Data centers: network topology

Ankit Singla

ETH Zürich Spring 2018
A server rack

A top-of-rack switch

A rack of servers
Lots of racks

How to network the racks?

Facebook: machine-machine traffic “doubling at an interval of less than a year”
Need high throughput networks to ...

- Support big data analytics
- Ease virtual machine placement
“Big switch” approach

Big crossbar
“Big switch” approach
Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google’s Datacenter Network

Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat

Google, Inc.
Side-note: workload management

If your Map-Reduce job needs 24 servers, which ones would you take?
“Big switch” approach
Alternative: tree network
Alternative: tree network
Alternative: tree network
Alternative: tree network

Congestion
Build with identical switches throughout?

“Scaling out” vs. “scaling up”
Connect many cheap, identical switches?

Goals: high capacity, low latency
If you know your application ...
... design for it

“Hopper”, NERSC
But, other apps may not work well ...

We want general purpose design!
What’s so hard about this?

Number of switches

Number of possible networks

4-port switches

6-port switches
How would YOU think about this problem?!
So people pick known good candidates
## Hypercube

<table>
<thead>
<tr>
<th>#Dim</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NerdBoy1392 [CC BY-SA 3.0], via Wikimedia
Clos networks

Use small, cheap elements to build large networks!!
Folded-Clos?
Folded-Clos?
Fat-tree
Fat-tree network

A Scalable, Commodity Data Center Network Architecture

Mohammad Al-Fares  Alexander Loukissas  Amin Vahdat

ACM SIGCOMM, 2008
Fat-tree network

$K = 6$

Image source: Francesco Celestino
Fat-tree network

A Scalable, Commodity Data Center Network Architecture

Mohammad Al-Fares  Alexander Loukissas  Amin Vahdat

ACM SIGCOMM, 2008
Fat-tree network
A Scalable, Commodity Data Center Network Architecture

Mohammad Al-Fares  Alexander Loukissas  Amin Vahdat

ACM SIGCOMM, 2008
A Scalable, Commodity Data Center Network Architecture
Fat-tree network

The fat-tree network is a type of network topology used in data centers. It is characterized by a tree-like structure with multiple layers of switches and routers. Each layer is connected to the next layer through a series of switches, allowing for a high degree of interconnectivity.

The diagram illustrates a fat-tree network with multiple layers of switches and routers. Each layer is connected to the next layer through a series of switches, allowing for a high degree of interconnectivity. The network is designed to support a high degree of bandwidth and scalability, making it suitable for use in large-scale data centers.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM SIGCOMM, 2008

A Scalable, Commodity Data Center Network Architecture

Mohammad Al-Fares  Alexander Loukissas  Amin Vahdat
Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google’s Datacenter Network

Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat
Google, Inc.
Explore Wiring Fail, Wiring Jobs, and more!  

Poor data center cable management. I'm expecting Shelob the spider from Lord of the Rings to emerge any moment.

See More

Aaaah! What a horrible data center disaster.

See More

Un giro nei datacenter di #Google

See More

Server Room disaster

See More
Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google’s Datacenter Network

Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat

Google, Inc.
Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google’s Datacenter Network

Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat

Google, Inc.
2.56Tbps of external connectivity in Watchtower fabrics enabled clusters to connect to inter cluster networks as another possible. The CBRs developed for WCC enhanced the relationship between physical distance and network software of the CBR switch would find application in between clusters. Moreover, the modular hardware and CBRs and the inter-cluster networking switches. CBR switch in these blocks and an external switch as Link to connect to external fabrics.

To those used for ToR connectivity. However, we reallocated one to three aggregation blocks. These aggregated interfaces could be deployed with IGP (Section 5.2) with external routing protocols. Places where we would have to integrate our in-house facing configuration change and because it limited the spreading peering functionality across the entire set of switches to peer with external routers rather than choosing option iv) because we wanted an isolated layer of external fabrics.

Figure 15:

Four options to connect to the external network or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage or other than personal use.

