Parallel Computing

Ekkapot Charoenwanit

Software Systems Engineering
TGGS
KMUTNB

Lecture 2:

Interconnection Networks

Bus-Based Network

In a ***bus-based*** network, nodes communicate over a shared medium common to all of them:

- Conventionally, a shared medium allows ***only one*** message to be sent ***at a time***.

Nodes broadcast their messages over the shared medium:

- Each node ***listens*** to every message and receives the ones for which it is the destination.
- Typically, before sending a message, a processor listens until the medium is not occupied, then attempts to send its message.

Bus-Based Network

Advantages:

- Cost scales linearly with the number of nodes p: $O(p)$.
- Distance between any two node is $O(1)$.
- A bus-based machine is ideal for broadcasting information among nodes.

Disadvantages:

- Performance does not scale well as p increases as a result of contention for the shared medium.
- Therefore, a bus-based machine is only limited to a few dozen of nodes.

Switch Networks

A ***switch network*** can be represented as a ***graph*** where ***vertices*** represent processors and switches, and ***edges*** represent communication paths.

- Each processor is connected to one switch.
- Switches connect processors and/or other switches.

Switch Networks: Diameter

Diameter:

- Maximum distance δ_{max} between any two switch nodes
  - Distance $\stackrel{\text{def}}{=}$ Length of a Shortest Path in Terms of Links
- The lower, the better.
- Determines lower bound on communication time

Switch Networks: Bisection Width

Bisection Width:

- Minimum number of links that must be removed in order to divide the network into two halves of equal size or size differing by at most one node
- The higher, the better.
- Better fault-tolerance

Imagine a network whose bisection width is ***one***. This means removing a certain link will split the network into two halves of equal size. All data that flow from one half to the other must pass through this link, meaning that this link is a ***bottleneck*** through which data must flow through ***sequentially***.

Switch Networks: Connectivity and Cost

Connectivity:

- Minimum number of links needed to be removed to break the network into two disconnected networks.
- The higher, the better.
- Measures multiplicity of paths

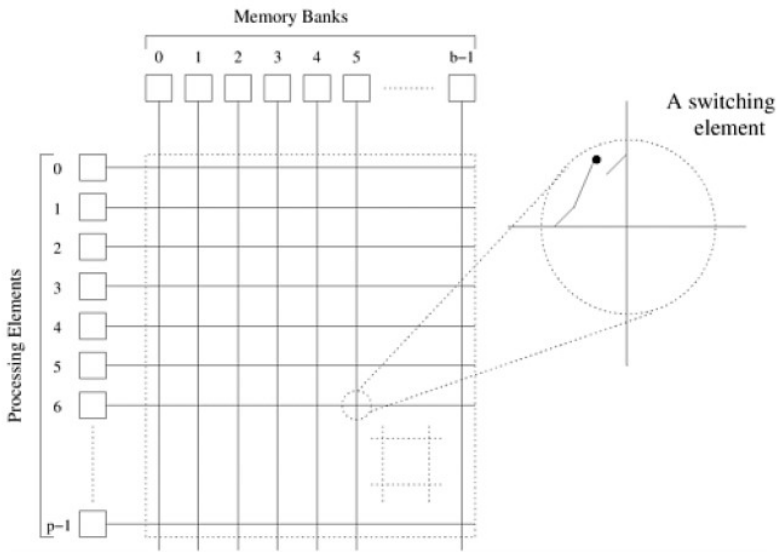
Cost:

- Number of ***links*** or ***switches***, depending on which one is ***asymptotically larger***
- The smaller, the better.

Crossbar Network

A simple way to connect p processors to b memory banks is to use a ***crossbar*** network:

- The network employs a grid of switches.
- The network is non-blocking:
  - The connection of a processor nodes to a memory bank does not block the connection of any other processors to any other memory banks.
- The number of switches is $pb = \Theta(pb)$.



Crossbar Network

Crossbar Network:

It is reasonable to assume that the number of memory banks b is at least the number of processors p:

- $b \geq p$
- As p increases, the cost of the network grows as $\Omega(p^2)$.
- The network is not scalable in terms of cost.
- The network is scalable in terms of performance, up to certain physical limitations.

