

# Foundations of High Performance Computing

## Lecture 7: network basics for parallel computing



2022-2023 Stefano Cozzini

**“Foundation of HPC” course**

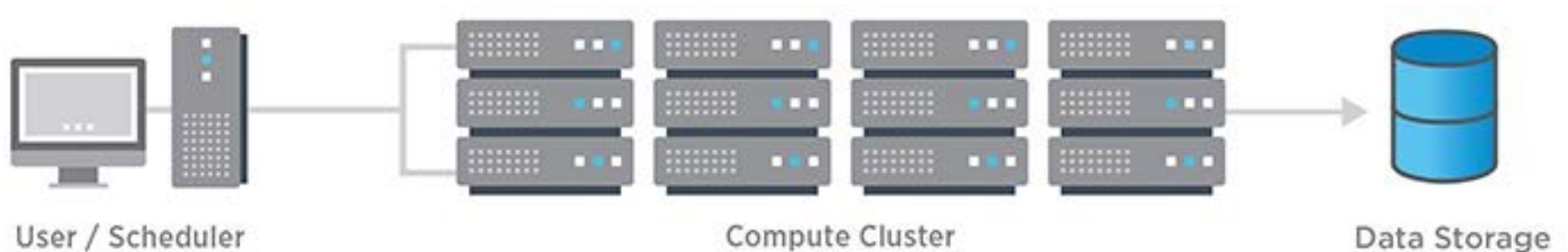
DATA SCIENCE &  
SCIENTIFIC COMPUTING

# Agenda


- Network basic for parallel architecture
- Network basic performance characteristics

# Recap on HPC architecture

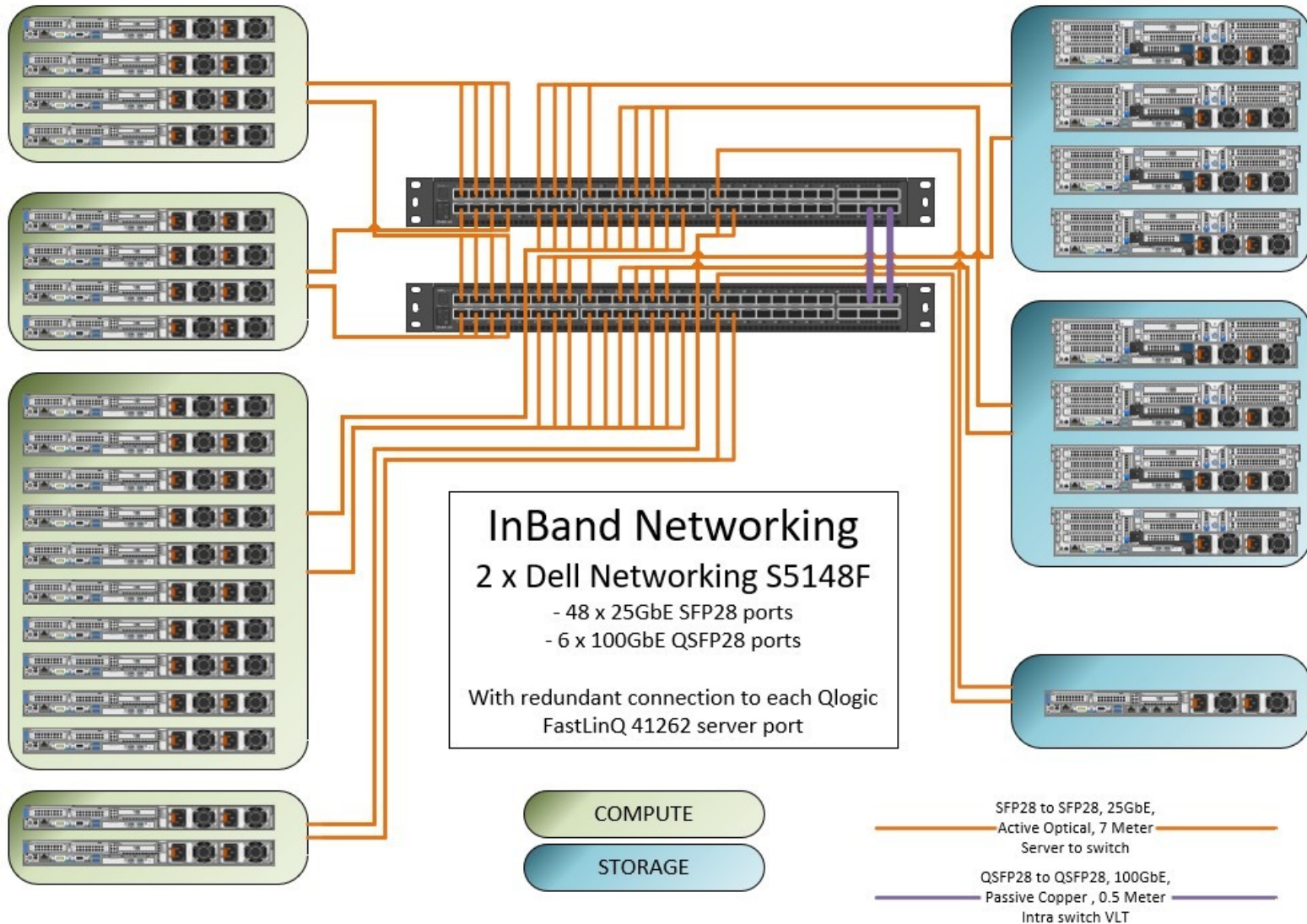
- Several computers (nodes) often in special cases for easy mounting in a rack
- One or more networks (interconnects) to hook the nodes together
- MP application' performance rely on the characteristics of the networks.



# Network cluster classification

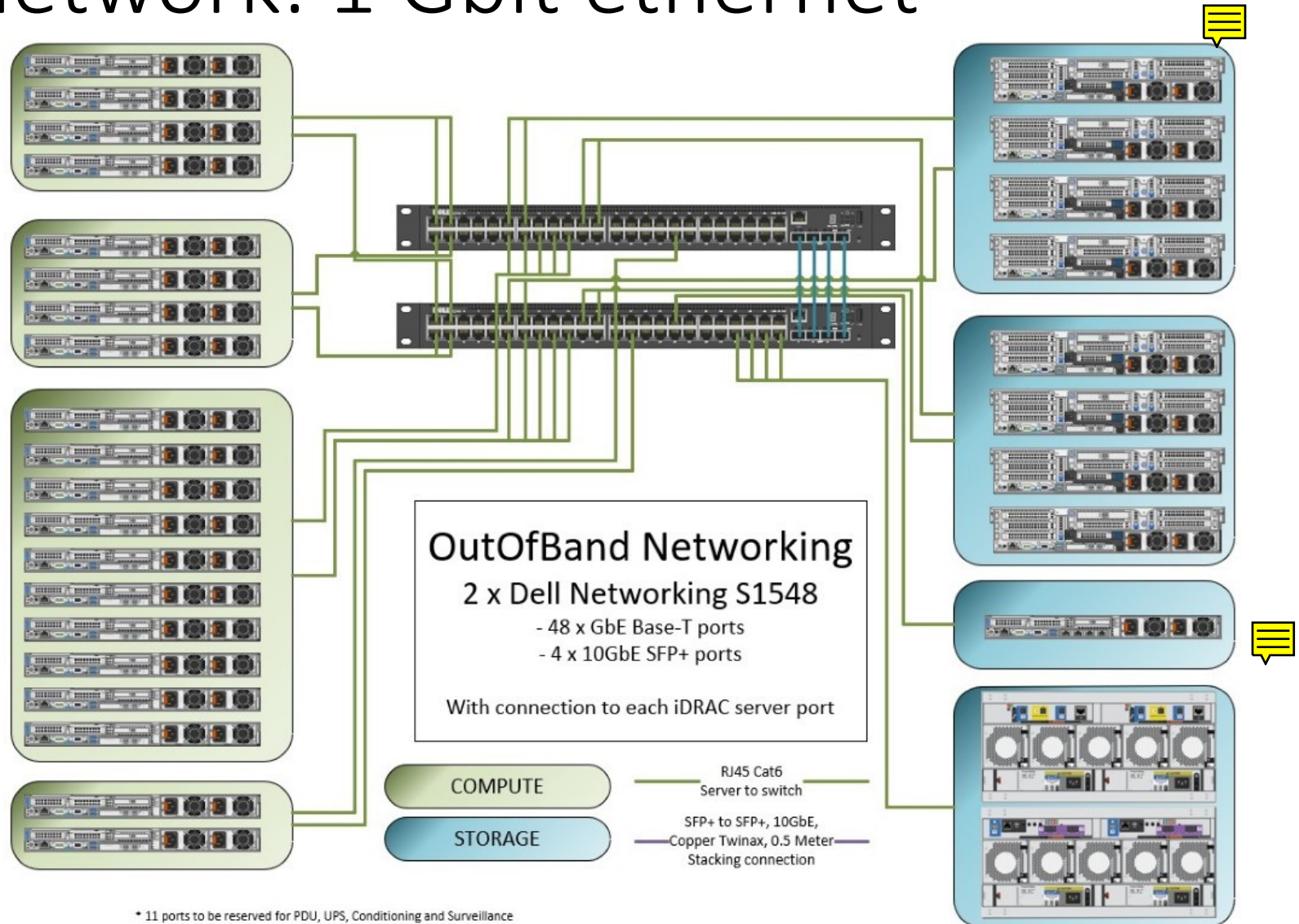
- HIGH SPEED NETWORK
  - parallel computation
  - low latency /high bandwidth
  - Usual choices: Infiniband...
- I/O NETWORK
  - I/O requests (NFS and/or parallel FS)
  - latency not fundamental/ good bandwidth 
  - GIGABIT could be ok /10Gb and/or Infiniband better
- In band Management network
  - management traffic of all services (LRMS/NFS/software etc..)
- Out of band Management network:
  - Remote control of nodes and any other device

