## Lecture 7a – Communication Costs

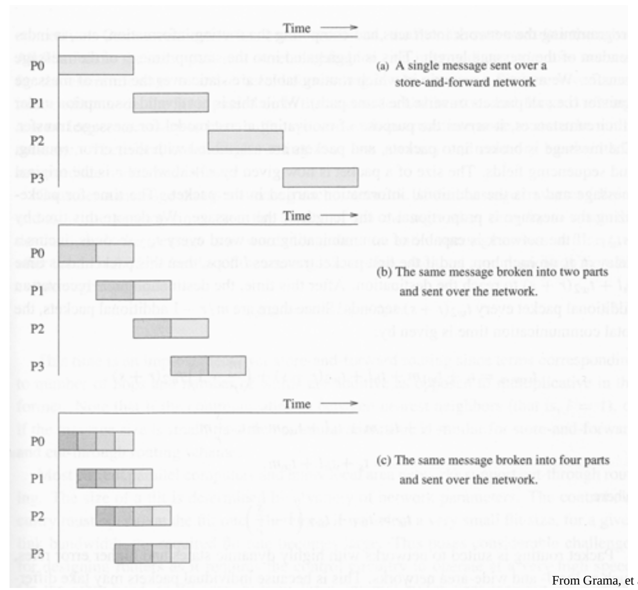
- **Communication of information between processing elements is one of the major overheads related to parallel computing on distributed-memory systems**
- **Cost factors include:**
  - Startup time, $t_s$, is the time required to handle a message at the sending and receiving nodes
  - Per-hop time, $t_h$, is the time it takes the header of a message to travel between two directly-connected nodes. This is also known as node latency
  - Per-word transfer time, $t_w$, is the average amount of time that a message of size m takes to traverse l links (or hops)

$$time_{communication} = t_s + (mt_w + t_h)l \quad \text{for store-and-forward routing}$$
$$= t_s + lt_h + mt_w \quad \text{for cut-through routing}$$

$t_h$ is generally considered to be small compared to $t_s$ and $t_w$



## Routing Strategies




From Grama, et al



## Communication Costs

- **So in order to optimize the cost of message transfers, we need to**
  - Communicate in bulk: aggregate a number of small messages into a single large message to reduce the effect of $t_s$
  - Minimize the volume of data: reduce the amount of data that is being passed
  - Minimize the distance of data transfer: minimize the number of hops, l, that a message must traverse
- **The first and second of these involves programming strategy and techniques**
- **The third involves the inter-connection of the processing elements**



## Routing

- **Efficient algorithms to route messages to processors are critical to achieve good parallel performance**
- **Routing mechanisms:**
  - Minimal: always select the shortest path. Provides the minimum $t_h$ but can lead to congestion
  - Non-minimal: can route messages along longer (than the shortest) paths to avoid congestion
  - Deterministic: a unique path based upon the source and destination
  - Adaptive: a path based upon the current status of the network and selects a path that avoids congestion

## Mapping Techniques to Determine Inter-Connections

- **Mapping techniques are used to determine optimal processor inter-connections and predict the efficiency of networks**
- **Binary Reflected Gray Code (BRG): G(i,d) denotes the i-th entry in a sequence of Gray codes of d bits. G(i,d+1) is derived from G(i,d) by reflecting the table and prefixing the reflected entry with 1 and the original entry with 0.**

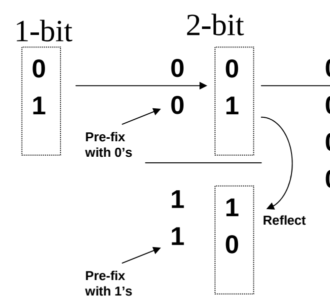
$$G(0,1) = 0$$
$$G(1,1) = 1$$

**A linear array composed of $2^d$ nodes can be embedded into a d-dimensional hypercube using this mapping**

$$G(i, x+1) = \begin{cases} G(i,x), & i < 2^x \\ 2^x + G(2^{x+1} - 1 - i, x), & i \geq 2^x \end{cases}$$



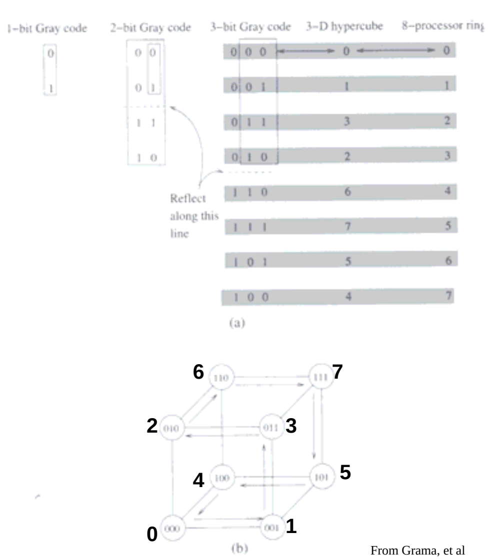
## Example of BRG Code: 8p Ring → 8p hypercube




| 3-bit | 8p ring | 8p hyper |
|---|---|---|
| 0 0 0 | 0 | 0 |
| 0 0 1 | 1 | 1 |
| 0 1 1 | 2 | 3 |
| 0 1 0 | 3 | 2 |
| 1 1 0 | 4 | 6 |
| 1 1 1 | 5 | 7 |
| 1 0 1 | 6 | 5 |
| 1 0 0 | 7 | 4 |



