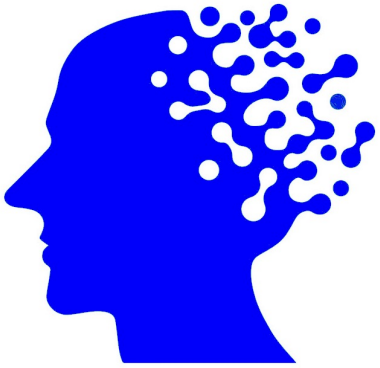


Spring 2024

Distributed Computing
- Course Introduction

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

03rd January 2024 (www.2power3.com/rajendra)

> Heartiest Welcome to ALL

- Welcome to the
  Distributed Computing course

IIITS Welcomes YOU ALL

> Distributed Computing?

- How will you design a distributed algorithm?

- Implement solutions using discrete events / MPI

> About Distributed Computing

- This course covers
  - Essential aspects that every serious programmer needs to know about
    - **Distributed algorithms**
    - **Design principles and their analysis**,

  with emphasis on

    - **Real-time implementations** and
    - **Scalable applications development**

> Scalability vs Efficiency

- How do you find the correct solution?

- Whether do Your Solutions solve the given problem?
- How will you improve you solution so that you can show a better performance?

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?

An Overview

Basic Problem-Solving:

- What are the constraints in solving a specific problem?
- How to do problem-solving in a sequential machine?
- How do we parallelize the solution?
- How to make the systems to co-ordinate to solve the specific problems given the specific constraints?

Distributed Computing

- A study of Algorithms for Distributed Systems

What is 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

- Is it different from parallel processing?

Parallel vs Distributed

- Parallel System?
  - Having n processors with a common shared memory
- Distributed System?
  - Having n processors but NO common shared memory

More Formally …

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

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

Need of Distributed Systems

Why do we need distributed systems?

- People are distributed but need to work together
- Hardware needs to be physically close to people (who are distributed)
- Information is distributed but needs to be shared (trustworthily)
- Hardware can be shared (increases computing power by doing work in parallel; more efficient resource utilization)

Examples of DS

- Intranets, Internet, World Wide Web
- Distributed / Supercomputers
- Grid / Cloud computing – AWS-EC2
- Electronic banking
- Airline Reservation Systems
- Railway Reservation Systems
- Peer-to-peer networks
- Sensor networks – IBM systems
- Web Searching / Web Crawling

. . . and so on

Course Content

- Course is divided into several modules:
- Covers Basics to Advanced Components
  - At least one example problem with detailed analysis in each topic
- Course is supposed to be an interactive course
- Class performance bonus would be given to students who solve a set of problems efficiently

➔ Course Content follows …

Course Content - Topics

- Introduction
- A Model of Distributed Computations
- Logical Time
- Global State/Snapshot Recording Algorithms
- Topology Abstraction and Overlays
- Message Ordering and Group Communication
- Termination Detection
- Distributed Mutual Exclusion Algorithms

Course Content – Topics (contd…)

- Deadlock Detection in Distributed Systems
- Distributed Shared Memory
- Check Pointing and Rollback Recovery
- Consensus and Agreement Algorithms
- Self-Stabilization Algorithms
- Authentication in Distributed Systems
- Peer-to-Peer Computing and Overlay Graphs

and Practice Problems …

Case Studies

- Discrete Event Simulations
  - Distributed Sorting on a Line Network
  - Distributed Sorting on Various Interconnection Networks
- Map Reduce and Big Data
  - How to process a huge volume of data
  - Specific focus would be on scalable data processing especially in text format

- Authentication & Security in DS
  - We will focus more on Decentralized Application development

Examinations

- Mid 1 Semester : 15 Marks
- Mid 2 Semester : 15 Marks
- End Semester : 20 Marks

- Practice Components: 30 Marks
  2 Assignments (20) + One Mini Project (10)

- Total Weightage (100) = Exams (50)
  + Practice Components (30) + Quiz (10)
  + Class Performance (10)

- Academic Code of Conduct
  - Explore PENALTIES

Subject to the Approval of The Class Committee

Assignments

- Two Assignments
- Must be solved by individuals & must be finished before the deadline specified
- All Assignments are COMPULSARY

- Total Weightage: 20%

- Solutions would be cross checked !!
- Solutions submitted after the deadline will not be considered for evaluation
- Submission Procedure would be given.

Mini Projects

- Mini Project spans over 4 weeks
- Must be solved by individuals & must be finished before the deadline specified

- Mini Project is COMPULSARY
- Total Weightage: 10%

- Solutions would be cross checked !!

- Solutions submitted after the deadline will not be considered for evaluation
- Submission Procedure would be given.

Scheduled Quizzes

- There would be at least two Scheduled Quizzes:
  - One before Mid Semester
  - One before End Semester
- Must be answered by individuals
- These quizzes will be conducted online until the physical classes resumes in the campus

- These quizzes are COMPULSARY
- Total Weightage: 10%;
- Date would be announced in prior and responses submitted after the deadline will not be considered for evaluation

Innovative Solutions

- There would be multiple chances:
  - A few would be held before MID Semester
  - A Few would be held after MID Semester
- Must be answered by individuals
- These challenges will be given in the classroom
- Total Weightage: 10%
- Dates WILL NOT BE announced in prior
- Responses submitted after the deadline will not be considered for evaluation

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

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

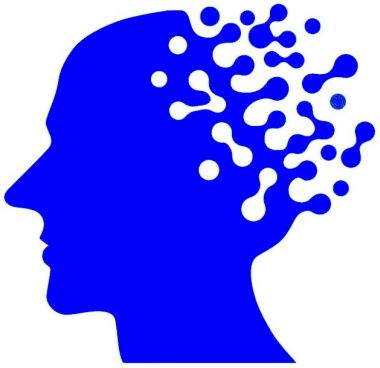
Best Approach

- 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
[rajendra.prasath@iiits.in]
IIIT Sri City, Chittoor



Spring 2024

Distributed Computing
- The Fundamental Aspects

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

10th January 2024 (www.2power3.com/rajendra)

> 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

What do we learn?

- 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

Reliable Network

- Hardware may fail! Power failures; Switches have a mean time between failures

Implications:

- Hardware: weight the risks of failure versus the required investment to build redundancy
- Software: we need reliable messaging: be prepared to retry messages, acknowledge messages, reorder messages (do not depend on message order), verify message integrity, and so on.

Zero Latency

Latency (not bandwidth):

How much time do the data take to move from one place to another? → measured by time

- ➔ The minimum round-trip time between two points on earth is determined by the maximum speed of information transmission: the speed of light.
- ➔ At 300,000 km/sec, it will take at least 30msec to send a ping from Europe to the USA and back

Implications:

- ➔ Strive to make as few calls over the network (other than LANs) as possible

Infinite Bandwidth

Bandwidth: how much data you can transfer over a period of time (may be measured in bits/second)

- ➔ It constantly grows
- ➔ Bandwidth may be lowered by packet loss: we may want to use larger packet sizes

Implications:

- ➔ Compression: simulate the environment to get an estimate for your needs

Secure Network

How to secure the underlying network?

Implications:

- ➔ You may need to build security into your applications from the beginning
- ➔ As a result of security considerations, you might not be able to access networked resources, different user accounts may have different privileges, and so on
- ➔ How to solve these issues efficiently?

Fixed Topology

Topologies do not change as long as you are in a closed environment

- ➔ In reality, servers may be added and removed often, clients (laptops, wireless ad hoc networks) are coming and going: the topology is changing constantly

Implications:

- ➔ Do not rely on specific endpoints or routes
- ➔ Abstract the physical structure of the network: the most obvious example is DNS names as opposed to IP addresses

Who is the Administrator?

Different Administrators may be associated with the network with different degrees of expertise

- ➔ Might make it difficult to locate problems
- ➔ Coordination of upgrades: will the new version of MySQL work as before with Ruby on Rails?
- ➔ Never underestimate the 'human' factor!

Zero Transport Cost

- Going from the application layer to the transport layer (2nd highest in the five layer TCP/IP reference model) is not free:
- Information needs to be serialized (marshalling) to get data onto the wire
- The cost (in terms of money) from setting and running the network is not zero.
- Have we leased the necessary bandwidth

Everything costs "Money" !!

Homogeneous Network

Homogeneous = of the same kind; uniform

- Even a home network may connect a Linux PC and a Windows PC. A homogeneous network today is the exception, not the rule!

