Advanced Computer Networks

Flow Scheduling in Data Center Networks (II)

CS740

https://pages.cs.wisc.edu/~mgliu/CS740/F25/index.html

Ming Liu
mgliu@cs.wisc.edu

Outline

- Last lecture
  - Flow scheduling in data center networks (I)
- Today
  - Flow scheduling in data center networks (II)
- Announcements
  - Project proposal due 10/02/2025 11:59 PM
  - Lab1 due 10/08/2025 11:59 PM

Unsolved Issued in Hedra

- #1: Networking bandwidth capacity (supply) is ephemeral!
- #2: Flow demand and flow # are unpredictable!
- #3: Real-time global view is hard to maintain!

How does CONGA help?

CONGA Technique #1

- #1: Networking bandwidth capacity (supply) is ephemeral!
- #2: Flow demand and flow # are unpredictable!
- #3: Real-time global view is hard to maintain!

Discounting Rate Estimator (DRE)

Discounting Rate Estimator (DRE)

- Per-egress port register (R)
  - Additive increase: R = R + packet_size, triggered every packet
  - Multiplicate decrease: R = R x (1 - alpha), triggered periodically (T_dre)

Discounting Rate Estimator (DRE)

- Per-egress port register (R)
  - Additive increase: R = R + packet_size, triggered every packet
  - Multiplicate decrease: R = R x (1 - alpha), triggered periodically (T_dre)
- First-order low pass filter applied to packet arrival
- React quickly to traffic bursts
  - Increments take immediately upon packet arrival

CONGA Technique #2

- #1: Networking bandwidth capacity (supply) is ephemeral!
- #2: Flow demand and flow # are unpredictable!
- #3: Real-time global view is hard to maintain!

CONGA Technique #2

- #1: Networking bandwidth capacity (supply) is ephemeral!
- #
- #

But,

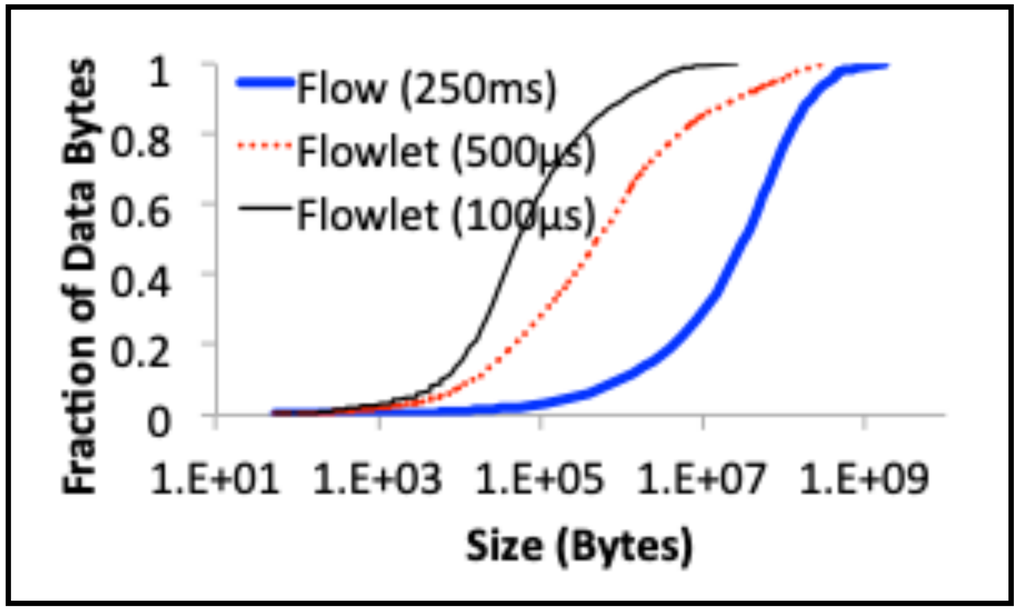
- How large is a flow?
- What is a flow size distribution?
- How many concurrent flows?
- …

Flowlet

- Bursts of packets from a flow separated by large enough gaps
  - “gap” is defined by time

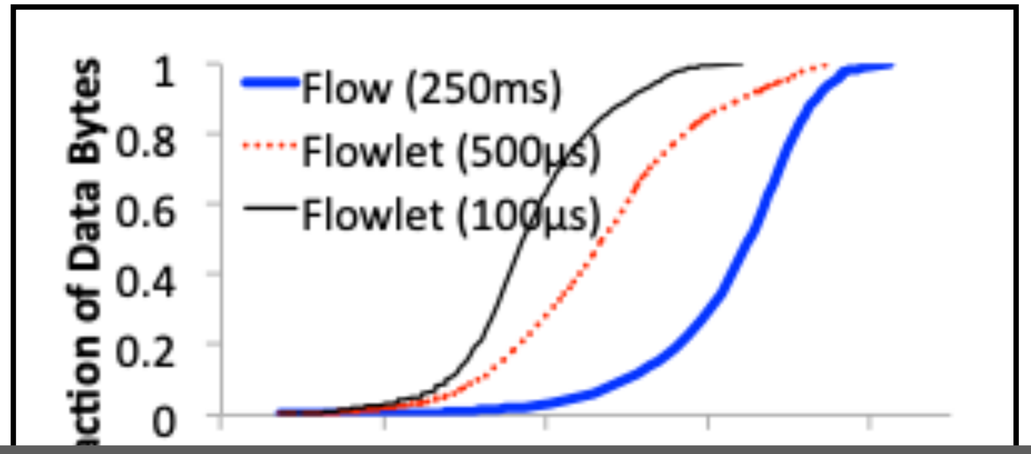
Flowlet

- Bursts of packets from a flow separated by large enough gaps
  - “gap” is defined by time



Flowlet

- Bursts of packets from a flow separated by large enough gaps
  - “gap” is defined by time



- 250ms: 50% of bytes are in a flow (> ~30MB)
- 500us: 50% of bytes are in a flow (> ~500KB)
- Fine-grained low balancing is possible

How do we detect a FlowLet?

