EE/CSCI 451: Parallel and Distributed Computation

Lecture #9

9/15/2020

Viktor Prasanna

prasanna@usc.edu

ceng.usc.edu/~prasanna

University of Southern California

USC University of Southern California

Announcement

- Midterm 1 date:
  - **9/25 (Friday) 3:30-5:30 pm**
  - Covers material covered till the end of next week
  - Sample midterms from 2019 are out on Piazza
- Special note: Discussion attendance is required on 9/18

- HW #3 due:
  - Sept. 17 (Friday), AOE, submit on Blackboard

- PHW #3:
  - Out Sept. 15 (Today)
  - Due Sept. 27 (Monday)

Announcement: Midterm1 Logistics

- **Online Proctored** Exam:
  - Time: Week 6 discussion session – 2 hours: 3:30-5:30PM (Los Angeles time)
  - Format: Open-book, open-notes **[Attendance is required, no make-up given]**
  - Proctoring: 2 proctors watching different subgroups of students in separate Zoom meetings, links will be sent to students in advance
    - Require **camera-enabled** device
- Receiving and returning your exam:
  - Exam will be released on Piazza under resource page at/around 3:26 PM
  - You will submit the completed exam on Blackboard (a submission portal will be created in advance)
- Completing your exam:
  - Download the assignment pages (exam pages) as pdf files on to your tablet and annotate it with your answers. Only hand-annotated pdf files are acceptable.
    - Require a **writable tablet** device
- Coverage: Week 1-Week 5 contents (Week 6 contents - analytical modeling & communication primitives - not covered)
- Special note 1: Important - discussion **attendance is required** on Week 5 (Sept 18)!
  - 10-min midterm trial run to make sure all students are prepared for and comfortable with the exam process
- Special note 2:
  - We have created a Piazza poll [link] to collect info regarding your available resources/capabilities to complete the exam wit h writable tablet. Everyone is **required to participate in the poll**

Course Project (1)

- Large software project
  - Scientific computing
  - Graph analytics
  - Data science
  - …
- Sample course projects
  - Multi-core implementation of data plane kernels for software defined networking (e.g., traffic classification, packet classification, etc.)
  - Accelerating Deep Neural Networks (CNN, LSTM, etc; Inference, Training, etc.) using GPU
  - Big data analytics using Spark (e.g., graph analysis, log processing, etc.)
  - …
- Students can work in teams

Course Project (2)

- Project timeline
  - Week 5-8: Identify team members and project topic
  - Week 9: Project proposal due (Oct. 16)
  - Weeks 12-13: Student Project Presentations
  - Week 13: Project final report due (Sunday Midnight AOE)

- Grading breakdown for the course project
  - Project Proposal: 25%
  - Project presentation: 25%
  - Project final report: 50%

Course Info.

- Grading
  - Homework 10%
    - Homeworks must be done independently
    - **10% late penalty per day** will be assessed with no credit received after the third day
  - Programming Assignments 10%
  - Course Project 15%
  - Midterm I (Sept 25 in lab session, 2 hours) 20%
  - Midterm II (Oct 23 in lab session, 2 hours) 20%
  - Final Exam 25%

Outline

- From last class
  - Interconnection networks
    - CLOS network
    - Butterfly network
    - Hypercube network
    - Performance metrices

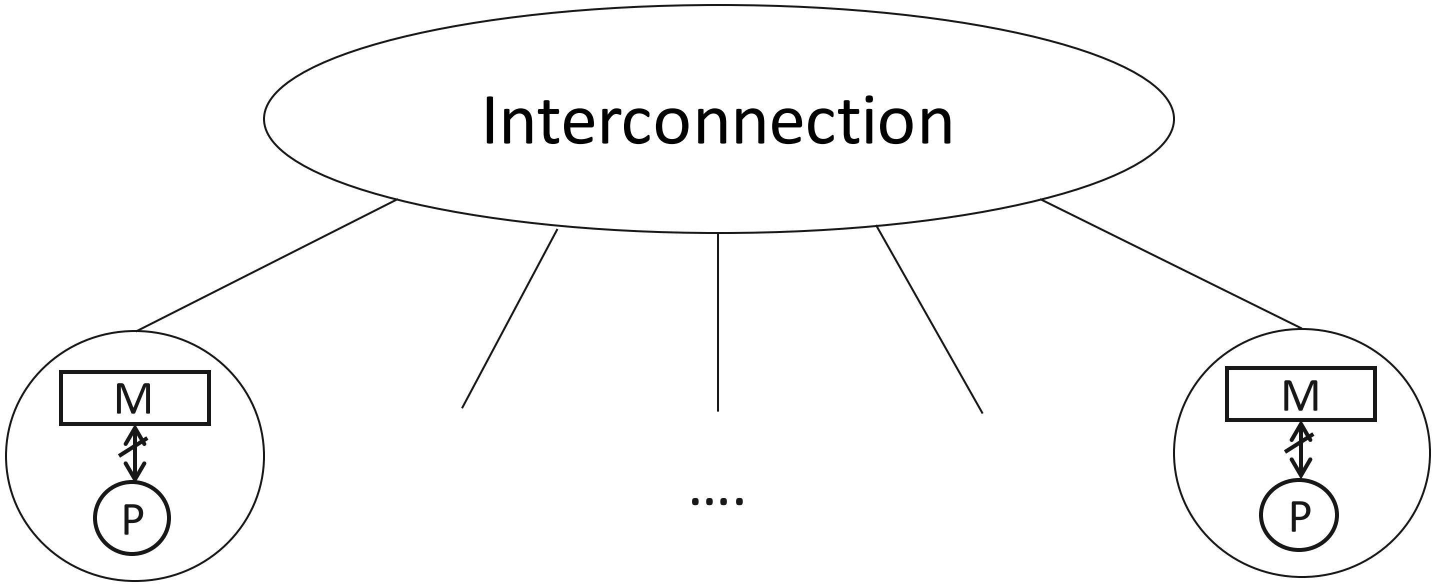
- Today (Chapters 2.5, 2.6)
  - Communication Cost in Parallel Machines
  - Communication Cost in Message Passing Machines
    - Routing mechanisms
    - Cut through routing
    - Cost models
  - LogP model
  - Routing in interconnection networks
  - Communication Cost in Shared Memory Machines

