Interconnection Networks
A crash course

Eitan Zahavi
Oct 2010
Introduction and Motivation

- Clustering means communication
- The more CPUs the higher the communication
  - Each processor performs 1/N of the overall work
  - But each one consumes some I/O for input some for output and some for cross processor data
  - Need to keep scaling the I/O by 1/N even though more I/O is produced
  - If not: Amdal’s law predicts that I/O will become the bottleneck

- The technologies built to support these requirements are known as Interconnection Networks

Interconnection Networks Topics

- **Topologies**
  - How machines are connected?

- **Switching**
  - How packets are forwarded?

- **Deadlock, livelock and starvation**
  - What can go wrong?

- **Routing**
  - The good the bad and the ugly of packet forwarding

- **HoL Blocking / Congestion Control**
  - What hinders performance?

- **QoS**
  - Differentiating traffic
Network Topology Properties

- Direct vs. Indirect
  - Direct: No “intermediate” nodes – connections are only between end-nodes/Hosts
  - Indirect: Communication between each two nodes is done through a switch

- Node degree
  - Number of channels (uni-directional) connecting to neighbor

- Regularity
  - When all nodes have same degree

- Symmetry
  - When it looks alike from every node
Network Topology Properties (Cont)

- **Diameter**
  - Maximum distance between two nodes

- **Cross Bisectonal Bandwidth CBB**
  - The minimal total cross-section BW for every 2 equal size groups of end-nodes

- **Fault Tolerance**
  - Minimal number of links to disconnect a node from any other node

- **Path Diversity**
  - Minimal number of paths from node to node.
    - Fully/Partially/No Disjoint? Minimal?
More Terminology

- **Flow**: Traffic from a source to a destination

- **Hotspot**: When two flows pass through the same link – the hotspot is that link

- **Hotspot Degree**: The number of flows contending on specific hotspot

- **Hot-Module / Destination Congestion**: When two sources send to the same destination

- **Assuming traffic in pairs (2 of N permutation)**
  - No Hot-Modules possible
“Blocking” Terminology

- The topology ability to route arbitrary permutation with no hotspot

- Non-Blocking
  - Any input can connect to any free output without affecting other pairs
  - Routes are computed independently

- Rearrangeably Non-Blocking
  - Considering all the permutation pairs it can be routed with no hotspot
  - However this may require rearranging of the paths previous pairs are using

- Blocking
  - Some permutations can not be routed without hotspots
Crossbar

- Implementable for $N < 36..64$
- Indirect, Regular, Symmetric
- Single stage
- Diameter = 2
- Node Degree = $n$
- Non fault tolerant

The “speed-up” non-blocking criteria theorem:
- Assuming a buffer is placed in every cross section it is proven that
- If BW of the crossbar equals the incoming BW.
  - Only half the BW can be maintained
- IF BW of the crossbar $> 2\times$ incoming BW
  - The crossbar can maintain full BW
- Similar rules can be derived for more realistic crossbar architectures [3]

In further discussions we assume all crossbars are non-blocking

---

Full Graph - Clique

- N = number of nodes
- Implementable for N < 36..64
- Direct, Regular Symmetric
- Diameter = 1
- Node Degree = N-1
- Non-Blocking
- CBB / Fault tolerance
  - If no “pass-through” traffic (node are not switches)
    - No fault tolerance, CBB = N-1
  - If “pass-through” traffic is allowed
    - Fault tolerance = N-2; CBB = N(N-1)

**d-dimentional \((n_0,n_1\ldots)\)-size mesh**

- Connect each node to all its neighbors in each dimension
- Indirect, Regular, Asymmetric
- Diameter = \(n_0+n_1+n_2\ldots-d\)
- Degree = \(d\)
- Blocking if max\(\{n\}\) > 2
- CBB: \(\prod_{i=1}^{d-1} n_i \mid n_1 \leq n_2 \leq \ldots \leq n_d\)
- Fault Tolerance = \(d\)

\[
\begin{array}{c|c|c|c}
\text{d} & \text{Example} & \text{n} & \text{Example} \\
0 & \{\} & \{2\} & \{4,3,2\} \\
1 & \{2\} & \{2,3\} & \\
2 & \{2,3\} & & \\
3 & & & \\
\end{array}
\]
Hyper Cubes / d-dimensional Cube