ACM SIGCOMM, 2015

Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google’s Datacenter Network

Centralized Control in Google’s Datacenter Network

Data center networks; modern storage infrastructures, and are a key enabler for cloud computing. Bandwidth demands in the cloud computing have substantially increased. The WAN connectivity layer on the north-facing CBR layer to the Datacenter Freedome, typically stays independent blocks to connect multiple clusters in the same campus. Campus Freedome.

The Freedome Block employs eBGP to connect to both north-facing peers. We use iBGP internal to the Freedome Blocks as shown in the top figure. Each block exposes 8x more south-facing ports (cluster facing) than north-facing ports (next-level in the hierarchy). Each Freedome Border Router ports to the campus connectivity layer to the north. The bottom left figure in Fig.

Figure 16:

A Datacenter Freedome typically comprises 4 independent Freedome Blocks to connect multiple clusters in the same campus. Two-stage fabrics used for inter-cluster and local to the campus at lower cost than existing solutions.

Figure 17:

The proliferation of Internet-connected sensors. As a result, internet connectivity is expected to explode with more photo/video content, logs, and datacenter bandwidth makes it prudent to trade cost for a range of protocols (e.g., IP multicast) or by pushing the envelope of chip memory (e.g., highest end switches available at any point in time [24].

WAN switches were different in scale and because of the complexity associated with traditional datacenter network architectures to be prohibitive. Maximum network scale was limited by the cost and capacity of the WAN. Intra-campus connectivity.

Increasing the performance and availability of WAN Internet deployment, losing a single switch/router can have substantial impact on applications. Because WAN Internet deployment, losing a single switch/router can have substantial impact on applications. Because WAN Internet deployment, losing a single switch/router can have substantial impact on applications. Because WAN Internet deployment, losing a single switch/router can have substantial impact on applications.
Variants of this design are common.

Link-state network carrying only LAs (e.g., 10/8)

Internet

**Int** (D/2 x Intermediate Switches)

**Aggr** (2x10G)

**ToR**

20(D/2 x 10G)

D x Aggregate Switches

D x Aggregates Switches

20(D/4 D x ToR Switches)

Fungible pool of servers owning AAs (e.g., 20/8)

VL2 @ Microsoft, ACM SIGCOMM’09

Greenburg, Hamilton, Jain, Kandula, Kim, Lahiri, Maltz, Patel, Sengupta

“Introducing data center fabric, the next-generation Facebook data center network”, Alexey Andreyev, 2015
Connect many cheap, identical switches?

But … can we do better?
Yes …
Yes ...
Shorter paths $\Rightarrow$ high capacity

A packet that travels on a short path consumes a small amount of network capacity
A simple upper bound on throughput

\[
\# \text{ flows} \cdot \text{ capacity used per flow} \leq \text{ total capacity}
\]
A simple upper bound on throughput

# flows \cdot \text{capacity used per flow} \leq \text{total capacity}
A simple upper bound on throughput

\# \text{flows} \times \text{throughput per flow} \times \text{mean path length} \leq \text{total capacity}
A simple upper bound on throughput

\[
\text{throughput per flow} \leq \frac{\text{total capacity}}{\# \text{ flows} \cdot \text{mean path length}}
\]
A simple upper bound on throughput

\[
\text{throughput per flow} \leq \frac{\sum_{\text{links}} \text{capacity}(\text{link})}{\# \text{ flows} \cdot \text{mean path length}}
\]

Lower bound this!

[Arxiv 2013, NSDI 2014] High Throughput Data Center Topology Design
Ankit Singla, P. Brighten Godfrey, Alexandra Kolla
Lower bound on mean path length
Lower bound on mean path length

<table>
<thead>
<tr>
<th>Distance</th>
<th># Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>$6^2 - 6$</td>
</tr>
</tbody>
</table>

(Ugliness omitted)

[Cerf et al., “A lower bound on the average shortest path length in regular graphs”, 1974]
... but cuts matter too
... but cuts matter too
Shorter paths ⇒ high capacity

A packet that travels on a short path consumes a small amount of network capacity
Degree-diameter problem

“Out of all these, give me one with lowest diameter ...”
Degree-diameter problem

What is the maximum number of nodes in any graph with degree $\partial$ and diameter $d$?

