A Model of Distributed Computations

Course: Distributed Computing

Faculty: Dr. Rajendra Prasath

Spring 2019

About this Course

This course covers essential aspects that every serious programmer needs to know about **the model of distributed computations, challenges and goals of Distributed Systems, Distributed Sorting on a line network and so on**

RECAP

What did you learn so far?

- 8 Important aspects of DS
  - Reliable network, Zero Latency, Infinite Bandwidth, Secure network, Fixed Topology, , Only one administrator, Zero Transport cost, Homogeneous Network
- Flynn's Classification
  - SISD, SIMD, MISD and, MIMD
- Interconnection Networks
  - Various Networks: Bus, Line, Ring, Star, Mesh, Torus, Tree, Hypercubes, k-ary d-cubes and so on
- Message Passing vs. Shared Memory systems
  - Synchronous vs. Asynchronous
  - Blocking vs. non-blocking

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
    - Perform specific internal 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?

Distributed Computing

- ➔ A Study of Distributed Systems

Define a Distributed System?

- ➔ A model in which components communicate among themselves by passing messages and coordinate (regulated by interaction or interdependence) to accomplish a specific task / problem given to them

Challenges and Goals with Distributed Systems

Challenges / Goals with DS

What are the challenges / goals with distributed systems?

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

Heterogeneity

Heterogeneity (= the property of consisting of different parts) applies to:

- ➔ Networks, Computers, Operating Systems, Languages, and so on
- ➔ Data types, such as integers, may be represented differently
- ➔ Application program interfaces may differ

Middleware: a software layer that provides a programming abstraction to mask the heterogeneity of the underlying platforms (networks, languages, H/W, ...)

- ➔ E.g. Java RMI

Openness

Each system is open to interaction with other systems

→ Key software interfaces are made publicly available.

→ E.g. Web services to support interoperable machine to machine interaction over a network

Properties:

- Once something is published, cannot be taken back
- No central arbiter of truth - different subsystems have their own
- Unbounded non-determinism: The amount of delay in servicing a request can become unbounded, still guaranteeing - requests will eventually be serviced

Security

Three aspects:

- Confidentiality (protection against disclosure to unauthorised individuals)
- Integration (protection against alteration or corruption)
- Availability (protection against interference with the means to access the resources)

Encryption techniques (cryptography) answer some of the challenges. Still:

- Denial of Service (DoS) Attacks: A service is bombarded by a large number of pointless requests

Scalability

When a system is said to be scalable?

If the system remains effective, without disproportional performance loss, when there is an increase in the number of resources, the number of users, or the amount of input data

- ➔ Factors: load, geographical distribution, number of different organizations and so on

Challenges:

- ➔ Control the cost of physical resources & performance loss
- ➔ Prevent software resources running out
- ➔ Avoid performance bottlenecks: use caching, replication

Fault-Tolerance

Some Components may fail while others continue executing.

We need to:

- Detect failures: use checksums to detect corrupted data
- Mask failures: retransmit a message when it failed to arrive
- Tolerate failures: do not keep trying to contact a web server if there is no response
- Recover from failures: make sure that the state of permanent data can be recovered
- Build redundancy: Data may be replicated to ensure continuing access

Concurrency

- Several clients may attempt to access a shared resource at the same time
- Requires proper synchronisation to make sure that data remains consistent

- Lot more to learn about this in DC … !!

Transparency

Any distributed system appears and functions as a normal centralized system … e.g, DeepBLUE system (30 nodes system)

- Access transparency: resources are accessed in a single, uniform way
- Location transparency: users should not be aware of where a resource is physically located
- Concurrency transparency: multiple users may compete for and share a single resource: this should not be apparent to them
- Replication transparency: even if a resource is replicated, it should appear to the user as a single resource (without knowledge of the replicas)
- Failure transparency: always try to hide any faults
- And more: mobility, performance, scaling, persistence, security, etc

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

Distributed Sorting – An example

Why Sorting?

Fundamental problem in computing

Distributed Sorting (DS):

- ➔ Initially, each process P_i has an element s_i for sorting
- ➔ n Elements are arranged in a Line network
- ➔ Position of each element has to be rearranged to satisfy the condition

$$s_i \leq s_i + 1$$

in each process P_i , $1 \leq i < n$, at the final state

- ➔ Find a strategy to minimize the amount of communication (in terms of the number of message exchanges)

Odd-Even Transposition Sort

Odd-Even Transposition Sorting - (n) rounds

(odd - i): $P_{i\ (=odd,\ )}$ $(v_i) \leftarrow\rightarrow P_{i+1}$ (v_{i+1}), if $v_i > v_{i+1}$
(even - i): $P_{i\ (=even,\ )}$ $(v_i) \leftarrow\rightarrow P_{i+1}$ (v_{i+1}), if $v_i > v_{i+1}$