Implications:

- Interoperability will be needed
- Use standard technologies such as XML

Summary

Focused on exploring the following:

- Fundamental aspects while building distributed applications
- Necessary Properties of a DS
- Desirable properties of a DS
- Networks and Message Passing Architectures of a DS
- 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

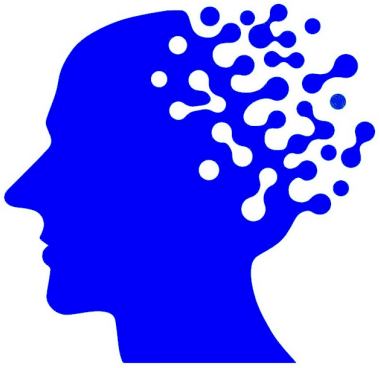
Best Approach

- 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



Spring 2024

Distributed Computing

- Basic Primitive Operations

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

11th January 2024 (www.2power3.com/rajendra)

> Distributed Computing?

- How will you design a Distributed Algorithm?

- Learn to Solve using Distributed Algorithms

> About this Lecture

What do we learn today?

- This covers a model of distributed computations that every algorithm designer needs to know
  - **Challenges and Goals**
  - **A model of distributed executions**

with an application to

- **Distributed Sorting on a line network**

Let us explore these topics ➔ ➔ ➔

Today's Focus

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

Challenges and Goals of Distributed Systems

Challenges / Goals of DS

What are the challenges / goals of 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:

- 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

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

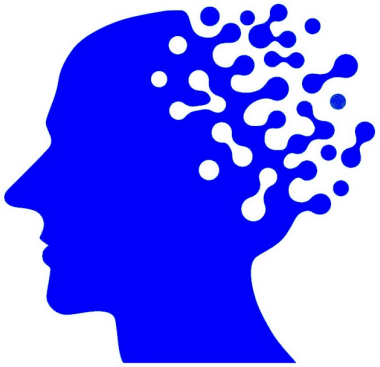
Best Approach

- 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



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

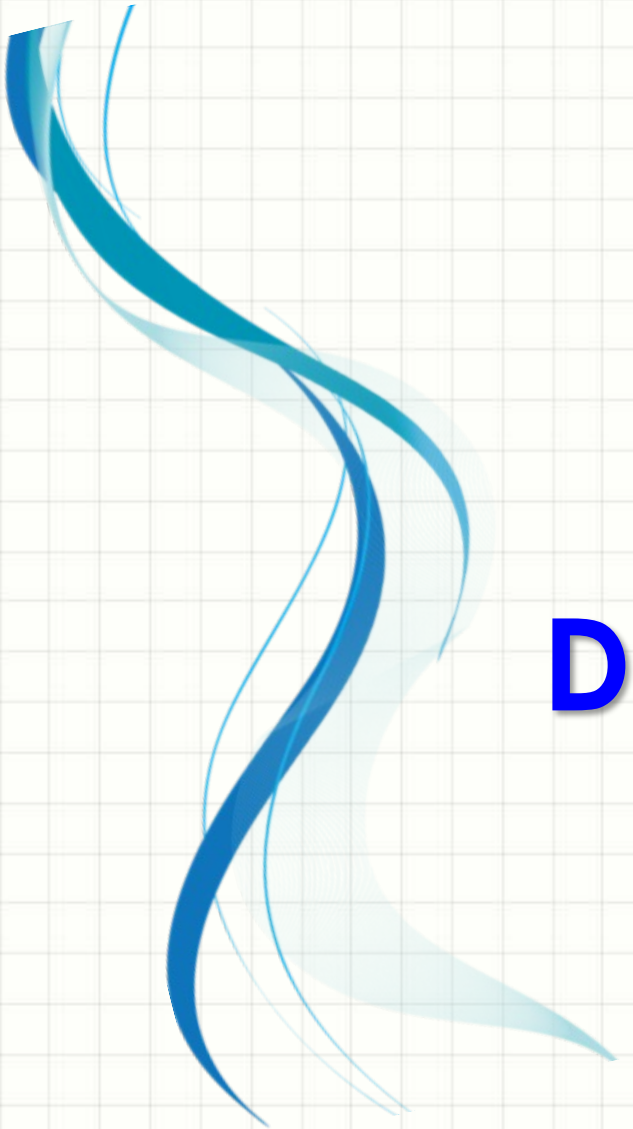
Recap: Challenges and Goals

- 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?

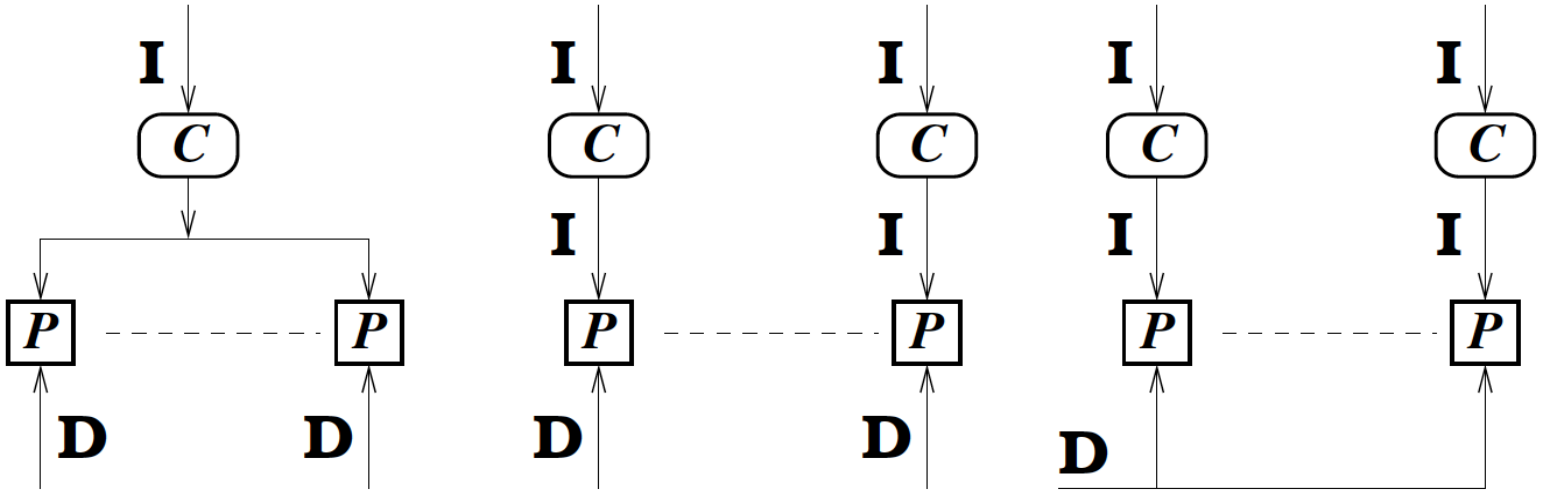
- What can you parallelize?
  - Identify Instructions that could execute in parallel?

