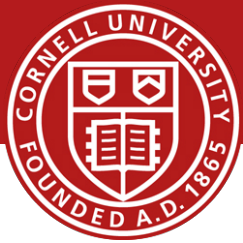


Data Center Network Topologies: FatTree

Hakim Weatherspoon

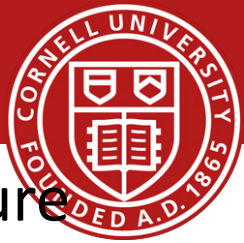
Assistant Professor, Dept of Computer Science

CS 5413: High Performance Systems and Networking

September 22, 2014

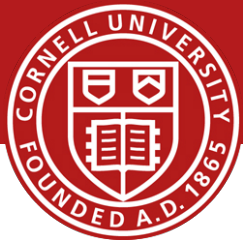
Slides used and adapted judiciously from Networking Problems in Cloud Computing EECS 395/495 at Northwestern University

Goals for Today



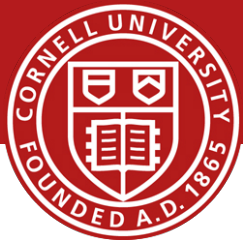
- A Scalable, Commodity Data Center Network Architecture
  - M. Al-Fares, A. Loukissas, A. Vahdat. ACM SIGCOMM Computer Communication Review (CCR), Volume 38, Issue 4 (October 2008), pages 63-74.
- Main Goal: addressing the limitations of today's data center network architecture
  - single point of failure
  - oversubscription of links higher up in the topology
    - trade-offs between cost and providing
- Key Design Considerations/Goals
  - Allows host communication at line speed
    - no matter where they are located!
  - Backwards compatible with existing infrastructure
    - no changes in application & support of layer 2 (Ethernet)
  - Cost effective
    - cheap infrastructure
    - and low power consumption & heat emission

Overview



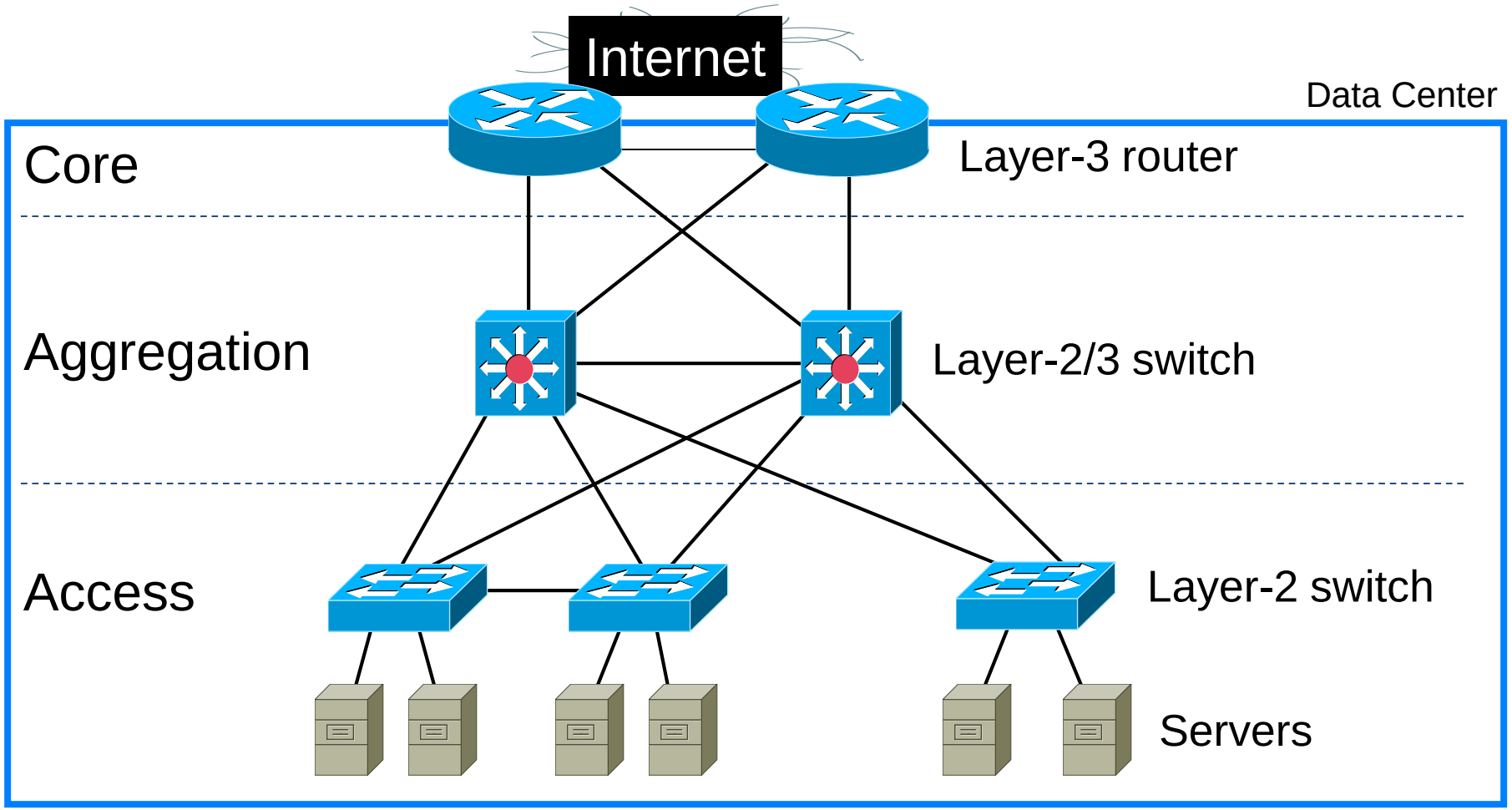
- Background of Current DCN Architectures
- Desired properties in a DC Architecture
- Fat tree based solution
- Evaluation
- Conclusion

