Sanyam Kaul

Roll No- CS23MTECH14011

Distributed Computing, Theory Assignment-1

17 Feb 2024

Spezialetti-Kerns Algorithm

VARIABLES & DATA STRUCTURES: -

- `marker_received: boolean` // Flag to track whether a marker has been received
- `master: identifier of initiator` // Identifier of the initiator process
- `id_border_set: set of identifiers` // Set of identifiers of neighboring regions
- `local_snapshot: local state of the process` // Local state of the process
- `parent: parent process in the spanning tree` // Parent process in the spanning tree
- `children: list of child processes in the spanning tree` // List of child processes
- `channel_states: states of incoming channels` // States of incoming channels

INITIALIZATION: -

```
marker_received = false

master = NULL

id_border_set = {}
```

```
local_snapshot = {}
parent = NULL
children = []
channel_states = {}
```

SNAPSHOT RECORDING PHASE: -

/ Update the master variable with the process ID which has sent the marker (if getting marker for the first time) or add the Process id to the id_border_set in case the marker's master value does not match the master variable value */*

```
Upon receiving a marker from process pj:
    if marker_received = false:
        marker_received = true
        master = pj.id
    if master ≠ pj_marker.master:
        id_border_set.add(pj_marker.master)
```
// marker came from different region so added to set

```
        stop_propagating_marker()
```
//if marker came from other region stop it so unnecessary snapshots are not generated

/ If the current process is the initiator of the Snapshot recording procedure then update the master variable with the the current process's Process ID */*

```
If initiator:
    master = self.process_id
```

```
record_channel_states_from(ci)  /*calculate the channel state by using the Chandy-Lamport Channel state calculation logic */

/* If markers received from all the channels then call the record_snapshot function  followed by the send snapshot to parent function*/
if all channels have marker_received:
    record_snapshot()
    send_snapshot_to_parent()  /* When markers from all the channels arrive and the termination condition is achieved send the calculated snapshot snapshot to the parent.*/


Upon receiving snapshot from child process ci:
    While (not snapshots from all children arrive):
        wait()
    send_snapshot_to_parent()

/* functions to record snapshots, record channel states and send final snapshot to parent */

Function record_snapshot():
    local_snapshot = record_local_state()

Function record_channel_states_from(process):
    If marker_received == false:
        Channel_state = NULL
    else:
```

```
        channel_states.merge(process.channel_states)  /* channel state will consist
```

of all the events occurred after the time when last snapshot was taken */

```
   Function send_snapshot_to_parent():
        if parent ≠ NULL:
             send(local_snapshot, id_border_set) to parent
```

SNAPSHOT DISSEMINATION PHASE: -

/ When the initiator process in a region will get the snapshot and id_border_set, final snapshot of the region will be constructed and shared with only the initiator processes of other regions by referring the id_border_set*/*

```
Upon receiving snapshot and id_border_set from parent:
       channel_states.merge(parent.channel_states)
       id_border_set.merge(parent.id_border_set)
       if not initiator:
           send_snapshot_to_children()
       else:
           exchange_snapshot_with_adjacent_regions()

   Function send_snapshot_to_children():
       for each child in children:
           send(local_snapshot, id_border_set) to child
```

```
    /* Using id_border_set to send the final snapshot to the initiators present in the neighboring regions*/
    Function exchange_snapshot_with_adjacent_regions():
        for each adjacent_region in id_border_set:
            exchange_snapshot_with(adjacent_region)

    Function exchange_snapshot_with(region):
        send(local_snapshot, id_border_set) to region
        receive(snapshot, border_set) from region
        merge_snapshots(snapshot)
        id_border_set.merge(border_set)

Function merge_snapshots(snapshot):
    channel_states.merge(snapshot.channel_states)
```