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)
Motivation

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

Single-rooted tree
- High oversubscription
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

![Diagram of multi-rooted tree network with Leaf and Spine nodes connected by multiple links, representing 1000s of server ports.](image-url)
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

1000s of server ports
Multi-rooted != Ideal DC Network

Ideal DC network:
Big output-queued switch

- No internal bottlenecks ➔ predictable
- Simplifies BW management
  [Bw guarantees, QoS, ...]
Multi-rooted != Ideal DC Network

Ideal DC network:
- Big output-queued switch
- No internal bottlenecks ➔ predictable
- Simplifies BW management
  [Bw guarantees, QoS, ...]

Can’t build it 😞

Multi-rooted tree

≈

Possible bottlenecks
Multi-rooted != Ideal DC Network

Ideal DC network:
Big output-queued switch

Can’t build it 😞

Multi-rooted tree

1000s of server ports

Need precise load balancing
Today: ECMP Load Balancing

Pick among equal-cost paths by a hash of 5-tuple

- Approximates Valiant load balancing
- Preserves packet order

\[ H(f) \% 3 = 0 \]
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)

$$H(f) \equiv 3 = 0$$
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 & stateless (v. bad with asymmetry due to link failures)
Dealing with Asymmetry

Handling asymmetry needs non-local knowledge
Dealing with Asymmetry

Handling asymmetry needs non-local knowledge
Dealing with Asymmetry

Handling asymmetry needs non-local knowledge
Dealing with Asymmetry

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Thrput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECMP (Local Stateless)</td>
<td></td>
</tr>
<tr>
<td>Local Cong-Aware</td>
<td></td>
</tr>
<tr>
<td>Global Cong-Aware</td>
<td></td>
</tr>
</tbody>
</table>
Dealing with Asymmetry: ECMP

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Thrput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECMP (Local Stateless)</td>
<td>60G</td>
</tr>
<tr>
<td>Local Cong-Aware</td>
<td></td>
</tr>
<tr>
<td>Global Cong-Aware</td>
<td></td>
</tr>
</tbody>
</table>
Dealing with Asymmetry: Local Congestion-Aware

Interacts poorly with TCP’s control loop

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Thrput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECMP (Local Stateless)</td>
<td>60G</td>
</tr>
<tr>
<td>Local Cong-Aware</td>
<td></td>
</tr>
<tr>
<td>Global Cong-Aware</td>
<td></td>
</tr>
</tbody>
</table>
Dealing with Asymmetry: Local Congestion-Aware

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Thrput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECMP (Local Stateless)</td>
<td>60G</td>
</tr>
<tr>
<td>Local Cong-Aware</td>
<td>50G</td>
</tr>
<tr>
<td>Global Cong-Aware</td>
<td></td>
</tr>
</tbody>
</table>

Interacts poorly with TCP’s control loop
Dealing with Asymmetry: Global Congestion-Aware

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Thruput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECMP (Local Stateless)</td>
<td>60G</td>
</tr>
<tr>
<td>Local Cong-Aware</td>
<td>50G</td>
</tr>
<tr>
<td>Global Cong-Aware</td>
<td></td>
</tr>
</tbody>
</table>
Dealing with Asymmetry: Global Congestion-Aware

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Thrput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECMP (Local Stateless)</td>
<td>60G</td>
</tr>
<tr>
<td>Local Cong-Aware</td>
<td>50G</td>
</tr>
<tr>
<td>Global Cong-Aware</td>
<td>70G</td>
</tr>
</tbody>
</table>
Dealing with Asymmetry: 
Global Congestion-Aware

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Thrput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECMP (Local Stateless)</td>
<td>60G</td>
</tr>
<tr>
<td>Local Cong-Aware</td>
<td>50G</td>
</tr>
<tr>
<td>Global Cong-Aware</td>
<td>70G</td>
</tr>
</tbody>
</table>

Global CA > ECMP > Local CA

Local congestion-awareness can be worse than ECMP
Global Congestion-Awareness (in Datacenters)

Opportunity → Latency → microseconds

Challenge → Topology → simple, regular

Traffic → volatile, bursty

Datacenter
Global Congestion-Awareness (in Datacenters)

Opportunity → Latency
Challenge → Topology

Datacenter

Simple & Stable
Responsive

Key Insight:
Use extremely fast, low latency distributed control
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.
CONGÁ’S DESIGN
Design

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

Track path-wise congestion metrics (3 bits) between each pair of leaf switches
Design: Leaf-to-Leaf Feedback

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

- **Congestion-To-Leaf Table @L0**
  - L0 → L2
  - Path=2
  - CE=0

- **Congestion-From-Leaf Table @L2**
  - L0 → L2
  - Path=2
  - CE=5

- pkt.CE $\leftarrow \max(\text{pkt.CE}, \text{link.util})$
Design: Leaf-to-Leaf Feedback

Track path-wise congestion metrics (3 bits) between each pair of leaf switches
Design: LB Decisions

Send each packet on least congested path

flowlet [Kandula et al 2007]
CONGA aggregates paths based on leaf uplink
Why is this Stable?

Stability usually requires a sophisticated control law (e.g., TeXCP, MPTCP, etc)
Why is this Stable?

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

Near-zero latency + flowlets $\Rightarrow$ stable
How Far is this from Optimal?

Given traffic demands $[\lambda_{ij}]$:

\[
\max_{l \in \text{Links}} \rho_l \text{ with CONGA}
\]

\[
\min_{f \in \text{feasible}} \max_{l \in \text{Links}} \rho_l
\]

Price of Anarchy

bottleneck routing game

(Banner & Orda, 2007)
How Far is this from Optimal?

Given traffic demands $[\lambda_{ij}]$:

$$\max_{l \in \text{Links}} \rho_l \text{ with CONGA}$$

$$\min_{f \in \text{feasible}} \max_{l \in \text{Links}} \rho_l$$

Price of Anarchy

bottleneck routing game
(Banner & Orda, 2007)

Theorem: PoA of CONGA = 2
Implementation

Implemented in silicon for Cisco’s new flagship ACI datacenter fabric

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

- Die area: <2% of chip
Evaluation

Testbed experiments
- 64 servers, 10/40G switches
- Realistic traffic patterns (enterprise, data-mining)
- HDFS benchmark

Large-scale simulations
- OMNET++, Linux 2.6.26 TCP
- Varying fabric size, link speed, asymmetry
- Up to 384-port fabric

40G fabric links

32x10G

32x10G

32x10G

32x10G

Link Failure
Empirical Benchmark
Symmetric (no failure)

Elephant flows (>10MB)

Mice flows (<100KB)
Empirical Benchmark
Symmetric (no failure)

CONGA/ECMP up-to 40% better than MPTCP for mice

CONGA/MPTCP up-to 35% better than ECMP for elephants
HDFS Benchmark
1TB Write Test, 40 runs

Cloudera hadoop-0.20.2-cdh3u5, 1 NameNode, 63 DataNodes

Link failure has almost no impact with CONGA

~2x better than ECMP

no link failure
Decouple DC LB & Transport

Big Switch Abstraction
(provided by network)

ingress & egress
(managed by transport)
Conclusion

CONGA: Globally congestion-aware LB for DC
... implemented in Cisco ACI datacenter fabric

Key takeaways

1. In-network LB is right for DCs
2. Low latency is your friend; makes feedback control easy
Thank You!