TECHNISCHE HOCHSCHULE DEGGENDORF

Datacenter Network Programming

Loadbalancing with ECMP and Conga

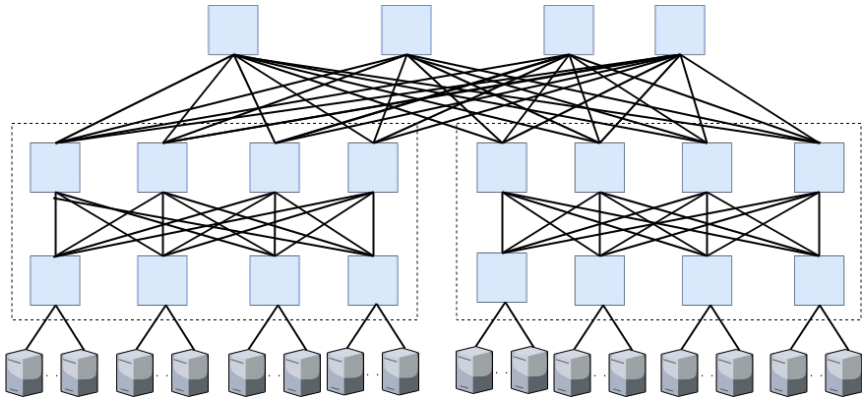
Today's agenda

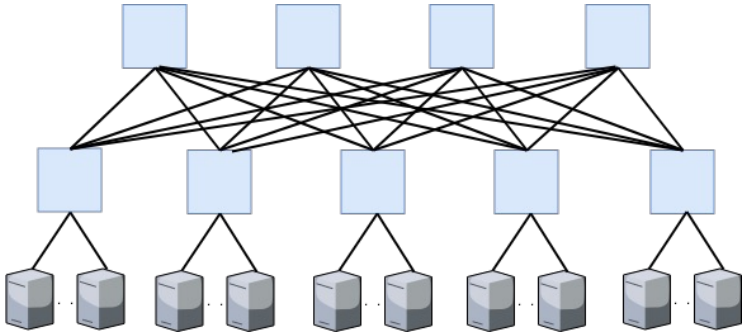
- **ECMP**
  - Motivation
  - Background
  - Load balancing solutions – Landscape
  - ECMP - Introduction
  - ECMP - Problems
  - ECMP – Challenges
  - ECMP – P4 Exercise
- **CONGA**
  - Motivation
  - Background
  - Design
  - Implementation

ECMP

Loadbalancing - Motivation

- **Data centers use multi-stage topologies such as Clos or fat-tree to deliver high bandwidth for internal data exchange.**

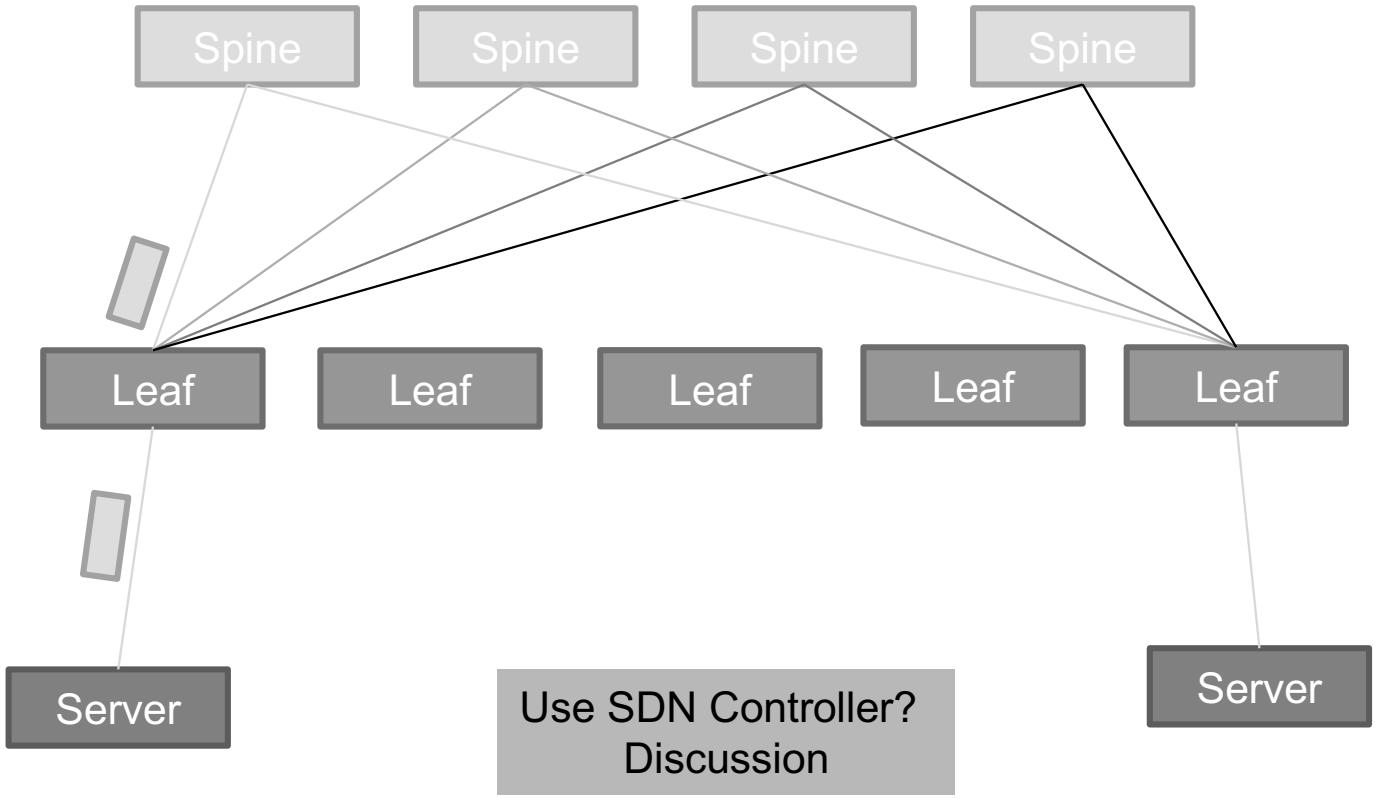




  - Multiple equal-cost paths for a pair of hosts
  - High bisection bandwidth
  - Volatile traffic
  - Multiple tenants

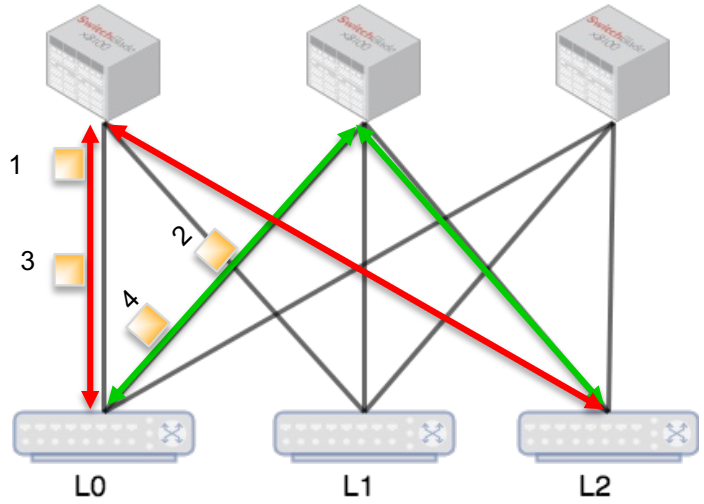
Loadbalancing - Motivation

Which path should we take?



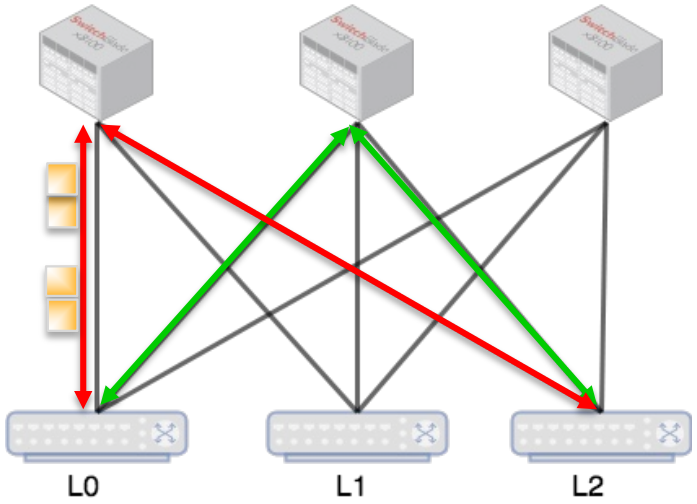
Load balancing granularity

- Three main ways to split traffic, by **packet**, flow or flowlet.
  - The main problem of assigning packets to different paths is that the packets experience different delays and therefore we can have TCP reordering. →Impact on TCP Throughput
  - This method is very fine-granular since the reaction size is the size of packets so it is easier to balance the load proportionally.
  - Control/vs Data Plane?



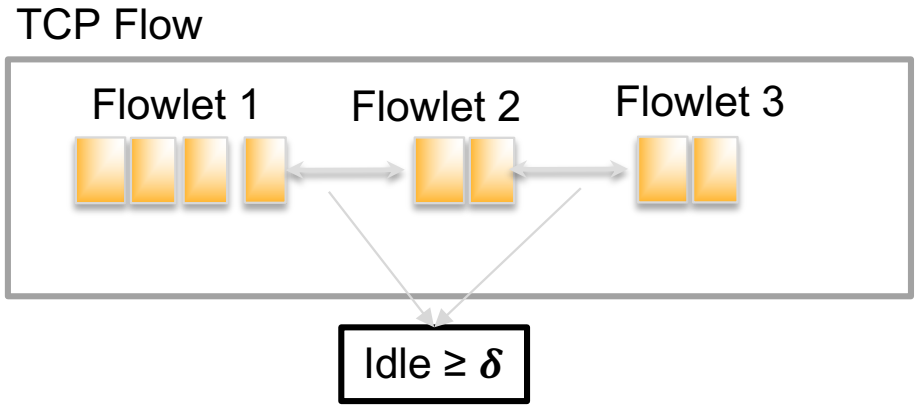
Load balancing granularity

- Three main ways to divide traffic, by packet, **flow** or flowlet.
  - Assign TCP Flows to one specific path, so all packets related to that flow will traverse the same path.
    → No packet reordering
  - This method is less finegranular since it is difficult to balance the load proportionally by not knowing the size of each flow.
  - Apply to elephant and/or mice flows?
  - Control versus Data Plane?



