



Spring 2026

COMP6103P

Advanced Computer Networking



Lectures

- Lectures Time:
 - Every Thursday 19:30 – 21:55 (March 5 – July 2)
- Course Website:
 - <https://snowzjx.me/COMP6103P/>
- Course Instructor:
 - Junxue ZHANG (snowzjx@ustc.edu.cn)
- TA:
 - Wenbo LI (wenboli@mail.ustc.edu.cn)
 - Maybe more TAs...



Course Instructors

- Junxue ZHANG
 - Professor, CS, USTC
 - Research Interest: Data Center Networking/High Performance Networking/Machine Learning Systems
 - Contact: snowzjx@ustc.edu.cn
 - Office: A510, #1 Academic Building, High Tech Campus
 - Office Hour: By Appointment (via email)



Lecture Formats

- Core purpose: Advanced Computer Networking
 - Latest research topics from academic
 - Latest application topics from industrial
 - Discuss papers from top-tier conference: SIGCOMM/SOSP/NSDI/OSDI
- Lower the barrier
 - Necessary background will also be covered
- Group Presentation (Tentative)
 - 4 Students form a group
 - Present one paper from our target conference



Course Prerequisite

- Basic knowledge on computer networking (brief concept is sufficient)
 - TCP/IP
 - Switch/Router
 - Internet/WAN/LAN
 - Socket Programming



Textbook/Course Materials

- No textbook is needed
- Recent papers from top-tier conference:
 - ACM SIGCOMM/SOSP
 - USENIX NSDI/OSDI
- How to obtain these papers?
 - ACM papers can be obtained from ACM digital (free access from campus network) library/conference website
 - <https://dl.acm.org/>
 - USENIX papers are fully opensource
 - E.g., <https://www.usenix.org/conference/nsdi25/presentation/du>



Course Outline

- Data Center Networking
 - Topology
 - Congestion Control
 - Flow Scheduling
 - Load Balance
 - RDMA
- SDN/Programmable Networking
- Advanced Internet Protocol
- Network Support for Machine Learning



Course Outline (Cont'd)

- Course outline may slightly change to include timely hot topics -> Please refer to course website
- Lecture notes will be made available on course website before lectures
- Arrangement for course presentations will also be available later on course website



Assignment & Grading Scheme

- Assignment: essay (40%)
 - Topic review (including at least 10 papers)
 - DDL: before final exam
- Final exam: open-book exam, no electrical devices are allowed (60%)
- Presentation: (up to 10% bonus)



Plagiarism & GenAI Policy

- Plagiarism: ZERO mark
- GenAI policy: LLM-based tools (e.g., ChatGPT, DeepSeek) and traditional AI-assisted writing tools (e.g., Grammarly) may be used, but only to improve the presentation and clarity of your essay—not to generate any substantive content.



Class WeChat Group

群聊：COMP6103P.03 ACN



该二维码7天内(3月11日前)有效，重新进入将更新



Data Center Networking

The Heart of Today's Computing Infrastructure



Topic Outline (~5 Lectures)

- **Background**
- **Topology**: A Scalable, Commodity Data Center Network Architecture, SIGCOMM 2008
- **Transports**: Data Center TCP (DCTCP), SIGCOMM 2010
- **Flow Scheduling**: Information-Agnostic Flow Scheduling for Commodity Data Centers, NSDI 2015
- **Load Balance**: CONGA: distributed congestion-aware load balancing for datacenters, SIGCOMM 2014
- **RDMA beyond TCP**: RDMA over Commodity Ethernet at Scale, SIGCOMM 2016



What are Data Centers

- Clusters of thousands of computers
- Where (almost) all important applications live

Google Datacenter



Microsoft Underwater Datacenter



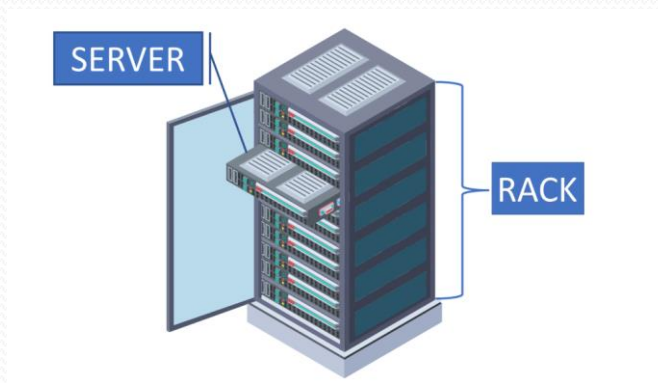
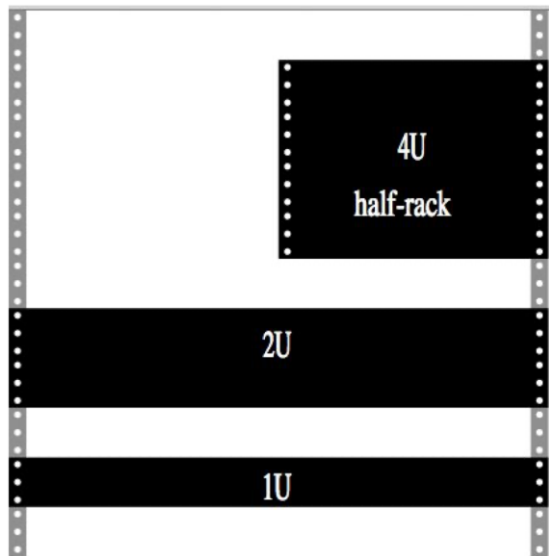






DCN is usually owned by ONE Authority

Data Center Racks





**DCN aims at connecting thousands of servers in
a relatively short distance in a regular way**



Fiber Cable used in DCN



QSFP 100Gbps Active Optical Cable (AOC)



