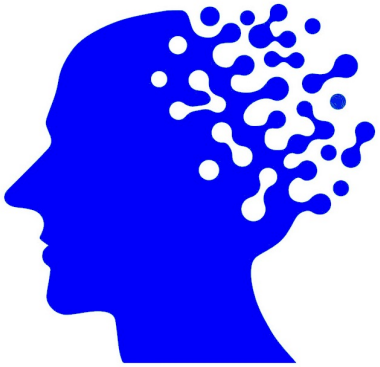


Spring 2023

Distributed Computing

- A Model of Distributed Executions

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

12th January 2023 (http://rajendra.2power3.com)

> About this Lecture

What do we learn today?

- This Lecture covers the essential aspects (of Distributed Algorithms / Systems) that every serious programmer needs to know about
  - **The Fundamentals of Distributed Algorithms**
  - **Design principles** and **Analysis**

with an emphasis on certain properties of

- **The Scalable Application Development**

Let us explore these topics ➔ ➔ ➔

Today's Focus

> Distributed Computing?

- How will you design a Distributed Algorithm?

- Learn to Solve using Distributed Algorithms

Recap: What do we learn?

Distributed Computing (DC)

- ➔ Core Theoretical Concepts
- ➔ Design Principles of DC
- ➔ Discrete Events Simulations
- ➔ Experimental Evaluations
- ➔ Designing Efficient Solution(s) ??
  - ➔ To Solve Some Interesting Problems !!

➔ An Overview of Distributed Computing

➔ ➔ Simple to advanced?

Recap: Distributed Systems

A Distributed System:

- A collection of independent systems that appears to its users as a single coherent system
- A system in which hardware and software components of networked computers communicate and coordinate their activity only by passing messages
- A computing platform built with many computers that:
  - Operate concurrently
  - Are physically distributed (have their own failure modes)
  - Are linked by a network
  - Have independent clocks

Recap: Characteristics

- Concurrent execution of processes:
  - Non-determinism, Race Conditions, Synchronization, Deadlocks, and so on
- No global clock
  - Coordination is done by message exchange
  - No Single Global notion of the correct time
- No global state
  - No Process has a knowledge of the current global state of the system
- Units may fail independently
  - Network Faults may isolate computers that are still running
  - System Failures may not be immediately known

