Geo-distributed Cloud Data Centers

Dr. Rajiv Misra

Associate Professor

Dept. of Computer Science & Engg.

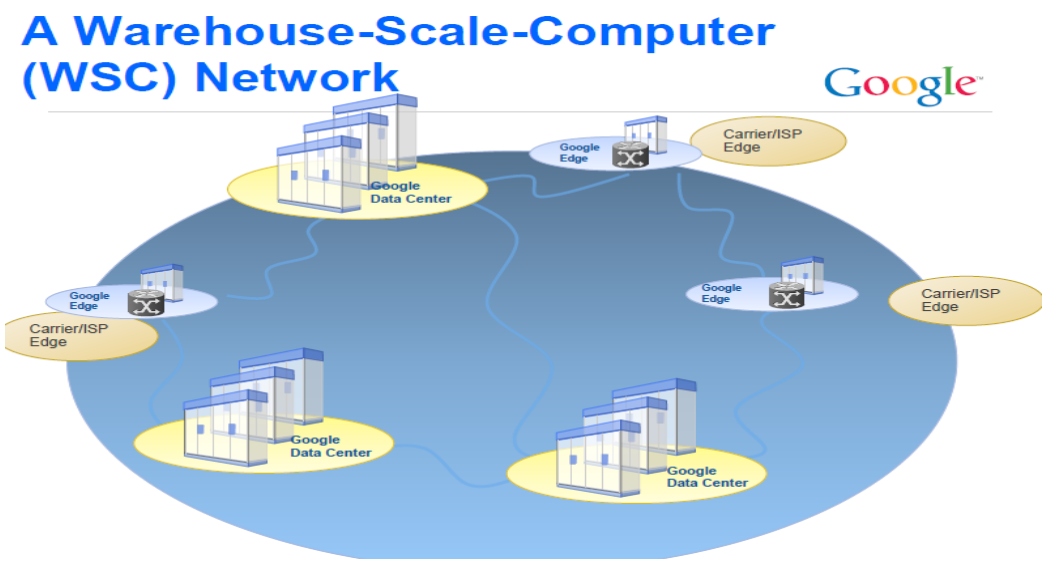
Indian Institute of Technology Patna

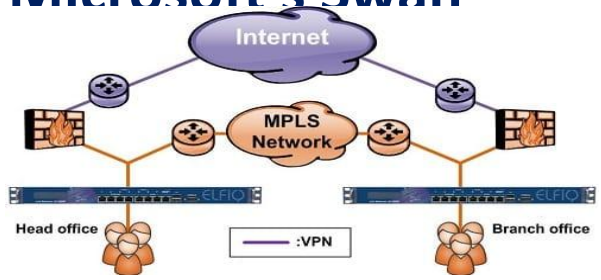
rajivm@iitp.ac.in

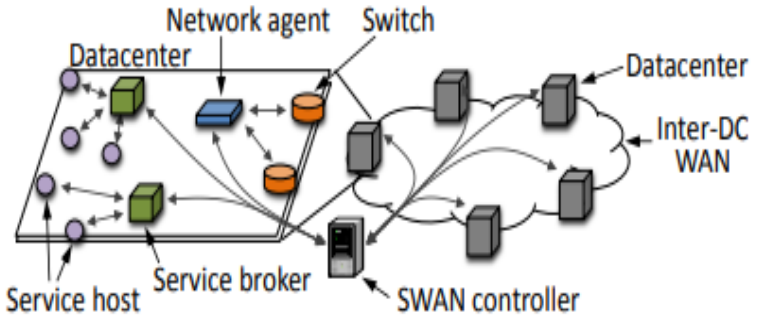
Preface

Content of this Lecture:

- In this lecture, we will study the **Geo-distributed cloud data centers**, interaction of data centers with users and with other data centers.
- We will also describe the data center interconnection techniques such as **(i) Traditional schemes such as MPLS, (ii) Cutting edge such as Google's B4 and (iii) Microsoft's Swan**

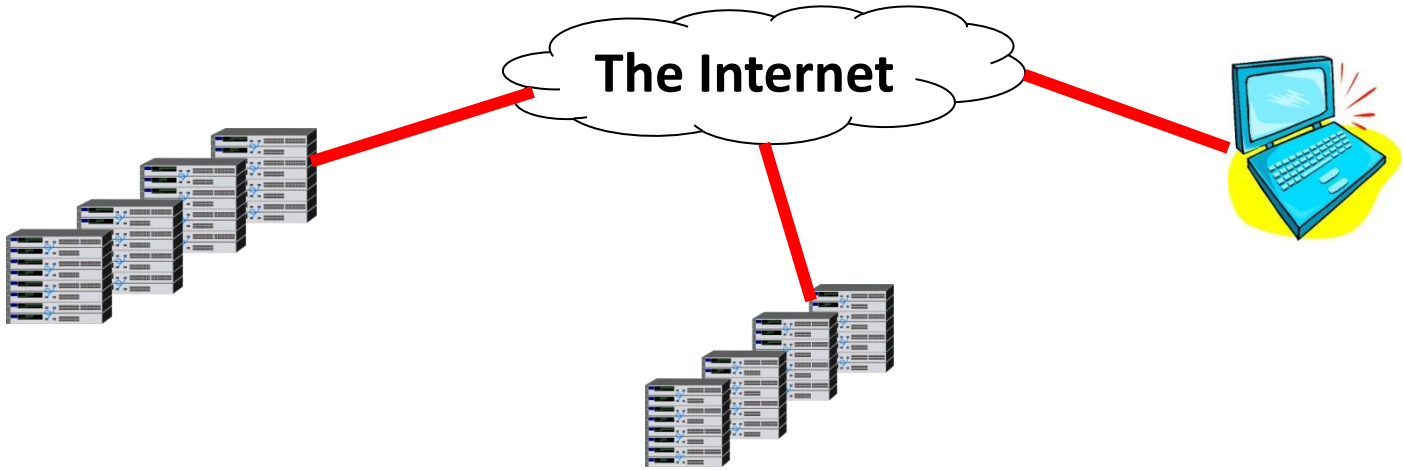






Inter-Data Center Networking: The Problem

- Today, the virtual use of any popular web application means to communicate with a server in a data center.
  - However the connectivity for this service depends on the internet
  - Internet becomes crucial in application service's performance
- Data centers also communicate with each other over the internet. **Example: replicating client data across multiple data centers.**
- In cloud scenario, the wide area connectivity or the internet is as crucial as the data center infrastructure.



Why Multiple Data centers ?

- Why does a provider like Google need such an extensive infrastructure with so many locations across a wide expanse of the globe?
- **Better data availability:** If one of the facilities goes down, due to a natural disaster, you could still have the data be available at some other location if it is replicated.
- **Load balancing:** Multiple facilities can spread incoming and outgoing traffic over the internet across a wider set of providers, over a wider geographic regions.
- **Latency:** If present in multiple paths of the globe then can reach clients in different locations at smaller distances, thus reduces latency.
- **Local data laws:** Several authority might require that companies store data from that country in that jurisdiction itself.
- **Hybrid public-private operation:** Can handle the average demand for service from the private infrastructure, and then offload peak demand to the public cloud.

Significant Inter-data center traffic

This paper was presented as part of the main technical program at IEEE INFOCOM 2011

A First Look at Inter-Data Center Traffic Characteristics via Yahoo! Datasets

Yingying Chen[1], Sourabh Jain[1], Vijay Kumar Adhikari[1], Zhi-Li Zhang[1], and Kuai Xu[2]

[1]University of Minnesota-Twin Cities
[2]Arizona State University

- **Study from five Yahoo data centers from 2011.**
- The study is based on **anonymized traces from border routers** that connect to these datacenters.