# Orfeo in band management network: 25 Gbit ethernet

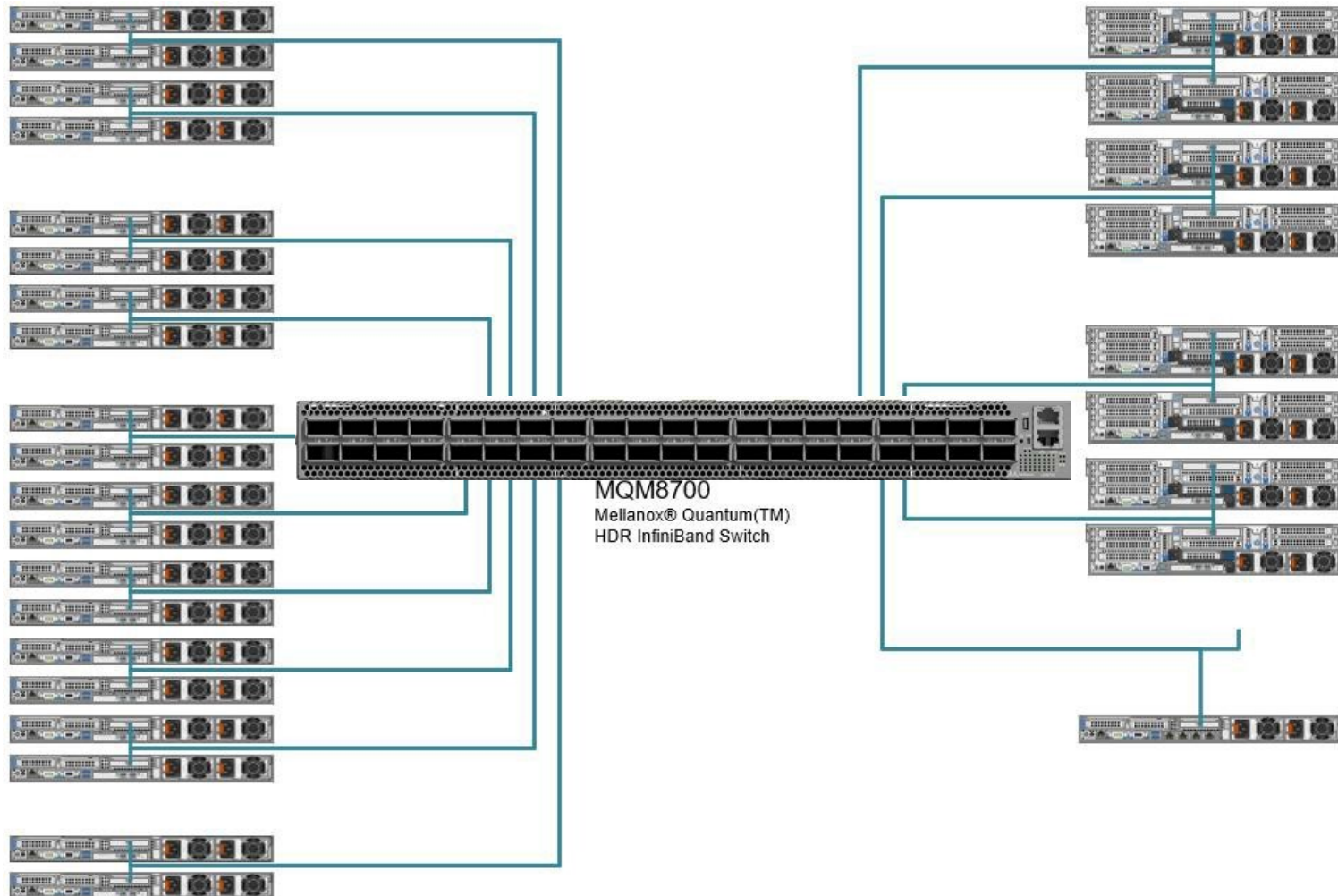




# Orfeo out of band management network: 1 Gbit ethernet



# Orfeo High Speed network: 100 Gbit Infiniband



IB HDR 200Gb/s to 2x100Gb/s  
QSFP56 to 2xQSFP56, LSZH

# Orfeo network classification

- HIGH SPEED NETWORK

100 Gbit HDR Infiniband

- I/O NETWORK

- In band Management network

25Gbit Ethernet

- Out of band Management network:

1Gbit Ethernet



# How to model network performance ?

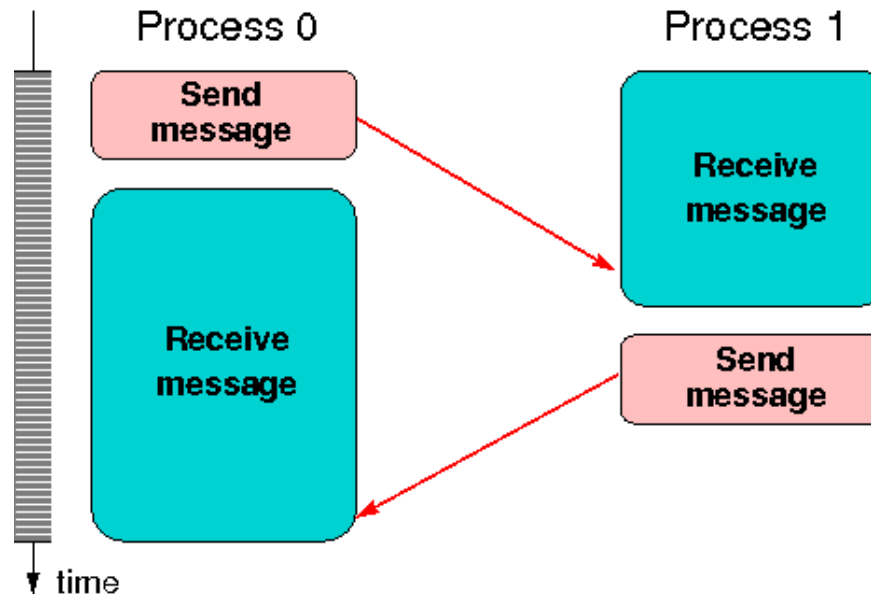
- Network capacity to transfer data
- Very simple model :
  - Total transfer time of a message

$$T_{comm} = \lambda + (\text{Size of message}) / b_{network}$$

- $\lambda$  is the latency of the network : i.e. the time to setup the communication channel
- $b_{network}$  is the asymptotic network bandwidth measured in Mb/sec.

