CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, George Varghese

SIGCOMM 2014 (Best Paper Award)

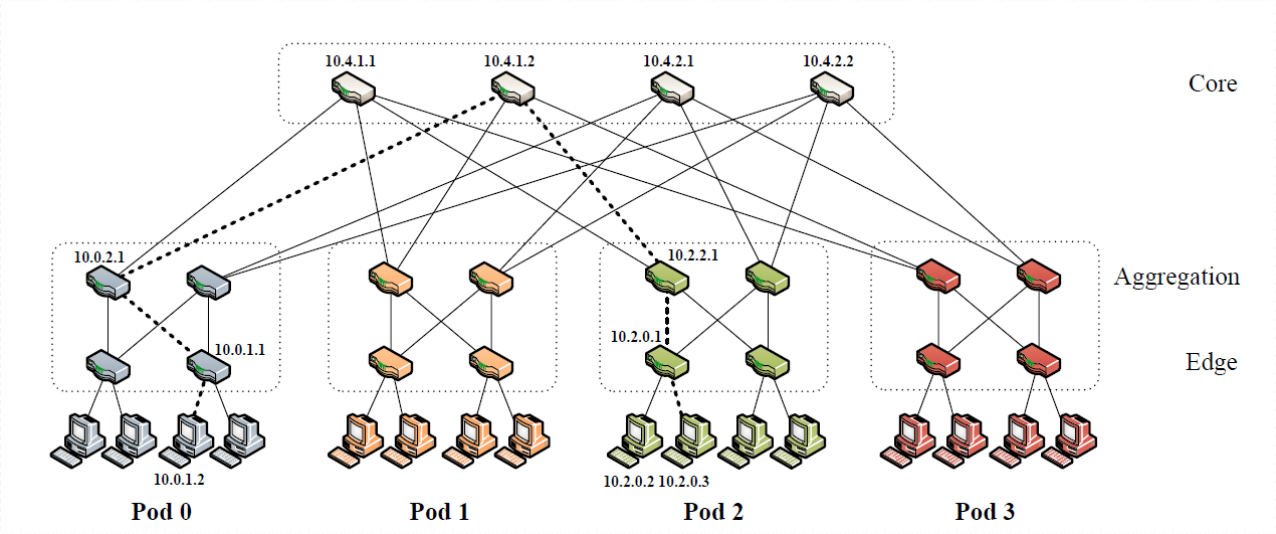
Another paper from Mohammad Alizadeh

Outline

- **Background (Some are not in the paper…)**
- CONGA
- Implementation and Evaluation
- Review

What is Load Balance?

- Load Balancing distributes network traffic across **multiple parallel paths** in a data center network to utilize network capacity efficiently.
  - Clos Topology
  - Fattree



Why Do We Need Load Balance?

- **Long-tailed Distribution Again!**
  - Many **small flows**
  - Few **large flows** consume most bandwidth
- An Example: if we randomly distribute the traffic…
  - Path 1: Large flow + Large flow
  - Path 2: small flow
  - Path 3: small flow
  - **Path 1: Congestion**
  - **Path 2/3: Underutilized**

Design Space of Load Balance

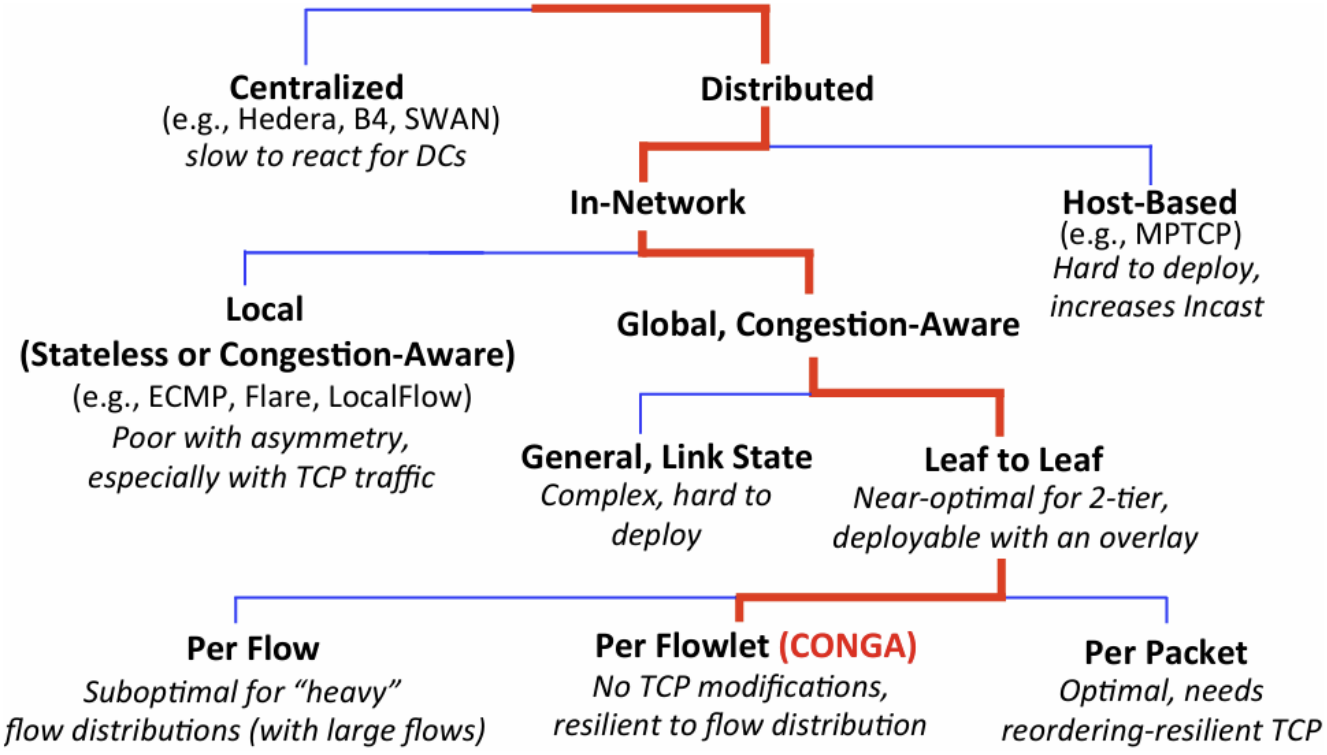


Figure 1: Design space for load balancing.

Design Space of Load Balance

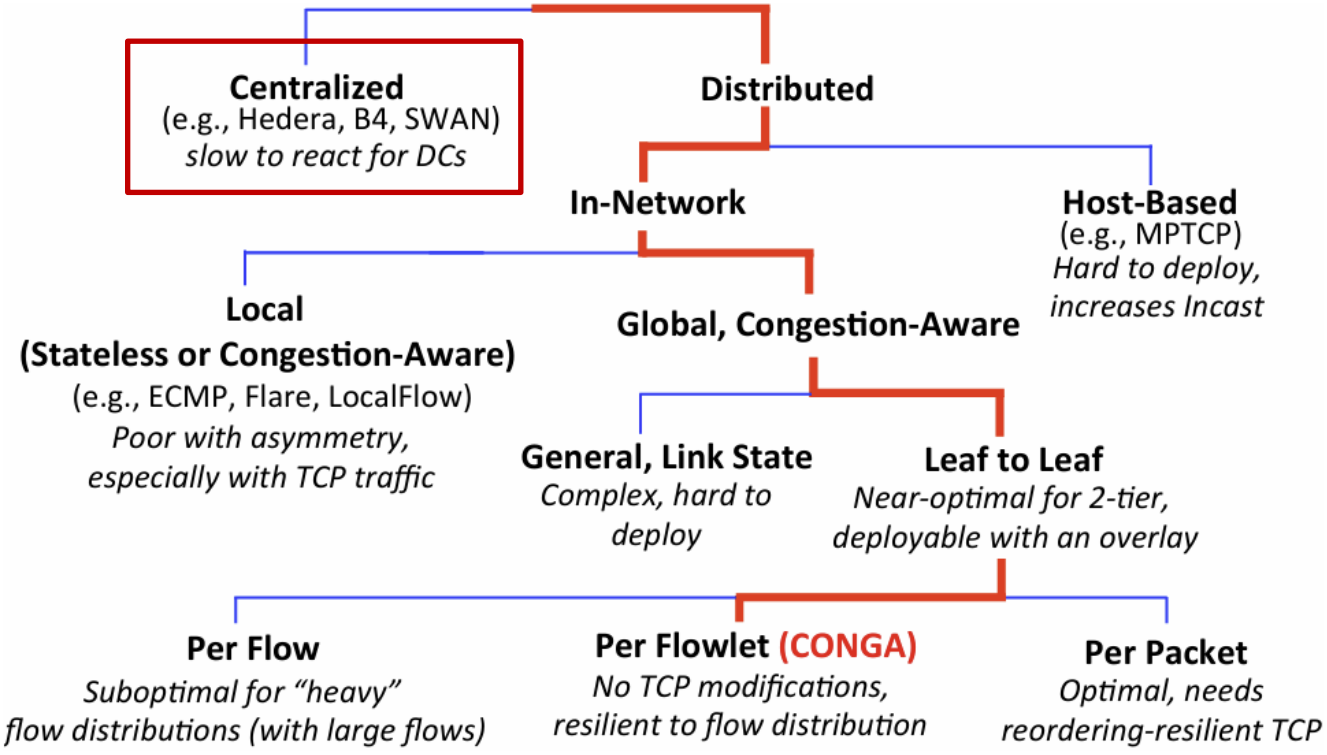
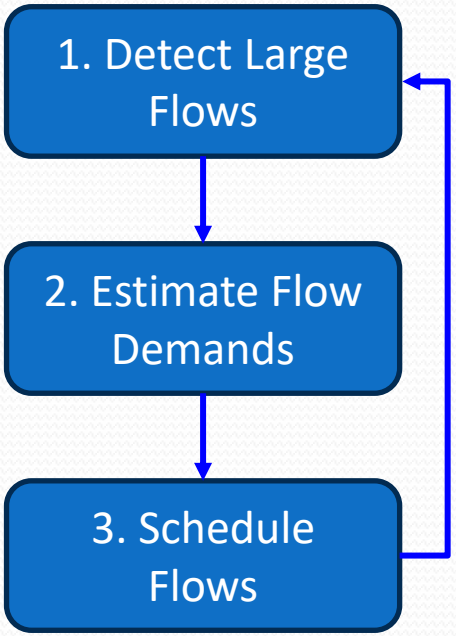


Figure 1: Design space for load balancing.

Why We Don't Like Centralized Solution?

- Hedera: Dynamic Flow Scheduling for Data Center Networks, NSDI 2010



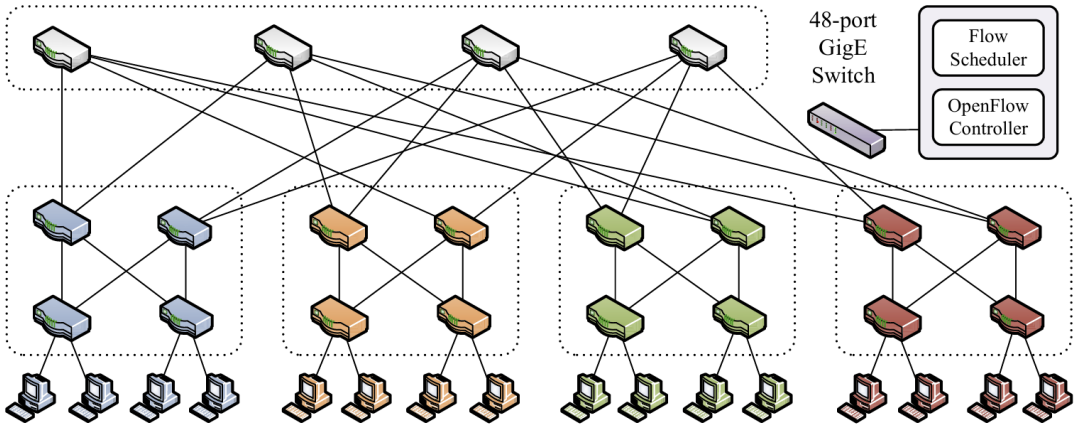


Figure 8: System Architecture. The interconnect shows the data-plane network, with GigE links throughout.

Why We Don't Like Centralized Solution?