**Bandwidth in DCN is very high: more than
100Gbps nowadays**



L3 Switch

- It uses **IP addresses** to make **forwarding decisions**
- **Forwarding:** Table lookup, which IP goes to which port

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

- **Routing Protocol/Algorithm:** How to generate these table entries?
 - Negotiation among switches: OSPF/BGP
 - Central controller (SDN)
- **In DCN, switches/routers use the same hardware, almost all papers use the term switch**



Applications Running in DC

- Diverse applications: Web server/AI/Big data Analytics
- **Server to Server Traffic** is more common in DCN
- **High Throughput** (large volume):
 - Machine Learning Training
 - Big Data Analytics
- **Low Latency** (small volume):
 - KV Cache
 - RPC Services
- High throughput \neq Low Latency



Long tail distribution

- **Few large flows dominate the bandwidth**
 - High throughput
- **Many small flows**
 - Low latency
 - Tail latency (90th/99th percentile)



DCN requires delivering high throughput and low latency communication simultaneously



Summary

	DCN	Internet
Ownership	Single administrative domain	Many
Scale	Short, within DC	Large, globally
Topology Structure	Structured, regular	Policy driven (AS)
Bandwidth	High (100-800 Gbps)	Low (~Mbps)
Latency	Ultra low (μ s–low ms)	Higher and variable (ms–100+ ms)
Traffic Pattern	Server-to-server/East-west Long-tail	Client-to-server/North-south
Switch	Shallow Buffer	Deep Buffer



Group Presentation Recommendations

- **Topology: VL2: a scalable and flexible data center network, SIGCOMM 2009 (Improvements over Fat-Tree)**
- **Transports: Congestion Control for Large-Scale RDMA Deployments, SIGCOMM 2015 (Transports for RDMA in DCN)**
- **Flow Scheduling: Coflow: A Networking Abstraction for Cluster Applications, SIGCOMM 2012 (From flow to coflow)**
- **Load Balance: Let It Flow: Resilient Asymmetric Load Balancing with Flowlet Switching, NSDI 17 (Simple is better)**



A Scalable, Commodity Data Center Network Architecture

Mohammad Al-Fares, Alexander Loukissas, Amin Vahdat

SIGCOMM 2008



Outline

- **Background**
- Fat Tree based solution
- Implementation and evaluation
- Review

How to connect so many servers?



Commodity switches have limited number of ports

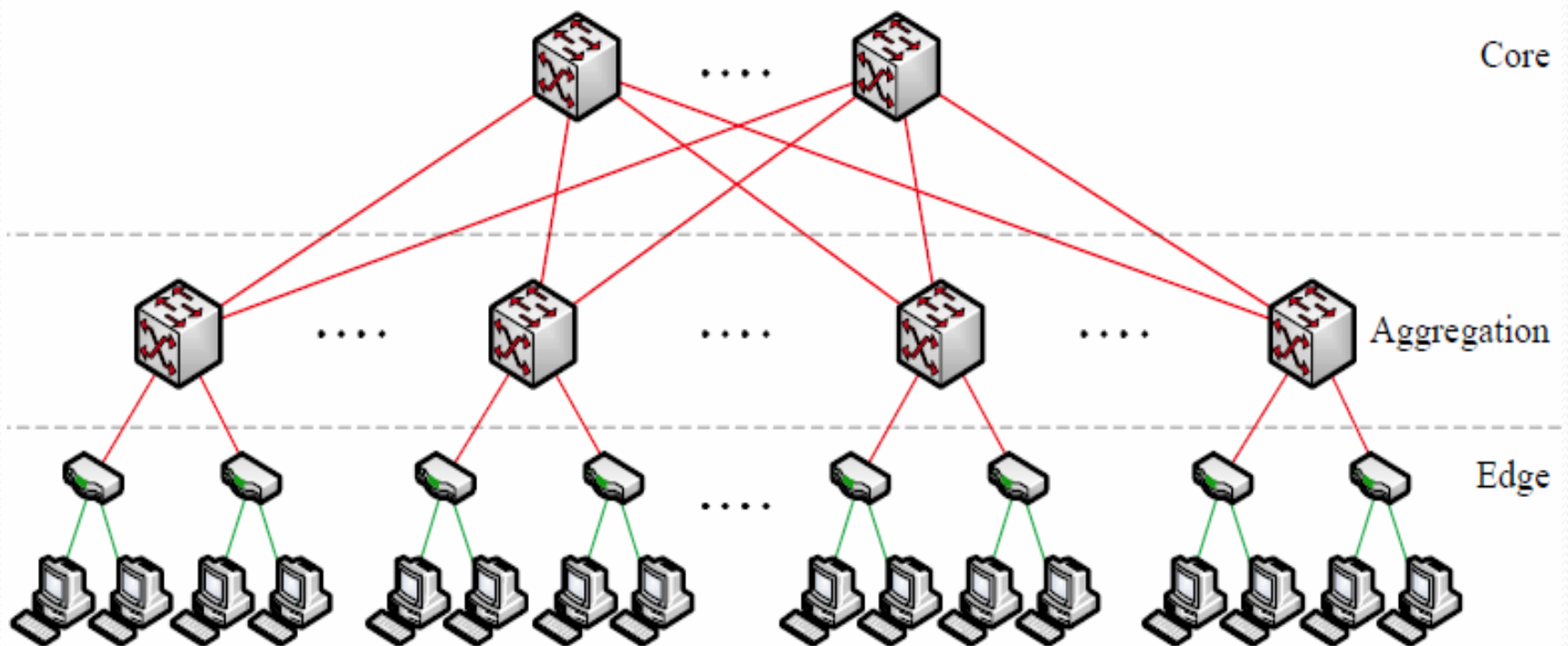


Can we build a More More More Advanced switch?
With 10000 ports?



Maybe, but EXPENSIVE!

A Layered Approach!