Petersen graph
### Degree-diameter problem

<table>
<thead>
<tr>
<th>k</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>10</td>
<td>20</td>
<td>38</td>
<td>70</td>
<td>132</td>
<td>196</td>
<td>336</td>
<td>600</td>
<td>1250</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>41</td>
<td>96</td>
<td>364</td>
<td>740</td>
<td>1320</td>
<td>3243</td>
<td>7575</td>
<td>17703</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>72</td>
<td>210</td>
<td>624</td>
<td>2772</td>
<td>5516</td>
<td>17030</td>
<td>57840</td>
<td>187056</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>110</td>
<td>390</td>
<td>1404</td>
<td>7917</td>
<td>19383</td>
<td>76461</td>
<td>307845</td>
<td>1253615</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>168</td>
<td>672</td>
<td>2756</td>
<td>11988</td>
<td>52768</td>
<td>249660</td>
<td>1223050</td>
<td>6007230</td>
</tr>
<tr>
<td>8</td>
<td>57</td>
<td>253</td>
<td>1100</td>
<td>5060</td>
<td>39672</td>
<td>131137</td>
<td>734820</td>
<td>4243100</td>
<td>24897161</td>
</tr>
<tr>
<td>9</td>
<td>74</td>
<td>585</td>
<td>1550</td>
<td>8200</td>
<td>75893</td>
<td>279616</td>
<td>1686600</td>
<td>12123288</td>
<td>65866350</td>
</tr>
<tr>
<td>10</td>
<td>91</td>
<td>650</td>
<td>2286</td>
<td>13140</td>
<td>134690</td>
<td>583083</td>
<td>4293452</td>
<td>27997191</td>
<td>201038922</td>
</tr>
<tr>
<td>11</td>
<td>104</td>
<td>715</td>
<td>3200</td>
<td>19500</td>
<td>156864</td>
<td>1001268</td>
<td>7442328</td>
<td>72933102</td>
<td>600380000</td>
</tr>
<tr>
<td>12</td>
<td>133</td>
<td>786</td>
<td>4680</td>
<td>29470</td>
<td>359772</td>
<td>1999500</td>
<td>15924326</td>
<td>158158875</td>
<td>1506252500</td>
</tr>
<tr>
<td>13</td>
<td>162</td>
<td>851</td>
<td>6560</td>
<td>40260</td>
<td>531440</td>
<td>3322080</td>
<td>29927790</td>
<td>249155760</td>
<td>3077200700</td>
</tr>
<tr>
<td>14</td>
<td>183</td>
<td>916</td>
<td>8200</td>
<td>57837</td>
<td>816294</td>
<td>6200460</td>
<td>55913932</td>
<td>600123780</td>
<td>7041746081</td>
</tr>
<tr>
<td>15</td>
<td>186</td>
<td>1215</td>
<td>11712</td>
<td>76518</td>
<td>1417248</td>
<td>8599986</td>
<td>90001236</td>
<td>1171988164</td>
<td>1001234898</td>
</tr>
<tr>
<td>16</td>
<td>198</td>
<td>1600</td>
<td>14640</td>
<td>132496</td>
<td>1771560</td>
<td>14882658</td>
<td>140559416</td>
<td>2025125476</td>
<td>12951451931</td>
</tr>
</tbody>
</table>

[Wikipedia: https://en.wikipedia.org/wiki/Table_of_the_largest_known_graphs_of_a_given_diameter_and_maximal_degree]
Degree-diameter problem

Just use the best-known degree-diameter graphs?!

[NSDI 2012] Jellyfish: Networking Data Centers Randomly
Ankit Singla, Chi-Yao Hong, Lucian Popa, P. Brighten Godfrey

Maciej Besta, Torsten Hoefler
Degree-diameter problem

Just use the best-known degree-diameter graphs?!

Problem: Lack of flexibility
Today’s structured networks
Lack of flexibility ... 

Coarse design points

- Hypercube: $2^k$ switches
- de Bruijn-like: $3^k$ switches
- 3-level fat tree: $5k^2/4$ switches

Facebook “adding capacity on a daily basis”

Unclear how to maintain structure incrementally

- Over-utilize switches? Uneven / constrained bandwidth
- Leave ports free for later? Wasted investment
Lack of flexibility ...

