Let It Flow

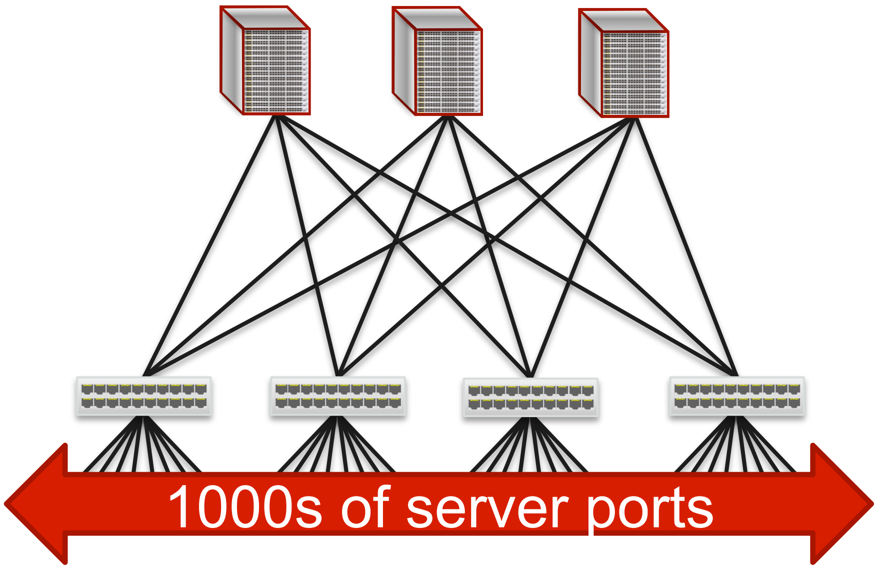
Resilient Asymmetric Load Balancing with Flowlet Switching

Erico Vanini*, Rong Pan*, Mohammad Alizadeh†, Parvin Taheri*, Tom Edsall*

* CISCO † MIT

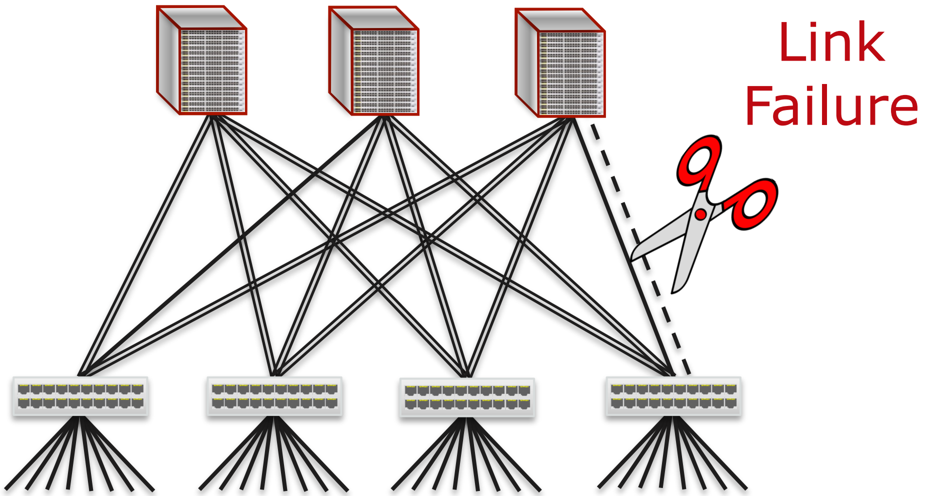
Load Balancing in Data Centers

Multi-rooted tree



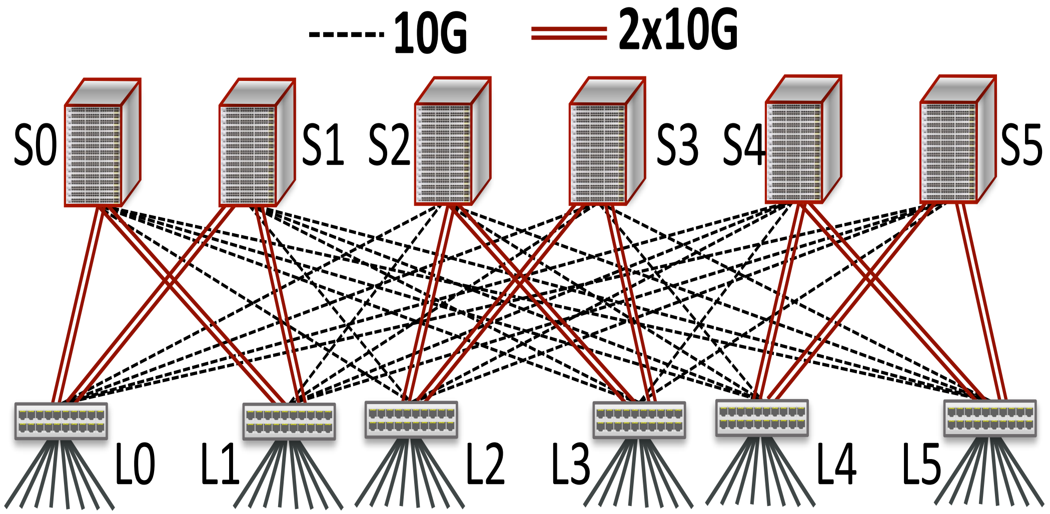
- **Goal:** Avoid congestion hotspots
- Active research area
- Solved for symmetric topologies
- Still open question in asymmetric scenarios