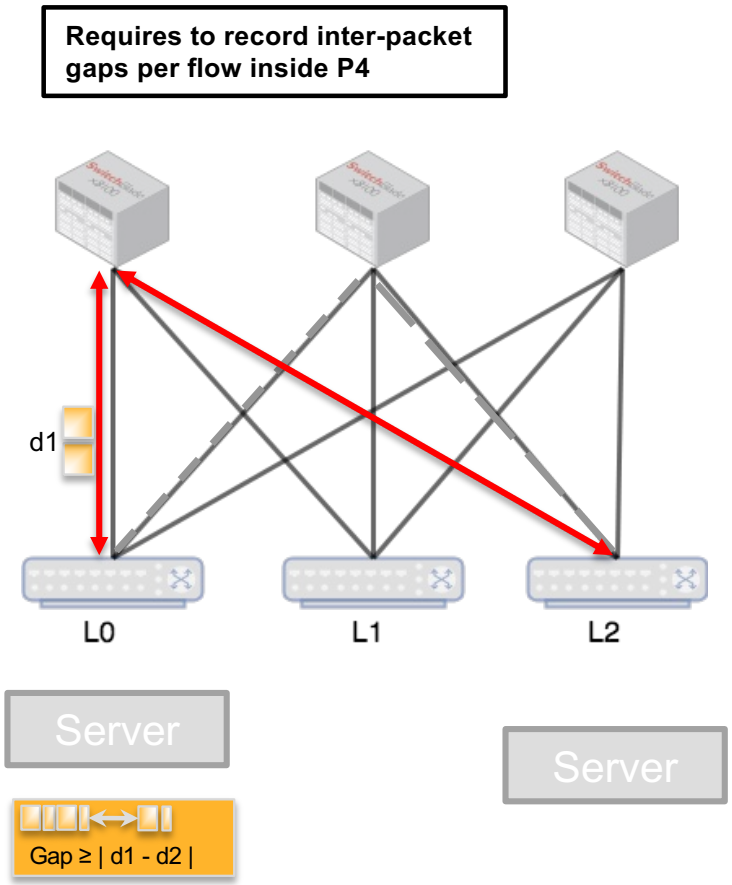
Load balancing granularity

- Three main ways to divide traffic, by packet, flow or **flowlet**.
  - Flowlet is a burst of packets from the same TCP flow separated by a certain time δ from other burst.
  - Large flows can be splited into many small flowlets
  - In general, flowlet load balancing will not cause TCP reordering if gap is large enough



Load balancing granularity

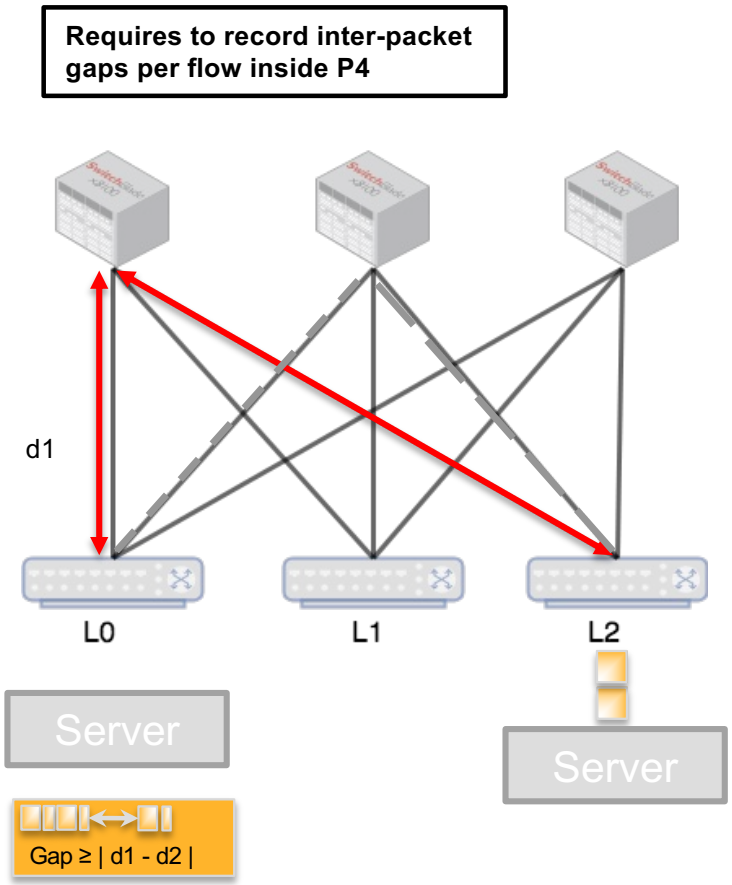
- Three main ways to divide traffic, by packet, flow or **flowlet**.



*Flowlet Switching (Kandula et al '04)

Load balancing granularity

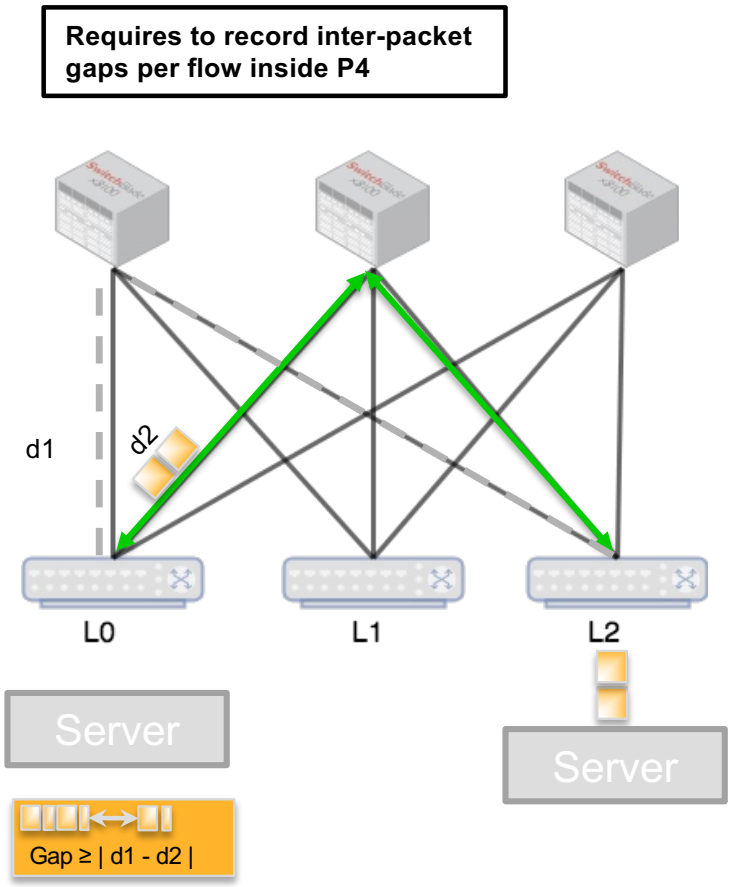
- Three main ways to divide traffic, by packet, flow or **flowlet**.



*Flowlet Switching (Kandula et al '04)

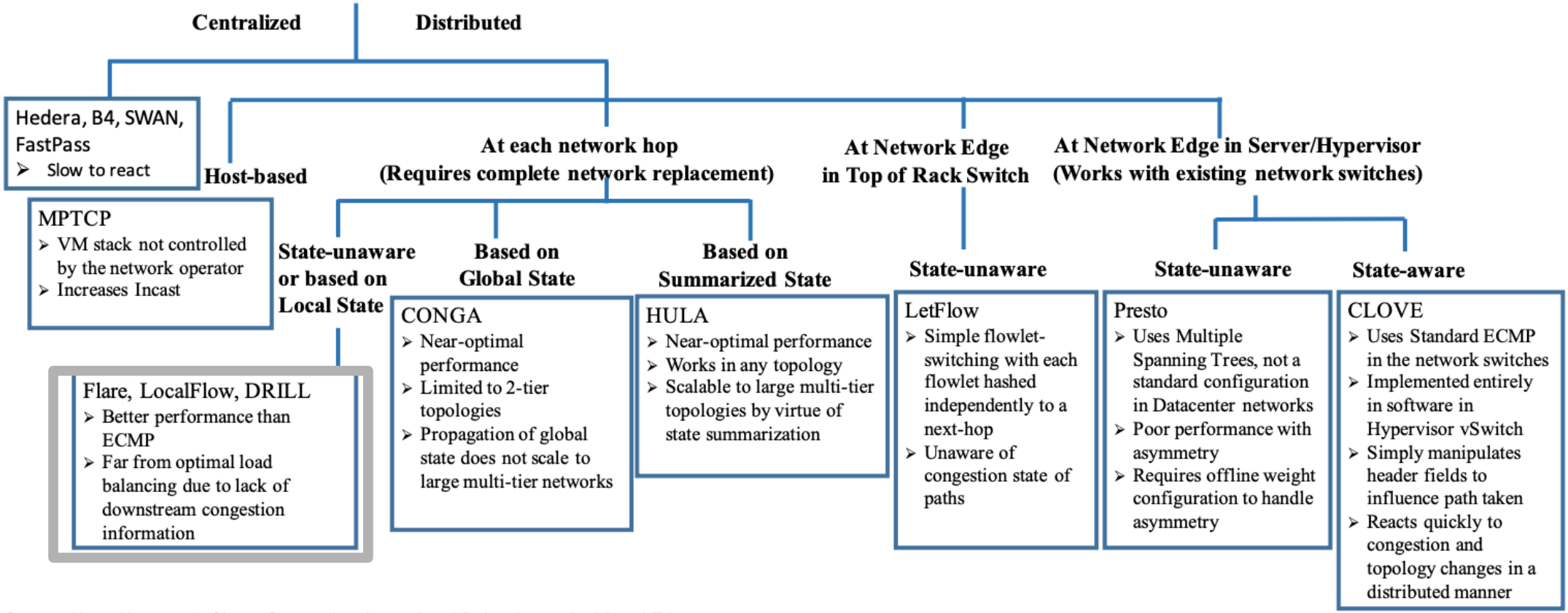
Load balancing granularity

- Three main ways to divide traffic, by packet, flow or **flowlet**.
  - In general, flowlet load balancing will not cause TCP reordering if gap is larger than RTT
  - Tradeoff between gap and #flowlets per flow
  - Tradeoff between gap and load-balancing opportunities



*Flowlet Switching (Kandula et al '04)

Loadbalancing - Background



Source: Naga Katta et al. Clove: Congestion-Aware Load Balancing at the Virtual Edge

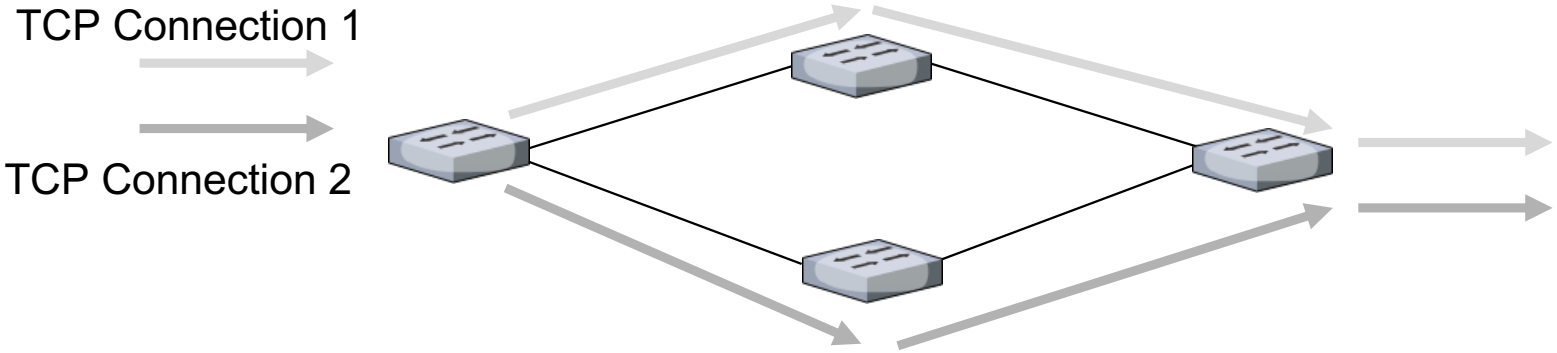
ECMP - Background

- **IP routing: Single path**
- **MPTCP: Multipath at transport layer (End host) (Problems?)**
- **SDN: can route each flow individually (Problems?)**
- **Equal Cost Multipath (ECMP) enable the usage of multiple equal cost paths from the source node to the destination node in the network.**
- **The advantage of using this algorithm is that the traffic can be split more uniformly to the whole network avoiding congestion, increasing bandwidth utilisation and preserving packet order.**

ECMP - Background

- **Equal-cost multipath (ECMP) is a network routing strategy that allows for traffic of the same session, or flow—that is, traffic with the same source and destination—:**
  - To be transmitted across multiple paths of equal cost.
  - To load balance traffic and increase bandwidth by fully utilizing otherwise unused bandwidth on links to the same destination.
- **When forwarding a packet, the next-hop path to use is taken into account the packet header fields that identify a flow.**

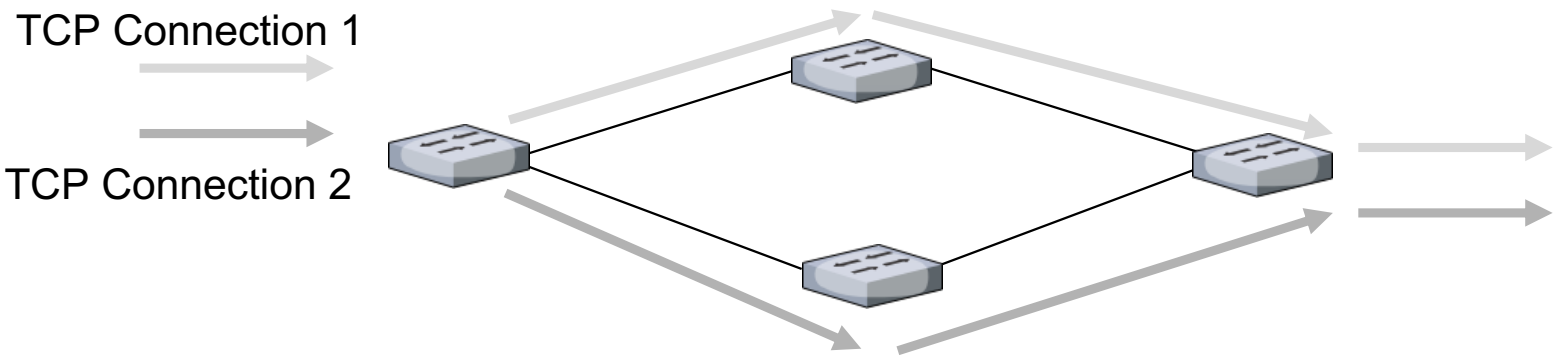
Equal-cost multipath routing (ECMP)