- Datacenter traffic is very **bursty** and **unpredictable**
  - Centralized solution needs more RTTs to communicate with the controller
  - It takes long for controller to solve the optimization problem with many flows: Hedera scheduler in [2] runs every 5 seconds
- Very **regular** topologies
  - Opportunities exist for distributed solutions to achieve optimal

Design Space of Load Balance

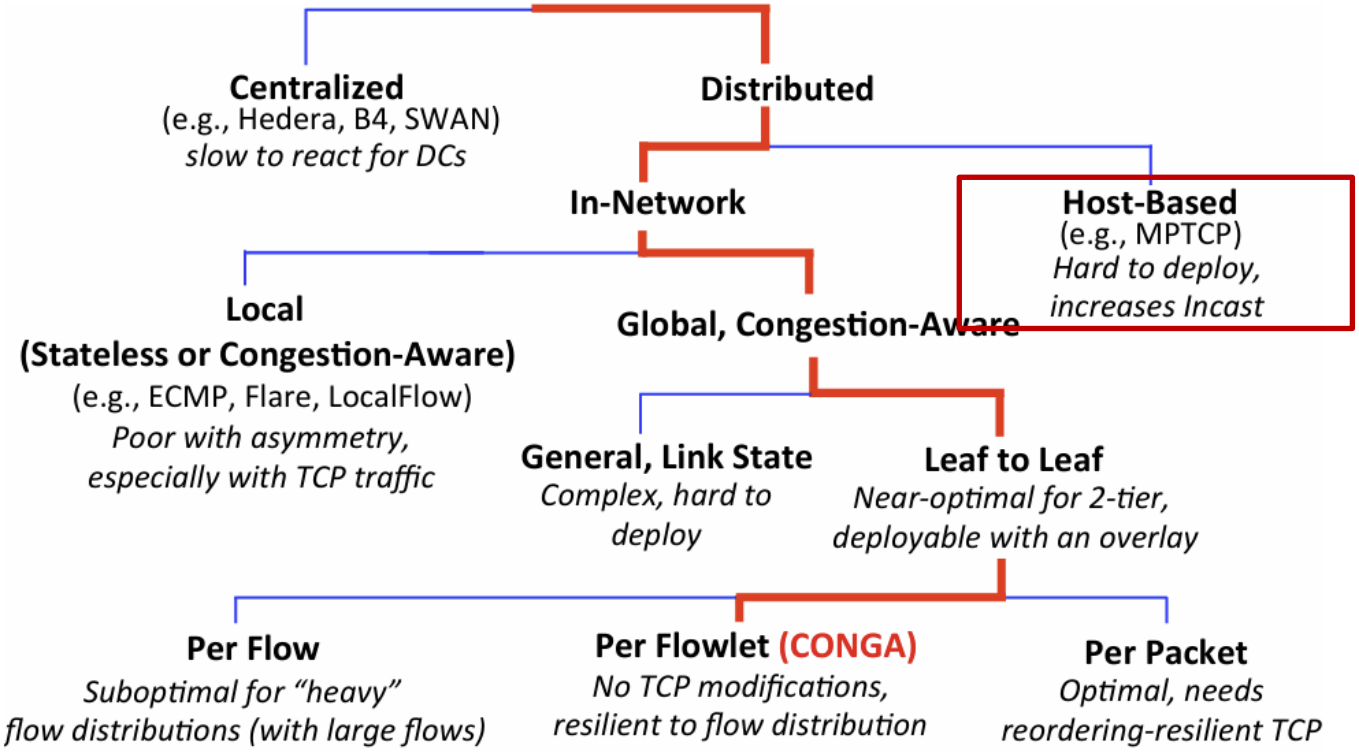
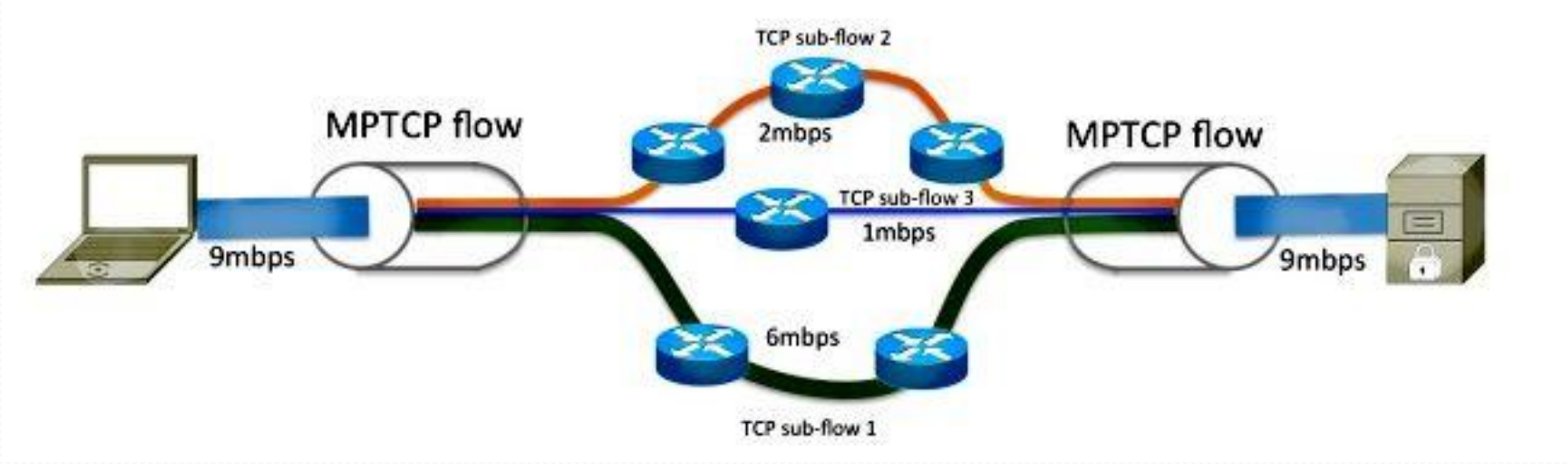


Figure 1: Design space for load balancing.

What is MPTCP?



- Multipath TCP splits one connection into **multiple TCP subflows**, allowing traffic to use **multiple paths simultaneously**.
- Originally designed for:
  - Multi-homed devices (e.g., **WiFi + Cellular**)
  - Wide-area networks with **few paths**

Why We Don't Like MPTCP

- Too many paths exist in DCN
  - Fat-tree has **$(k/2)^2$ shortest-paths** between any two hosts
- Make **Incast** more severe
  - **Degrade performance of short flows**
- Require Linux **kernel support**: Kernel TCP support
  - Kernel-bypass Mechanism: **RDMA**
    - Will be covered in more details in next lecture

Design Space of Load Balance

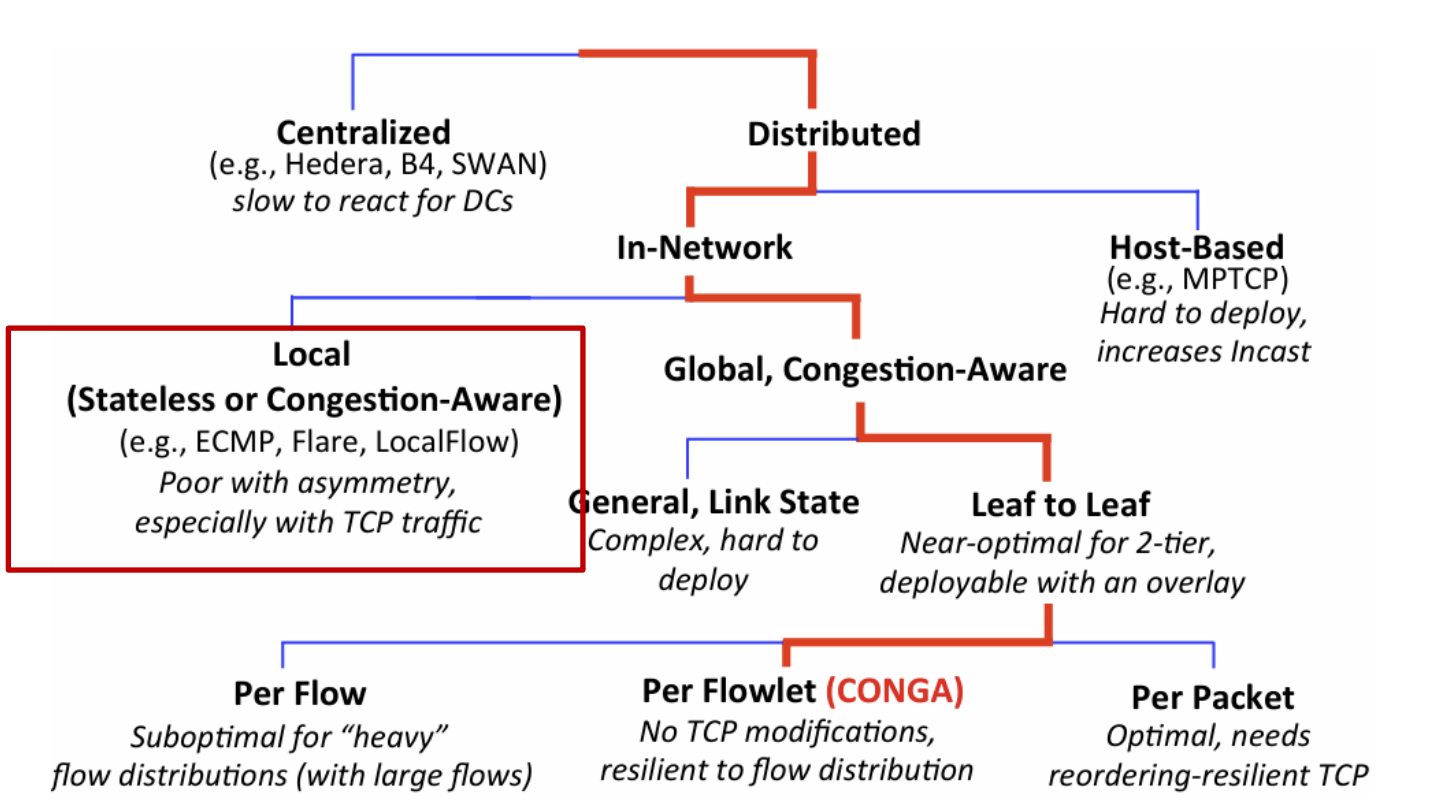
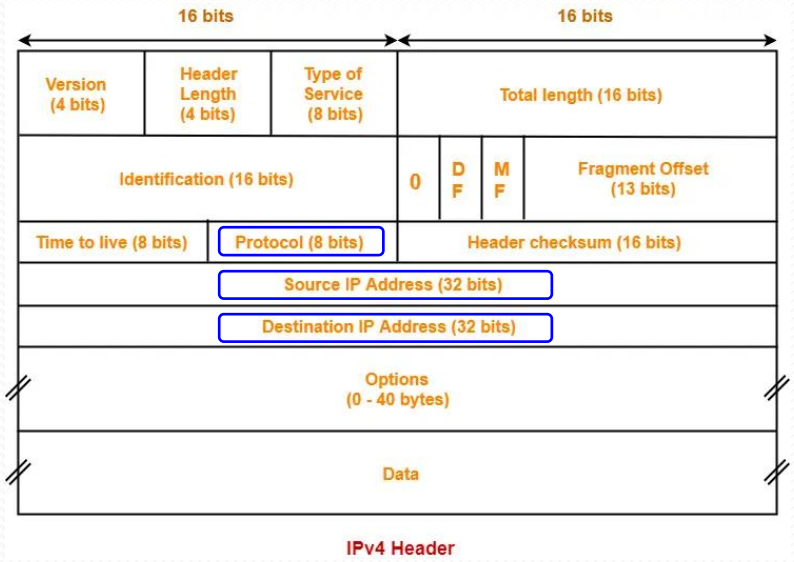


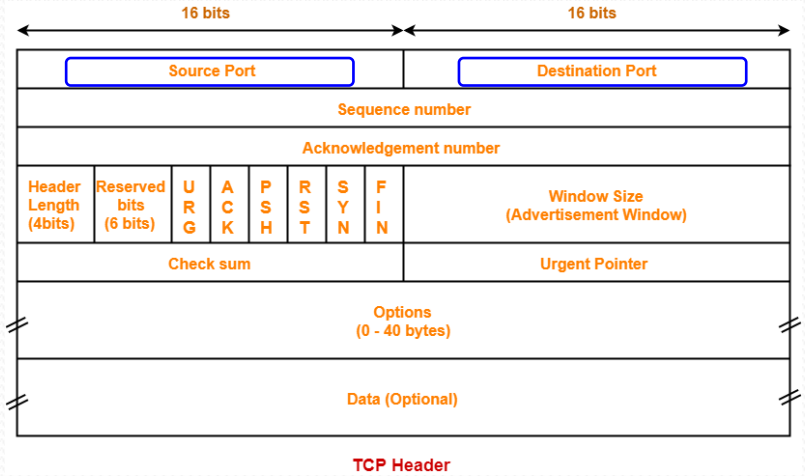
Figure 1: Design space for load balancing.