Asymmetry Is Common in Practice



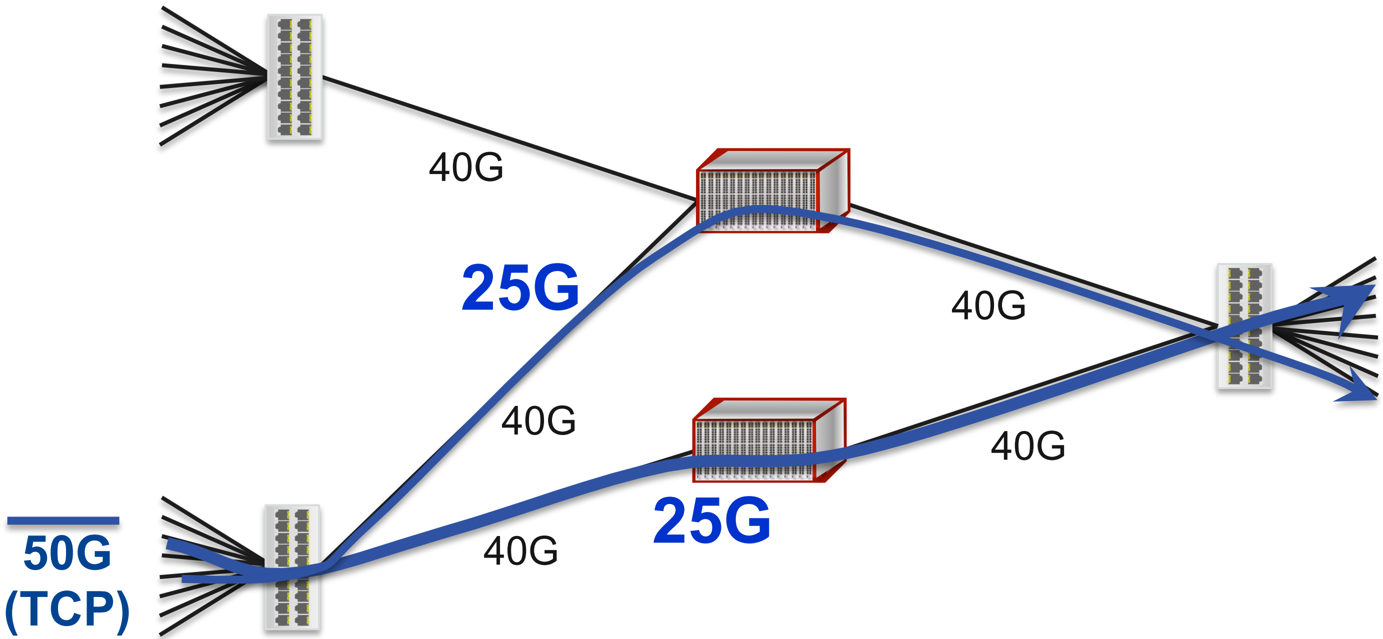
Asymmetry Is Common in Practice

Imbalanced striping: # of ports indivisible by # of switches in other tier

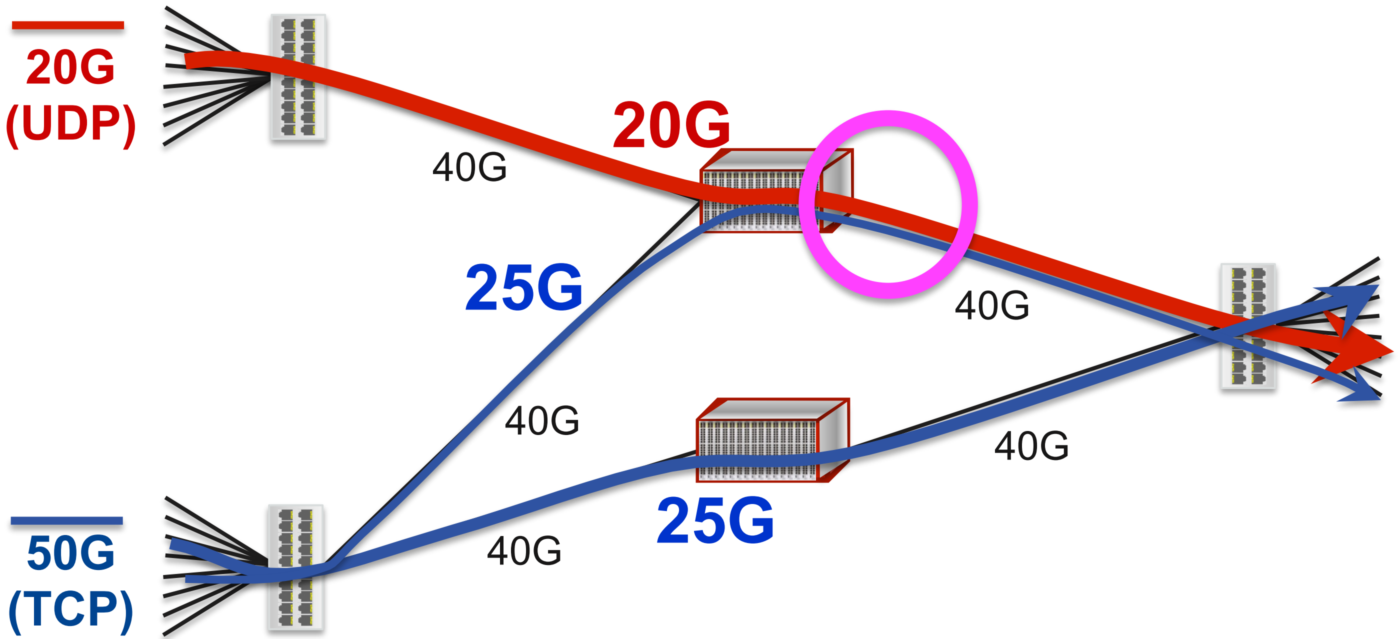


Zhou et al. "WCMP: Weighted cost multipathing for improved fairness in data centers," CoNEXT 2014.