Flowlet Detection

- “Smart” edge switch
  - FlowLet ID = Hash (5-tuple)
  - Port Number: forwarding port
  - Valid Bit: 1 if the FlowLet is active
  - Age Bit: 1 if the entry is expired

FlowLet ID	Port Number	Valid Bit	Age Bit
1234	4	1	1
5678	5	0	0

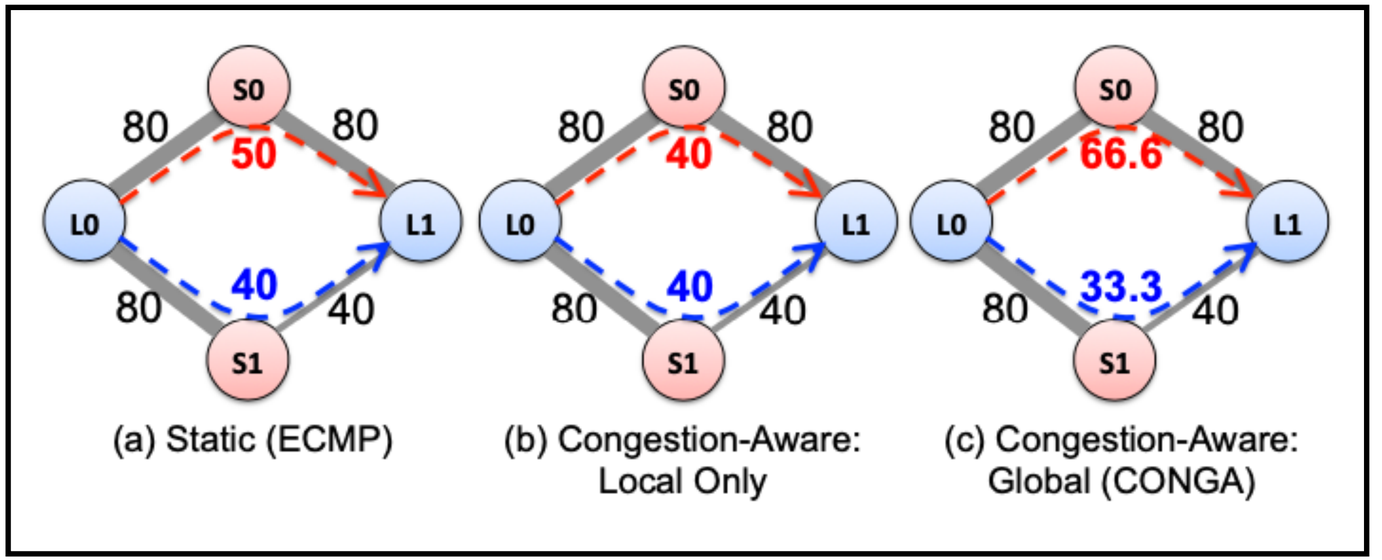
Flowlet Detection

- “Smart” edge switch
  - FlowLet ID – Hash (5-tuple)
  - Age Bit: 1 if the entry is expired

Is this enough?

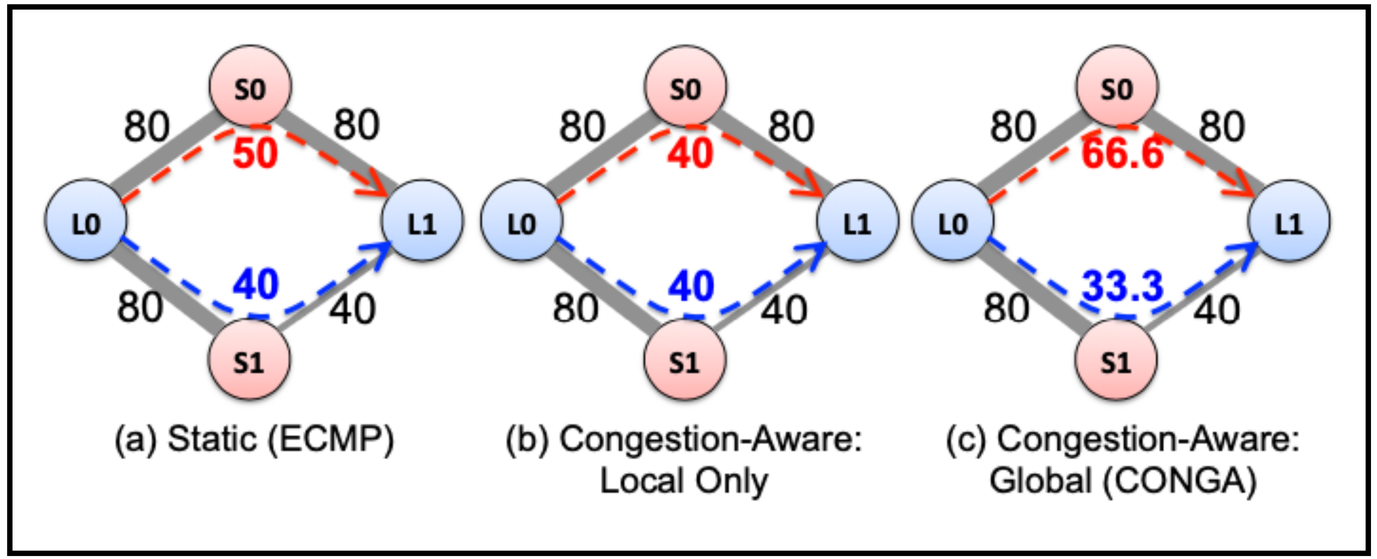
FlowLet ID	Port Number	Valid Bit	Age Bit
1234	4	1	1
5678	5	0	0

Congestion Awareness



Congestion Awareness

- Timely link utilization
  - Based on DRE
  - DRE Register / Link Capacity, quantized into 3-bits
- Congestion should be global



CONGA Technique #3

- #1: Networking bandwidth capacity (supply) is ephemeral!
- #2: Flow demand and flow # are unpredictable!
- #3: Real-time global view is hard to maintain!

Distributed Load Balancing and In-Network

Distributed Load Balancing and In-Network

- Why distributed, not central?
  - Data center traffic is very bursty and unpredictable
  - Data center topology is regular

Distributed Load Balancing and In-Network

- Why distributed, not central?
  - Data center traffic is very bursty and unpredictable
  - Data center topology is regular

- Why in-network, not host?
  - The endhost cannot capture Burstiness
  - The transport stack is already fat with many functionalities
  - Kernel-bypass I/O systems emerge

Distributed Load Balancing and In-Network

- Why distributed, not central?
  - Data center traffic is very bursty and unpredictable
  - Data center topology is regular

- Why in-network, not host?
  - The endhost cannot capture Burstiness
  - The transport stack is already fat with many functionalities
  - Kernel-bypass I/O systems emerge

Make the load-balancing decision for the first packet of a FlowLet at the edge switch

Combine Everything Together

- #1: Networking bandwidth capacity (supply) is ephemeral! ✓
- #2: Flow demand and flow # are unpredictable! ✓
- #3: Real-time global view is hard to maintain! ✓

How does Conga work?

