RAMAIAH
Institute of Technology

M.S. Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of Computer Science and Engineering

Course Name: Distributed Systems
Course Code: CSE751/CSE20
Credits: 3:0:0 / 3:0:0:1

Term: Oct 2021- Feb 2022

Faculty:
Sini Anna Alex

Variations of the Chandy-Lamport algorithm

- Several variants of the Chandy-Lamport snapshot algorithm followed.
- These variants refined and optimized the basic algorithm.
- For example, Spezialetti and Kearns algorithm optimizes concurrent initiation of snapshot collection and efficiently distributes the recorded snapshot.
- Venkatesan's algorithm optimizes the basic snapshot algorithm to efficiently record repeated snapshots of a distributed system that are required in recovery algorithms with synchronous checkpointing.

Spezialetti-Kearns algorithm

- There are two phases in obtaining a global snapshot: locally recording the snapshot at every process and distributing the resultant global snapshot to all the initiators.
- Spezialetti and Kearns optimized the Chandy-Lamport algorithm by exploiting the work of combining concurrently initiated snapshots (in the first phase) to efficiently distribute the resultant global snapshot to only the concurrent initiators (in the second phase).
- A process needs to take only one snapshot, irrespective of the number of concurrent initiators and all processes are not sent the global snapshot.

Spezialetti-Kearns algorithm

Efficient snapshot recording

- In the Spezialetti-Kearns algorithm, a markers carries the identifier of the initiator of the algorithm. Each process has a variable *master* to keep track of the initiator of the algorithm.
- A key notion used by the optimizations is that of a *region* in the system. A region encompasses all the processes whose *master* field contains the identifier of the same initiator.
- When the initiator's identifier in a marker received along a channel is different from the value in the *master* variable, the sender of the marker lies in a different region.
- The identifier of the concurrent initiator is recorded in a local variable *id-border-set*.

Spezialetti-Kearns algorithm

- The state of the channel is recorded just as in the Chandy-Lamport algorithm (including those that cross a border between regions).
- Snapshot recording at a process is complete after it has received a marker along each of its channels.
- After every process has recorded its snapshot, the system is partitioned into as many regions as the number of concurrent initiations of the algorithm.
- Variable *id-border-set* at a process contains the identifiers of the neighboring regions.

Efficient dissemination of the recorded snapshot

- In the snapshot recording phase, a forest of spanning trees is implicitly created in the system. The initiator of the algorithm is the root of a spanning tree and all processes in its region belong to its spanning tree.

Efficient dissemination of the recorded snapshot

- If p_i receives its first marker from p_j then process p_j is the parent of process p_i in the spanning tree.
- When an intermediate process in a spanning tree has received the recorded states from all its child processes and has recorded the states of all incoming channels, it forwards its locally recorded state and the locally recorded states of all its descendent processes to its parent.
- When the initiator receives the locally recorded states of all its descendents from its children processes, it assembles the snapshot for all the processes in its region and the channels incident on these processes.
- The initiator exchanges the snapshot of its region with the initiators in adjacent regions in rounds.

Process

During the progress of a distributed computation, processes may periodically turn **active** or **passive**.