Coarse design points

- Hypercube: $2^k$ identical switches
- de Bruijn-like: $3^k$ identical switches
- 3-level fat tree: $5k^2/4$ identical switches

Unclear how to maintain structure incrementally

- Over-utilize switches? Uneven / constrained bandwidth
- Leave ports free for later? Wasted investment
How do we handle heterogeneity?
Forget about structure – let’s have no structure at all!
Jellyfish

Servers connected to top-of-rack switch

Switches form uniform-random interconnections
Out of this large space, pick uniformly at random!
Capacity as a fluid

Jellyfish random graph

432 servers, 180 switches, degree 12
Capacity as a fluid

Jellyfish random graph
432 servers, 180 switches, degree 12

Jellyfish
Crossota norvegica
Photo: Kevin Raskoff
Really? Random could work?

Uniform-Weight Synthetic Workloads

categories of synthetic workloads – (a) uniform weight

performance of

evaluation of error bars in networks of size greater than

value is referred to as

comparison against other

throughput with the random graph's throughput for

graph

compute the throughput of

1) Synthetic Workloads:

We evaluate three traffic matrices with equal weight

100 iterations, and all error bars are

Uniform-Weight Synthetic Workloads

across flows and (b) non-uniform weight across flows.

categories of synthetic workloads – (a) uniform weight

performance of

10

otherwise stated, each data-point is an average across

value is referred to as

comparison against other

throughput with the random graph's throughput for

graph

compute the throughput of

1) Synthetic Workloads:

We evaluate three traffic matrices with equal weight

100 iterations, and all error bars are

Comparision of TMs on topologies

(a) All to All TM

(b) Random Matching TM

(c) Longest Matching TM

Comparison of TMs on topologies

(a) All to All TM

(b) Random Matching TM

(c) Longest Matching TM

Comparision of TMs on topologies

2

6	×

100

6	×

1000

6	×

10000

Most topologies are in Figure 5, while Figure 6 shows

across flows: all to all, random matching with one

scale.

formance (relative throughput

in all TMs, Jellyfish achieves consistently highest per-

formance at the largest scale. However,

longest matching. Furthermore, for the networks of Figure 5,

pliance varies substantially, by around

2

6	×

100

6	×

1000

6	×

10000

Overall,

Jellyfish, Slim Fly, Long Hop, and HyperX.

server, and longest matching. Figures 5 and 6 show

across flows: all to all, random matching with one
Really? Random could work?

![Graph showing comparison of different topologies such as HyperX, Jellyfish, Long Hop, and Slim Fly across different scales of servers. The graph illustrates relative throughput compared to Jellyfish, with a focus on how different topologies perform under various traffic matrix types such as All to All, Random Matching, and Longest Matching. The x-axis represents the number of servers, while the y-axis shows the relative throughput. The graph highlights the performance of Jellyfish as the top topology under All to All traffic and more than 1000 servers, with other topologies performing less consistently.](image-url)
Really? Random could work?

Number of servers

Rel. Throughput

HyperX

Jellyfish

Long Hop

Slim Fly

Relative Throughput

0.2

0.4

0.6

0.8

1

1.2

1.4

1.6

100  1000  10000

Number of servers

Jellyfish
In fact, nothing will do much better!

A simple upper bound on throughput

\[
\text{throughput per flow} \leq \frac{\sum_{\text{links}} \text{capacity}(\text{link})}{\# \text{flows} \cdot \text{mean path length}}
\]

Lower bound this!
Random graphs vs. bound

Throughput (Ratio to Upper-bound) vs. Network Size
Random graphs vs. bound

Random graphs within a few percent of optimal!
These stunts are performed by trained professionals …
Impact so far ...
Xpander: Towards Optimal-Performance Datacenters