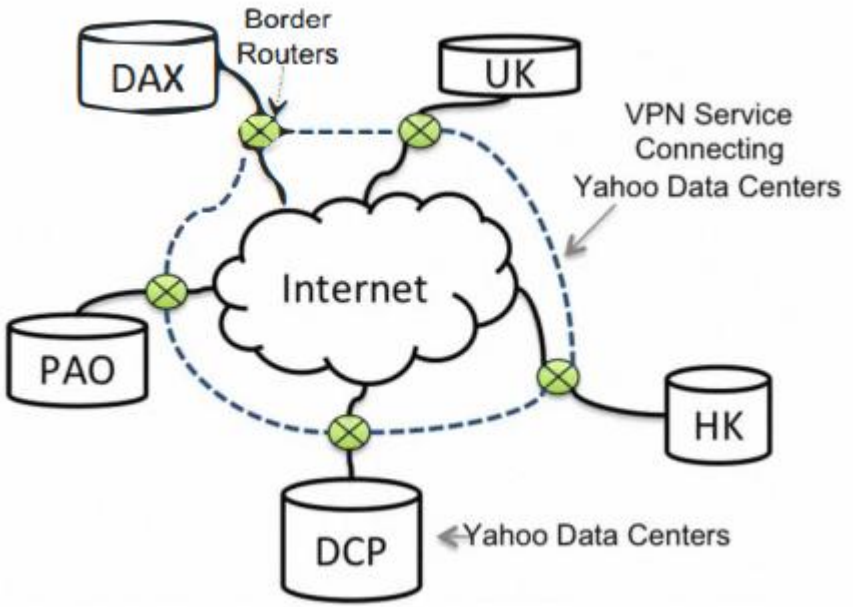
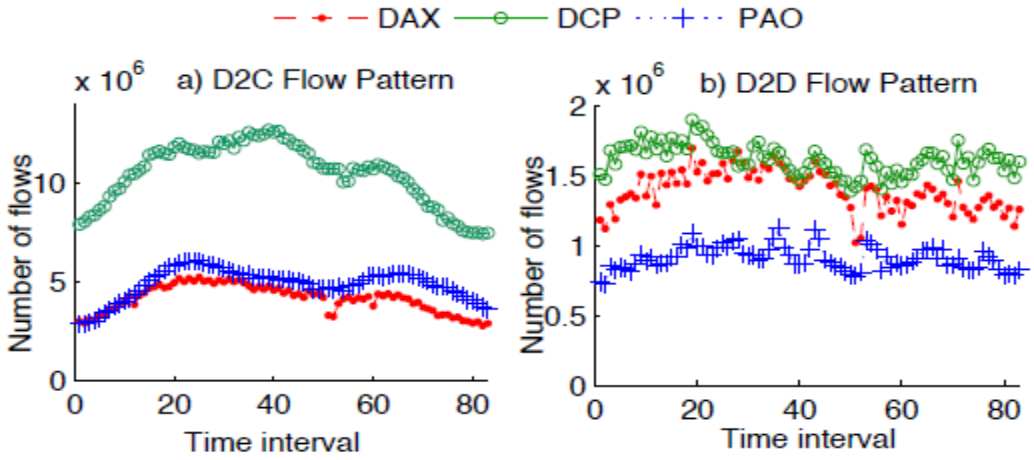


Fig.: Overview of five major Yahoo! data centers and their network connectivity.

Significant Inter-data center traffic

- Here two plots showing the **number of flows between clients and the data centers.**
- On the right, between the data centers themselves. The three lines on each plot are for three different data center locations. Notice that the y-axis on these two plots are different. In terms of number of flows, the traffic between data centers is 10% to 20% of the traffic from data centers to clients.
- Flows between data centers could be very long lived and carry more bytes than those between clients and data centers.



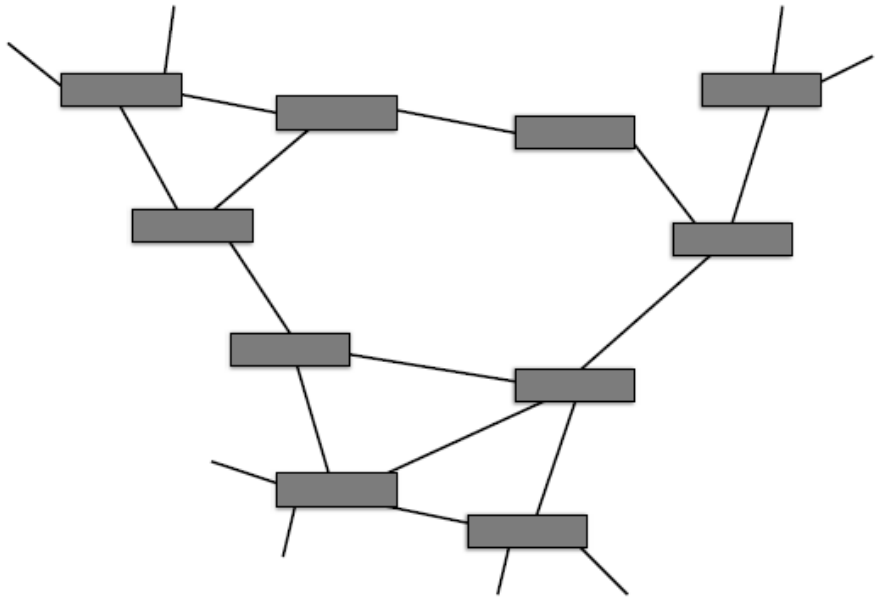
Why are these networks different ?

- Persistent **dedicated,100s of Gbps** connectivity between a (small) set of end-points

- Microsoft: “expensive resource, with amortized annual cost of 100s of millions of dollars”
- **[Achieving High Utilization with Software-Driven WAN, Hong et al., ACM SIGCOMM’13]**