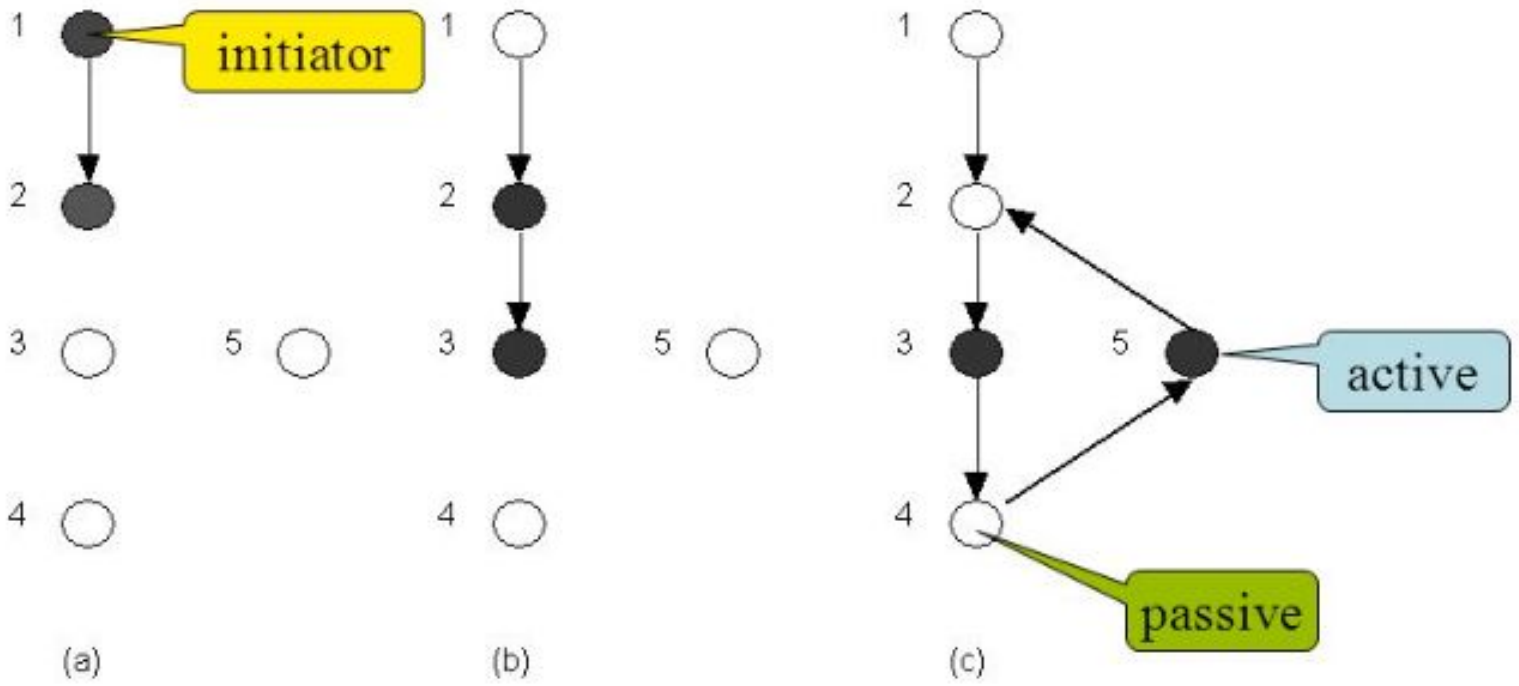
A distributed computation termination when:

(a) every process is **passive**,

(b) all channels are **empty**, and

(c) the global state satisfies the **desired postcondition**

Spezialetti-Kearns algorithm



Basic Scheme for termination

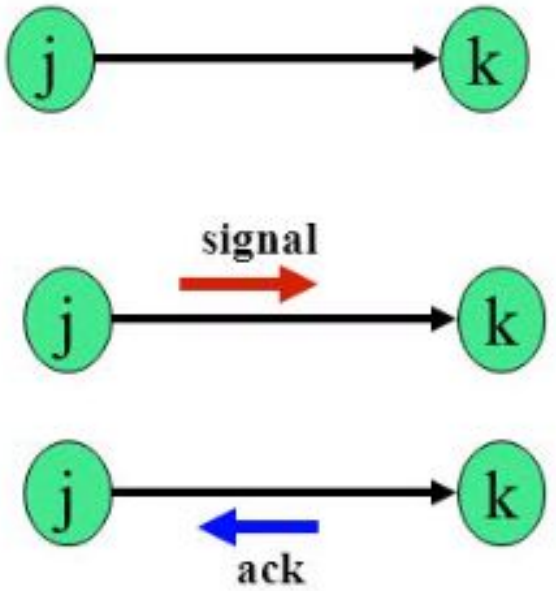
The basic scheme

An initiator initiates termination detection by sending signals (messages) down the edges via which it engages other nodes.