ABSTRACT

Despite extensive efforts to meet ever-growing demands, today's datacenters often exhibit far-from-optimal performance in terms of network utilization, resiliency to failures, cost efficiency, incremental expandability, and more. Consequently, many novel architectures for high performance datacenters have been proposed. We show that the benefits of state-of-the-art proposals are, in fact, derived from the fact that they are (implicitly) utilizing "expander graphs" (aka expanders) as their network topologies, thus unveiling a unifying theme of these proposals. We observe, however, that these proposals are not optimal with respect to performance, do not scale, or suffer from seemingly insurmountable deployment challenges. We leverage these insights to present Xpander, a novel datacenter architecture that achieves near-optimal performance and provides a tangible alternative to existing datacenter designs. Xpander's design turns ideas from the rich graph-theoretic literature on constructing optimal expanders into an operational reality. We evaluate Xpander via theoretical analyses, extensive simulations, experiments with a network emulator, and an implementation on an SDN-capable network testbed. Our results demonstrate that Xpander significantly outperforms both traditional and proposed datacenter designs. We discuss challenges to real-world deployment and explain how these can be resolved.
3.3 Watchtower: Global Deployment

Our deployment experience with Firehose 1.1 was largely positive. We showed that services could enjoy substantially more bandwidth than with traditional architectures, all with lower cost per unit bandwidth. Firehose 1.1 went into production with a handful of clusters and remained operational until recently. The main drawback to Firehose 1.1 was the deployment challenges with the external copper cabling. We used these experiences to design Watchtower, our third-generation cluster fabric. The key idea was to leverage the next-generation merchant silicon switch chips, 16x10G, to build a traditional switch chassis with a backplane. Figure 9 shows the half rack Watchtower chassis along with its internal topology and cabling. Watchtower consists of eight line cards, each with three switch chips. Two chips on each line card have half their ports externally facing, for a total of 16x10GE SFP+ ports. All three chips also connect to a backplane for port to port connectivity. Watchtower deployment, as seen in Figure 9, was substantially easier than the earlier Firehose deployments. The larger bandwidth density of the switching silicon also allowed us to build larger fabrics with more bandwidth to individual servers, a necessity as servers were employing an ever-increasing number of cores.

Fiber bundling further reduced the cabling complexity of Watchtower clusters. Figure 10 shows a Watchtower fabric deployment without any cable bundling. Individual fibers of varying length need to be pulled from each chassis location, leading to significant deployment overhead. The bottom figure shows how bundling can substantially reduce complexity. We deploy two chassis in each rack and co-locate two racks. We can then pull cable bundles to the midpoint of the co-located racks, where each bundle is split to each rack and then further to each chassis. Finally, manufacturing fiber in bundles is more cost effective than individual strands. Cable bundling helped reduce fiber cost (capex + opex) by nearly 40% and expedited bringup of Watchtower fabric by multiple weeks. Table 3 summarizes the bundling and cost savings.
Augmenting Data Center Networks with Multi-Gigabit Wireless Links

Daniel Halperin*, Srikanth Kandula*, Jitendra Padhye†, Paramvir Bahl*, and David Wetherall*
Microsoft Research† and University of Washington*

We want to understand the demands of data center applications and explore ways to augment their networks. By experimenting with four DC traffic traces, we find that real workloads have few hotspots. This indicates that a small set of links could be used to alleviate hotspots. When the traffic matrix is sparse (i.e., only a few ToR pairs are involved), the network may not be needed. Instead, performance may be improved by adding capacity to a small set of links. We propose establishing one additional flyway, in the form of an optical link, to provide additional bandwidth at hotspots. This would dramatically reduce the number of oversubscribed links, which represent a point of failure in the network.

Our scheme additive. We first measure 60 GHz propagation, link stability, and interference and link reliability. Using directional antennas, many complications are avoided. For example, 60 GHz devices with directional antennas can be deployed more easily than multi-Gigabit serial interfaces. Directional antennas can be used to align with intended senders and receivers, leading to lower interference and higher link reliability. For a given DC layout, the average size of the maximal independent set is approximately 80.

Second, the radio environment is largely static since people walk between racks much less often than in a home environment. This means that the links can form well in a DC environment. This primarily means that the links can form without the need for an external infrastructure to flood control messages electronically, i.e., with no moving parts. MEMS switches, which are capable of switching extremely fast, are a promising technology. For example, a typical DC rack has O(100) servers spread across 5 racks, running car simulation software. In most of the datasets, the servers were in racks underneath the ToR switch, and the cluster was exchanging about O(100) users every few minutes. Some de-
The rising tide of data-intensive, massive scale cluster computing is motivating a fresh look at the design of data center networks. For many years, optical circuit switching has led electrical packet switching in high bandwidth transmission. A single optical fiber can accommodate many times the bandwidth of an electrical fiber. This fact motivates us to explore how optical switching technology can be adapted from earlier solutions from the telecom and supercomputing worlds into this traditionally packet-switched environment to create a hybrid architecture.