- 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*

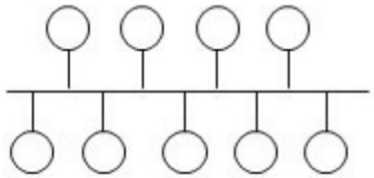
C *Control Unit*

P *Processing Unit*

I *instruction stream*

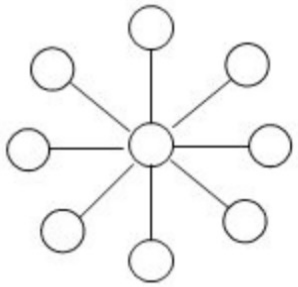
D *data stream*

Interconnection Topologies

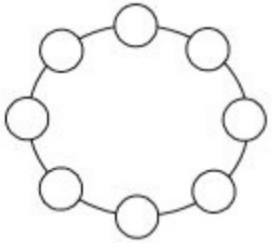


a) Bus

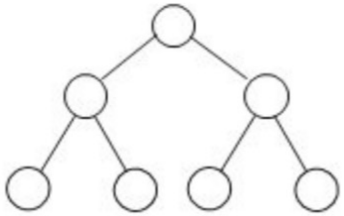
b) Linear array



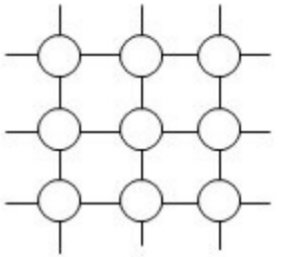
c) Star



d) Ring

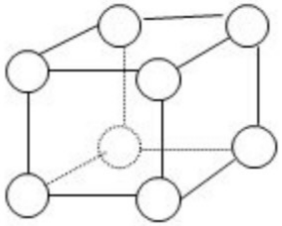


e) Tree



f) Near-neighbor mesh

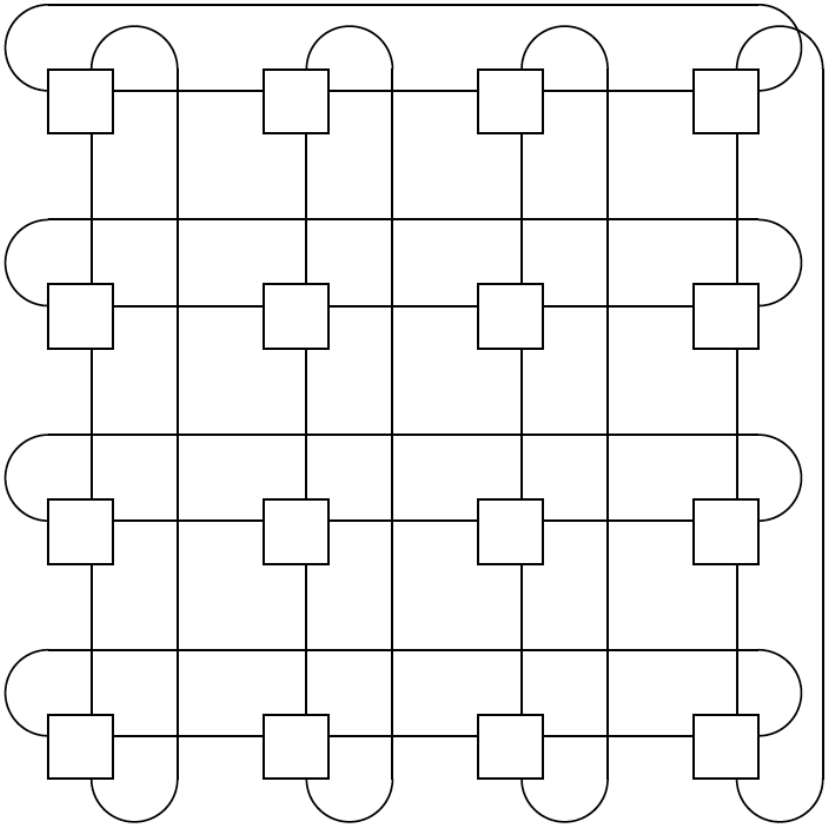
g) Completely connected



h) 3–cube (hypercube)

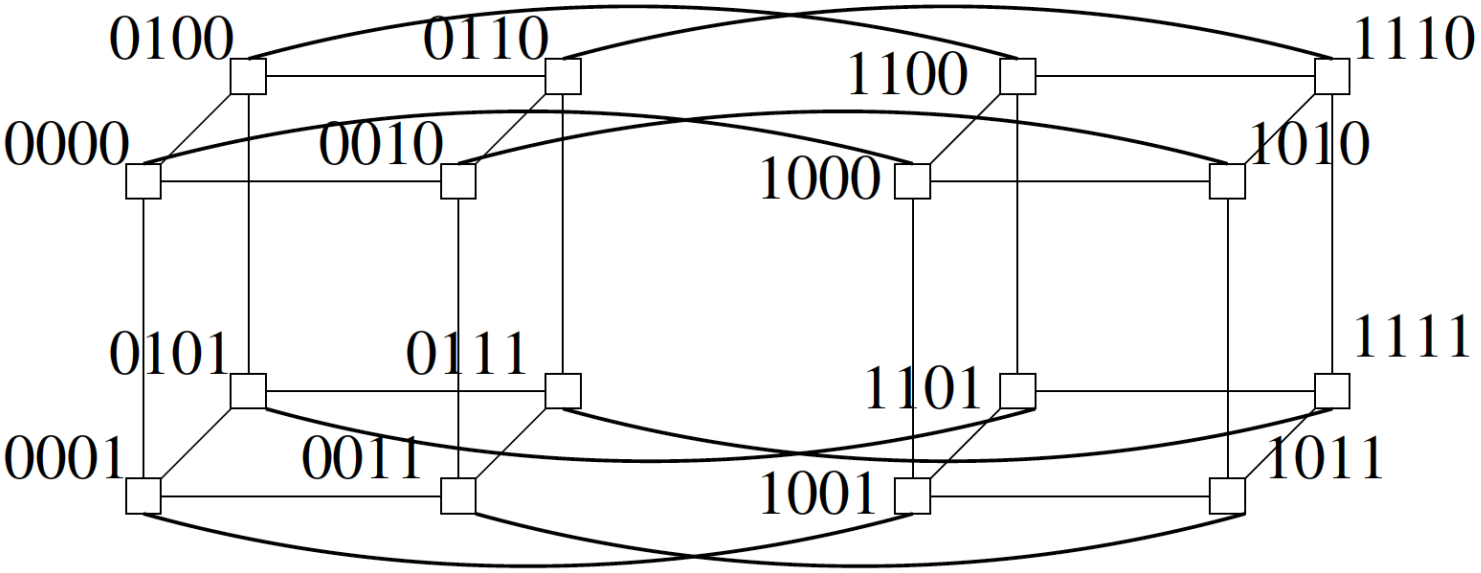
Interconnection Topologies (contd)

➔ 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 , ... , e_i^x , e_i^{x+1}, ... 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

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

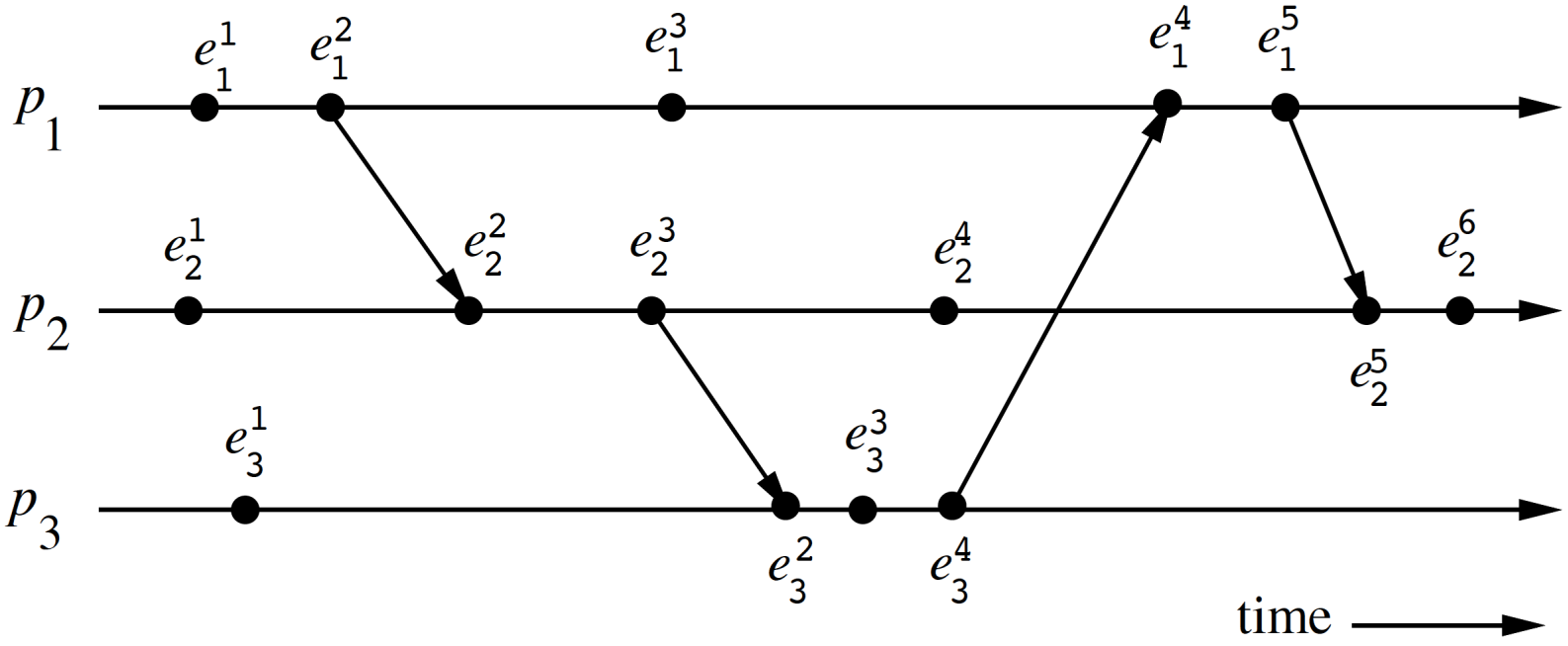
- Relation $\rightarrow_{msg}$ defines causal dependencies between the pairs of corresponding send and receive events

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

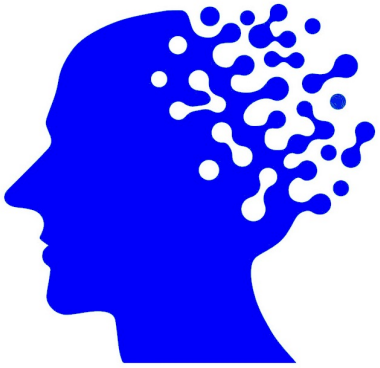
Best Approach

- 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



Spring 2024

Distributed Computing

- Distributed Sorting Algorithms

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

19th January 2024 (http://www.2power3.com/rajendra)

> Distributed Computing?

- How will you design a Distributed Algorithm?

- Learn to Solve using Distributed Algorithms

> About this Lecture

What do we learn today?

