Logistics

- Overdue for class representatives. Please we are looking for nominations!
- Next lecture on 08/03, 4-6PM at the Anthony Low Bldg 124 Innovations Theatre (replaces 11/03 lecture, Canberra day)
- Assignment 1 will be released on 11/03 (Mon, next week)

Overview: Routing and Communication Costs

- routing mechanisms and communication costs
- routing techniques: store-and-forward (SF), cut-through (CT)
- communication algorithms and cost modelling. Special focus on broadcast:
  - in particular, two broadcast variants: one-all and all-all
  - impact of store-and-forward (SF) and cut-through (CT) on communication cost
  - we consider rings, meshes, torus, trees and hypercubes in the analysis
- improving the speed of (some) communication operations

References. Grama et. al.:

- Section 2.5 and 2.6
- Ch 4 (includes algorithms/cost analysis beyond one-all/all-all broadcast)

Routing Mechanisms

Efficient algorithms for routing messages through the network are essential for the performance of parallel systems

Given a source and destination node in the network, determine which path(s) a message takes through the network to reach its destination. Can be classified into:

- Minimal versus non-minimal routing. Minimal takes (one of the) shortest paths, each link brings you closer to destination. Non-minimal may route the message along a longer path to avoid network congestion.
- Deterministic versus adaptive routing. Deterministic determines a unique path solely based on the source and destination node. Adaptive routing uses information on the state of the network to determine the path of the message.

Example of a minimal deterministic routing is the so-called dimension-order routing in 2D meshes (e.g., XY-routing) and hypercubes (E-cube routing)

Communication Costs

Communication cost model of a point-to-point message parameterized by:

- start-up time (t_s): time required to handle a message at the sending node and at the receiver. This includes the time to prepare the message for transmission (adding header, trailer, error correction info, etc.), execute the routing algorithm, and the time to establish an interface between the local node and routing switch.
- per-hop time (t_h): time taken for the header of the message to travel between two directly-connected nodes. Related to the latency in the routing switch
- per-word transfer time (t_w): time taken by each word to traverse the link.

Store-and-Forward (SF) Routing Technique

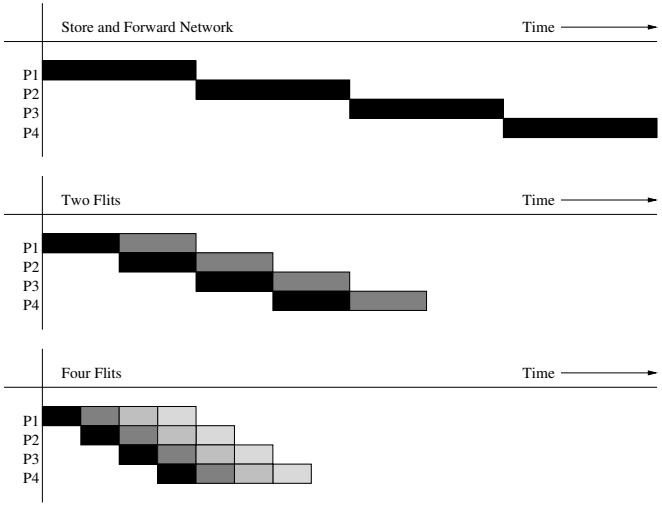
- common on networks for early parallel systems
- each intermediate node in the path from the sending to receiving node forwards the message **ONLY AFTER** the entire message has been received and stored
- assume a message of size m traversing l links, point-to-point comm. cost is:
  $t_{\text{comm}} = t_s + l(t_h + mt_w)$
- usually the per-hop time $t_h << mt_w$ (even for "small" m), and therefore:
  $t_{\text{comm}} \approx t_s + lmt_w$
- poor utilization of communication resources as only one link is active at a time for the communication at hand

Cut-Through (CT) Routing Technique (aka wormhole routing)

- the CT term is more broadly used in long-haul networks (e.g. the Internet); for high-performance networks (this course) CT is aka wormhole routing
- message split into small-sized fixed-length segments called flow-control digits (flits)
- message header takes lt_h to arrive (acts as a tracer that sets up the connection)
- flits are forwarded ASAP at each intermediate node, in a pipelined fashion, increasing communication resources usage (all links asymptotically used at a time)
- communication cost can be (asymptotically) modelled as (why?):
  $t_{\text{comm}} = t_s + lt_h + mt_w$
- compared to store-and-forward routing, l does not multiply mt_w. Besides, its implementation requires less memory at the intermediate nodes (why?)
- more sophisticated versions extend the concept to multi-lane CT routing (aka virtual channels) to handle more efficiently variable message sizes
- pioneered (mostly) by Bill Dally for the Torus Routing chip (1986), with milestone papers on deadlock avoidance (1987) and virtual channels (1992)