Dealing with Asymmetry

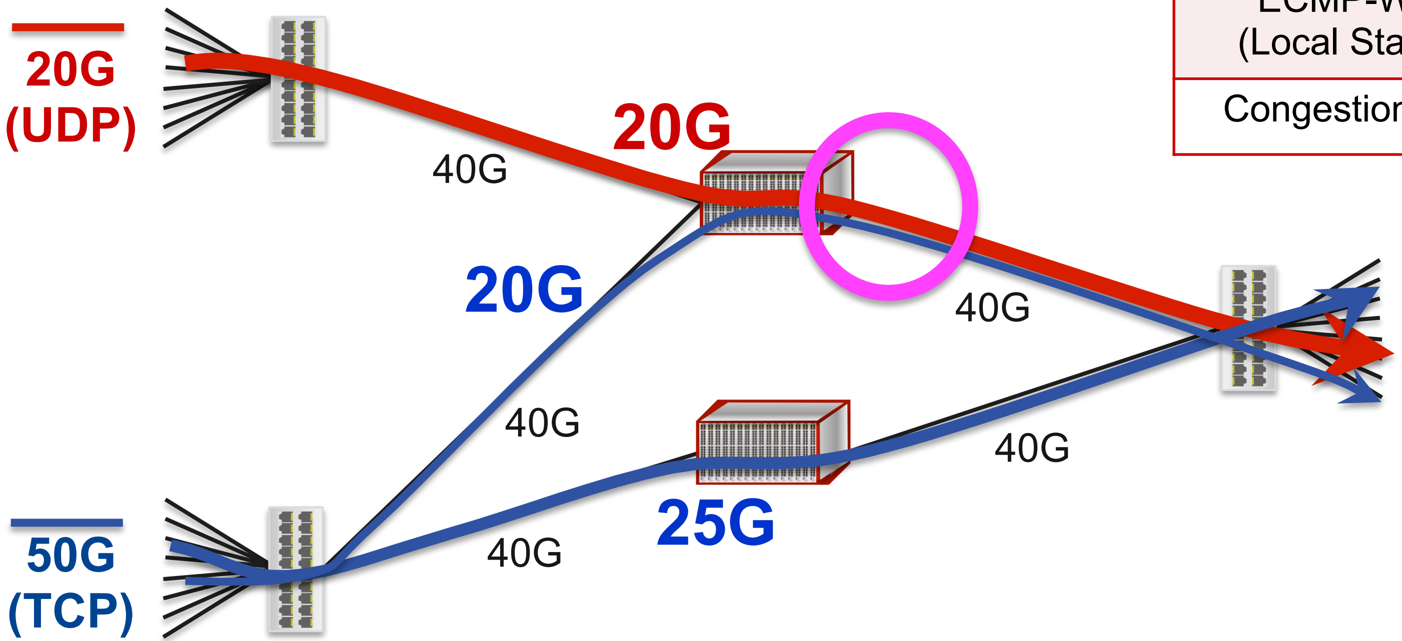


Dealing with Asymmetry

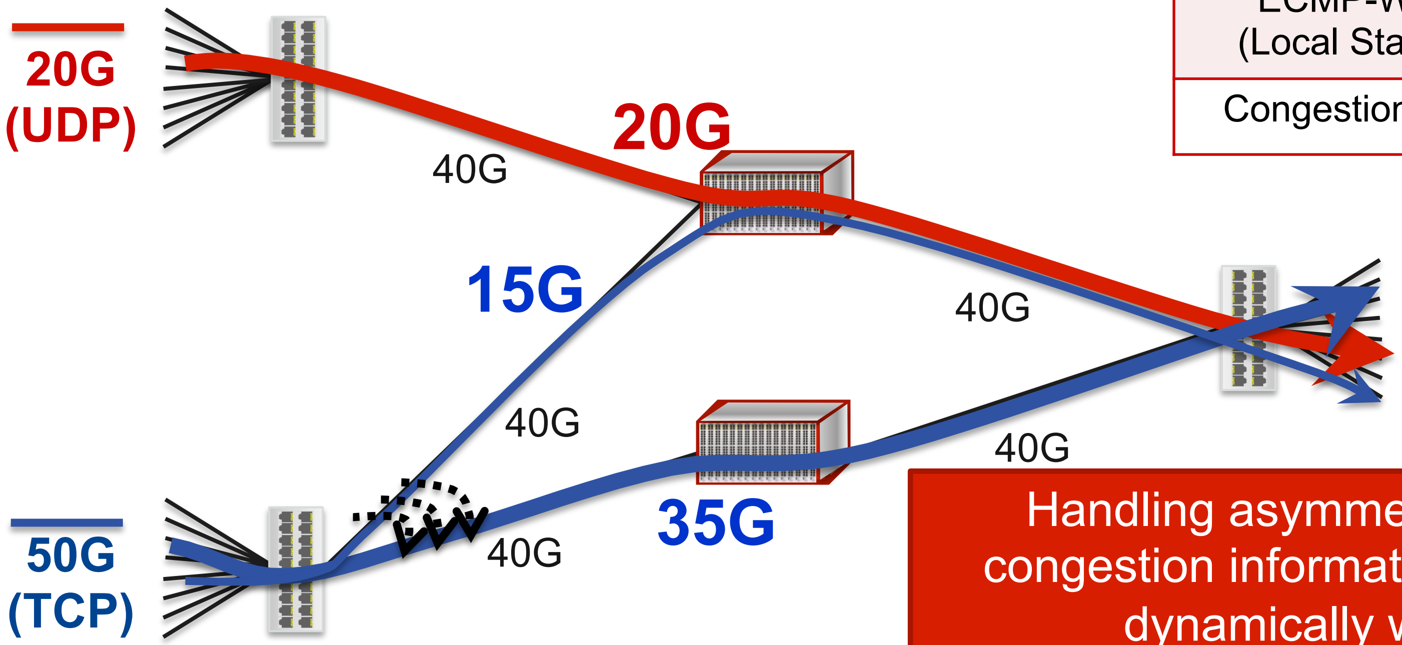


Dealing with Asymmetry

Scheme	Thrput
ECMP-WCMP (Local Stateless)	**65G**
Congestion-Aware	



Dealing with Asymmetry



Scheme	Thrput
ECMP-WCMP (Local Stateless)	**65G**
Congestion-Aware	**70G**

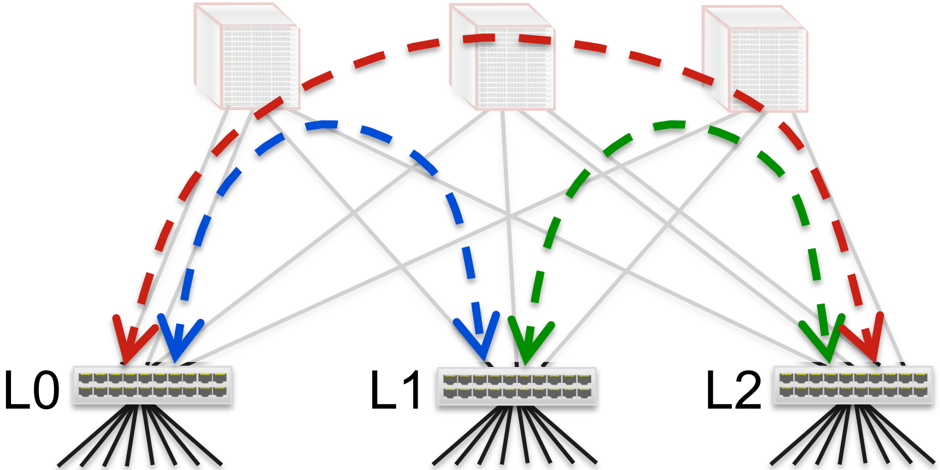
Handling asymmetry needs path congestion information, which varies dynamically with traffic.

Example: CONGA

1. Leaf switches (top-of-rack) track congestion to other leaves on different paths
2. Use this information to minimize bottleneck utilization

Dest Leaf \ Path	0	1	2	3
L1	5	3	7	2
L2	1	1	5	4

Congestion-To-Leaf Table @L0



Existing Load Balancing Schemes

Congestion-aware decisions: complex

- Measure and feed back congestion in real time
- CONGA, Hedera, HULA, MPTCP, FlowBender,…

Congestion-oblivious decisions: simple

- Random, round robin, hashing decision process
- ECMP, WCMP, Packet-Spray, Presto,…

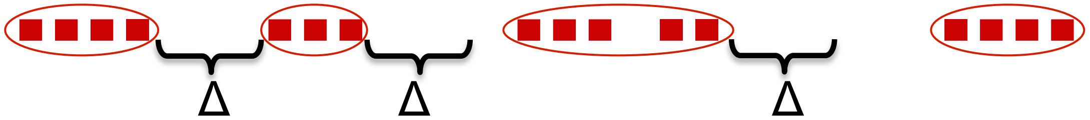
Is there a simple load balancing scheme (with congestion-oblivious decisions) that can handle asymmetry?