- This covers a model of distributed computations that every algorithm designer needs to know
  - **Challenges and Goals**
  - **A model of distributed executions**

with an application to

- **Distributed Sorting on a line network**

Let us explore these topics ➔ ➔ ➔

Today's Focus

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: 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

Recap: 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

Challenges / Goals of DS

What are the challenges / goals of distributed systems?

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

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

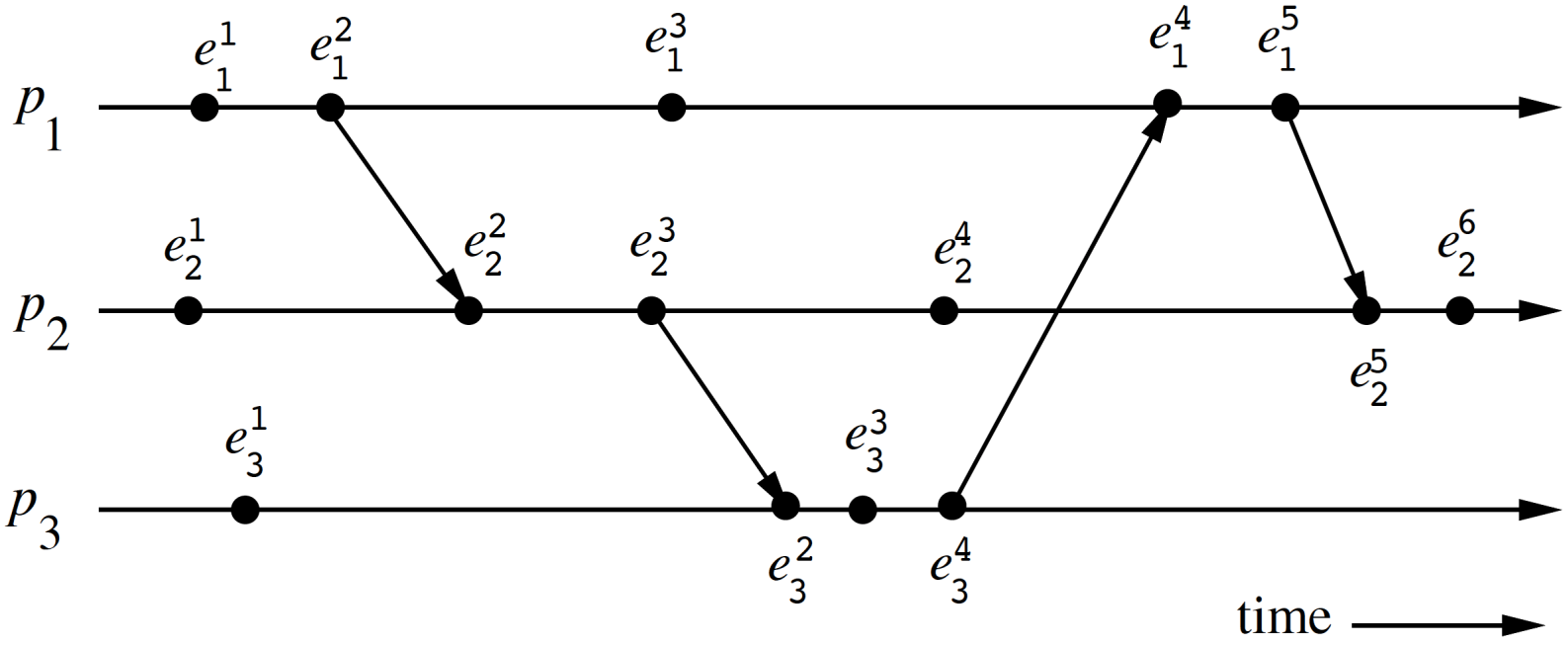
- The execution of a process consists of a sequential execution of its actions.
- The actions are atomic and modeled as three types of events: internal events, message send events, and message receive events
- Let e_i^x denote the x^{th} event at process p_i .
- For a message m, let send(m) and receive(m) 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 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

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

Requires knowledge about Global position

Example:

- Consider Sorting of 5 elements

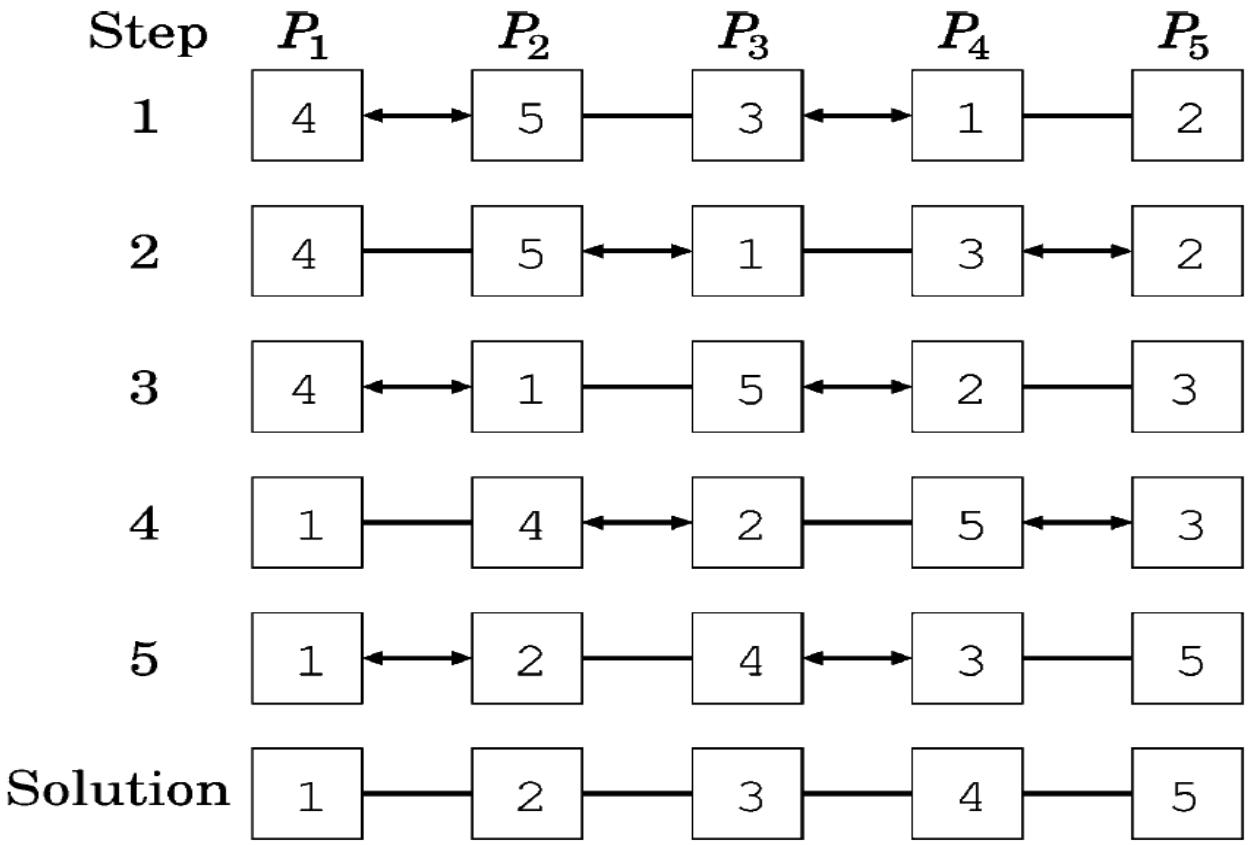
  4 5 3 1 2

- Each element is kept with a process P_i

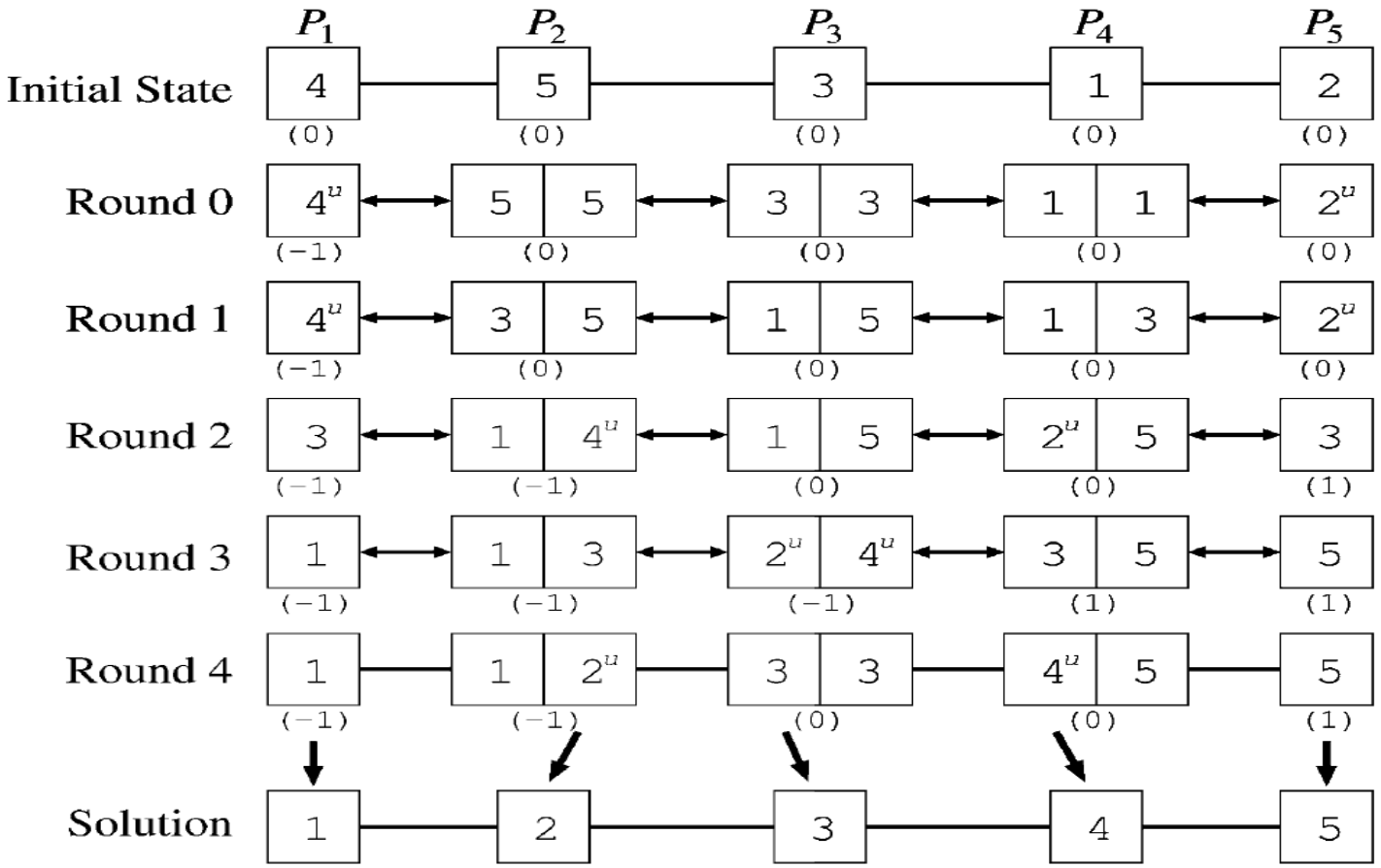
- Line network – the underlying network that connects all processes

Odd-Even Transposition Sort

An Illustrative Example:



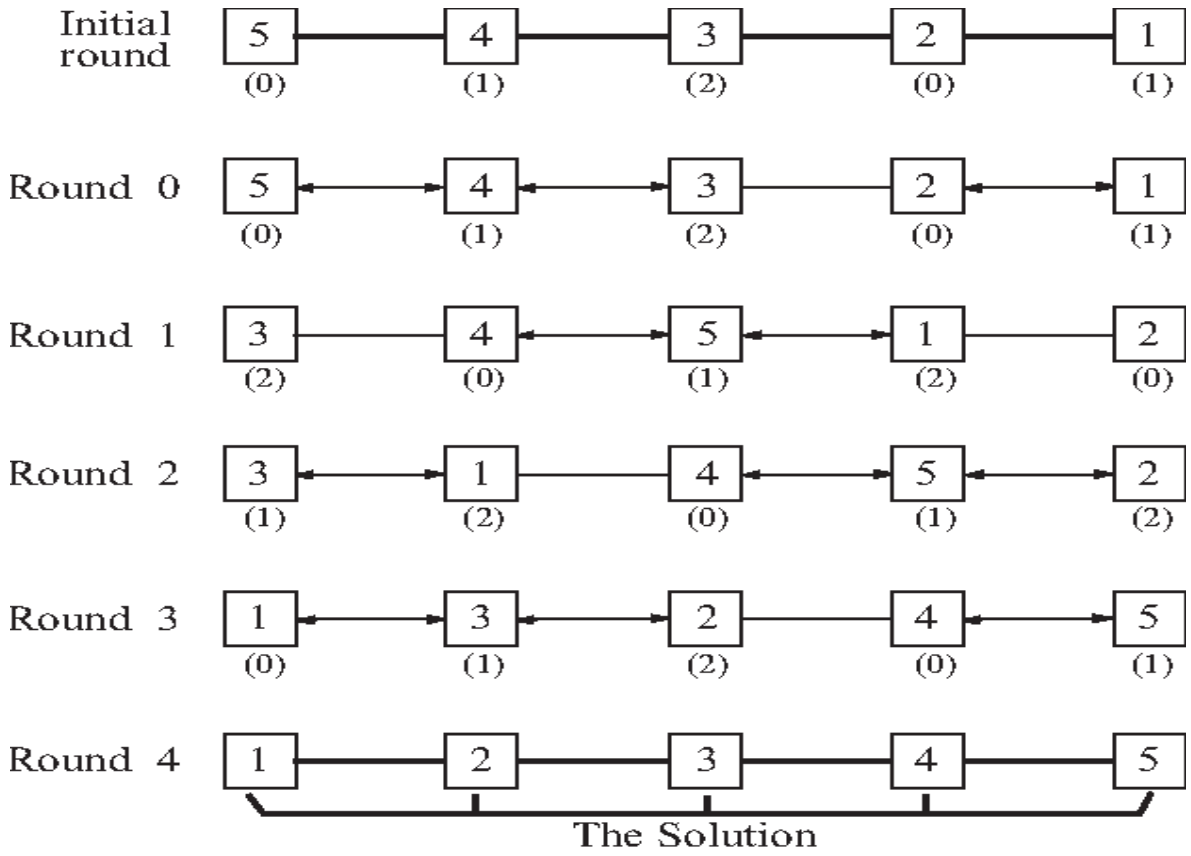
Distributed Sorting – Sasaki's (n-1) round



[Sasaki, 2002]

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

Distributed Sorting – An alternative (n-1) round



DO NOT MAKE copies of elements at intermediate nodes;
No Rule to Select the Final Solution;
No Global position; 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 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 !!

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

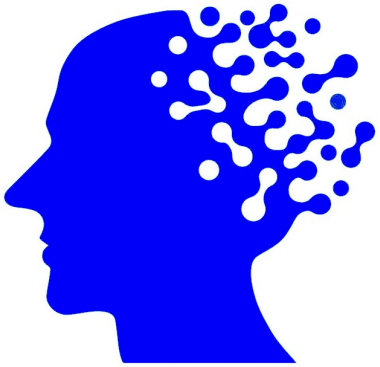
Best Approach

- 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



Spring 2024

Distributed Computing

- Causal Ordering

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

31th January 2024 (http://rajendra.2power3.com)

> Distributed Computing?

- How will you design a Distributed Algorithm?

- Learn to Solve using Distributed Algorithms

> About this Lecture

What do we learn today?

- This covers a model of distributed computations that every algorithm designer needs to know
  - **Causal Ordering in Distributed Systems**
  - **Total Ordering**
  - **CO related concepts in of distributed executions**