- **ECMP**
  - Multipath routing strategy that splits traffic over multiple paths for load balancing
- **Why not just round-robin packets?**
  - Reordering (lead to triple duplicate ACK in TCP?)
  - Different RTT per path (for TCP RTO)…
  - Different MTUs per path

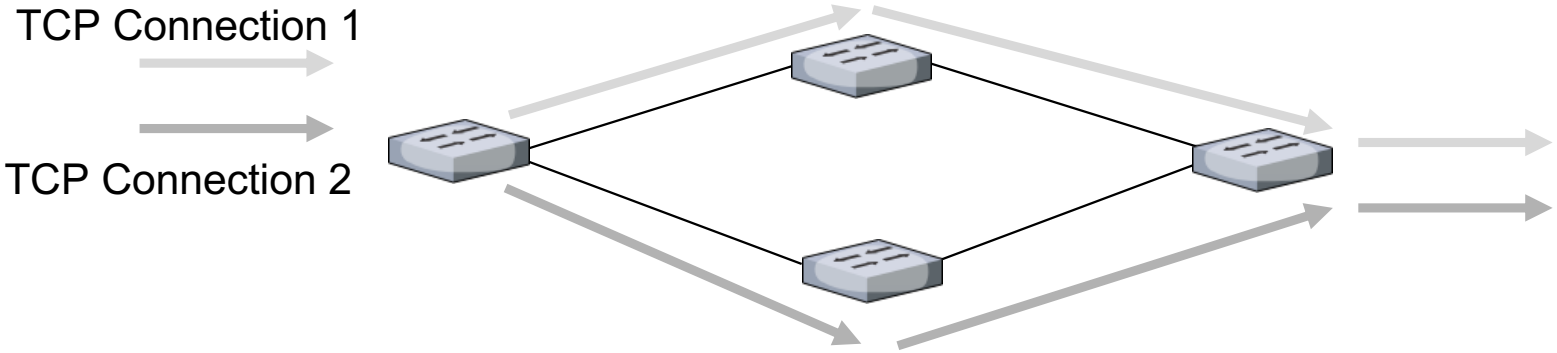
Source: http://www.cs.princeton.edu/courses/archive/spring11/cos461/

Equal-cost multipath routing (ECMP)



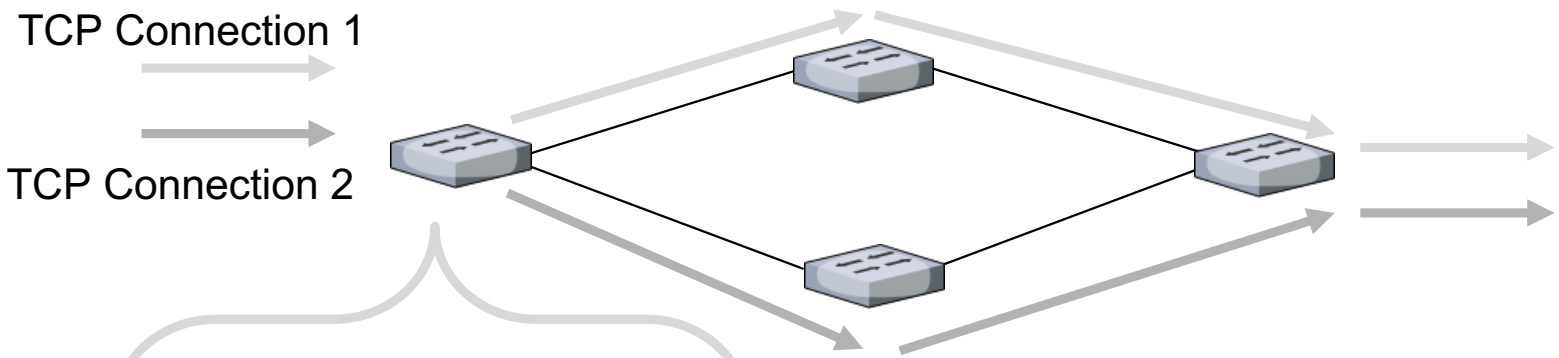
- ECMP implementation enables us to evenly spread flows across the network leading to best utilization of multiple links towards the destination.
- It reduces the time taken in data transfer up to large extent due to less congestion.

Equal-cost multipath routing (ECMP)

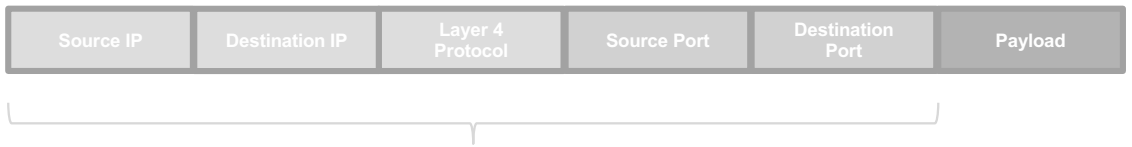


- **Path-selection via hashing**
  - In hashed routing a modulo <n> hash of various header information is performed. The ECMP employs a hashing function, for example, a CRC-16, by using some header fields to perform the calculation.
  - We perform modulus operation on the hash value with total number of best possible routes. The output of this modulus indicates which of the paths to use.

Equal-cost multipath routing (ECMP)

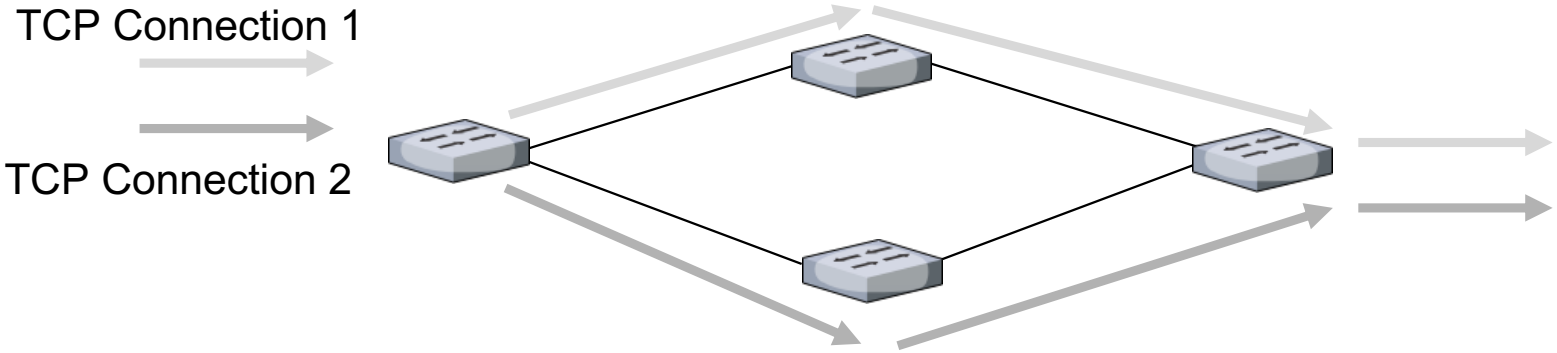



```
Header fields {
hash_input[0] = ip.srcip
hash_input[1] = ip.dstip
hash_input[2] = ip.protocol
hash_input[3] = source port
hash_input[4] = destination
port
}
```



ECMP Hash Fields

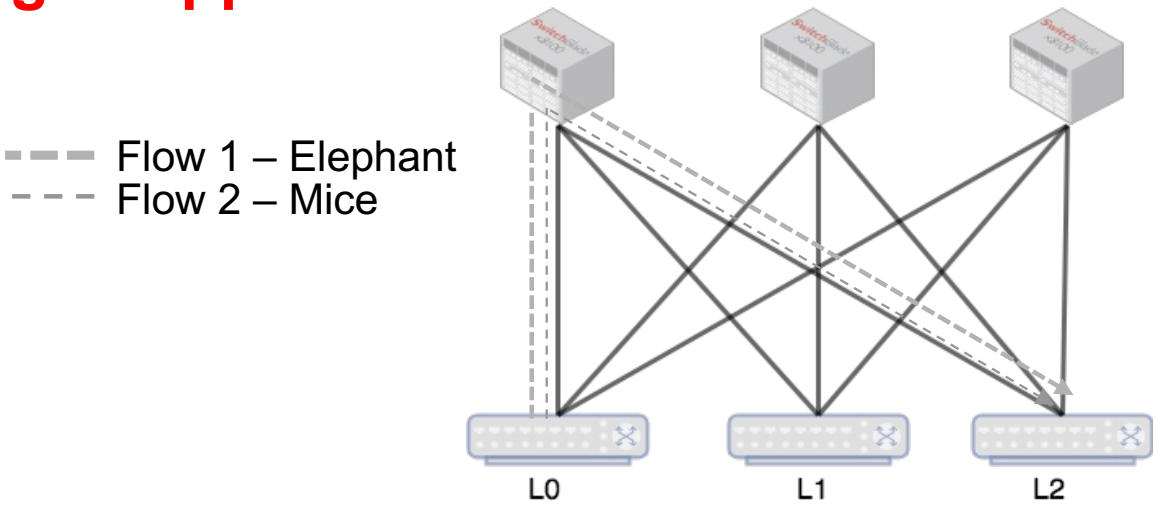
Equal-cost multipath routing (ECMP)



- Ideally, the header information and hash algorithm are chosen such that an arbitrary traffic pattern will be equally distributed on the paths while ensuring that a single flow remains on the same path for the life of the flow in order to preserve inorder packet delivery.

ECMP - Problems

- **ECMP is static and agnostic to congestion**
- **Hash collisions migh happen**
- **Coarse-grained**



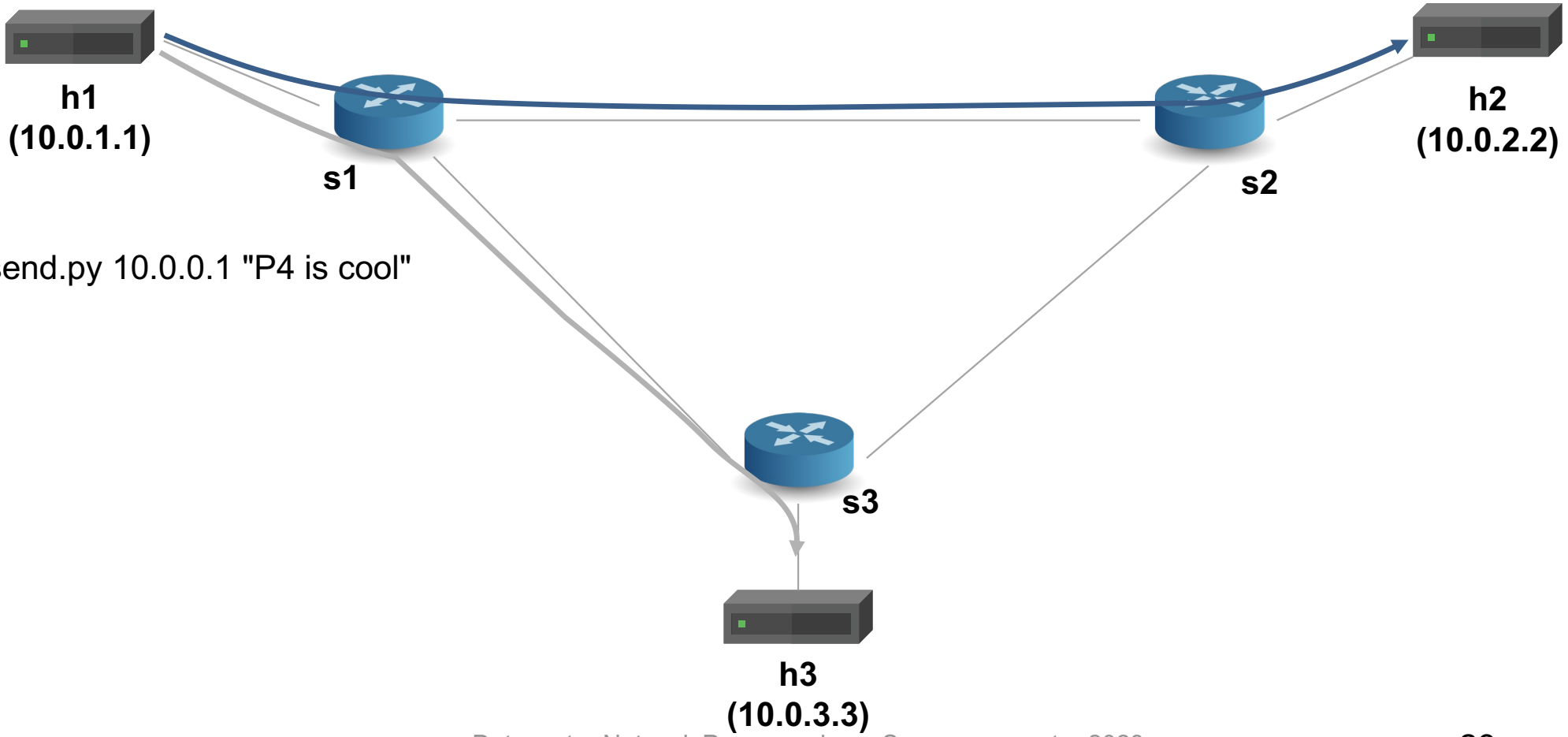
- **The tail latency increases when we have flows of different characteristics, for example, mice and elephant flows, traversing on the same path.**