Multistage Network

Bus vs Crossbar:

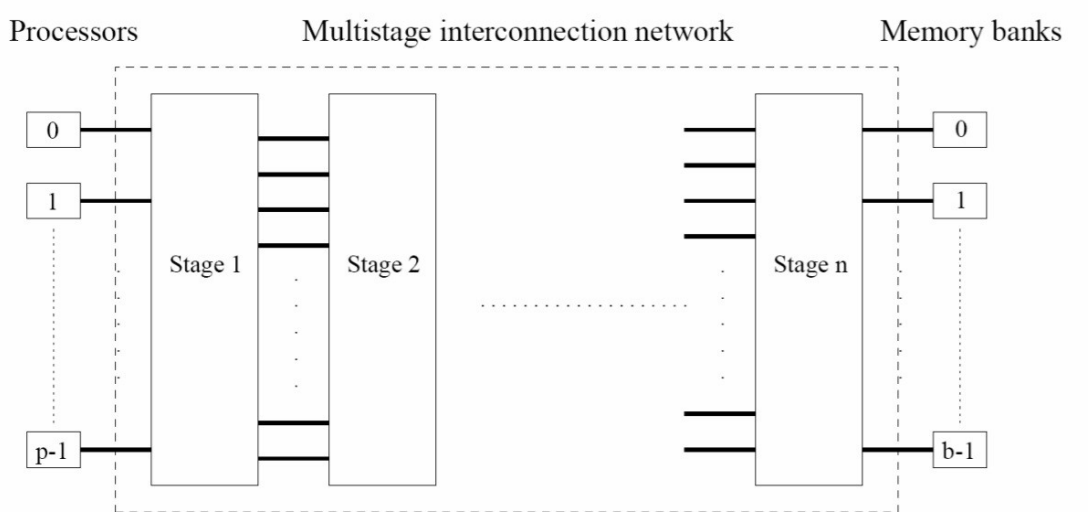
- The crossbar network is scalable in terms of performance, but not scalable in terms of cost $\Omega(p^2)$.
- The bus network is scalable in terms of cost $O(p)$, but not scalable in terms of performance.

An intermediate class of networks called ***Multistage Network*** lies between these two extremes:

- More scalable in terms of cost than the crossbar network
- More scalable in terms of performance than the bus network

Multistage Network

The diagram shows a general multistage network that connects p processors to b memory banks.



Omega Network

Omega Network:

A commonly used multistage network is the ***omega*** network:

- The network consists of $\log p$ stages.
- Each stage consists of an interconnection pattern that connect p inputs to p outputs.
- A link exists between input i and output j iff

$$j = \begin{cases} 2i, & 0 \leq i \leq p/2 - 1 \\ 2i + 1 - p & p/2 \leq i \leq p - 1 \end{cases}$$

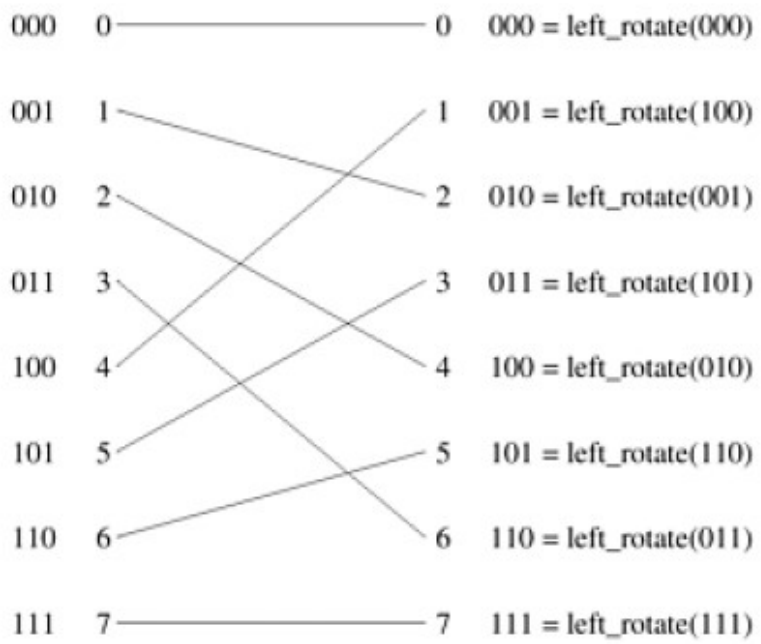
Multistage Network: Omega Network

$$j = \begin{cases} 2i, & 0 \leq i \leq p/2 - 1 \\ 2i + 1 - p & p/2 \leq i \leq p - 1 \end{cases}$$

