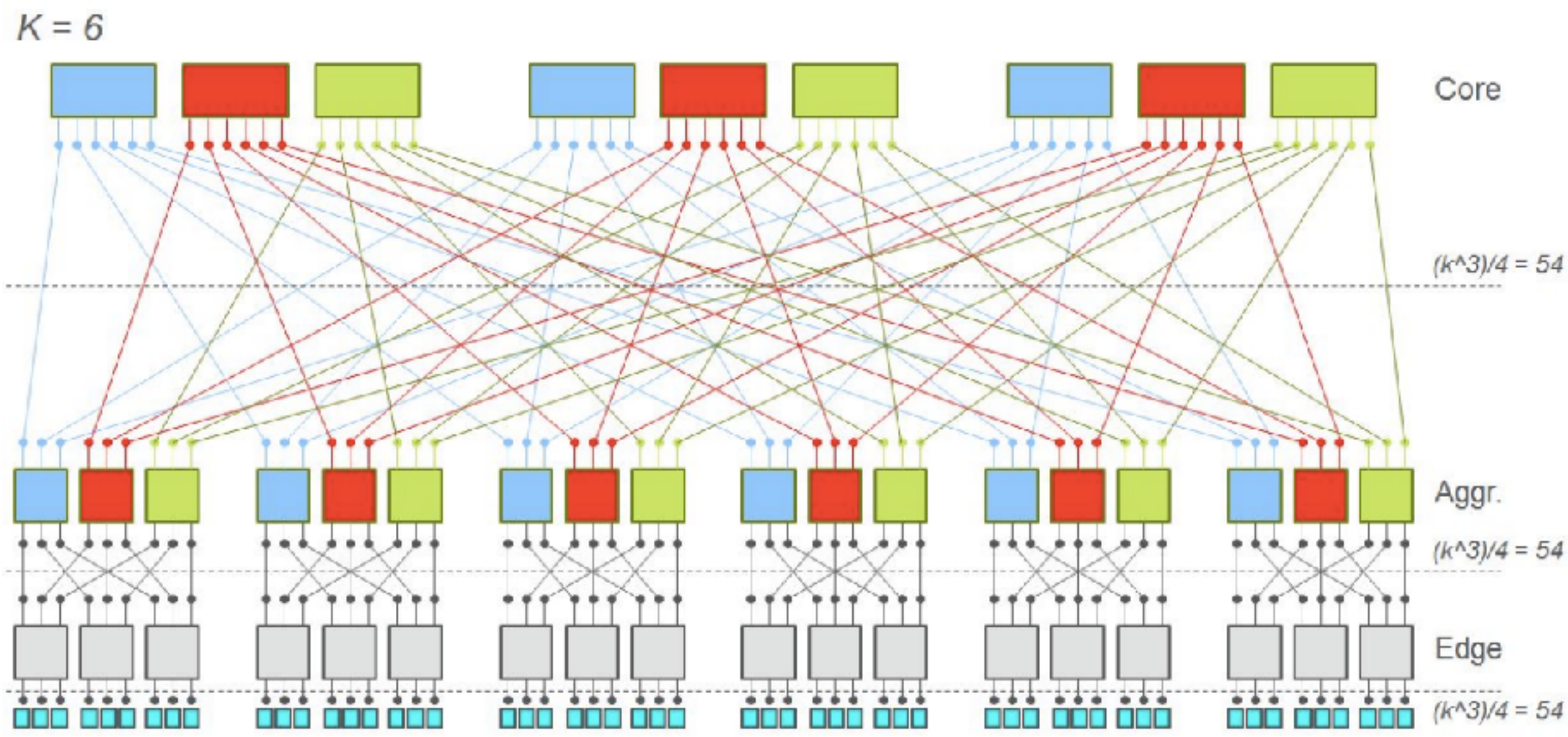
Diameter, Bisection Width, Bisection Bandwidth, Oversubscription

What a real Datacenter Interconnect really looks like?