Models

- Parallel programming models
    - Shared
    - Message Passing
    - …

- Parallel computation models
    - PRAM
    - LogP

Communication Cost



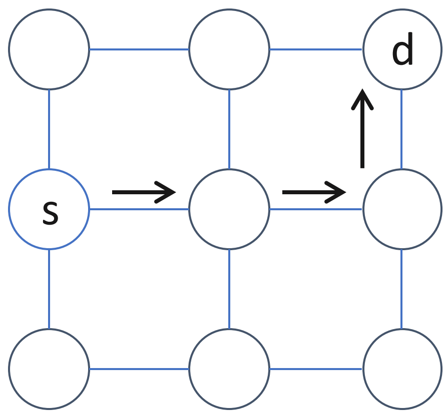
- Process-Memory
- Process to Process

Communication Cost in Message Passing Machines (1)

- Message from source(s) to destination(d)

- t_s (startup time) — time spent in handling the message at source and destination
  - Prepare message (packet: header, data, error correction information)
  - Send message to router
  - Routing algorithm execution (at the source node)
  - Processing at the receiving end

Incurred per message transfer

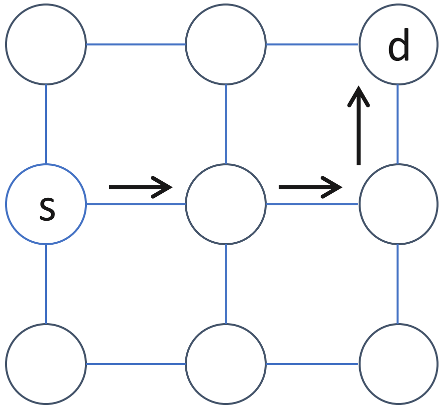


Communication Cost in Message Passing Machines (2)

- t_h (per hop time) — time for header of message to travel from one node to the next node along the path
  - Routing algorithm execution (at the node)
  - Buffer management

- t_w (per word transfer time) — incurred by every data word between adjacent nodes along the path
  - Channel bandwidth = r (words/sec)

$$t_w = \frac{1}{r}$$



Routing Mechanisms (1)

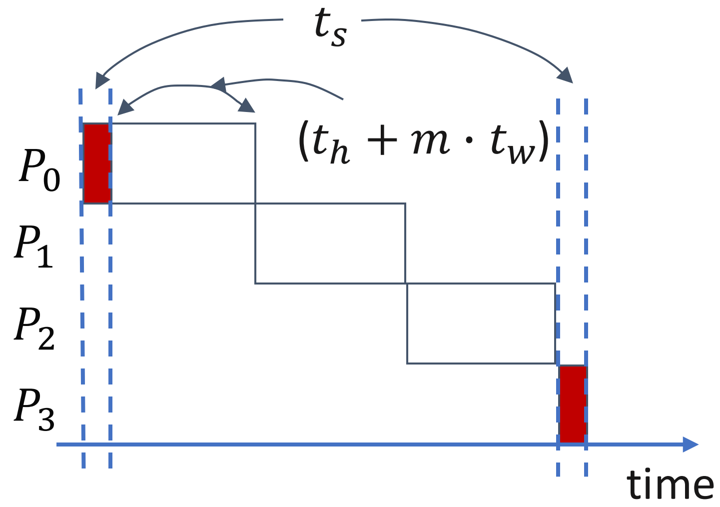
1. Store and Forward Routing

- Message length = m words
- Number of hops = l

$$P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3$$

source destination

Example:
l = 3



- Store: each intermediate node

receives the entire message from its predecessor

- Forward: forward to the next node

Total time for communication = $t_s + (t_h + m \cdot t_w) \cdot l$

Routing Mechanisms (2)