The equation above represents a ***left-rotation*** operation on the binary representation of input port i to figure out output port j:

- This interconnection pattern is called a ***perfect shuffle***.
- At each stage, a perfect shuffle interconnection pattern feeds into a set of $p/2$ switches.
- Each switch can be in one of the ***two modes***:
  - In the ***pass-through*** mode, the inputs are sent straight through the outputs.
  - In the ***cross-over*** mode, the inputs are crossed over and then sent out.

Omega Network



Omega Network

Each switch can be in one of the ***two modes***:

- In the ***pass-through*** mode, the inputs are sent straight through the outputs as in Figure **(a)**.

- In the ***cross-over*** mode, the inputs are crossed over and then sent out as in Figure **(b)**.

(a) (b)

Multistage Network: Omega Network

The ***omega network*** has $\log p$ stages, each of which consists of $p/2$ switches:

- In total, there are $\frac{p}{2} \cdot \log p = \Theta(p \log p)$ switches.
- Therefore, the cost of the network is $\Theta(p \log p)$.
- This is asymptotically smaller than the cost of the ***crossbar network*** $\Omega(p^2)$.

Omega Network

Routing:

Let s be the binary representation of a processor that needs to access a memory bank t.

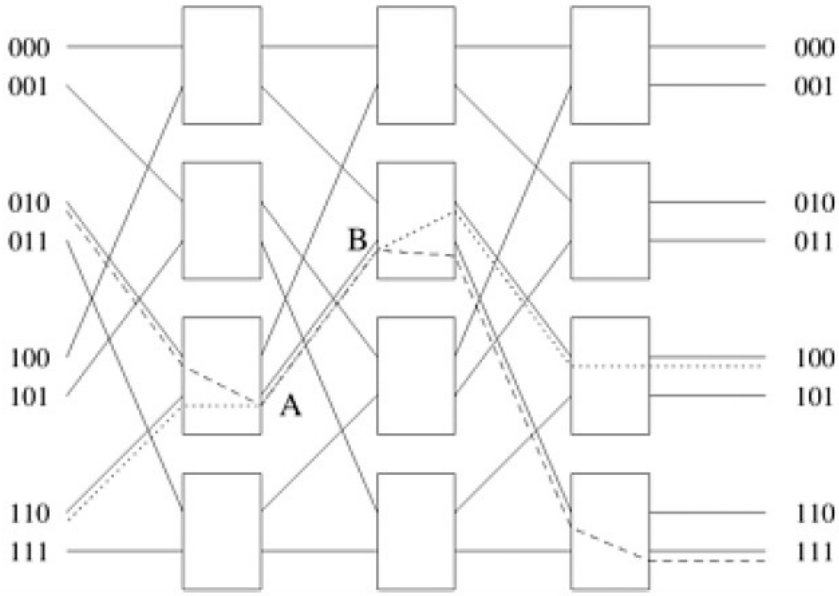
- The data traverses the link to the first switch.
- If the most significant bit of s and t are the same, the data is routed in the ***pass-through*** mode.
- Otherwise, the data is routed in the ***cross-over*** mode.
- This scheme is repeated at the next switch using the next most significant bits of s and t.

Omega Network

The omega network is a ***blocking*** network as can be seen in the figure.

Processors 2 (010) is accessing memory bank 7 (111).
Processors 6 (110) is accessing memory bank 4 (100).

Although the two processors are accessing two different memory banks, there is a contention on the communication link **AB** as shown in the figure.