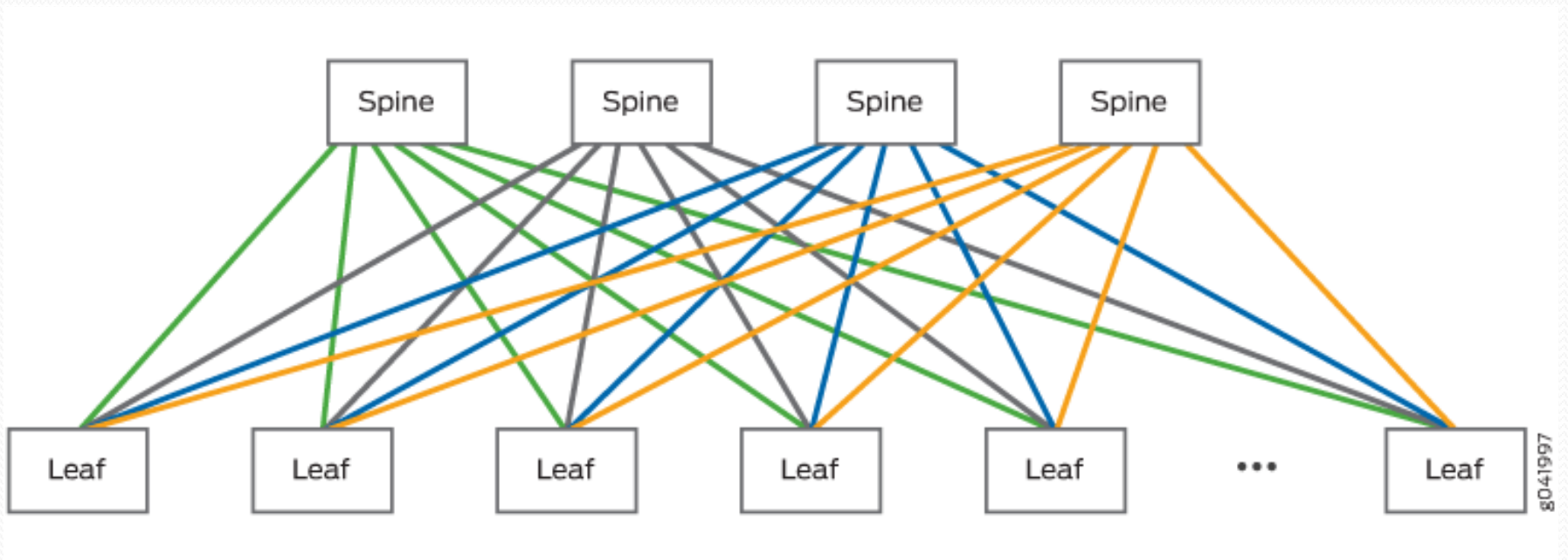




Conventional Data Center Network Topologies

- Three tiers: Core, Aggregation, Edge (ToR switch)
- Two types of switches (a little bit outdated):
 - 48-port GigE switch, with four 10 Gbps uplinks, used at the edge of the tree
 - 128-port 10 Gbps switch for higher levels of a communication hierarchy

A Layered Approach!



CLOS network is a multistage switching network.

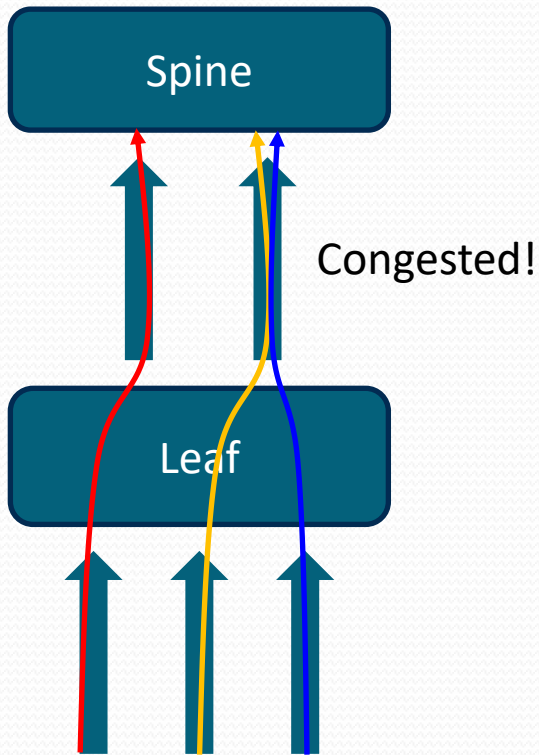


Oversubscription Ratio

- **Oversubscription**: the ratio of the total bisection bandwidth of a particular communication topology to the worst-case achievable aggregate bandwidth among the end hosts.
 - Ideal: 1:1, all hosts may potentially communicate with arbitrary other hosts at the full bandwidth of their network interface
 - Typical designs are oversubscribed by a factor of 2.5:1 to 8:1



Problems of the Topology



To achieve 1:1
oversubscription ratio,
more ports are need for
uplink

More switches are
needed!

More money!

Problems of the Topology

- **Cost:** if maintain a certain ratio of oversubscription, cost scale dramatically