# How can we estimate/measure latency and bandwidth ?

- Using a simple “Ping-Pong” program :
  - Two processes on the network exchange point-to-point message.
  - A single message of N Bytes is sent forward and backward: data transfer is  $2N$



# Ping-Pong algorithm

```
1 myID= get_process_ID()
2 if(myID.eq.0) then
3   targetID= 1
4   S = get_walltime()
5   call Send_message(buffer,N,targetID)
6   call Receive_message(buffer,N,targetID)
7   E = get_walltime()
8   MBYTES = 2*N/(E-S)/1.d6 ! MBytes/sec rate
9   TIME = (E-S)/(2*1.d6) ! transfer time in microsecs
10                                ! for single message
11 else
12   targetID= 0
13   call Receive_message(buffer,N,targetID)
14   call Send_message(buffer,N,targetID)
15 endif
```

# Ping Pong implementations

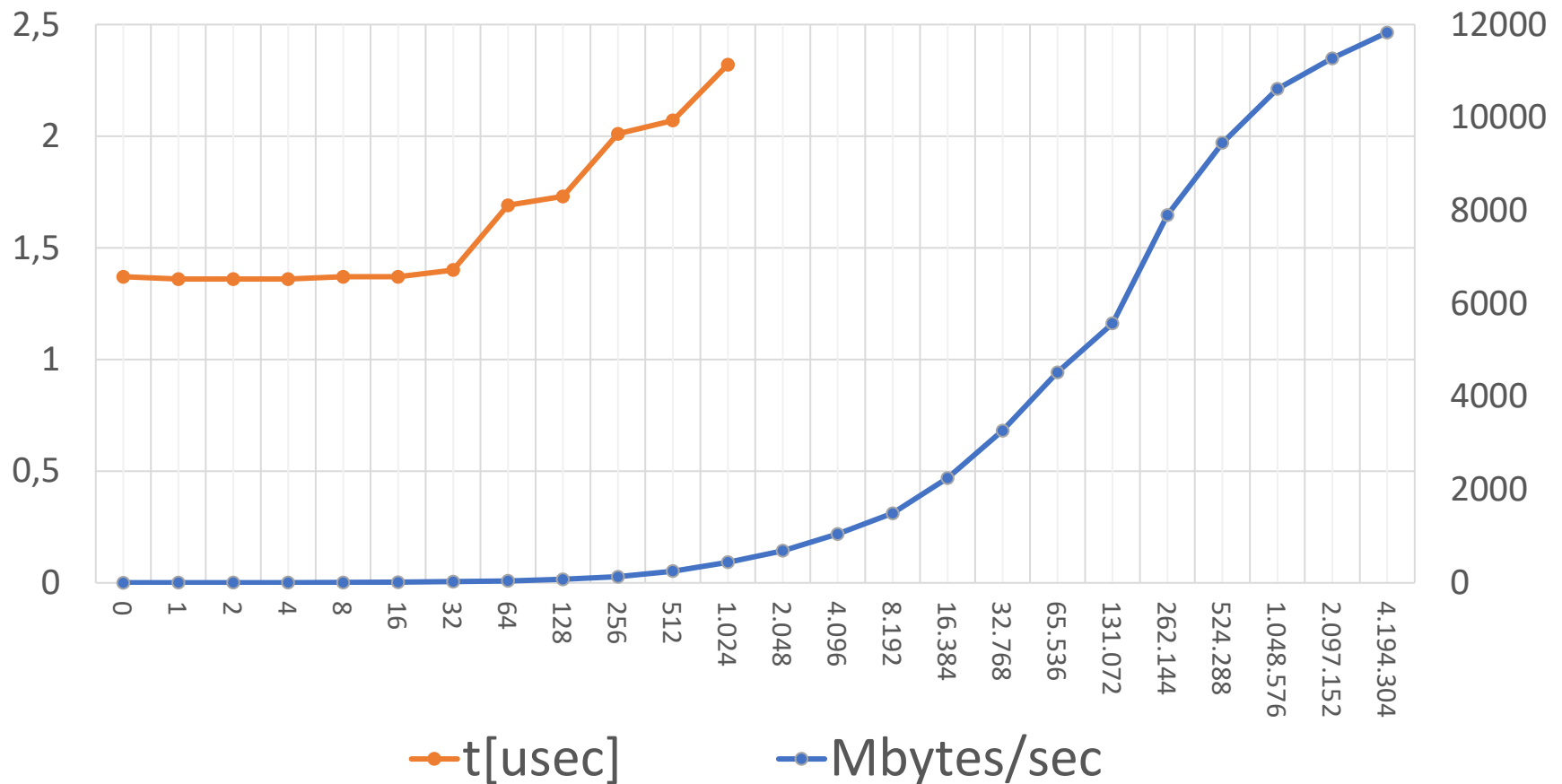
- Available on the most common benchmark suite:
- IMB: Intel MPI benchmark
  - [intel/mpi-benchmarks \(github.com\)](https://github.com/intel/mpi-benchmarks)
- OSU microbenchmarks
  - [MVAPICH :: Benchmarks \(ohio-state.edu\)](http://mvapich.cse.ohio-state.edu/Benchmarks)

# Measuring MPI point-to-point performance on Orfeo

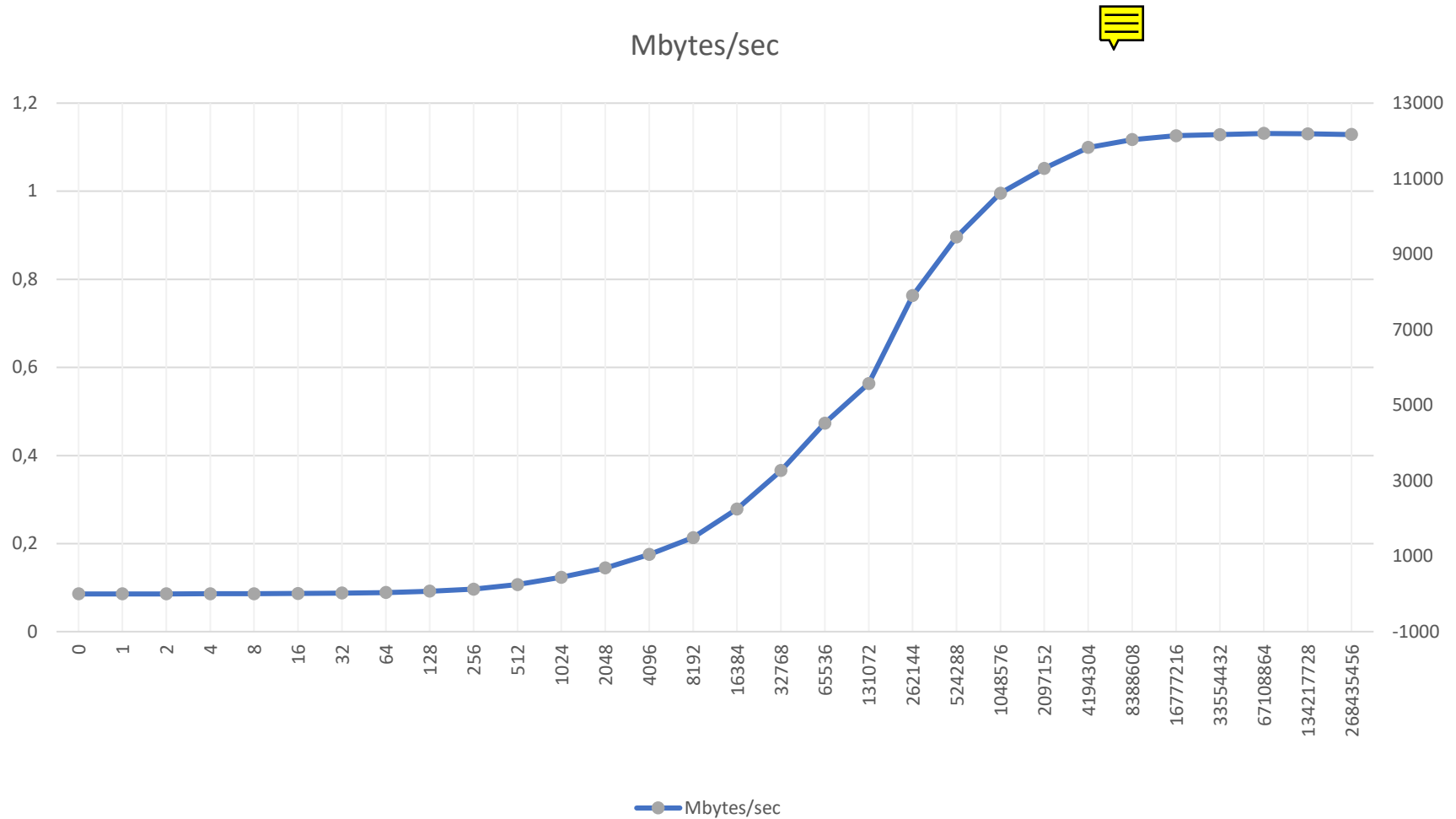
- Download Intel MPI benchmark
- Compile it
- Run it
- Get results and interpret it
- See README file in MPI directory..