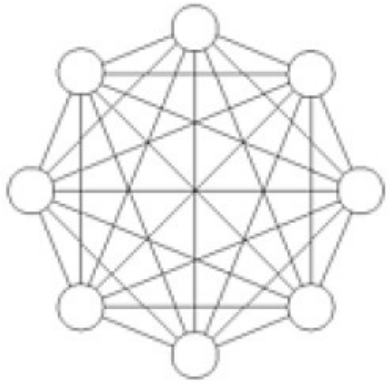
Fully-Connected Network

A ***fully-connected*** network is a network where each node is connected to every other node:

- Any pair of nodes can send a message to each other in a ***single step***.
- The number of links is $\binom{p}{2} = p(p-1)/2$ links.

Properties:

- Diameter: 1
- Bisection Width: $p^2/4$
- Connectivity: $p-1$
- Cost: $p(p-1)/2$ [#Links]



Linear Array Network

Suppose a linear array consists of p processors.

Properties:

- Diameter: $p - 1$
- Bisection Width: 1
- Connectivity: 1
- Cost: $p - 1$ [#Links]

1D-Ring Network

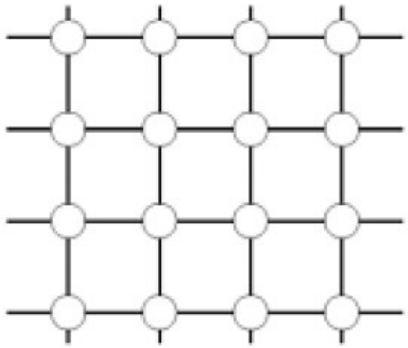
Suppose a 1D-ring consists of p processors.

Properties:

- Diameter: $p/2$
- Bisection Width: 2
- Connectivity: 2
- Cost: p [#Links]

2D-Mesh Network without Wraparound

Suppose a 2D-mesh consists of p processors with $\sqrt{p}$ rows and $\sqrt{p}$ columns.

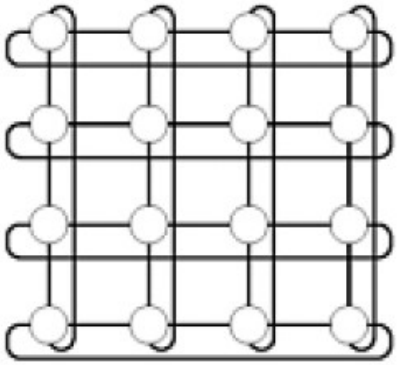


Properties:

- Diameter: $2(\sqrt{p}-1)$
- Bisection Width: $\sqrt{p}$
- Connectivity: 2
- Cost: $2(p-\sqrt{p})$ [#Links]

2D-Mesh Network with Wraparound

Suppose a 2D-mesh consists of p processors with $\sqrt{p}$ rows and $\sqrt{p}$ columns.



Properties:

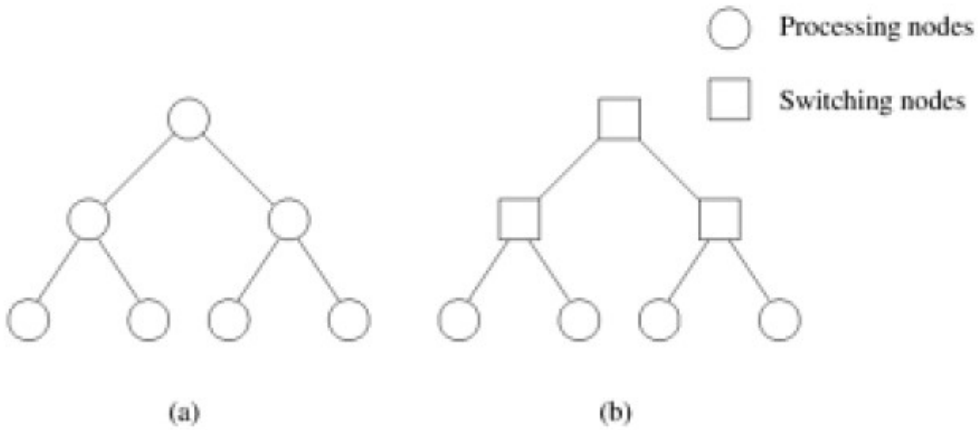
- Diameter: $2(\lfloor \sqrt{p}/2 \rfloor)$
- Bisection Width: $2\sqrt{p}$
- Connectivity: 4
- Cost: $2p$ [#Links]

Tree-Based Network

A ***tree-based*** network is one where there is only path between any nodes any pair of nodes:

- The linear array is a special kind of tree-based networks.