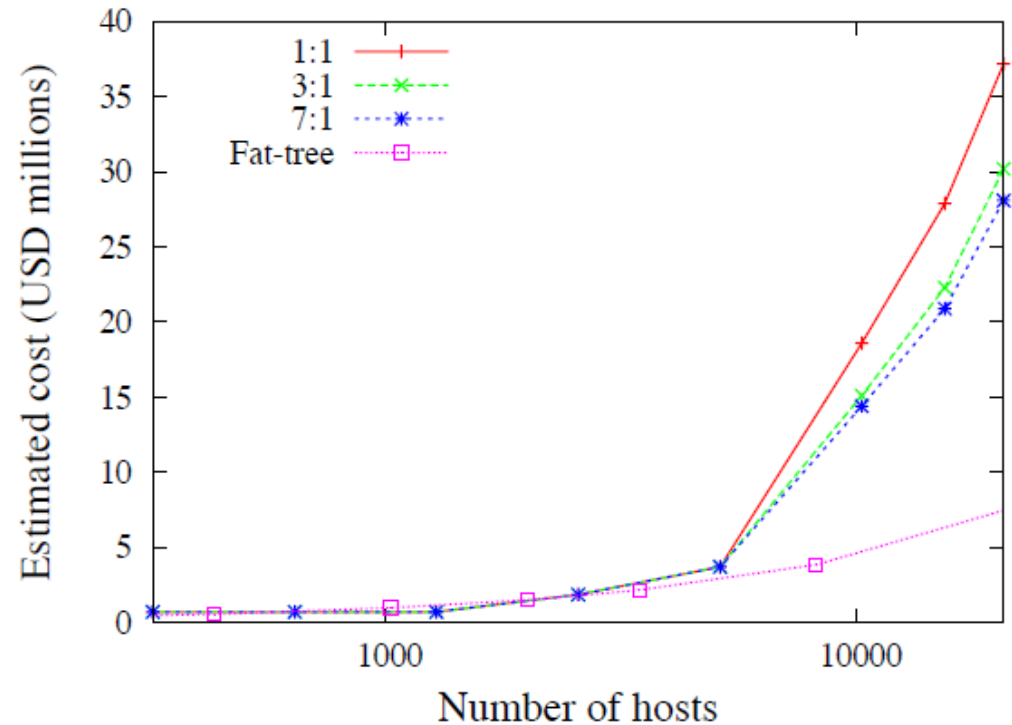


Figure 2: Current cost estimate vs. maximum possible number of hosts for different oversubscription ratios.

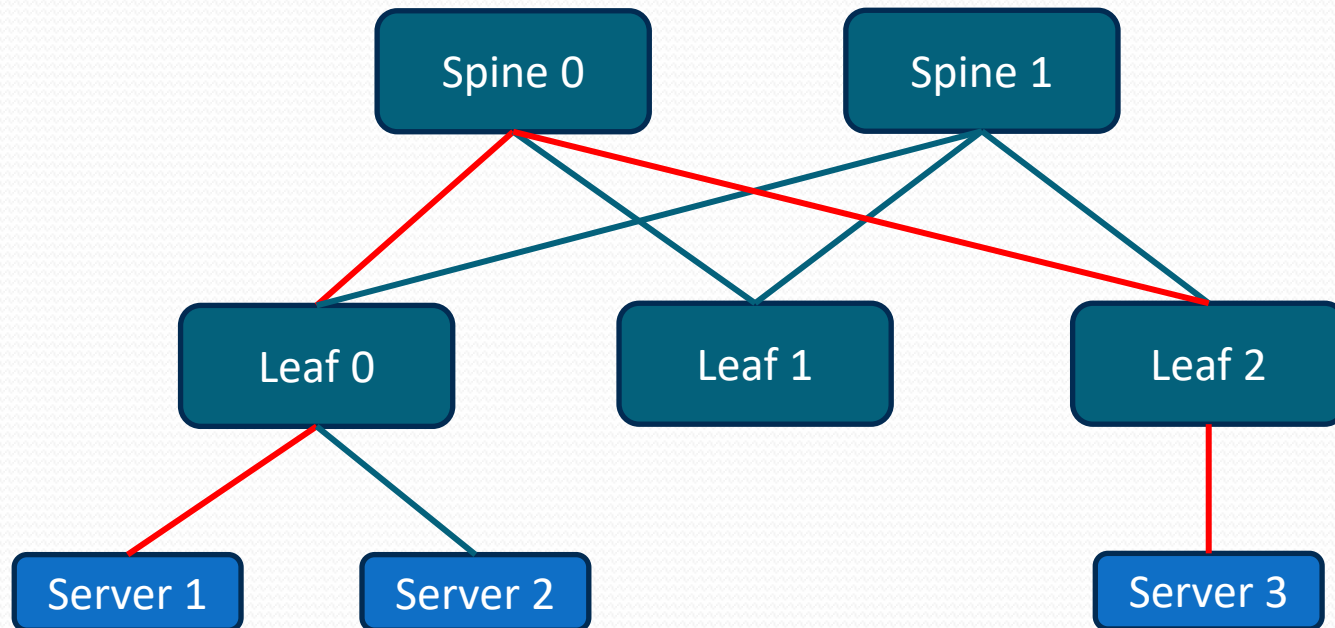


Multi-path is Natural in CLOS Network

- **Multi-path Routing:** Delivering full bandwidth between arbitrary hosts in larger clusters requires a “multi-rooted” tree with multiple core switches
 - ECMP performs static load-splitting among flows.
 - Limit the multiplicity of paths to 8–16