Challenges

- **How can we improve load balancing in data center networks?**
- **Scalable handle millions of flows traversing numerous links**
- **Smart enough to avoid congestion in the network dynamically**
- **Avoid overhead or bottleneck on the monitoring technique**

ECMP - Exercise

ECMP: Topology



Hashes and Random Numbers – Stateless Externs

Definition

```
/* Definition in v1model.p4 */

enum HashAlgorithm {
    crc32,
    crc32_custom,
    crc16,
    crc16_custom,
    random,
    identity
}

extern void hash<T, D>(out T              result,
                       in  HashAlgorithm  algo,
                       in  T              base,
                       in  D              data,
                       in  T              max);

extern void random<T>(out T result, in T lo, in T hi);

extern Checksum16 {
    Checksum16();
    bit<16> get<D>(in D data);
}
```

Usage (Computing a Flow Hash)

```
bit<10> flow_hash;

if (hdr.ipv4.isValid()) {
    hash(flow_hash, HashAlgorithm.crc32,
         10w0,
         { hdr.ethernet.dstAddr, hdr.ethernet.srcAddr,
           hdr.ipv4.srcAddr, hdr.ipv4.dstAddr,
           hdr.ipv4.protocol },
         10w1023);
} else if (hdr.ipv6.isValid()) {
    hash(flow_hash, HashAlgorithm.crc32_custom,
         10w0,
         { hdr.ethernet.dstAddr, hdr.ethernet.srcAddr,
           hdr.ipv6.srcAddr, hdr.ipv6.dstAddr,
           hdr.ipv6.nextHdr },
         10w1023);
} else {
    hash(flow_hash, HashAlgorithm.crc16,
         10w0,
         { hdr.ethernet.dstAddr, hdr.ethernet.srcAddr,
           hdr.ethernet.etherType },
         10w1023);
}
```

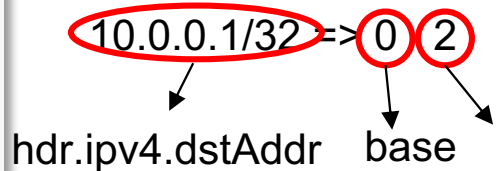
https://github.com/p4lang/p4c/blob/master/p4include/v1model.p4
https://p4.org/assets/p4_d2_2017_p4_16_tutorial.pdf

ECMP - Ingress Tables

```
table ecmp_group {
      Key = {
          hdr.ipv4.dstAddr: lpm;
      }
      actions = {
              drop;
              set_ecmp_select;
      }
      size = 1024;
}

table ecmp_nhop {
       key = {
           meta.ecmp_select: exact;
       }
       actions = {
               drop;
               set_nhop;
       }
       size = 2;
}
```

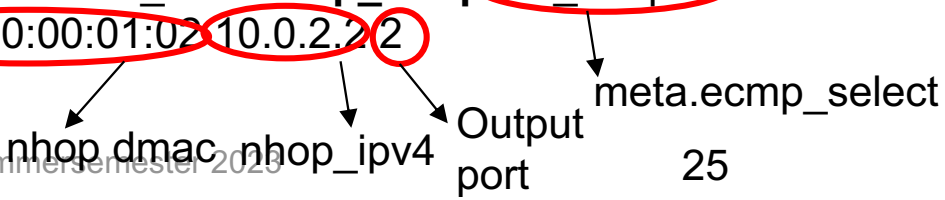
- **ecmp_group** – either drop the packet or selects the ecmp so can be use it to make the forwarding decision

Example: table_add **ecmp_group** set_ecmp_select



- **ecmp_nhop** – Either drops the packet or routes de packet to the next hop based on the ecmp_select value

Example: table_add **ecmp_nhop** set_nhop 0 =>
00:00:00:00:01:02 10.0.2.2 2



ECMP – Actions

```
struct standard_metadata_t {
control MyIngress(
           inout headers hdr,
           inout metadata meta,
           inout standard_metadata_t standard_metadata)
{
       action drop() {
           mark_to_drop();
}

       action set_ecmp_select(bit<16> ecmp_base, bit<32>
ecmp_count) {

     /* TODO: hash on 5-tuple and save the hash result
in meta.ecmp_select so that the ecmp_nhop table can use
it to make a forwarding decision accordingly */
}

       action set_nhop(bit<48> nhop_dmac, bit<32>
nhop_ipv4, bit<9> port) {
         hdr.ethernet.dstAddr = nhop_dmac;
         hdr.ipv4.dstAddr = nhop_ipv4;
         standard_metadata.egress_spec = port;
         hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
}
```

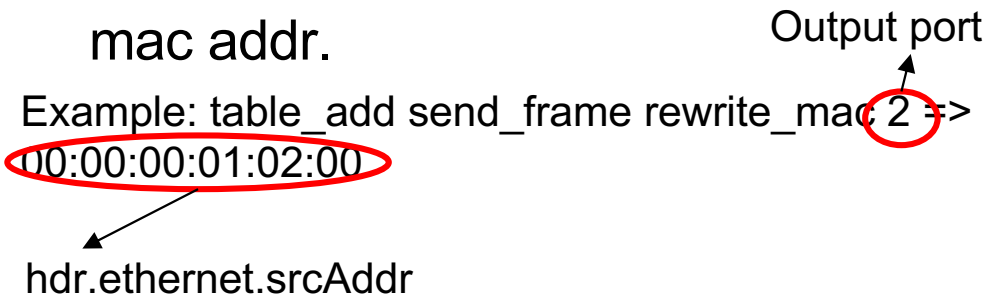
- **`set_ecmp_select`** – perform the hash and return the modulus of it
- **`drop`** – Drops the packet
- **`set_nhop`** – It will rewrite the dmac Addr, IP dst addr, decrease the TTL and forward the packet to the specified port

Example to perform a hash: https://github.com/p4lang/p4c/blob/master/testdata/p4_16_samples/hash-bmv2.p4

ECMP - Egress Tables and Actions

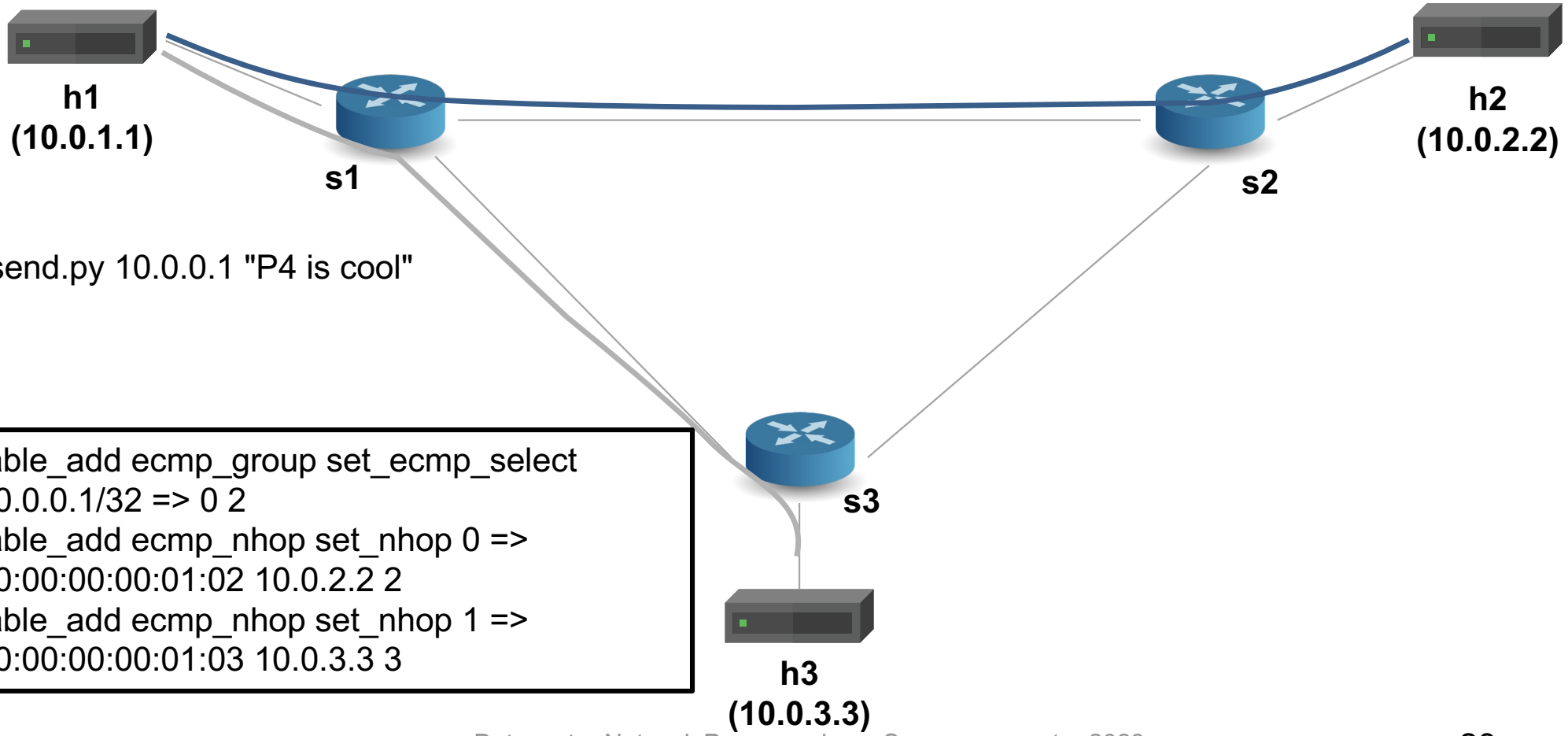
```
table send_frame {
     Key = {
         standard_metadata.egress_port: exact;
     }
     actions = {
             rewrite_mac;
             drop;
     }
     size = 256;
}

action rewrite_mac(bit<48> smac) {
      hdr.ethernet.srcAddr = smac;
}

action drop() {
       mark_to_drop();
}
```

- **`send_frame`** – either drop the packet or rewrite the src mac addr.
- **`rewrite_mac`** – We rewrite the src mac addr.