ECMP – Decisions Made By Local Switch

- Equal-Cost Multi-Path
- **hash(5-tuple) % N_paths**
  - src IP
  - dst IP
  - src port
  - dst port
  - protocol
- **Not sense congestion**
- **Local Decision**



IPv4 Header

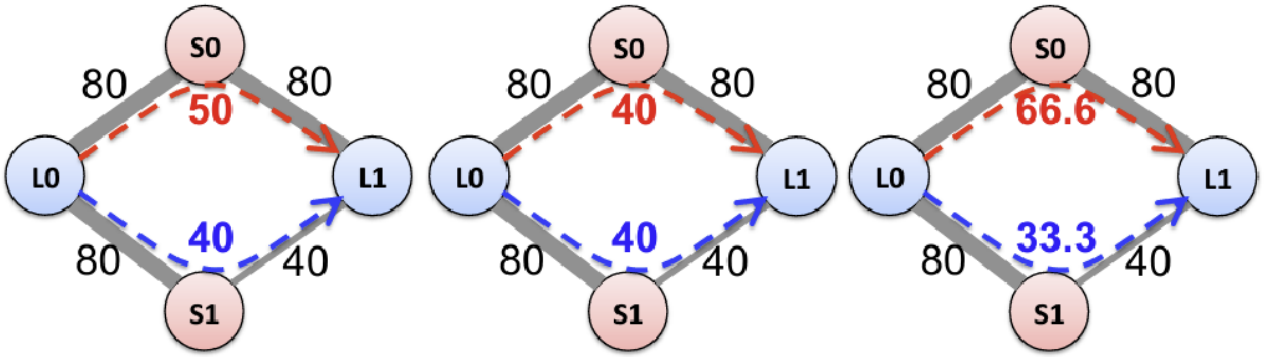


TCP Header

Why We Don't Like ECMP?

- Poor performance with topology **asymmetry**
  - **Asymmetry?**
    - Regular Topology
    - **But failure still exist**
      - **Link failure**
      - **Link degradation (100Gbps -> 25Gbps)**

Why We Don't Like ECMP?



(a) Static (ECMP) (b) Congestion-Aware: Local Only (c) Congestion-Aware: Global (CONGA)

Figure 2: Congestion-aware load balancing needs non-local information with asymmetry. Here, L0 has 100Gbps of TCP traffic to L1, and the (S1, L1) link has half the capacity of the other links. Such cases occur in practice with link-aggregation (which is very common), for instance, if a link fails in a fabric with two 40Gbps links connecting each leaf and spine switch.

But Actually, People Like ECMP

- ECMP is still widely adopted nowadays
  - All **switches support** ECMP
  - **Simple** to deploy
  - **Asymmetry** is rare in real world

Design Space of Load Balance

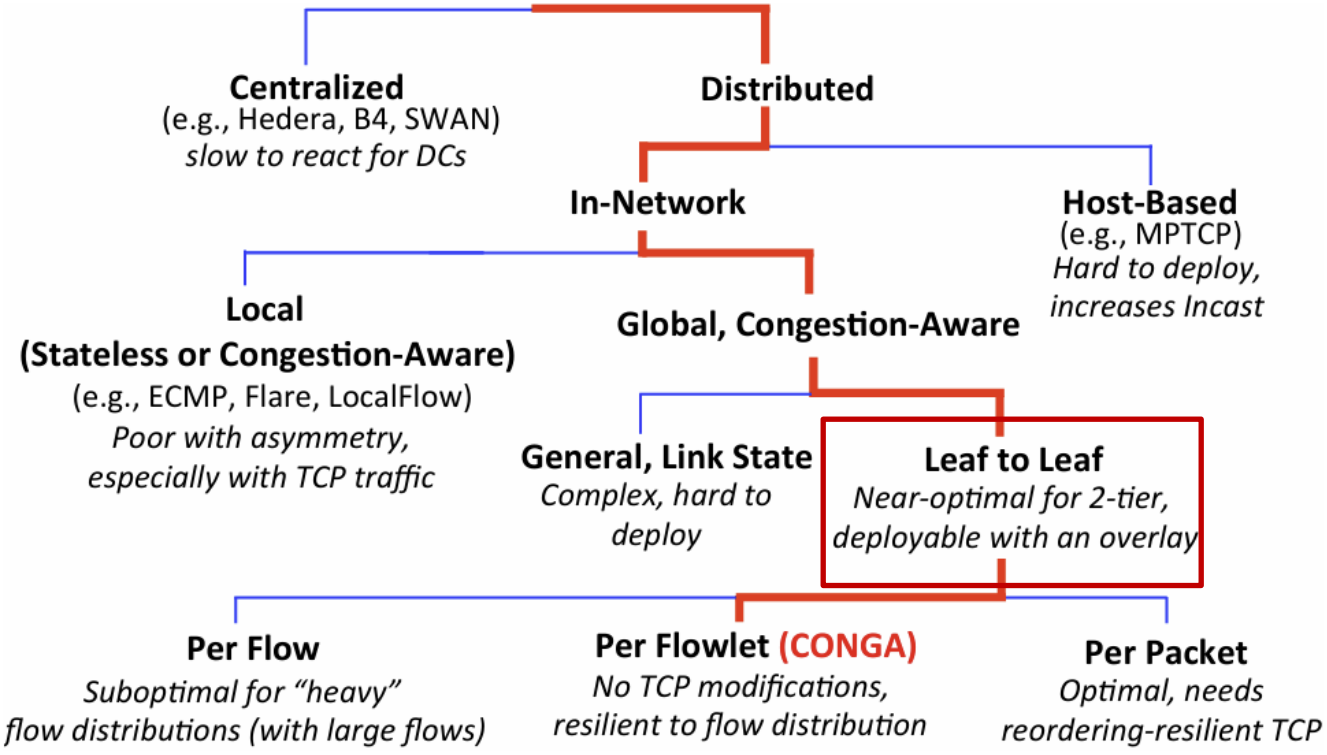


Figure 1: Design space for load balancing.

Why Leaf to Leaf Solution?

- Leaf switches know the source rack or destination rack
  - Ideal place to do load balance decision
  - Even with overlay networking
- **Congestion Feedback**
  - A good place to collect congestion information of a path

Design Space of Load Balance

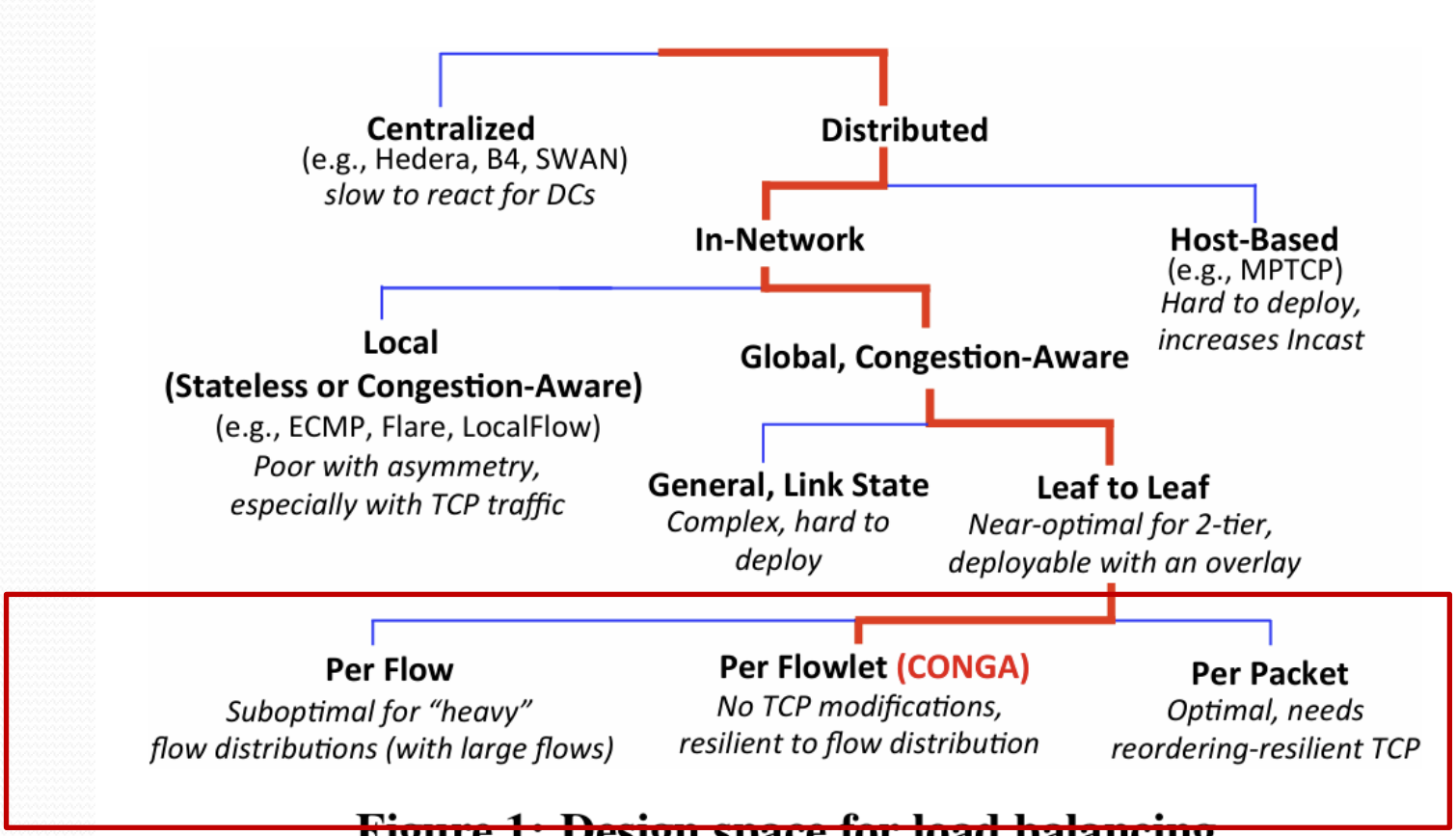


Figure 1: Design space for load balancing.

What are Different Granularities?

- Per-flow: One flow will traverse to only one path
  - The packets within one flow use the same path
- Per-packet: One packet can choose on path
  - It can achieve optimal load balance
  - IF not consider the performance degradation caused by TCP reordering!