- A sub-class of d-dimensional mesh where \( n = \{2, 2, \ldots\} \)
- Indirect, Regular, Symmetric
- Diameter = \( d-1 \)
- Degree = \( d \)
- Non-Blocking
- CBB = \( 2^{d-1} \)
- Fault Tolerance = \( d \)

k-ary n-cubes - Torus

- Closing each dimension in a loop
- Indirect, Regular, Symmetric
- Diameter = \((n_0 + n_1 + n_2 + \cdots - d) / 2\)
- Degree = \(d\)
- Blocking if \(\max \{n\} > 2\)
- CBB \(\prod_{i=1}^{d-1} n_i \mid n_1 \leq n_2 \leq \ldots \leq n_d\)
- Fault Tolerance = \(d\)
Multistage Interconnection Networks

- We discussed networks built with a single type of nodes
  - Full Graph – Clique
  - d-dimentional \((n_0,n_1\ldots)-size\)
  - Hyper Cubes / d-dimensional cube
  - Torus

- MINs are built out of two types of vertex
  - End-Nodes or Hosts
  - Switches (non-blocking)
  - The end-nodes connect to the edges of a network of switches
Butterfly k-ary n-fly

- $k^n$ network nodes
- Example:
  - 2-ary 3-fly
  - Routing from 000 to 010
- DST address used to directly route packet
  - Bit $n$ used to select output port at stage $n$
- Path Diversity = 1
- Diameter = $n$
- CBB = $k^n$
CLOS($m,n,r$)

- Hosts: $r \times n$ ; Switches: $2r \ (n \times m) + m \ (r \times r)$
- Blocking theorems
  - Rearrangebly Non-Blocking iff $m \geq n$
  - Non-Blocking iff $m \geq 2n-1$
- Path Diversity : $m$
Fat - Trees

- Fatter links as you go up
- CBB = N
- Indirect, Symmetric (hosts only), Irregular (port BW)
- Degree = K
- Non-Blocking
- Totally impractical to build

k-ary n-tree (n,k) and GFT(h,m,w)

- A folded CLOS; 2*k number of ports per switch
- Height of tree is \( n \) [4]
- Path Diversity is \( k^{(n-1)} \)
- \( CBB = \max k^n \) but lower ratios possible

XGFT ... PGFTs

- XGFT was presented at 1995 by Sabine at al.
  - Defined as $XGFT(h, m_1..m_h, w_1..w_h)$ where
    - $h$: height of “tree” (leafs at level 0, roots at h)
    - $m_i$: number children of switches at level $0 < i \leq h$
    - $w_i$: number parents of switches at level $i-1$ has

- $PGFT(h, m_1..m_h, w_1..w_h, p_1..p_h)$ \[5\]
  - $p_i$: number of parallel ports connecting switches in level $i$ and $i-1$
  - Supports many real life fat trees being used when non maximal topologies are built

---

Dragonfly Topology

- First introduced as a scalable topology using bi-partite as local group by Satoshi Matsuoka, [6]
- With the goal of reducing the number of long cables J. Kim, W.J. Dally, S. Scott, and D. Abts [7]
- Two levels
  - Global channels connecting virtual switches
  - Local channels connecting switches within a virtual switch
- Global Network:
  - Consists of $n$ groups

---

Dragonfly Topology

- Local Network:
  - Can use Full-Graph, Fat Tree, 1D/2D/3D Torus
  - Each of the $a$ switches has:
    - $p$ hosts
    - $h$ global connection
Generalized Hyper Cube
(a generalized Flattened Butterfly) [8]