Diameter, Bisection Width, Bisection Bandwidth, Oversubscription

Modern Datacenter Interconnect



Diameter, Bisection Width, Bisection Bandwidth, Oversubscription

Case study:

The evolution of Google's datacenter interconnect

Google datacenter interconnect principles

- High bisection bandwidth and graceful fault tolerance
- Low Cost
- Centralized control

Centralized Control

- **Custom control plane**
  - Existing protocols did not support routing along multiple paths
  - Protocol overhead of running distributed protocols on large scale
  - Easier network manageability
- Treat the network as a **single switch with O(10,000) ports**
- Anticipated some of the principles of **Software Defined Networking**

Issues

High congestion as utilization approached **just 25%**

- Bursty flows
- Limited buffer on commodity switches
- Intentional oversubscription for cost saving
- Imperfect flow hashing

Congestion Solutions

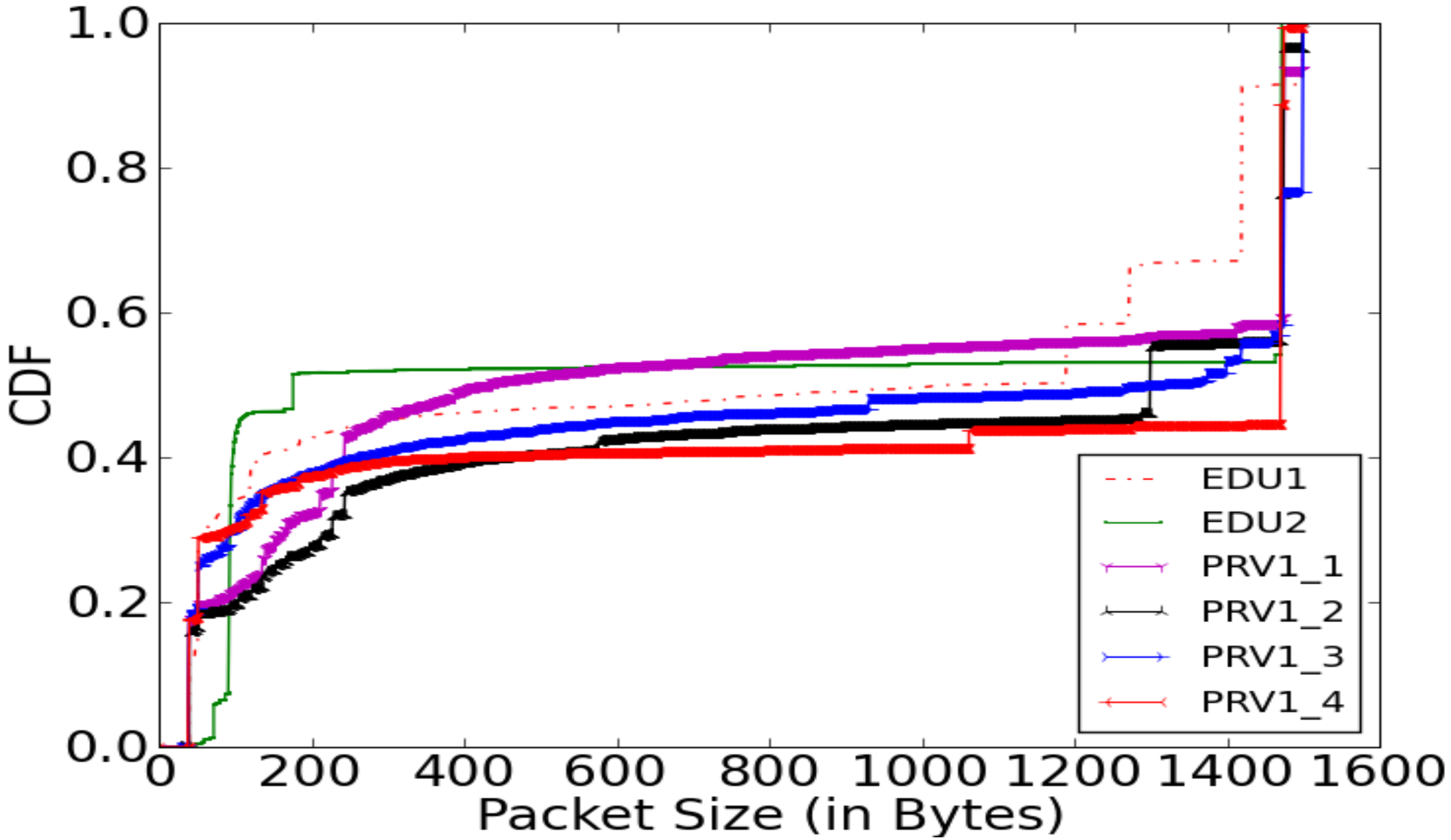
We will see more later

- Configure switch hardware schedulers to drop packets based on QoS
- Tune host congestion window
- Explicit Congestion Notification
- Dynamic buffer sharing on merchant silicon to absorb bursts
