

# High-Performance Computing Networks at BYU

Lloyd Brown

October 10, 2013

- 1 Outline
- 2 HPC Introduction
  - What is HPC?
  - Types of HPC
- 3 Types of Communication
- 4 Infiniband
  - Physical Layer Characteristics
  - Encoding
  - Measured Performance
    - Bandwidth Comparison
    - Latency Comparison
  - Subnet Management
- 5 Topologies
  - Evaluating Topologies
- 6 Multi-path Ethernet and TCP/IP
- 7 Conclusions
- 8 Questions

# What makes a supercomputer, super?

HPC or High-Performance Computing, is characterized by workloads and hardware requirements

# What makes a supercomputer, super?

HPC or High-Performance Computing, is characterized by workloads and hardware requirements

- Significantly larger compute capability than an average system

# What makes a supercomputer, super?

HPC or High-Performance Computing, is characterized by workloads and hardware requirements

- Significantly larger compute capability than an average system
- Used to solve problems that are too large to easily be solved on a single, traditional system

# What makes a supercomputer, super?

HPC or High-Performance Computing, is characterized by workloads and hardware requirements

- Significantly larger compute capability than an average system
- Used to solve problems that are too large to easily be solved on a single, traditional system
- May utilize specialty hardware and software

# What makes a supercomputer, super?

HPC or High-Performance Computing, is characterized by workloads and hardware requirements

- Significantly larger compute capability than an average system
- Used to solve problems that are too large to easily be solved on a single, traditional system
- May utilize specialty hardware and software
- No specific threshold for capacity

# Nature of HPC Computing

In HPC, speedup comes from one of two sources:



# Nature of HPC Computing

In HPC, speedup comes from one of two sources:

- Using faster resources (eg. faster clock speeds)

# Nature of HPC Computing

In HPC, speedup comes from one of two sources:

- Using faster resources (eg. faster clock speeds)
- Using more resources (eg. using more processors) or *Parallelism*

# Nature of HPC Computing

In HPC, speedup comes from one of two sources:

- Using faster resources (eg. faster clock speeds)
- Using more resources (eg. using more processors) or  
*Parallelism*

Physics generally limits us on the faster resources, so we spend more time on parallelism.

# Parallelism and Communication Needs

- When utilizing multiple resources (eg. multiple processors), the program must:

# Parallelism and Communication Needs

- When utilizing multiple resources (eg. multiple processors), the program must:
  - Split up the workload

# Parallelism and Communication Needs

- When utilizing multiple resources (eg. multiple processors), the program must:
  - Split up the workload
  - Provide necessary coordination among resources

# Parallelism and Communication Needs

- When utilizing multiple resources (eg. multiple processors), the program must:
  - Split up the workload
  - Provide necessary coordination among resources
- The algorithm and data determine the nature of communication needs

# Parallelism and Communication Needs

- When utilizing multiple resources (eg. multiple processors), the program must:
  - Split up the workload
  - Provide necessary coordination among resources
- The algorithm and data determine the nature of communication needs
- Therefore for HPC problems, communication is key.



# Parallelism and Communication Needs

- When utilizing multiple resources (eg. multiple processors), the program must:
  - Split up the workload
  - Provide necessary coordination among resources
- The algorithm and data determine the nature of communication needs
- Therefore for HPC problems, communication is key.
  - For inter-process communication

# Parallelism and Communication Needs

- When utilizing multiple resources (eg. multiple processors), the program must:
  - Split up the workload
  - Provide necessary coordination among resources
- The algorithm and data determine the nature of communication needs
- Therefore for HPC problems, communication is key.
  - For inter-process communication
  - For communicating with data storage

# What kind of communication are we talking about?

# What kind of communication are we talking about?

- Programs that utilize multiple processors to split up work, need to communicate between threads or processes, to coordinate efforts, report on results, etc.