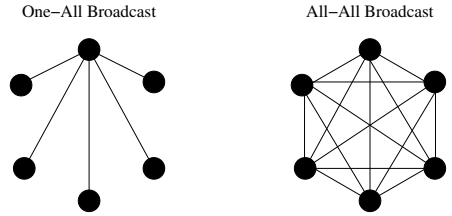
Pipelining of Flow Control Digits (aka flits)

node P1 sends a message to node P4 that passes through P2/P3 nodes
(shaded regions represent the time the message is in transit)



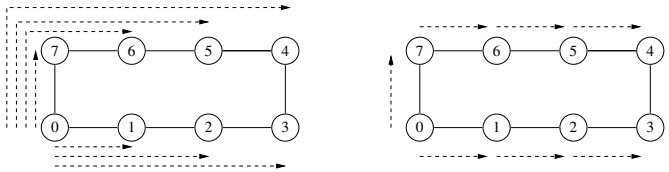
Communication Costs on Static Networks

- for a single point-to-point message transfer, communication costs can be modelled as (recap):
  - store-and-forward: $t_{\text{comm}} = t_s + l(mt_w + t_h) \approx t_s + lmt_w$
  - cut-through: $t_{\text{comm}} = t_s + lt_h + mt_w$
- how can we extend this to derive cost models for communication patterns?
- for simplicity, let us focus on broadcast communication pattern:



Store-and-Forward/One-All/Ring

- (from now on) let us assume that:
  - node 0 is the source of the broadcast
  - each node can only send a single message at a time
  - the network uses deterministic minimal routing (shortest path)
- two possible algorithms to perform one-all broadcast on a ring:



(left: naive; right: broadcast 2-way split)

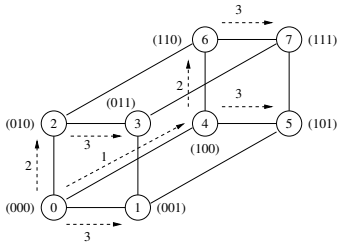
- naive communication cost modelled as (assuming that p is an odd number):

$$t_{\text{one-all}} = (p-1)t_s + \sum_{l=1}^{\lfloor \frac{p}{2} \rfloor} 2l(t_h + mt_w)$$

- broadcast 2-way split communication cost can be modelled as:

$$t_{\text{one-all}} = \lceil \frac{p}{2} \rceil (t_s + t_h + mt_w)$$

Store-and-Forward/One-All/Hypercube



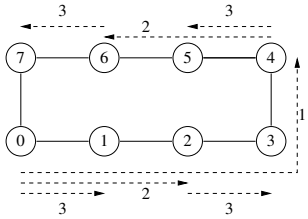
- initial stage involves the source node sending the message along the highest dimension of hypercube (dimension specified by the most significant bit in the binary representation of the node label)
- then, the process continues, stage by stage, along successively lower dimensions
- at each stage, the number of pairs of communication nodes doubles
- total communication cost modelled as:

$$t_{\text{one-all}} = \log_2(p)(t_s + t_h + mt_w)$$

Question: would you expect any improvement from Cut-Through routing?

Cut-Through/One-All/Ring

- let us redesign the algorithm such that it is more amenable to CT (on a ring!)
- basic idea: map the hypercube algorithm to the ring (assume p is a power of 2)



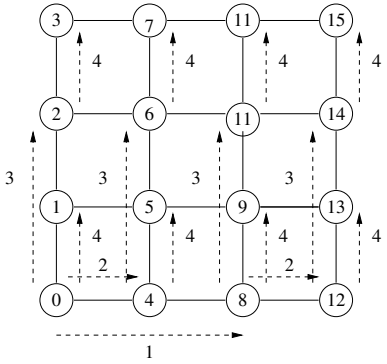
- all messages are sent in the same direction
- source node sends initial message to furthest
- at each step, the distance of communication halves while the number of pairs of communication nodes doubles

- communication cost can be modelled as:

$$t_{\text{one-all}} = \sum_{i=1}^{\log_2(p)} (t_s + mt_w + \frac{p}{2^i} t_h) = \log_2(p)(t_s + mt_w) + (p-1)t_h$$

- $\approx \frac{p}{2\log_2(p)}$ faster than SF 2-way split broadcast
- Question: Can we do it even better?