- A hyper cube with different radix per axis

# Properties Table

<table>
<thead>
<tr>
<th>Topology</th>
<th>Cross Bar</th>
<th>Full Graph</th>
<th>Mesh</th>
<th>Hyper Cube</th>
<th>Torus</th>
<th>Butterfly</th>
<th>CLOS(m,n,r)</th>
<th>GFT</th>
<th>PGFT(h,M,W,P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbolic</td>
<td>CB(N)</td>
<td>FG(N)</td>
<td>d-n-mesh</td>
<td>HC(d)</td>
<td>k-ary n-cube</td>
<td>k-ary n-fly</td>
<td>CLOS(m,n,r)</td>
<td>k-ary n-tree</td>
<td>PGFT(h,M,W,P)</td>
</tr>
<tr>
<td>Type</td>
<td>Indirect</td>
<td>Indirect</td>
<td>Indirect</td>
<td>Indirect</td>
<td>Indirect</td>
<td>MIN</td>
<td>MIN</td>
<td>MIN</td>
<td>MIN</td>
</tr>
<tr>
<td>Regular</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Symmetric</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No*</td>
<td>No*</td>
<td>No*</td>
<td>No*</td>
<td>No*</td>
</tr>
<tr>
<td>End-Nodes</td>
<td>N</td>
<td>N</td>
<td>N0<em>N1</em>..Nd-1</td>
<td>2^d</td>
<td>N0<em>N1</em>..Nd-1</td>
<td>k^n</td>
<td>n^r</td>
<td>k^n</td>
<td>M1<em>M2</em>M3...Mh</td>
</tr>
<tr>
<td>Switches</td>
<td>1</td>
<td>0</td>
<td>Nodes are SW</td>
<td>Nodes are SW</td>
<td>Nodes are SW</td>
<td>n^k^(n-1)</td>
<td>2^r+m</td>
<td>n^k^(n-1)</td>
<td>???</td>
</tr>
<tr>
<td>Max Degree (K)</td>
<td>N</td>
<td>N-1</td>
<td>2*^d</td>
<td>d</td>
<td>2*^d</td>
<td>k</td>
<td>min(n+m, 2r)</td>
<td>2k</td>
<td>max(Wi<em>Pi+Mi-1</em>Pi-1)</td>
</tr>
<tr>
<td>Max Diameter</td>
<td>2</td>
<td>1</td>
<td>(N0+N1+..+Nd-1)-d</td>
<td>d-1</td>
<td>floor(N0/2)+...+floor(Nd-1/2)</td>
<td>n+1</td>
<td>4</td>
<td>2n</td>
<td>2h</td>
</tr>
<tr>
<td>Min CBB</td>
<td>NA</td>
<td>N^2/4</td>
<td>N0*N1..Nd-2 : N0&lt;N1&lt;N2</td>
<td>2^(d-1)</td>
<td>N0*N1..Nd-2 : N0&lt;N1&lt;N2</td>
<td>k^n</td>
<td>r^m</td>
<td>k^n</td>
<td>???</td>
</tr>
<tr>
<td>Min Paths</td>
<td>1</td>
<td>1</td>
<td>Minimal: ~d</td>
<td>d</td>
<td>Minimal: ~d</td>
<td>1</td>
<td>m</td>
<td>k^(n-1)</td>
<td>???</td>
</tr>
<tr>
<td>Fault Tolerance</td>
<td>1</td>
<td>1</td>
<td>d-1</td>
<td>d-1</td>
<td>2d-1</td>
<td>0</td>
<td>0/m-1</td>
<td>0/k-1</td>
<td>min(Wi<em>Pi+Mi-1</em>Pi-1)-1</td>
</tr>
<tr>
<td>Non-Blocking</td>
<td>Strictly</td>
<td>Strictly</td>
<td>No</td>
<td>Rearrangibly</td>
<td>Rearrangibly</td>
<td>No</td>
<td>m&gt;=n</td>
<td>m&gt;=2n-1</td>
<td>Rearrangibly</td>
</tr>
</tbody>
</table>

Homework: Fill in the table for Generalized Hyper-Cubes
Google published a paper about FBFLY …

- “Energy Proportional Datacenter Networks” (ISCA10)

<table>
<thead>
<tr>
<th>Metric</th>
<th>PGFT</th>
<th>3D Torus</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBB</td>
<td>From 1 and down</td>
<td>1/N</td>
</tr>
<tr>
<td>Applications</td>
<td>Arbitrary</td>
<td>3D/2D only</td>
</tr>
<tr>
<td>Interconnect Technology</td>
<td>Mostly InfiniBand™</td>
<td>Mostly Proprietary</td>
</tr>
<tr>
<td>Mechanical and Cabling Complexity</td>
<td>Bi-Partite connections</td>
<td>Nice Cubes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No long cables</td>
</tr>
<tr>
<td>Built by</td>
<td>Mellanox, QLogic, Voltaire, IBM, HP, Fujitsu, Hitachi, Dawning …</td>
<td>IBM, Cray, Mellanox,…</td>
</tr>
</tbody>
</table>
Interconnection Networks Topics

- Topologies
  - How machines are connected?

- Switching
  - How packets are forwarded?

- Deadlock, livelock and starvation
  - What can go wrong?