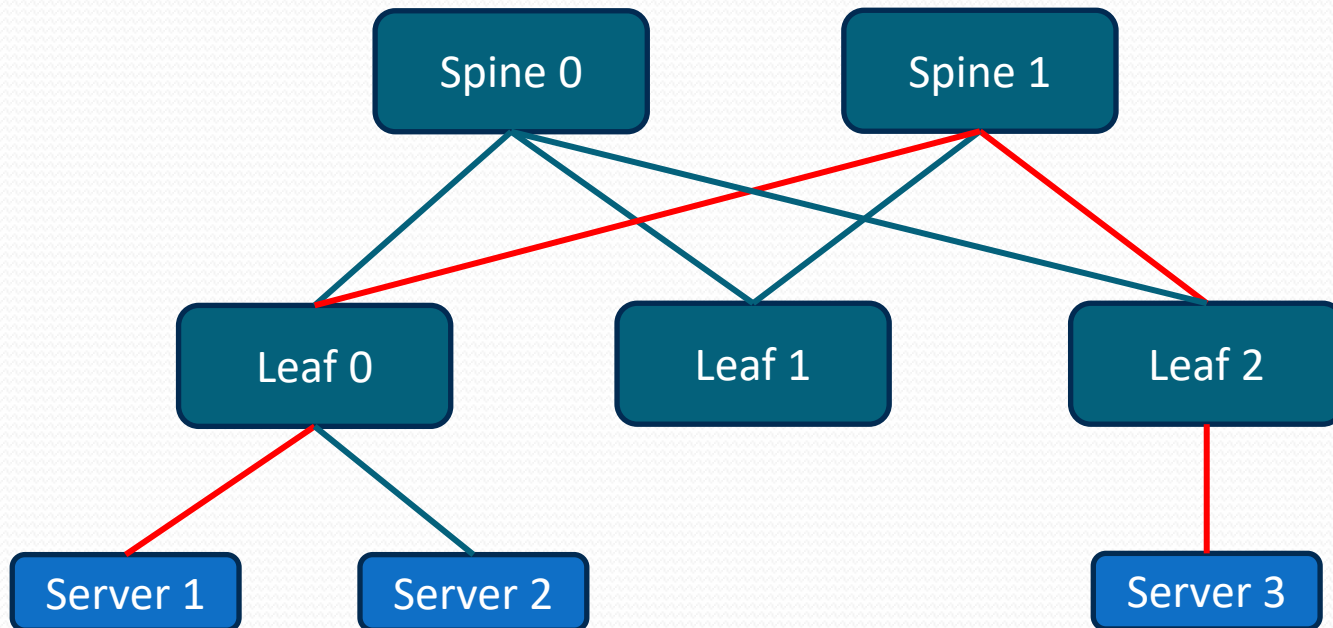


Multi-path Example





Multi-path Example

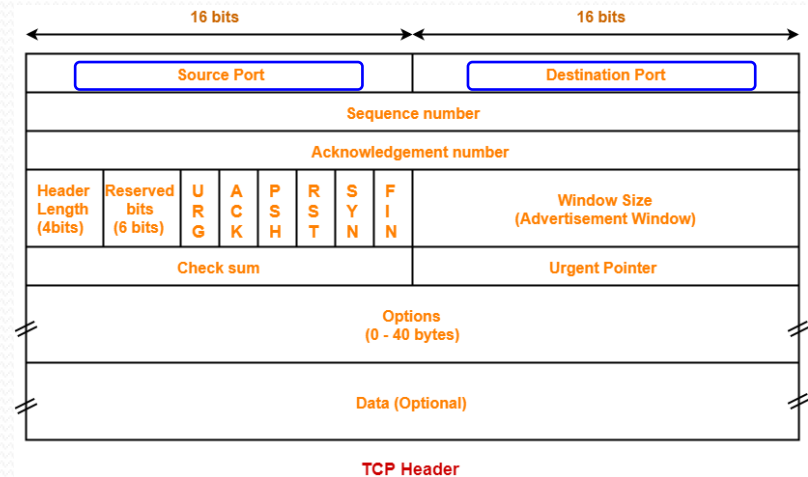
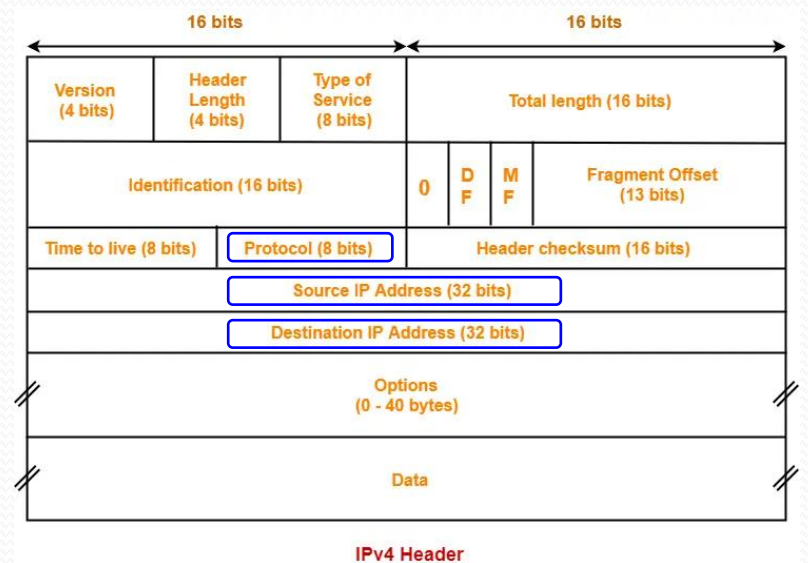


After reaching the top of the network, the downward path is identical!



ECMP

- Equal-Cost Multi-Path
- $\text{hash}(5\text{-tuple}) \% N_{\text{paths}}$
 - src IP
 - dst IP
 - src port
 - dst port
 - protocol
- Per flow
 - Avoid TCP reordering





Desired Properties for a DC Network Architecture

- **Scalable interconnection bandwidth:** an arbitrary host can communicate with any other host at the **full bandwidth** of its local network interface (non-blocking).
- **Economies of scale:** make **cheap** off-the-shelf Ethernet switches the basis for large scale data center networks.
- **Backward compatibility:** the entire system should be **backward compatible** with hosts running Ethernet and IP.