- Carefully configure switch hashing to support ECMP load balancing

Case study:

The traffic in Google's datacenter interconnect

Packet Size Distribution



Any interesting observations?

Observations from the Interconnect

- **Link utilization low at edge and aggregate level**
- **Core most utilized**
    - Hot-spots exists (> 70% utilization)
    - < 25% links are hotspots
    - Loss occurs on less utilized links (< 70%)
        - Implicating momentary bursts
- **Time-of-Day variations exists**
    - Variation an order of magnitude larger at core

Insights from the Interconnect

- **75% of traffic stays within a rack (Clouds)**
  - Applications are **not uniformly** placed
- **Half packets are small (< 200B)**
  - **Keep alive integral** in application design
- **At most 25% of core links highly utilized**
  - Need effective routing algorithms to reduce utilization
  - Load balance across paths and migrate applications
- **Questioned popular assumptions**
  - Do we need more bisection? **No**
  - Is centralization feasible? **Yes**

What is REALLY different when compared to the Internet?

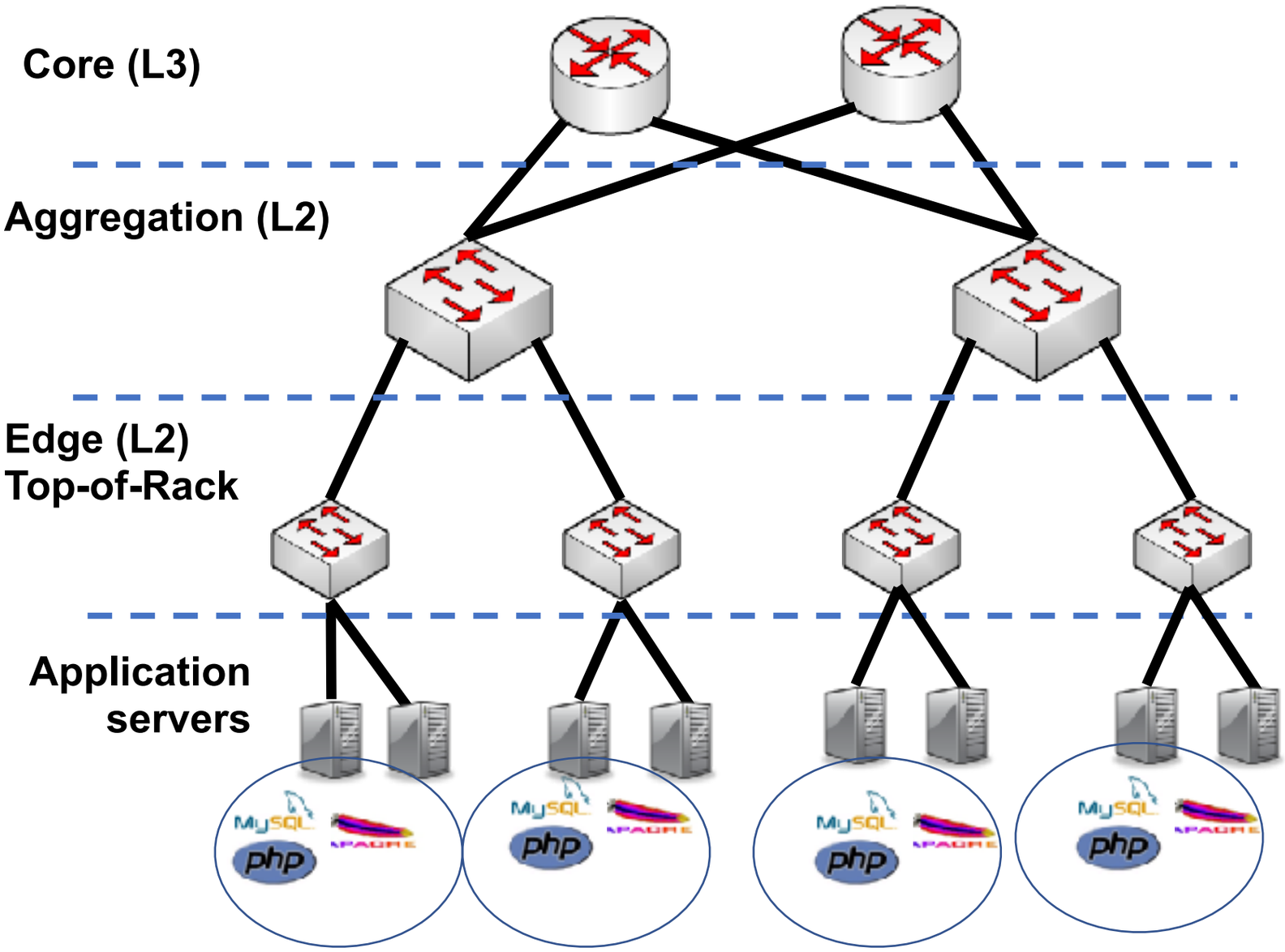
What is REALLY different from the Internet