- **`drop`** – We drop the packet

ECMP: Topology




```
table_add ecmp_group set_ecmp_select
10.0.0.1/32 => 0 2
table_add ecmp_nhop set_nhop 0 =>
00:00:00:00:01:02 10.0.2.2 2
table_add ecmp_nhop set_nhop 1 =>
00:00:00:00:01:03 10.0.3.3 3
```

CONGA

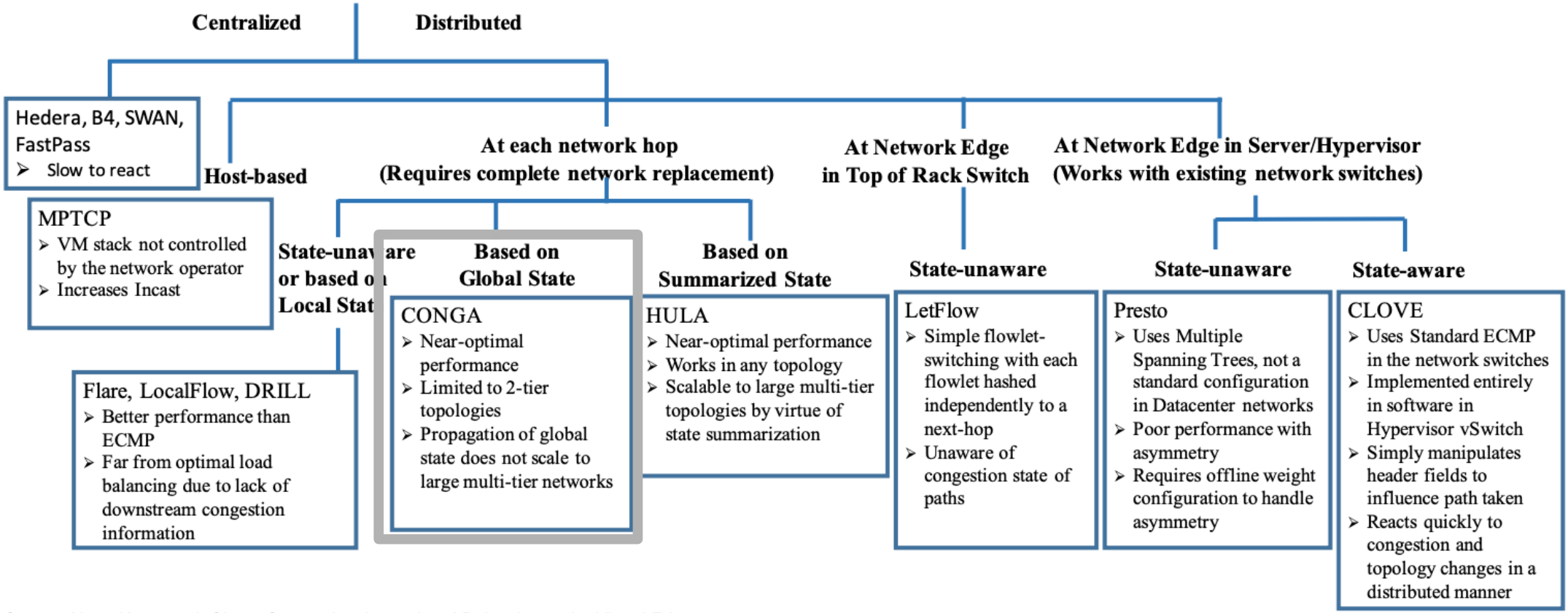
CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

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

Cisco Systems

Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

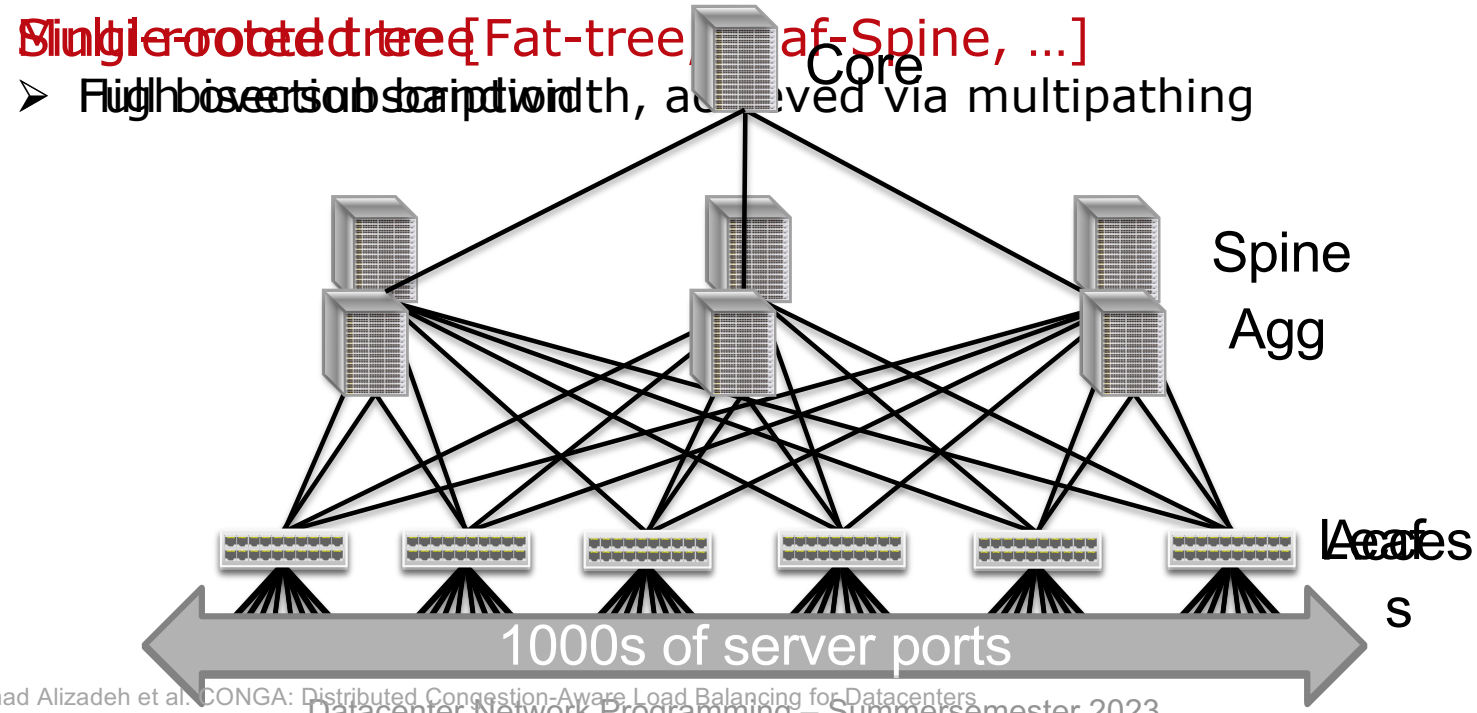
CONGA - Background



Source: Naga Katta et al. Clove: Congestion-Aware Load Balancing at the Virtual Edge

Motivation

- **DC networks need large bisection bandwidth for distributed apps (big data, HPC, web services, etc)**



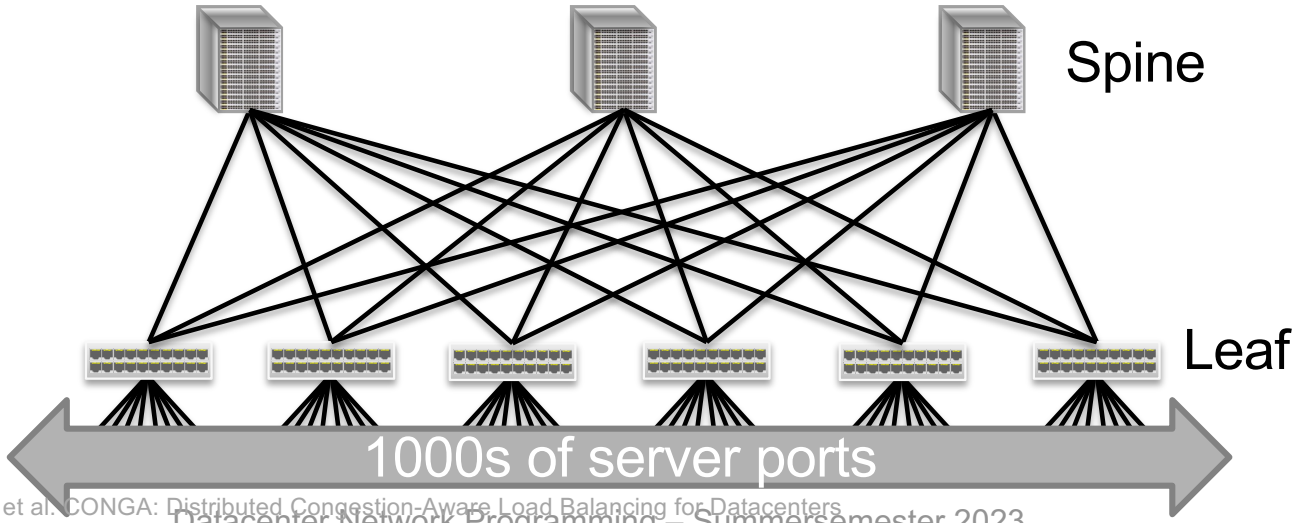
Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Motivation

- **DC networks need large bisection bandwidth for distributed apps (big data, HPC, web services, etc)**

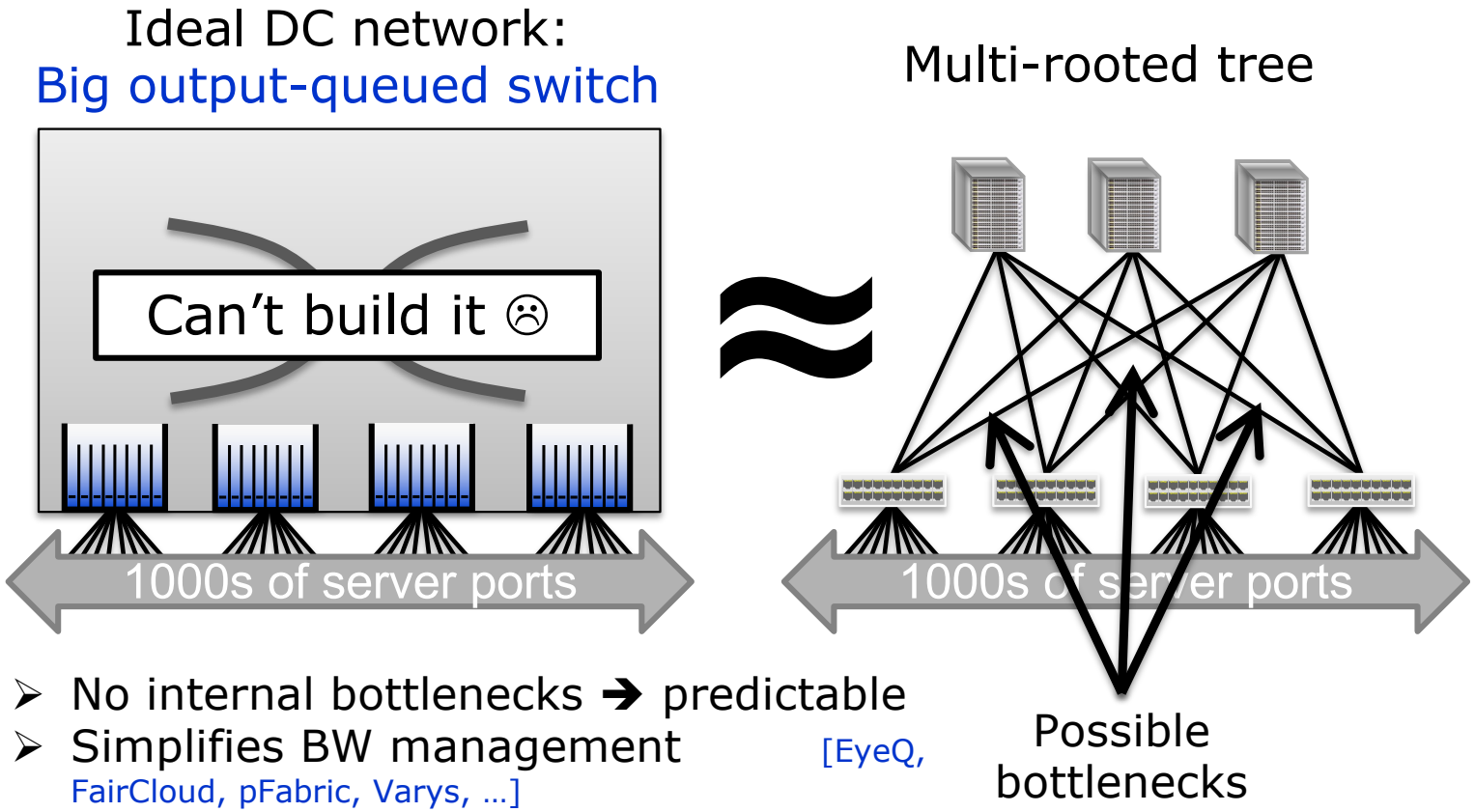
Multi-rooted tree [Fat-tree, Leaf-Spine, …]