# What kind of communication are we talking about?

- Programs that utilize multiple processors to split up work, need to communicate between threads or processes, to coordinate efforts, report on results, etc.
- Communication between threads/processes on the same host (“*Intra-node*” communication) is extremely fast (usually via shared memory)

# What kind of communication are we talking about?

- Programs that utilize multiple processors to split up work, need to communicate between threads or processes, to coordinate efforts, report on results, etc.
- Communication between threads/processes on the same host ("*Intra-node*" communication) is extremely fast (usually via shared memory)
- If the processes are on different hosts, we have to go out to some communication fabric ("*Inter-node*" communication)

# What kind of communication are we talking about?

- Programs that utilize multiple processors to split up work, need to communicate between threads or processes, to coordinate efforts, report on results, etc.
- Communication between threads/processes on the same host (“*Intra-node*” communication) is extremely fast (usually via shared memory)
- If the processes are on different hosts, we have to go out to some communication fabric (“*Inter-node*” communication)
  - There's a lot of research in speeding up *intra-node* communication, but that's more of a Computer Science or Electrical Engineering problem. We'll spend our time today on *inter-node* communication

# What is Infiniband? And why do I care?

Infiniband is the most common high-performance interconnect used in HPC. It:



# What is Infiniband? And why do I care?

Infiniband is the most common high-performance interconnect used in HPC. It:

- is switched-fabric architecture (more like Fibre Channel than like Ethernet)

# What is Infiniband? And why do I care?

Infiniband is the most common high-performance interconnect used in HPC. It:

- is switched-fabric architecture (more like Fibre Channel than like Ethernet)
- utilizes multiple speeds, lanes, and links

# What is Infiniband? And why do I care?

Infiniband is the most common high-performance interconnect used in HPC. It:

- is switched-fabric architecture (more like Fibre Channel than like Ethernet)
- utilizes multiple speeds, lanes, and links
- provides:
  - extremely high bandwidth

# What is Infiniband? And why do I care?

Infiniband is the most common high-performance interconnect used in HPC. It:

- is switched-fabric architecture (more like Fibre Channel than like Ethernet)
- utilizes multiple speeds, lanes, and links
- provides:
  - extremely high bandwidth
  - extremely low latency (one-way  $< 5 \mu s$ , compared to approx.  $22 \mu s$  for 1GbE)

# What is Infiniband? And why do I care?

Infiniband is the most common high-performance interconnect used in HPC. It:

- is switched-fabric architecture (more like Fibre Channel than like Ethernet)
- utilizes multiple speeds, lanes, and links
- provides:
  - extremely high bandwidth
  - extremely low latency (one-way  $< 5 \mu\text{s}$ , compared to approx.  $22 \mu\text{s}$  for 1GbE)
- Speedup comes mostly from:
  - Short protocol stack (very little above layer 2)

# What is Infiniband? And why do I care?

Infiniband is the most common high-performance interconnect used in HPC. It:

- is switched-fabric architecture (more like Fibre Channel than like Ethernet)
- utilizes multiple speeds, lanes, and links
- provides:
  - extremely high bandwidth
  - extremely low latency (one-way  $< 5 \mu s$ , compared to approx.  $22 \mu s$  for 1GbE)
- Speedup comes mostly from:
  - Short protocol stack (very little above layer 2)
  - Low-latency switching (very little decision making in the switch)

# What is Infiniband? And why do I care?

Infiniband is the most common high-performance interconnect used in HPC. It:

- is switched-fabric architecture (more like Fibre Channel than like Ethernet)
- utilizes multiple speeds, lanes, and links
- provides:
  - extremely high bandwidth
  - extremely low latency (one-way  $< 5 \mu s$ , compared to approx.  $22 \mu s$  for 1GbE)
- Speedup comes mostly from:
  - Short protocol stack (very little above layer 2)
  - Low-latency switching (very little decision making in the switch)
  - Remote Direct Memory Access (RDMA)

# Lanes/Links/Speeds

Infiniband utilizes multiple lanes per physical link. Each link has a certain speed based on the standard:



# Lanes/Links/Speeds

Infiniband utilizes multiple lanes per physical link. Each link has a certain speed based on the standard:

	<i>SDR</i>	<i>DDR</i>	<i>QDR</i>	<i>FDR</i>
<i>1x</i>	2.5 Gb/s	5 Gb/s	10 Gb/s	14 Gb/s
<i>4x</i>	10 Gb/s	20 Gb/s	40 Gb/s	56 Gb/s
<i>12x</i>	30 Gb/s	60 Gb/s	120 Gb/s	168 Gb/s