2. Cut through routing

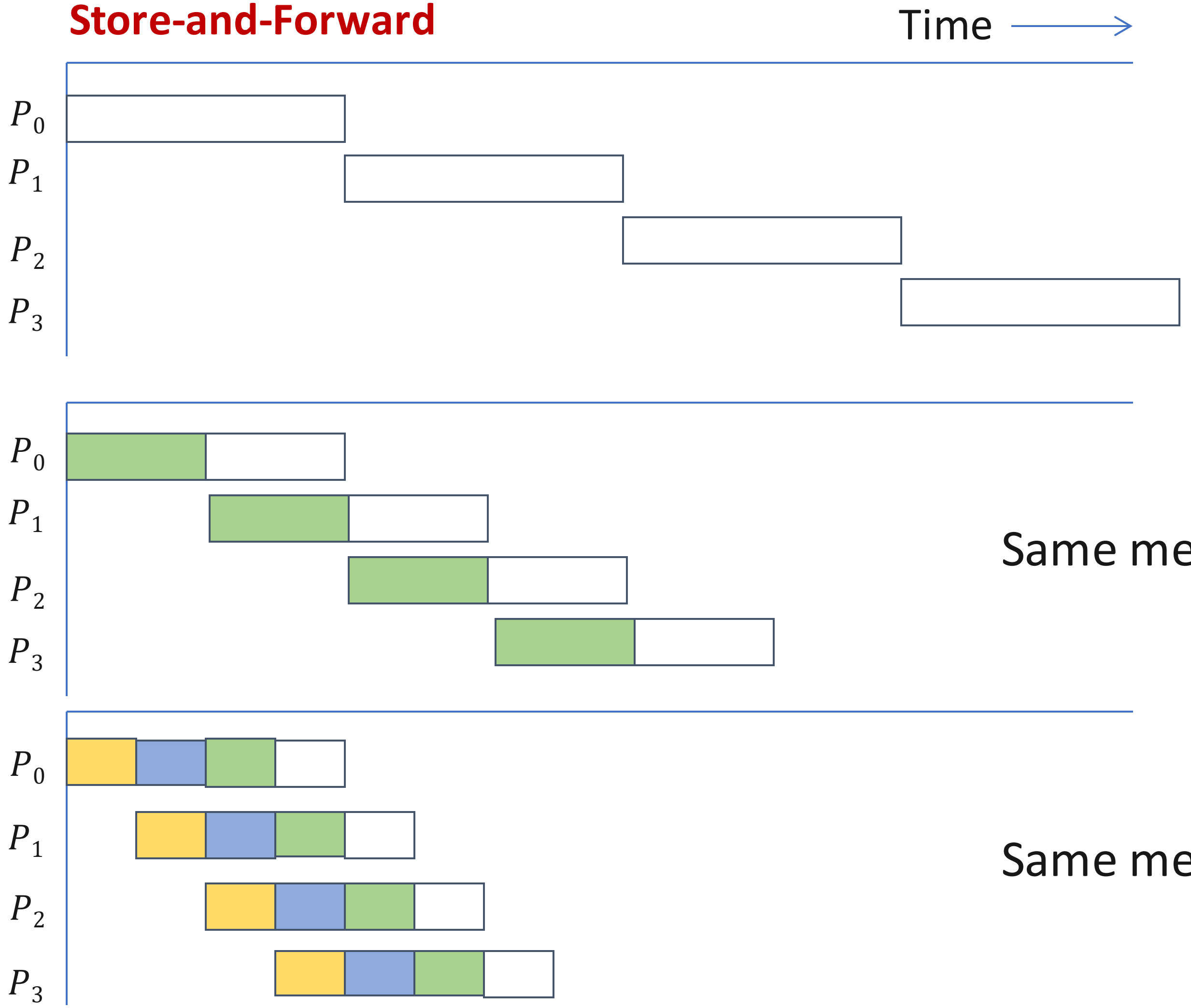
Objective: Reduce overall time for communication

Some ideas to improve store and forward routing (to reduce communication time):

- Reduce packet header size
- Use fixed size message (to reduce buffer management overhead)
- All packets follow the same route (reduce routing complexity)
- Simple error detection/correction (needed in large scale distributed systems)

Pipelined Message Routing

Store-and-Forward



A single message

Same message with packet size = $\frac{m}{2}$

Same message with packet size = $\frac{m}{4}$

Cut Through Routing (1)

- Message = Sequence of fixed size units

FLow control digITS (FLITS)

(Small size)

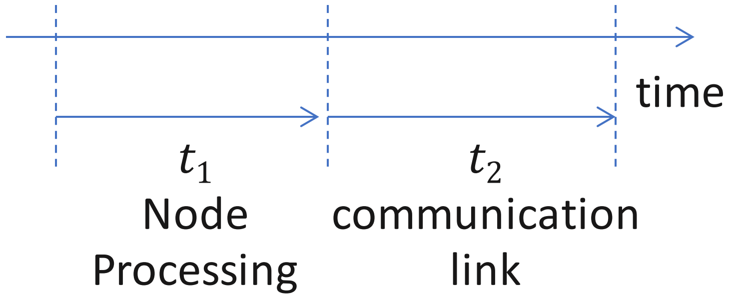
  - Establish connection between source and destination
  - Pipeline the data transfer — each intermediate node receives a FLIT (not the entire message) and forwards along the **same** path

  - No need for large buffer space & buffer management at each node
  - No overhead of (handling variable size) packets

Cut Through Routing (2)

FLIT size (f) = ?