LetFlow

Simple:

Randomly assign Flowlets to available paths

Flowlets:



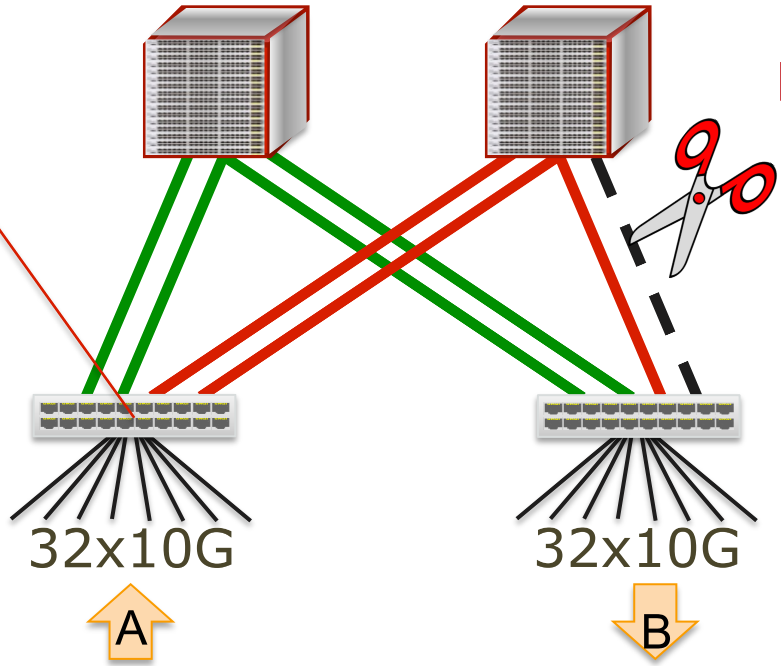
"Flowlets are bursts from same flow separated by at least Δ"

"the main origin of flowlets is the burstiness of TCP at RTT and sub-RTT scales."

Kandula et al, "[Dynamic load balancing without packet reordering](#)", (2007)

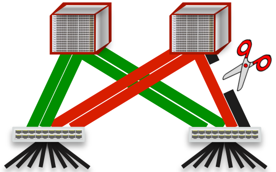
Simple Asymmetric Scenario

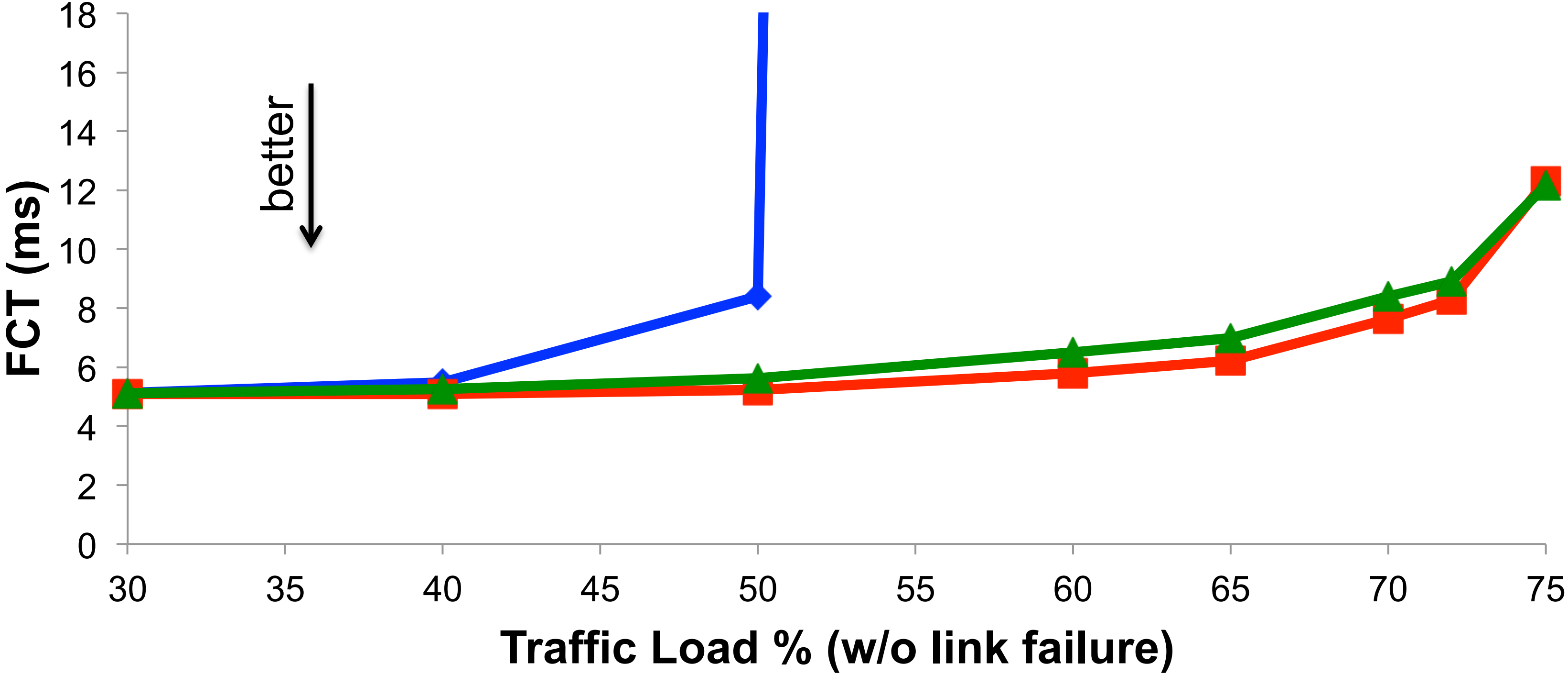
Detect and randomly assign Flowlets to available paths



Extremely simple!
- No measurements
- No feedback
- No congestion state

LetFlow

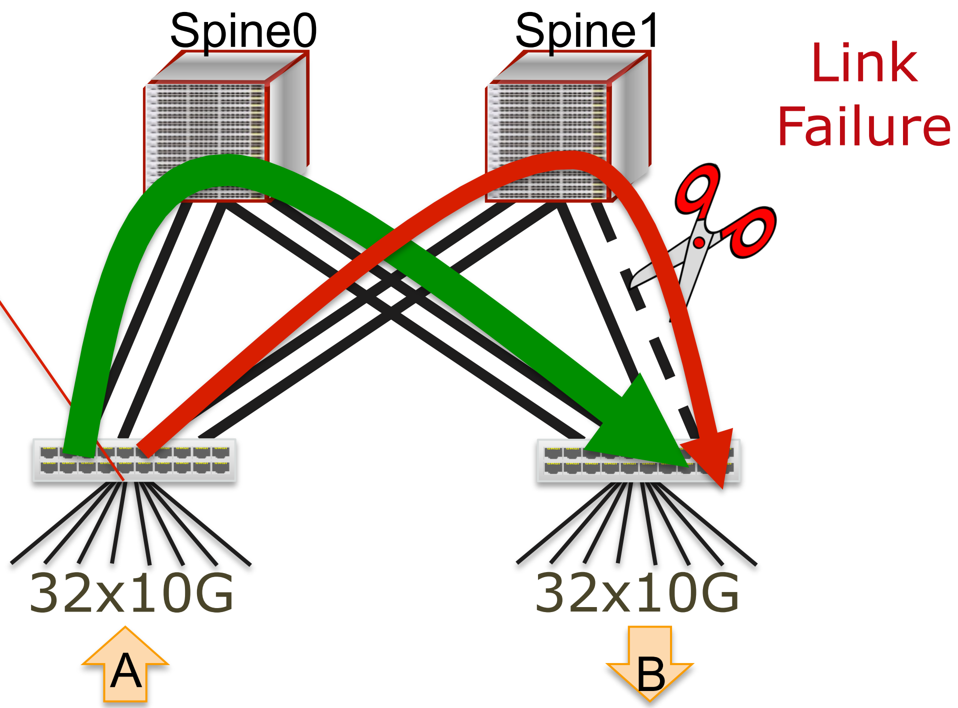




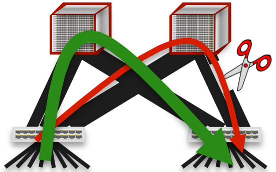
FCT (ms)
18
16
14
12
10
8
6
4
2
0
better
30
35
40
45
50
55
60
65
70
75
Traffic Load % (w/o link failure)

What's Going On?