# Encoding Overhead

Infiniband uses bit-line encodings to guarantee bit transitions for clock synchronization:

- SDR, DDR, QDR - 8b/10b encoding (8 data bytes encoded in 10 bytes total; 20% overhead)
- FDR and beyond - 64b/66b encoding (64 data bytes encoded in 66 bytes total; 3% overhead)

# Encoding Overhead

Infiniband uses bit-line encodings to guarantee bit transitions for clock synchronization:

- SDR, DDR, QDR - 8b/10b encoding (8 data bytes encoded in 10 bytes total; 20% overhead)
- FDR and beyond - 64b/66b encoding (64 data bytes encoded in 66 bytes total; 3% overhead)


	<i>SDR</i>	<i>DDR</i>	<i>QDR</i>	<i>FDR</i>
<i>1x</i>	2.5 Gb/s raw 2 Gb/s net	5 Gb/s raw 4 Gb/s net	10 Gb/s raw 8 Gb/s net	14 Gb/s raw 13.6 Gb/s net
<i>4x</i>	10 Gb/s raw 8 Gb/s net	20 Gb/s raw 16 Gb/s net	40 Gb/s raw 32 Gb/s net	56 Gb/s raw 54.3 Gb/s net
<i>12x</i>	30 Gb/s raw 24 Gb/s net	60 Gb/s raw 48 Gb/s net	120 Gb/s raw 96 Gb/s net	168 Gb/s raw 162.9 Gb/s net

# Performance at BYU's FSL

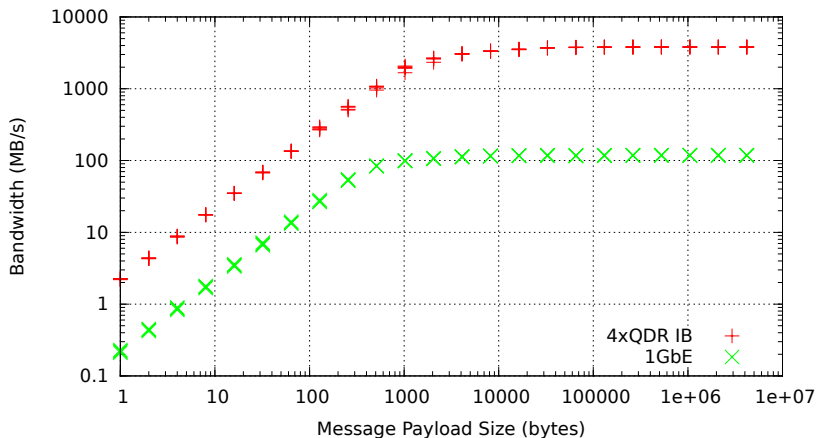
The graphs shown in the next couple of slides represent the bandwidth and latency performance of 4xQDR Infiniband vs 1Gb/s Ethernet at the Fulton Supercomputing Lab.

- All tests were performed host-to-host with one intervening switch (eg. host-switch-host)
- All tests utilize increasing message sizes, to demonstrate where one effect ends and the other starts
- Tests were performed using the “osu\_bw” and “osu\_latency” binaries from the OSU Micro-Benchmarks for MPI (a.k.a. “OMB”)<sup>1</sup>