Requires knowledge about Global position

Example:

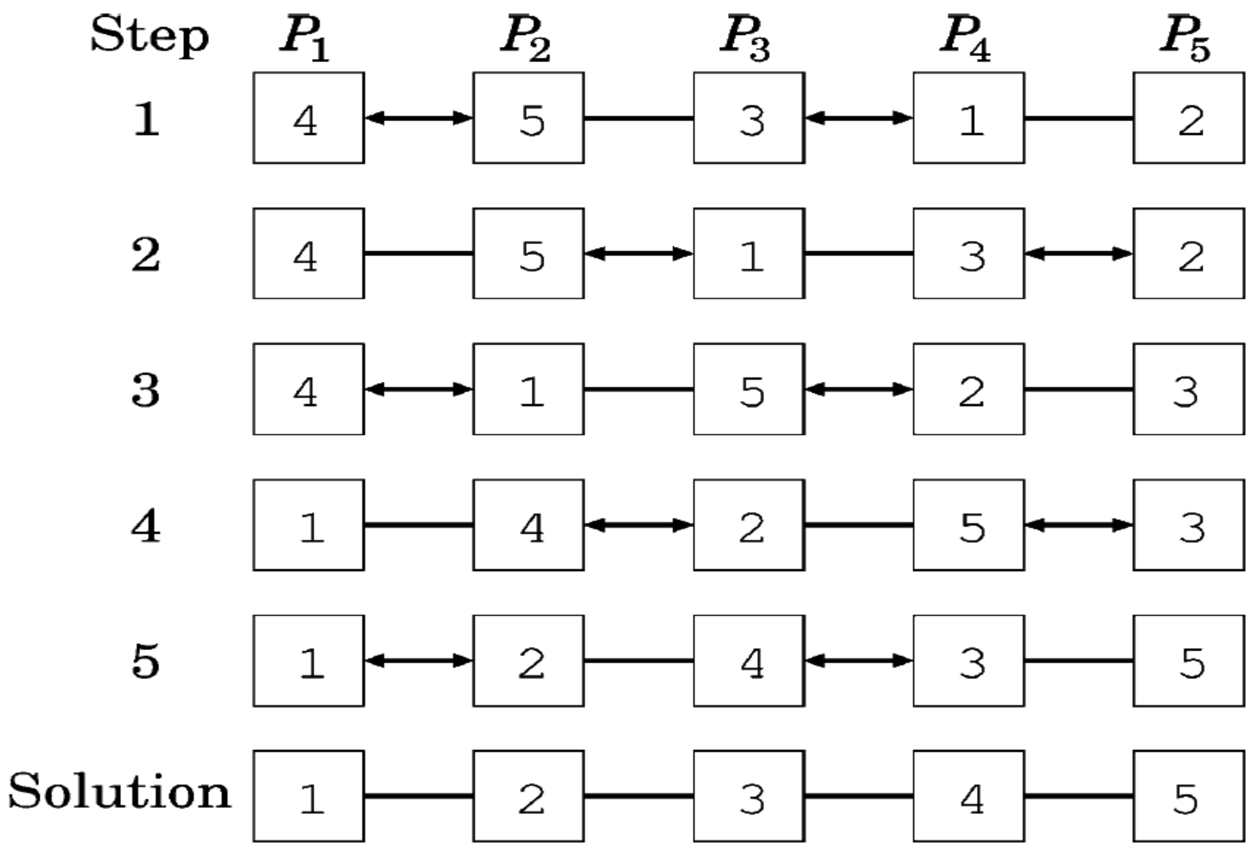
- Consider Sorting of 5 elements

  4 5 3 1 2

- Each element is kept with a process P_i
- Line network – the underlying network that connects all processes

