

Monsoon 2022



Distributed Computing - Termination Detection Algorithm



Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor



21st September 2022 (<http://rajendra.2power3.com>)

> Distributed Computing?

- How will you design a Distributed Algorithm?



- Learn to Solve using Distributed Algorithms

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

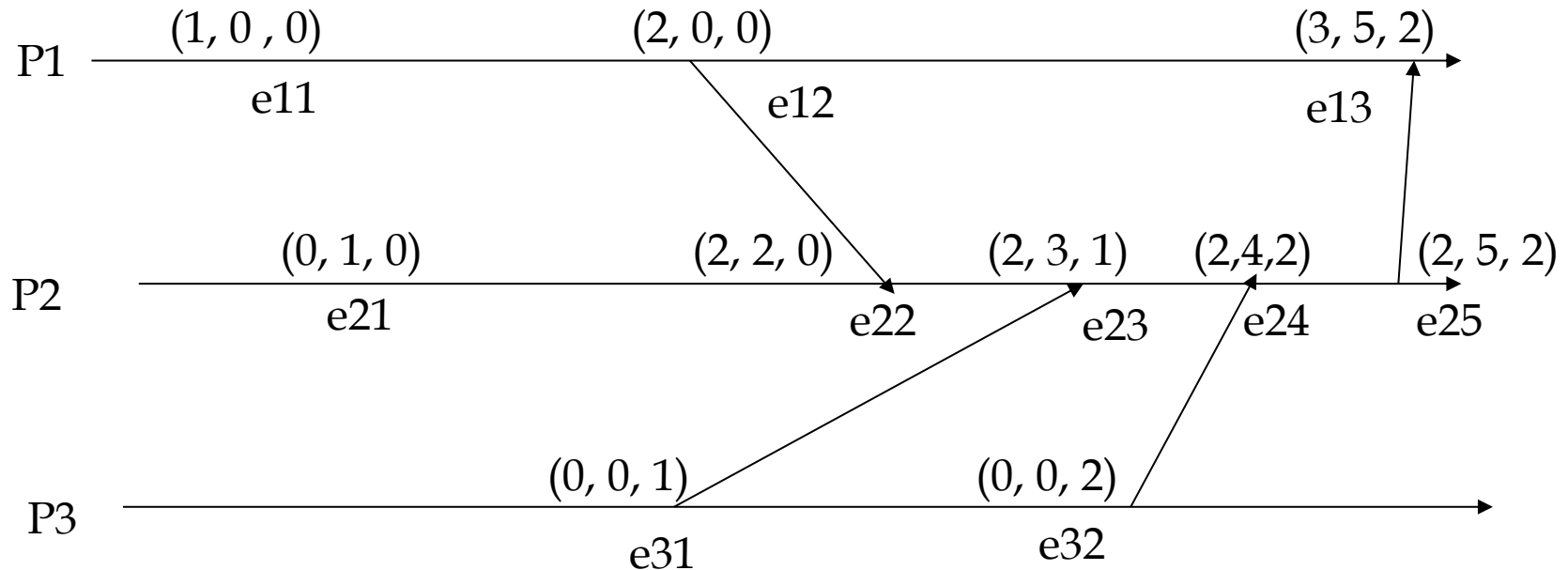
Recap: Characteristics

- **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 did you learn so far?

- Goals / Challenges in Message Passing systems
- Distributed Sorting
- Space-Time diagram
- Partial Ordering / Total Ordering
- Causal Precedence Relation
 - Happens Before
- Concurrent Events
 - How to define Concurrent Events
 - Logical vs Physical Concurrency
- Causal Ordering
- Logical Clocks vs Physical Clocks

Recap: An illustrative example



Less than or equal:

→ $ts(a) \leq ts(b)$ if $ts(a)[i] \leq ts(b)[i]$ for all i
 $(3, 3, 5) \leq (3, 4, 5)$

→ $ts(e11) = (1, 0, 0)$ and $ts(e22) = (2, 2, 0)$
 This implies $e11 \square e22$

> About this Lecture

What do we learn today?