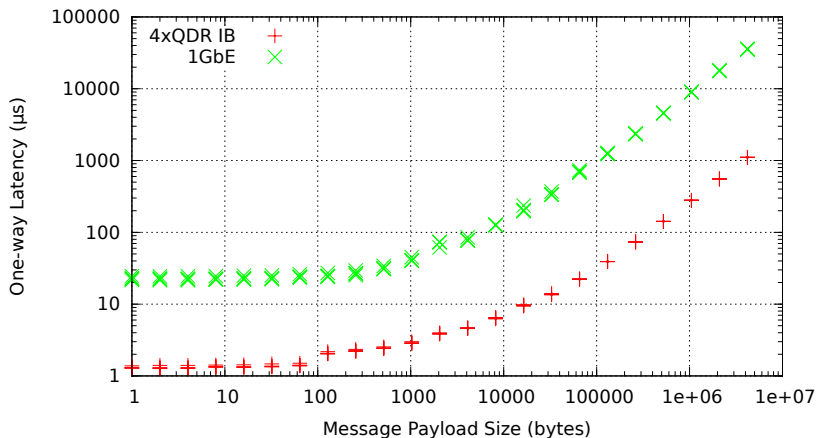
---

<sup>1</sup><http://mvapich.cse.ohio-state.edu/benchmarks/> 

Bandwidth Comparison - 4xQDR IB vs 1Gb/s Ethernet



One-way Latency Comparison - 4xQDR IB vs 1Gb/s Ethernet



# How Infiniband is Managed

Infiniband is designed as a trusted network. The network is managed by a *subnet manager* which does the following:

# How Infiniband is Managed

Infiniband is designed as a trusted network. The network is managed by a *subnet manager* which does the following:

- Periodically sweep the network, looking for topology changes, checking for errors, etc.



# How Infiniband is Managed

Infiniband is designed as a trusted network. The network is managed by a *subnet manager* which does the following:

- Periodically sweep the network, looking for topology changes, checking for errors, etc.
- Build a cohesive model of the network topology

# How Infiniband is Managed

Infiniband is designed as a trusted network. The network is managed by a *subnet manager* which does the following:

- Periodically sweep the network, looking for topology changes, checking for errors, etc.
- Build a cohesive model of the network topology
- Load the switch forwarding tables with the LID/Port mapping

# Infiniband Topologies

Infiniband puts very little restriction on the physical topology of the network.

- The Subnet Manager loads all the forwarding tables into the switches

# Infiniband Topologies

Infiniband puts very little restriction on the physical topology of the network.

- The Subnet Manager loads all the forwarding tables into the switches
  - as long as you can build an appropriate graph parsing algorithm, and implement it in a subnet manager, you can use a topology

# Infiniband Topologies

Infiniband puts very little restriction on the physical topology of the network.

- The Subnet Manager loads all the forwarding tables into the switches
  - as long as you can build an appropriate graph parsing algorithm, and implement it in a subnet manager, you can use a topology
  - allows some much more interesting topologies than those commonly Ethernet and TCP/IP networks usually use.<sup>2</sup>

---

<sup>2</sup>Technically you can use any topology with Ethernet as well. It just takes a huge amount of very-messy work, for very little benefit. I don't recommend trying it.

# Possible Topologies

# Possible Topologies

- Tree/Fat-Tree

# Possible Topologies

- Tree/Fat-Tree
- Fully-connected Mesh



# Possible Topologies

- Tree/Fat-Tree
- Fully-connected Mesh
- Torus

# Possible Topologies

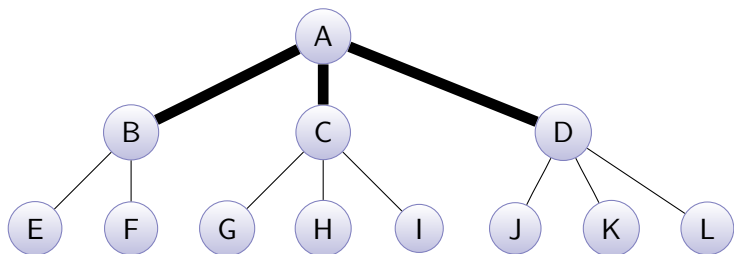
- Tree/Fat-Tree
- Fully-connected Mesh
- Torus
- Hypercube

# Possible Topologies

- Tree/Fat-Tree
- Fully-connected Mesh
- Torus
- Hypercube
- Folded-Clos Network

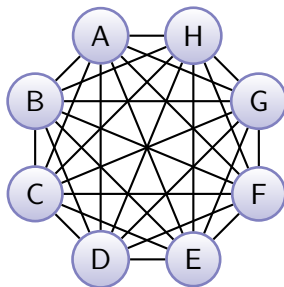
# Fat Tree Example

A *Fat Tree* is basically a tree with increased bandwidth (faster links or more links) between upper tiers relative to lower tiers; Ethernet has no problems with this one, so it's not terribly exciting