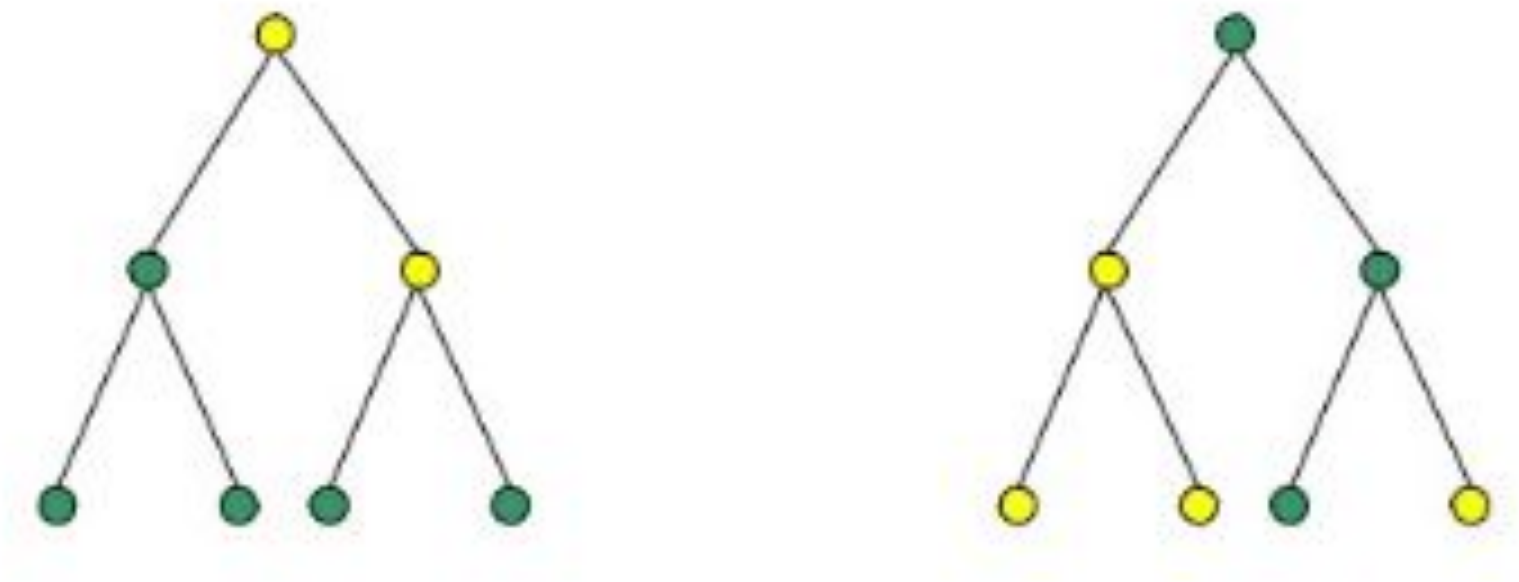
At a "suitable time," the recipient sends an ack back.

When the initiator receives ack from every node that it engaged, it detects termination.

Node j **engages** node k.



Spezialetti-Kearns algorithm



Efficient dissemination of the recorded snapshot

- This algorithm assumes bidirectional channels in the system.
- The message complexity of snapshot recording is O(e) irrespective of the number of concurrent initiations of the algorithm.
- The message complexity of assembling and disseminating the snapshot is $O(rm^2)$ where r is the, number of concurrent initiations.

Snapshot algorithms for non-FIFO channels

- In a non-FIFO system, a marker cannot be used to delineate messages into those to be recorded in the global state from those not to be recorded in the global state.
- In a non-FIFO system, either some degree of inhibition or piggybacking of control information on computation messages to capture out-of-sequence messages.