Background

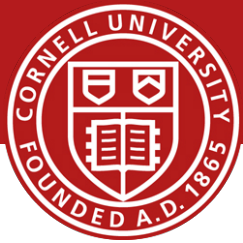


- *Topology:*
  - 2 layers: 5K to 8K hosts
  - 3 layer: >25K hosts
  - Switches:
    - Leaves: have N GigE ports (48-288) + N 10 GigE uplinks to one or more layers of network elements
    - Higher levels: N 10 GigE ports (32-128)
- *Multi-path Routing:*
  - Ex. ECMP
    - without it, the largest cluster = 1,280 nodes
    - Performs static load splitting among flows
    - Lead to oversubscription for simple comm. patterns
    - Routing table entries grows multiplicatively with number of paths, cost ++, lookup latency ++

Common Data Center Topology

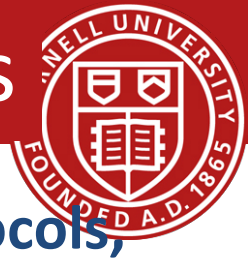


Background



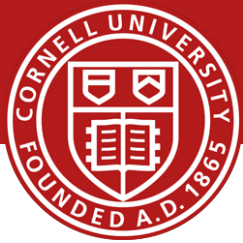
- *Oversubscription:*
  - Ratio of the worst-case achievable aggregate bandwidth among the end hosts to the total bisection bandwidth of a particular communication topology
  - Lower the total cost of the design
  - Typical designs: factor of 2:5:1 (400 Mbps)to 8:1(125 Mbps)
- *Cost:*
  - Edge: $7,000 for each 48-port GigE switch
  - Aggregation and core: $700,000 for 128-port 10GigE switches
  - Cabling costs are not considered!