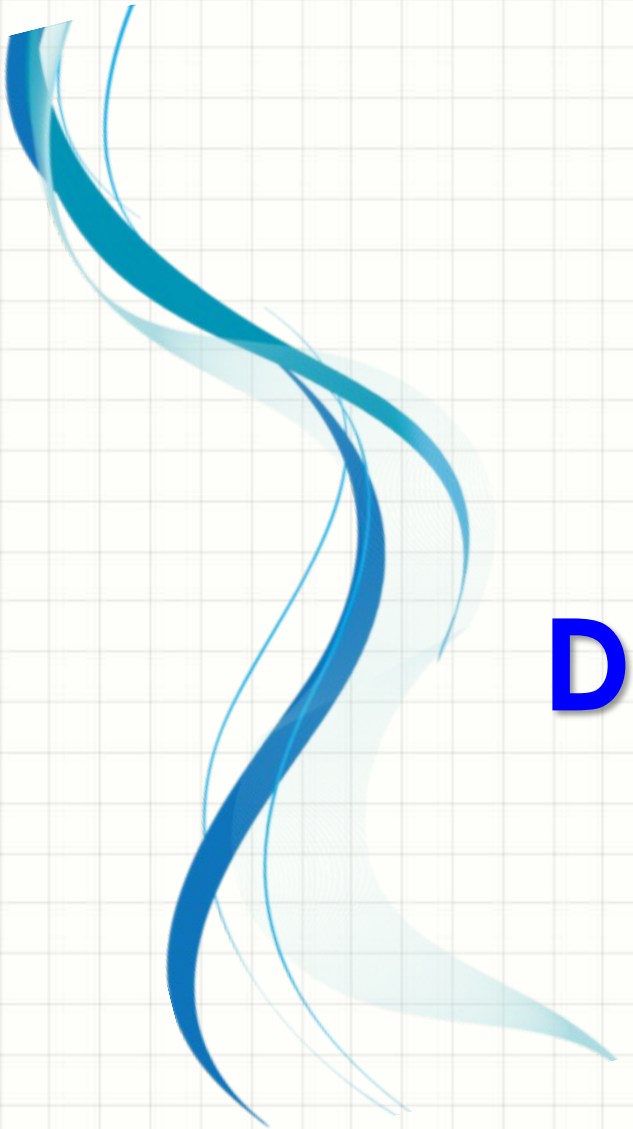
Recap: Challenges and Goals

- Some Important aspects of DC:
  - Reliable network
  - Zero Latency
  - Infinite Bandwidth
  - Secure network
  - Fixed Topology,
  - Only one administrator
  - Zero Transport cost
  - Homogeneous Network
- Remember these points while developing scalable applications

Recap: Challenges / Goals of a DS

What are the challenges / goals of distributed systems?

- Heterogeneity
- Openness
- Security
- Scalability
- Failure Handling
- Concurrency
- Transparency



FUNDAMENTALS OF DISTRIBUTED SYSTEMS

Can you parallelize all apps?

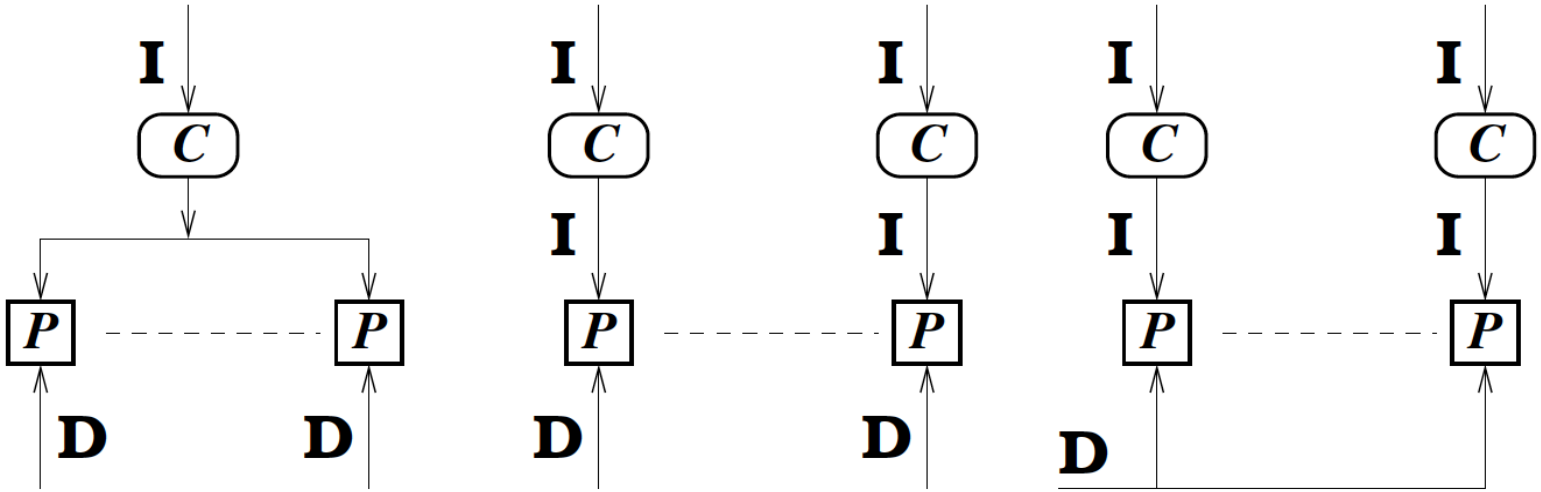
- What can you parallelize?
  - Identify Instructions that could execute in parallel?

- If the proportion of an application that you can parallelize is x ($0<x<1$),
  Maximum speed up = $1/(1-x)$
- Let us assume that a task needs t_s time to run on one machine. How much time does it take to run the same task on p CPUs?
  - Let the running time be t_p of a task running on p CPUs.
  - Now $t_p=t_o+t_s*(1-x+x/p)$, where t_o is the overhead added due to parallelisation (communication, synchronization, etc.)