CONGA — Host Server

CONGA — Host Server

- Hosts send TCP/IP traffic
  - No load balancing decision is made

CONGA — Source Leaf Switch

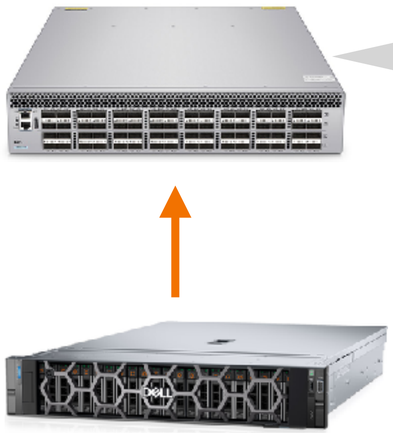
CONGA — Source Leaf Switch

1. FlowLet detection

FlowLet Table

FlowLet	Port	Valid	Age
1234	4	1	1
5678	5	0	0

CONGA — Source Leaf Switch



1. FlowLet detection
2. Choose the path (load balancing)

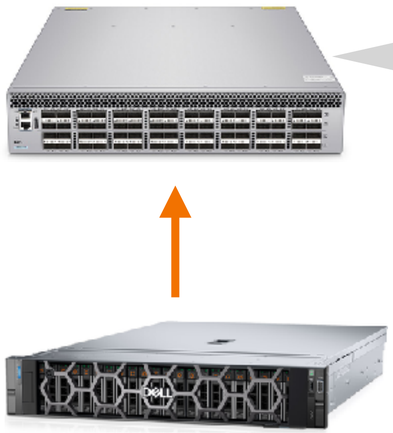
FlowLet Table

FlowLet	Port	Valid	Age
1234	4	1	1
5678	5	0	0

Congestion-To-Leaf Table

Dst Leaf	Path 1	Path 2	Path k
1	0b000	0b000	0b111
2	0b111	0b110	0b101

CONGA — Source Leaf Switch



1. FlowLet detection
2. Choose the path (load balancing)

FlowLet Table

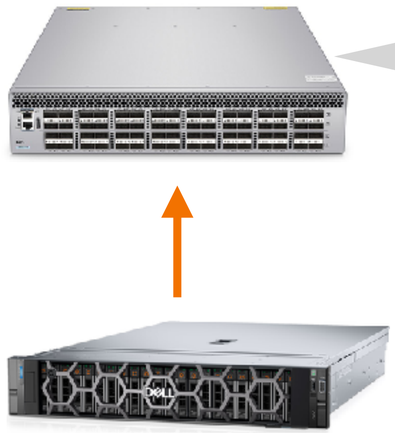
FlowLet	Port	Valid	Age
1234	4	1	1
5678	5	0	0

Congestion-To-Leaf Table

Dst Leaf	Path 1	Path 2	Path k
1	0b000	0b000	0b111
2	0b111	0b110	0b101

CONGA — Source Leaf Switch

- Source Leaf switches forward traffic
  - Setup the FlowLet table
  - Perform load balancing, i.e., minimal load
  - Update the link load



1. FlowLet detection
2. Choose the path (load balancing)
3. Update the link load

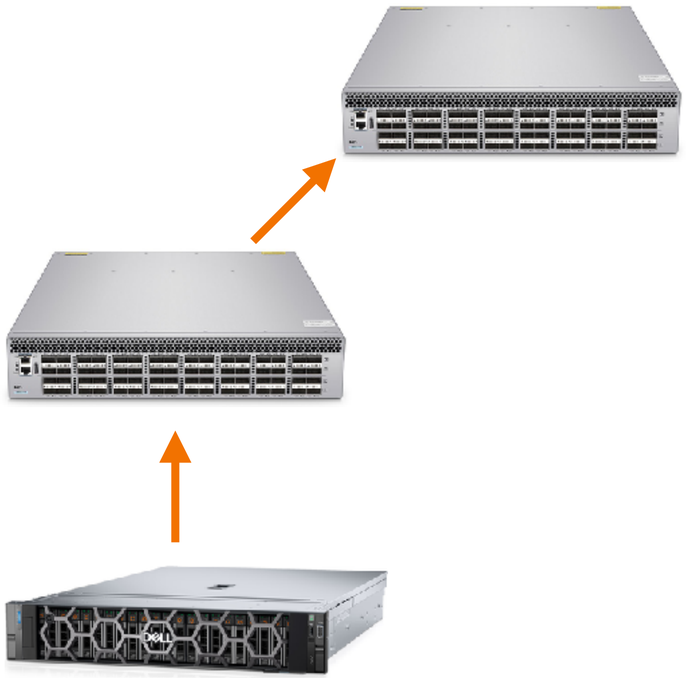
FlowLet Table

FlowLet	Port	Valid	Age
1234	4	1	1
5678	5	0	0

Congestion-To-Leaf Table

Dst Leaf	Path 1	Path 2	Path k
1	0b000	0b000	0b111
2	0b111	0b110	0b101

CONGA — Spine Switch

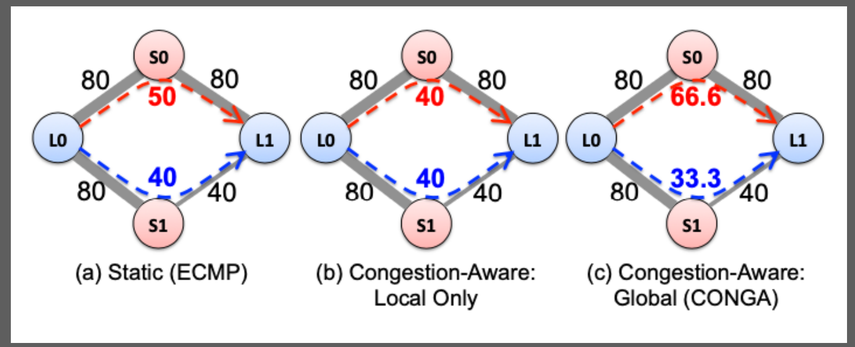


CONGA — Spine Switch

Do we perform load balancing again?

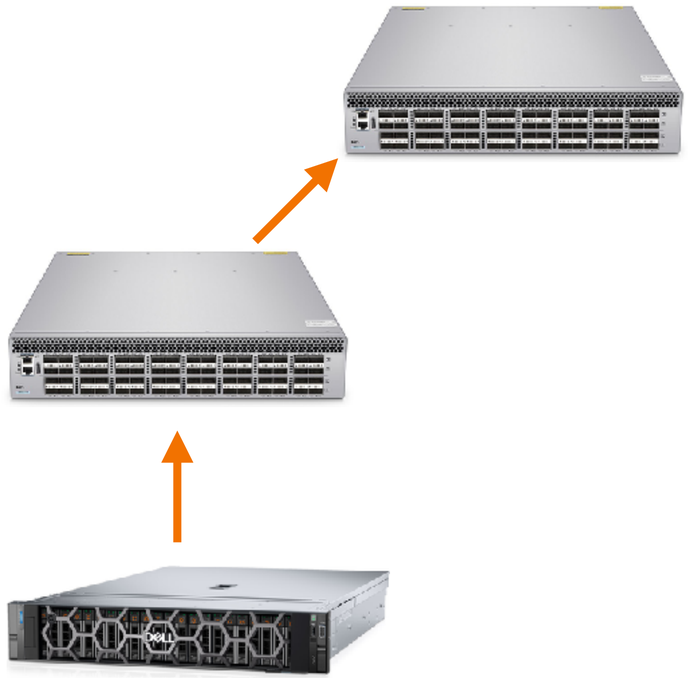
CONGA — Spine Switch

Do we perform load balancing again?