- Routing
  - The good the bad and the ugly of packet forwarding

- HoL Blocking / Congestion Control
  - What hinders performance?

- QoS
  - Differentiating traffic
Switching

- Switching Techniques at a glance
  - Circuit switching – when a src to dst path is setup before communication of complete message
  - Packet switching - the complete packet is received at each switch before being forward to the next (store and forward)
  - Virtual Cut Through – packet can start transmission to next switch after header is processed
  - Wormhole Switching – packets are not buffered within the network. The head passes through and the body follows

- Most Interconnect Networks apply VCT switching
  - The reason is that VCT provide both short latency compared to Packet switching and avoids much of the blocking of Wormhole
Switch Arbitration

- At the heart of the crossbar stands an arbiter
- The issue of how to architect the switch and optimize the arbitration is covered by many researchers
- Many heuristics for best “matching” were proposed
- Some even propose central buffer pool

EE Course: Fast Router Design 049045
Interconnection Networks Topics

- Topologies
  - How machines are connected?

- Switching
  - How packets are forwarded?

- Deadlock, livelock and starvation
  - What can go wrong?

- Routing
  - The good the bad and the ugly of packet forwarding

- HoL Blocking / Congestion Control
  - What hinders performance?

- QoS
  - Differentiating traffic
Lossy vs. Loss-less Network

- Interconnect Networks are mostly Loss-less

Why?
- Lossy networks require end-to-end transport that is able to retransmit lost data – like TCP – these take CPU resources
- If packet drop is not negligible significant bandwidth may be lost on retransmissions
- To avoid drop – large costly buffers are used which introduce latency as they fill up
Link Level Flow Control

- Interconnect is made loss-less by assuring there is enough buffering to receive all sent packets
- LLFC is a signaling or counting method which provides the transmitter the information about the receiver buffers
- Alternatively the receiver may send the transmitter ON/OFF commands
- The minimal amount of buffering required for VCT
  - A complete message (largest allowed message size)
  - Equivalent buffers to the time it takes for updates to reach the transmitter
  - The amount of data stored on the transmitter to receiver wire
InfiniBand™ LLFC

- A method to guarantee LLFC under some BER
  - FibreChannel which is lossless will get stuck on that
  - Both transmitter and receiver count data sent/received
  - Receiver updates transmitter what count it can reach
  - Transmitter updates receiver how much was sent – to handle BER

- Architecture Registers:
  - Flow Control Total Blocks Sent (FCTBS):
    - TX tracks the total number of sent flits
    - Periodically the TX provides an update to RX (to overcome BER)
  - Adjusted Blocks Received (ABR):
    - RX Tracks total number of received flits
    - It also updates this number periodically
  - Flow Control Credit Limit (FCCL):
    - Periodically RX provides the TX
    - FCCL = ABR + <available buffer space>
IEEE 802 Data Center Bridging (DCB) 802.1Qbb

Credit Loops? What are these?

- If traffic to DST-1 waits on traffic for DST-2
- And traffic to DST-2 waits on traffic for DST-3
- And traffic to DST-3 waits on traffic for DST-1
- We have a dependency loop and the fabric deadlocks
Credit loop is forms a loop on the channel connection graph

1. Name channels on the network “link” level topology
Credit Loop Check: Step 2

Channel Connection Graph: Built by going from each source to each dest while connecting the channels on the path.

Channel Naming

directed graph loop detection algorithm
Up/Down Routing avoids Credit Loops

- Given a topology: nodes and links
- Rank the nodes according to some levels
- Provide "direction" to links pointing towards the higher rank
- To prevent cycles in the channel graph it is enough to prevent one of the 4 possible "turns":
  - From an up link to down link
  - From an up link to up link
  - From a down to down
  - From a down to up ← this is the illegal turn
- Shown to create congestion near the "roots" on the tree

Spare Buffers avoids Credit-Loops

- If there is a loop a deadlock can be prevented
  - Differentiate Traffic
    - Looping traffic – already on the loop
    - External traffic – joining the loop
  - Avoid arbitrating external traffic unless there are 2 MTU worth of credits
- This eliminates the locking of the loop as there are always at least one MTU credits
- To guarantee fairness need a special mechanism to coordinate which external input is granted
Introducing Virtual Lanes

- Some topologies can not be routed with minimal hops without introducing credit-loops
- Independent buffer-sets are used to resolve this issue
- For example:
  - Paths crossing 6 use VL=1, the rest VL=0

Livelock – or – Routing Loop

Routing in a loop will cause packet duplication and lock the network

The simplest solution:

- Spanning Tree – or Multiple Spanning Trees

A multi-path solution: *minimal-path routing*

- Each hop get closer to the destination
- Does not create loops by definition
- Does not mean a single path!
Interconnection Networks Topics

- **Topologies**
  - How machines are connected?