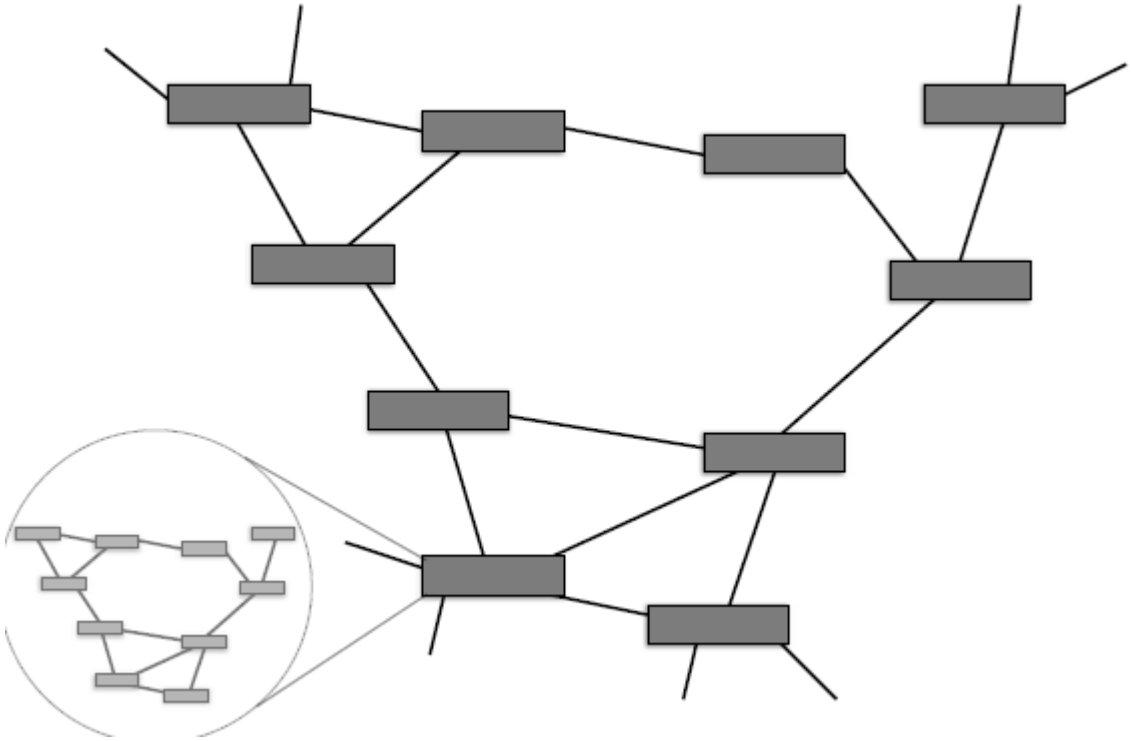
(i) MPLS: Traditional WAN approach

- The traditional approach to traffic engineering in such networks is to use **MPLS (Multiprotocol Label Switching)**
- Network with several different sites spread over defined area, connected to each other perhaps over long distance fiber links.



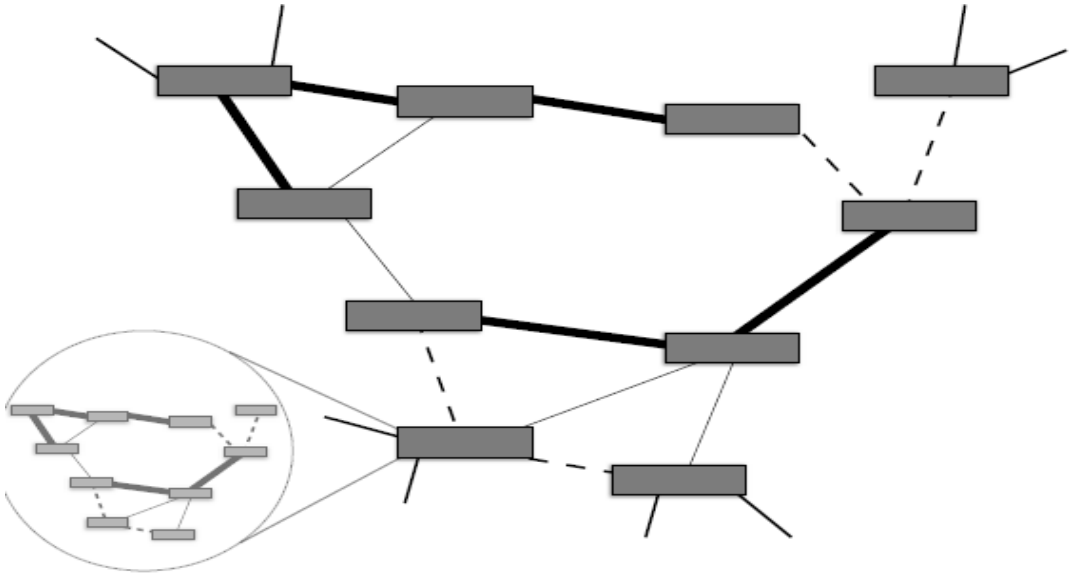
1. Link-state protocol (OSPF / IS-IS)

- Use **link-state protocol (OSPF or IS-IS)** to flood information about the network's topology to all nodes.
- So at the end of such a protocol, every node has a map of the network.



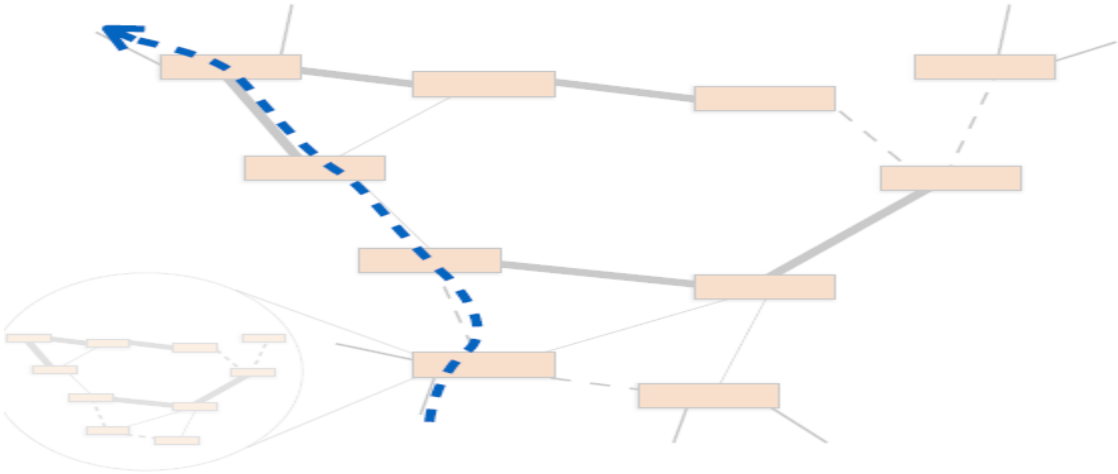
2. Flood available bandwidth information

- For traffic engineering, also **spread, information about the bandwidth usage** on these different links in the network.
- Given that there's already traffic flowing in this network, some links will have spare capacity and some won’t.
- Both IS-IS and OSPF have extensions that allow the flooding of available bandwidth information together with their protocol messages.