➢ Full bisection bandwidth, achieved via multipathing



Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Multi-rooted != Ideal DC Network



- No internal bottlenecks ➔ predictable
- Simplifies BW management [EyeQ, FairCloud, pFabric, Varys, …]

Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

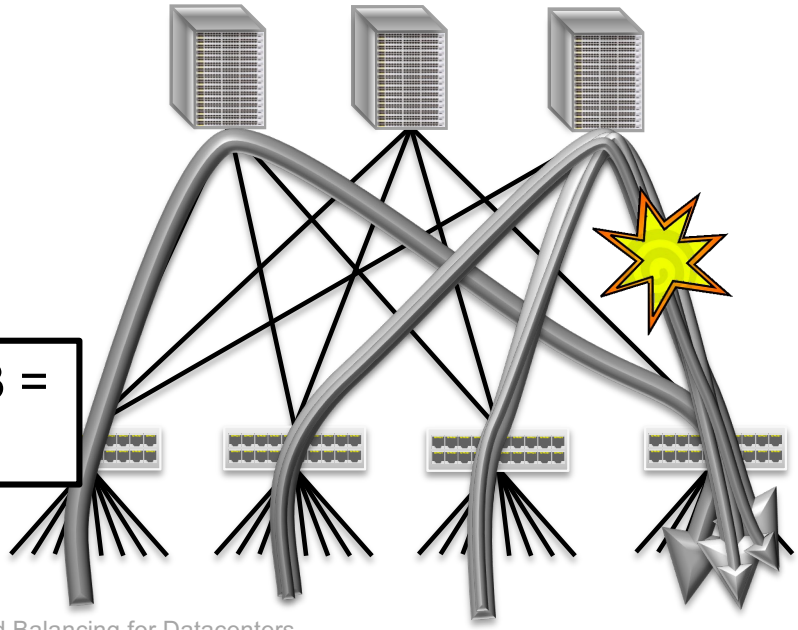
Today: ECMP Load Balancing

- **Pick among equal-cost paths by a hash of 5-tuple**
  - **Approximates Valiant load balancing**
  - **Preserves packet order**

Problems:

- Hash collisions
  (coarse granularity)
- Local & statel
  (v. bad with asym
  due to link failures)

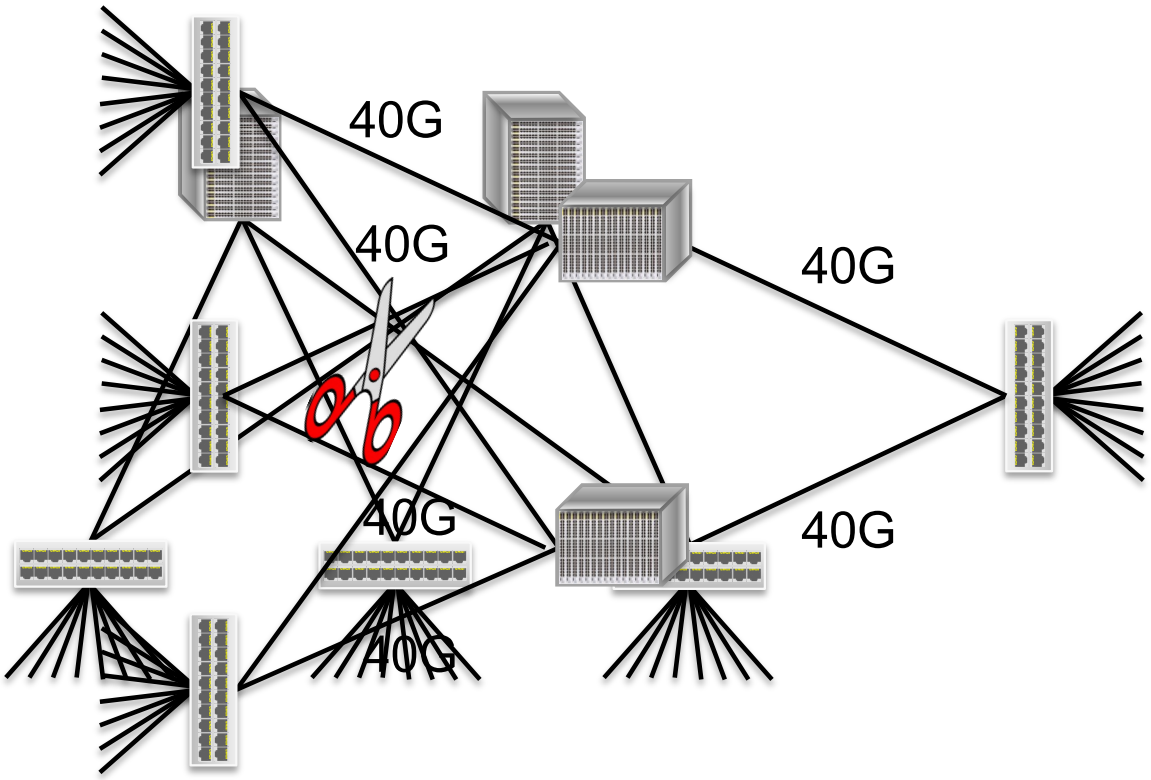
H(f) % 3 = 0



Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Dealing with Asymmetry

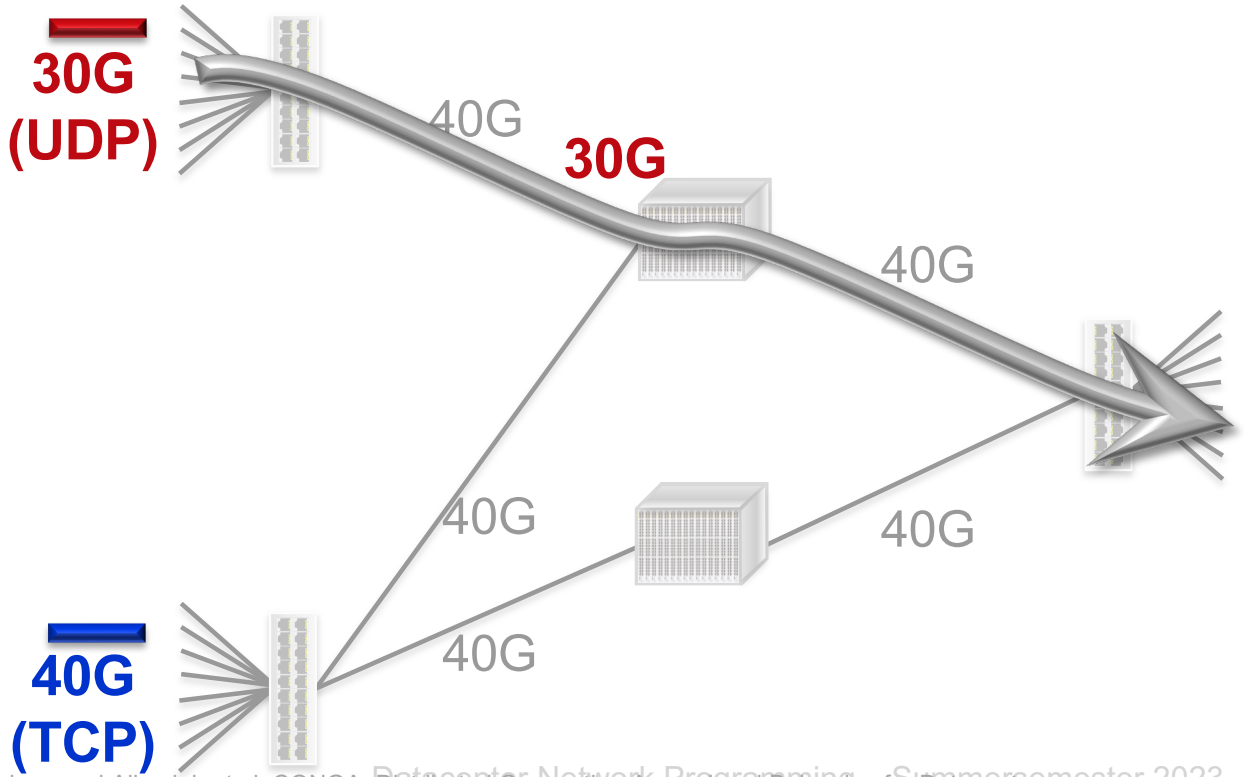
- **Handling asymmetry needs non-local knowledge**



Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Dealing with Asymmetry

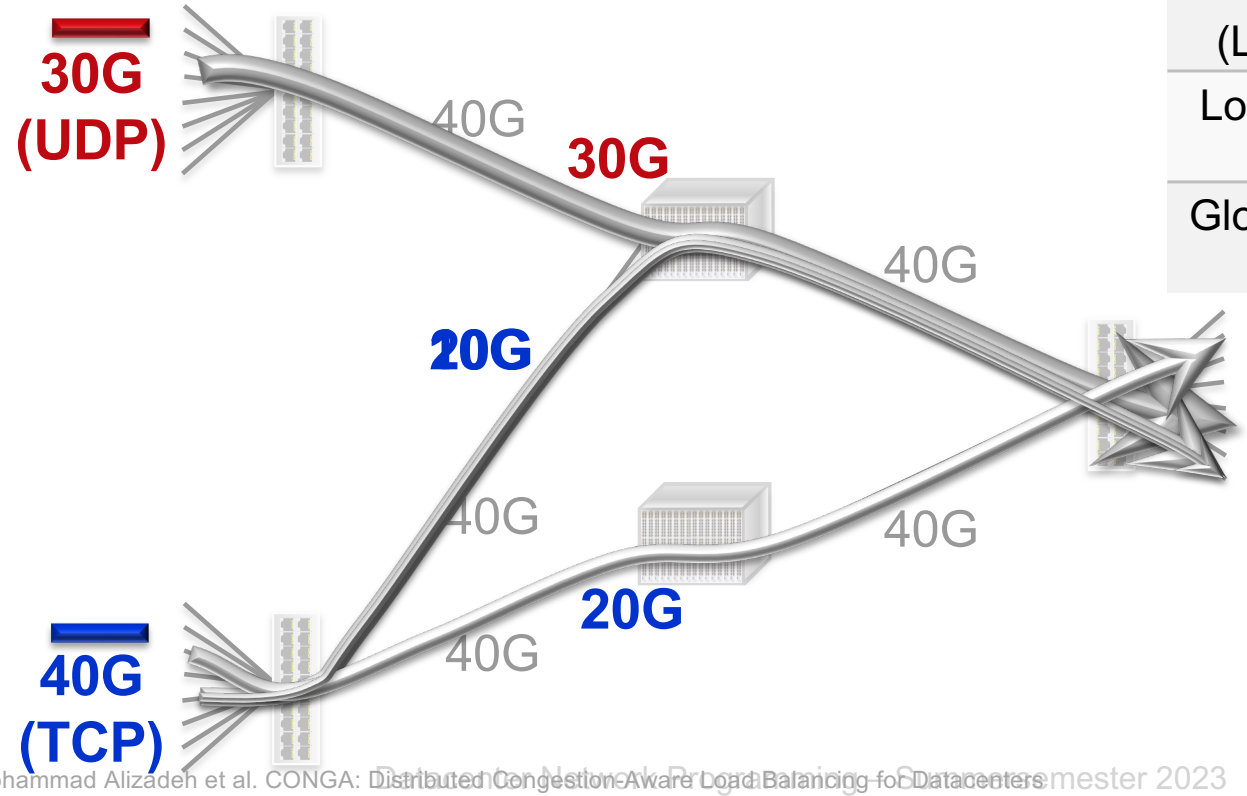
- **Handling asymmetry needs non-local knowledge**