- **Switching**
  - How packets are forwarded?

- **Deadlock, livelock and starvation**
  - What can go wrong?

- **Routing**
  - The good the bad and the ugly of packet forwarding

- **HoL Blocking / Congestion Control**
  - What hinders performance?

- **QoS**
  - Differentiating traffic
Routing

- Deterministic
  - Where the routing does not change with time or condition of the network
  - **PRO:** Traffic is always in-order, credit loops freedom
  - **CON:** There is always a counter permutation [9]

- Oblivious
  - Where routing may change but without any knowledge of the interconnect condition
  - **PRO:** does better for the worst permutation
  - **CON:** OOO is guaranteed

- Adaptive
  - Routing adapts to some metrics of the interconnect performance

---

Deterministic Routing Mechanisms

- **For InfiniBand™**
  - Each switch carry OutPort(Dest) table
  - A centralized engine configures all switches

- **For Data Center Bridging DCB**
  - Transparent Interconnection of Lots of Links (TRILL)
    - Incoming RBridge encapsulates traffic
    - Forward to destination RBridge (using a table)
    - Can send over standard Eth
    - Can support multi-path via multiple addresses

- **Spanning Tree Protocol (STP)**
  - Used in standard Eth equipment
  - Distributed Algorithmic and thus VERY deterministic

Deterministic for Orthogonal Topologies

- Orthogonality means
  - All nodes can be arranged in orthogonal d-dimensional space such that every link produces displacement in one dimension
  - Mesh, Torus, Generalized Hyper Cubes

- Dimension Order Routing DOR
  - First move in first dimension then the second, etc
  - Algorithmic (can be implemented efficiently)
  - Credit-loop-freedom
  - Requires at least 2 VL’s

- Fault Tolerance
  - Through use of intermediate step [10]
  - Select common reachable node
  - Route DOR to and from that node
  - To avoid credit loops change VL going through the node
  - Segment Based [11]
  - Change the dimensions order

Deterministic for Fat-Trees

- No credit loops possible if topology is Fat-Tree
  - Up/Down by definition (once split do not re-converge)
- FT is rearrangeably non-blocking
  - Some permutations can be routed with no hot-spot
  - But which permutation count? Not a routing question
  - \( D \mod K : \text{at level } l \text{ select the parent floor}(D/K^{l-1}) \mod K \)
    - Strictly non-blocking for common MPI communicators [12]
  - \( S \mod K : \text{at level } l \text{ select the parent floor}(S/K^{l-1}) \mod K \)
- Fault Tolerance
  - Technologies like IB provide automatic path migration
  - FT have superior path diversity
  - No risk for credit-loops with migration

Deterministic for Irregular Networks

- What happens when random topologies are used?
- How to preserve credit-loop freedom
  - Enhancing Up/Down to restrict some turns
    - Find a clever way to restrict “turn” space
  - Layered Routing [13]
    - Use minimal routes
    - Try existing layers or create a new one
  - Routing when multiple roots exist with fault tolerance [14]
    - Based on up/down idea when multiple roots can be identified
    - Use layers (VLs) when needed – across roots

Oblivious Routing

- Per Packet/Flit/Cell Random
  - When order of packet delivery does not matter it is always possible to spread traffic across all possible paths
  - Good as long as there is no hot-module or blocks the entire interconnect

- Valiant
  - Select random intermediate – path through node
  - Spreads traffic when deterministic creates hotspots with high degree
  - The random selection falls behind a more tailored one

Adaptive Routing

- The higher the knowledge about the full state of the network the higher the effectiveness

- Methods differentiate on
  - Trigger for adaptation
    - On backpressure
    - On a cue from downstream switches [17]
  - Method for selecting the new output port
    - Random, Greedy (best), GreedyRandom[16], RandomGreedy

- AR always provide packets Out of Order [18]
  - Can be left for the software to re-order
  - Re-order time is not bounded – May be limited if N on the wire

Adaptive Routing Credit Loops

- A sufficient condition to avoid deadlock with AR
  - If output port is defined only during arbitration
  - If deterministic routing is guaranteed
  - If fallback to deterministic is triggered by back-pressure
  - AR can not deadlock
Interconnection Networks Topics

- Topologies
  - How machines are connected?