Current Data Center Network Architectures



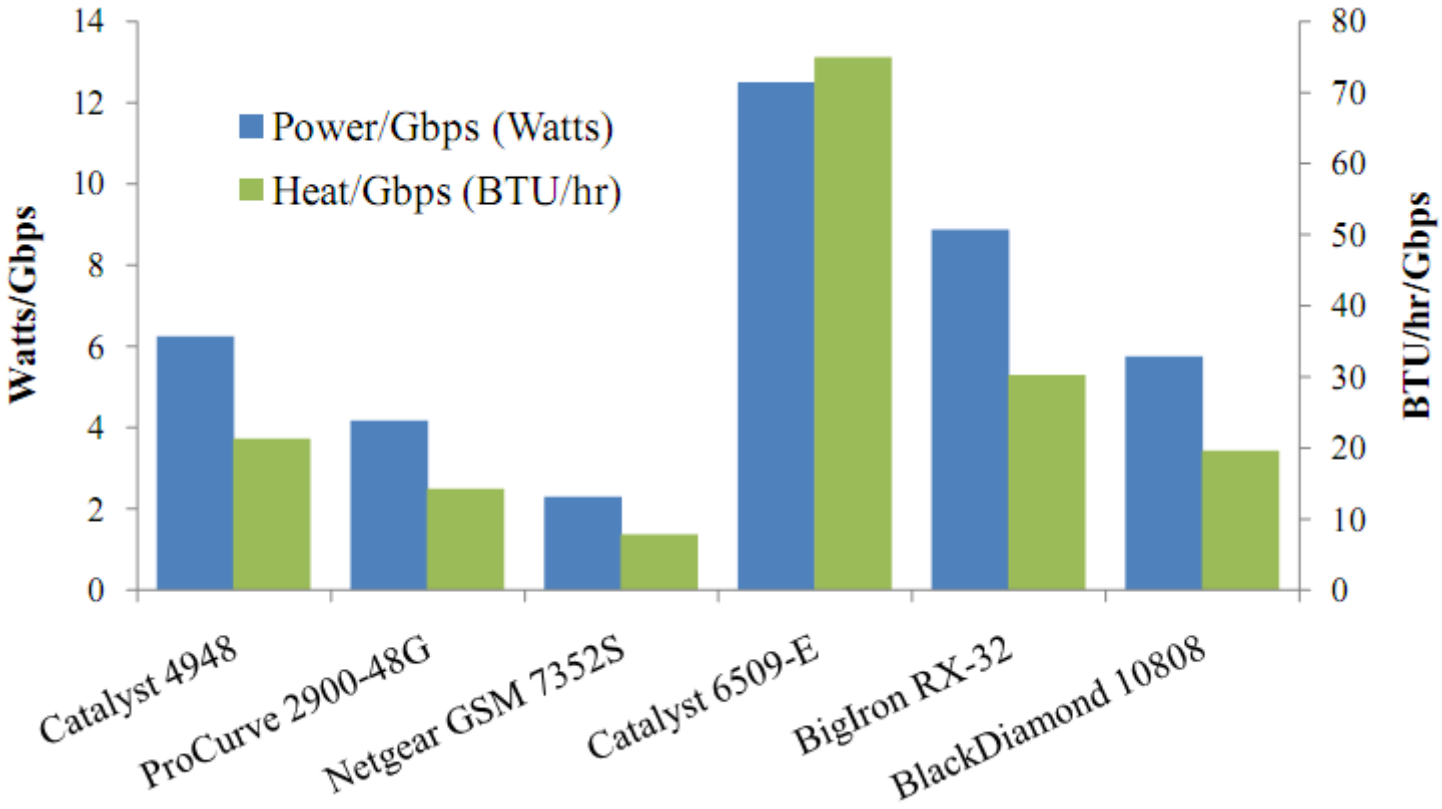
- **Leverages specialized hardware and communication protocols, such as InfiniBand, Myrinet.**
  - These solutions can scale to clusters of thousands of nodes with high bandwidth
  - Expensive infrastructure, incompatible with TCP/IP applications
- **Leverages commodity Ethernet switches and routers to interconnect cluster machines**
  - Backwards compatible with existing infrastructures, low-cost
  - Aggregate cluster bandwidth scales poorly with cluster size, and achieving the highest levels of bandwidth incurs non-linear cost increase with cluster size

Problems with common DC Topology

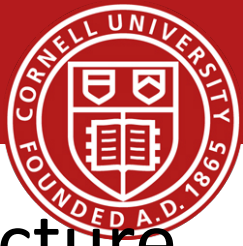


- Single point of failure
- Over subscript of links higher up in the topology
  - Trade off between cost and provisioning

Cost of maintaining switches

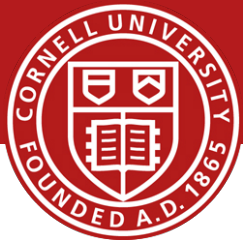


Properties of the solution



- Backwards compatible with existing infrastructure
  - No changes in application
  - Support of layer 2 (Ethernet)
- Cost effective
  - Low power consumption & heat emission
  - Cheap infrastructure
- Allows host communication at line speed