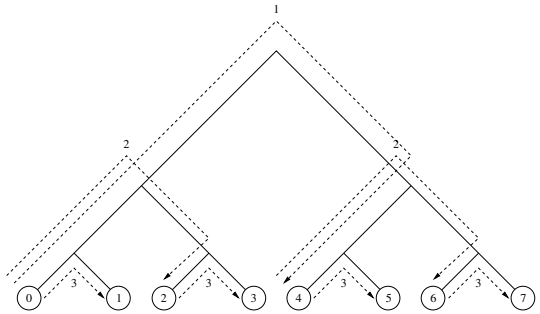
Cut-Through/One-All/2D Mesh



- Two steps:
  1. source node (node 0 in the fig) performs a row-wise broadcast to the remaining nodes of the same row (arrows 1 and 2 in the fig)
  2. all nodes within the row of source node perform a column-wise broadcast each within their resp cols (arrows 3 and 4 in the fig)
- communication cost can be modelled as:

$$t_{\text{one-all}} = 2\log_2(\sqrt{p})(t_s + mt_w) + 2t_h(\sqrt{p} - 1)$$

Cut-Through/One-All/Balanced Binary Tree

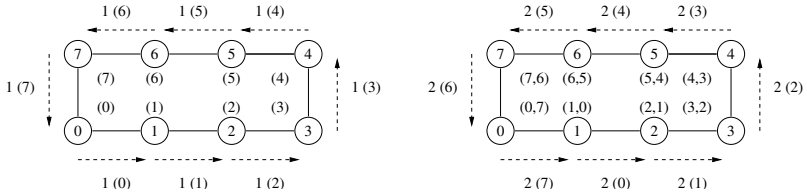


- processing nodes at the leaves, intermediate nodes are just switching units
- assume t_h between switches and nodes is equivalent to t_h between switches
- broadcast completes in $\log_2(p)$ stages; at each stage the number of parallel message transfers doubles (as with the hypercube)
- communication cost can be modelled as:

$$t_{\text{one-all}} = \log_2(p)(t_s + mt_w) + (\sum_{i=1}^{\log_2(p)} 2i)t_h$$

Store-and-Forward/All-All/Ring

- naive approach: perform p one-all broadcasts in sequence
  - BUT ... communication cost scales as $pt_{\text{one-all}}$ (i.e., p times $t_{\text{one-all}}$)
- smarter approach: to perform all p one-all broadcasts simultaneously recasted as $p-1$ shifts (leverages available links more efficiently!)



(left: stage 1; right: stage 2)

- communication cost can be modelled as:

$$t_{\text{all-all}} = (p-1)(t_s + t_h + mt_w)$$

Store-and-Forward/All-All/Torus

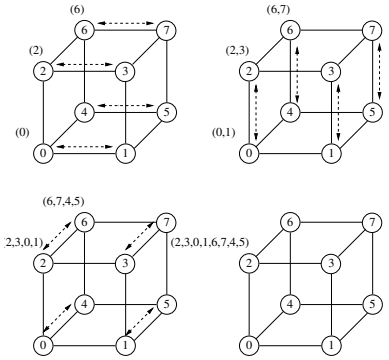
Three steps:

1. each process in an arbitrarily selected row performs an all-all broadcast within the row using the algorithm SF/All-All/Ring
2. each node collects the $\sqrt{p}$ messages they have received from the other nodes in that row, and consolidates them into a single message of length $m\sqrt{p}$
3. each column performs an all-all broadcast within the column

- communication cost can be modelled as:

$$\begin{aligned} t_{\text{all-all}} &= \underbrace{(\sqrt{p}-1)(t_s+t_h+mt_w)}_{\text{step 1}} + \underbrace{(\sqrt{p}-1)(t_s+t_h+m\sqrt{p}t_w)}_{\text{step 3}} \\ &= 2(\sqrt{p}-1)(t_s+t_h) + (p-1)mt_w \end{aligned}$$

Store-and-Forward/All-All/Hypercube



- performs as many steps as number of hypercube dimensions (recall that $d = \log_2(p)$ in a hypercube)
- at each stage (i.e., dimension) pairs of processes along such dimension exchange data
- with each stage the message length doubles
- communication cost can be modelled as:

$$\begin{aligned} t_{\text{all-all}} &= \sum_{i=1}^{\log_2(p)} 2(t_s + t_h + 2^{i-1}mt_w) \\ &= 2\log_2(p)(t_s+t_h) + 2(p-1)mt_w \end{aligned}$$

Improving the speed of (some) communication operations

- the algorithms and communication costs modelled so far assume that the message (or its flips) always follow the same path among the sender and receiver
- if we split messages into parts, and let each part follow different paths among the sender/receiver, we may sometimes improve network utilization (see Grama et. al, Ch 4.7.1 for examples)
- they also assume that a node cannot be transmitting to several neighbours at a time
- some parallel computers have hardware able to send messages in multiple directions at the same time (aka all-port communication) or even specialized networks for certain kind of communication primitives (e.g., IBM BG/L supercomputer, see Lin & Snyder, Ch 2, "Supercomputers")