Let us explore these topics ➔ ➔ ➔

Today's Focus

Recap: Distributed Systems

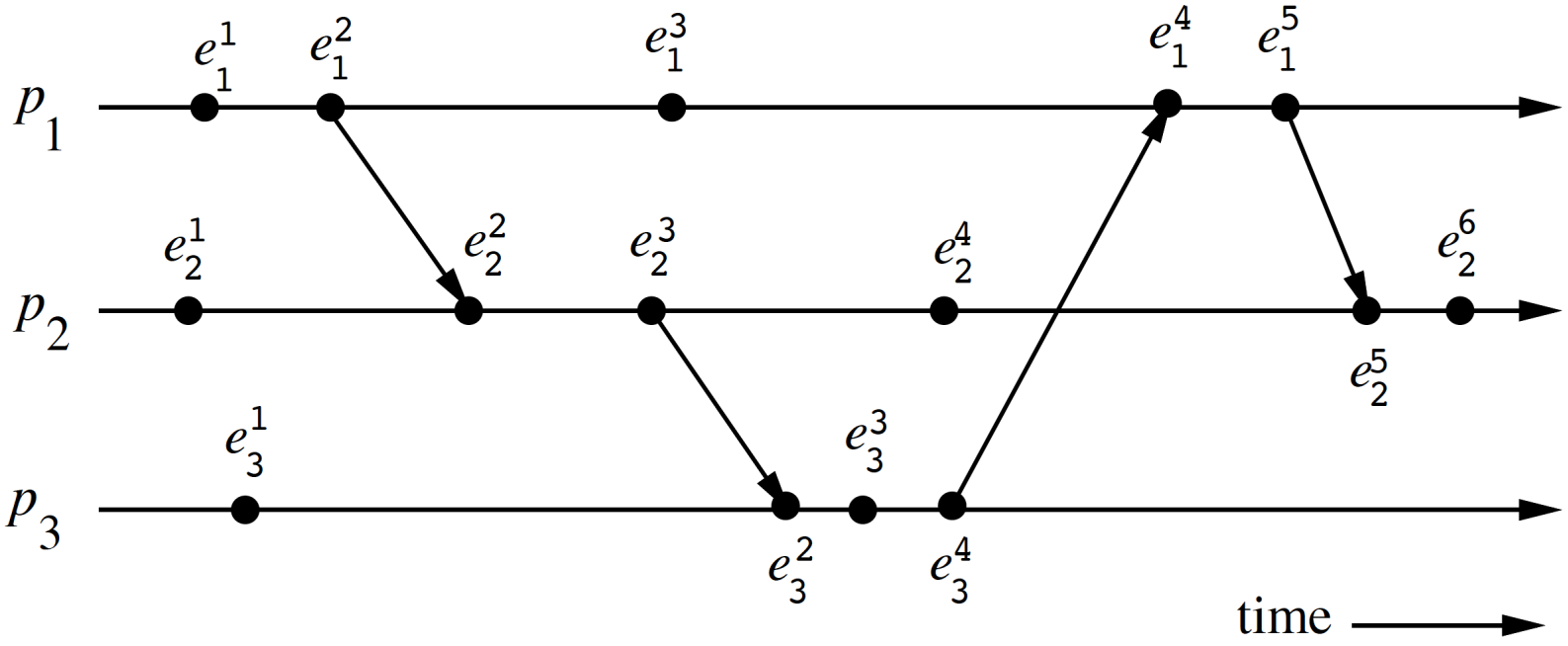
A Distributed System:

- 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

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

What did you learn so far?

- Goals / Challenges with Distributed Systems
- Message Passing systems
  - Basic Primitive operations
  - 3 types of EVENTs: Internal, send and receive
  - States of a process and Channel
- Distributed Sorting
  - Odd – Even Transposition Sort
  - Sasaki's (n-1) rounds algorithms
  - Did you try (n-2) rounds algorithm for distributed sorting on line network?
  - Implementing Discrete Events Simulation

Recap

Causal Ordering

A Model of Distributed Computations

Focused Topics:

- Causal Precedence Relation
- Models of Communication Networks
- Causal Ordering
- Global State and Local States
- Cuts of a Distibuted System
  - PAST and FUTURE events

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 Executions (contd)

- 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

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

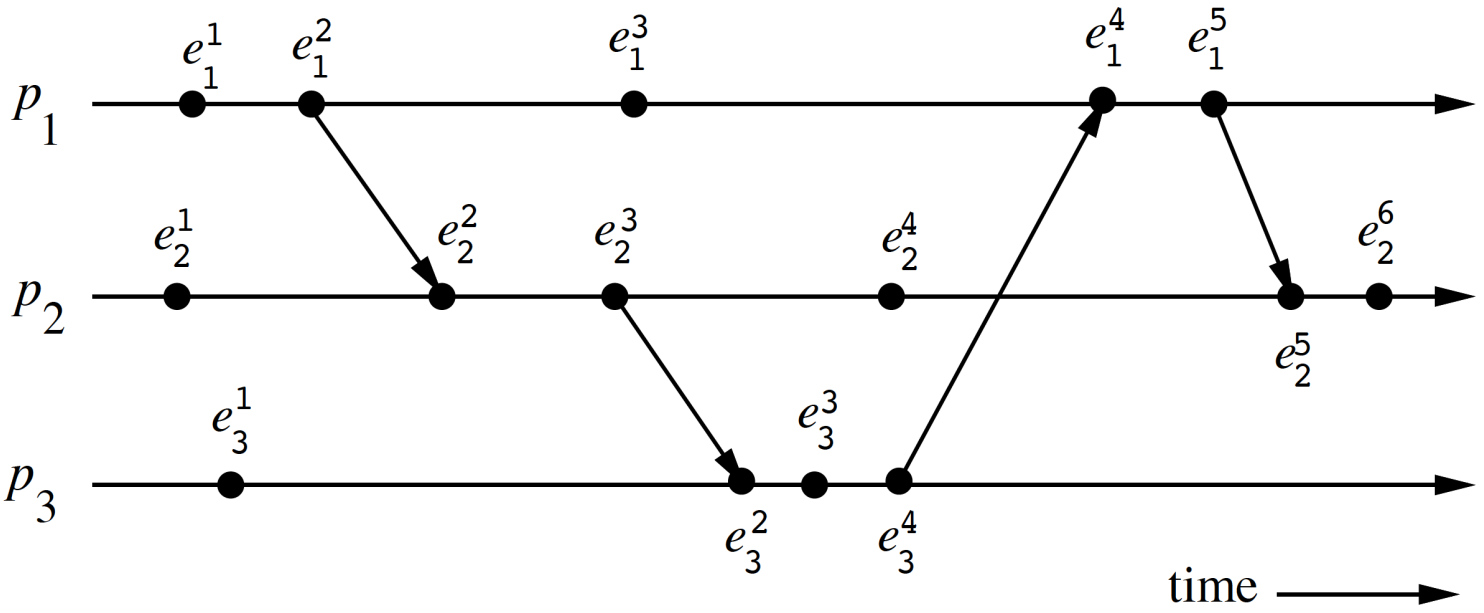
- Relation $\rightarrow_{msg}$ defines causal dependencies between the pairs of corresponding send and receive events

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

Partial Order / Total Ordering

A relation $\leq$ is a total order on a set S ("$\leq$ totally orders S") if the following properties hold:

1. Reflexivity: $a \leq a$ for all a in S

2. Antisymmetry: $a \leq b$ and $b \leq a$ implies $a = b$

3. Transitivity: $a \leq b$ and $b \leq c$ implies $a \leq c$

4. Comparability (trichotomy law):
   For any a, b in S, either $a \leq b$ or $b \leq a$.

First 3 properties → the axioms of a partial order
Addition of trichotomy law defines a total order $H=(H, \rightarrow)$

Causal Precedence Relation

- The execution of a distributed app. results in a set of distributed events
- Let $H = \cup_i h_i$ denote the set of events executed in a distributed computation.
- Define a binary relation $\rightarrow$ on the set H that expresses causal dependencies between events in the distributed execution.

$$\forall e_i^x,\ \forall e_j^y \in H,\quad e_i^x \rightarrow e_j^y \quad \Leftrightarrow \quad \begin{cases} e_i^x \rightarrow_i e_j^y \quad i.e., (i = j) \wedge (x < y) \\ or \\ e_i^x \rightarrow_{msg} e_j^y \\ or \\ \exists e_k^z \in H : e_i^x \rightarrow e_k^z \ \wedge \ e_k^z \rightarrow e_j^y \end{cases}$$

- The causal precedence relation induces an irreflexive partial order on the events of a distributed computation that is *denoted* as $H=(H, \rightarrow)$

Causal Precedence Relation (contd)

- The relation $\rightarrow$ is as defined by Lamport "happens before"

An event e_1 happens before the event e_2 and denoted by $e_1 \rightarrow e_2$ if the following holds true:

- e_1 occurs before e_2 on the same process OR
- e_1 is the send message and e_2 is the corresponding receive message OR
- There exists another event e' such that e_1 happens before e' and e' happens before e_2

Causal Precedence Relation (contd)

- For any two events e_i and e_j, $e_i \not\rightarrow e_j$ denotes the fact that event e_j does not directly or transitively dependent on event e_i

  That is, event e_i does not causally affect event e_j
- In this case, event e_j is not aware of the execution of e_i or any event executed after e_i on the same process.

