CSE 535 Assignment 3

Swethasri Kavuri
SBU ID: 111465291

1: Summarize and compare the specifications of the algorithms and of the correctness properties. Include at least these 4 attributes: specification sizes; ease of understanding; how closely are different aspects of the algorithms followed; and what properties are specified and in what way.

Specification 1 (LamportMutex.tla):

- **Summary:** Process's clock value is initialized with 1 and it can make one request at a time. Process clock is updated upon receiving an event and acknowledgement is sent.

- **Constants:**

N	Total Process
maxClock	Max Clock of Process.

- **Initial state:**

clock[p]	Process p's clock set to 1 in the beginning
req[p][q]	Stores clock of Request from q to p, and is set to 0 at the beginning
ack[p]	Set of Process that send ack message for p's request.
network[p][q]	When process p sends a message to process q, they are stored in First In First Out Order
crit	Processes in Critical Section

- **Actions:**

Request(p) When process hasn't sent any request	**1.** req[p][p] is set to clock value of P.. **2.** Send a request message to all other processes and update ack with p.
ReceiveRequest(p, q): When p receives message from q	**1.** Set p's req to the clock of q's message received. **2.** Set p's clock to MAX(current_clock_of_p,clock_Of_q's_message) **3**. Delete received message from the network
Receive Ack(p, q): If q sends an 'ack' to p	**1.**Update p's ack set with process q **2**.Delete acknowledge message from the network.
Enter(p): When p gets 'ack' from all other processes	**1.**If p's request has highest priority than any other process,it enters critical sections.
Exit(p): When p enters	**1.**Delete p from critical section and send a release message to other processes. **2**.Update p's req and ack to None.
ReceiveRelease(p,q) When q sends release to p	**1.** Remove q's message from p.

State Constraint: ClockConstraint == \A p \in Proc : clock[p] <= maxClock

Invariants:

1. Mutex: == \A p, q \in crit : p = q (Only one process in Critical region)
2. BoundedNetwork == \A p,q \in Proc : Len(network[p][q]) <= 3 (Maximum number of messages = 3 in given network)

Properties:

1.Liveness == \A p \in Proc : <>(p \in crit)

Specification size: 123 lines

Ease of understanding:

1. Easier than other specifications
2. Able to relate and understand , if we go through lecture notes and slides.

Specification 2 (Spec2.tla):

- **Summary:** Process's clock value is initialized with 0...NumNats and it can make one request at a time. Process clock is updated upon receiving an event and acknowledgement is sent only if p has not sent a message to q with timestamp greater than the received message from q.

- **Constants:**

NumProcs	Total Process
NumNats	Max Clock of Process.
_ \ll _	p < q

- **Initial state:**

state[p]	Process p's state set to idle in the beginning
msgQ[p][q]	Stores messages from p to q, and is set to empty at the beginning
reqSet[p]	Set of req messages to/from process p is set to empty at the beginning
clock[p]	Process p clock's range is from 0 to NumNats In the beginning
lastTSent[p][q]	P to Q message's latest clock value is set to 0 in the beginning.
lastTReceived[p][q]	Q to P message's latest clock value is set to 0 in the beginning.

- **Actions:**

Request(p) P->'idle'	1. Change P from 'idle' ->'waiting' 2. Change P's clock to any greater value than its current ,in the range of 0 to NumNats 3. Send 'aquire' to other process and it to its own request set. 4. Change lastTSent to timestamp of the latest message sent to other processes.
Acquire(p) P->'waiting' and sent 'acquire' to other processes.	1. If P's req has lower timestamp than other process request,and if P has received ack from other process,then set P->'owner'. 2. Delete P's req from request set
Release(p) P->'owner' and wants to come out of critical section	1. Change P->'idle' from P-'owner' and send release to other process 2. Change lastTSent to timestamp of the latest message sent to other processes.
RcvMsg(p, q) P receives Q's message	1. Set p's clock to MAX(current_clock_of_p,clock_Of_q's_ message) 2. Change lastTRcvd to timestamp of the latest message received to other processes. 3. If an acquire message is received,send an acknowledgement message to q only if the q's message's clock is greater than the clock of p's latest message to q, and add it to reqSet 4. If a release message is received, remove it from reqSet
Tick(p)	1.Set P's clock to next greater value in the range

State Constraint: ClockConstraint == \A p \in Proc : clock[p] < NumNats
Invariants:Mutex == \A s,t \in Proc : state[s] = "crit" /\ state[t] = "crit" => s = t

Properties: Liveness == \A p \in Proc : /\ WF_vars(Acquire(p))
/\ \A q \in Proc \ {p} : WF_vars(RcvMsg(p,q))

Specification size: 100 lines of code

Ease of understanding:

- Difficult to understand compared to Spec 1.
- Able to relate with Lamport's paper.