- Switching
  - How packets are forwarded?

- Deadlock, livelock and starvation
  - What can go wrong?

- Routing
  - The good the bad and the ugly of packet forwarding

- HoL Blocking / Congestion Control
  - What hinders performance?

- QoS
  - Differentiating traffic
HoL Blocking / Congestion Control

- Hotspots or Hot-Modules create back-pressure
- Once a switch buffers fills LLFC stops the output of the driving switch
- The speed of congestion build up is
  - \( \frac{\text{Switch-Buffer-Size[B]}}{(\text{Fill Rate} - \text{Drain Rate [B/s])}} \) [sw/s]
- Once that back-pressure reach the sources they are throttled
- So what is the problem?

- Head of Line Blocking
  - The filled buffers can not accept traffic
  - No matter what is it’s destination
First Experiences with Congestion Control in InfiniBand Hardware

Ernst Gunnar Gran
Simula Research Laboratory

Quantized Congestion Notification QCN 802.1Qau

- **Congestion Detection**
  - Variable random sampling of frames
  - For each sampled frame calc feedback value Fb:
    - \( F_b = -(Q_{off} + wQ_{\Delta}) \)
      - \( Q_{off} = Q_{depth} - Q_{\text{equilibrium}} \)
      - \( Q_{\Delta} = Q_{depth} - Q_{prev-depth} \)
  - CN is generated if \( F_b < 0 \) and sent to source
    - Carry congestion point ID and Fb

- **Source Reaction**
  - Reduction in transmission rate proportional to Fb
  - Recovery by timer from last CN in 3 phases:
    - Binary, Linear, Hyper-Linear

- **Extensions**
  - Notification duplication and spread

---

*Figure 3.* QCN load sensor mechanism algorithm. The sampling rate \( P_s \) is a function of the measured feedback \( F_b \). For low congestion levels 1 frame is sampled for every 100 received frames. The sampling rate increases linearly with the feedback up to 1 sampled frame for every 10 received frames.
CC Homework

- Analyze the IB and DCB CC
  - Can the system reach a steady state?
    - When there are 2 flows contending
    - When there are N flows contending
      - Hint: What is required from the mark-rate and CCTI Timer values?
RECN An Alternate Approach

- IB CC is removing Congestion but the problem is HoL
- Why can’t we just remove HoL and Keep the tree?
- Use “Set Aside Buffers” SAB for congested flows
- Upstream switch is notified only if downstream SAB filled
- Upstream switch knows it is sending to a congested tree
- Create and store packets in a new allocated SAB
- Since traffic contributing to congestion is not filling the regular buffers, HoL blocking is avoided

- Was implemented in Advanced Switching product

Interconnection Networks Topics

- **Topologies**
  - How machines are connected?

- **Switching**
  - How packets are forwarded?

- **Deadlock, livelock and starvation**
  - What can go wrong?

- **Routing**
  - The good the bad and the ugly of packet forwarding

- **HoL Blocking / Congestion Control**
  - What hinders performance?

- **QoS**
  - Differentiating traffic
QoS – Quality of Service

- Differentiate the BW and Latency provided to different traffic classes
  - VoIP, Video, Ack’s …
- It is assumed that one traffic class can not be stuck behind other class traffic
- So Buffering (LLFC) must be different for each class
- Required N classes
  - But M VL’s required for credit-loop avoidance
  - Need at least M*N VL’s
QoS on Loss-less networks

- On Lossy networks
  - Each hop can drop packets
  - Each hop arbitration imposes its own BW ratio etc
  - On N levels the obtained ratio is a complex function

- On Loss-Less networks
  - Once there is back pressure a single arbiter defines the ratios...
Enhanced Transmission Selection (ETS)

802.1Qaz

- An ETS-compliant bridge shall support at least three ETS traffic classes.
- The actual ETS algorithm is not specified in the 802.1Qaz draft.
- ETS queues are low-priority queues: packets are transmitted from the ETS queues only if there’s no traffic available for transmission in credit-based shaping or strict priority queues (usually the credit-based shaping queues have higher priority than the strict priority queues).
- Weighted round robin definitely meets the requirement ... but it’s not hard to meet them when the draft requires bandwidth granularity of 1% and +/-10% accuracy.
InfiniBand™ VLs/SLs

- VL : Virtual Lane
- SL: Service Level
- At max 16 VLs
- Packets carry SL in their headers
  - Can not change (protected by CRC)
  - There are 16 SL’s
- OutVL = ProgramableSL2VL(SL, InPort, OutPort)
- Each VL has its own set of buffer and flow-control
- Arbitration: Two Priority Levels
  - Non-deficit slotted arbiter
Interconnection Networks Topics

- Topologies
  - How machines are connected?