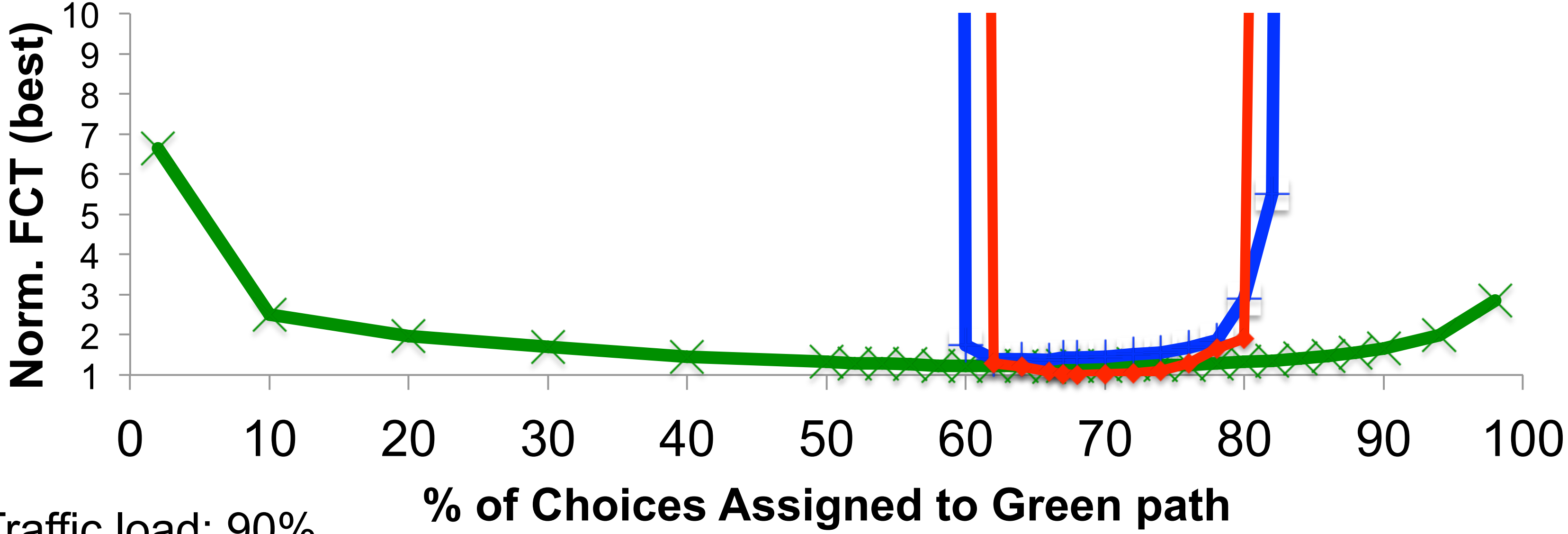
Force % of
choices
per path



Flowlets are Robust

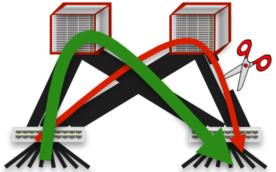


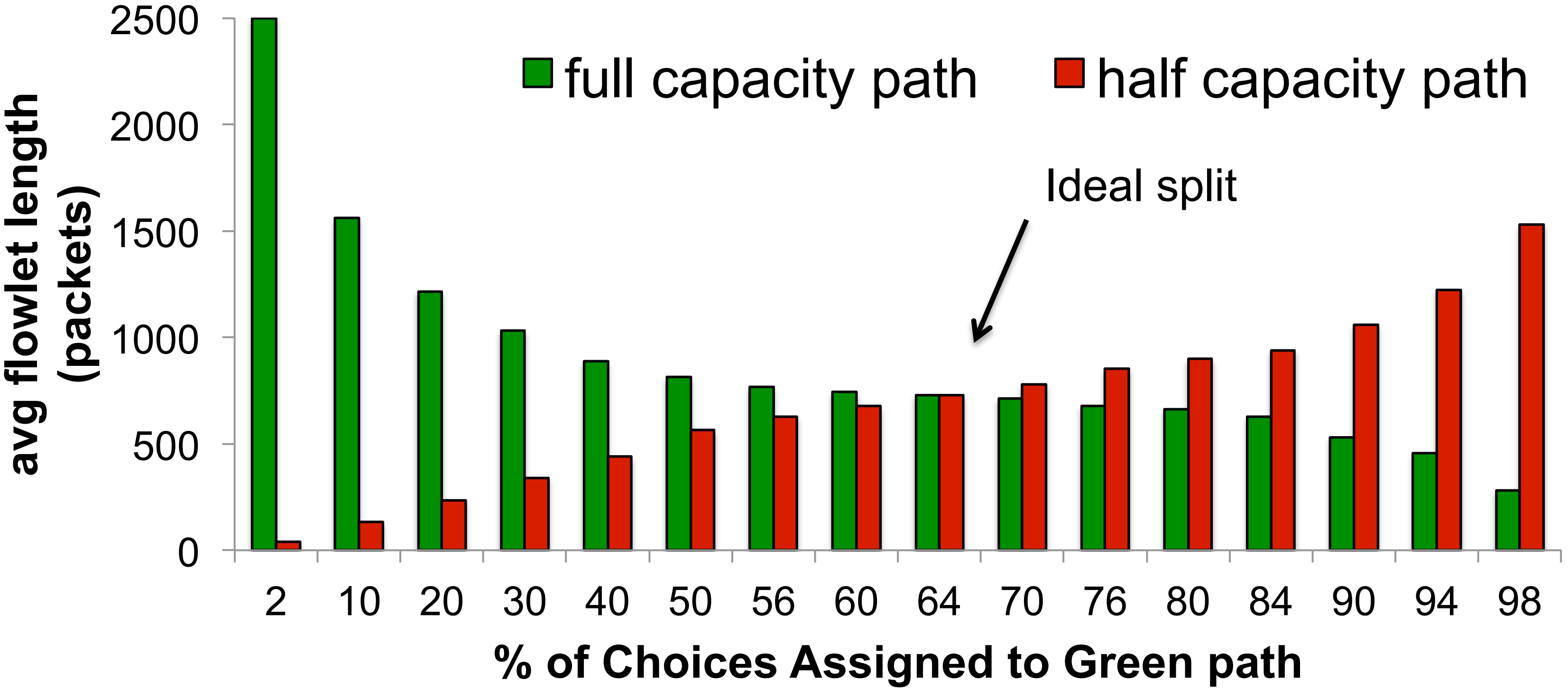
Performance is not sensitive to load balancing decisions