For (optimized) pipelined data delivery: $t_1 \approx t_2$ <-- (pipeline of processing nodes and links)



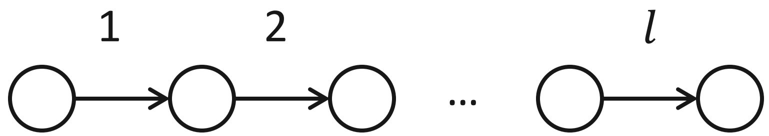
Note: $t_2 \propto f$

f should not be too small (t_1 cannot be very small — high speed control circuit)

f should not be large → need large buffer

f = 4 to 32 bytes

Cut Through Routing (3)



$$t_{comm} = \underbrace{t_s + l \cdot t_h}_{\text{Pipeline delay}} + \underbrace{t_w \cdot m}_{\text{Communication time on a link}}$$

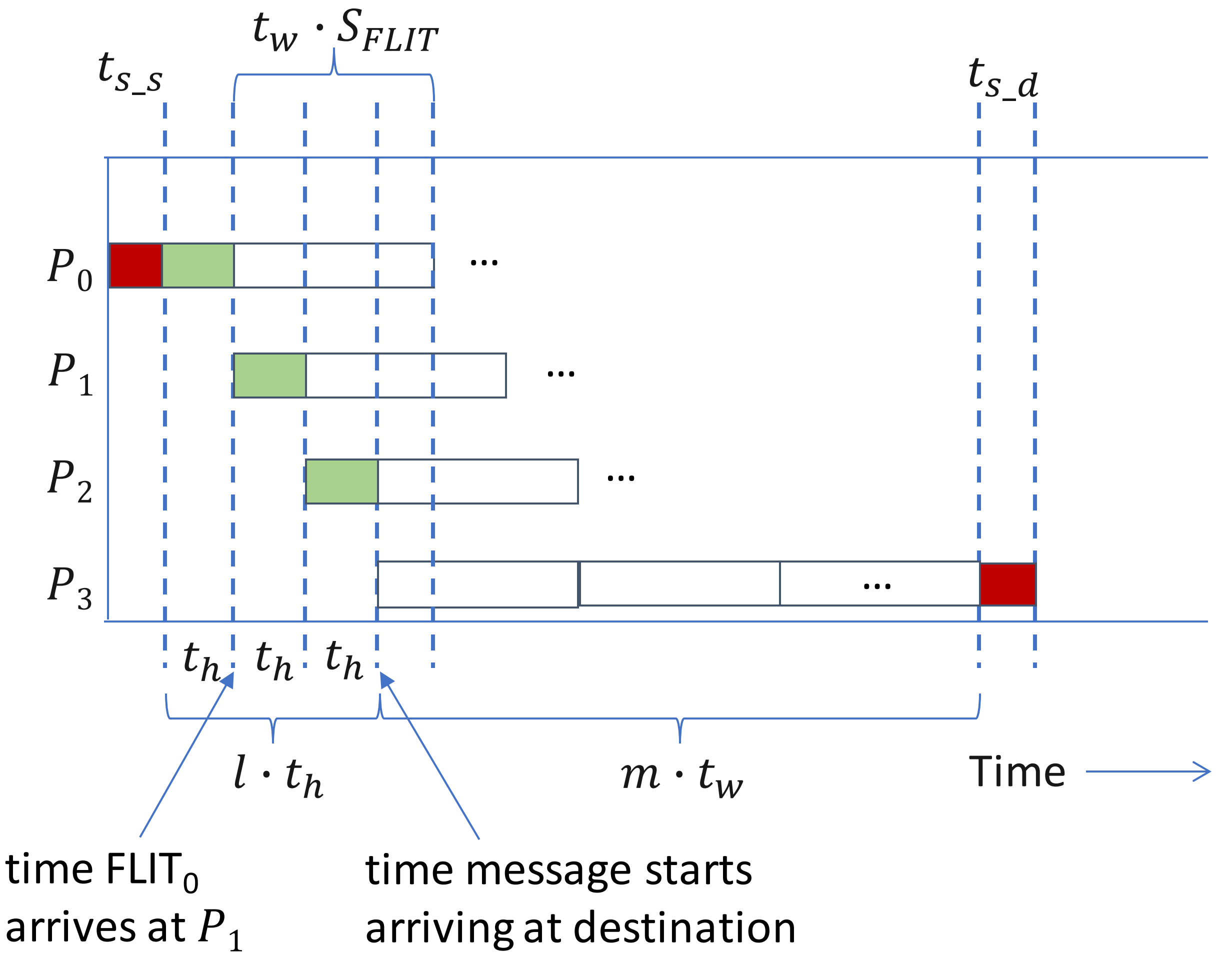
Processing at source and destination
+
set up the path

Pipelined transfer of m units of data

Pipelined Data Communication

Note: There is a small overhead in processing FLIT at each intermediate node, smaller than processing a packet. t_h is smaller than store and forward scenario.