Clos Networks/Fat-Trees



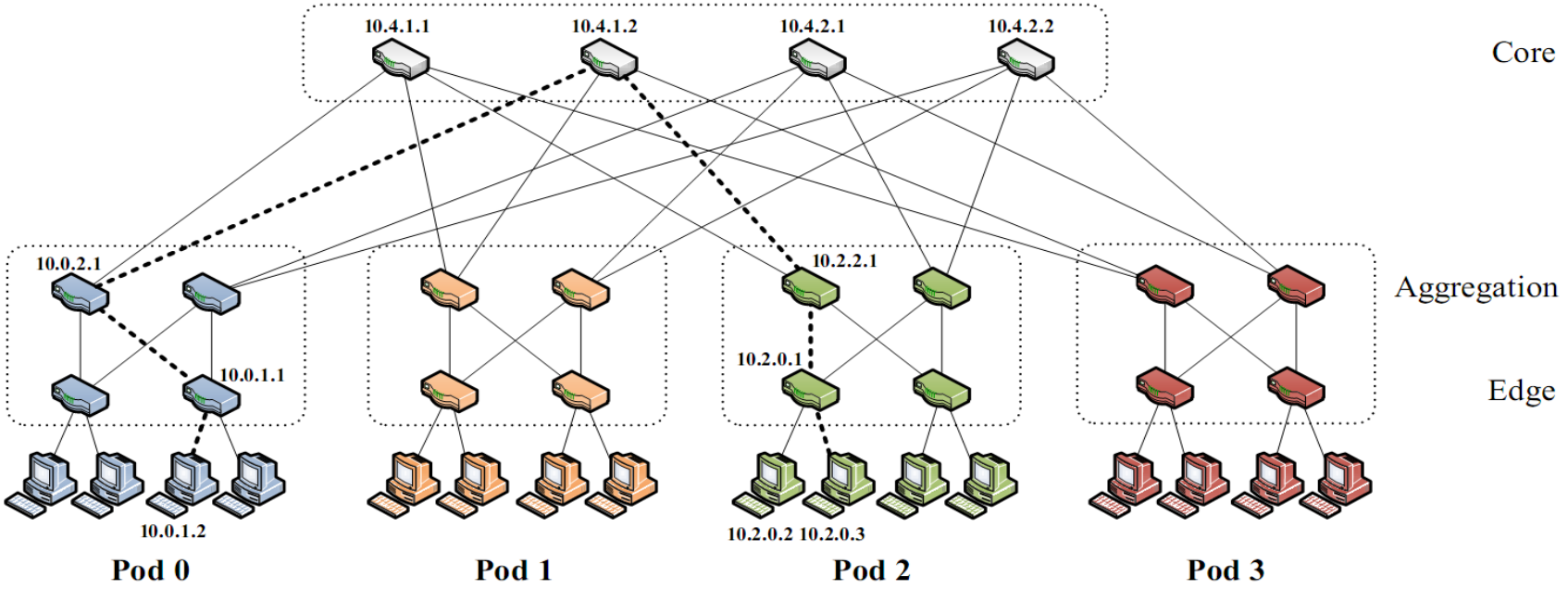
- Adopt a special instance of a Clos topology

- Similar trends in telephone switches led to designing a topology with high bandwidth by interconnecting smaller commodity switches.