Traffic load: 90%

Flowlet Length





Flowlets are Elastic

- Flowlets change size based on congestion on the path
- Uncongested path → larger flowlets
- Congested path → smaller flowlets

→ Flowlet sizes **implicitly** encode path congestion information

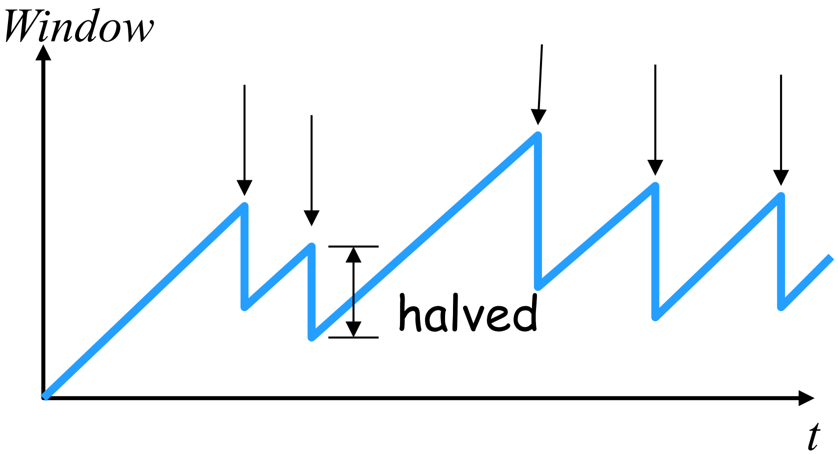
… this determines the amount of traffic on each path – not just load balancing decisions

LetFlow *is* congestion-aware,
despite simple random decisions

Why Are Flowlets Elastic?

- Because of congestion control (e.g., TCP)

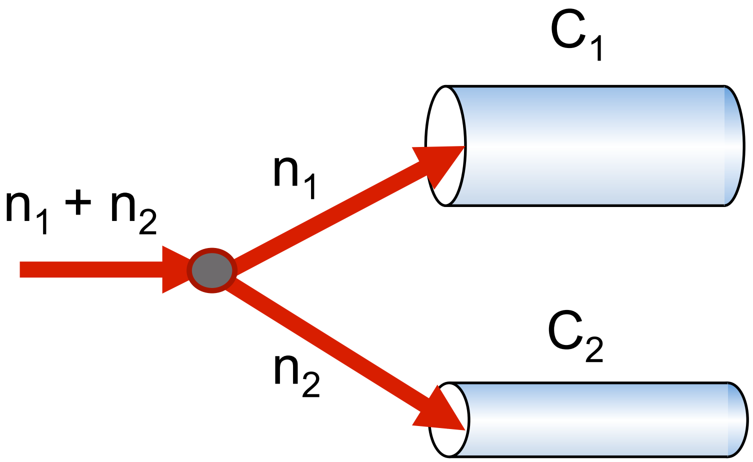
- A flowlet gap occurs on
  - Window cuts (Loss/ECN)
  - Latency spikes (ACK clocking)



- But, there's a more basic reason, applicable to any congestion control protocol …

LetFlow Analysis

- Assume flows transmit as Poisson processes

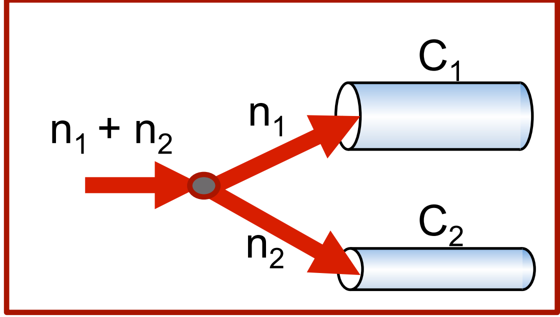


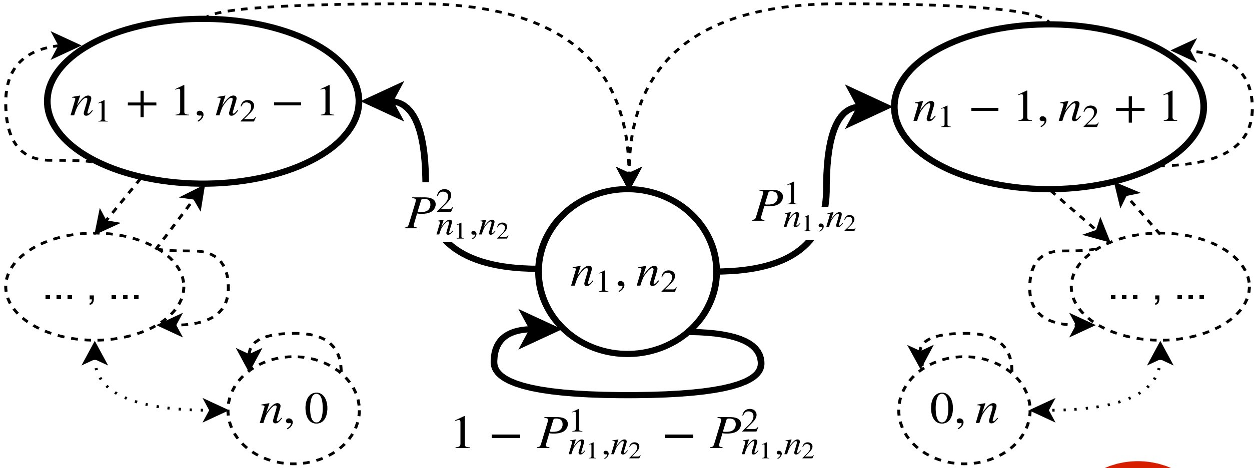
Avg. rate of each flow:

$$\lambda_1 = C_1 / n_1$$

$$\lambda_2 = C_2 / n_2$$

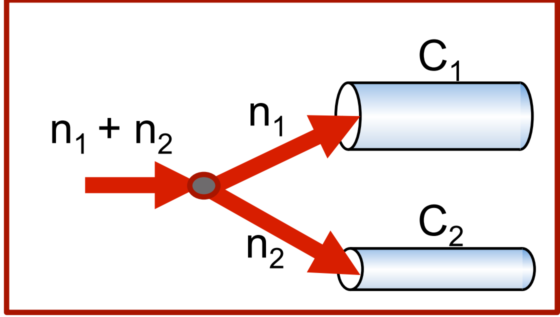
LetFlow Analysis





State transition probability $P^j_{n1,n2} \approx \frac{C_j}{2(C_1+C_2)} e^{-\lambda_j \Delta}, j \in \{1,2\}$

LetFlow Analysis



Takeaways

1. Flows move from low rate paths (small λ) to high rate paths (large λ)
2. The flowlet timeout (Δ) is important
   - Shouldn't be too small or large