- Switching
  - How packets are forwarded?

- Deadlock, livelock and starvation
  - What can go wrong?

- Routing
  - The good the bad and the ugly of packet forwarding

- HoL Blocking / Congestion Control
  - What hinders performance?

- QoS
  - Differentiating traffic
NoC Topics

- **Topologies**
  - Mesh and beyond

- **Switching**
  - Wormhole

- **Deadlock, livelock and starvation**
  - Applies greatly

- **Routing**
  - XY, XY/YX

- **HoL Blocking / Congestion Control**
  - Even worse

- **QoS**
  - Many services, IO/Mem/CPU-to-CPU
BACKUP
CLOS – Rearranging Proof

Proof: By induction
Clos(1, 1, r) – you have r boxes, each box is 1 x 1

This is a crossbar, which we know is rearrangeable.
Assume that for the case Clos(n-1, n-1, r), n>=2, the statement is true. For the case Clos(n, n, r), we use the first switch in the middle to reduce the requirement to Clos(n-1, n-1, r).
Because \( n \) inputs, \( n \) outputs, we can always find a perfect matching. If we take out a middle box, and now have \((n-1)\) inputs, \((n-1)\) outputs.
CLOS – Strictly non-blocking

Clos Network is non-blocking in strict sense when $m \geq 2n-1$.

$n \times (2n-1)$

Each box has 2n-1 output pins

Each box has 2n-1 input pins
CLOS – Non-blocking – Proof

Proof by contradiction

From i to j, we cannot make connection, e.g. from 1 to 2, we cannot make connection.

Only time we can’t make a connection is if all paths are taken.

Input i has taken n-1 signals, output j has taken n-1 signals. Thus, at most 2n-2 paths are taken.

However, we have 2n-1 boxes for 2n-1 distinct paths between i and j. So we will always have at least one path to go through.
At most \( n-1 \) boxes taken from 1, and \( n-1 \) boxes taken from 2, so \( 2n-2 \) boxes are taken.

\[ n \times (2n-1) \]

\[ n \]

\[ 1 \]

\[ n \]

\[ 2 \]

\[ n \]

\[ \ldots \]

\[ \ldots \]

\[ n \]

\[ r \]

\[ n \]

\[ 2n-1 \]

\[ n-1 \]

\[ \ldots \]

\[ \ldots \]

\[ n \]

\[ r \]
XGFT Recursive Structure
Formalism: XGFT Recursive Definition

$$\lambda_h := \prod_{j=1}^{h} w_j = \text{number of top nodes in tree of height } h$$

$$\Gamma_l^h := \prod_{j=l+1}^{h} m_j = \text{number of sub-trees of height } l \text{ in a tree of height } h$$

$$\lambda_l^h := \lambda_l \Gamma_l^h = \prod_{j=1}^{l} w_j \prod_{j=l+1}^{h} m_j = \text{number of nodes in level } l \text{ of a tree of height } h$$

$$V_h := \{(l, i) \mid 0 \leq l \leq h \land 0 \leq i < \lambda_l^h\}$$

$$E_h^j := \{(l, a), (l + 1, b)\} \mid 0 \leq l < h \land (l, a), (l, b) \in V_h^j \land \{(l, a - j\lambda_h), (l + 1, b - j\lambda_h)\} \in E_h$$

$$E_{h+1} := \bigcup_{j=0}^{m_{h+1}} E_h^j \bigcup \left\{(h, a), (h + 1, b)\right\} \mid a \% \lambda_h = \left\lfloor \frac{b}{w_{h+1}} \right\rfloor$$

$$\text{Num Leaves} = \prod_{i=1}^{h} m_i$$

$$\text{In level 1} = w_1 \prod_{i=2}^{h} m_i$$

$$|V_h| = \sum_{i=0}^{h} \prod_{j=i+1}^{h} m_j \prod_{j=1}^{i} w_j$$