Cut Through Routing (4)



$t_s = t_{s\_s} + t_{s\_d}$

Note: S_{FLIT} is the size of FLIT payload

Number of FLITs = $\frac{m}{S_{FLIT}}$

$l = 3$ in this example

Total time $= t_s + l \cdot t_h + m \cdot t_w$

Cut Through Routing (5)

Note 1:

$$\left.\begin{array}{c} t_s \\ t_h \\ t_w \end{array}\right]$$

Hardware, software layers (OS, buffer management)
Message Semantics

t_s ~ 10's µSec

t_h ~ µSec

t_w ~ 10^{-10} Sec/byte (10 GB/Sec link)

Cut Through Routing (6)

Note 2:

- Latency ? Throughput?
- Short messages (Control message of each link)
  - latency is important
- Long messages (Data communication)
  - Effective throughput is important
- Virtual channels
  - In cut through routing, a long message may hold up links — delaying the (short) messages
  - Idea: share a physical channel as virtual channels among several messages
  - Deadlock may occur

Cut Through Routing (7)

- Communication cost model

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

- Cut through routing is widely used
- Some optimizations (application developer may do)

  Communicate in bulk (startup latency t_s)

  - Aggregate short message into one long message

  Reduce total data volume communicated

  Minimize distance between source and destination (number of hops l)

Cut Through Routing (8)

Minimizing l (number of hops) is hard

- Program has little control in program to processor mapping
- Machines (platform) use various routing techniques to minimize congestion

  (ex: randomized routing)
- Per hop time is usually relatively small compared with t_s and $t_w \cdot m$ for large m

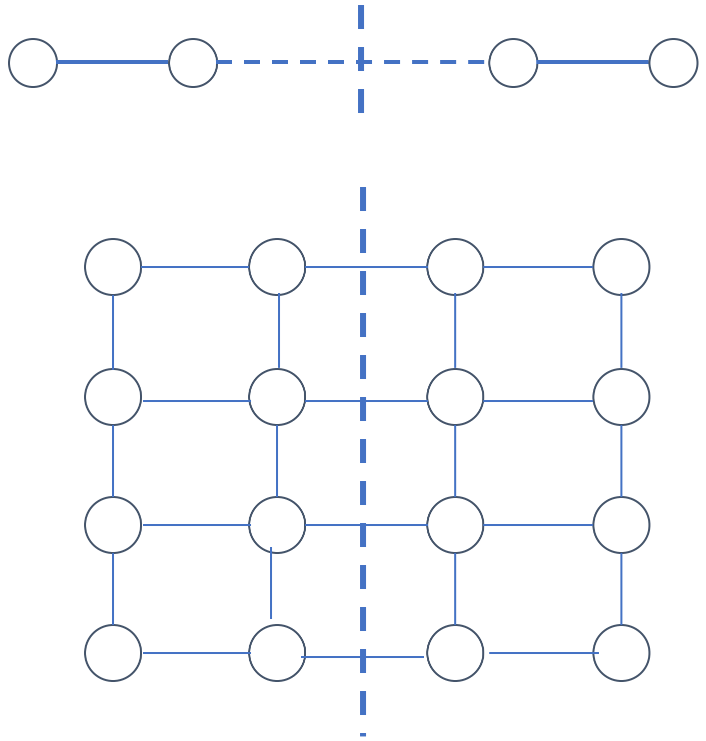
Simple communication cost model: $t_{comm} = t_s + t_w \cdot m$

- Same amount of time to communicate between any two nodes

  (fully connected?)
- Use this cost model to design and optimize instead of parallel architecture (mesh, hypercube) specific algorithms
- Simplified cost model does not take congestion into account

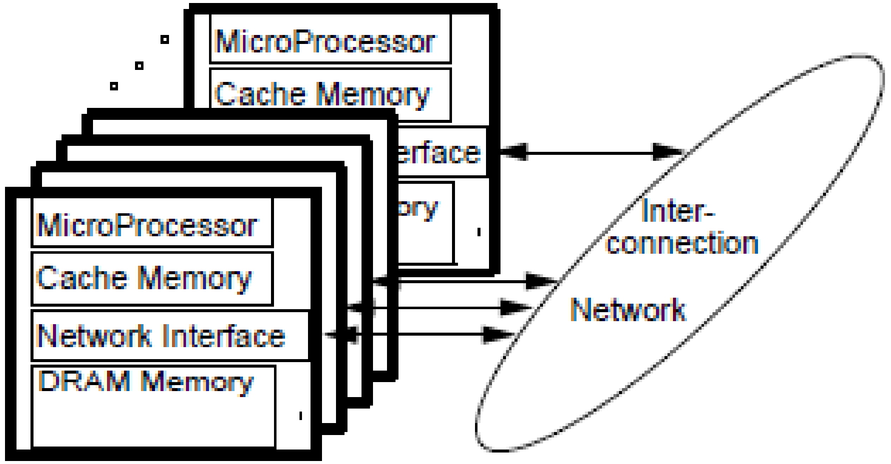
Cut Through Routing (9)

- Effective bandwidth (at the application layer)
  - Parallel architecture
  - Communication pattern
  - Routing algorithm
  - Communication schedule
  - Resulting congestion



LogP Model (1)

- Parallel Architecture = Processor + Memory + Interconnection



This organization characterizes most massively parallel processors (MPPs)
(From LogP paper)

LogP Model (2)