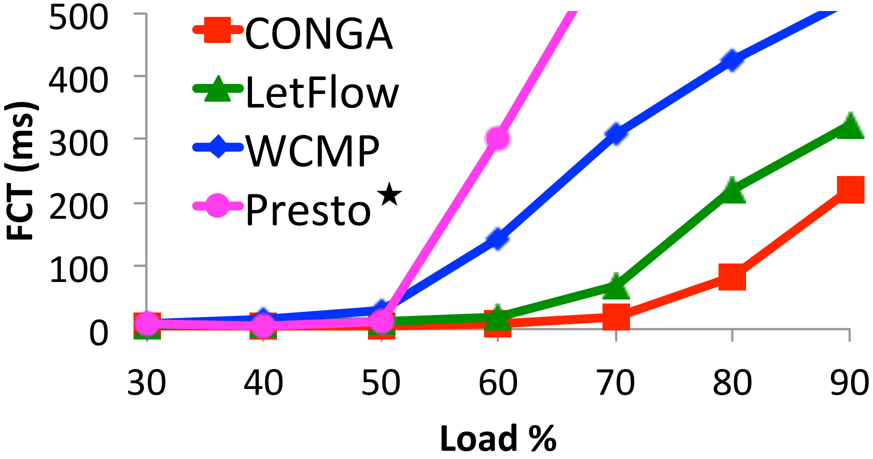
State transition probability $P^{j}_{n1,n2} \approx \frac{C_j}{2(C_1+C_2)} e^{-\lambda_j \Delta}, j \in \{1,2\}$

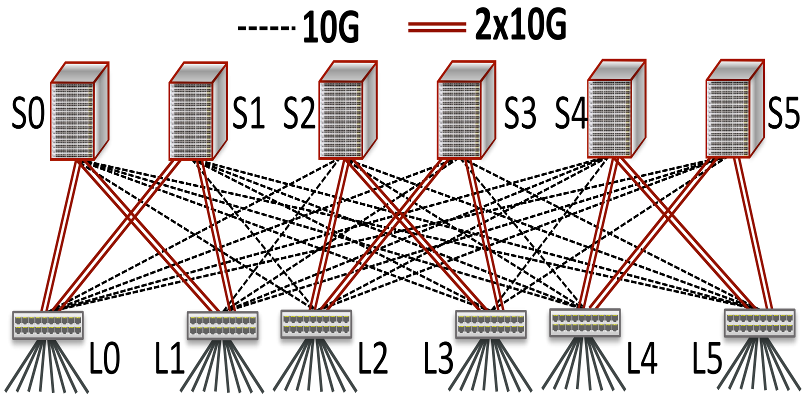
Experiments Summary

Different workloads: web search, data mining, enterprise

- Testbed experiments: ECMP, CONGA, LetFlow
  - 2 leaves 2 spines, 64 servers: symmetric & asymmetric topologies
- Simulations: ECMP, WCMP, Presto★, CONGA, LetFlow
  - Large topology: 6 leaves 6 spines, 288 servers
  - Complex asymmetric topologies: speed mismatch, combined workloads, multitier
  - Different protocols: TCP, DCTCP, DCQCN

Large Scale Simulations

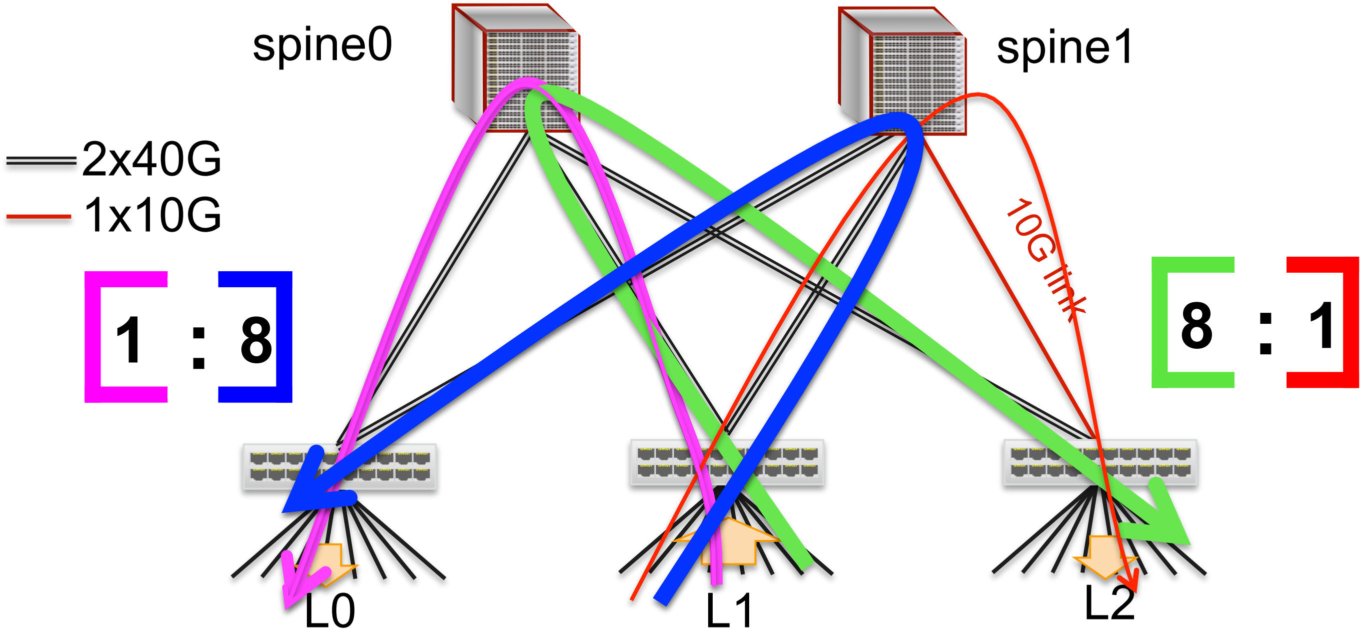




topology from: WCMP, Zhou, Junlan, et al. 2014

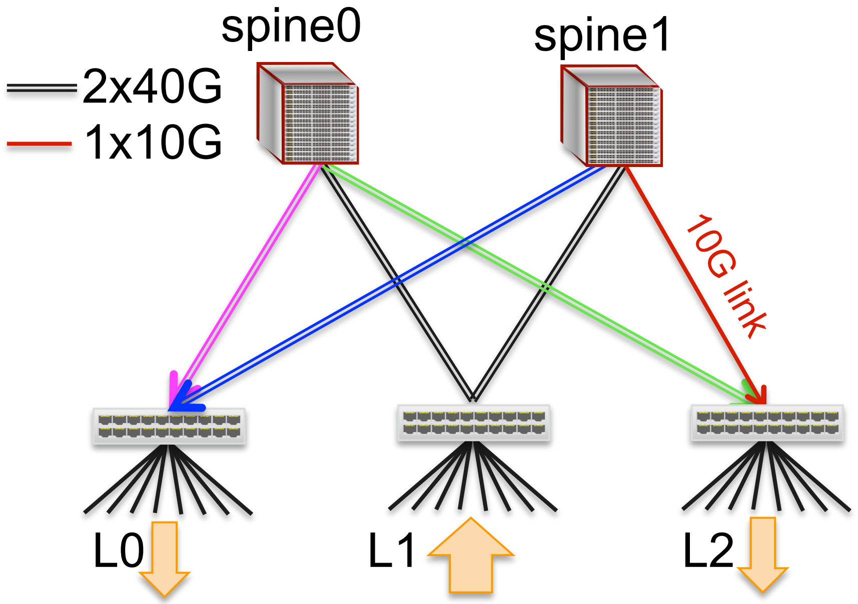
LetFlow within 2X of CONGA;
Both are much better than other schemes