- Static Tree: Figure **(a)**
- Dynamic Tree: Figure **(b)**



Tree-Based Network: Static Tree

Suppose a ***static tree*** consists of p processors of height h:

- The network is a ***complete binary tree*** so $p = 2^{h+1} - 1$.
  - $h = log\,(p + 1) - 1$

Properties:

- Diameter: $2h = 2(\log(p + 1)/2)$
- Bisection Width: 1
- Connectivity: 1
- Cost: $p - 1$ [#Links]

Tree-Based Network: Dynamic Tree

Suppose a ***dynamic tree*** consists of p processors and n switches of height h:

$$n = 2p - 1$$

Properties:

- Diameter: $2h = 2 \log p$
- Bisection Width: 1
- Connectivity: 1
- Cost: $3p - 2$ [#Links]

Tree-Based Network

Routing:

To route a message, the source node sends the message up the tree until the node at the root of the smallest subtree that contains both source and destination nodes.

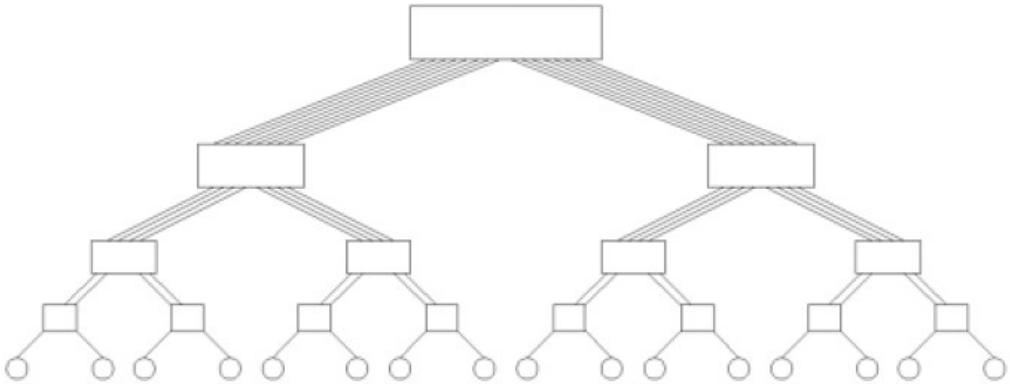
Therefore, tree-based networks suffer from a ***communication bottleneck*** at higher levels of the tree.

- Imagine many nodes in the left subtree of a node communicate with many nodes in the right subtree.
- Under such a circumstance, the root node must handle all the messages.

Tree-Based Network

The communication bottleneck at higher levels of a tree-based network can be alleviated in dynamic tree networks by increasing the number of communication links and switching nodes closer to the root.

This network is called a ***fat tree***.



Hypercube

Suppose a d-dimension hypercube of p processors.

$$p = 2^d.$$

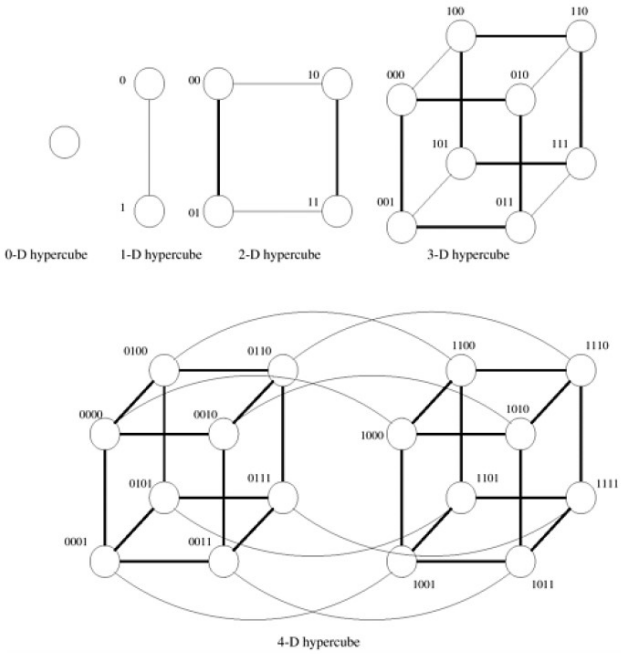
- The distance between any two processors is given by the number bit positions at which the two differ.
- Each processor has $\log p$ neighbors.