# Measuring MPI point-to-point performance on Orfeo



# Measuring MPI performance on Orfeo



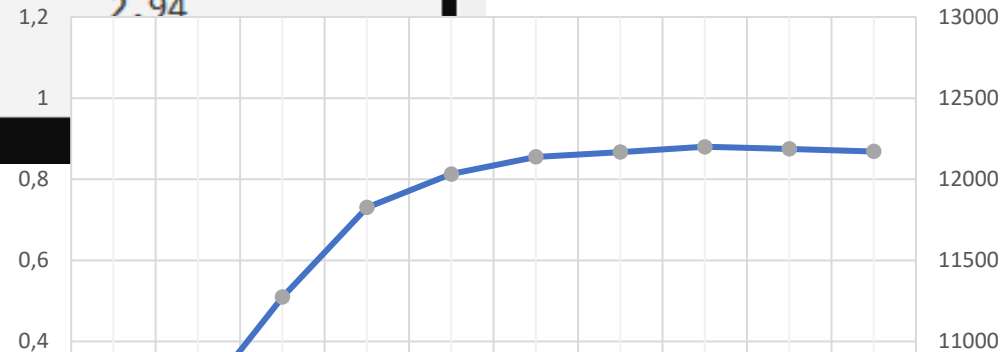
# Extrapolating values for internode communication

```
#-----
# Benchmarking PingPong
# #processes = 2
#-----
```

$\lambda$

#bytes	#repetitions	t[usec]	Mbytes/sec
0	1000	1.36	0.00
1	1000	1.36	0.74
2	1000	1.36	1.47
4	1000	1.36	2.94
8	1000	1.36	
16	1000	1.36	
32	1000	1.40	

Mbytes/sec



$b_{network}$

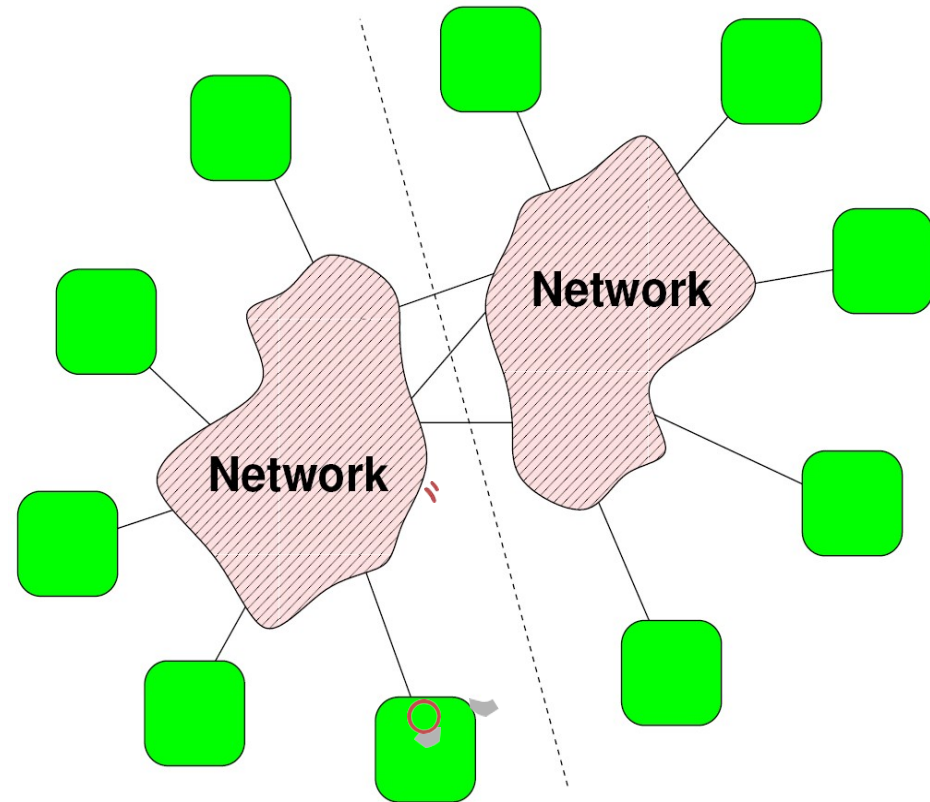
4194304	10	355.03	11813.78
8388608	5	697.14	12032.92
16777216	2	1382.15	12138.48
33554432	1	2757.62	12167.91
67108864	1	5500.71	12200.04
134217728	1	11012.36	12187.92
268435456	1	22054.87	12171.25

# Network topology

- How the components are connected.
- Important properties
  - **Diameter**: maximum distance between any two nodes in the network (hop count, or # of links).
  - **Nodal degree**: how many links connect to each node.
  - **Bisection bandwidth**: The smallest bandwidth between half of the nodes to another half of the nodes.
- A good topology: small diameter, small nodal degree, large bisection bandwidth

# Bisection bandwidth: $B_b$

- Split  $N$  nodes into two groups of  $N/2$  nodes such that the bandwidth between these two groups is minimum
- general metric for the data transfer “capability” of a system
- More meaningful metric in terms of system scalability:  $B_b / \text{Nodes}$





# Common Topologies in HPC

- Bus
- Crossbar switches
- Fat tree
- CBB (Constant Bi-sectional Bandwidth)
- Mesh
  - 3D torus

# Bus topology

- Bus can be used by one connection at a time
- Bandwidth is shared among all devices
- Bisection BW is constant:  $Bb/Nnodes \sim 1/Nnodes$