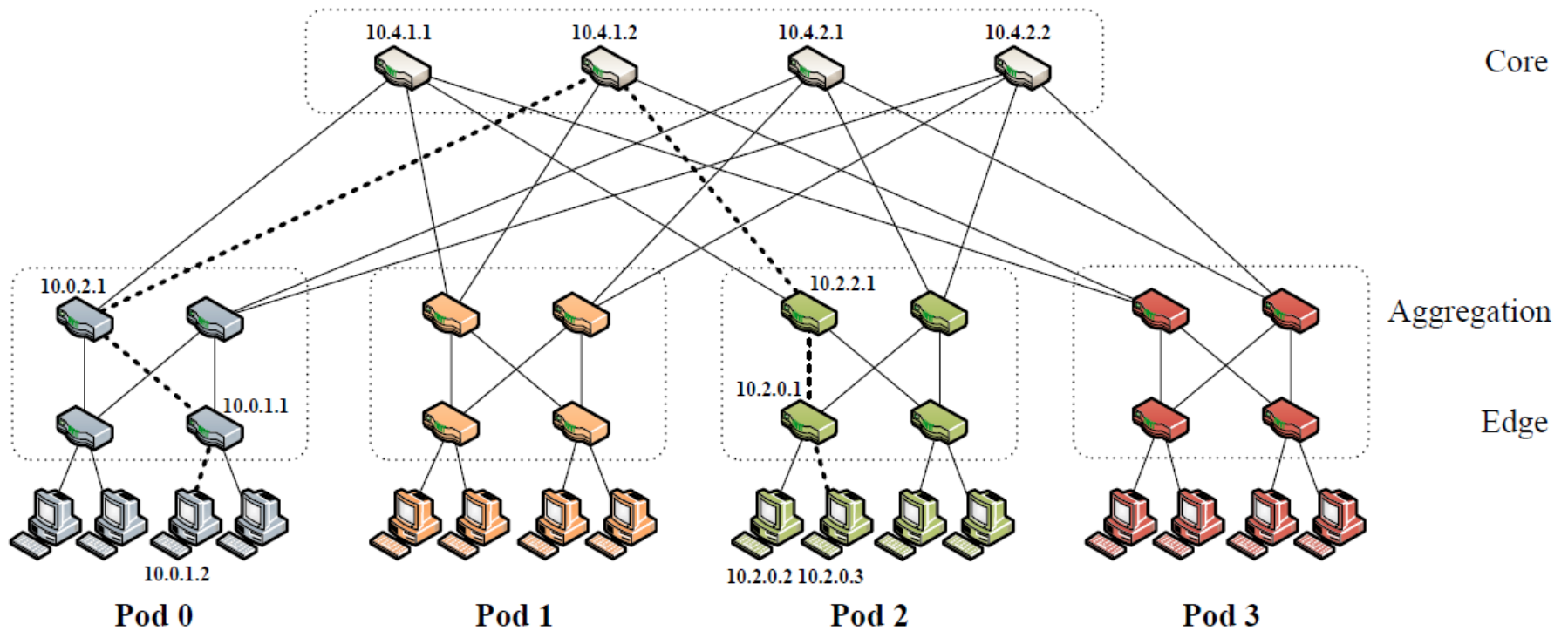


Outline

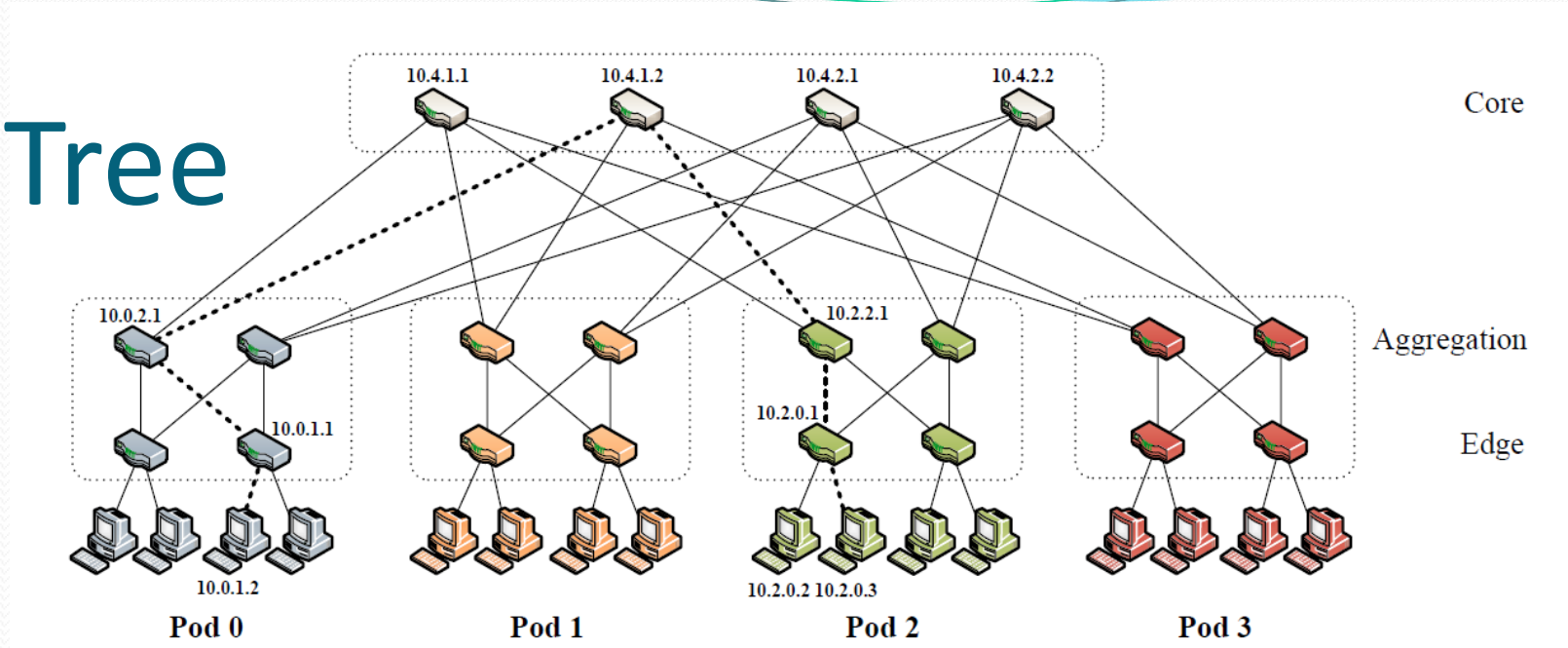
- Background
- **Fat Tree based solution**
- Implementation and evaluation
- Review