Properties:

- Diameter: $\log p$
- Bisection Width: $p/2$
- Connectivity: $\log p$
- Cost: $(p \log p)/2$ [#Links]

Hypercube

In general, a d -dimensional hypercube is constructed by connecting corresponding nodes of ***two*** $(d-1)$ dimensional hypercubes.



Communication Cost

One of the major overheads in the execution of parallel programs arises from the exchange of data between processors.

The cost of communication depends on a variety of factors such as the parallel programming paradigm, the network topology, data handling and routing, as well as associated software protocols.

In this lecture, we will focus on ***Message Passing cost***.

Message Passing Cost

The time taken to communicate a message between two processors is the sum of the following ***three*** components:

- ***Startup time*** t_s
- ***Per-hop time*** t_h
- ***Per-word transfer time*** t_w

Message Passing Cost

Startup time t_s

- The startup time is the time required to handle a message at the sending and receiving processors.
- This includes
  - the time to prepare the message (adding header, trailer, and error correction information)
  - the time to execute the routing algorithm
  - the time to establish an interface between the local node and the router
- This delay is incurred only once for a single message transfer.

Message Passing Cost

Per-hop time t_h

- After a message leaves a node, it takes a finite amount of time to reach the next node in its path.
- The time taken by the header of a message to travel between two directly-connected nodes in the network is called ***the per-hop time***, also known as ***node latency***.
- The per-hop time is directly related to the ***latency within the routing switch*** for determining which output buffer or channel the message should be forwarded to.

Message Passing Cost

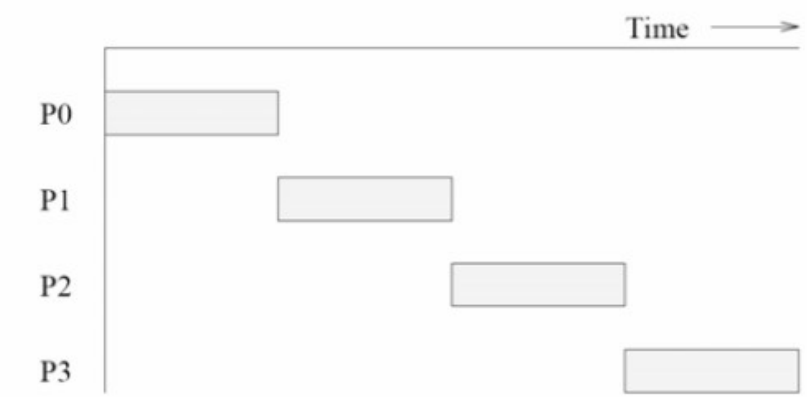
Per-word transfer time t_w

- If the ***channel bandwidth*** is r words per second, then each word takes time $t_w = 1/r$ to traverse the link.
- This time is called the ***per-word transfer time***.
- This time includes network as well as buffering overheads.

Store-and-Forward Routing

Store-and-Forward Routing

- A switching node forwards a message to the next node only after it has received and stored the ***entire*** message.



Store-and-Forward Routing

Suppose that a message of size m is being transmitted through a network that uses ***store-and-forward routing***.

Assume that it will need to traverse l communication links, at each of which the message incurs a cost t_h for the ***header*** to be processed and t_w for the rest of the message to transverse the link.

Therefore, the total communication cost in a store-and-forward routing network is

$$t_{comm} = t_s + (t_h + mt_w)l$$

In typical parallel computers, the per-hop time is relatively small and, in most parallel algorithms, it is smaller than mt_w even for small values of m so it can be safely ignored.