Flynn's Classification

- Single Instruction Single Data (SISD) Stream
  - Traditional von Neumann architecture
- Single Instruction Multiple Data (SIMD) Stream
  - Scientific Applications, Vector Processors, array processors and so on
- Multiple Instruction Single Data (MISD) Stream
  - Visualization is an example
- Multiple Instruction Multiple Data (MIMD) Stream
  - Distributed Systems

Flynn's Classification



(a) SIMD *(b) MIMD* *(c) MISD*

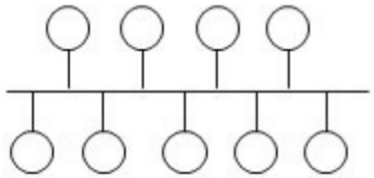
C *Control Unit*

P *Processing Unit*

I *instruction stream*

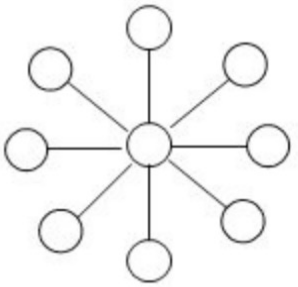
D *data stream*

Interconnection Topologies

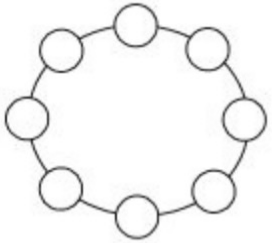


a) Bus

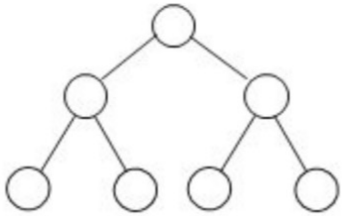
b) Linear array



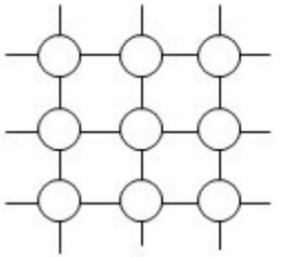
c) Star



d) Ring

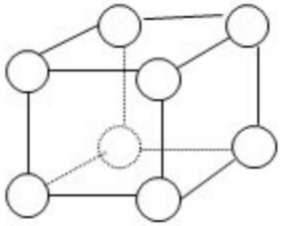


e) Tree



f) Near-neighbor mesh

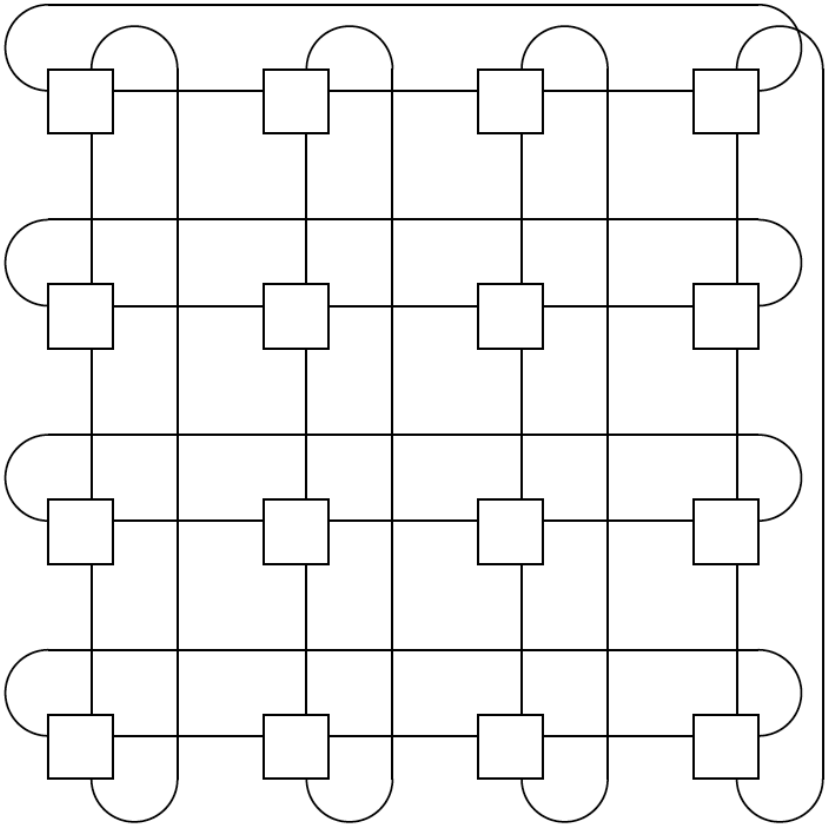
g) Completely connected



h) 3–cube (hypercube)