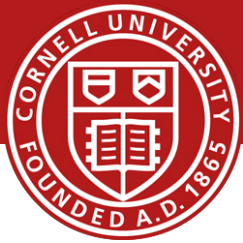
FatTree-based DC Architecture

- *Inter-connect racks (of servers) using a fat-tree topology*

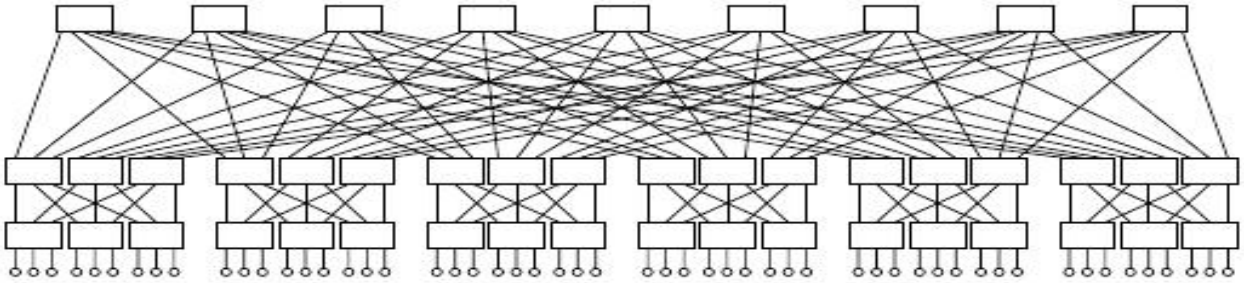
  K-ary fat tree: three-layer topology (edge, aggregation and core)
  - each pod consists of $(k/2)^2$ servers & 2 layers of k/2 k-port switches
  - each edge switch connects to k/2 servers & k/2 aggr. switches
  - each aggr. switch connects to k/2 edge & k/2 core switches
  - $(k/2)^2$ core switches: each connects to k pods



FatTree-based DC Architecture



- **Why Fat-Tree?**
  - Fat tree has identical bandwidth at any bisections
  - Each layer has the same aggregated bandwidth
- Can be built using cheap devices with uniform capacity
  - Each port supports same speed as end host
  - All devices can transmit at line speed if packets are distributed uniform along available paths
- Great scalability: k-port switch supports $k^3/4$ servers

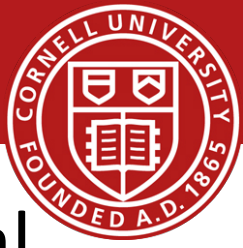


FatTree Topology is great, But…

Does using fat-tree topology to inter-connect racks of servers in itself sufficient?

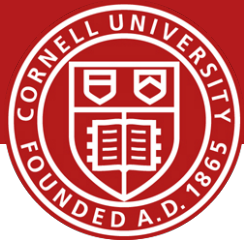
- What routing protocols should we run on these switches?
- Layer 2 switch algorithm: data plane flooding!
- Layer 3 IP routing:
  - shortest path IP routing will typically use only one path despite the path diversity in the topology
  - if using equal-cost multi-path routing at each switch independently and blindly, packet re-ordering may occur; further load may not necessarily be well-balanced
  - Aside: control plane flooding!

Problems with Fat-tree



- Layer 3 will only use one of the existing equal cost paths
  - Bottlenecks up and down the fat-tree
    - Simple extension to IP forwarding
- Packet re-ordering occurs if layer 3 blindly takes advantage of path diversity ; further load may not necessarily be well-balanced
- Wiring complexity in large networks
  - Packing and placement technique

FatTree Modified



- Enforce a special (IP) addressing scheme in DC
  - unused.PodNumber.switchnumber.Endhost
  - Allows host attached to same switch to route only through switch
  - Allows inter-pod traffic to stay within pod

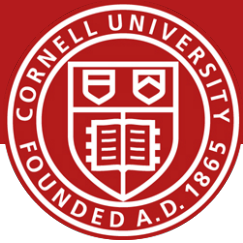
FatTree Modified

- Use two level look-ups to distribute traffic and maintain packet ordering
  - First level is prefix lookup
    - used to route down the topology to servers
  - Second level is a suffix lookup
    - used to route up towards core
    - maintain packet ordering by using same ports for same server
    - Diffuses and spreads out traffic

Prefix	Output port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

Suffix	Output port
0.0.0.2/8	2
0.0.0.3/8	3

FatTree Modified



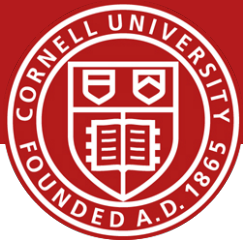
Diffusion Optimizations (routing options)

1. Flow classification, Denote a *flow* as a sequence of packets; pod switches forward subsequent packets of the same flow to same outgoing port. And periodically reassign a minimal number of output ports
   - Eliminates local congestion
   - Assign traffic to ports on a per-flow basis instead of a per-host basis, Ensure fair distribution on flows