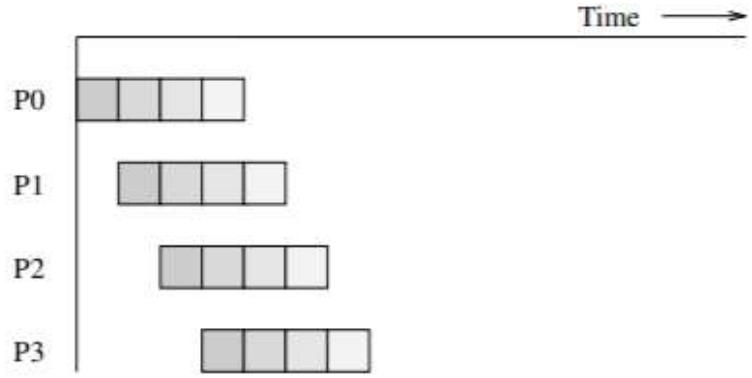
$$t_{comm} = t_s + mt_w l$$

Packet Routing

Store-and-Forward outing makes poor use of communication resources.

In ***packet routing***, messages are broken into ***packets*** (of smaller size) and pipeline them through network.

- Packets may take ***different paths*** so each packet must carry routing, error-checking and sequencing information.



Packet Routing

Suppose the size of a single packet is given by $r + s$, where

- r is the size pertaining to ***actual data***
- s is the size pertaining to ***additional information*** carried by the packet

The time for packetizing a message is proportional to the size of the message. We denote this time by mt_{w1}.

If the network is capable of communicating one word every t_{w2} seconds, incurs a delay of $t_h$seconds on each hop and if the ***first packet*** traverses l hops, then this packet takes $t_h l + t_{w2}(r + s)$ seconds to reach the destination.

After this time, the destination node receives the ***remaining packets*** every $t_{w2}(r + s)$ seconds.

Since there are $\frac{m}{r} - 1$ remaining packets, the total communication time is

$$t_{comm} = t_s + mt_{w1} + t_h l + t_{w2}(r + s) + (\frac{m}{r} - 1)t_{w2}(r + s)$$

Packet Routing

The total communication time is

$$t_{comm} = t_s + mt_{w1} + t_h l + t_{w2}(r+s) + (\frac{m}{r} - 1)t_{w2}(r+s)$$

Let $t_w = t_{w1} + (1 + \frac{s}{r})t_{w2}$.

This simplifies to:

$$t_{comm} = t_s + t_h l + mt_w$$

Note that this cost model assumes that all the packets of a message ***follows the same path***, which may not be the case for a packet routing network.

Simplified Cost Model

$$t_{comm} = t_s + t_h l + m t_w$$

The equation implies that, in order to optimize the cost of message transfers, we need to:

- ***communicate in bulk:***
  - Instead of sending small messages and paying for a startup cost t_s for each, we want to aggregate small messages into a single large message and amortize the startup latency across a larger message.
  - On typical platforms such as clusters and message-passing machines, t_s is much smaller than t_h or t_w.
- ***minimize the volume of data:***
  - To minimize the overhead paid in terms of per-word transfer time t_w, it is desirable to reduce the volume of data communicated as much as possible.
- ***minimize distance of data transfer:***
  - We should minimize the number of hops l that a message must traverse.

Simplified Cost Model

Observations:

- the per-hop time t_h is typically dominated by the startup latency t_s for small messages or by the transfer time mt_w for large messages.
- the maximum number of hops l in most networks is relatively small so the per-hop time t_h can be ignored with little loss in accuracy.

All of these point to the following simpler cost model:

$$t_{comm} = t_s + mt_w$$

Simplified Cost Model

Simplified Cost Model:

$$t_{comm} = t_s + mt_w$$

This equation has significant implications for ***architecture-independent*** parallel algorithms.

Since the cost model implies that it takes the same amount of time for any pair of processors to communicate, the equation corresponds very well to a ***fully-connected network***.

- Instead of designing an architecture-specific parallel algorithm, we design a parallel algorithm with this cost model in mind and port it to any target parallel computer architecture.
- It is also important to keep in mind that this cost model assumes ***uncongested*** networks so it is valid as long as the underlying communication pattern does not congest the network.