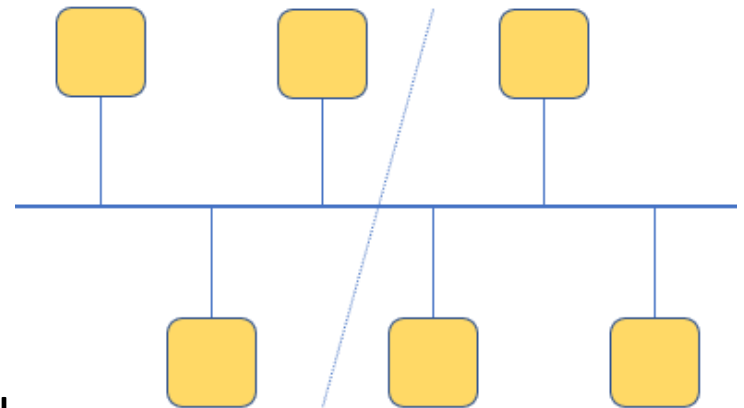
- Examples: PCI bus

- Advantages

- Low latency
- Easy to implement

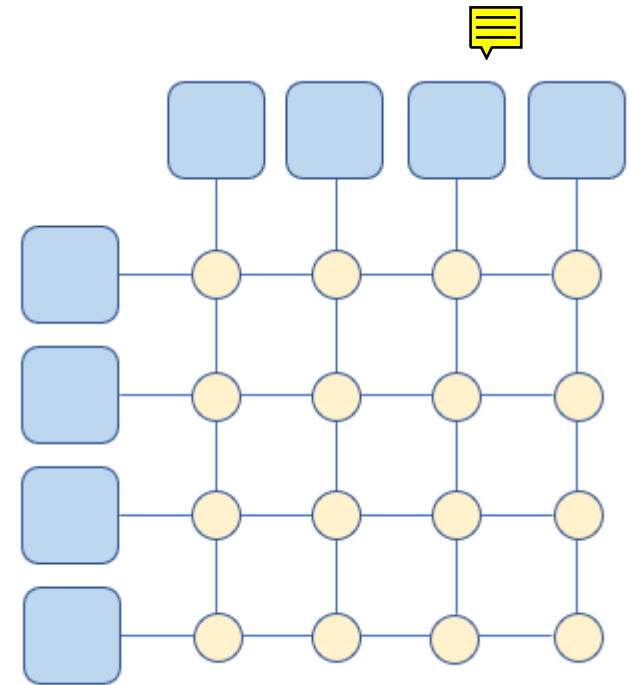
- Disadvantages

- Shared bandwidth, not scalable
- Problems with failure resiliency (one defective agent may block bus)



# Non blocking crossbar switch

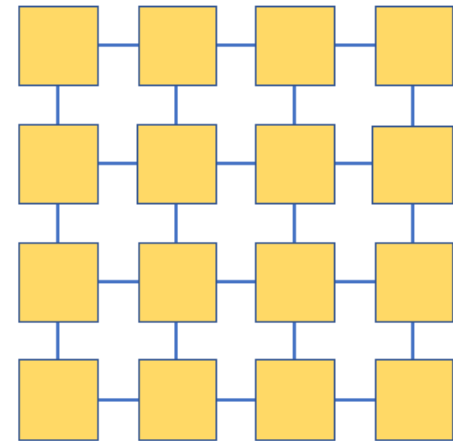
- A non-blocking crossbar can mediate a number of connections between a group of input and a group of output elements
- This can be used as a 4-port non-blocking switch
- Switches can be cascaded to form hierarchies (common case)
- Allows scalable communication at high hardware/energy costs
- Not feasible for large HPC installations



# Meshes

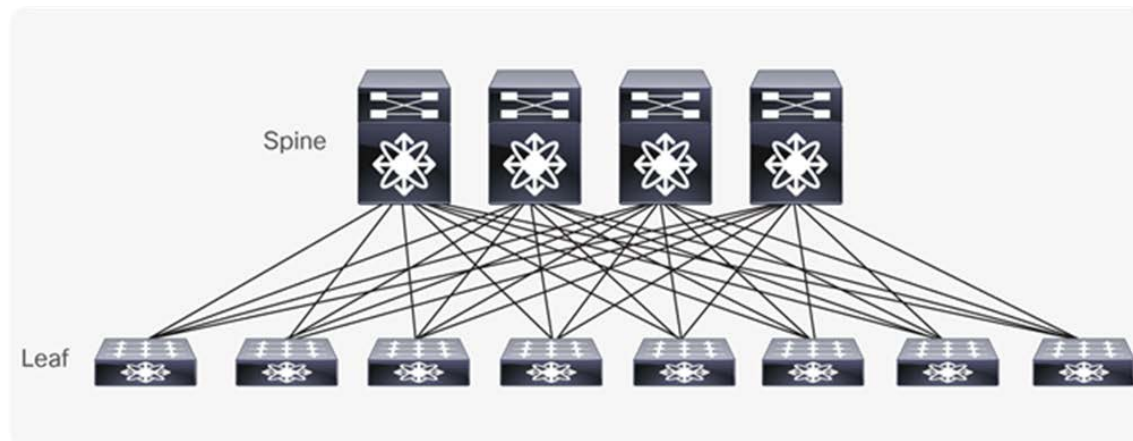


- Fat trees can become prohibitively expensive in large systems
- Compromise: Meshes
  - n-dimensional Hypercubes
  - Toruses (2D / 3D)
  - Many others (including hybrids)
- Each node is a “router”
- Direct connections only between direct neighbors
- Different from a crossbar!
- Intelligent resource management and routing algorithms are essential



# Switches and Fat-Trees


- HPC clusters are built with switched networks
- Compute nodes (“devices”) are split up in groups – each group is connected to single (non-blocking crossbar-) switch (“leaf/edge switches”)
- Leaf switches are connected with each other using an additional switch hierarchy (“spine switches”) or directly (for small configs.)
- “Perfect” world: fat- trees