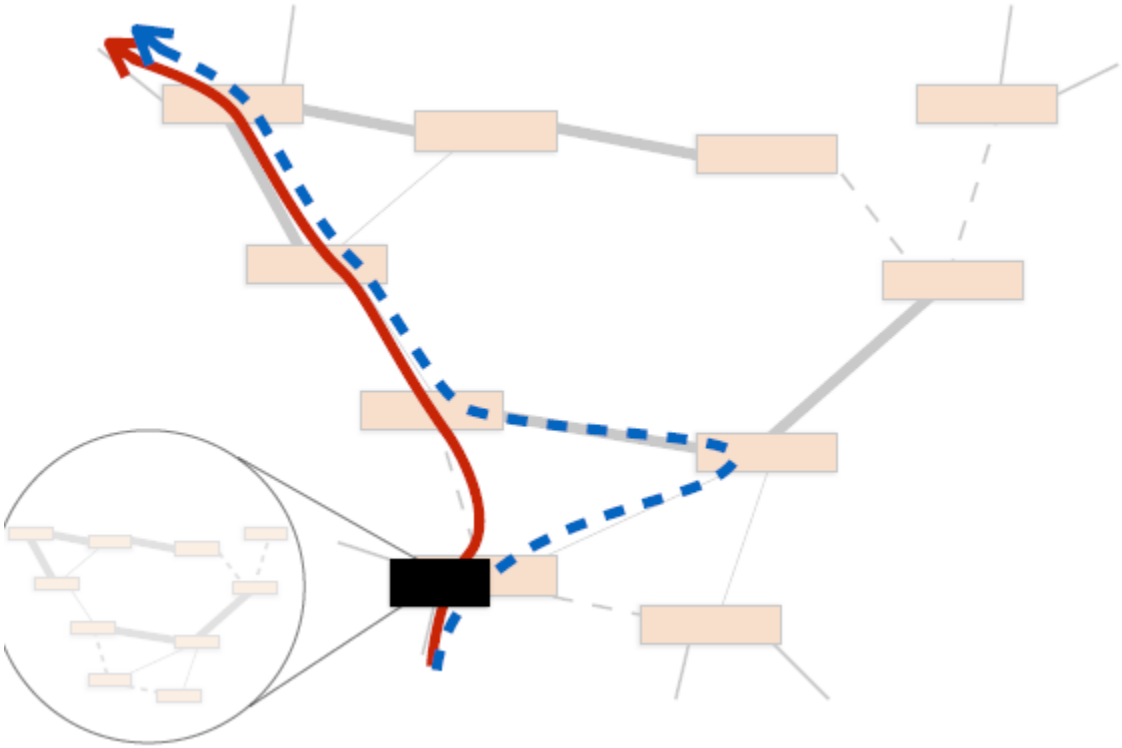
3. Fulfill tunnel provisioning requests

- Knowing the set of the network, when the router receives a new flow set of requests, it'll set up a tunnel along the shortest path on which enough capacity's available. It sends protocol messages to routers on the path setting up this tunnel.
- Further, **MPLS also supports the notion of priorities.** Thereby if a higher priority flow comes in with the request for a path, lower priority flows might be displaced. These flows might then use higher latency or higher cross paths through the network.



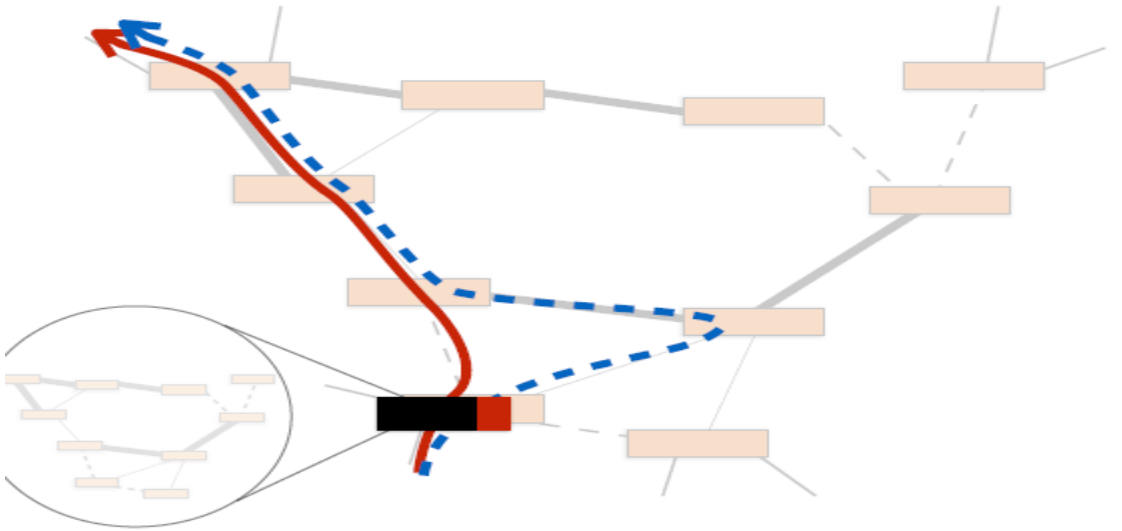
4. Update network state, flood information

- After a flow is assigned a terminal, the routers also update the network state.



4. Update network state, flood information

- When a data packet comes into the ingress router, the router looks at the packet's header and decides what label, that is what tunnel this packet belongs to. Then it encapsulates this packet with that tunnel's label and sends it along the tunnel.
- The egress router then decapsulates the packet, looks at the packet header again and sends it to the destination. **In this scheme, only the ingress and egress routers read the packet header.** Every other router on the path just looks at the assigned label.



Simple forwarding along the path

- Making forwarding along the path very simple. This is the reason the protocol is called **Multi-Protocol Label Switching (MPLS).**

- Also, MPLS can run over several different protocols, as long as the ingress and egress routers understand that protocol, and can map onto labels, that's why the name is: **multi-protocol label switching.**

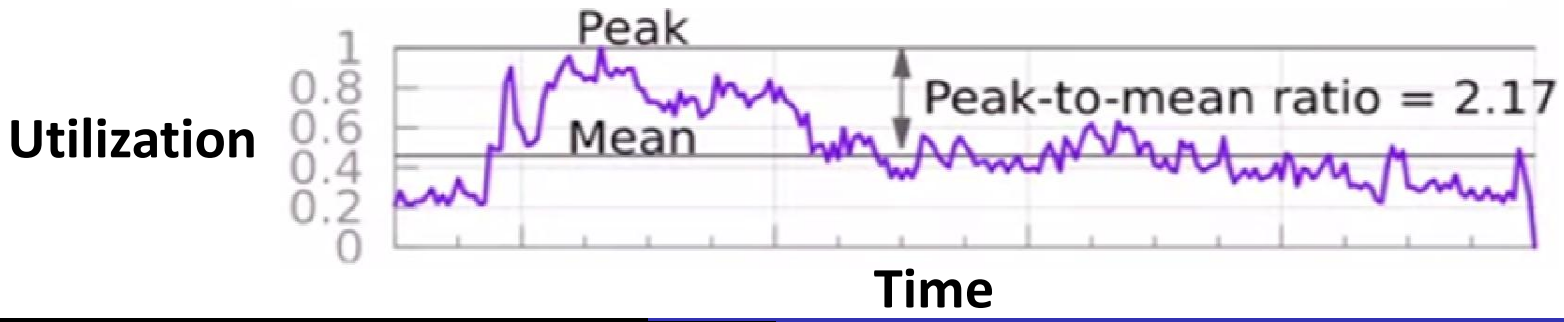
Problem 1: Inefficiency

- First problem is **inefficiency in terms of usage of the expensive bandwidth.** Typically, these networks would be provisioned for the peak traffic.
- As this image shows here, if you have the traffic over time, the y-axis utilization, provision the network for peak traffic. Now, the mean usage of the network might be very small. In this example, it's **2.17 times smaller than the peak.**

ACM SIGCOMM, 2013

Achieving High Utilization with Software-Driven WAN

Chi-Yao Hong (UIUC) Srikanth Kandula Ratul Mahajan Ming Zhang
Vijay Gill Mohan Nanduri Roger Wattenhofer (ETH)
Microsoft



Problem 1: Inefficiency

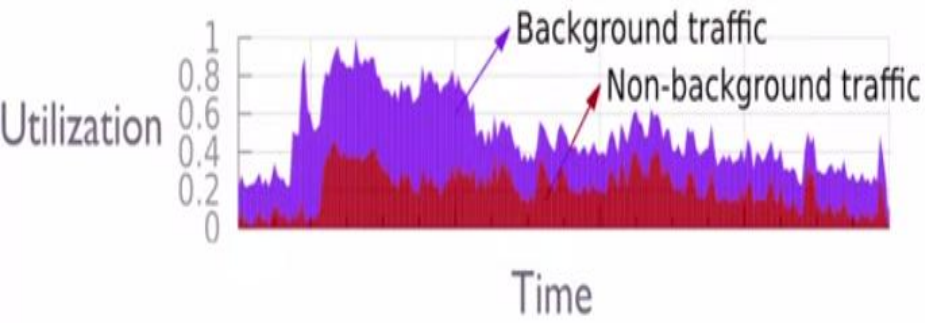
- Most of this traffic is actually **background traffic, with some latency sensitive traffic** as well.
- So you can provision for the peak of the latency sensitive traffic, and then fill the gaps with the background which is not latency sensitive.