- **Single entity**
  - **Google owns everything, from the OS to the network hardware**
  - Discussion: how could we exploit this property?
- **Link Layer?**
  - We never exploited anything about "ASes" in link layer
- **Network Layer?**
  - Do we still **need** BGP?
  - Could we still **use** BGP?
- **Transport Layer?**
  - A lot of failure modes of TCP go away (OS owned by Google)
  - Is TCP still a good solution?
- **Reality: Increasingly less separation between link and network layers**

What is REALLY different from the Internet

- **Fixed (structured) topology, complete control and knowledge**
  - **The topology is designed, owned, and managed by Google**
  - Discussion: how could we exploit this property?
- **Link Layer and Network Layer**
  - More efficient algorithms for route computation

Example: Simplification of Routing



Can you think of a simple mechanism?

What is REALLY different from the Internet

- **Fixed (structured) topology, complete control and knowledge**
  - **The topology is designed, owned, and managed by Google**
  - Discussion: how could we exploit this property?
- **Link Layer and Network Layer**
  - More efficient algorithms for route computation
  - Could "bake in" routing **results** into switch routing tables
  - Software-defined networks, centralized control
  - **Other benefits:**
    - Better control over "load balancing"
    - Avoid convergence issues (but new issues come up)
- **Transport Layer?**
  - We never made any assumptions about topology in L4 design
  - Is TCP still a good idea?

What is REALLY different from the Internet

- **Tiny round trip times**
  - **Less than 5 microseconds (for a single packet)**
  - Discussion: how could we exploit this property?
- **Link Layer and Network Layer?**
  - Millisecond-level convergence times no longer “sufficient”
  - Even more motivation for software-defined, centralized control
- **Transport layer?**
  - Most flows small; can be completed within a couple of RTT
  - Even 3-way hand-shake takes 7.5microseconds
  - TCP is not going to work well!