Note the following two rules:

- For any two events e_i and e_j

$$e_i \not\rightarrow e_j \text{ does not imply } e_j \not\rightarrow e_i$$

- For any two events e_i and e_j

$$e_i \rightarrow e_j \Rightarrow e_j \not\rightarrow e_i$$

Concurrent Events

- For any two events e_i and e_j:

  if $e_i \not\rightarrow e_j$ and $e_j \not\rightarrow e_i$,

  then events e_i and e_j are said to be concurrent (denoted as $e_i \parallel e_j$)

Example:

$$e_3^3 \parallel e_2^4 \text{ and } e_2^4 \parallel e_1^5 \text{ but } e_3^3 \text{ not } \parallel e_1^5$$

- The relation $\parallel$ is not transitive;

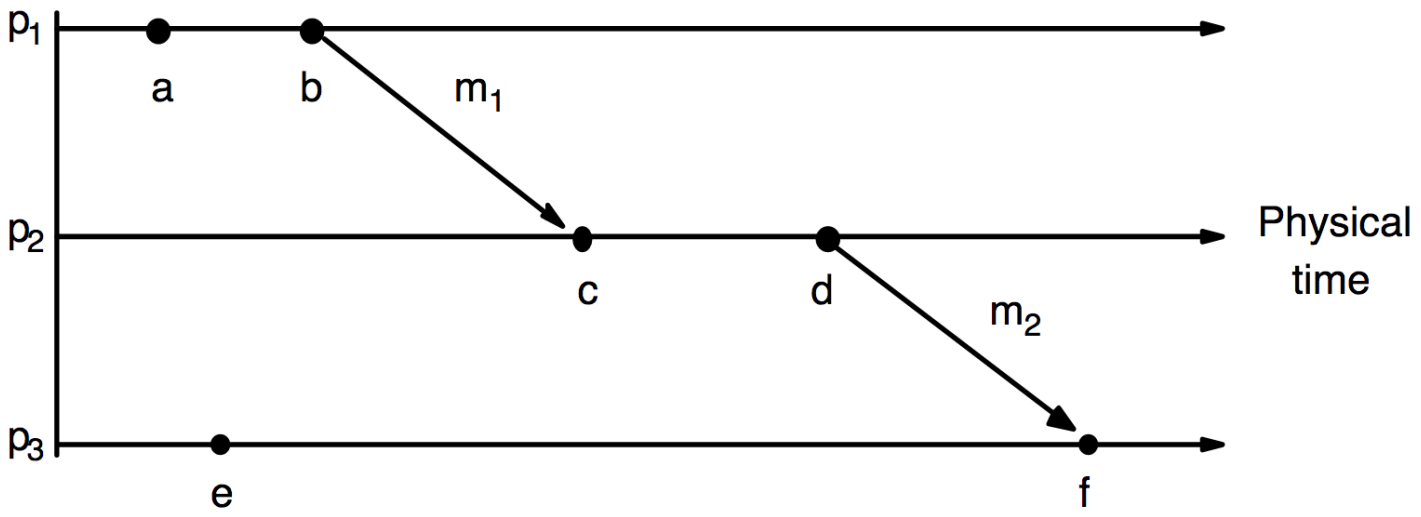
  that is,

  $(e_i \parallel e_j) \wedge (e_j \parallel e_k)$ does not imply $e_i \parallel e_k$

- For any two events e_i and e_j in a distributed execution,

  $e_i \rightarrow e_j$ OR $e_j \rightarrow e_i$, OR $e_i \parallel e_j$

Concurrency – An Example



$a \rightarrow b$ (at $p1$) $c \rightarrow d$ (at $p2$)

$b \rightarrow c$ ($m1$)

also $d \rightarrow f$ ($m2$)

Not all events are related by $\rightarrow$, e.g., a $\not\rightarrow$ e and e $\not\rightarrow$ a
they are said to be concurrent; write as $a \parallel e$

Logical vs. Physical Concurrency

- Two events are logically concurrent if and only if they do not causally affect each other.
- In physical concurrency: events occur at the same instant in physical time.
- Two+ events may be logically concurrent even though they do not occur at same instant in physical time.
- If processor speed and message delays would have been different, the execution of these events could have very well coincided in physical time.
- Whether a set of logically concurrent events coincide in the physical time or not, does not change the outcome of the computation.
- A set of logically concurrent events may not have occurred at the same instant in physical time, we can assume that these events occurred at the same instant in physical time.

Models of Communication networks

- There are several models of the service provided by communication networks:

  FIFO, Non-FIFO, and causal ordering

- In the FIFO model, each channel acts as a first-in first-out message queue and thus, message ordering is preserved by a channel.
- In the non-FIFO model, a channel acts like a set in which the sender process adds messages and the receiver process removes messages from it in a random order.

Causal Ordering

- The "causal ordering" model is based on Lamport's "happens before" relation
- A system that supports the causal ordering model satisfies the following property:

CO: For any two messages m_{ij} and m_{kj},
if $send(m_{ij}) \rightarrow send(m_{kj})$,
then $receive(m_{ij}) \rightarrow receive(m_{kj})$

- This property ensures that causally related messages destined to the same destination are delivered in an order that is consistent with their causality relation.
- Causally ordered delivery of messages implies FIFO message delivery. (Note that CO $\subset$ FIFO $\subset$ Non-FIFO.)
- Causal ordering model considerably simplifies the design of distributed algorithms because it provides a built-in synchronization.

Global State

- A collection of the local states of its components:
  - The processes and the communication channels
- The state of a process is defined by the local contents of processor registers, stacks, local memory, etc
- The state of channel depends the set of messages in transit in the channel

- An internal event changes only state of the process
- A send event changes
  - state of the process that sends the message and
  - the state of the channel on which the message is sent.
- Similarly a receive event changes
  - the state of the process that receives the message and
  - the state of the channel on which the message is received

Global State (contd)

Notations

- LS_i^x denotes the state of p_i after occurrence of event e_i^x and before the event e_i^{x+1}
- LS_i^0 denotes the initial state of process p_i
- LS_i^x is a result of the execution of all the events executed by process pi till e_i^x

- Let $send(m) \leq LS_i^x$ denote the fact:

$$\exists y,\ 1 \leq y \leq x\ s.t\ e_i^y = send(m)$$

- Let $rec(m)\ (not \leq)\ LS_i^x$ denote the fact:

$$\forall y,\ 1 \leq y \leq x\ s.t\ e_i^y\ \ (not\ equal\ to)\ rec(m)$$

Global State (contd)

- The global state of a distributed system is a collection of the local states of the processes and the channels.

A global state GS is defined as,

$$GS = \{\bigcup_i LS_i^{x_i}, \bigcup_{j,k} SC_{jk}^{y_j, z_k}\}$$

- For a global state to be meaningful, the states of all the components of the distributed system must be recorded at the same instant
- Two important situations (Impossible !!):
  - the local clocks at processes were perfectly synchronized
  - there were a global system clock that can be instantaneously read by the processes

A Consistent Global State

Basic idea:

- A state should not violate causality – an effect should not be present without its cause
- A message cannot be received if it was not sent.
- Such states are called consistent global states and are meaningful global states.
- Inconsistent global states are not meaningful in the sense that a distributed system can never be in an inconsistent state

What is a Consistent Global State?

Definition:

➔ A global state is a consistent global state iff

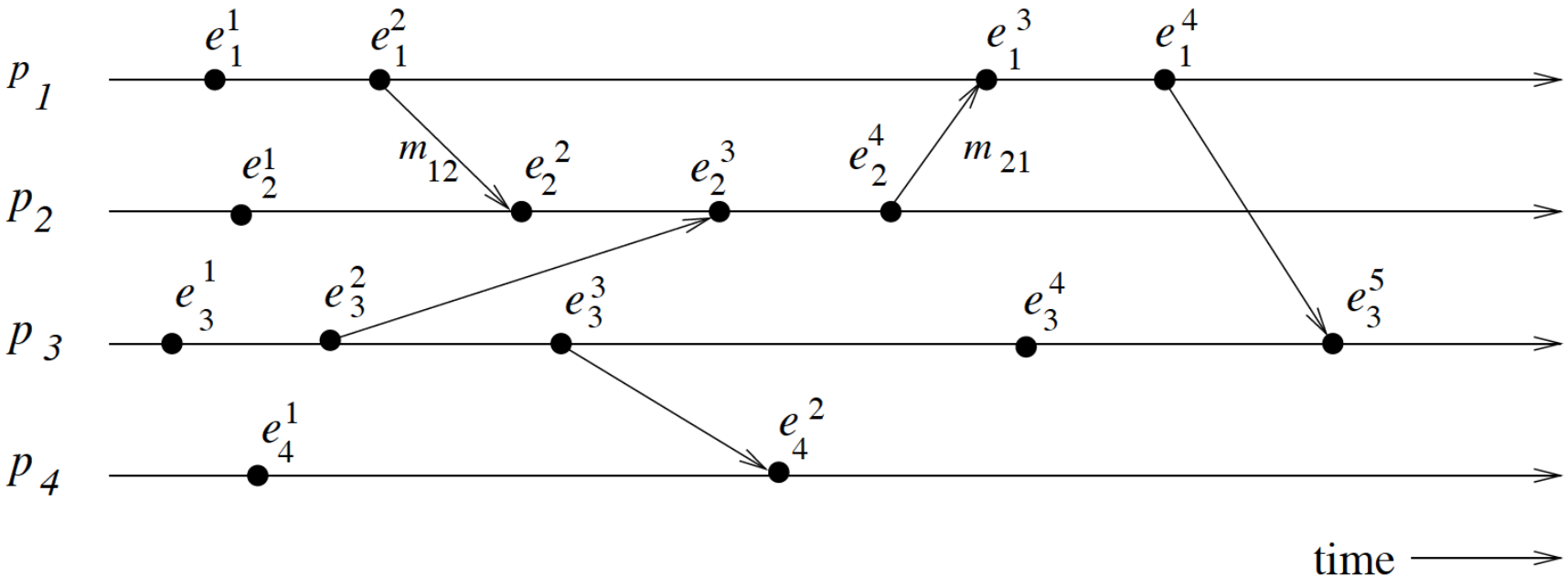
$$\forall m_{ij} : \; send(m_{ij}) \not\leq LS_i^{x_i} \;\Leftrightarrow\; m_{ij} \notin SC_{ij}^{x_i, y_j} \bigwedge rec(m_{ij}) \not\leq LS_j^{y_j}$$

Where the global state is given by

$$GS = \{\bigcup_i LS_i^{x_i}, \bigcup_{j,k} SC_{jk}^{y_j, z_k}\}$$

➔ This implies that the channel state and process state must not include any message that process p_i sent after executing event

Consistent Global State – An Example



Consistent Global State – Details

- A global state GS1 = {LS_1^1 , LS_2^3 , LS_3^3 , LS_4^2 } is inconsistent because
  - the state of p_2 has recorded the receipt of message m_{12}
  - The state of p_1 has not recorded its send

- A global state GS2 consisting of local states {LS_1^2 , LS_2^4 , LS_3^4 , LS_4^2 } is consistent;
- all the channels are empty except C_{21} that contains message m_{21}.