CONGA — Spine Switch

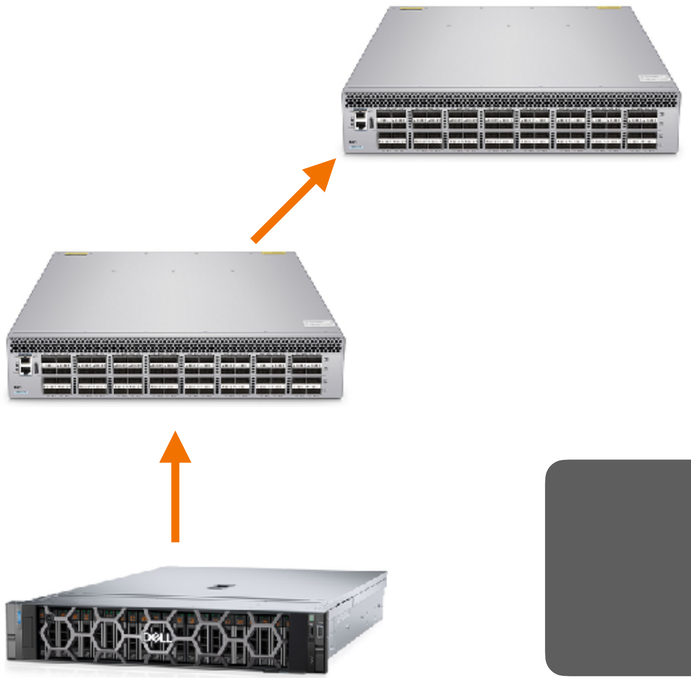
- Spine switches forward traffic
  - Directly choose the next hop based on the existing LB choice
  - Bookkeep the link load



1. Reuse the load-balancing decision
2. Encode the link load

CONGA — Spine Switch

- Spine switches forward traffic
  - Directly choose the next hop based on the existing LB choice
  - Bookkeep the link load



1. Reuse the load-balancing decision
2. Encode the link load

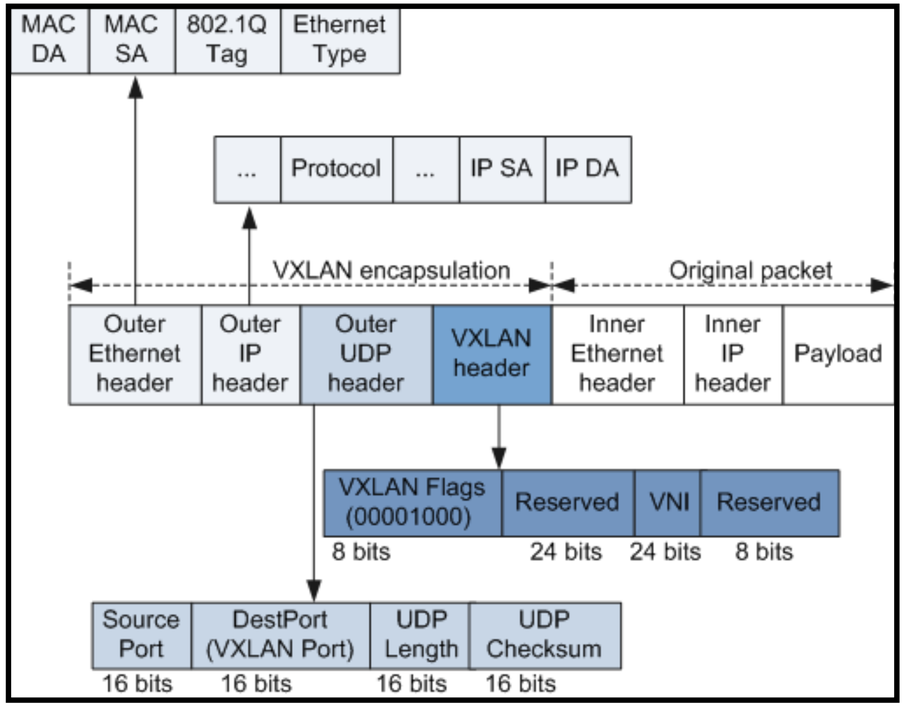
But how?

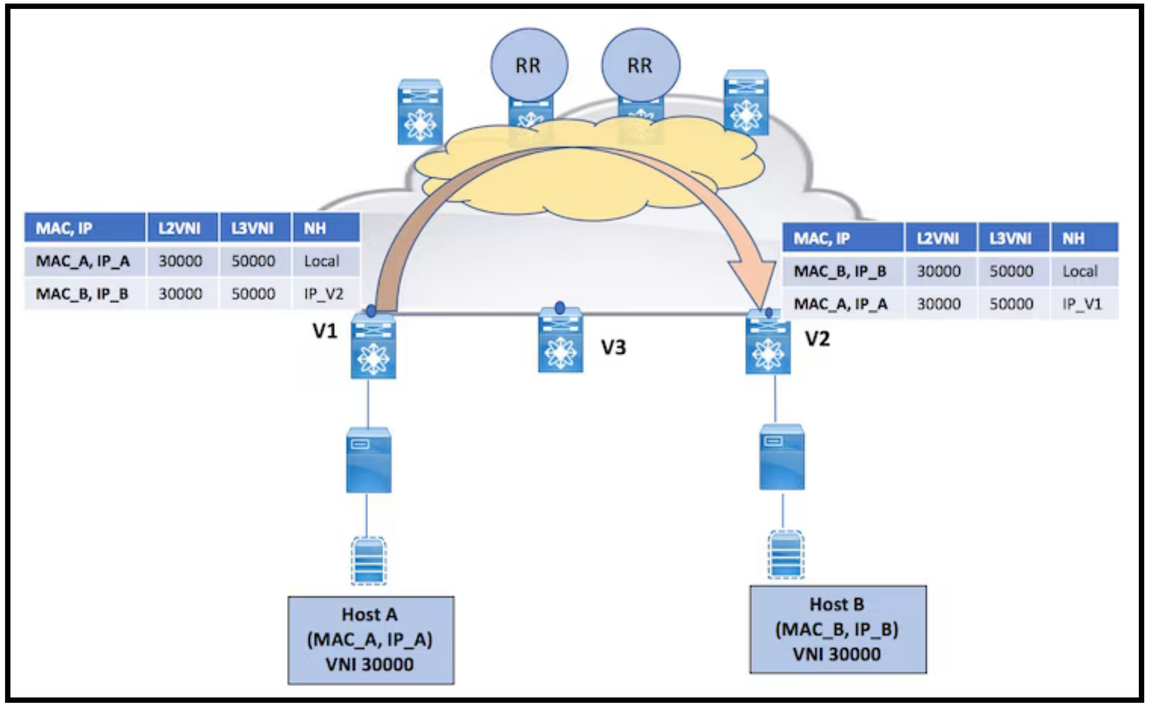
Path Revisit: Server —> Leaf —> Spine

How does CONGA perform addressing and routing?