Scheme	Thrput
ECMP (Local Stateless)	
Local Cong-Aware	
Global Cong-Aware	

Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

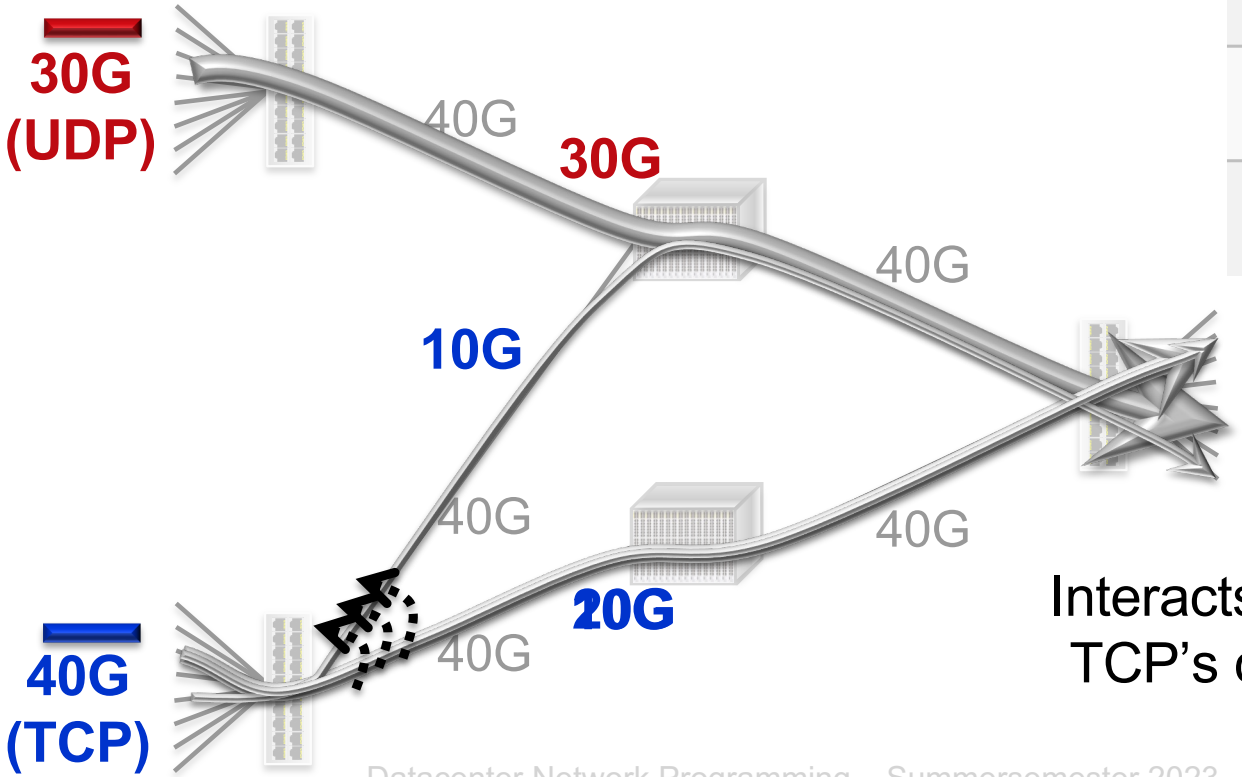
Dealing with Asymmetry: ECMP



Scheme	Thrput
ECMP (Local Stateless)	**60G**
Local Cong-Aware	
Global Cong-Aware	

Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Dealing with Asymmetry: Local Congestion-Aware



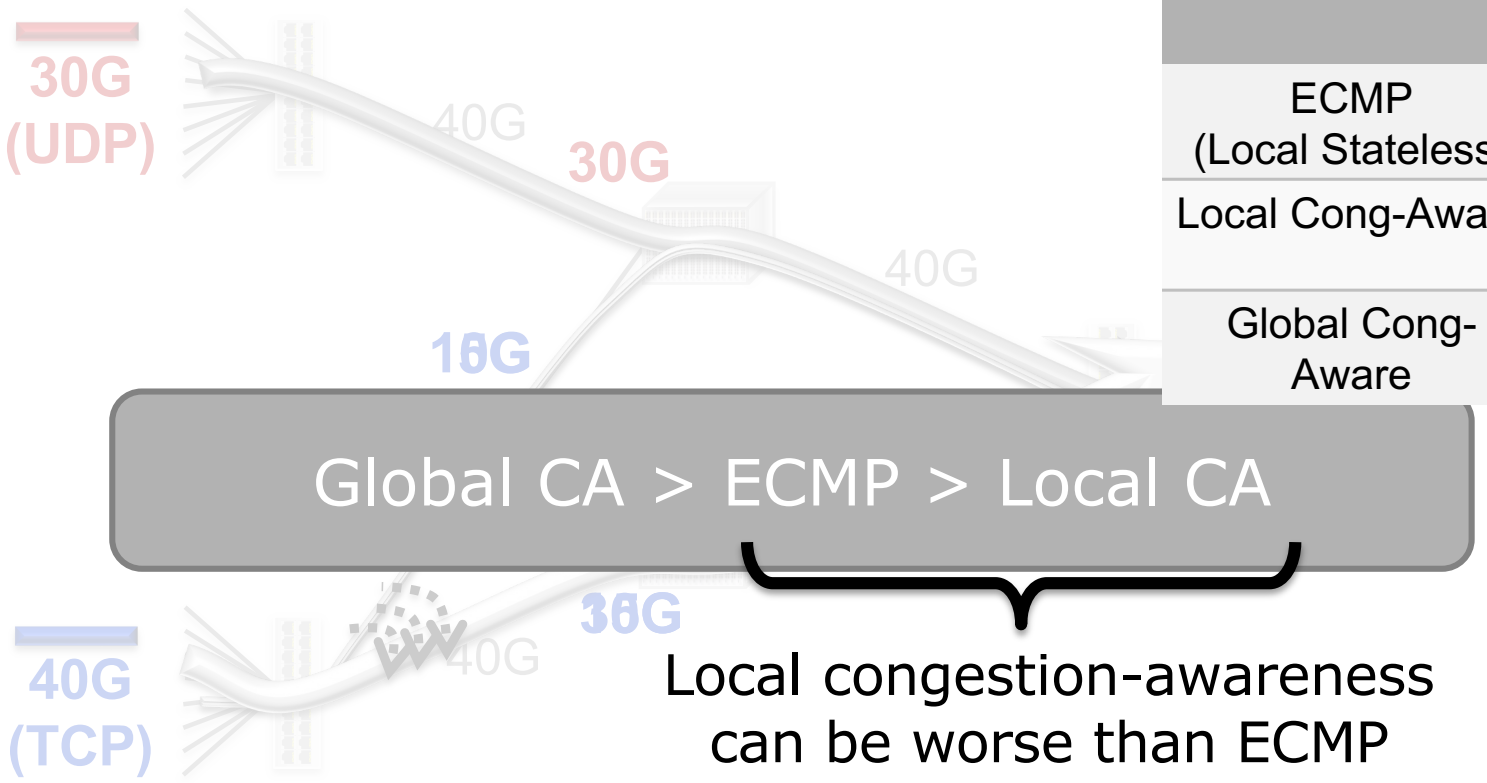
Scheme	Thrput
ECMP (Local Stateless)	**60G**
Local Cong-Aware	**50G**
Global Cong-Aware	

Interacts poorly with TCP's control loop

Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Dealing with Asymmetry

- **Handling asymmetry needs non-local knowledge**



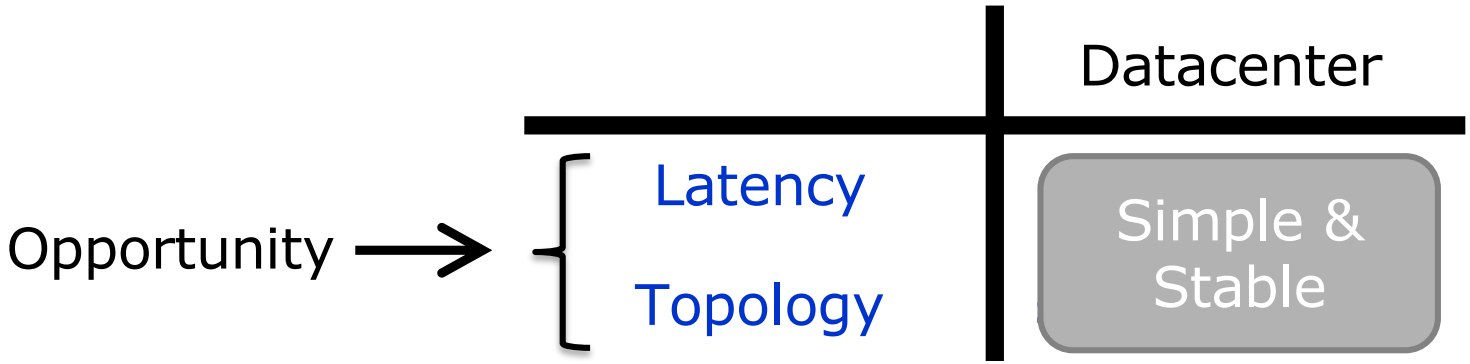
Scheme	Thrput
ECMP (Local Stateless)	**60G**
Local Cong-Aware	**50G**
Global Cong-Aware	**70G**

Global CA > ECMP > Local CA

Local congestion-awareness can be worse than ECMP

Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Global Congestion-Awareness (in Datacenters)

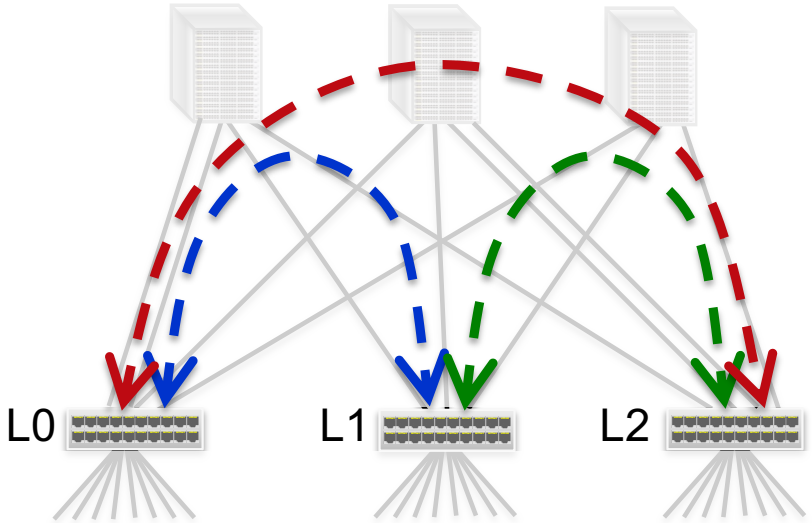


Key Insight:
Use *extremely* fast, low latency distributed control

Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

CONGA

1. Leaf switches (top-of-rack) track congestion to other leaves on different paths in near real-time
2. Use greedy decisions to minimize bottleneck util