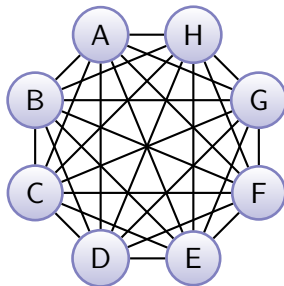
# Fully-connected Mesh Example

- Pro: Shortest hop-count (1 hop) from any point to any other point



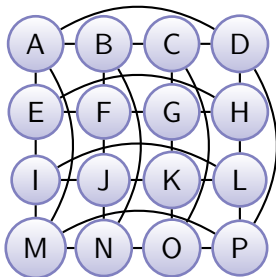
## Fully-connected Mesh Example

- Pro: Shortest hop-count (1 hop) from any point to any other point
- Con: takes a huge amount of cables, and the cable count increases very, very quickly.



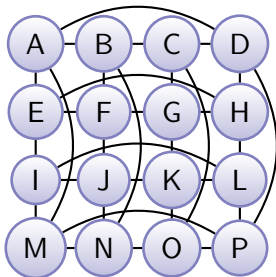
# Torus example

- Pro: Excellent for large topologies (no core switches to buy)



# Torus example

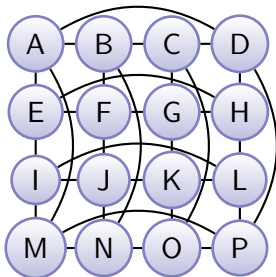
- Pro: Excellent for large topologies (no core switches to buy)
- Con: Higher hop count than other options, depending on size and shape





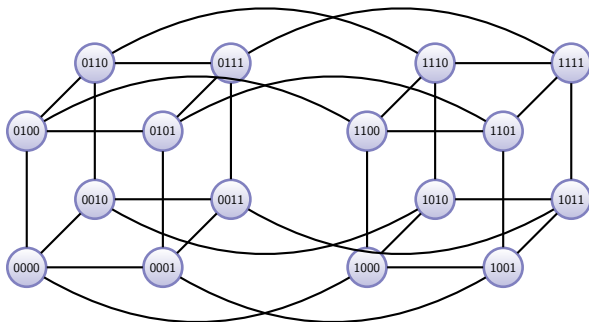
# Torus example

- Pro: Excellent for large topologies (no core switches to buy)
- Con: Higher hop count than other options, depending on size and shape
- Con: Less desirable bandwidth ratios (MBB to Client BW; discussed later)



# Hypercube example (4-dimensional)<sup>3</sup>

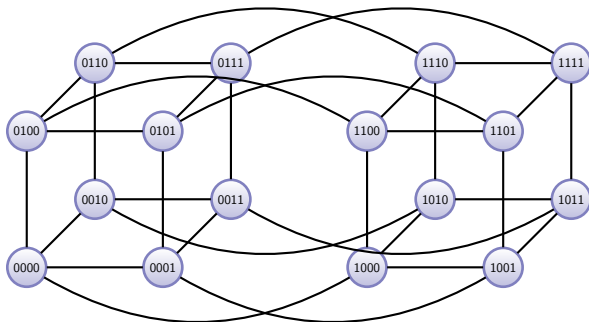
- Pro: for  $d$  dimensions, no more than  $d$  hops from any other point in the topology



<sup>3</sup>Note that this is really just a special case of a Torus. »

## Hypercube example (4-dimensional)<sup>3</sup>

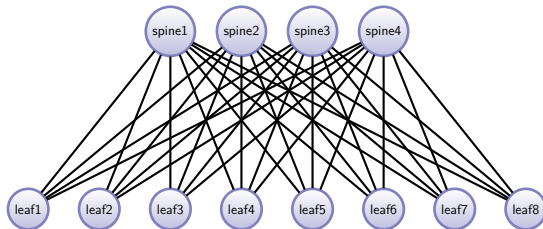
- Pro: for  $d$  dimensions, no more than  $d$  hops from any other point in the topology
- Con: cables/ports at each endpoint increase linearly with the dimension



<sup>3</sup>Note that this is really just a special case of a Torus. »