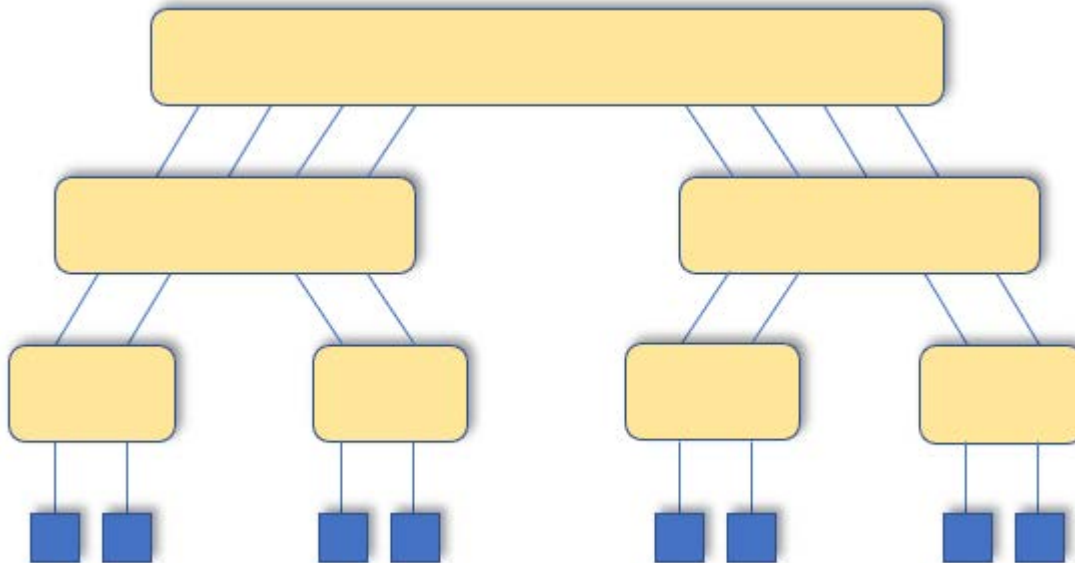






# Fat-trees switch hierarchy..

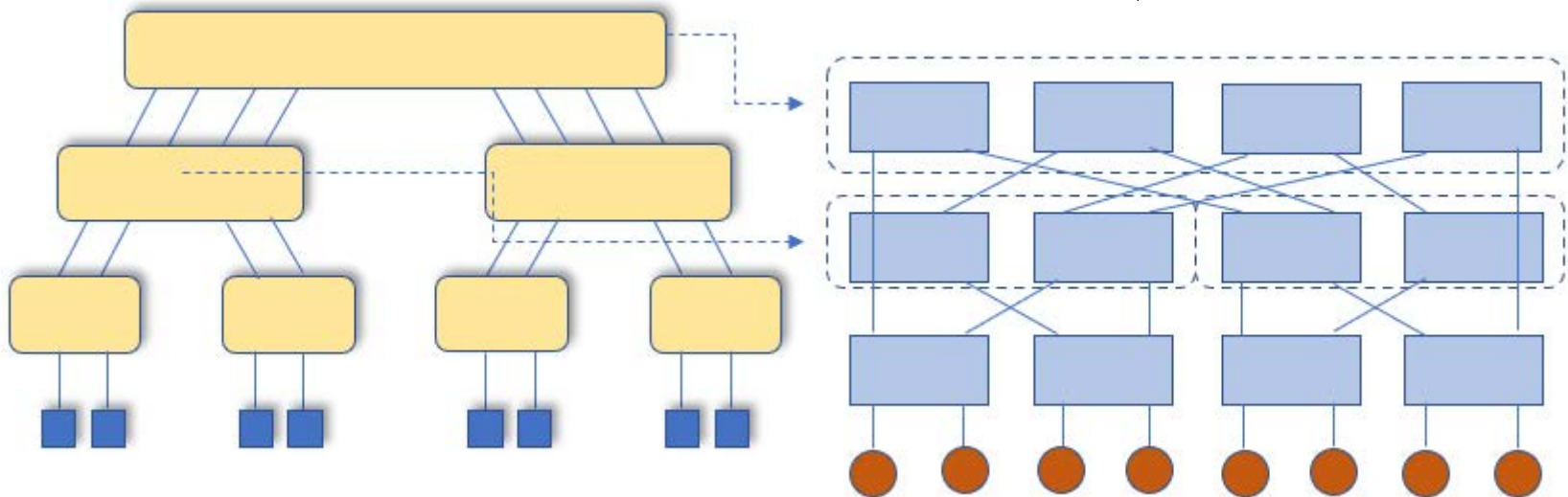
- Fully non-blocking: 
  - Each level double the number of link of the switches
  - Not practical. Root is NXN switch





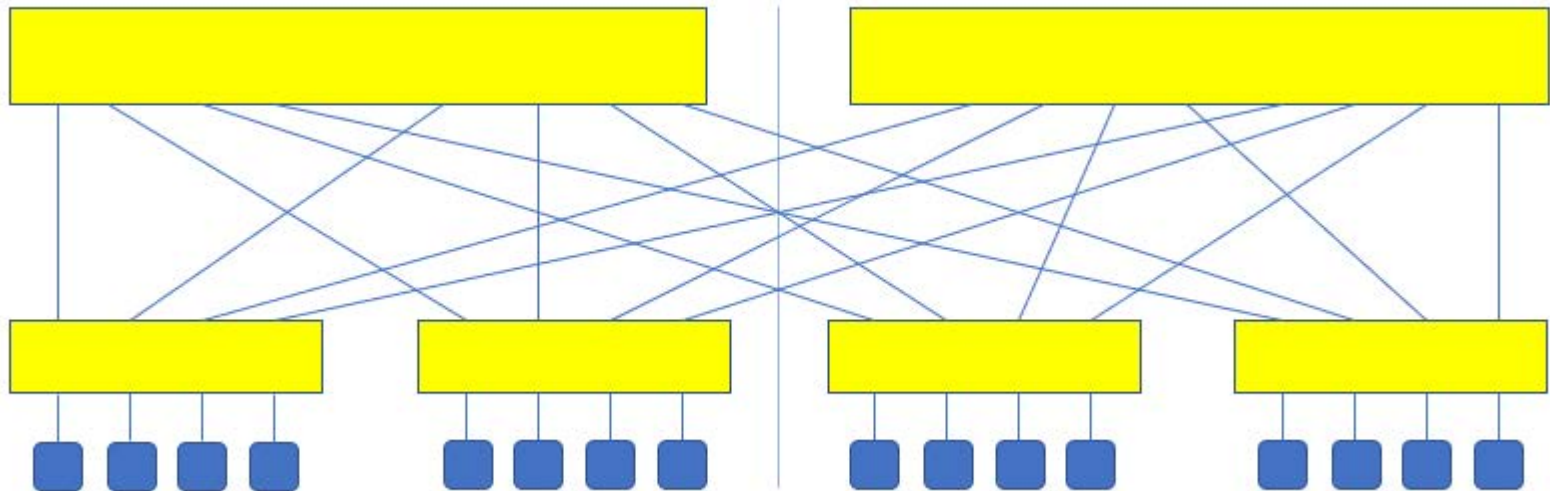
# Practical fat-tree implementation

- Use smaller switches to approximate large switches.
- Most commodity large clusters use this topology.
- Also call constant bisection bandwidth network (CBB)
  - $N_{\text{nodes}}/2$  end-to-end connections with full bandwidth
  - $B_b = B * N_{\text{nodes}}/2$
  - $B_b / N_{\text{nodes}} = \text{const.} = B/2$



# Two level CBB example


- $N_{\text{nodes}}/2$  end-to-end connections with full bandwidth: 8
- $B_b = B * N_{\text{nodes}}/2 = 8B$
- $B_b / N_{\text{nodes}} = \text{const.} = B/2$

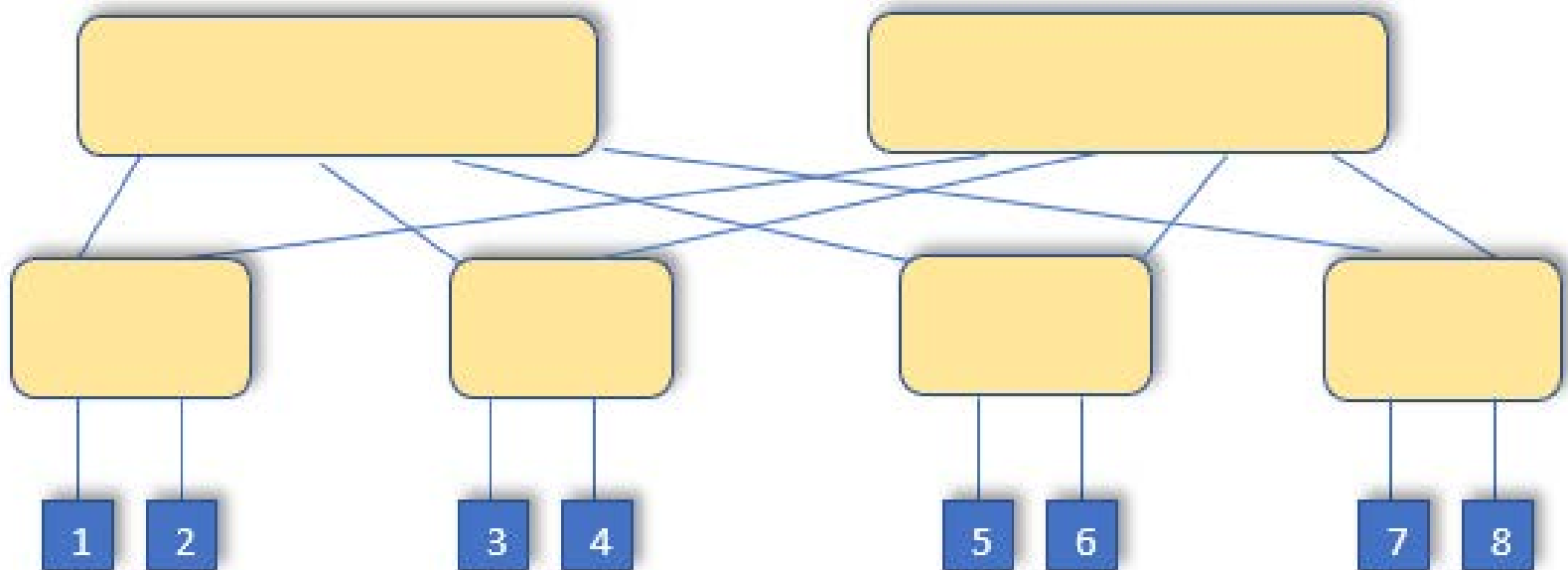


# Fat tree and static routing

- Generally, CBB are using static routing algorithm.
  - path taken between any two node pairs is statically computed
  - Full bandwidth is not always seen in practice.
  - The number of potential routes  $R$  for a total node count of  $N$ :  $R=N(N-1)=N^2-N$ .
  - Number of routes  $o(N^2)$
  - Number of Intermediate Spine link is  $o(N)$
- ➔ There are scenarios where certain host communications will use the same Intermediate Spine link