Fat-Tree

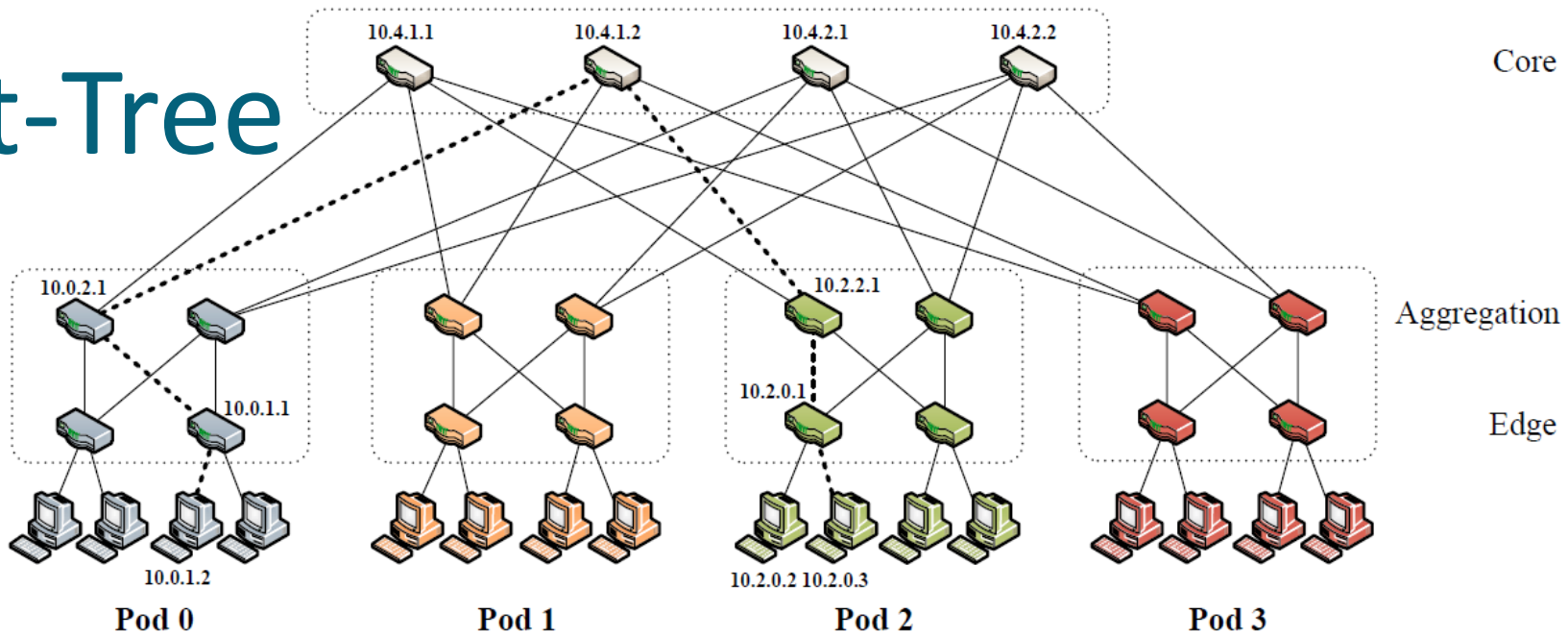


Fat-Tree



- k pods, each containing two layers of $k/2$ switches.
- Each k -port switch in the lower layer is directly connected to $k/2$ hosts.
- Each of the remaining $k/2$ ports is connected to $k/2$ of the k ports in the aggregation layer.

Fat-Tree



- $(k/2)^2$ k -port core switches. Each has one port connected to each of k pods.
- The i^{th} port of any core switch is connected to pod i
- $(k/2)^2$ **shortest-paths** between any two hosts



Fat-Tree

- Focus on designs up to $k = 48$.
- Use identical 48-port GigE switches.
- The network supports 27,648 hosts, made up of 1,152 subnets with 24 hosts each. There are **576** equal-cost paths between any given pair of hosts in different pods.
- The cost of deploying such a network architecture would be \$8.64M, compared to \$37M for the traditional techniques.



Architecture Design

- Motivation

- There are $(k/2)^2$ shortest-paths between any two hosts on different pods, but only one is chosen.
 - Each path has 5 hops (in terms of switch number)
- Protocols like OSPF selects path based **on hop counts**.
 - it is possible for a small subset of core switches, perhaps only one, to be chosen as the intermediate links between pods.
- Need a simple, fine-grained method of traffic diffusion.



Addressing

- All IP addresses in the network within the private 10.0.0.0/8 block.
- The pod switches are given addresses of the form $10.pod.switch.1$,
 - pod denotes the pod number (in $[0, k-1]$),
 - $switch$ denotes the position of that switch in the pod (in $[0, k-1]$, starting from left to right, bottom to top).
- Give core switches addresses of the form $10.k.j.i$,
 - j and i denote that switch's coordinates in the $(k/2)^2$ core switch grid (each in $[1, (k/2)]$, starting from top-left).



Two-level Routing Table

- Each entry in the main routing table will potentially have an additional pointer to a small secondary table of *(suffix, port)* entries.
 - A first-level prefix is **terminating** if it does not contain any second level suffixes,
 - A secondary table may be pointed to by more than one first-level prefix.

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



Two-level Routing Table

- Entries in the **primary table** are left-handed (i.e., $/m$ **prefix** masks of the form $1^m 0^{32-m}$), entries in the **secondary** tables are right-handed (i.e. $/m$ **suffix** masks of the form $0^{32-m} 1^m$).
- If the longest-matching prefix search yields a **non-terminating** prefix (**not in the pod...**), then the longest-matching suffix in the secondary table is found and used.