# Folded Clos Network Example

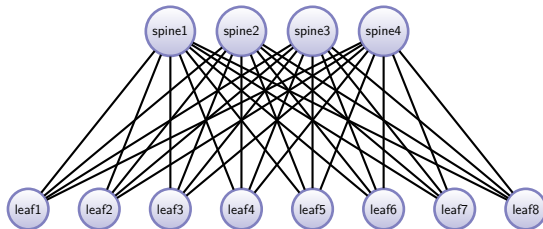
- Pros:
  - Most common approach for small or medium-scale Infiniband fabrics



# Folded Clos Network Example

## ■ Pros:

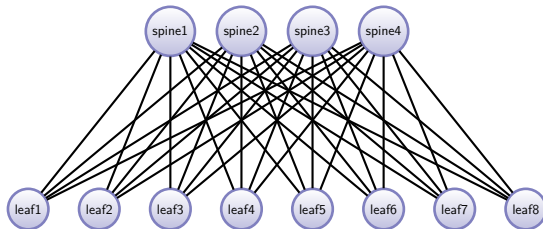
- Most common approach for small or medium-scale Infiniband fabrics
- Well understood (how larger IB switches are designed internally)



# Folded Clos Network Example

## ■ Pros:

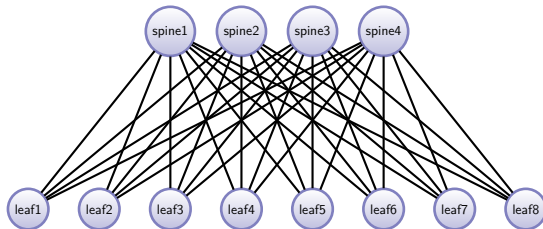
- Most common approach for small or medium-scale Infiniband fabrics
- Well understood (how larger IB switches are designed internally)
- Redundant; 1 link from each leaf to each spine



# Folded Clos Network Example

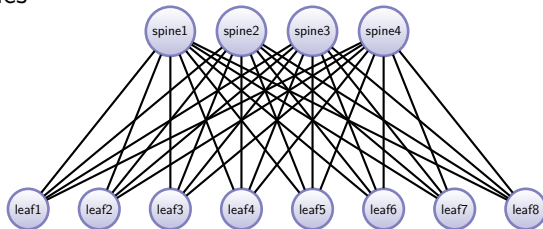
## ■ Pros:

- Most common approach for small or medium-scale Infiniband fabrics
- Well understood (how larger IB switches are designed internally)
- Redundant; 1 link from each leaf to each spine
- Easy to expand (up to the port count of the switches)



# Folded Clos Network Example

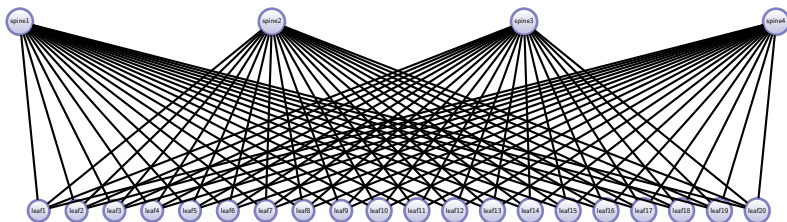
- Pros:
  - Most common approach for small or medium-scale Infiniband fabrics
  - Well understood (how larger IB switches are designed internally)
  - Redundant; 1 link from each leaf to each spine
  - Easy to expand (up to the port count of the switches)
- Con: Scalability limited by the port count of spine & leaf switches





# BYU Supercomputing's Clos Network

Note that this only shows the switches involved; there are 16 hosts attached to each leaf switch.