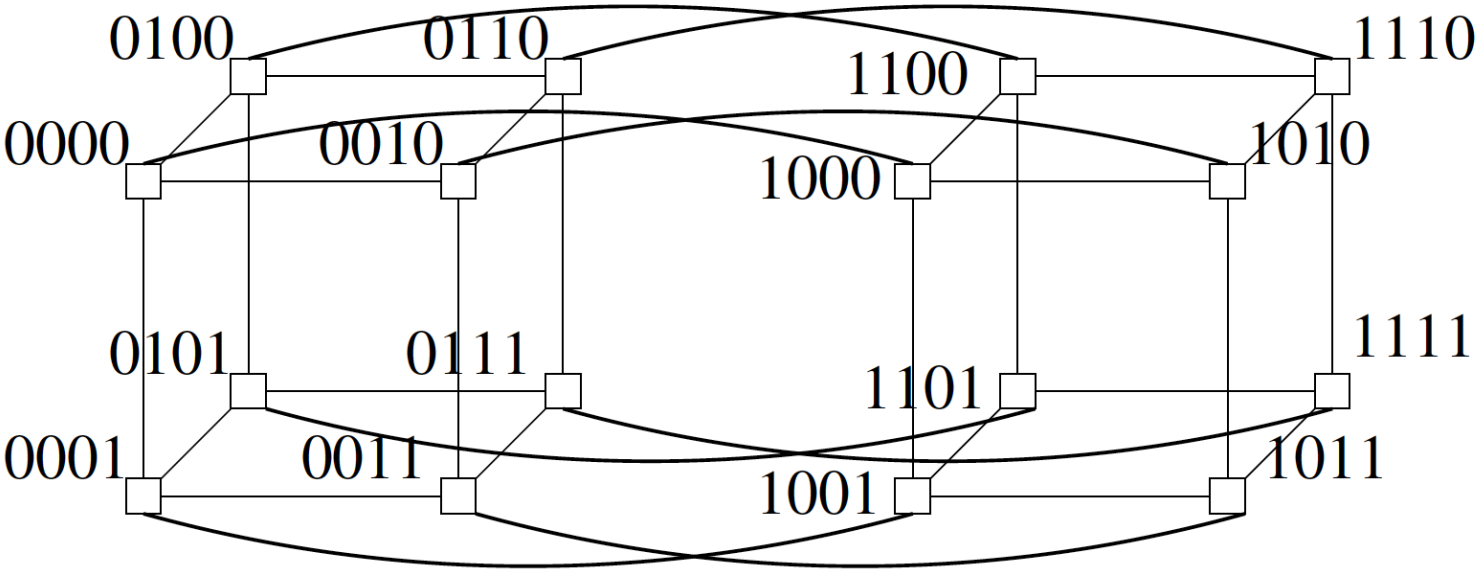
Interconnection Topologies (contd)

➔ Wrap – around mesh (torus)



k-ary d-cubes

➔ Hypercube of Dimension 4



➔ k-ary d-cudes (generalized version)

Message Passing Systems

Basic Primitive Operations

- Send
  - *send* message from process A to Process B

    A → B
- Receive
  - *receive* message at Process B from Process A

    B ← A
- Compute at A and / or B
  - do the specific computations at A and / or B

Message Passing vs. Shared Memory

- Emulate MP on SM
  - Partition the shared address space
  - Send / Receive messages via shared address space
- Emulate SM on MP
  - Model each shared location as a separate process
  - Send: Write to the shared object by sending messages to owner process for the object
  - Receive: Read from shared object by sending query to the owner process of the object

Synchronous vs. Asynchronous

- Synchronous (send / receive)
  - Handshake between sender and receiver
  - *send* completes only when *receive* completes
  - receive completes only when copying of the data to the buffer is over
- Asynchronous (send)
  - No need for handshake between sender and receiver
  - Control returns to the invoking process when data copied out of user-specified buffer

Blocking vs. non-blocking

- Blocking
  - Control returns to the invoking process after the the task completes
- Non-blocking
  - Control returns to the invoking process when data copied out of user-specified buffer
  - *send* completes even before copying the data to the user buffer
  - *receive* may happen even before data may have arrived from the sender
  - How to order EVENTs?