- This covers a model of distributed computations that every algorithm designer needs to know
 - **Global States**
 - **Global Snapshot algorithm**
 - **Termination Detection algorithm**
 - **Using Distributed Snapshots**

Let us **explore** these topics ➔ ➔ ➔

A Consistent **Global** State

Definition:

→ A global state is a consistent global state iff

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

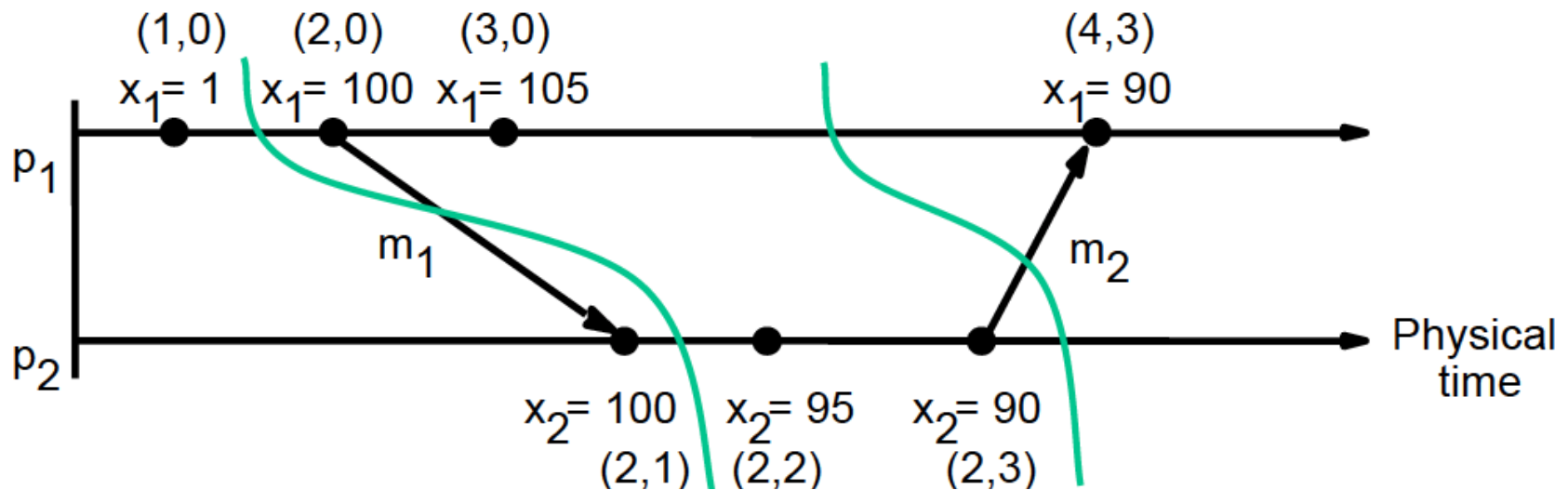
Where the global state is given by

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

→ 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

- Let $V(s_i)$ be the vector timestamp of state s_i received from P_i .
- S is a consistent global state if and only if:
 $V(s_i)[i] \geq V(s_j)[i]$ for all i, j in $[1, N]$



Reachable

When is a state said to be reachable?

- A state S' is reachable from a state S if there exists a **consistent run** (Ordering of events satisfies all happened-before relations) from S to S' .
- May exist more than one consistent run, since the ordering from happened-before relation is a partial order

Data Sensitive Applications

Banking Example

→ A Few Banking Operations

deposit(amount)

deposit amount in the account

withdraw(amount)

withdraw amount from the account

getBalance() → amount

return the balance of the account

setBalance(amount)

set the balance of the account to amount

Termination Detection

- A Fundamental Problem: Determine the termination status of a distributed computation
- A non-trivial task: NO process has complete knowledge of the global state, and global time does not exist
- A distributed computation is globally terminated if every process is locally terminated and there is no message in transit between any processes
- “Locally terminated” state is a state in which a process has finished its computation and will not restart any action unless it receives a message
- In the termination detection problem, a particular process (or all of the processes) must infer when the underlying computation has terminated