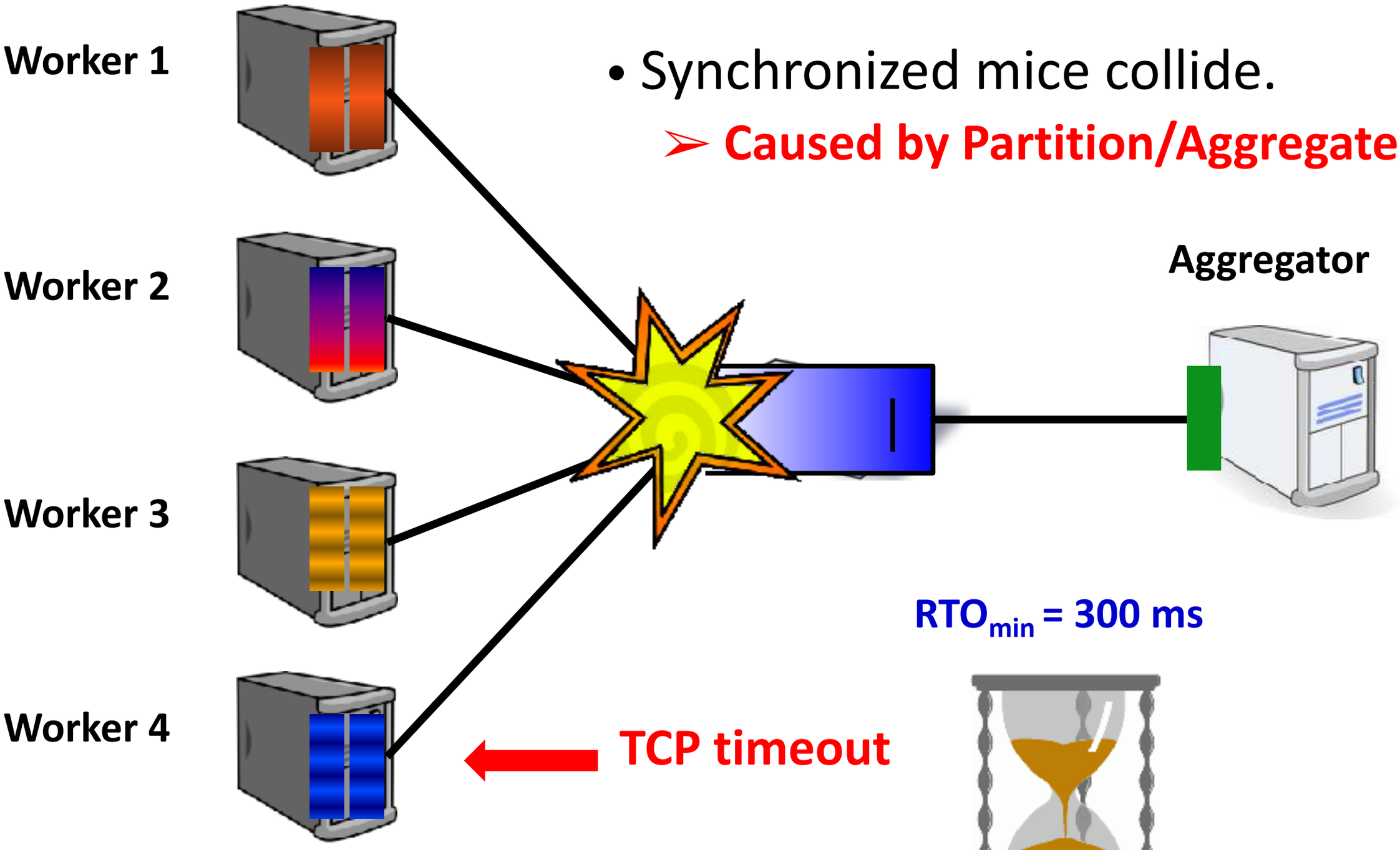
Datacenter Transport Design:

One of the most active research areas

TCP in datacenter context

- **TCP is too inefficient**
  - Three-way handshake takes **too long**
  - Does not work well with **short flows**
  - Not designed for **low latency**
  - Has no notion of **deadlines**
  - **Does NOT work well with “Incast”**
  - **Queue build-up due to long flows; short flows suffer**