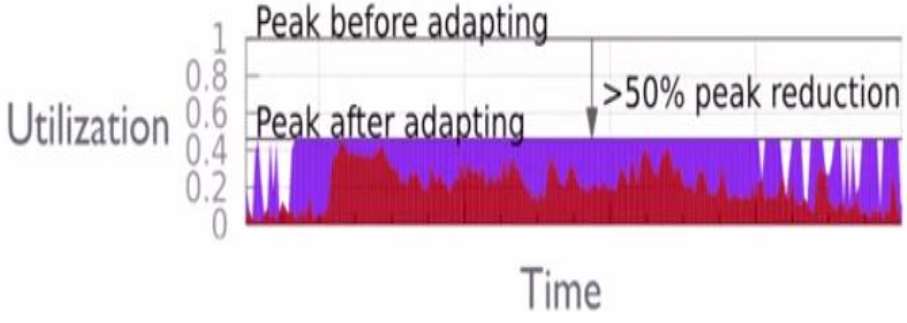
ACM SIGCOMM, 2013

Achieving High Utilization with Software-Driven WAN

Chi-Yao Hong (UIUC) Srikanth Kandula Ratul Mahajan Ming Zhang
Vijay Gill Mohan Nanduri Roger Wattenhofer (ETH)

Microsoft





Problem 1: Inefficiency

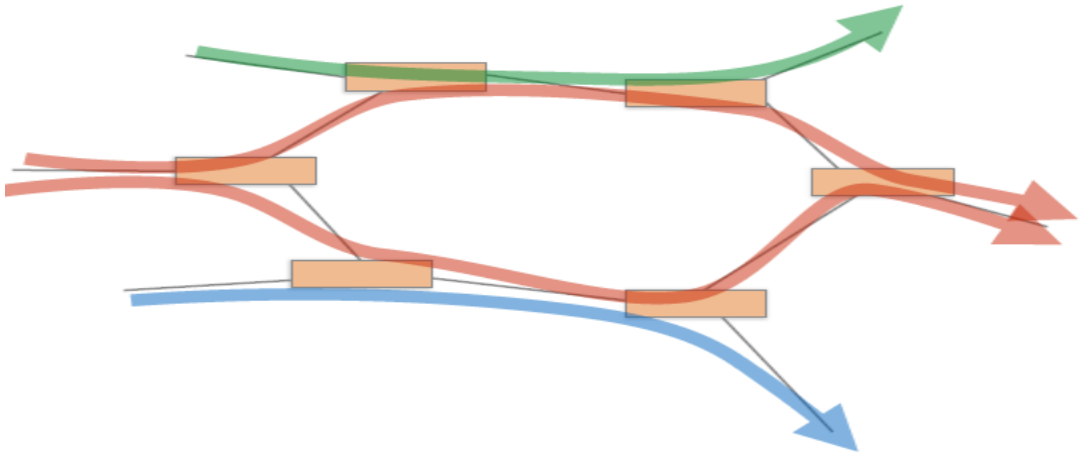
- So unless you differentiate traffic by service, you cannot do such an optimization. This is not easy to do with the MPLS approach because **it does not have a global view of what services are running in the network,** what parts of the network they are using and such.

- Also, a related point is that regardless of whether they are multiple services or not, MPLS, the routers make local greedy choices about scheduling flows. So **traffic engineering is sub optimal.**

- For these reasons, such networks typically run around 30% utilization to have enough headroom for these inefficiencies, and this is expensive.

Problem 2: Inflexible sharing

- Another big problem with the MPLS approach, is that it only provides link level fairness. So at any link, the flows can share capacity fairly. **But, this does not mean network wide fairness.**
- For example, we have the green flow sharing capacity across that length with a red flow. The blue flow also shares capacity with the red flow. But, the blue and green flows both get capacity half the red flow, because the red flow uses multiple paths. So we have link level fairness, but we do not have the network wide fairness.
- The network wide fairness is hard to achieve, **unless you have a global view of the network.**



Cutting-edge WAN Traffic Engineering (TE)

Google's B4

ACM SIGCOMM, 2013

B4: Experience with a Globally-Deployed Software Defined WAN

Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh,
Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla,
Urs Hölzle, Stephen Stuart and Amin Vahdat
Google, Inc.
b4-sigcomm@google.com

Microsoft's Swan

ACM SIGCOMM, 2013

Achieving High Utilization with Software-Driven WAN

Chi-Yao Hong (UIUC) Srikanth Kandula Ratul Mahajan Ming Zhang
Vijay Gill Mohan Nanduri Roger Wattenhofer (ETH)
Microsoft

1. Leverage service diversity: some tolerate delay

- **To get very high bandwidth utilization in the WAN** because of natural fluctuations over time. So, to leverage diversity in the services some services need a certain amount of bandwidth at a certain moment of time and they're inflexible and some other services can use to kind of fill in whatever room is left over.

- **For example, latencies instead of queries.**

2. Centralized TE using SDN, OpenFlow

- Software define networking approach gather information about the state of the network.
- Make a centralize decision about the flow of traffic and then push those decisions down to lower levels to actually implement them.
- But bringing all that information together in one place is a relatively complex decision.

3. Exact linear programming is too slow

- Traditionally, with a optimization technique like linear programming, which is a way to take a set of constraints on required amounts of data flow over parts of a network and come up with an optimal solution. But to apply it to the situation where we need to **make relatively quick decisions.**

- Part of the complexity comes from the multitude of services, the different priorities. If we have just one service, we could run it in flow algorithm, and that would be much faster.

- So it require something faster if it's not guaranteed to be exactly optimal.

4. Dynamic reallocation of bandwidth

- **The demands on the network change over time.** So to make continual decisions about what traffic is highest priority to move across which links at a given moment is a challenge with linear programming to make quick decisions.

- So these are online algorithms. But they're not online in the same way as things inside the data center might be.

- For example, Google runs its traffic engineering 500 or so times a day. So, it's not as fine grained as things we might need inside a data center. Traffic between these facilities is relatively stable it seems.

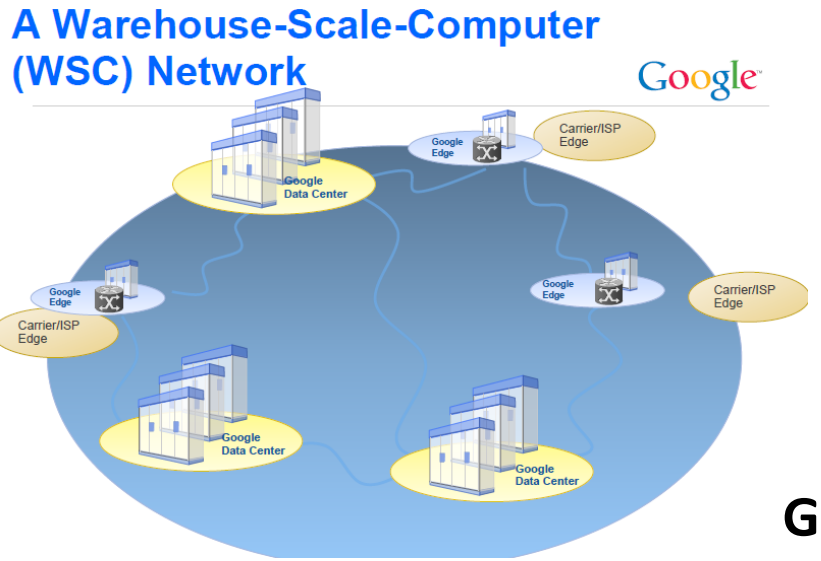
5. Edge rate limiting

- The commonality in the architecture is to implement an **enforcement of the flow rates.**

- So when the traffic enters the network and will do that at the edge, rather than at every hop along the path.