Design Space of Load Balance

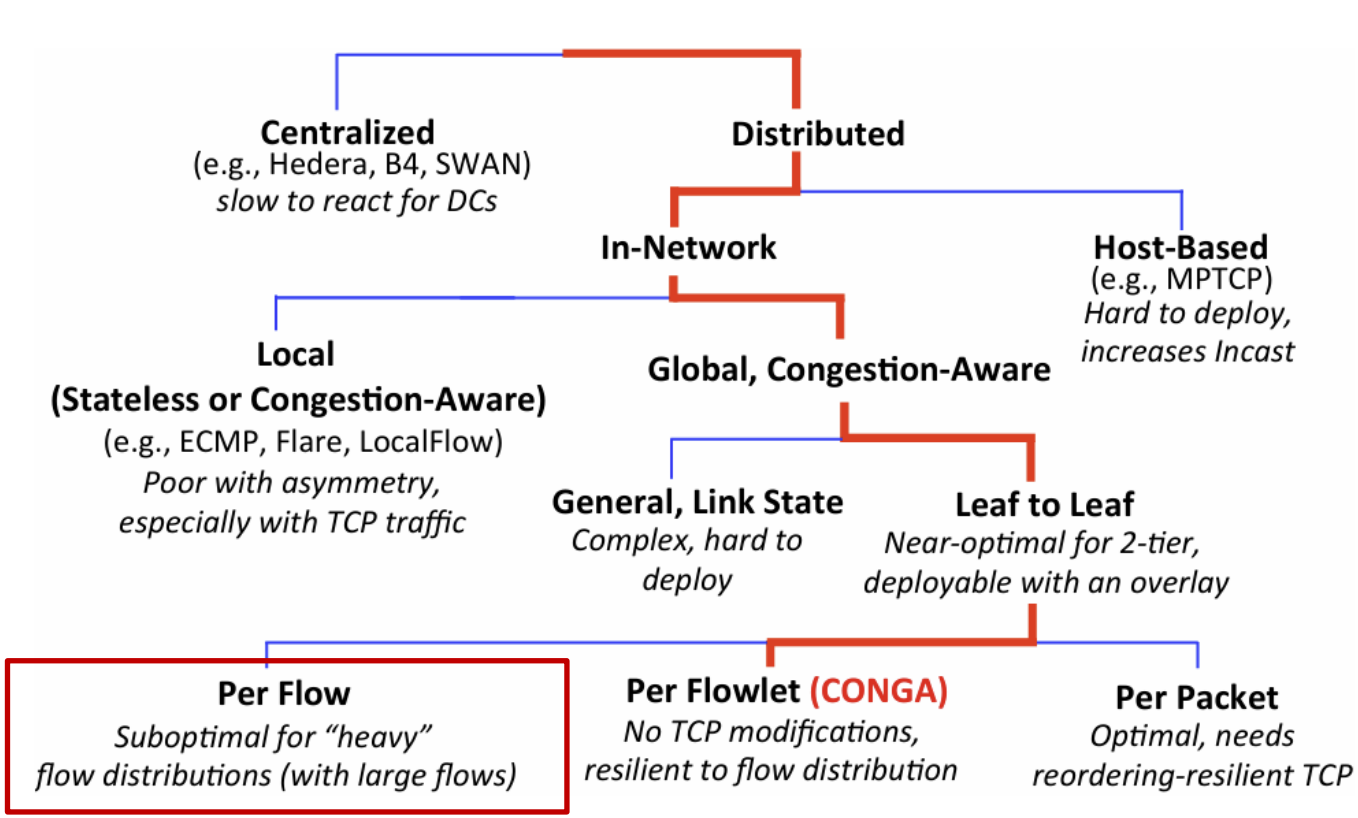


Figure 1: Design space for load balancing.

Per-Flow ?

- **Advantage**
  - No modification to TCP
- **Disadvantage**
  - Long tail distribution
  - If large flows are scheduled onto one path, performance degrades

Design Space of Load Balance

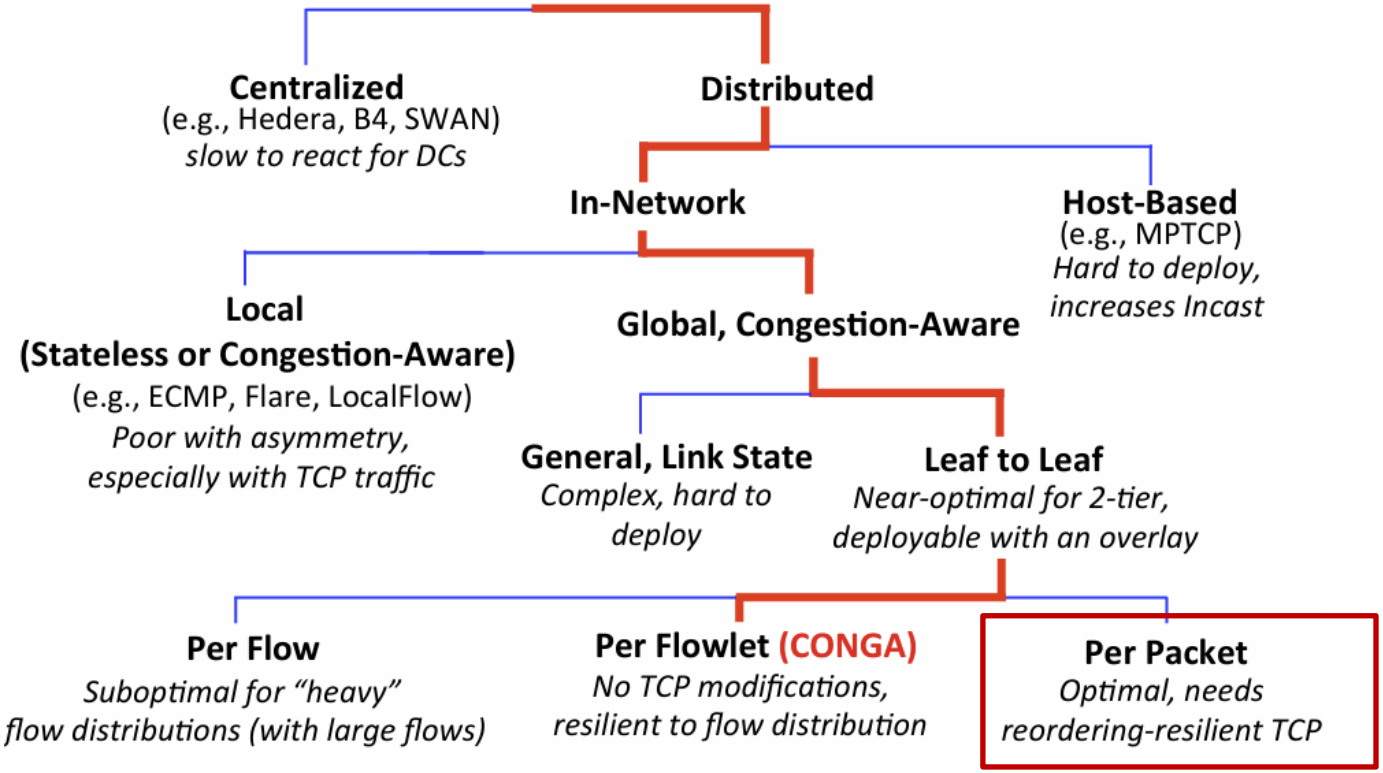
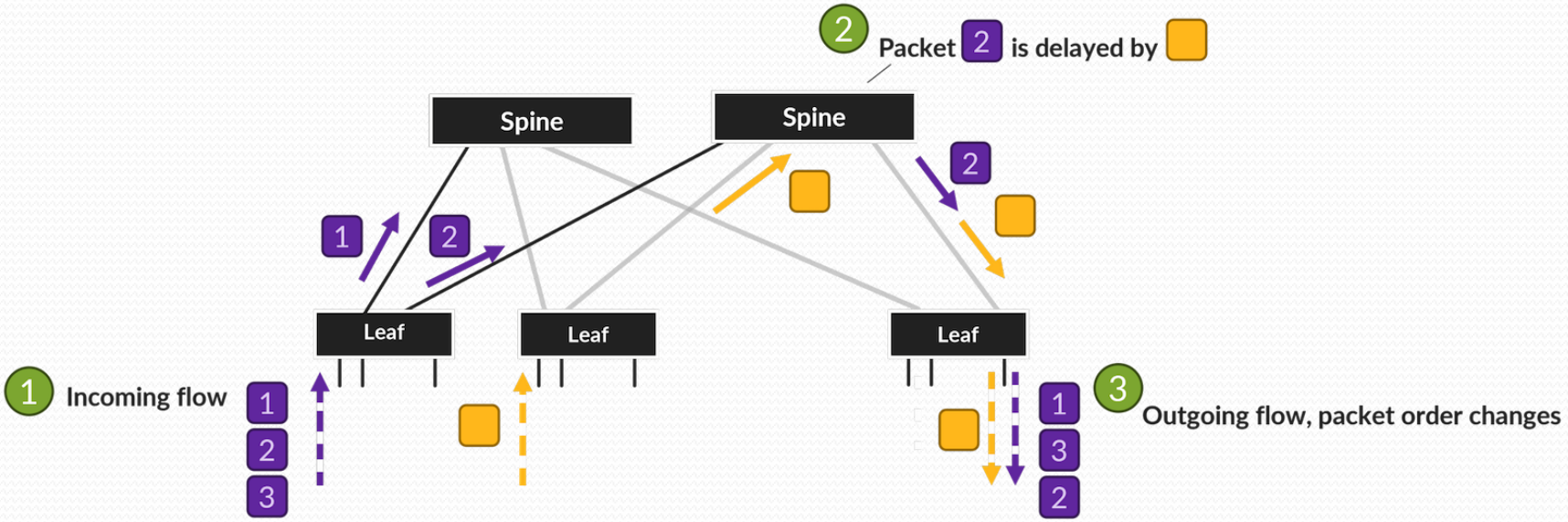


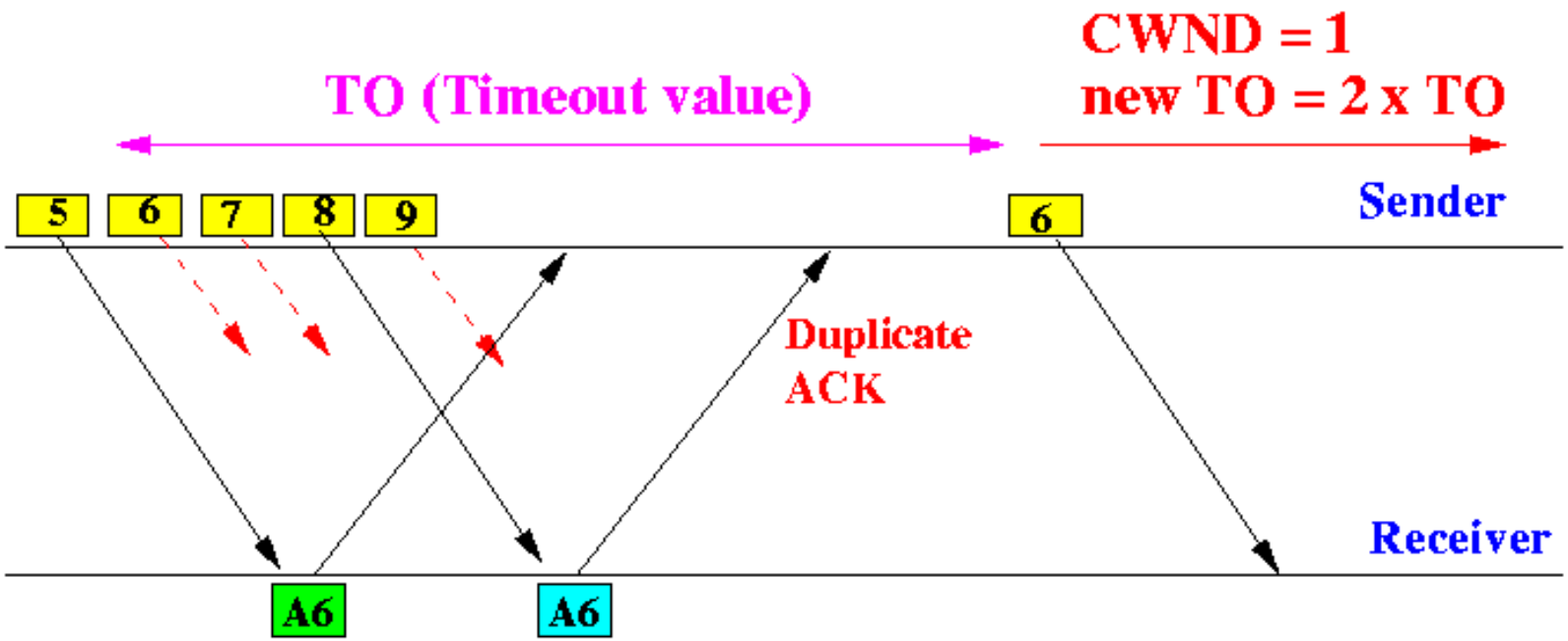
Figure 1: Design space for load balancing.

Per-Packet ?

Though per-packet can achieve optimal performance, but it may cause **TCP reordering**



Dup ACK Again!