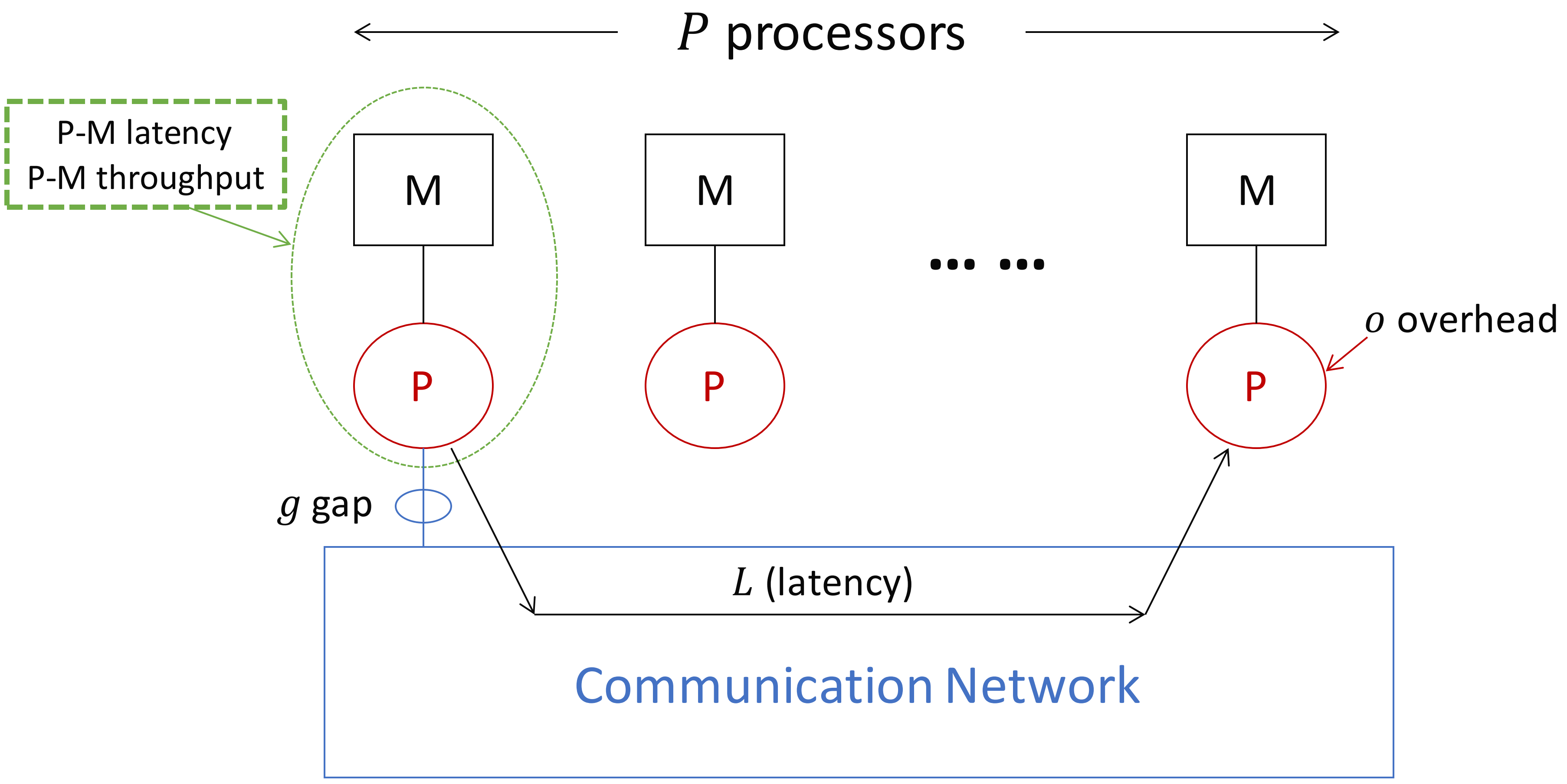
- Paper*: LogP: Towards a Realistic Model of Parallel Computation* EECS Department, University of California, Berkeley

- Authors: David E. Culler, Richard Karp, David A. Patterson, *et al.*

- Published in: Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming (PPOPP '93), Pages 1-12, New York, NY, USA, 1993

- Link: http://dl.acm.org/citation.cfm?id=155333

LogP Model (3)



P processors
P-M latency
P-M throughput
M
M
M
… …
o overhead
P
P
P
g gap
L (latency)
Communication Network

LogP Model (4)

- L: an upper bound on the *latency*, or delay, incurred in communicating a message containing a word (or a small number of words) from its source module to its target module
- o: the *overhead*, defined as the length of time that a processor is engaged in the transmission or reception of each message
- g: the *gap,* defined as the minimum time interval between consecutive message transmissions or consecutive message receptions at a processor
- P: the number of processors/memory modules

LogP Model (5)

Unit of Time

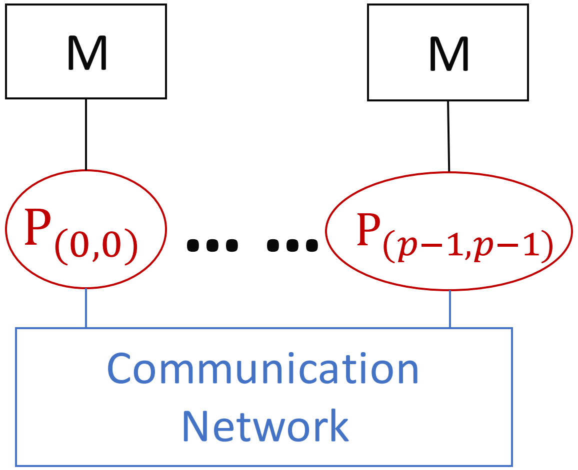
- 1 unit= 1 processor cycle
- L, o, g are all multiples of this
- For example:
  - processor cycle = 0.5 nsec (= 1 unit)
  - L = 1000 cycles (depends on communication type: blocking, buffering)
  - o = 100 cycles
  - g = 1 cycle