(ii) B4: Google's WAN (2011)

- **Google's B4 was the first highly visible SDN success.** It is a private WAN connecting Google's data centers across the planet. It has 12 locations spread across three different continents.
- It has a number of unique characteristics: **i) massive bandwidth requirements deployed to a modest number of sites, ii) elastic traffic demand that seeks to maximize average bandwidth, and iii) full control over the edge servers and network,** which enables rate limiting and demand measurement at the edge



Google's B4 worldwide deployment (2011)

What happens at one site, one data center

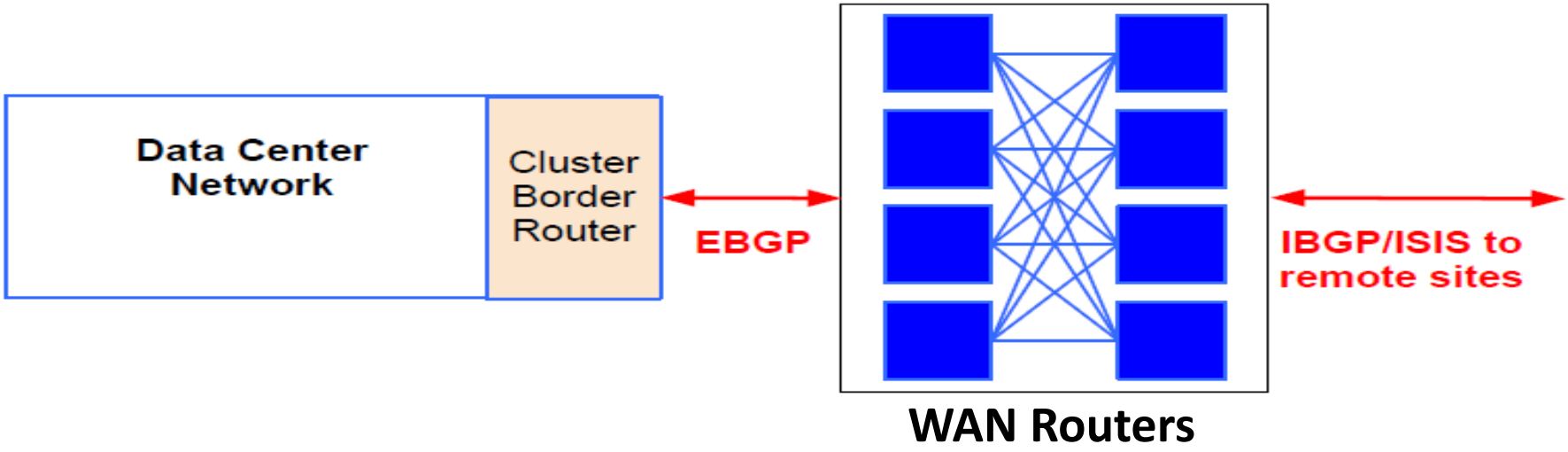
Google's B4: view at one site

- **Inside one site, it has the data center network, of cluster & border routers**.

- In the traditional setting, these are connected using eBGP to run routers, which would then also interface using iBGP or IS-IS to the other data center sites.