Path Revisit: Server —> Leaf —> Spine

- VXLAN tunneling between source and destination leaf





Special Packet Header

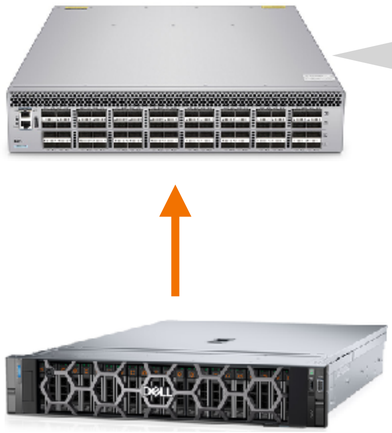
- LBTag (4 bits)
  - Set by the source leaf when making the LB decision
- CE (3 bits)
  - Indicate the path congestion extent
  - Set by every traversed switch along the path
- FB_LBTag (4 bits)
  - Used by the destination leaf switch to piggyback which LBTag is taken
- FB_Metric (3 bits)
  - Used by the destination leaf switch to piggyback the path CE value

Revist CONGA — Host Server

- Hosts send TCP/IP traffic
  - No load balancing decision is made
  - Build the VXLAN packet header

Revisit CONGA — Source Leaf Switch

- Source Leaf switches forward traffic
  - Setup the FlowLet table
  - Perform load balancing, i.e., minimal load
  - Update the link load
  - Modify the conga packet header



1. FlowLet detection
2. Choose the path (load balancing)
3. Update the link load
4. Encode the LB decision

FlowLet Table

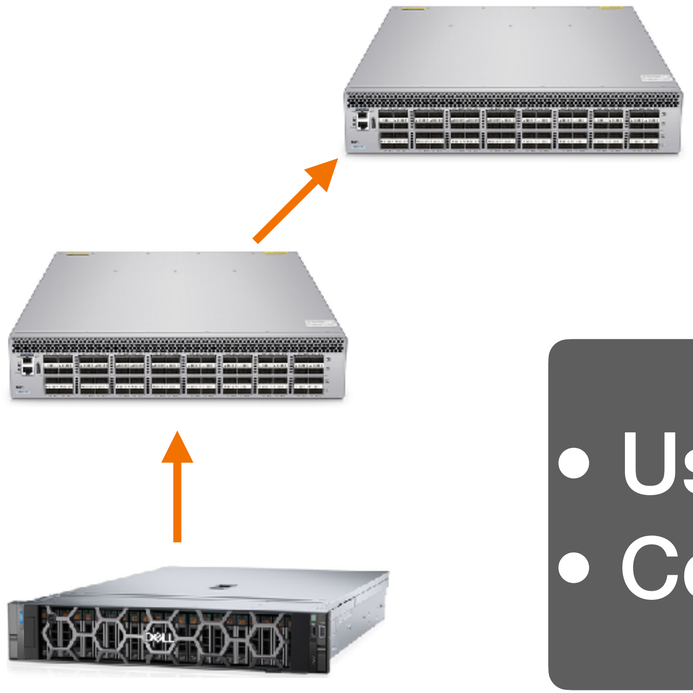
FlowLet	Port	Valid	Age
1234	4	1	1
5678	5	0	0

Congestion-To-Leaf Table

Dst Leaf	Path 1	Path 2	Path k
1	0b000	0b000	0b111
2	0b111	0b110	0b101

Revisit CONGA — Spine Switch

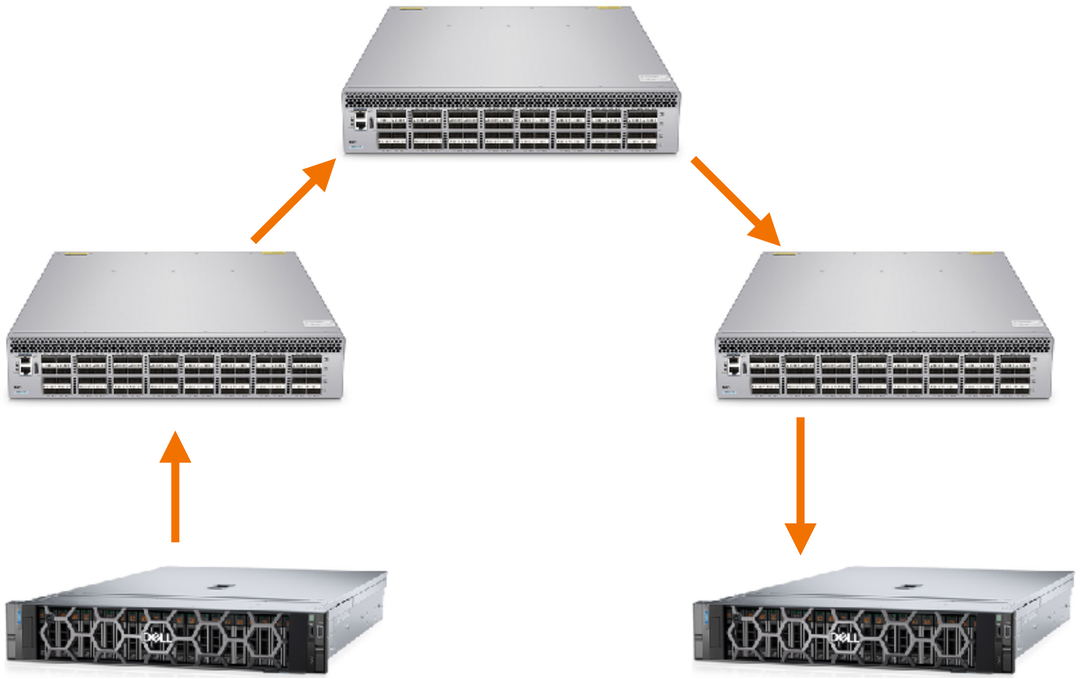
- Spine switches forward traffic
  - Directly choose the next hop based on the existing LB choice
  - Bookkeep the link load