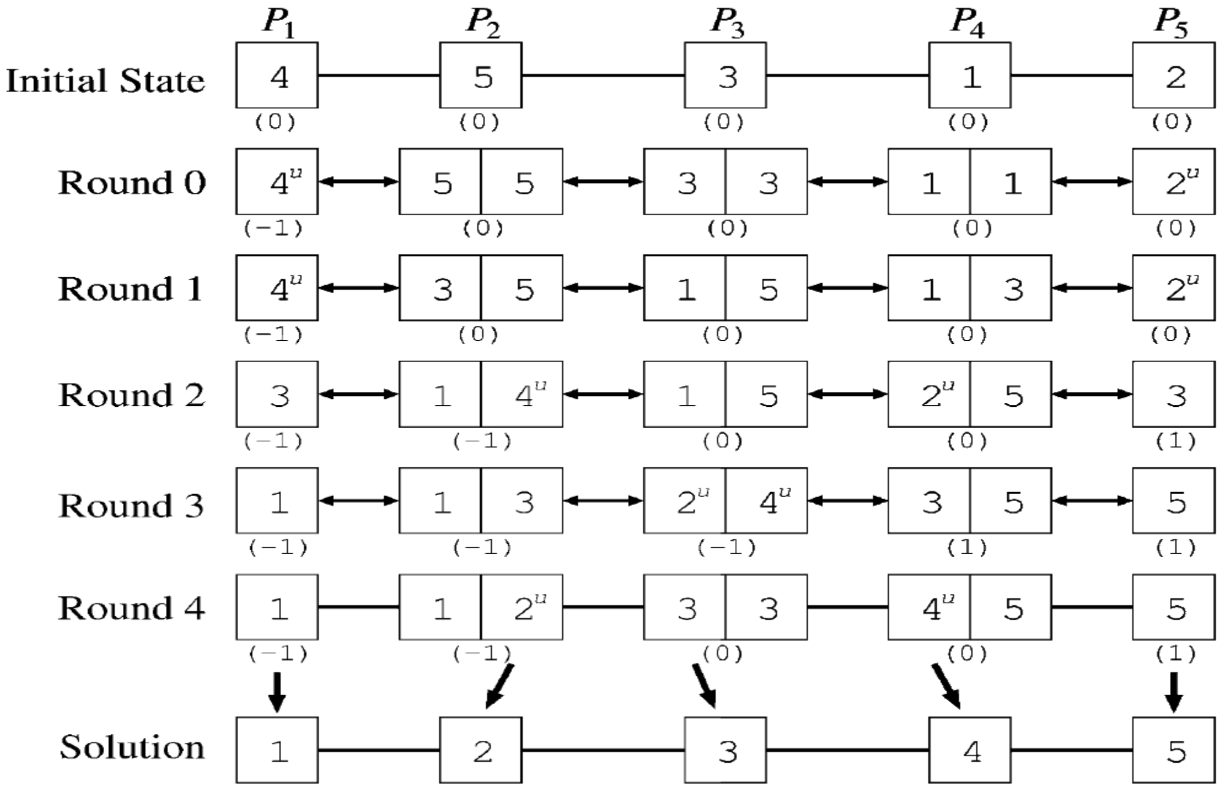
Odd-Even Transposition Sort

An Illustrative Example:



Distributed Sorting – Sasaki's (n-1) round

[Sasaki, 2002]



No Global position; Make copies of elements at intermediate nodes; Rule to select Final Solution; Computing n at runtime

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

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

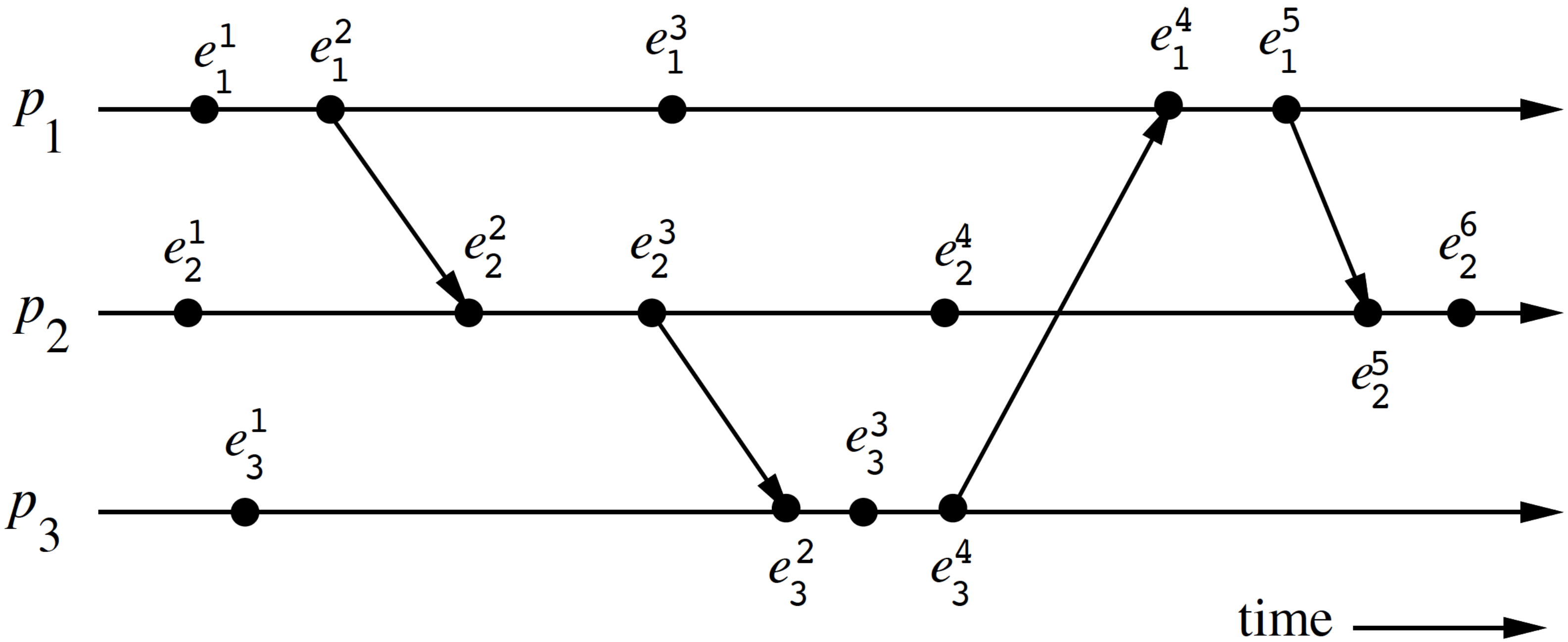
- Relation $\rightarrow_{msg}$ defines causal dependencies between the pairs of corresponding send and receive events