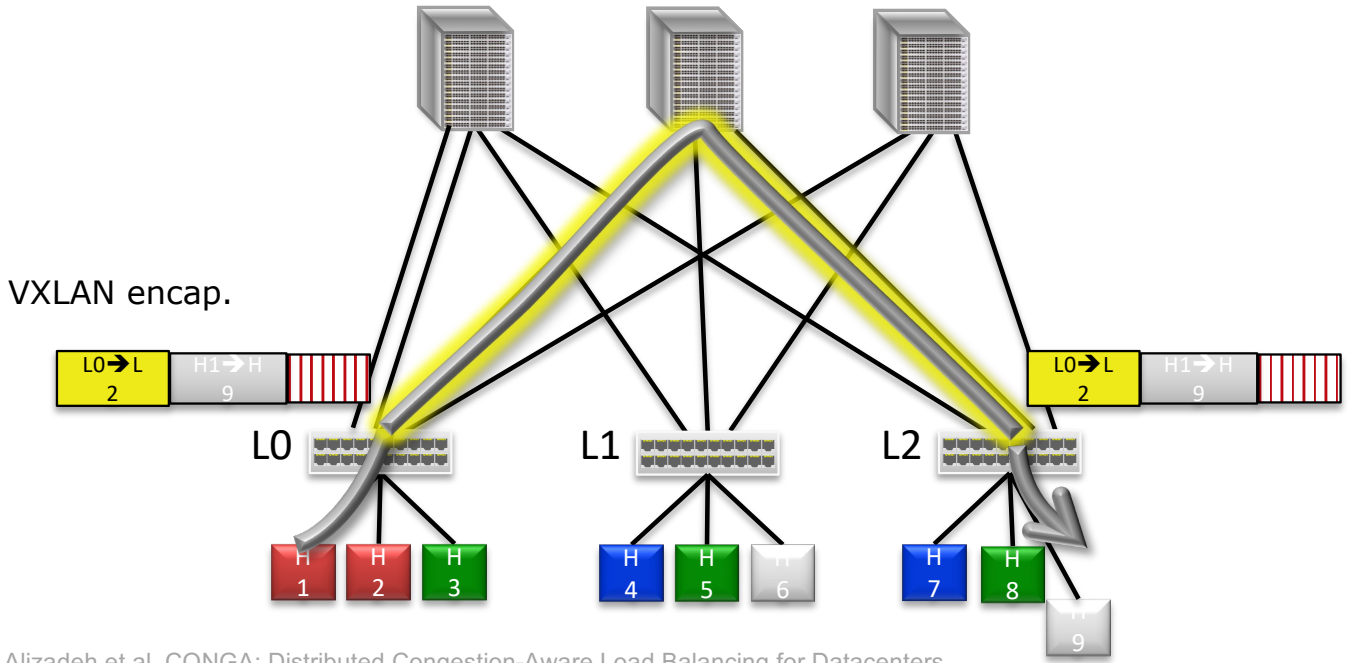


- **Fast feedback loops between leaf switches, directly in dataplane**

Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

DESIGN

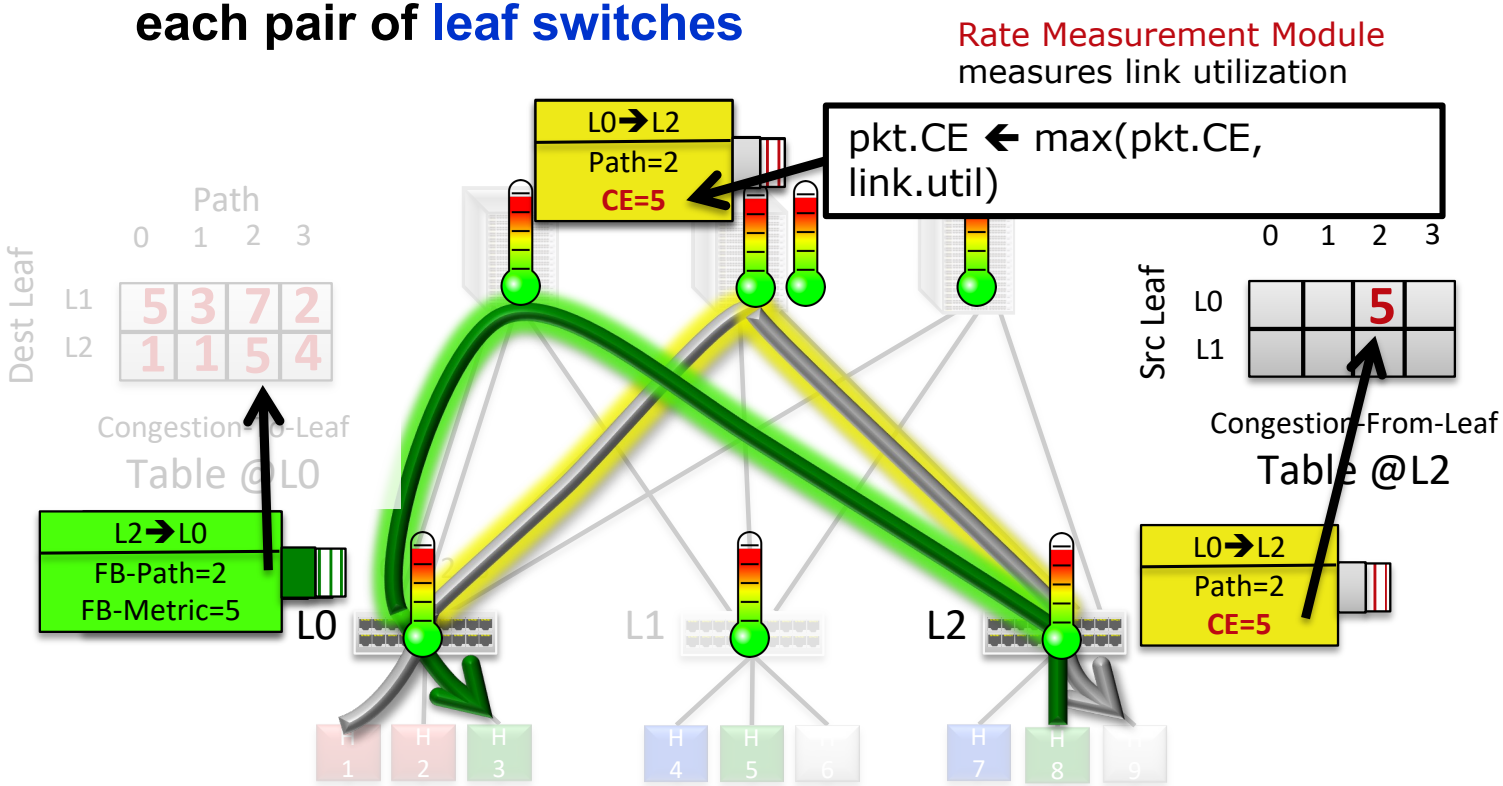
- **CONGA operates over a standard DC overlay (VXLAN)**
- **Already deployed to virtualize the physical network**



Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Design: Leaf-to-Leaf Feedback

- **Track path-wise congestion metrics (3 bits) between each pair of leaf switches**

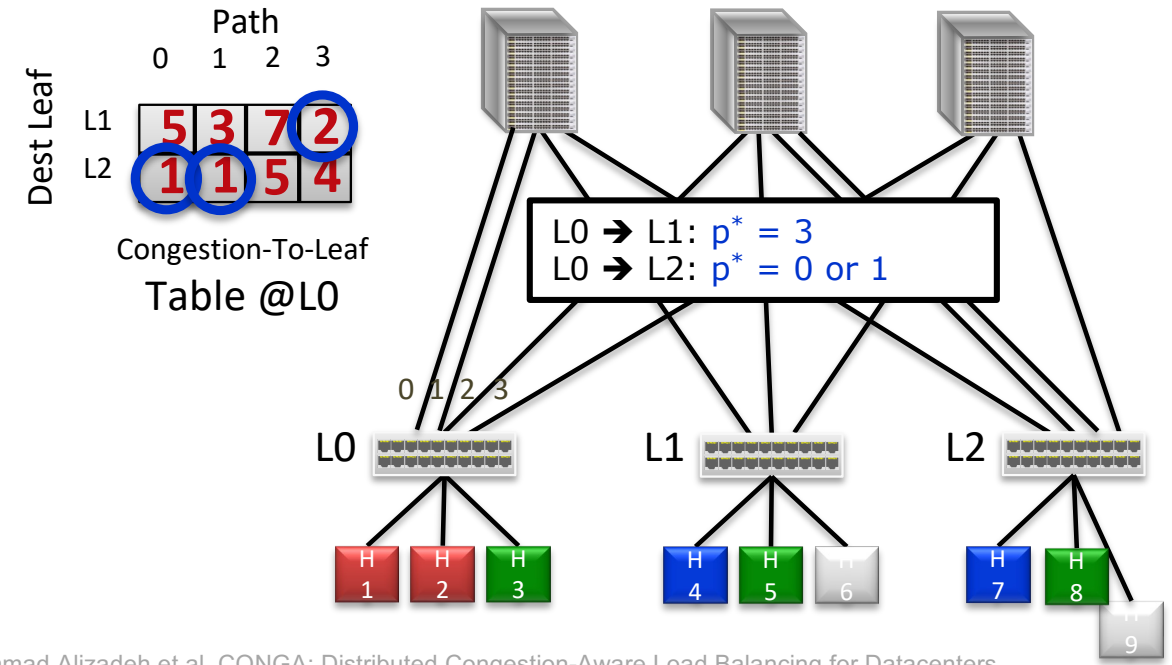


Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Design: LB Decisions

- **Send each packet on least congested path**

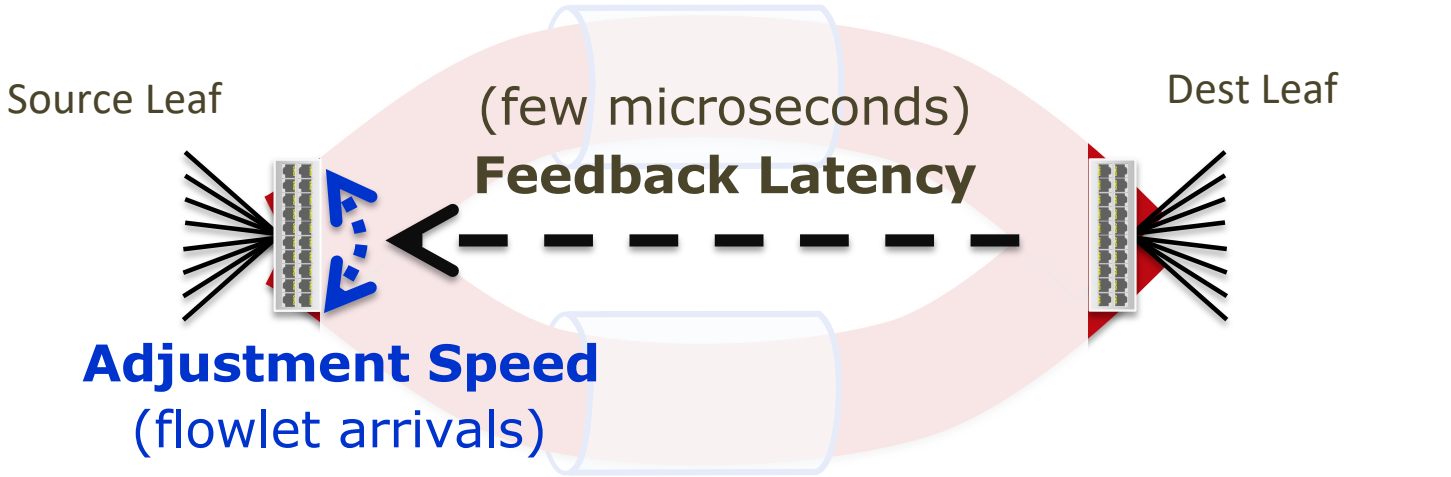
flowlet [Kandula et al 2007]



Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Why is this Stable?

Stability usually requires a sophisticated control law (e.g., TeXCP, MPTCP, etc)



Near-zero latency + flowlets ➔ stable

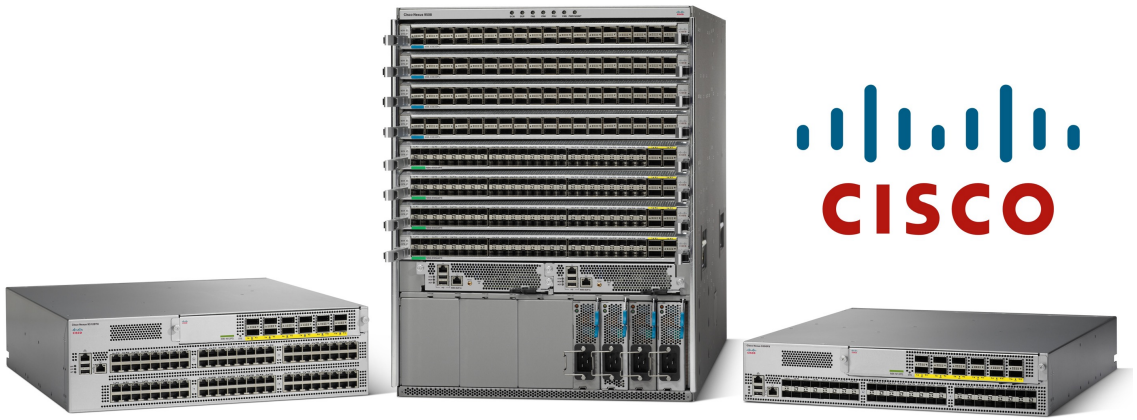
Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters

Implementation

- **Implemented in silicon for Cisco's flagship ACI datacenter fabric**

➢ Scales to over 25,000 non-blocking 10G ports (2-tier Leaf-Spine)

➢ Die area: <2% of chip



Source: Mohammad Alizadeh et al. CONGA: Distributed Congestion-Aware Load Balancing for Datacenters