LogP Model (6)

Example

Matrix multiplication $(C = A \times B)$ on a $p \times p$ 2-D mesh of processes

- Matrix size = $n \times n$
- A, B, C are partitioned into $\frac{n}{p} \times \frac{n}{p}$ blocks
- $\mathrm{P}_{(i,j)}$ owns $A_{(i,j)}$ and $B_{(i,j)}$ in local memory $(0 \leq i, j < p)$
- $\mathrm{P}_{(i,j)}$ is responsible for computing $C_{(i,j)}$



LogP Model (7)

Example Matrix

	$0,0$	…	$0, p-1$	
n		…		
	$p-1,0$	…	$p-1, p-1$	$\frac{n}{p}$

Message Passing Program

SPMD Model

Process (i, j)

$C_{(i,j)} \leftarrow 0$ // $\frac{\mathrm{n}}{p} \times \frac{n}{p}$ block

Do $k = 0$ to $p - 1$

Read $A_{(i,k)}$ from $P_{(i,k)}$ // $\frac{\mathrm{n}}{p} \times \frac{n}{p}$ block

Read $B_{(k,j)}$ from $P_{(k,j)}$ // $\frac{\mathrm{n}}{p} \times \frac{n}{p}$ block

$C_{(i,j)} \leftarrow C_{(i,j)} + A_{(i,k)} \otimes B_{(k,j)}$

End

$\otimes$: MM

Total Execution Time (1)

LogP Model

Example:

- Processor 2 GHz
- L = 100 cycles = 50 nsec
- o = 1000 cycles = 500 nsec
- g = 1 nsec 1 GWords/sec bandwidth/link

Total Execution Time (2)

LogP Model

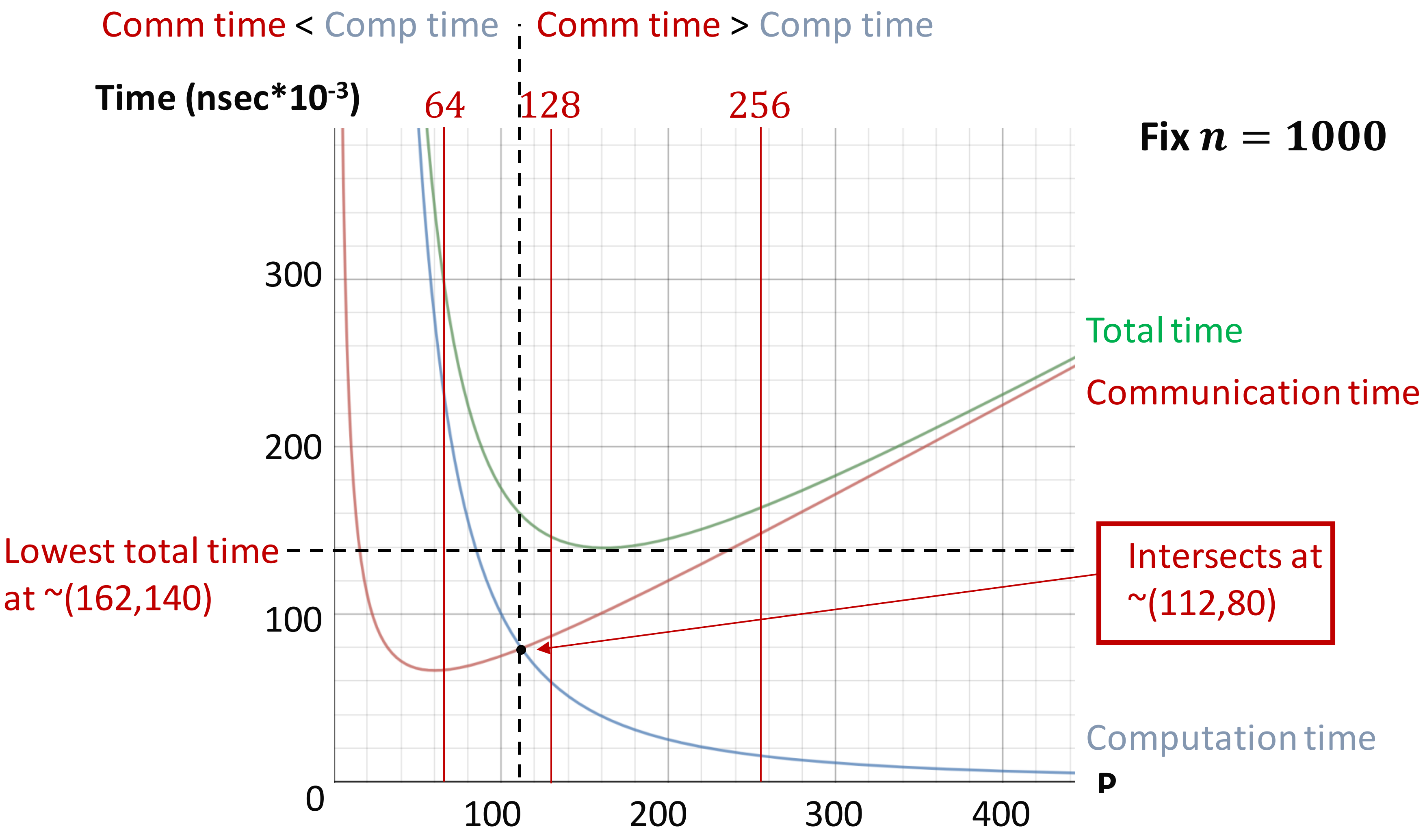
- Total communication time = $p * \left[50 + 500 + 2\left(\frac{n^2}{p^2}\right)\right]$ nsec