Lai-Yang algorithm

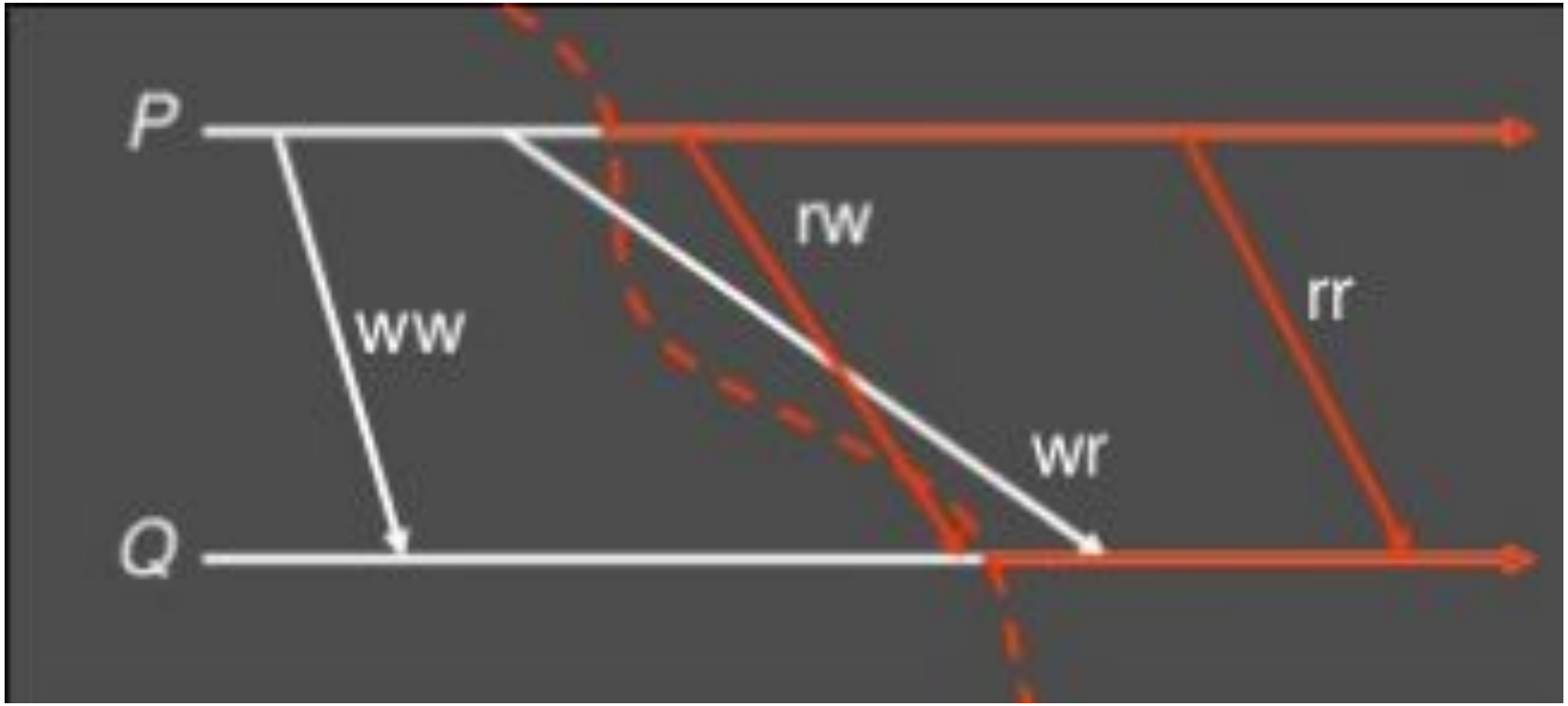
The Lai-Yang algorithm fulfills this role of a marker in a non-FIFO system by using a coloring scheme on computation messages that works as follows:

1. Every process is initially white and turns red while taking a snapshot. The equivalent of the "Marker Sending Rule" is executed when a process turns red.
2. Every message sent by a white (red) process is colored white (red).
3. Thus, a white (red) message is a message that was sent before (after) the sender of that message recorded its local snapshot.
4. Every white process takes its snapshot at its convenience, but no later than the instant it receives a red message.

Lai-Yang algorithm

4. Every white process records a history of all white messages sent or received by it along each channel.
5. When a process turns red, it sends these histories along with its snapshot to the initiator process that collects the global snapshot.
6. The initiator process evaluates $transit(LS_i, LS_j)$ to compute the state of a channel C_{ij} as given below:
   SC_{ij} = white messages sent by p_i on C_{ij} − white messages received by p_j on C_{ij}
   $= \{send(m_{ij}) | send(m_{ij}) \in LS_i \} - \{rec(m_{ij}) | rec(m_{ij}) \in LS_j \}$.

Lai-Yang algorithm



Thank you