Data-intensive applications that operate on large volumes of data to destinations in a fine-grained manner (e.g., 1, 2, 3, 2, 3, 1, 2, 3, 1) can contribute to a non-uniform traffic matrix. Second, operations such as extra batching, to allow them to do so. For the circuits to provide benefits, the traffic must be "pair-wise." While one network is being configured, the other can be used for traffic. This requires switch modification, and most existing routing protocols implement support for this in the form of traffic demultiplexing (or circuit rerouting) to match different racks at a later time; as noted earlier, this reconfiguration takes a few milliseconds, during which time the fast circuit reconfiguration to match these demands.

To ensure that latency sensitive applications can be served, the responsible ToR switch modification, and most existing routing protocols implement support for this in the form of traffic demultiplexing (or circuit rerouting) to match different racks at a later time; as noted earlier, this reconfiguration takes a few milliseconds, during which time the fast circuit reconfiguration to match these demands.

We isolate the two networks and to de-multiplex traffic at either the ToR switches by borrowing from Hedera [22], in contrast, estimates traffic demands at various levels. The bulk of inter-processor communication was bounded in degree. For the circuits to provide benefits, the traffic must be "pair-wise." While one network is being configured, the other can be used for traffic. This requires switch modification, and most existing routing protocols implement support for this in the form of traffic demultiplexing (or circuit rerouting) to match different racks at a later time; as noted earlier, this reconfiguration takes a few milliseconds, during which time the fast circuit reconfiguration to match these demands.

Traffic demultiplexing, if necessary, can be similarly accomplished in the switch of one of the two ToR switches. We discuss our particular design choices in Section 3, which are the result of allowing each to be used concurrently. The major design choice involves the traffic demands. Applications have the most accurate information about their demands, but this design requires modifying applications. As we discuss in Section 4, our c-Through design may need to implement additional mechanisms, such as extra batching, to allow them to do so.

For many years, optical circuit switching has led electrical packet switching in high bandwidth transmission. A single optical fiber can accommodate many times the bandwidth of an electrical fiber. This fact motivates us to explore how optical switching technology can be adapted from earlier solutions from the telecom and supercomputing worlds into this traditionally packet-switched environment to create a hybrid architecture.

### System Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Electrical</th>
<th>Optical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic bandwidth</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Latency</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Energy efficiency</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Scalability</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Cost</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

### Hybrid Architecture: Hybrid PaC (HyPaC)

HyPaC combines the best of both worlds, exploiting the differing characteristics of optics and electrical switchings transparently to hosts and applications. Its drawback is that it requires switch modification, and most existing routing protocols implement support for this in the form of traffic demultiplexing (or circuit rerouting) to match different racks at a later time; as noted earlier, this reconfiguration takes a few milliseconds, during which time the fast circuit reconfiguration to match these demands.

For the circuits to provide benefits, the traffic must be "pair-wise." While one network is being configured, the other can be used for traffic. This requires switch modification, and most existing routing protocols implement support for this in the form of traffic demultiplexing (or circuit rerouting) to match different racks at a later time; as noted earlier, this reconfiguration takes a few milliseconds, during which time the fast circuit reconfiguration to match these demands.

We choose for c-Through is to buffer additional data in TCP socket ends. Such buffering could also be accomplished in the application itself. In response to this now-well-known limitation, the network—Hybrid Packet and Circuit—asking both packet and electrical switching: optics provides higher bandwidth, but is transparent to applications and does not require switch changes. Packet switching networks, Circuit switching networks, data plane, control plane, and electrical switching: optics provides higher bandwidth, but cannot provide full bisection bandwidth at the packet granularity.