1. Reuse the load-balancing decision
2. Encode the link load

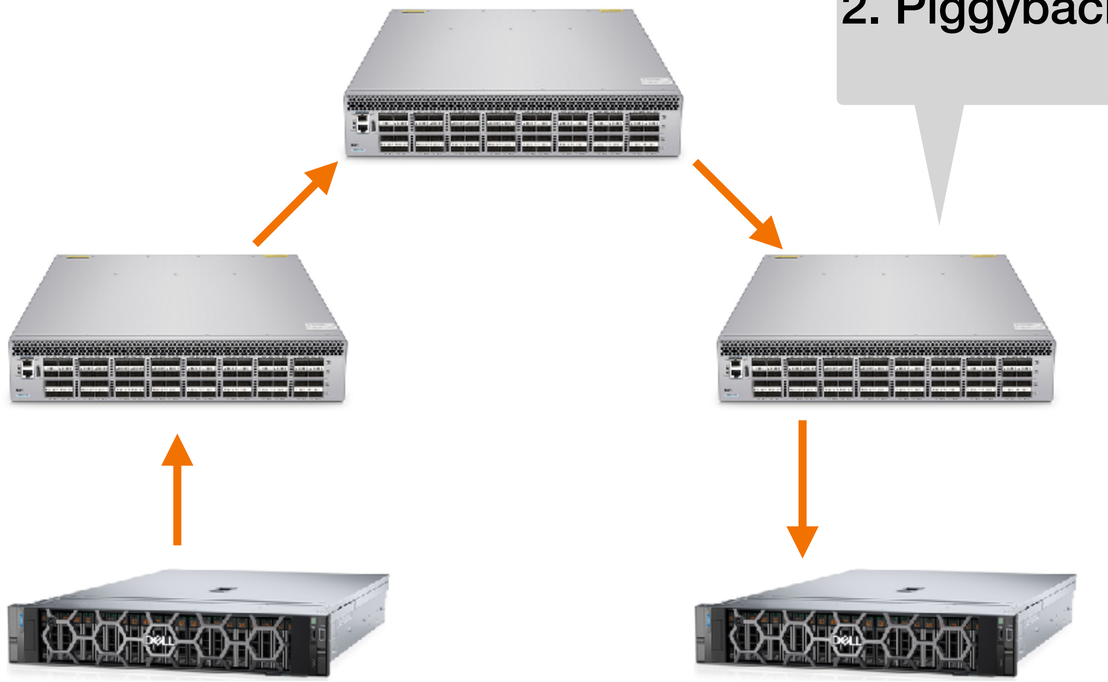
- Use the LBTag to forward
- Conditional update the CE value

CONGA — Destination Leaf Switch



CONGA — Destination Leaf Switch

- Destination leaf switches return the load status opportunistically
  - Reuse the ACK packets
  - Not Timely



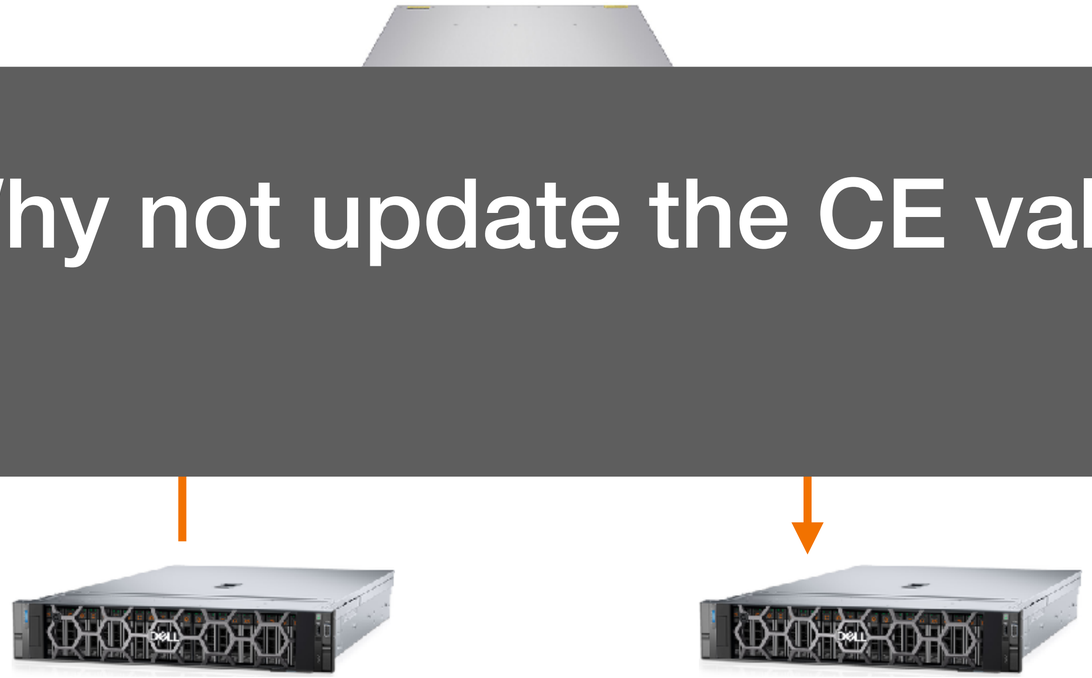
1. Piggyback the load balancing decision
2. Piggyback the CE value

Congestion-From-Leaf Table

Src Leaf	Path 1	Path 2	Path k
1	0b000	0b000	0b111
2	0b111	0b110	0b101

CONGA — Destination Leaf Switch

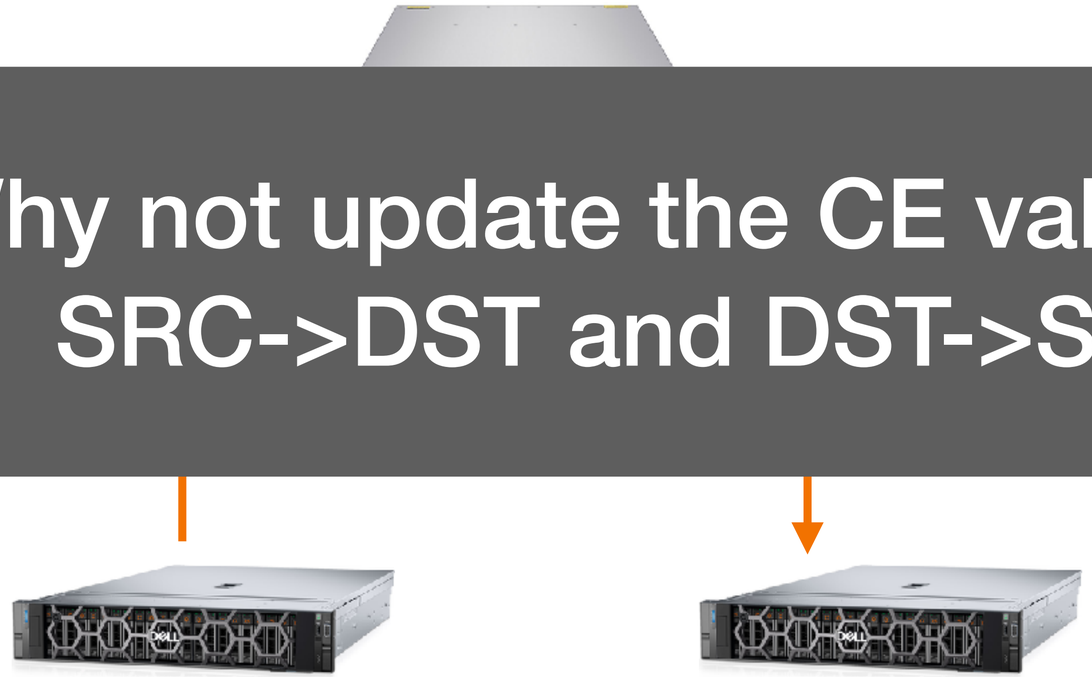
- Destination leaf switches return the load status opportunistically
  - Reuse the ACK packets
  - Not Timely



Why not update the CE value along the reply path?