Run / Consistent Run

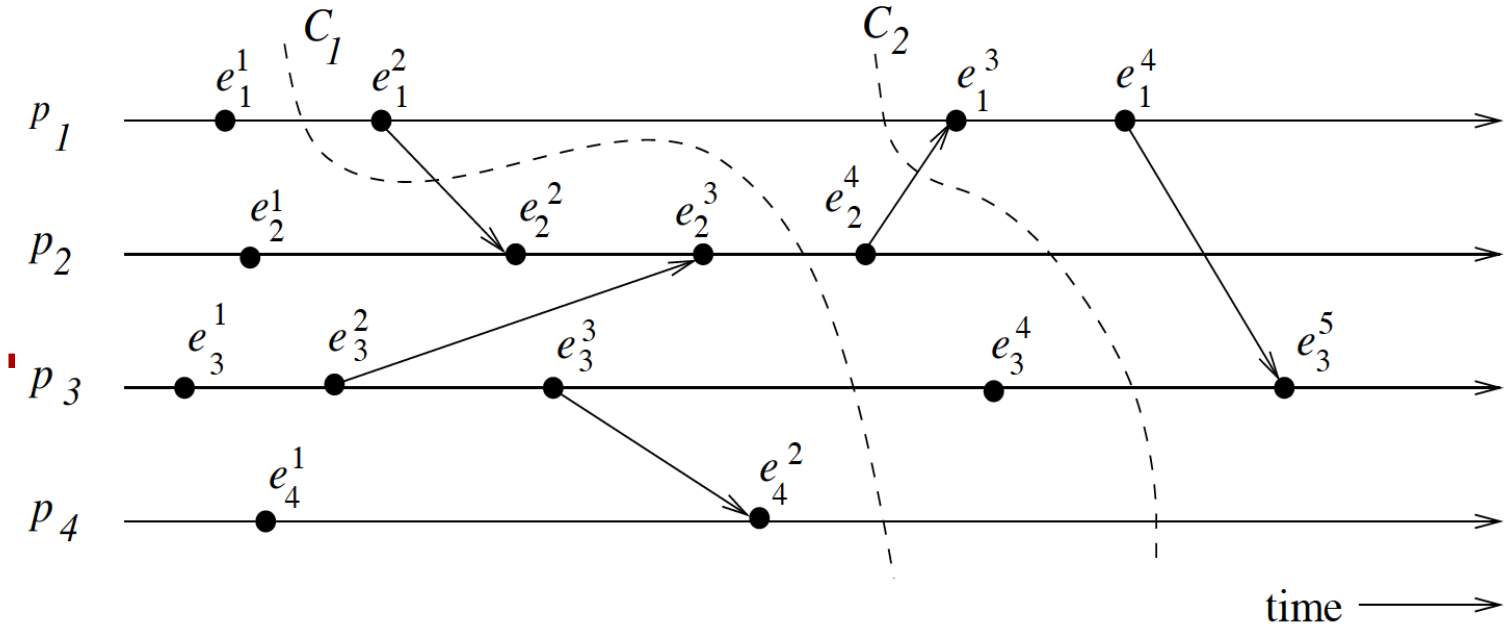
Run:

- A run is an ordering of the events that satisfies the happened-before relation in one process

Consistent Run:

- A consistent run is an ordering of the events that satisfies all the happened-before relations

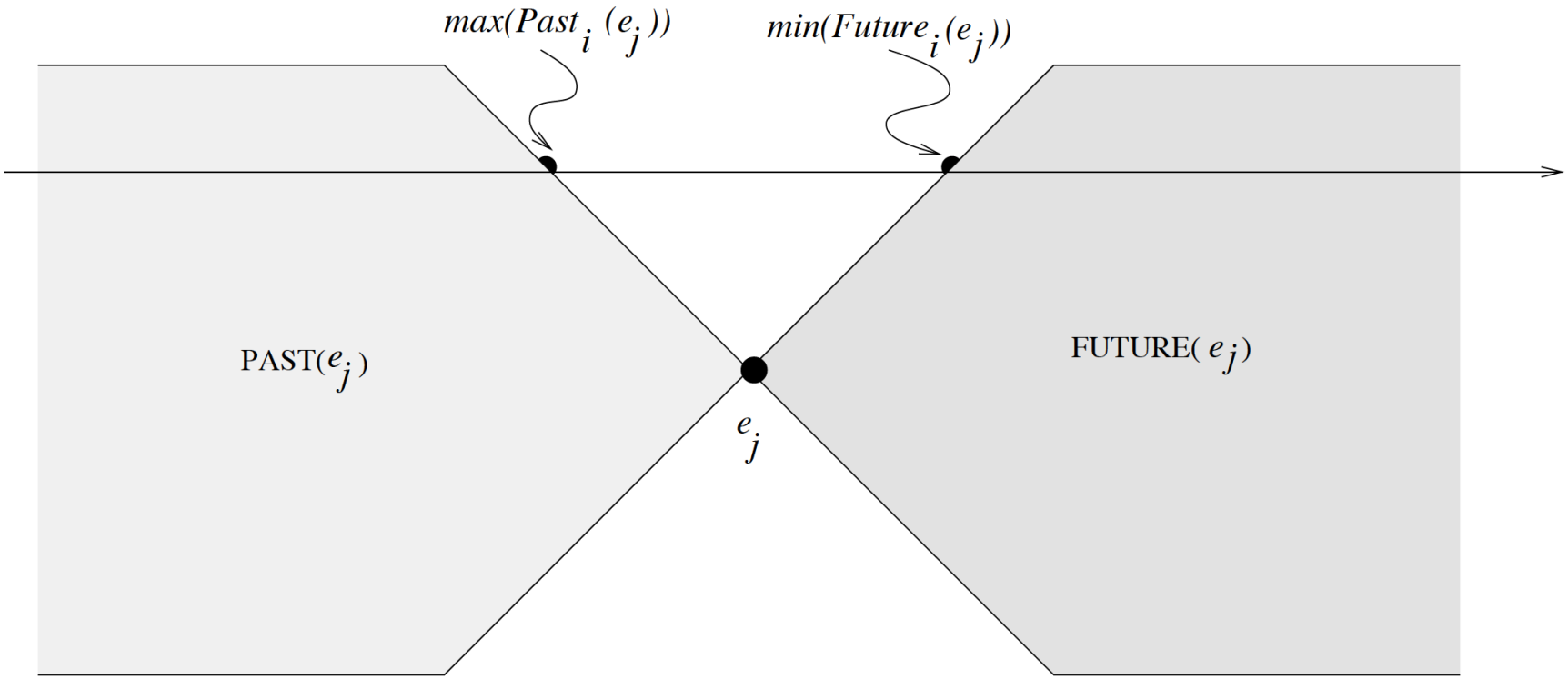
Cuts of a Distributed Computation



Cuts of a Distributed Computation

- In a consistent cut, every message received in the PAST of the cut was sent in the PAST of that cut
  - In previous figure, cut C2 is a consistent cut
- All messages that cross the cut from the PAST to the FUTURE are in transit in the corresponding consistent global state.
- A cut is inconsistent if a message crosses the cut from the FUTURE to the PAST
  - In previous figure cut C1 is an inconsistent cut

Past and Future Cones of an event



Physical vs Logical clocks?

- Logical Clocks
  - Design and Implementation
- Three Different Ways
  - Scalar Time
  - Vector Time
  - Matrix Time
- Virtual Clocks
  - Time Wrap Mechanism
- Clock Synchronization
  - NTP Synchronization Protocol

Summary

- A Model of Distributed Computations
  - Distributed Sorting
    - Design and Implementation Issues
  - Causal Precedence Relations
  - Global State and Cuts of a DS
  - PAST and FUTURE events
  - What about the ordering of events?
    - How do we efficiently handle the ordering of events (discrete events)?
    - 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