Specification 3 (Mutex.tla):

- **Summary :**Process's clock value is initialized with 1 and it can make one request at a time. Process clock is updated upon receiving an event and acknowledgement is sent.Site and communicator are present in this spec.Site requests access to critical section and communicator handles received messages
- **Constants:**

N	Total Process
maxClock	Max Clock of Process.

- **Initial state:**

network[p][q]	Messages sent from Site p to Site q
reqQ[p]	Set of req messages to Site p, is set to empty at the beginning. Messages are sorted as per their logical clock
clock[p]	Process p clock's range is set to 1
acks[p]	All the ack messages to Site p.
pc[p]	P's state is 'start' and 'comm' for sites and communicators respectively

Actions: (P-> represents P's state)

start	Change P->'start' to P->'ncrit'
ncrit	Change P->'ncrit' to P->'try'
try	If P->'try', then send request message across all the sites in network, and then Change P->'try' to P->'enter'
enter	If P->'enter', then it checks the priority based on time stamp, and also checks if it has received ack from all other sites. If So, then Change P->'enter' to P-> 'crit'
crit	Change P->'crit' to P->'exit'
exit	If P->'exit' then send release message to other sites and then Change P->'exit' to P->'start'

- **State Constraint:**ClockConstraint == \A s \in Sites : clock[s] <= maxClock
- **Invariants**: Mutex == \A s,t \in Sites : pc[s] = "crit" /\ pc[t] = "crit" => s = t

```
TypeInvariant ==
    /\ network \in [Sites -> [Sites -> Seq(Message)]]
    /\ clock \in [Sites -> Nat]
    /\ reqQ \in [Sites -> Seq([site : Sites, clk : Nat])]
    /\ \A s \in Sites : \A i \in 1 .. Len(reqQ[s])-1 : beats(reqQ[s][i], reqQ[s][i+1])
    /\ acks \in [Sites -> SUBSET Sites]
Invariant ==
    /\ (* each queue holds at most one request per site *)
      \A s \in Sites : \A i \in 1 .. Len(reqQ[s]) :
        \A j \in i+1 .. Len(reqQ[s]) : reqQ[s][j].site # reqQ[s][i].site
    /\ (* requests stay in queue until "free" message received *)
      \A s,t \in Sites :
        (\E i \in 1 .. Len(network[s][t]) : network[s][t][i].kind = "free")
        => \E j \in 1 .. Len(reqQ[t]) : reqQ[t][j].site = s
    /\ (* site is in critical section only if at the head of every request queue *)
      \A s \in Sites : pc[s] = "crit" => \A t \in Sites : Head(reqQ[t]).site = s
```

Properties: Liveness == \A s \in Sites : pc[s] = "enter" ~> pc[s] = "crit"
Specification size: 150 lines
Ease of understanding: Difficult to understand

2. Summarize and compare the efficiency of running TLC, separately for checking safety and liveness. Include at least these attributes: ease of setting up and running; and for different process numbers and maximum clock values, the number of states searched and the time taken to check (in a similar form as Page 278 of Lamport's PODC 2000 Tutorial slides).

Specification LamportMutext.tla

Procs	MaxClock	States found	Distinct States	Time(MM:SS) **safety**	Time(MM:SS) **liveness**
2	5	1326	685	00:13	00:14
2	6	2001	1043	00:12	00:13
2	7	2812	1475	00:10	00:12
2	8	3759	1981	00:13	00:15
2	9	4842	2561	00:14	00:18
3	2	2203	511	00:15	00:16
3	3	41533	10209	00:11	00:13
3	4	276114	70472	00:12	00:23

Table Specification Lamport Mutex.tla safety and liveness statistics

Specification 2

Procs	MaxClock	States found	Distinct States	Time(MM:SS) **safety**	Time(MM:SS) **liveness**
2	5	35833	5733	00:15	00:18
2	6	119876	18009	00:13	00:15
2	7	347524	49365	00:16	00:21
2	8	903671	122045	00:19	00:31
2	9	2157020	278208	00:26	01:15
3	2	6828	887	00:18	00:19
3	3	301566	35706	00:16	00:20
3	4	5606761	622560	00:46	04:39

Table Specification 2 safety and liveness statistics

Specification Mutex.tla

Procs	MaxClock	States found	Distinct States	Time(MM:SS) **safety**)	Time(MM:SS) **liveness**
2	5	23888	7711	00:09	00:11
2	6	43121	13863	00:08	00:17
2	7	67446	21625	00:09	00:19
2	8	95988	30727	00:16	00:18
2	9	128362	41025	00:13	00:23
3	2	21646	3990	00:15	00:22
3	3	578165	99411	00:14	01:15

3	4	7763351	1309654	01:34	43:08

Table Specification Mutex.tla safety and liveness statistics

Summary:

1. All the specs requires Number of process and clock value as inputs
2. For Spec 2 we need to specify order of process.
3. Liveness takes longer than safety

References:

- https://lamport.azurewebsites.net/tla/tla.html
- Discussed Spec 2 implementation with Saish Sali
- https://amturing.acm.org/p558-lamport.pdf