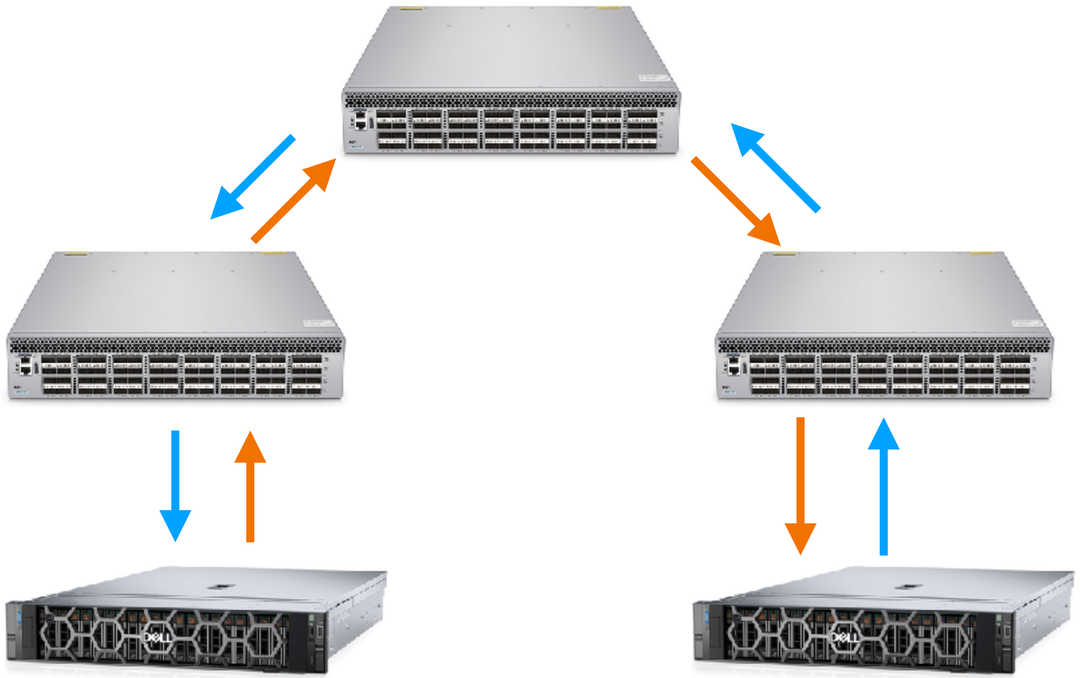
CONGA — Destination Leaf Switch

- Destination leaf switches return the load status opportunistically
  - Reuse the ACK packets
  - Not Timely



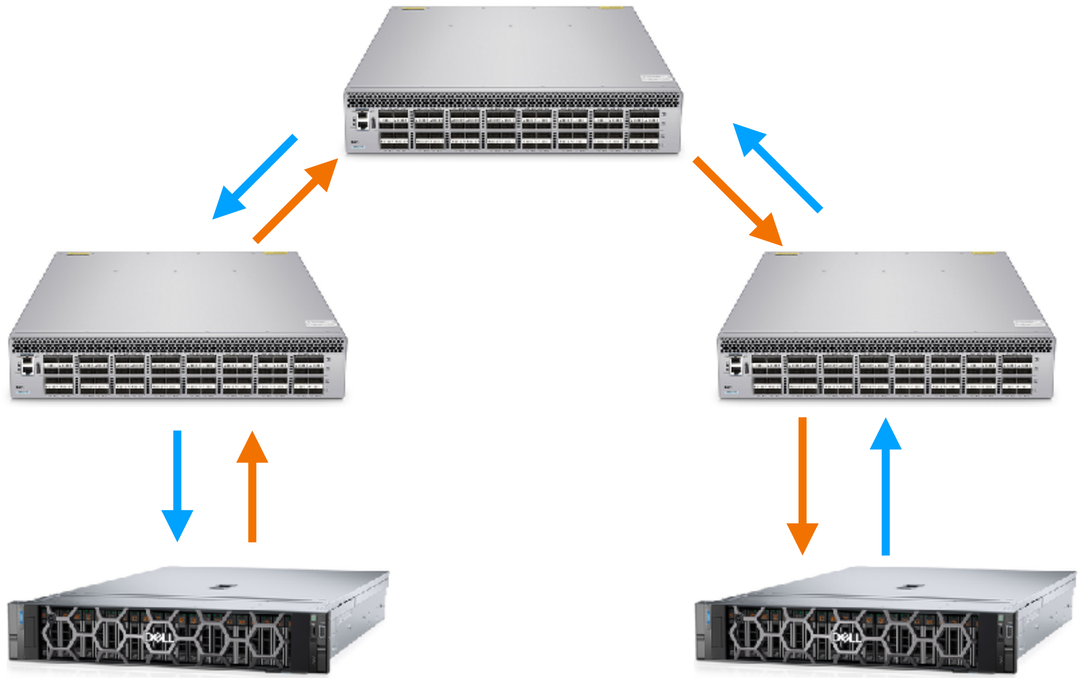
Why not update the CE value along the reply path?
SRC->DST and DST->SRC are not the same!