# Example

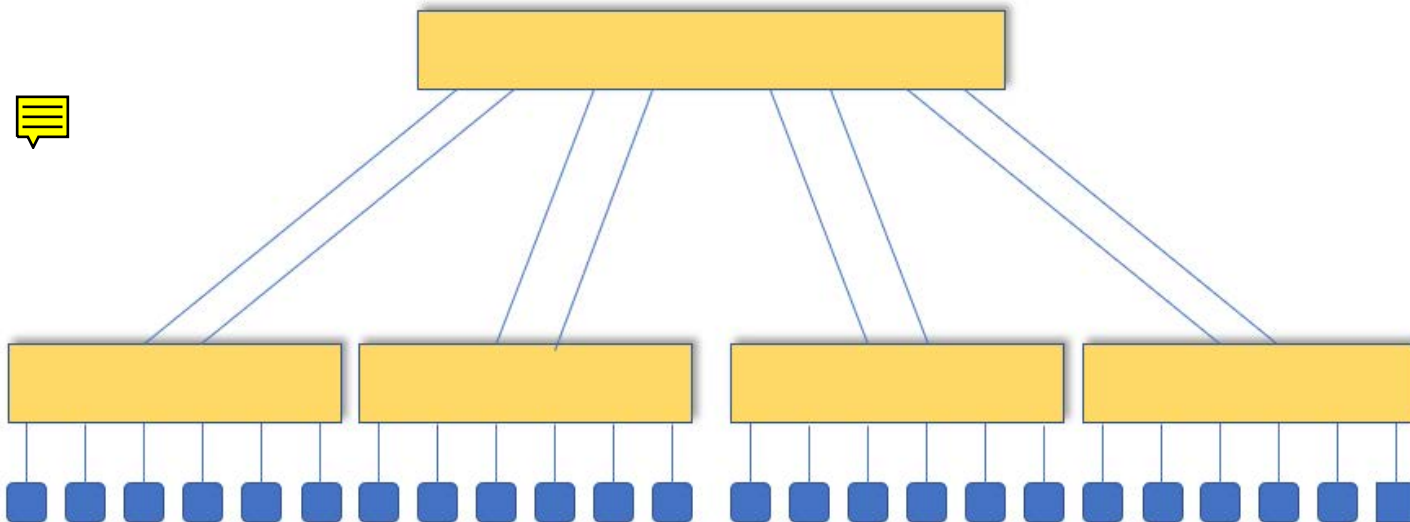
- For 1->5,2->6,3->7,4->8 is ok 
- For 1->5,2->7,3->6,4->8 is no longer fine if there is static routing





# Oversubscription

- Spine does not support  $N_{\text{nodes}}/2$  full BW end-to-end connections
- $B_b/N_{\text{nodes}} = \text{const.} = B/(2k)$ , with  $k$  oversubscription factor ( $k=3$  for the example)
- Resource management (job placement) is crucial





# High speed networks



- Infiniband
  - The de-facto standard
  - 27% of ToP500 are based on infiniband
- Omni Path
  - started by Intel in 2015
  - one of the youngest HPC interconnects
  - 8.6% of Top500 are Omni-Path systems
- Both are used behind a MPI implementation..




# Infiniband speed: physical layer..

- InfiniBand uses serial stream of bits for data transfer
- Linkwidth
  - 1x – One differential pair per Tx/Rx
  - 4x – Four differential pairs per Tx/Rx 
  - 12x - Twelve differential pairs per Tx and per Rx
- LinkSpeed
  - Single Data Rate (SDR) - 2.5Gb/s per lane (10Gb/s for 4x)
  - Double Data Rate (DDR) - 5Gb/s per lane (20Gb/s for 4x)
  - Quad Data Rate (QDR) - 10Gb/s per lane (40Gb/s for 4x)
  - Fourteen Data Rate (FDR) - 14Gb/s per lane (56Gb/s for 4x)
  - Enhanced Data rate (EDR) - 25Gb/s per lane (100Gb/s for 4x) 
- Linkrate
  - Multiplication of the link width and link speed
  - Most common shipping today is 4x ports DFR/EDR



# Infiniband speed: data encoding

- For SDR, DDR and QDR, links use 8b/10b encoding:
  - every 10 bits sent carry 8bits of data
- Thus single, double, and quad data rates carry 2, 4, or 8 Gbit/s useful data, respectively.
- For FDR and EDR, links use 64b/66b encoding
  - every 66 bits sent carry 64 bits of data.

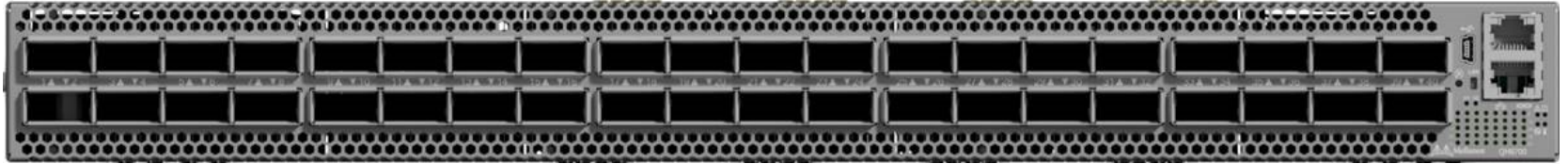


# InfiniBand performance

	<b>SDR</b>	<b>DDR</b>	<b>QDR</b>	<b>FDR</b>	<b>EDR</b>	<b>HDR</b>
Signaling rate (Gbit/s)	2.5	5	10	14.0625	25.78125	50
Encoding (bits)	8/10	8/10	8/10	64/66	64/66	64/66
Theoretical throughput 1x (Gbit/s)	2	4	8	13.64	25	50
Theoretical throughput 4x (Gbit/s)	8	16	32	54.54	100	200
Theoretical throughput 12x (Gbit/s)	24	48	96	163.64	300	600



# ORFEO IB network

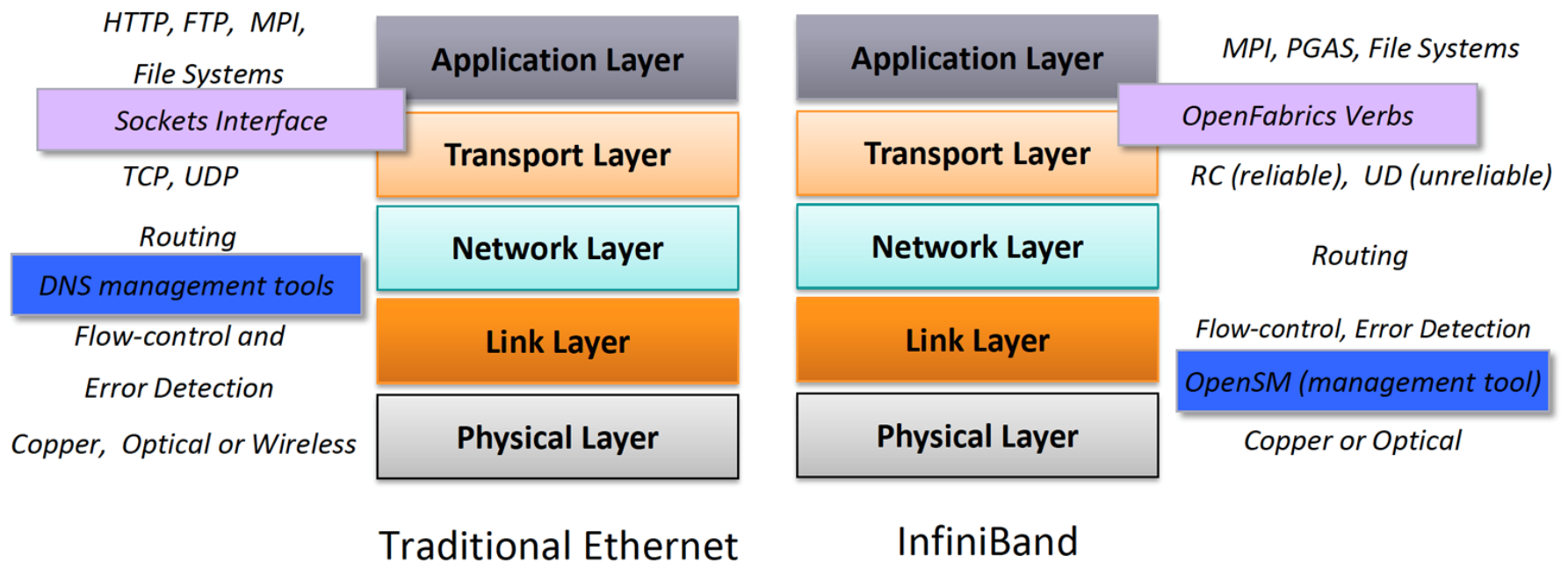


## Performance

- 40 x HDR 200Gb/s ports in a 1U switch
- 80 x HDR100 100Gb/s ports (using splitter cables)
- 16Tb/s aggregate switch throughput
- Sub-130ns switch latency