## 8p ring → 8p hypercube




From Grama, et al

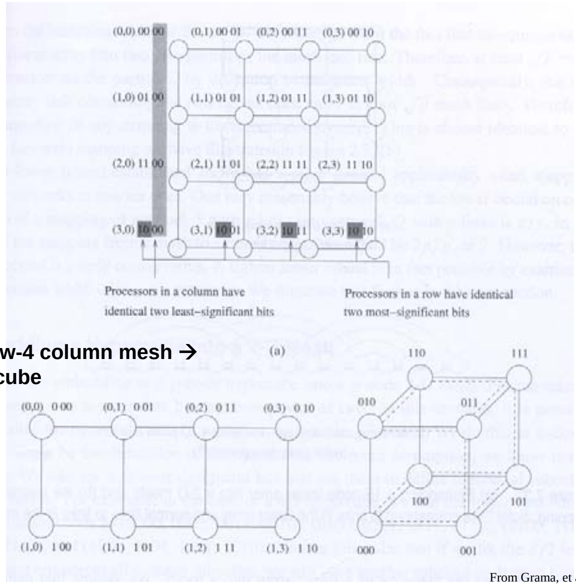


## Embedding Other Networks on Hypercubes:

- **Hypercube is a rich topology, many other networks can be "easily" mapped onto it.**
- **Mapping a linear array into a hypercube:**
  - A linear array (or ring) of $2^d$ processors can be embedded into a d-dimensional hypercube by mapping processor i onto processor G(i,d) of the hypercube
- **Mapping a $2^r$ x $2^s$ mesh on a hypercube:**

  processor(i,j)---> G(i,r)||G(j,s) (|| denote concatenation)

## Mapping Meshes → Hypercubes




**Example: 2 row-4 column mesh → 8-node hypercube**

From Grama, et al



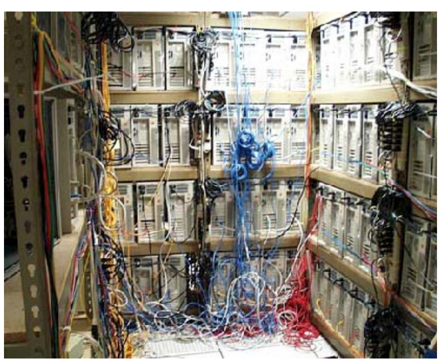
## Trade-Off Among Different Networks

| Network | Minimum latency | Maximum Bw per Proc | Wires | Switches | Example |
|---|---|---|---|---|---|
| Completely connected | Constant | Constant | O(p*p) | - | - |
| Crossbar | Constant | Constant | O(p) | O(p*p) | Cray |
| Bus | Constant | O(1/p) | O(p) | O(p) | SGI Challenge |
| Mesh | O(sqrt p) | Constant | O(p) | - | Intel ASCI Red |
| Hypercube | O(log p) | Constant | O(p log p) | - | Sgi Origin |
| Switched | O(log p) | Constant | O(p log p) | O(p log p) | IBM SP-2 |



## Beowulf

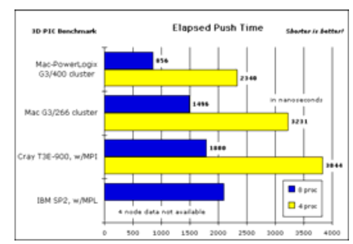
- **Cluster built with commodity hardware components**
  - PC hardware (x86,Alpha,PowerPC)
  - Commercial high-speed interconnection (100Base-T, Gigabit Ethernet, Myrinet,SCI)
  - Linux, Free-BSD operating system

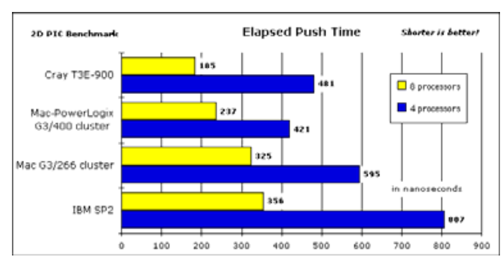


http://www.beowulf.org



## Apple: PowerPC cluster

## Clusters of SMP

- **The next generation of supercomputers will have thousand of SMP nodes connected.**
  - Increase the computational power of the single node
  - Keep the number of nodes “low”
  - New programming approach needed, MPI+Threads (OpenMp,Pthreads,….)
  - See www.top500.org



Jaguar


Sequoia

http://www.llnl.gov/asci



## Tianhe-2 (1st)

- **China’s National University of Defense Technology**
- **54.9 Petaflops at peak**
- **1,024 Terabytes of memory**
- **3,120,000 Xeon cores**
- **Sustained performance of up to 33.8 Petaflops**

- **http://en.wikipedia.org/wiki/Tianhe-2**



## Titan (2nd)

- **DoE Oak Ridge National Lab**
- **27.1 Petaflops at peak**
- **560,640 core processors (70,080 8-core nodes) Opteron with NVIDIA K20x**
- **710 Terabytes of memory**
- **Sustained performance of up to 17.6 Petaflops**

- **http://en.wikipedia.org/wiki/Titan_(supercomputer)**



## Sequoia – BlueGene/Q (3rd)

- **DoE LLNL**
- **20.1 Petaflops at peak**
- **1,573 Terabytes of memory**
- **1,572,864 cores IBM Power BQC**
- **Sustained performance of up to 17.2 Petaflops**

- **http://en.wikipedia.org/wiki/IBM_Sequoia**