CONGA — Leaf/Spine Switch



CONGA — Leaf/Spine Switch

- Source leaf switches update the congestion-to-leaf table



Congestion-To-Leaf Table

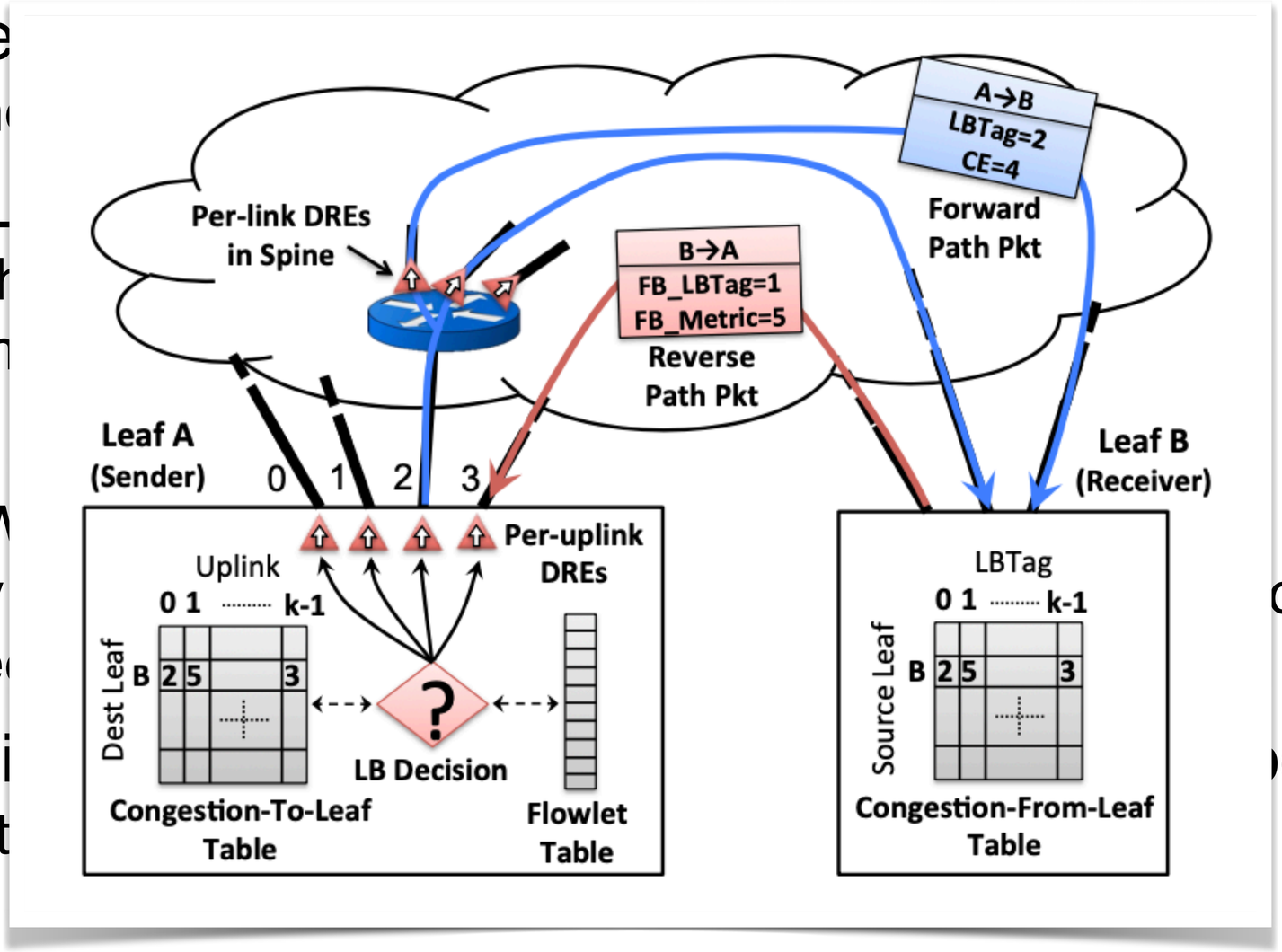
Dst Leaf	Path 1	Path 2	Path k
1	0b000	0b000	0b111
2	0b111	0b110	0b101

Communication Path in CONGA

- Hosts send TCP/IP traffic
  - Build the VXLAN packet header
- Source Leaf switches forward traffic
  - Setup the FlowLet table
  - Perform load balancing, i.e., minimal load
  - Update the link load and congestion-to-leaf table
- Spine switches forward traffic
  - Directly choose the next hop based on the existing LB choice
  - Bookkeep the link load
- Destination leaf switches return the load status opportunistically
  - Reuse the ACK packets

Communication Path in CONGA

- Hosts se
  - Build th
- Source L
  - Setup th
  - Perform
  - Update
- Spine sw
  - Directly
  - Bookke
- Destinati
  - Reuse t



oice

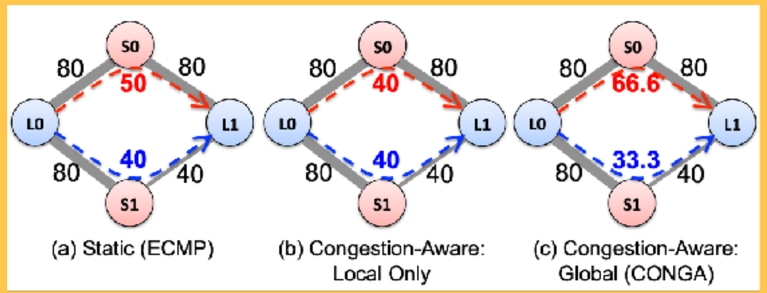
ortunistically

CONGA is a distributed, in-network, global congestion-aware load balancer based on leaf-to-leaf feedback and flowlet switching.

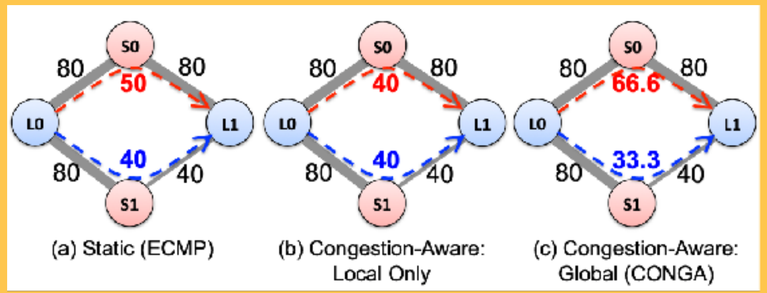
CONGA is a distributed, in-network, global congestion-aware load balancer based on leaf-to-leaf feedback and flowlet switching.

CONGA is a distributed, in-network, global congestion-aware load balancer based on leaf-to-leaf feedback and flowlet switching.

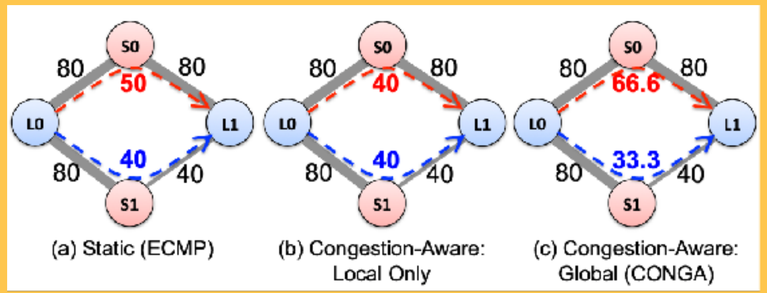
CONGA is a distributed, in-network, global congestion-aware load balancer based on leaf-to-leaf feedback and flowlet switching.

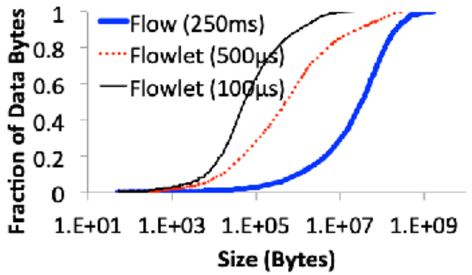


CONGA is a distributed, in-network, global congestion-aware load balancer based on leaf-to-leaf feedback and flowlet switching.



CONGA is a distributed, in-network, global congestion-aware load balancer based on leaf-to-leaf feedback and flowlet switching.





Flow Scheduling Problem Solved?

Flow Scheduling Problem Solved?

- #1: No-trivial hardware support
- #2: Micro-second traffic under 100+Gbps requires timely reaction
- #3: VXLAN has poor flexibility and requires lots of manual efforts

Summary

- Today
  - Flow scheduling in data center networks (II)

- Next two lectures
  - Load balancers in the data centers
  - Maglev (NSDI'16)
  - Duet (SIGCOMM'14)