TCP Cannot Distinguish Between Packet Loss and Reordering

- Why?
  - TCP does not have any mechanism for re-ordering, as **TCP is originally designed for single path**
  - Dup-ACK causes TCP to assume (which is wrong) that packet has been lost
    - **Timeout: Latency**
    - **Retransmission: Bandwidth waste**

Summary

- **An ideal data center load balancing scheme should:**
  - Break the trade-off between **per-flow** and **per-packet** load balancing
  - Utilize **global congestion information**

Outline

- Background
- **CONGA**
- Implementation and Evaluation
- Review

Design Choice of CONGA

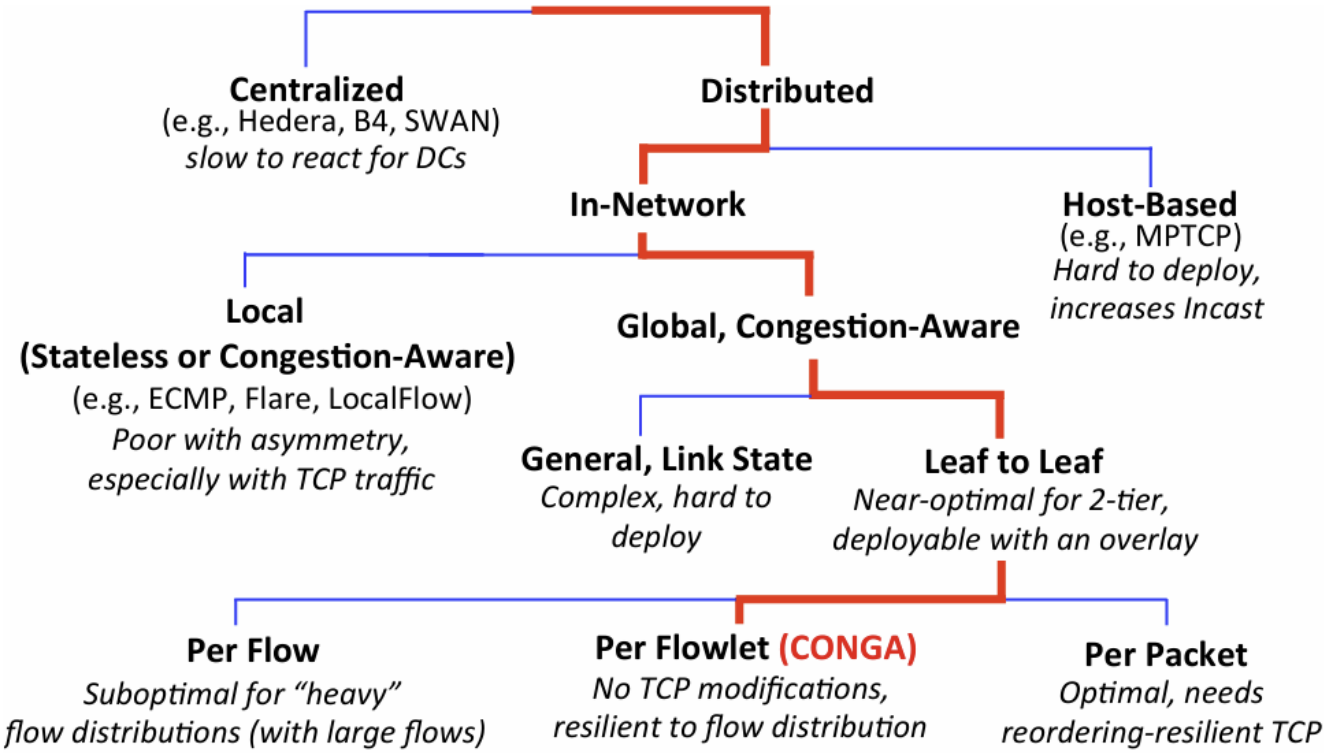


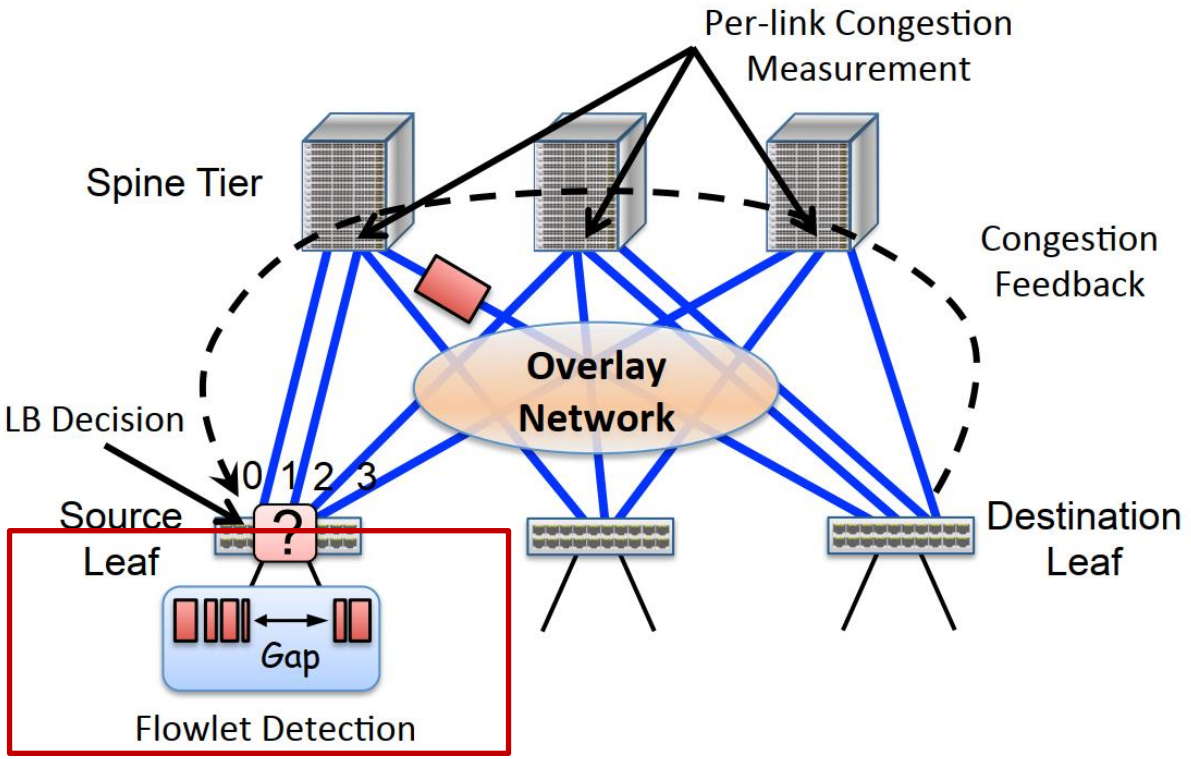
Figure 1: Design space for load balancing.

Design Choice of CONGA

- **Flowlet**-based load balancing
- **Congestion-aware** path selection
- **Leaf-to-Leaf** distributed control

Flowlet

- Flowlets are bursts of packets from a flow that are separated by **large enough gaps**

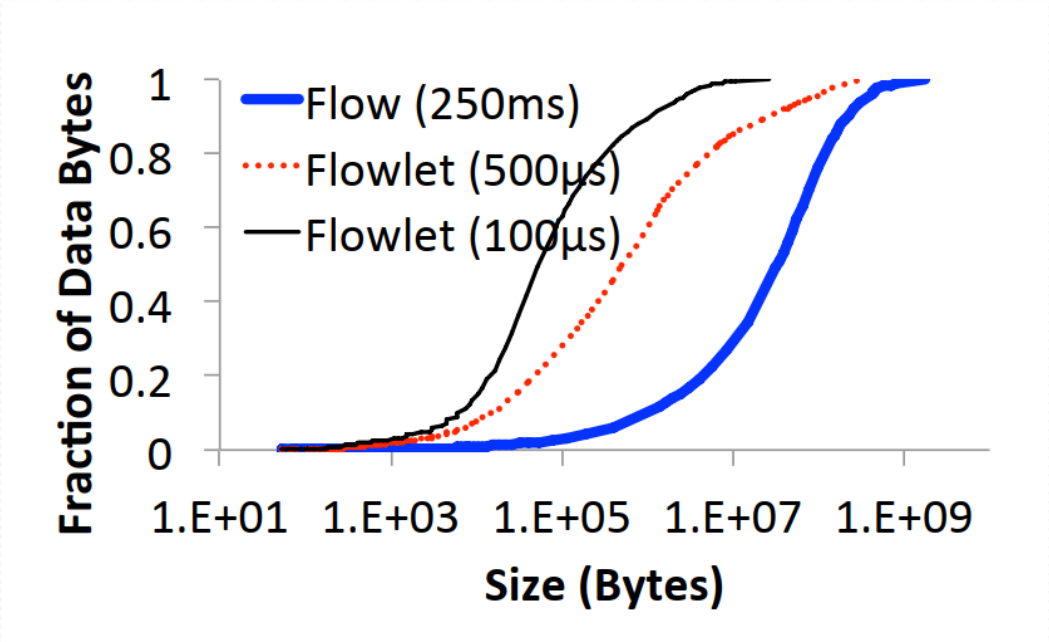


Why Flowlet Exists?

- **Bursty Traffic**
  - TCP Segmentation Offload (TSO)
  - Large Send Offload (LSO)
- Instead of sending packets evenly, the NIC often sends **large bursts**

Flowlet

- CONGA Instruments a production cluster with over **4500** virtualized and bare metal hosts across **~30** racks of servers to obtain packet traces of traffic



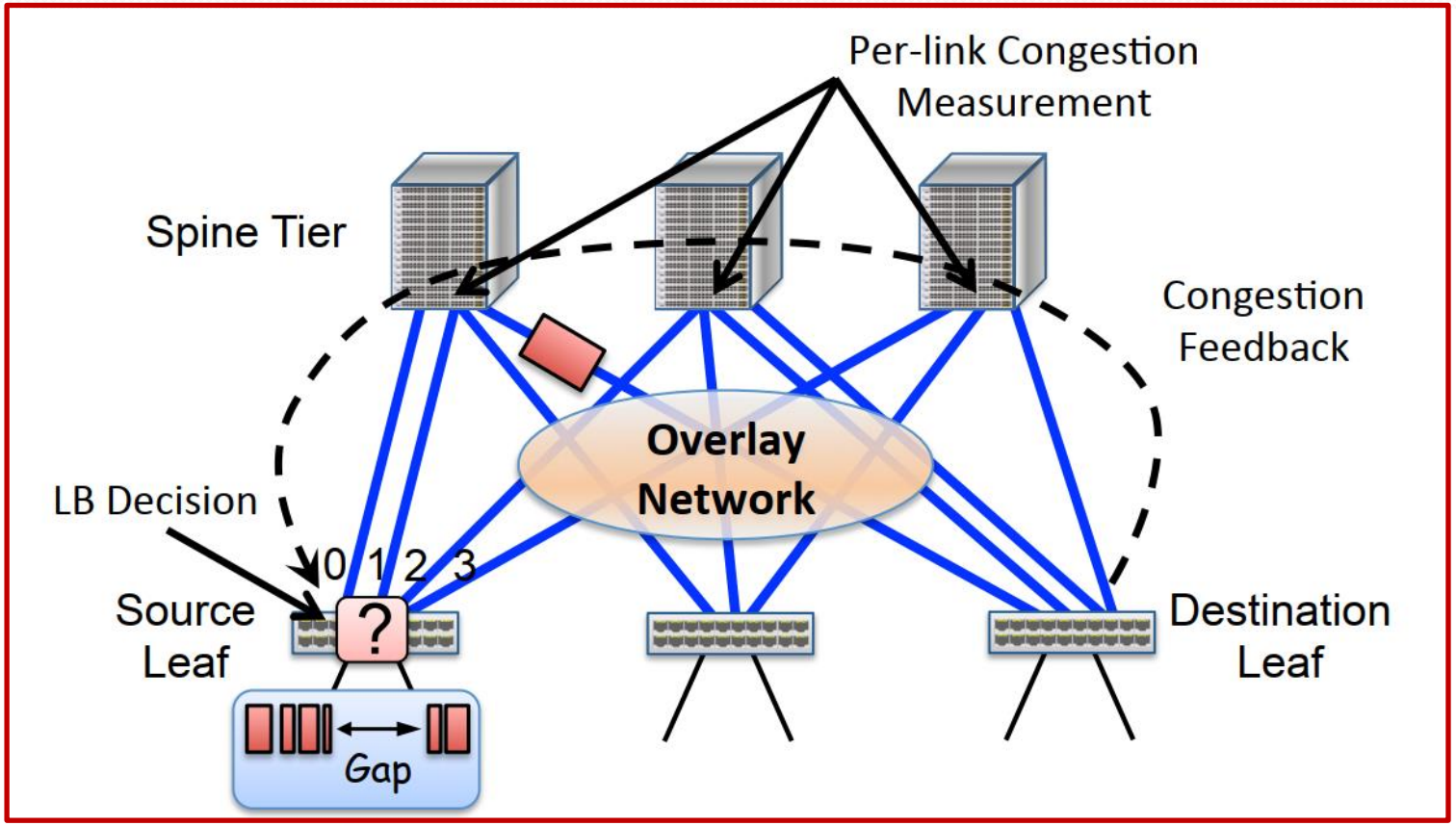
Flowlet

- **Flow (250ms):** Packets are grouped into the same **flow** if the gap between packets is **smaller than 250 ms**.
  A gap larger than **250 ms** indicates the start of a **new flow**.
- **Flowlet (500 µs):** A new flowlet is created if the gap between packets exceeds 500 microseconds.
- **Flowlet (100 µs):** A more aggressive definition: a new flowlet is created when the packet gap exceeds 100 microseconds.