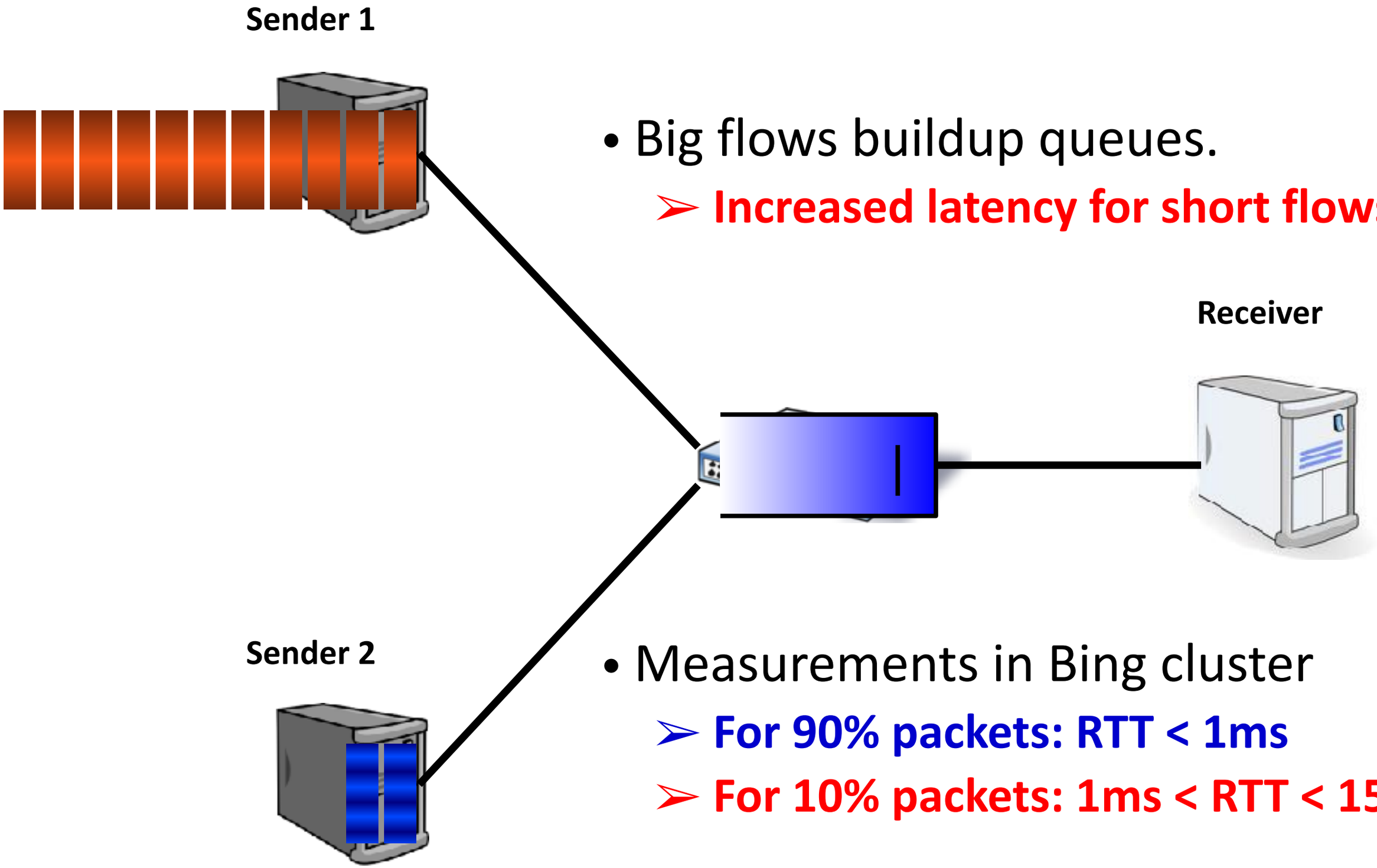
Incast



- Synchronized mice collide.
  - ➢ **Caused by Partition/Aggregate.**

RTO_{min} = 300 ms

Queue Buildup



- Big flows buildup queues.
  - ➢ **Increased latency for short flows.**

- Measurements in Bing cluster
  - ➢ **For 90% packets: RTT < 1ms**
  - ➢ **For 10% packets: 1ms < RTT < 15ms**

TCP in datacenter context

- **TCP is too inefficient**
  - Three-way handshake takes **too long**
  - Does not work well with **short flows**
  - Not designed for **low latency**
  - Has no notion of **deadlines**
  - **Does NOT work well with "Incast"**
    - **Queue build-up due to long flows; short flows suffer**
- **How would you solve these problems?**