# What are some important characteristics for evaluating networks and topologies?

# What are some important characteristics for evaluating networks and topologies?

- Total host bandwidth

# What are some important characteristics for evaluating networks and topologies?

- Total host bandwidth
- Latency/hop-count

# What are some important characteristics for evaluating networks and topologies?

- Total host bandwidth
- Latency/hop-count
- Cost

# What are some important characteristics for evaluating networks and topologies?

- Total host bandwidth
- Latency/hop-count
- Cost
- Ease of expansion

# What are some important characteristics for evaluating networks and topologies?

- Total host bandwidth
- Latency/hop-count
- Cost
- Ease of expansion
- Minimum Bisection Bandwidth

# What are some important characteristics for evaluating networks and topologies?

- Total host bandwidth
- Latency/hop-count
- Cost
- Ease of expansion
- Minimum Bisection Bandwidth
- MBB to Client BW Ratio



# What's “Minimum Bisection Bandwidth”?

# What's “Minimum Bisection Bandwidth”?

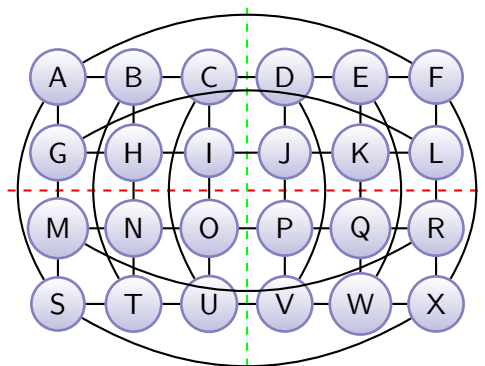
- If you were to draw a line across a topology, such that half the clients/switches/whatever are on each side of the line, the total bandwidth of all the links “cut” by that line is the *bisection bandwidth*

# What's “Minimum Bisection Bandwidth”?

- If you were to draw a line across a topology, such that half the clients/switches/whatever are on each side of the line, the total bandwidth of all the links “cut” by that line is the *bisection bandwidth*
- Of all the possible *bisection bandwidth* lines, the one with the minimum bandwidth is called the *minimum bisection bandwidth*

# MBB Example - Torus

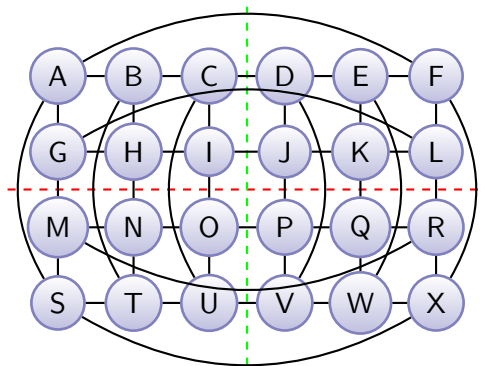
Which bisection line represents the minimum bandwidth bisection (assume all links are the same speed)?



# MBB Example - Torus

Which bisection line represents the minimum bandwidth bisection (assume all links are the same speed)?

- Green line cuts 8 links; red line cuts 12 links; Green is the *minimum*



# Why is MBB important?

# Why is MBB important?

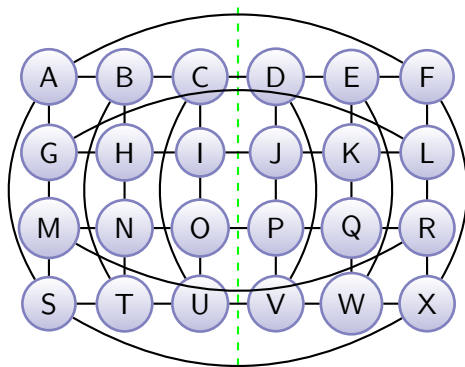
- MBB represents the available bandwidth during a worst-case scenario:

# Why is MBB important?

- MBB represents the available bandwidth during a worst-case scenario:
  - All the clients on one side of the MBB line are trying to communicate with someone on the other side of the line, as fast as possible



# MBB Example - Torus



# MBB vs Client BW - Torus

# MBB vs Client BW - Torus

- Frequently we compare MBB to total Client Bandwidth on one side of the MBB line

# MBB vs Client BW - Torus

- Frequently we compare MBB to total Client Bandwidth on one side of the MBB line
- Generally the smaller the ratio of MBB:Half-client-bandwidth, the better

# MBB vs Client BW - Torus

- Frequently we compare MBB to total Client Bandwidth on one side of the MBB line
- Generally the smaller the ratio of MBB:Half-client-bandwidth, the better
- For example, using the diagram on the Torus slide:

# MBB vs Client BW - Torus

- Frequently we compare MBB to total Client Bandwidth on one side of the MBB line
- Generally the smaller the ratio of MBB:Half-client-bandwidth, the better
- For example, using the diagram on the Torus slide:
  - Assuming each node is a switch, with 16 hosts hanging off it, what is the MBB:HalfClientBW ratio for the Green (MBB) line?
    - assume host-switch and switch-switch links are the same BW

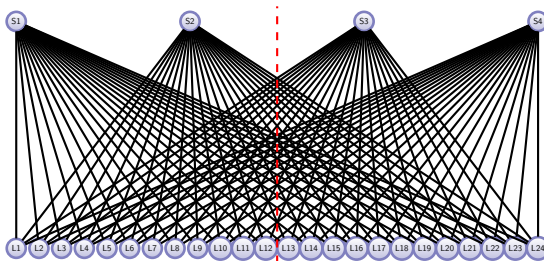
# MBB vs Client BW - Torus

- Frequently we compare MBB to total Client Bandwidth on one side of the MBB line
- Generally the smaller the ratio of MBB:Half-client-bandwidth, the better
- For example, using the diagram on the Torus slide:
  - Assuming each node is a switch, with 16 hosts hanging off it, what is the MBB:HalfClientBW ratio for the Green (MBB) line?
    - assume host-switch and switch-switch links are the same BW
  - Each half has 12 switches, or  $12 \times 16 = 192$  hosts, and the green line bisects 8 links, for a ratio of 24:1

# MBB vs Client BW - Clos

Anyone want to try this one?

- Assume that 16 hosts are attached to each of the 24 switches at the bottom, and none to the 4 on the top
- 4 links coming out of each of the 24 switches on the bottom (1 to each of 4 core switch)





# MBB vs Client BW - Clos

- Bisection line cuts 2 of 4 links per leaf switch (48 total links cut)

# MBB vs Client BW - Clos

- Bisection line cuts 2 of 4 links per leaf switch (48 total links cut)
- 12 switches per half \* 16 clients = 192 clients

# MBB vs Client BW - Clos

- Bisection line cuts 2 of 4 links per leaf switch (48 total links cut)
- 12 switches per half \* 16 clients = 192 clients
- $192:48 \Rightarrow 4:1$

# Multi-path Ethernet and TCP/IP

- Current efforts are underway to create multi-path options for more-common Ethernet and TCP/IP networks

# Multi-path Ethernet and TCP/IP

- Current efforts are underway to create multi-path options for more-common Ethernet and TCP/IP networks
- Some approaches:
  - ECMP - Equal Cost Multi-Path Routing (layer 3)
    - Several layer 3 routing protocols support ECMP, including OSPF, IS-IS, and EIGRP

# Multi-path Ethernet and TCP/IP

- Current efforts are underway to create multi-path options for more-common Ethernet and TCP/IP networks
- Some approaches:
  - ECMP - Equal Cost Multi-Path Routing (layer 3)
    - Several layer 3 routing protocols support ECMP, including OSPF, IS-IS, and EIGRP
  - TRILL - Transparent Interconnection of Lots of Links - Multi-path layer-2 Ethernet<sup>4</sup>

---

<sup>4</sup>The best reference I'm aware of is *Introduction to Trill* by Radia Perlman and Donald Eastlake, available at <http://www.ipjforum.org/?p=582>

# What do I need to learn from this?

# What do I need to learn from this?

- Not everything is Ethernet and TCP/IP



# What do I need to learn from this?

- Not everything is Ethernet and TCP/IP
- A Tree-like topology may not be the best arrangement for a specific application, especially in data centers

# What do I need to learn from this?

- Not everything is Ethernet and TCP/IP
- A Tree-like topology may not be the best arrangement for a specific application, especially in data centers
- You absolutely *must* understand the communication patterns of your application, in order to select the correct technology and topology

# What do I need to learn from this?

- Not everything is Ethernet and TCP/IP
- A Tree-like topology may not be the best arrangement for a specific application, especially in data centers
- You absolutely *must* understand the communication patterns of your application, in order to select the correct technology and topology
- What you're used to doing now, may change in the future

# Questions?

Any questions?