Flowlet

- The x-axis shows the size of traffic units (bytes) in log scale
- The y-axis shows the **fraction of total data bytes** contributed by units smaller than that size
- Flowlets are much smaller.
  - With a 500 µs flowlet gap, 50% of data comes from units smaller than ~200 KB.
  - Even smaller with shorter gaps. With a 100 µs gap, 50% of data comes from units smaller than ~50 KB.

Workflow of CONGA



Workflow of CONGA

- CONGA dynamically assigns each **flowlet** to the **least congested path** using **leaf-to-leaf congestion feedback**
  - Step1: Detect flowlets
  - Step2: Select the best path
    - **Only the first packet of a flowlet triggers a path decision**
  - Step3: Forward packets
  - Step4: Collect congestion information
  - Step 5: Send feedback to the source

Workflow of CONGA

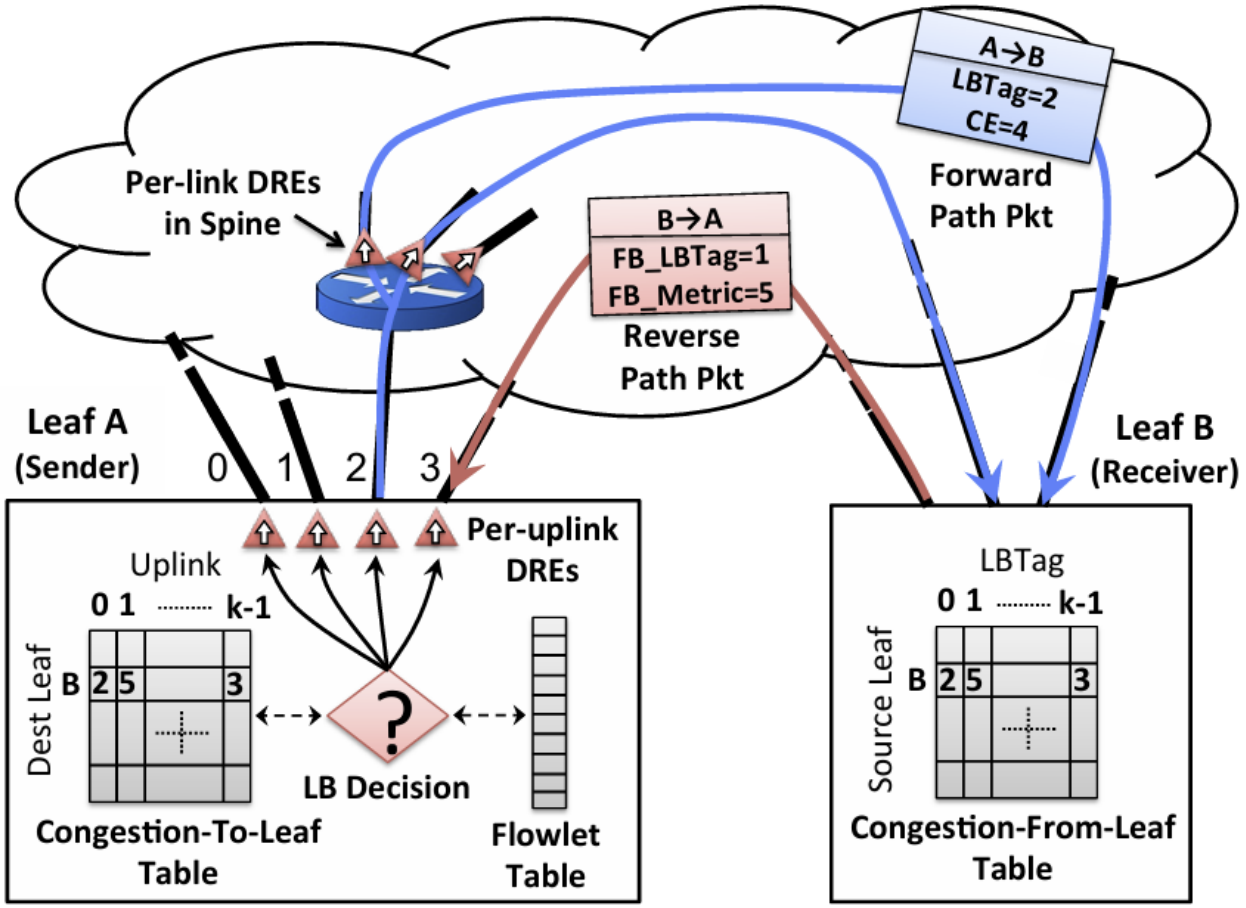


Figure 6: CONGA system diagram.

Packet Tag in CONGA

- CONGA reuses the **VXLAN header (**assume it is like a tunnel/header in header**)** space to carry its load-balancing metadata.
  - LBTag (4 bits):
    - Destination leaf
    - Target spine (next hop)
  - CE(3bits): used by **switches along the packet's path** to convey the extent of congestion
  - FB_LBTag(4bits)andFB_Metric(3bits): used by **destination leaves** to piggyback congestion information back to the source leaves.
    - FB_LBTag indicates the LBTag the feedback is for
    - FB_Metric provides its associated congestion metric

Discounting Rate Estimator (DRE)

- When packets are transmitted:
  - The switch **adds the packet size** to the link load
  - Periodically, the load value **decays over time**
- Why not directly use the queue size?
  - Queue size monitoring is difficult and complex

How to update CE field

- Initially CE = 0
- When the packet traverses a switch:
  - The switch estimates the **current load of the outgoing link** using **DRE**.
  - The switch compares this value with the existing CE in the packet.
  - The CE field is updated as: **CE = max(CE, local_link_load)**
- When the packet reaches the destination leaf switch:
  - The CE field represents the **bottleneck congestion of the entire path**.
  - The destination leaf records this value and later **feeds it back to the source leaf**.

Outline

- Background
- CONGA
- **Implementation and Evaluation**
- Review

Implementation of CONGA

- CONGA is primarily implemented at **leaf switches**.
- Responsibilities of leaf switches:
  - **Source leaf**
    - Detects flowlets
    - Selects the best path (spine)
  - **Destination leaf**
    - Collects congestion information
    - Sends feedback to the source leaf
- Spine switches only perform **simple congestion measurement via DRE**

Implementation of CONGA

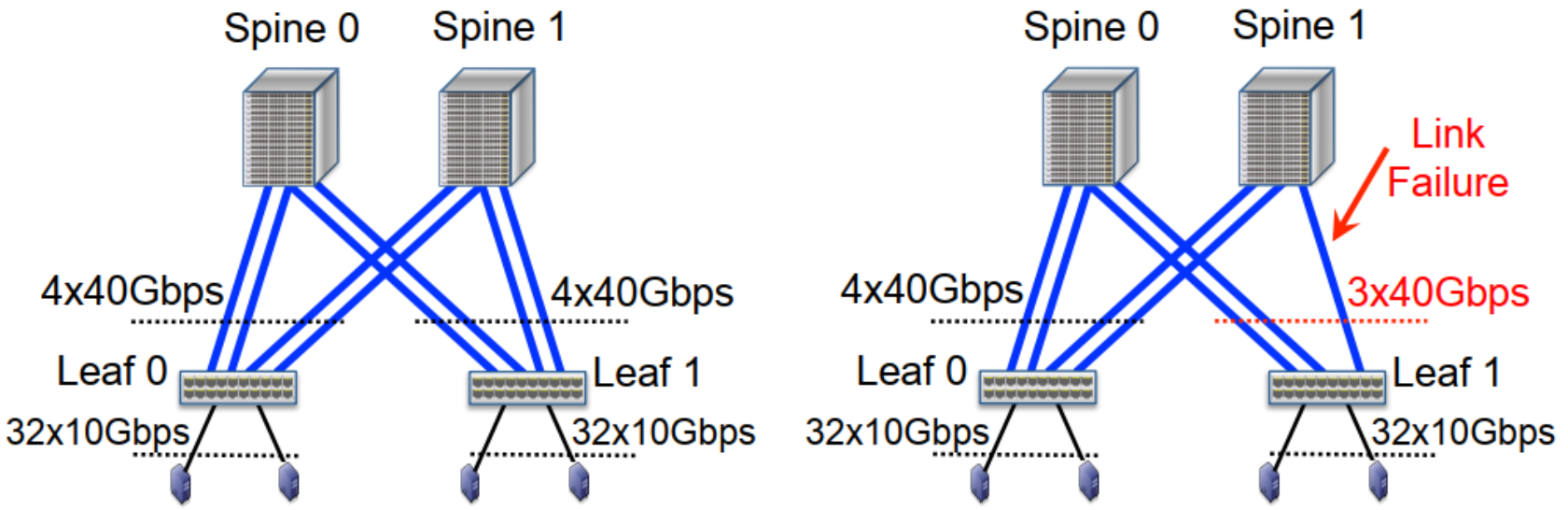
- CONGA requires **modifications to switch ASICs**
- Switches must support:
  - reading and modifying packet headers in the data plane
  - updating the CE field at line rate
  - running the DRE algorithm for each link
  - maintaining per-destination congestion tables
- These operations must be performed per packet at line rate
- **CONGA cannot be deployed on commodity switches without hardware support**
  - The paper is from Cisco

Evaluation of CONGA

- Goal: Evaluate the effectiveness of CONGA in realistic datacenter environments.
- Evaluation uses:
  - **Hardware testbed**
  - **Packet-level simulations**
- Compared schemes:
  - **ECMP**
  - **MPTCP**
  - **CONGA**
  - **CONGA-Flow (per-flow variant)**

CONGA-Flow uses a large flowlet timeout so that one decision ≈ one flow.

Testbed Setup



(a) Baseline (no failure) (b) With link failure

Figure 7: Topologies used in testbed experiments.

Workloads

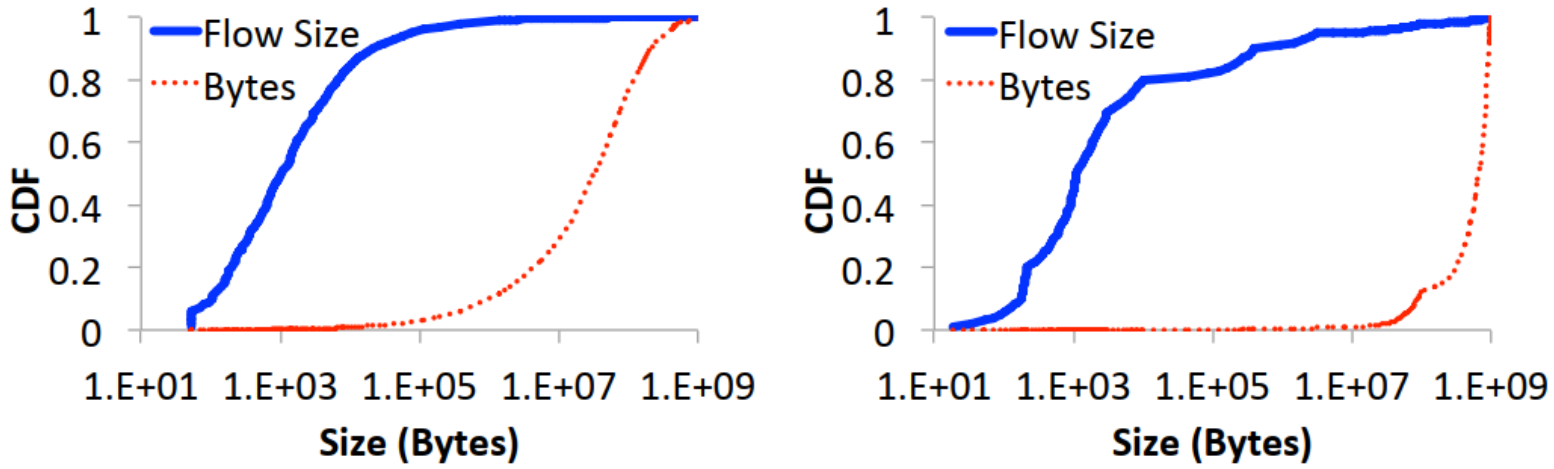
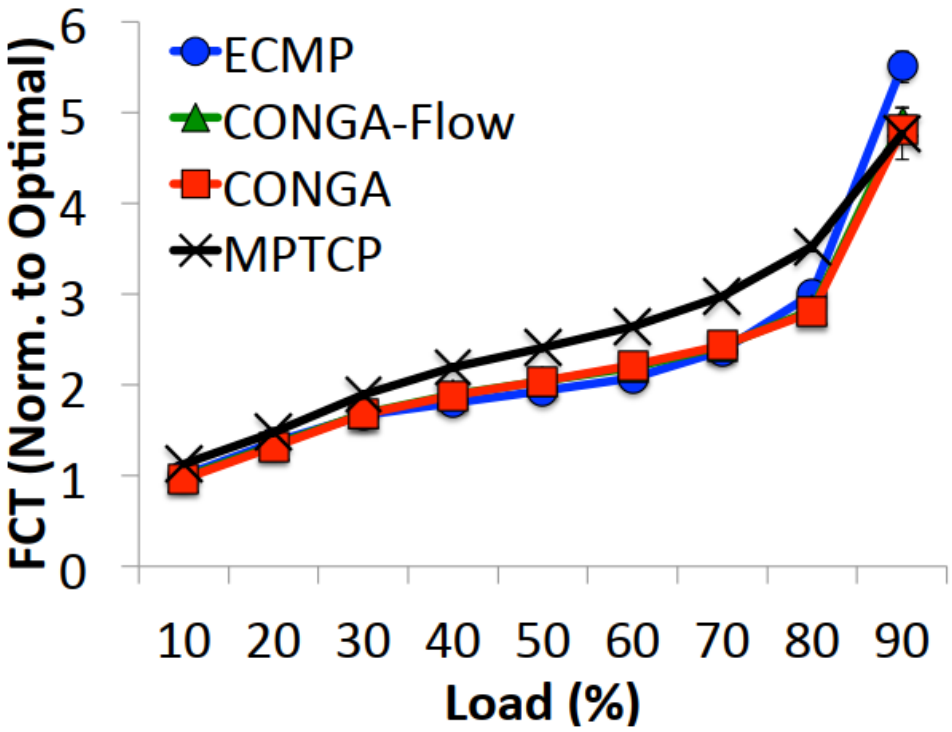