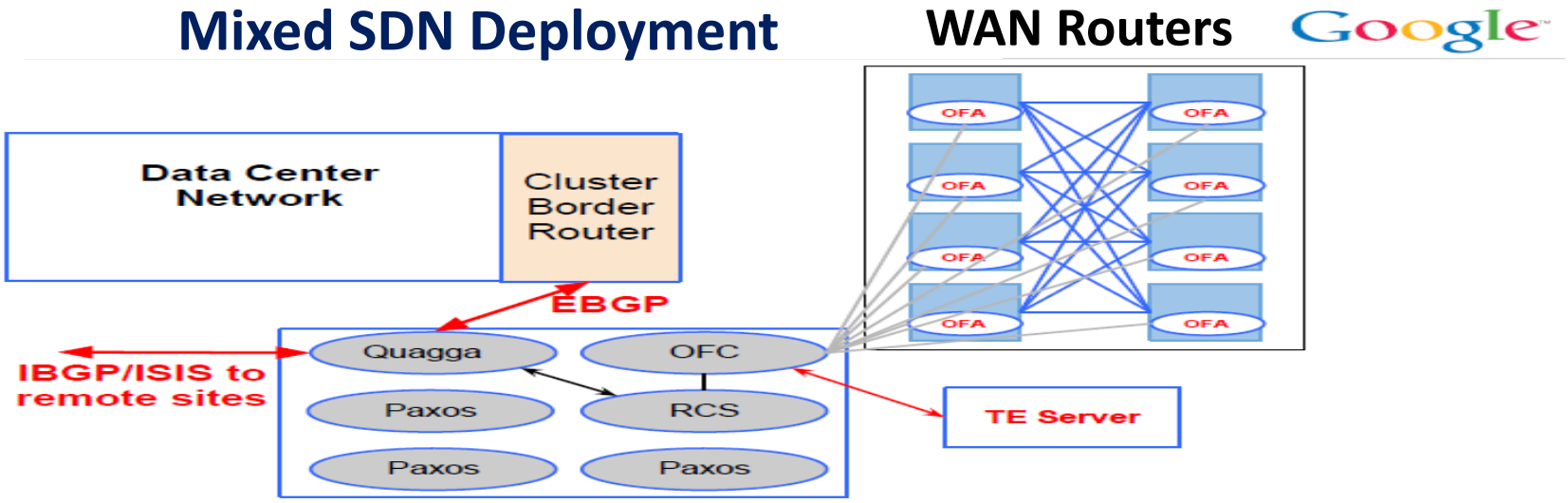
Mixed SDN Deployment

Google



Google's B4: view at one site

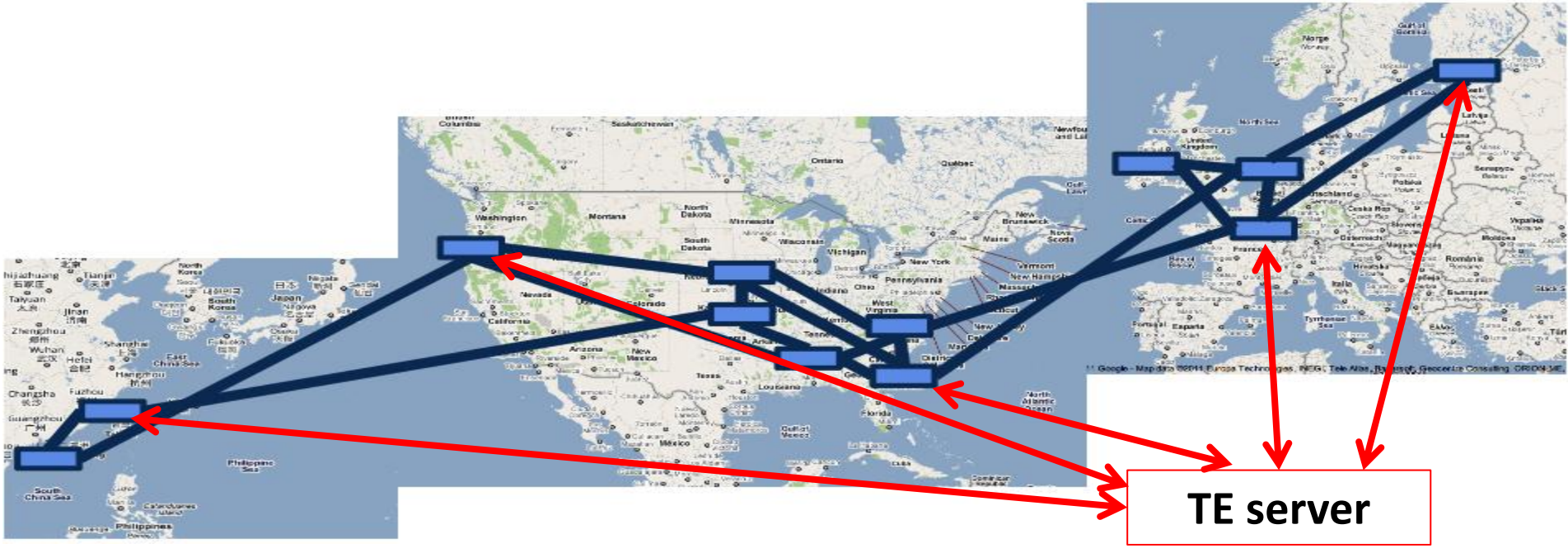
- For finer control over routing, this will be moved to a software router, i.e. **Quagga software switch that run it on a server.**
- The interface with open flow to set up routing rules on those routers. So, Quagga will run the routing protocols between the cluster border routers, and also the other sites and then, OpenFlow uses the routing information from Quagga and sets up forwarding rules in the WAN routers.
- Now we have software control and the traffic engineering (TE) server, which manages what rules exactly are installed.



Google's B4: Traffic Engineering

- The TE server Collects the topology information, the available bandwidth information and has the last information about flow demands between different sites.

- The TE server pushes out the flow allocations to the different data centers. At the data centers the multiple controllers then enforce those flow allocations on the centers.



Google's B4: Design Choices

- **BGP routing as "big red switch":**
- They also keep available the BGP forwarding state because each of their switches allows them to have multiple forwarding tables at the same time they can afford to have BGP forwarding tables.
- Also, in addition to the traffic engineering, if the traffic engineering scheme does not work. They can discard those routing tables and use the BGP forwarding state instead. **So the BGP routing tables serve as a big red switch.**

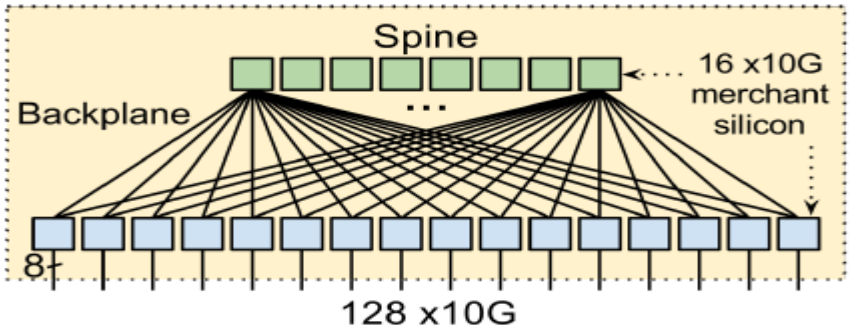
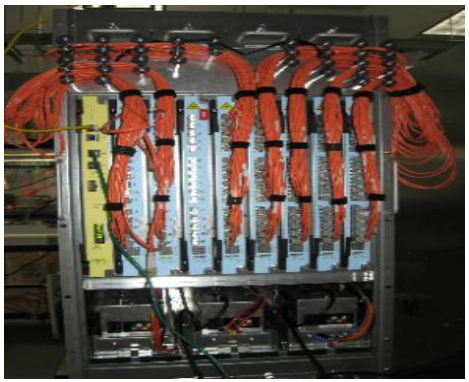


Figure: A custom-built switch and its topology

Google's B4: Design Choices

- **Hardware: custom, cheap, simple:**

- In 2011, there were not many software defined networking switches on the market. **So Google built their own using cheap commodity equipment.**



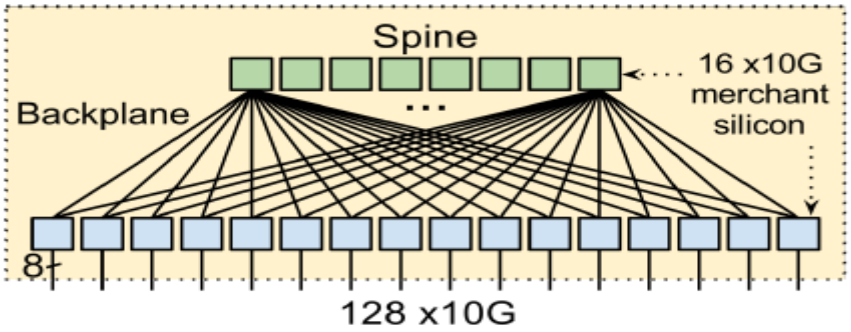


Figure: A custom-built switch and its topology

Google's B4: Design Choices

- **Software smart with safeguard:**

- Most of these smarts then reside in the software.
- Switches are simple in the software, which is the **open flow control logic at each site replicated for fault tolerant using paxos.**
- Further, for scalability of this system they use a hierarchy of controllers. **The software solution achieves nearly 100% utilization and solves the trafficking problem in 0.3 seconds.**

Google's B4: Design Choices

- **Hierarchy of controllers:**
- At the top level we have the **global controller**, which is talking to an **SDN gateway**. The gateway can be thought of as a **super controller** that talks to the controller's at all these different data center sites.
- Each site might itself have **multiple controllers**, because of the scale of these networks. **This hierarchy of controllers simplifies things from the global perspective.**

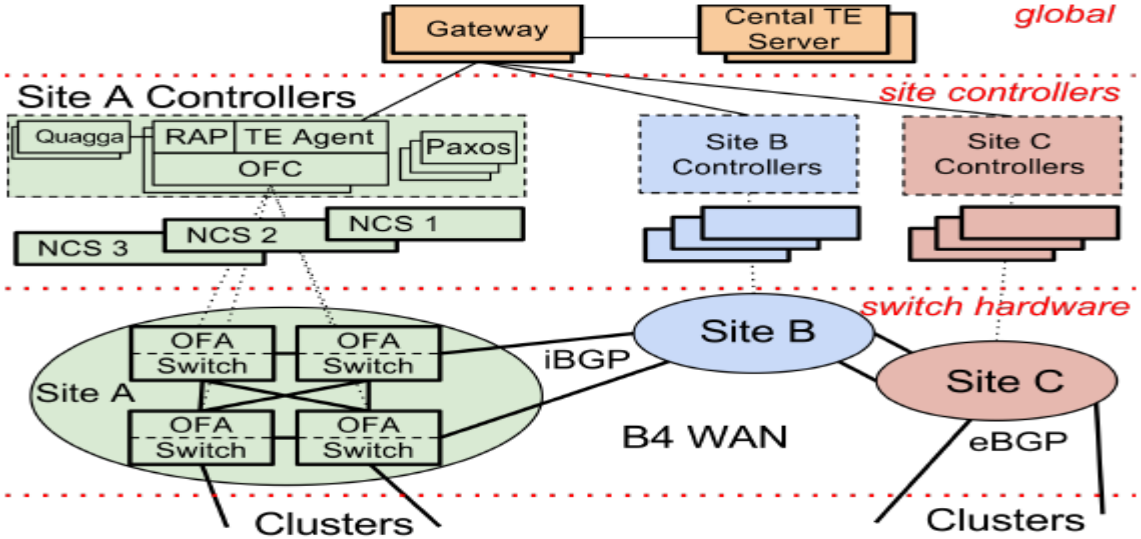


Figure: B4 architecture overview

Google's B4: Design Choices

- **Aggregation: flow groups, link groups:**

- Earlier, traffic engineering at this global scale is not at the level of mutual flows but of flow groups. That also helps scaling. Further, each pair of sites is not connected by just one link.