FatTree Modified

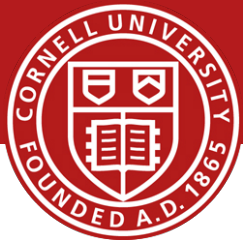
2. Flow scheduling, Pay attention to routing ***large flows***, edge switches detect any outgoing flow whose size grows above a predefined threshold, and then send notification to a ***central scheduler***. The central scheduler tries to assign non-conflicting paths for these large flows.
   - Eliminates global congestion
   - Prevent long lived flows from sharing the same links
   - Assign long lived flows to different links

Fault Tolerance



- In this scheme, each switch in the network maintains a BFD (Bidirectional Forwarding Detection) session with each of its neighbors to determine when a link or neighboring switch fails
  - Failure between upper layer and core switches
    - Outgoing inter-pod traffic, local routing table marks the affected link as unavailable and chooses another core switch
    - Incoming inter-pod traffic, core switch broadcasts a tag to upper switches directly connected signifying its inability to carry traffic to that entire pod, then upper switches avoid that core switch when assigning flows destined to that pod

Fault Tolerance



- Failure between lower and upper layer switches
  - Outgoing inter- and intra pod traffic from lower-layer,
    - the local flow classifier sets the cost to infinity and does not assign it any new flows, chooses another upper layer switch
  - Intra-pod traffic using upper layer switch as intermediary
    - Switch broadcasts a tag notifying all lower level switches, these would check when assigning new flows and avoid it
  - Inter-pod traffic coming into upper layer switch
    - Tag to all its core switches signifying its ability to carry traffic, core switches mirror this tag to all upper layer switches, then upper switches avoid affected core switch when assigning new flaws

Packing

- Increased wiring overhead is inherent to the fat-tree topology
- Each pod consists of 12 racks with 48 machines each, and 48 individual 48-port GigE switches
- Place the 48 switches in a centralized rack
- Cables moves in sets of 12 from pod to pod and in sets of 48 from racks to pod switches opens additional opportunities for packing to reduce wiring complexity
- Minimize total cable length by placing racks around the pod switch in two dimensions

Packing

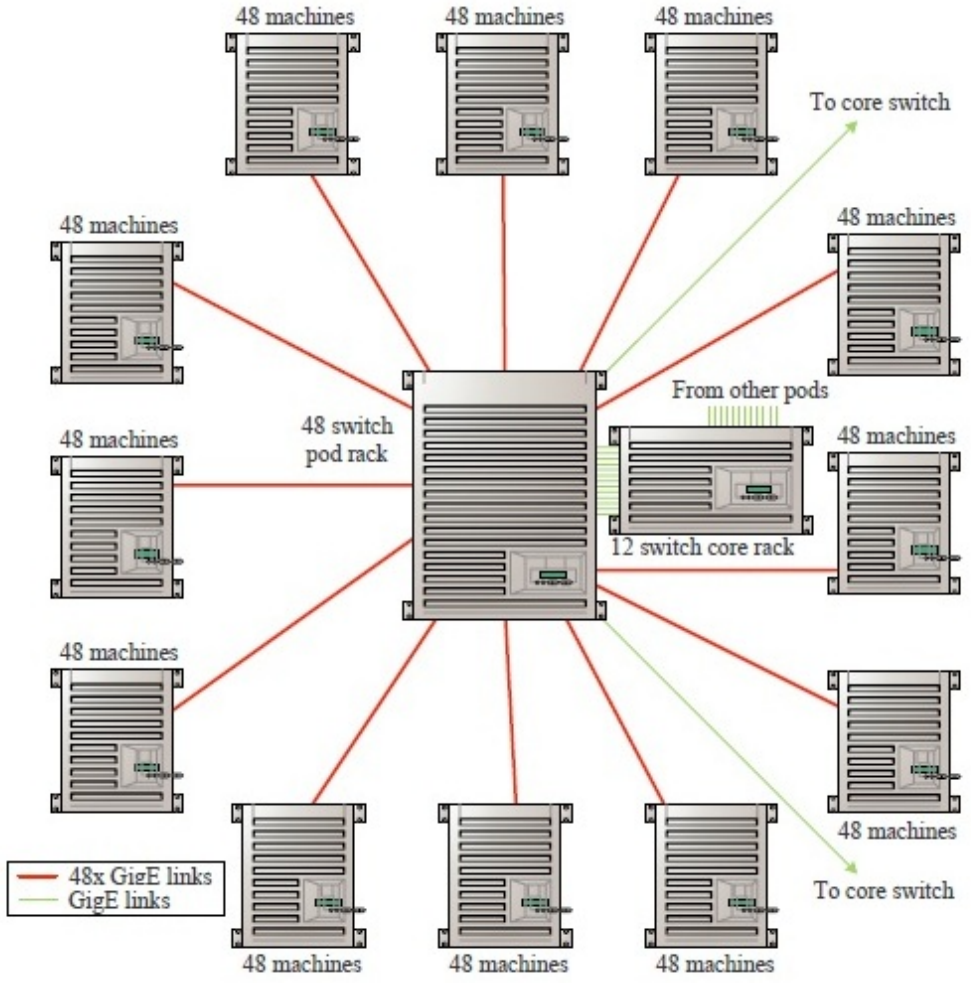
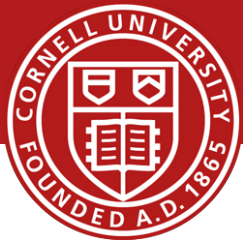