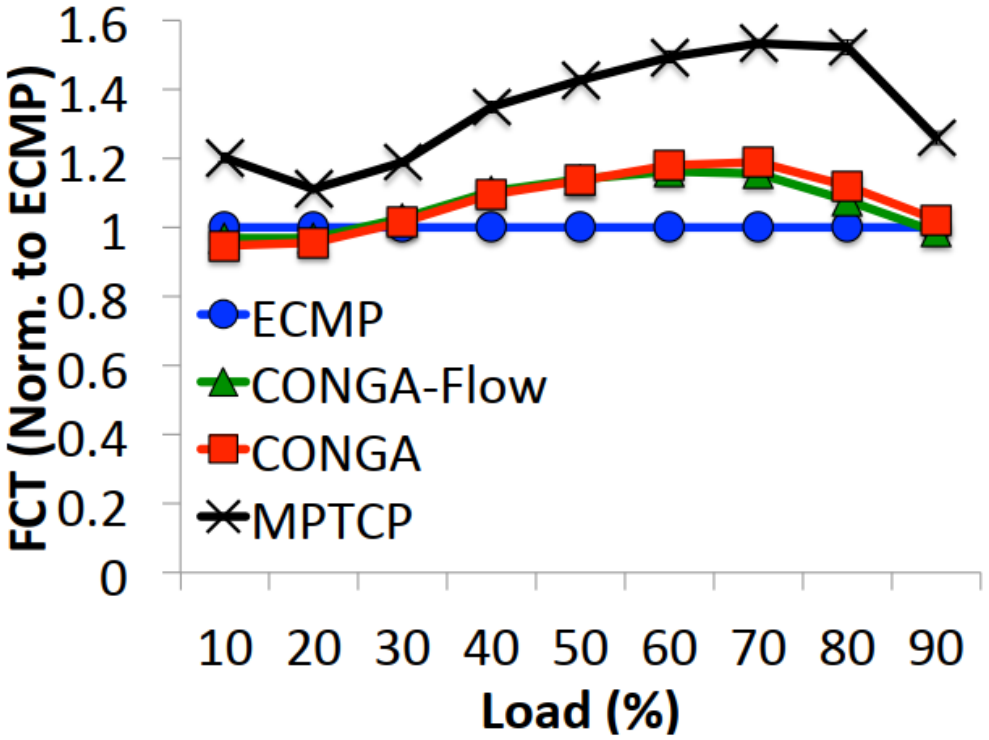
Figure 8: Empirical traffic distributions. The Bytes CDF shows the distribution of traffic bytes across different flow sizes.

Flow Completion Time – Enterprise Workload



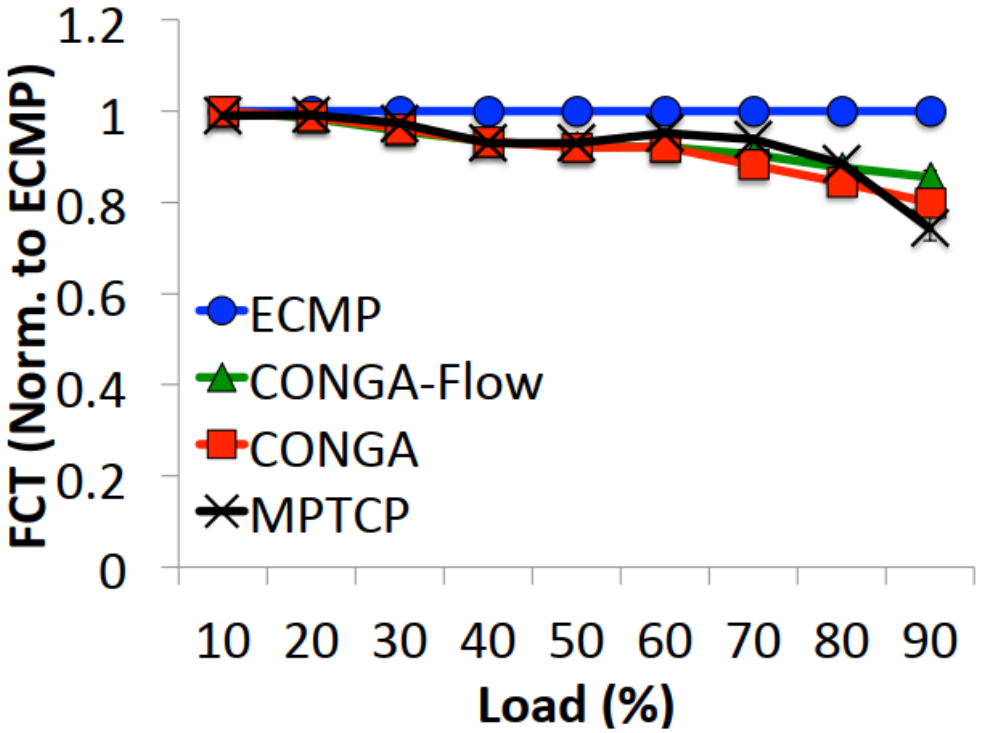
(a) Overall Average FCT

Flow Completion Time – Enterprise Workload



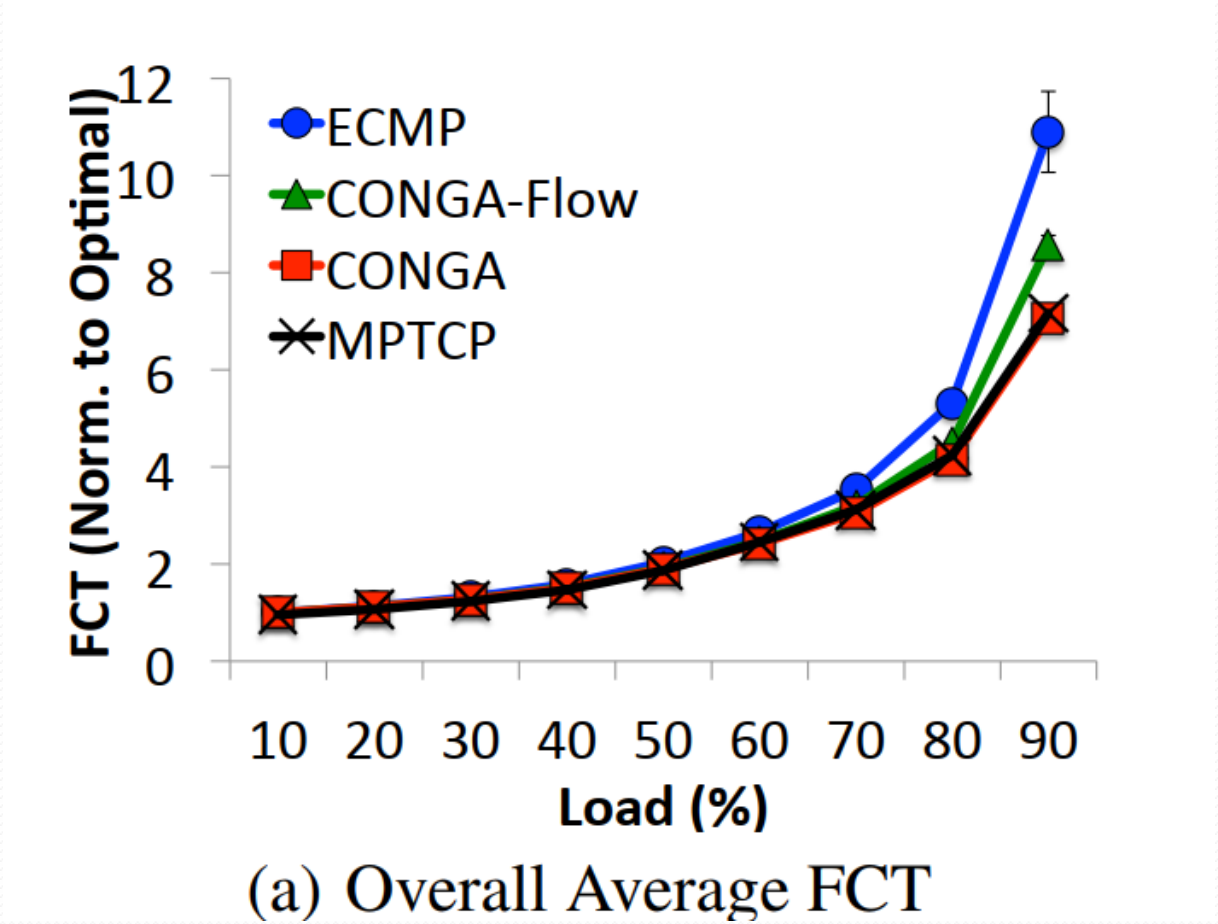
(b) Small Flows (<100KB)

Flow Completion Time – Enterprise Workload



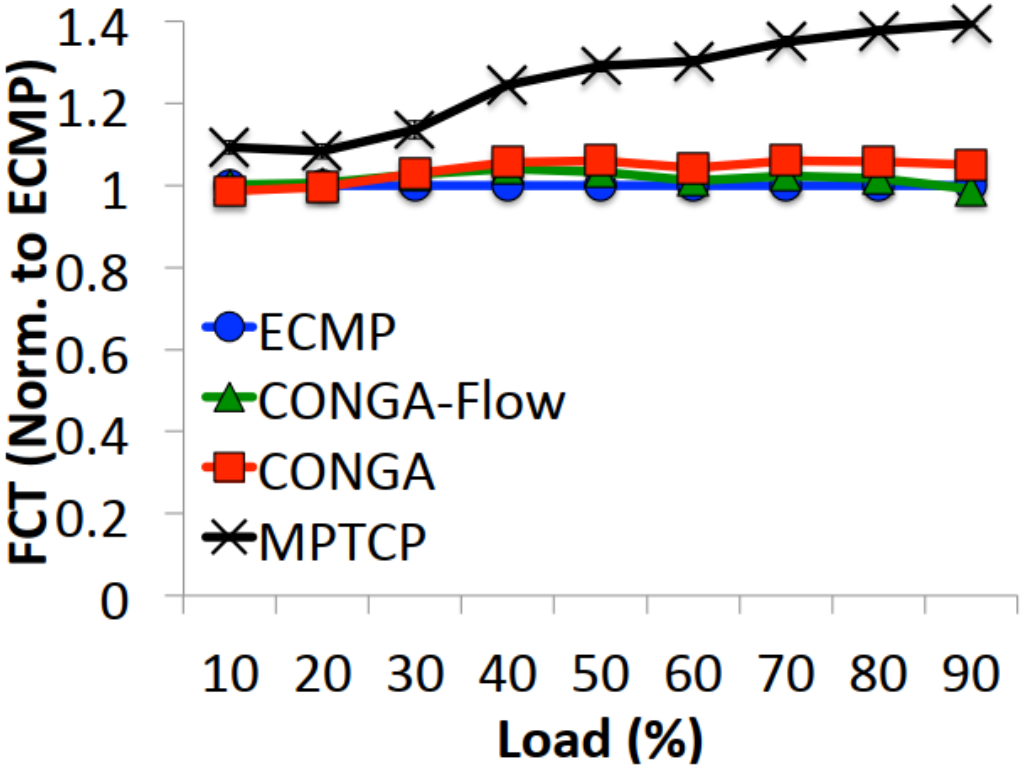
(c) Large Flows (> 10MB)