A Distributed Program

- A distributed program is composed of a set of n asynchronous processes, $p_1, p_2, \ldots, p_i, \ldots, p_n$
- The processes do not share a global memory and communicate solely by passing messages
- The processes do not share a global clock that is instantaneously accessible to these processes
- Process execution and message transfer are asynchronous
- Without loss of generality, we assume that each process is running on a different processor
- Let C_{ij} denote the channel from process p_i to p_j and let m_{ij} denote a message sent by p_i to p_j
- The message transmission delay is finite and unpredictable

A Model of Distributed Executions

- The execution of a process consists of a sequential execution of its actions.
- The actions are atomic and modeled as three types of events: internal events, message send events, and message receive events
- Let e_i^x denote the x^{th} event at process p_i .
- For a message m, let send(m) and receive(m) denote send and receive events, respectively.
- The occurrence of events changes the states of respective processes and channels.
- Internal event → changes state of the process
- Send and Receive events change the state of the process that sends / receives the message & the state of the channel on which the message is sent / received respectively

A Model of Distributed Executions

- The events at a process are linearly ordered by their order of occurrence.
- The execution of process p_i produces a sequence of events $e_i^1, e_i^2, \ldots, e_i^x, e_i^{x+1}, \ldots$ and is denoted by H_i where

$$H_i = (h_i, \rightarrow i)$$

h_i is the set of events produced by p_i and binary relation $\rightarrow i$ defines a linear order on these events

- **Linear Relation:** Mathematically, the independent variable is multiplied by the slope coefficient, added by a constant, which determines the dependent variable
- Relation $\rightarrow i$ expresses causal dependencies among the events of p_i

A Model of Distributed Executions (contd)

- The send and the receive events signify the flow of information between processes and establish causal dependency from the sender process to the receiver process
- Define a relation $\rightarrow_{msg}$ that captures the causal dependency due to message exchanges as follows:

  For every message m that is exchanged between two processes, we have

$$send(m) \rightarrow_{msg} receive(m)$$

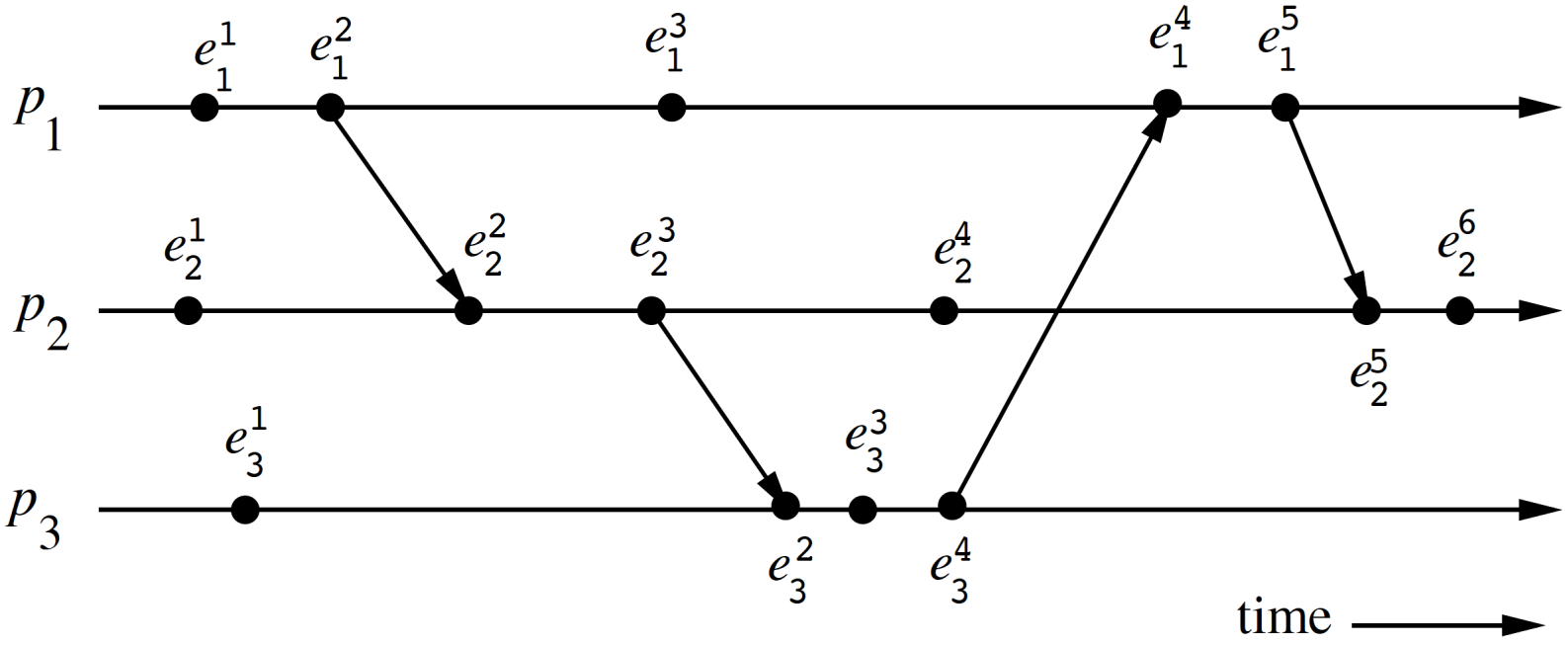
- Relation $\rightarrow_{msg}$ defines causal dependencies between the pairs of corresponding send and receive events

A State-Time diagram

- The evolution of a distributed execution is depicted by a space-time diagram
- A horizontal line represents the progress of a specific process
- A dot indicates an event
- A slant arrow indicates a message transfer

- Since an event execution is atomic (indivisible and instantaneous), it is justified to denote it as a dot on a process line

A State-Time diagram – An Example



➔ **For Process p_1:**

Second event is a message send event

First and Third events are internal events

Fourth event is a message receive event

Applications

- Mobile Systems
- Sensor networks
- Pervasive Computing
  - Smart workplace
  - Intelligent Home
- Peer-to-peer computing
- Distributed Agents
- Distributed Data Mining
- Grid Computing
- Security aspects in Distributed Systems

Summary

Focused on exploring the following:

- Goals and Challenges of DS
  - Fundamental aspects while building distributed applications
- A model of Distributed Computations
  - Primitives of Distributed Communications
    - Message Passing is the main focus
  - Properties of distributed Computations
  - Events and their ordering
    - How to handle Causal Precedence ?
    - Lamport's Logical Clocks ?
    - Many more to come up … stay tuned in !!

Penalties

- Every Student is expected to strictly follow a fair Academic Code of Conduct to avoid penalties

- Penalties is heavy for those who involve in:
  - Copy and Pasting the code
  - Plagiarism (copied from your neighbor or friend – in this case, both will get "0" marks for that specific take home assignments)
  - If the candidate is unable to explain his own solution, it would be considered as a "copied case"!!
  - Any other unfair means of completing the assignments

Help among Yourselves?

- **Perspective Students** (having CGPA above 8.5 and above)
- **Promising Students** (having CGPA above 6.5 and less than 8.5)

- **Needy Students** (having CGPA less than 6.5)
  - Can the above group help these students? (Your work will also be rewarded)

- You may grow a culture of **collaborative learning** by helping the needy students

How to reach me?

➔ Please leave me an email:

rajendra [DOT] prasath [AT] iiits [DOT] in

➔ Visit my homepage @

➔ https://www.iiits.ac.in/people/regular-faculty/dr-rajendra-prasath/

(OR)

➔ http://rajendra.2power3.com

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may ask for one-to-one meeting

Best Approach

- You may leave me an email any time
  (email is the best way to reach me faster)

Questions
It's Your Time

Contact Information

Dr. Rajendra Prasath
IIIT Sri City, Chittoor