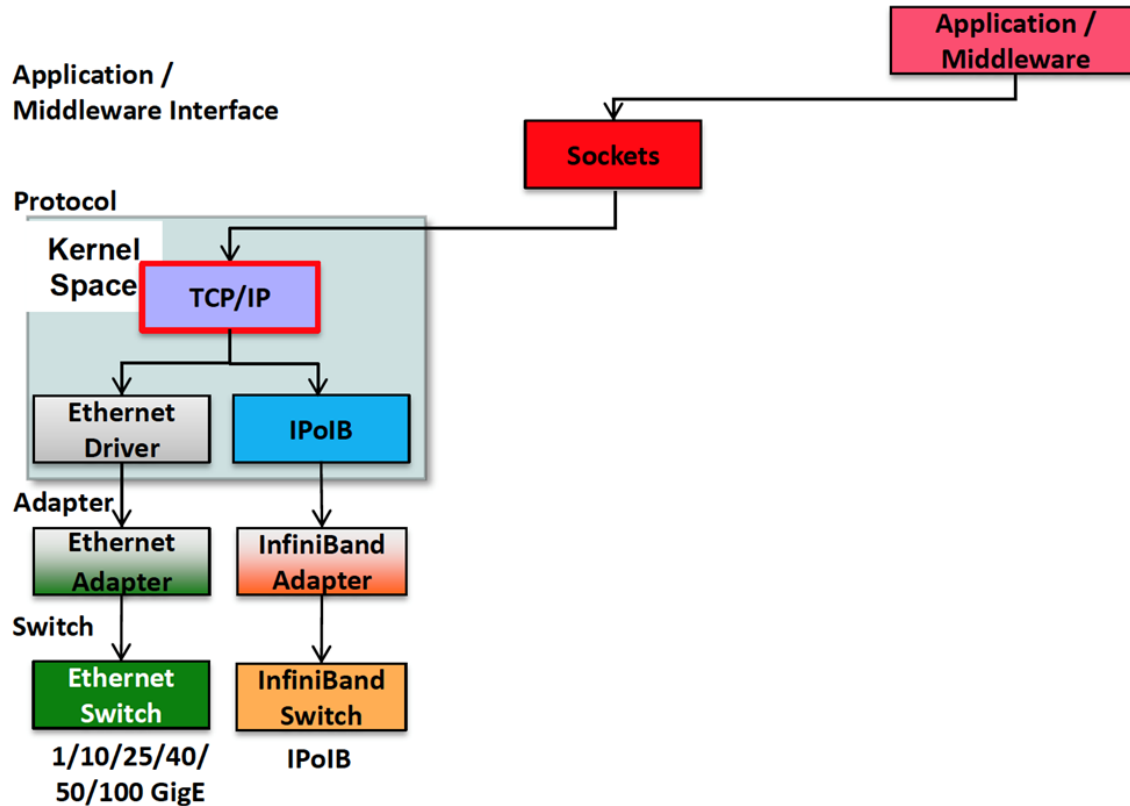
# Infiniband vs Ethernet..



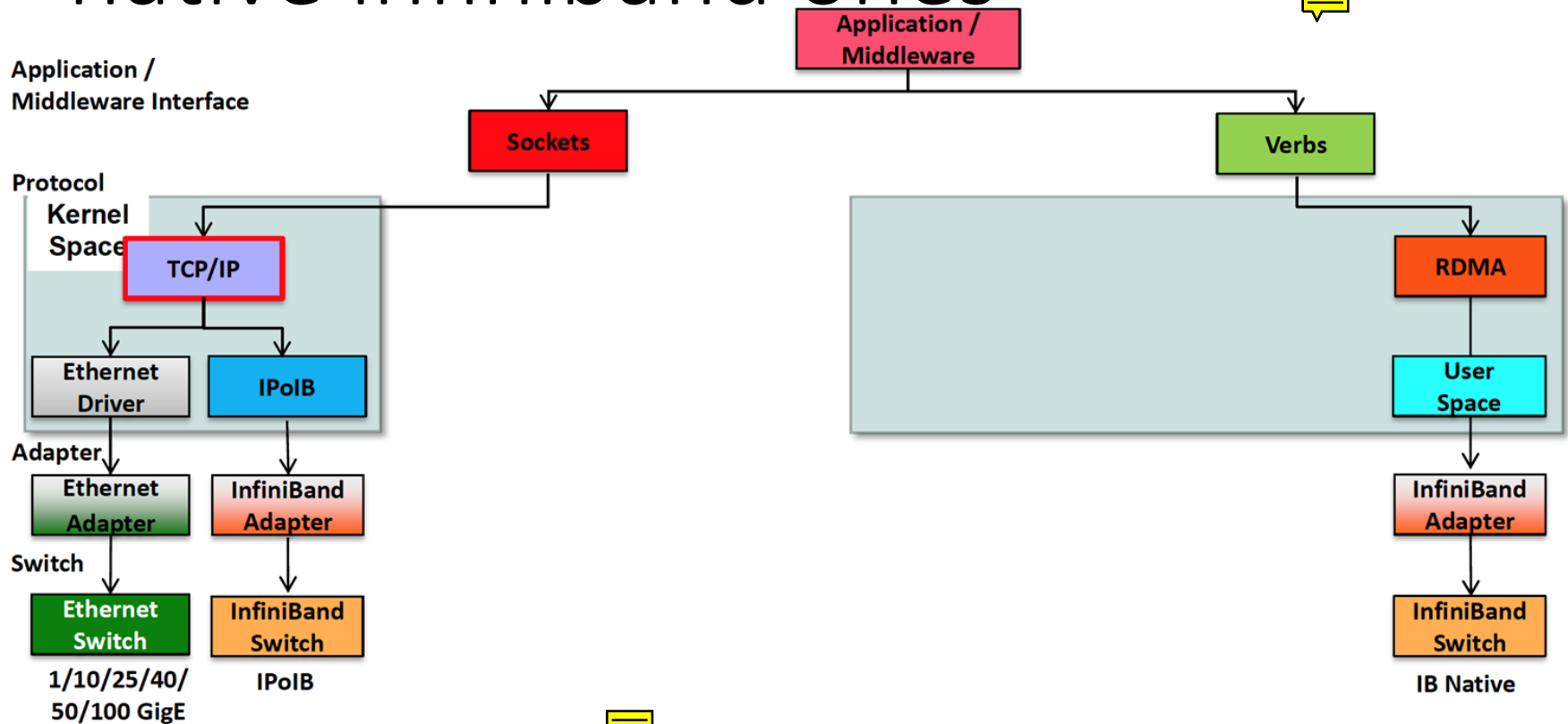




# TCP/IP and IPoIB protocol



# TCP/IP and IPoIB protocol vs native infiniband ones



# Our network: ORFEO ones..

- We can assume full non-blocking network:  
P/2 pair of nodes communicate in parallel at full speed

$$T_{comm} = \lambda + \text{message-size} / b_{network}$$

- Where

$\lambda = 1.35$  microsecond

$b_{network} = 12\text{Gb/second}$

# Exercise/Tutorial:

- Confirm the data of the previous slide for all the kind of nodes and all the kind of MPI libraries.