### Design Choices and Trade-offs

#### 1. Estimating Cross-Rack Traffic Demands

- **Estimating Cross-Rack Traffic Demands**
  - **Packet Switching:** Estimates traffic demands at various levels. The bulk of inter-processor communication was bounded in degree.
  - **Electrical Switching:** Provides high bandwidth but cannot provide full bisection bandwidth at the packet granularity.

#### 2. Circuit Reconfiguration

- **Circuit Reconfiguration:** Takes a few milliseconds, during which time the fast circuit reconfiguration to match these demands.

#### 3. Traffic Demultiplexing

- **Traffic Demultiplexing:** If necessary, can be similarly accomplished in the switch of one of the two ToR switches. We discuss our particular design choices in Section 3, which are the result of allowing each to be used concurrently. The major design choice involves the traffic demands. Applications have the most accurate information about their demands, but this design requires modifying applications. As we discuss in Section 4, our c-Through design may need to implement additional mechanisms, such as extra batching, to allow them to do so.

#### 4. Optimal Design Choices

- **Optimal Design Choices:** Hybrid PaC (HyPaC) combines the best of both worlds, exploiting the differing characteristics of optics and electrical switchings transparently to hosts and applications. Its drawback is that it requires switch modification, and most existing routing protocols implement support for this in the form of traffic demultiplexing (or circuit rerouting) to match different racks at a later time; as noted earlier, this reconfiguration takes a few milliseconds, during which time the fast circuit reconfiguration to match these demands.

### Experimental Experience

The experimental experience provides useful insights on the applicability of the hybrid architecture across a range of deployment scenarios. This is because the traffic demands between them and lower demand to others.

### Conclusion

The hybrid architecture of the center network architecture (or HyPaC for short) which augments packet switching networks, Circuit switching networks, data plane, control plane, and electrical switching: optics provides higher bandwidth, but cannot provide full bisection bandwidth at the packet granularity. This fact motivates us to explore how optical switching technology can be adapted from earlier solutions from the telecom and supercomputing worlds into this traditionally packet-switched environment to create a hybrid architecture.

---

**Table 1: Fundamental Requirements of HyPaC Architecture.**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Electrical</th>
<th>Optical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic bandwidth</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Latency</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Energy efficiency</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Scalability</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Cost</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>
ProjecToR: Agile Reconfigurable Data Center Interconnect

Monia Ghorbadi  Ratul Mahajan  Amar Phanishayee
Nikhil Devanur  Janardhan Kulkarni  Gireeja Ranade
Pierre-Alexandre Blanche†  Houman Rastegarfar†  Madeleine Glick†  Daniel Kilper†
Microsoft Research  †University of Arizona

Mirror assembly

Received beam

Diffracted beam

Towards destination

Photodetectors

DMDs

Lasers

Top-of-Rack

Array of Micromirrors

Received beam

Diffracted beam

Towards destination

Photodetectors

DMDs

Lasers

Top-of-Rack

Array of Micromirrors
Impact so far ...

Beyond fat-trees without antennae, mirrors, and disco-balls

Simon Kassing  
ETH Zürich  
simon.kassing@inf.ethz.ch

Asaf Valadarsky  
Hebrew University of Jerusalem  
asaf.valadarsky@mail.huji.ac.il

Gal Shahaf  
Hebrew University of Jerusalem  
gal.shahaf@mail.huji.ac.il

Michael Schapira  
Hebrew University of Jerusalem  
schapiram@huji.ac.il

Ankit Singla  
ETH Zürich  
ankit.singla@inf.ethz.ch

Addressing some practical issues, e.g., routing without structure
Next lecture ...
How do we do routing?
Useful tools for routing

1. Exploiting structure in the network

- Root switch A
- Root switch B
- Server X
- Server Y
Useful tools for routing

10.2.*

10.2.4.*

10.2.3.*
Useful tools for routing

2. Incorporate routing information in node names

Routing: an information encoding problem

1. Structure the network itself
2. Incorporate routing information in node names
3. Store information in routing tables
4. Store maps of the network at some or all devices
Forwarding: using encoded routing information

A Scalable, Commodity Data Center Network Architecture

Mohammad Al-Fares  Alexander Loukissas  Amin Vahdat