- Total computation time = $\left[p \cdot 2\left(\frac{n}{p}\right)^3\right] * 0.5$ nsec

  (Streaming memory access) Best case

- If $\frac{n}{p}$ is large, computation time dominates (Data reuse is high)

Total Execution Time (3)



Routing in Interconnection Networks (1)

- Routing Mechanism
  - Path to be taken from source to destination of a message
  - Based on source nodes , destination nodes, state of the network (for example: level of congestion)
- Minimal routing — shortest path from source to destination
  - Note: minimal routing may result in congestion when several messages are routed concurrently
    - Ex: matrix transpose in a 2-D mesh
- Adaptive routing — routing based on network state
  - Ex: to avoid or minimize network congestion

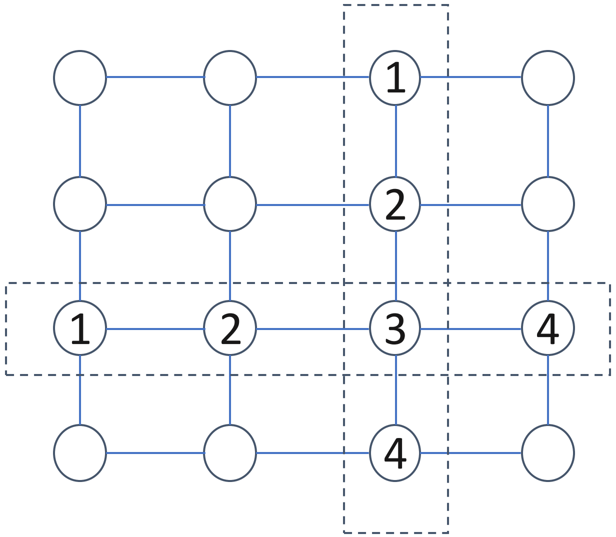
Routing in Interconnection Networks (2)

- Ex: Dimension ordered routing
  - X-Y routing on a 2-D mesh (without wrap around)
  - For each message go along x axis to its destination column and then travel along the column to reach the final destination

This is minimal routing

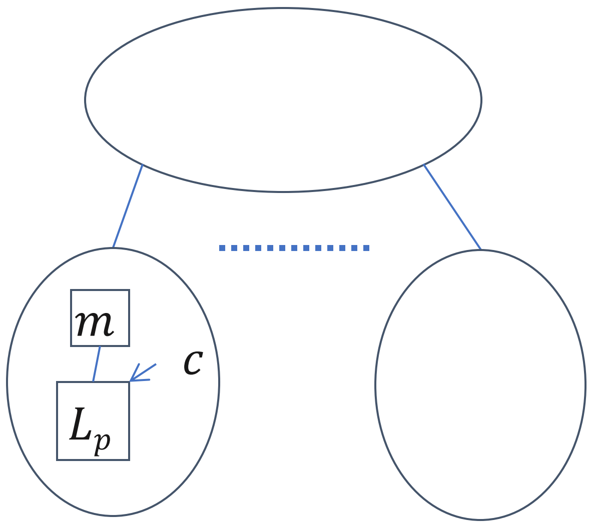
For example: Matrix transpose
Assume
one step – communication along x axis
communication along y axis

$n \times n$ mesh
i-th row data passes through PE(i, i)
PE(i, i) needs $\Theta(n)$ storage



Communication Cost in Shared Address Space Machines (1)

- Cache coherent multi-processors
  - Hardware mechanisms to support access to variables across processors
  - User does not explicitly manage data placement or communication
  - Compilation and run time support for optimization



Communication Cost in Shared Address Space Machines (2)

- Communication cost is hard to model
  - Data layout determined by the system (compiler, run time system)

    Cost of access to local data << remote data
  - Cache behavior

    Data used by a process can be >> cache size

    Cache coherence protocol
  - Compiler optimizations

    Ex: prefetching
  - Runtime optimizations

    Ex: process migration

    data migration

Summary

- Communication Cost in Message Passing Machines
  - – Store and forward
  - – Cut through
- Cost models
  - $t_{comm} = t_s + l \cdot t_h + t_w \cdot m$
  - $t_{comm} = t_s + t_w \cdot m$
- LogP Parallel Machine Model
- Routing in interconnection networks
- Communication Cost in Shared Address Space Machines