Multi Destination Scenario



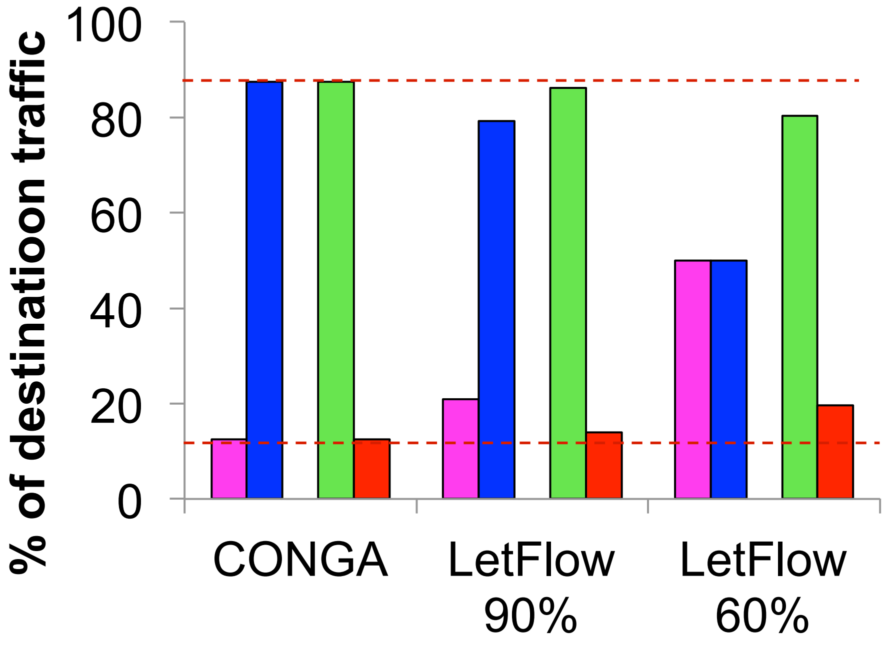
Traffic Load uniform to the 2 destinations

Multi Destination Scenario

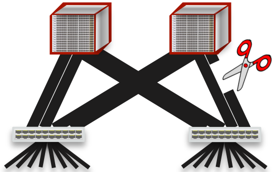


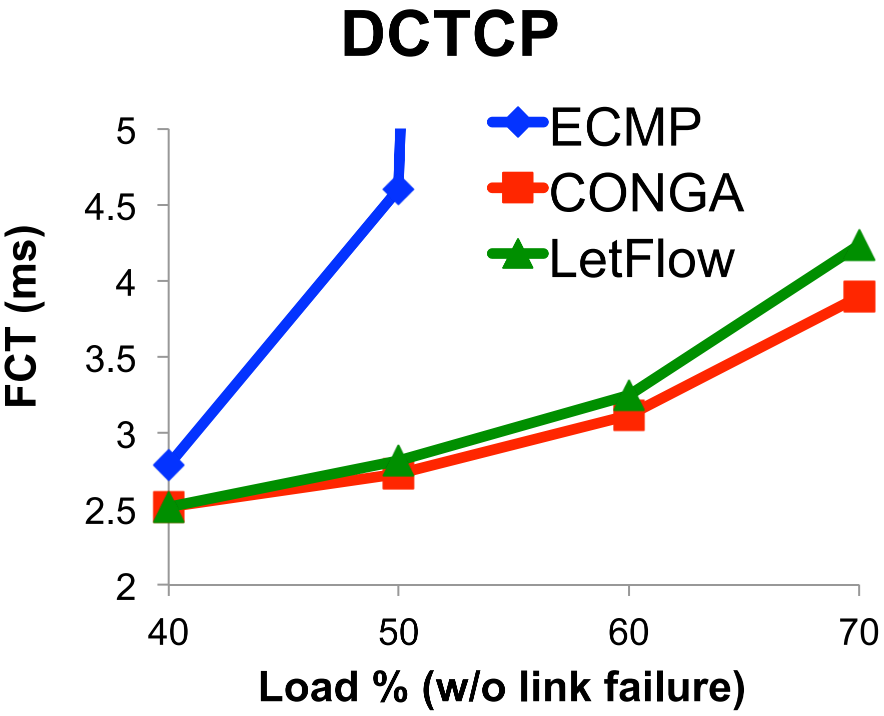
Traffic Load uniform to the 2 destinations

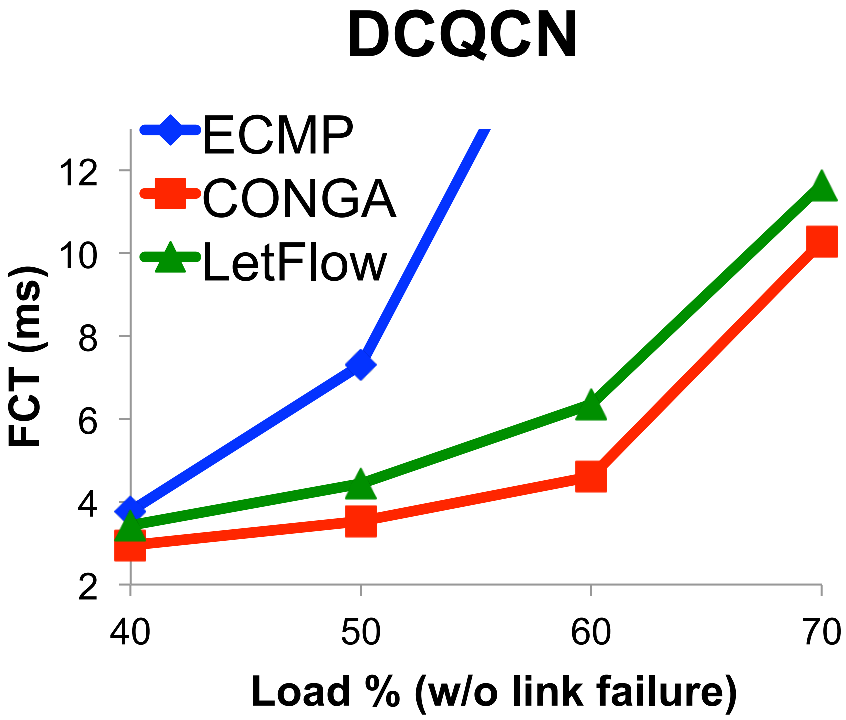
FCT is similar between Conga and LetFlow



Other Transport Protocols







Conclusion

- Flowlet switching is a powerful technique for asymmetric load balancing
- LetFlow: a simple LB mechanism that handles asymmetry
  - Random decisions but implicitly congestion-aware
  - Suitable for standalone switches – does not need feedback
- Letflow is stochastic and reactive in nature
  - Cannot proactively prevent congestion / queue buildup like more sophisticated schemes

LET it FLOW !