Figure 8: Proposed packaging solution. The only external cables are between the pods and the core nodes.

Evaluation

- Benchmark suite of communication mappings to evaluate the performance of the 4-port fat-tree using the TwoLevelTable switches, FlowClassifier ans the FlowScheduler and compare to hirarchical tree with 3.6:1 oversubscription ratio

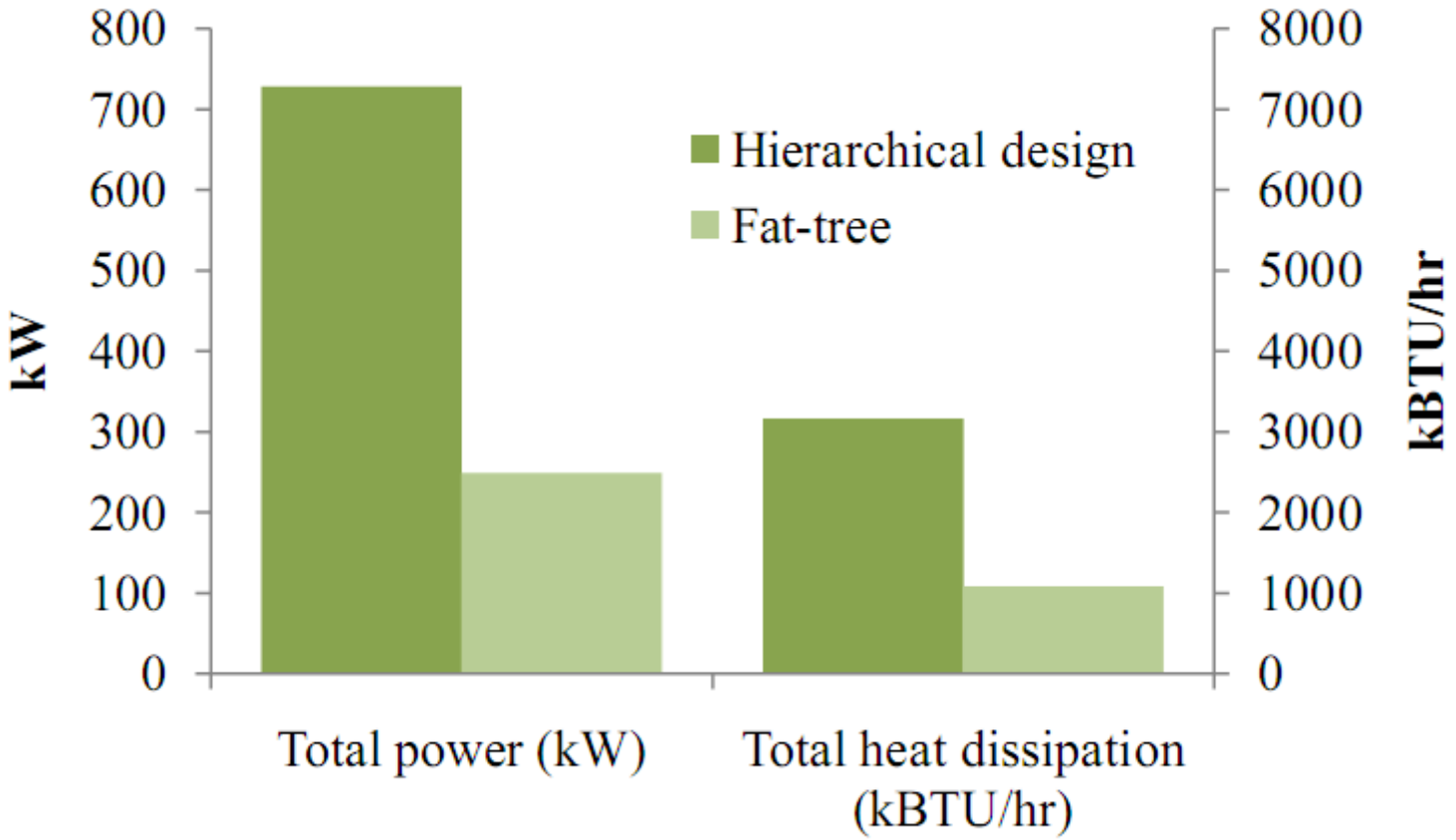
Results: Network Utilization



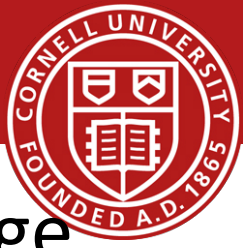
Test	Tree	Two-Level Table	Flow Classification	Flow Scheduling
Random	53.4%	75.0%	76.3%	93.5%
Stride (1)	100.0%	100.0%	100.0%	100.0%
Stride (2)	78.1%	100.0%	100.0%	99.5%
Stride (4)	27.9%	100.0%	100.0%	100.0%
Stride (8)	28.0%	100.0%	100.0%	99.9%
Staggered Prob (1.0, 0.0)	100.0%	100.0%	100.0%	100.0%
Staggered Prob (0.5, 0.3)	83.6%	82.0%	86.2%	93.4%
Staggered Prob (0.2, 0.3)	64.9%	75.6%	80.2%	88.5%
Worst cases:				
Inter-pod Incoming	28.0%	50.6%	75.1%	99.9%
Same-ID Outgoing	27.8%	38.5%	75.4%	87.4%

Table 2: Aggregate Bandwidth of the network, as a percentage of ideal bisection bandwidth for the Tree, Two-Level Table, Flow Classification, and Flow Scheduling methods. The ideal bisection bandwidth for the fat-tree network is 1.536Gbps.

Results: Heat & Power Consumption



Perspective



- Bandwidth is the scalability bottleneck in large scale clusters
- Existing solutions are expensive and limit cluster size
- Fat-tree topology with scalable routing and backward compatibility with TCP/IP and Ethernet