- These are massive capacity links that are formed from a trunk of several parallel high capacity links.

- All of these are aggregated and exposed to the traffic engineering layer as one logical link**. It is up to the individual site to the partition traffic, multiplex and demultiplex traffic across the set of links.**

(iii) Microsoft's Swan

- Microsoft has publicly disclosed the design for optimizing wide area traffic flow in their WAN.

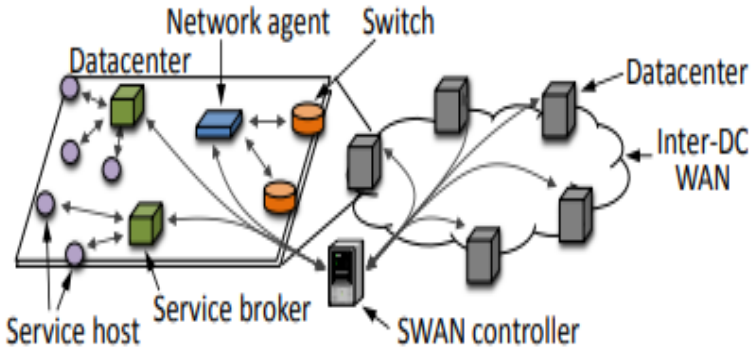
- Interesting feature in this design is the way **to make changes to the traffic flow without causing congestion.**

ACM SIGCOMM, 2013

Achieving High Utilization with Software-Driven WAN

Chi-Yao Hong (UIUC) Srikanth Kandula Ratul Mahajan Ming Zhang
Vijay Gill Mohan Nanduri Roger Wattenhofer (ETH)

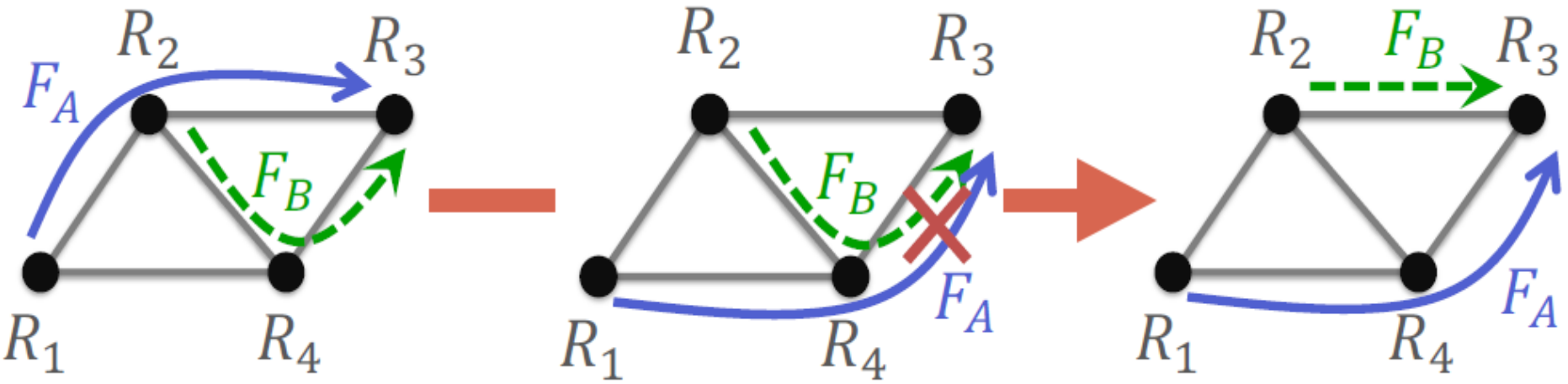
Microsoft



Architecture of Microsoft's Swan

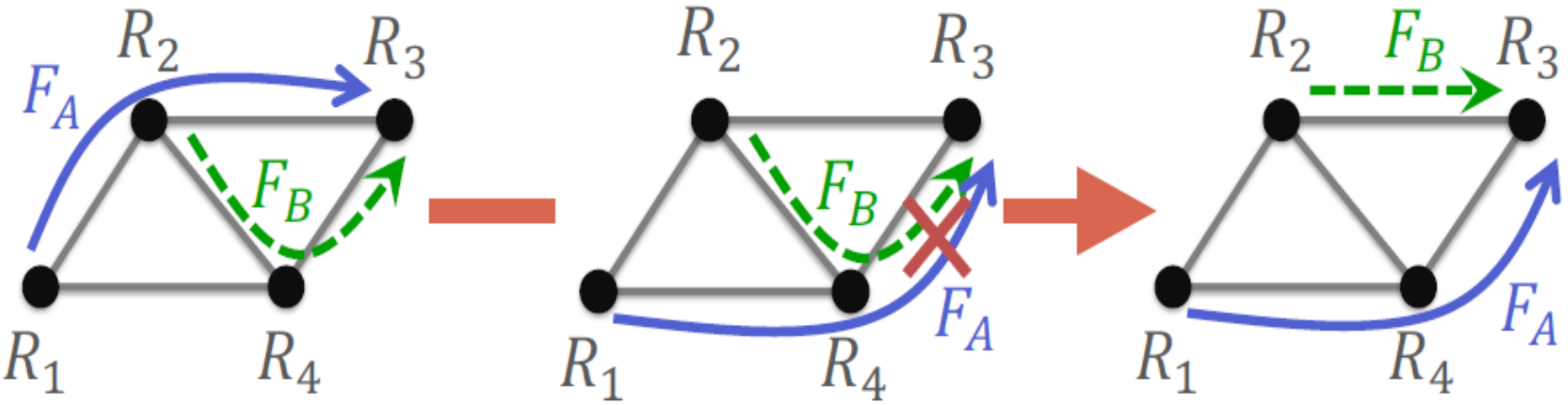
Microsoft's Swan

- The network on the left have two flows, the blue one and the green one that are each using in this simple example 100% of the bandwidth of the links that they flow across.
- To move from the left scenario to the right scenario. If we do using any naive way, because we can't control exactly when every packet will flow across every link. There's going to be some period of time where, due to timing uncertainty, we have both of the flows sharing to some extent one of the links.



Microsoft's Swan

- Can think of different ways to do this but we always going to run into some link that may get used by both lows at the same time. So, take a look at this design for SWAN there's an approach to making those updates with a certain amount of spare capacity so that congestion can be avoided.
- This approach that takes the optimization one step further **so that providing a pretty strong guarantee on lack of congestion even while the network data flow of changing.**



Conclusion

- In this lecture, we have discussed the **geo-distributed cloud data centers**, interaction of data centers with users and other data centers.

- Also discuss various data center interconnection techniques such as **(i) MPLS (ii) Google's B4 and (iii) Microsoft's Swan**