Flow Completion Time – Data Mining Workload



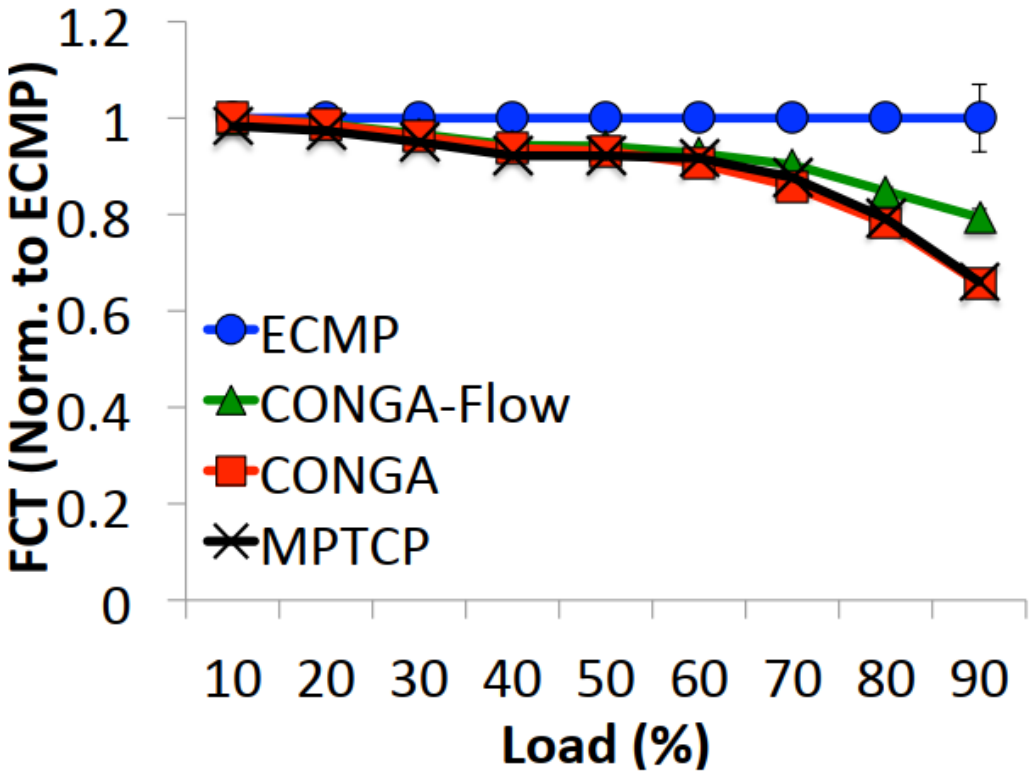
(a) Overall Average FCT

Flow Completion Time – Data Mining Workload



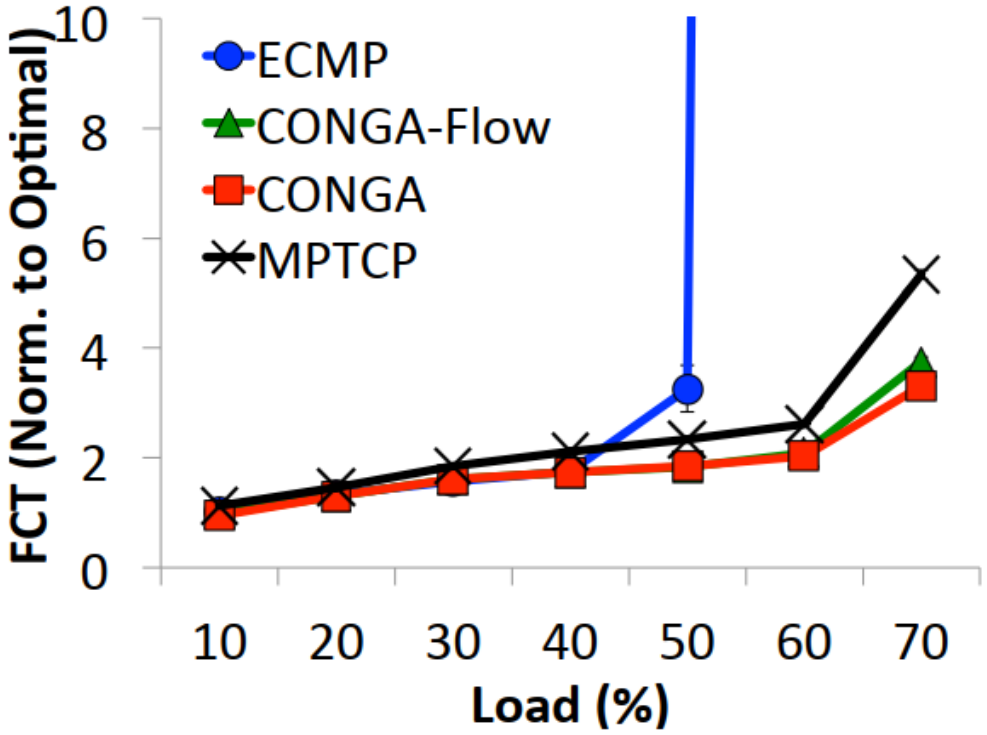
(b) Small Flows (< 100KB)

Flow Completion Time – Data Mining Workload



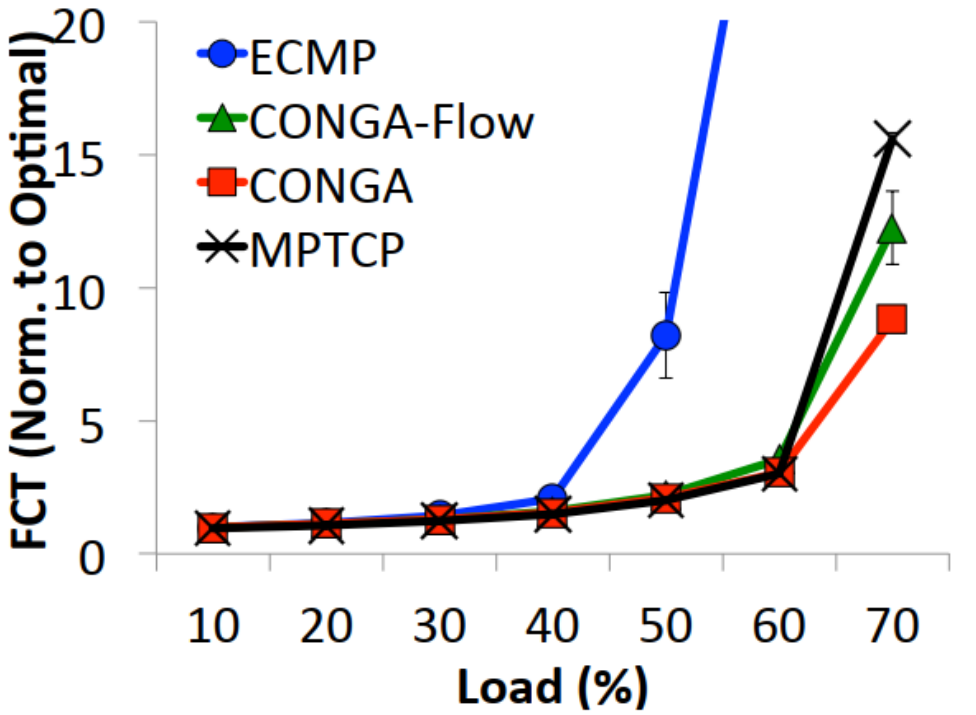
(c) Large Flows (> 10MB)

Impact of Link Failure



(a) Enterprise workload: Avg FCT

Impact of Link Failure

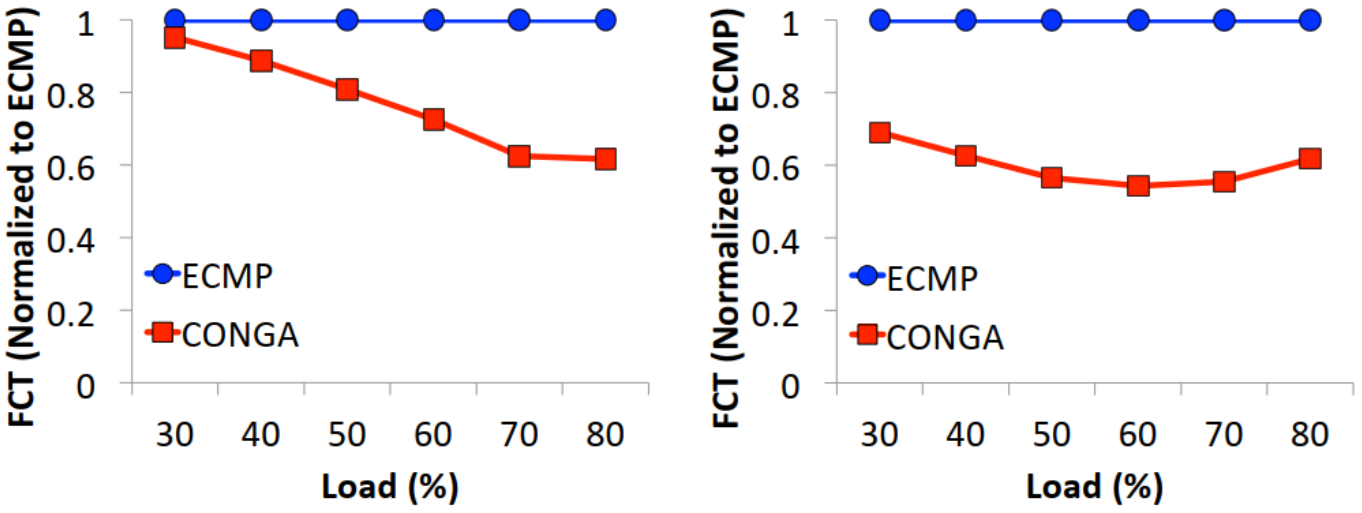


(b) Data-mining workload: Avg FCT

Large Scale Simulation

- We used OMNET++ and the Network Simulation Cradle to port the actual Linux TCP source code (from kernel 2.6.26) to our simulator.
- 384 servers, 8 leaf switches, and 12 spine switches, and for oversubscription ratios ranging from 1:1 to 5:1

Large Scale Simulation



(a) 10Gbps access links (b) 40Gbps access links

Figure 15: Overall average FCT for a simulated workload based on web search [4] for two topologies with 40Gbps fabric links, 3:1 oversubscription, and: (a) 384 10Gbps servers, (b) 96 40Gbps servers.

Outline

- Background
- CONGA
- Implementation and Evaluation
- **Review**

Review

- We introduce the background of load balance
- Why existing load balance techniques are not optimal
  - ECMP: Particularly bad for asymmetry topology
  - Per-packet: TCP Re-ordring
  - MPTCP: Not suitable for data centers
    - More incast
    - Too many available paths

Review

- CONGA:
  - **Flowlet**-based load balancing
  - **Congestion-aware** path selection
  - **Leaf-to-Leaf** distributed control
- Implementation of CONGA
  - Require switch ASIC modification
    - Too complicated
    - Impractical with commodity switches

Review

- Why not host-based solutions?
  - Hosts have limited visibility than switches
  - But we have a paper:
    - No switch modifications
    - Totally host based
    - More design for grey failures
      - Random drop
      - Packet black hole

Hong Zhang, **Junxue Zhang**, Wei Bai, Kai Chen, Chowdhury Mosharaf, **Resilient Datacenter Load Balancing in the Wilds**, *SIGCOMM 2017*