- Large number of commodity switches have the potential of displacing high end switches in DC the same way clusters of commodity PCs have displaced supercomputers for high end computing environments

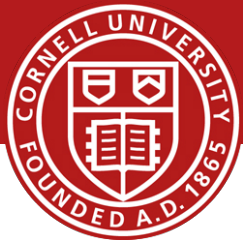
Other Data Center Architectures

- A Scalable, Commodity Data Center Network Architecture
  - a new Fat-tree "inter-connection" structure (topology) to increases "bi-section" bandwidth
    - needs "new" addressing, forwarding/routing
- VL2: A Scalable and Flexible Data Center Network
  - consolidate layer-2/layer-3 into a "virtual layer 2"
  - separating "naming" and "addressing", also deal with dynamic load-balancing issues

Other Approaches:

- PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric
- BCube: A High-Performance, Server-centric Network Architecture for Modular Data Centers

Before Next time



- Project Proposal
  - **CHANGE: Due *today*, Sept 22**
  - Meet with groups, TA, and professor
- Lab2
  - Multi threaded TCP proxy
  - **CHANGE: Due this tomorrow, Tuesday, Sept 23**

- ***Required review and reading for Friday, September 26***
  - "VL2: a scalable and flexible data center network", A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. *ACM Computer Communication Review (CCR)*, August 2009, pages 51-62.
  - http://dl.acm.org/citation.cfm?id=1592576
  - http://ccr.sigcomm.org/online/files/p51.pdf

- Check piazza: http://piazza.com/cornell/fall2014/cs5413
- Check website for updated schedule