Important aspects

- Messages used in the underlying computation are called **basic messages**, and messages used for the purpose of termination detection are called **control messages**
- A termination detection (TD) algorithm must ensure the following:
 - Execution of a TD algorithm cannot indefinitely delay the underlying computation
 - The termination detection algorithm must not require addition of new communication channels between processes

System Model

- At any given time, a process can be in only one of the two states: **active** where it is doing local computation and **idle** otherwise and will be reactivated only on the receipt of a message from another process.
- An active process can become idle at any time but an idle process can become active only on the receipt of a message from another process
- Only active processes can send messages
- A message can be received by a process when it is in one of two states: active or idle. On receipt of a message, an idle process becomes active
- The sending of a message and the receipt of a message occur as atomic actions

Termination Detection - Definition

- Let $P_i(t)$ denote the state (active or idle) of process P_i at instant t
- Let $c_{i,j}(t)$ denote the number of messages in transit in the channel at instant t from process P_i to process P_j
- A distributed computation is said to be **terminated** at time instant t_k iff:

for all i ,

$$(P_i(t_k) = \text{idle}) \wedge (\text{for all } i, j \text{ such that } c_{i,j}(t_k) = 0)$$

- Thus, a distributed computation has terminated iff all processes have become idle and there is no message in transit in any channel

TD using Distributed Snapshots

- **Assumption:** There is a logical bidirectional communication channel between every pair of processes
- Communication channels are reliable
- Message delay is arbitrary but finite

TD using Distributed Snapshots

Main idea:

- When a process goes from active to idle state, it issues a request to all other processes to take a local snapshot, and also requests itself to take a local snapshot
- When a process receives the request, if it agrees that the requester became idle before itself, it grants the request by taking a local snapshot for the request
- A request is successful if all processes have taken a local snapshot
- The requester or any external agent may collect all the local snapshots of a request
- If a request is successful, a global snapshot of the request can thus be obtained and the recorded state will indicate termination of the computation

A Formal Description

- Each P_i has a logical clock x initialized to zero at t_0
- A process **increments its x by one each time it becomes idle**
- A basic message sent by a process at its logical time x is of the form $B(x)$
- A control message that requests processes to take local snapshot issued by P_i at its logical time x is of the form $R(x, i)$
- Each process synchronizes its logical clock x loosely with the logical clocks x' 's on other processes in such a way that it is the maximum of clock values ever received or sent in messages
- A process also maintains a variable k such that when the process is idle, (x, k) is the maximum of the values (x, k) on all messages $R(x, k)$ ever received or sent by the process
- **Logical time is compared** as follows: $(x, k) > (x', k')$ iff $(x > x')$ or $((x = x') \text{ and } (k > k'))$ i.e., a tie between x and x' is broken by the process identification numbers k and k'

Algorithm

→ Four Rules:

- (R1): When process i is active, it may send a basic message to process j at any time by doing
send a $B(x)$ to j .
- (R2): Upon receiving a $B(x')$, process i does
let $x := x' + 1$;
if (i is idle) \rightarrow go active.
- (R3): When process i goes idle, it does
let $x := x + 1$;
let $k := i$;
send message $R(x, k)$ to all other processes;
take a local snapshot for the request by $R(x, k)$.
- (R4): Upon receiving message $R(x', k')$, process i does
[[$((x', k') > (x, k)) \wedge (i \text{ is idle}) \rightarrow$ let $(x, k) := (x', k')$;
take a local snapshot for the request by $R(x', k')$;
 \square
 $((x', k') \leq (x, k)) \wedge (i \text{ is idle}) \rightarrow$ do nothing;
 \square
 $(i \text{ is active}) \rightarrow$ let $x := \max(x', x)$].

→ The last process to terminate will have the largest clock value.

Summary

→ Global Snapshots

- Global State of a DS

- Chandy - Lamport's GS Recording Algorithm

- Initiating / Propagating / Terminating the Snapshot Algorithm

→ Termination Detection

- Definition

- A Formal Description

- TD using Global Snapshots

→ 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