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



Two-level Routing Table

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

Figure 4: Two-level table example. This is the table at switch 10.2.2.1. An incoming packet with destination IP address 10.2.1.2 is forwarded on port 1, whereas a packet with destination IP address 10.3.0.3 is forwarded on port 3.

- The routing table of any pod switch will contain no more than $k/2$ prefixes and $k/2$ suffixes.



Routing Algorithm

How to generate these tables?

- Pod switches
 - In each pod switch, we assign terminating prefixes for subnets contained in the same pod. For inter-pod traffic, we add a /0 prefix with a secondary table matching host IDs.
 - **Employ the host IDs as a source of deterministic entropy** will cause traffic to be **evenly spread upward** among the outgoing links to the core switches



Routing Algorithm

- Core switches
 - Once a packet reaches a core switch, there is exactly one link to its destination pod, and that switch will include a terminating /16 prefix for the pod of that packet ($10.pod.0.0/16$, *port*).



Flow Classification

- Recognize subsequent packets of the same flow, and forward them on the same outgoing port
- Periodically reassign a minimal number of flow output ports to minimize any disparity between the aggregate flow capacity of different ports



Flow Scheduling/Load Balance

- **Recall the long-tail distribution**
- **Edge Switch**
 - Assign new flows to least-loaded port
 - Detect any outgoing flow whose size grows above a predefined threshold, and periodically send notifications to a central scheduler specifying the source and destination for all active large flows.
- **Central Scheduler**
 - Balance all those large flows



Short Summary

- Flow Classification
 - Local Optimization
 - Per-switch
 - Solve problem of ECMP collision
 - Do not have views of the global network
- Flow Scheduling/Load Balance
 - Global Optimization
 - Central controller



Flow Scheduling/Load Balance

- Flow scheduling: How to decide the flow priority when conflict occurs?
- Load balancing: The routing uses the host IDs as a source of deterministic entropy, but it can only ensure fairly distribution across host pairs/not flows. How to deal with the problem?
- **We will discuss these problems with two dedicated papers in the following lectures.**



Outline

- Background
- Fat Tree based solution
- **Implementation and evaluation**
- Review



Implementation

- Implement router prototype with Click.
 - Support the two-level routing table
 - 4-port
- The Click Modular Router Project
 - A software architecture for building flexible and configurable routers
 - <https://github.com/kohler/click>

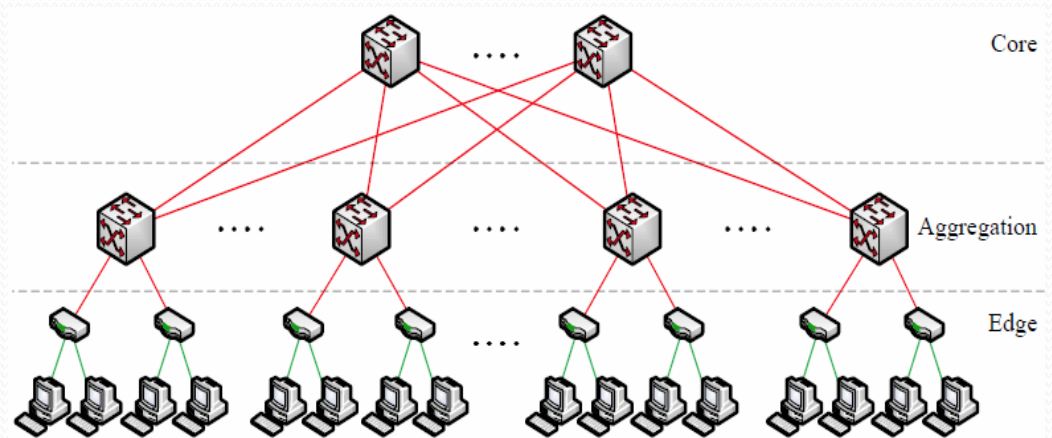


Experiment Description

- Implement a 4-port fat-tree ($k=4$): there are 16 hosts, four pods (each with four switches), and four core switches.
 - Multiplex these 36 elements onto ten physical machines, interconnected by a 48-port ProCurve 2900 switch with 1 Gigabit Ethernet links.
 - Each pod of switches is hosted on one machine; each pod's hosts are hosted on one machine; and the two remaining machines run two core switches each.

Experiment Description

- Hierarchical tree network,
 - four machines running four hosts each, and four machines each running pod switches with one additional uplink.
 - The four pod switches are connected to a 4-port core switch running on a dedicated machine





Benchmark Suite

- **Random**: A host sends to any other host in the network with uniform probability.
- **Stride(i)**: A host with index x will send to the host with index $(x+i) \bmod 16$.
- **Staggered Prob ($SubnetP, PodP$)**: Where a host will send to another host in its subnet with probability $SubnetP$, and to its pod with probability $PodP$, and to anyone else with probability $1-SubnetP-PodP$.



Benchmark Suite

- **Inter-pod Incoming:** Multiple pods send to different hosts in the same pod, and all happen to choose the same core switch. That core switch's link to the destination pod will be oversubscribed. The worst-case local oversubscription ratio for this case is $(k-1):1$.



Benchmark Suite

- **Same-ID Outgoing:** Hosts in the same subnet send to different hosts elsewhere in the network such that the destination hosts have the same host ID byte. Static routing techniques force them to take the same outgoing upward port. The worst-case ratio for this case is $(k/2):1$.



Results

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%

Aggregate Bandwidth of the network, as a percentage of ideal bisection bandwidth



Outline

- Background
- Fat Tree based solution
- Implementation and evaluation
- **Review**



Review

- What is the datacenter network? What is the desired property of the datacenter network?
- What is the traditional CLOS topology for the datacenter, its limitations?
- How Fat-tree differs from the traditional design? In
 - Topology
 - Addressing
 - Routing Algorithm