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

A Few 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

- 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
  - Distributed Sorting
  - Events and their ordering
    - How to handle Causal Precedence ?
    - Lamport's Logical Clocks ?
    - Many more to come up … stay tuned in !!

How to reach me?

- Please leave me an email:
  rajendra [DOT] prasath [AT] iiits [DOT] in

- Visit my homepage @

  - http://www.iiits.ac.in/FacPages/index-rajendra.html

  OR

  - http://rajendra.2power3.com

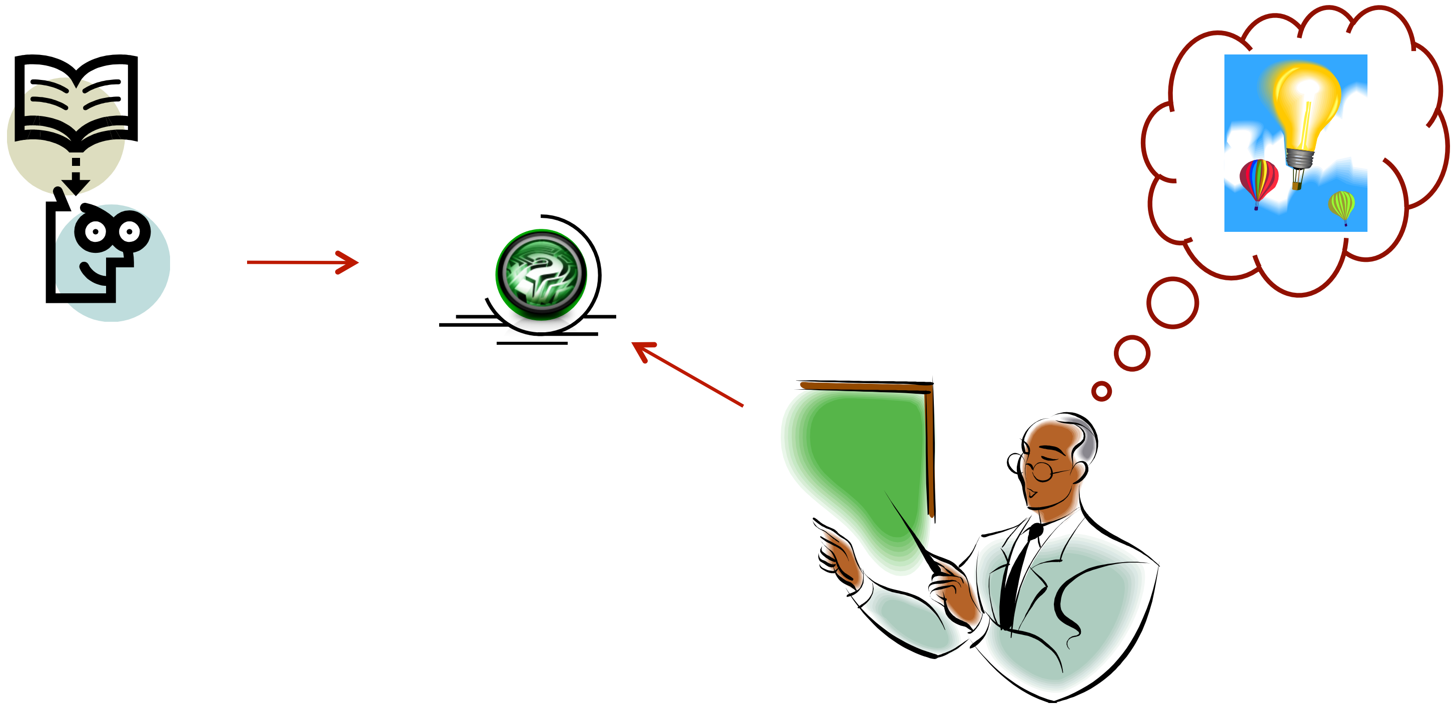
Help among Yourselves?

- **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

Thanks …



… Questions ???