MODULE *Mutex_og*

```
EXTENDS Naturals, Sequences, FiniteSets, TLC

CONSTANTS N, MaxClock

Pid ≜ 1 . . N
ClockVal ≜ 0 . . MaxClock + 1
Message ≜ [time : ClockVal, type : {"Request", "Release", "AckReq"}]

  --algorithm LogicalClocks

variables
  channel = [source ∈ Pid ↦ [destination ∈ Pid ↦ ⟨⟩]],
  crit = {}

define
  LogClockLt(reqs, p, q) ≜
     ∨ reqs[q] = 0
     ∨ reqs[p] < reqs[q]
     ∨ reqs[p] = reqs[q] ∧ p < q

  ChanHead(dst, type) ≜
    {src ∈ Pid : ∧ Len(channel[src][dst]) > 0
                 ∧ Head(channel[src][dst]).type = type
    }

  Max(a, b) ≜ IF a ≤ b THEN b ELSE a
end define

macro Receive(type, clock, src, time)begin
  with s ∈ ChanHead(self, type) do
    src := s ;
    time := Head(channel[src][self]).time ;
    channel := [channel EXCEPT ![src][self] = Tail(channel[src][self])]
  end with
end macro

macro Broadcast(clock, msg)begin
  channel :=
    [channel EXCEPT ![self] =
      [dst ∈ Pid ↦
        IF dst = self THEN channel[self][self]
                      ELSE Append(channel[self][dst], msg)]]
end macro

macro SendTo(clock, dst, msg)begin
  channel :=
    [channel EXCEPT ![self][dst] = Append(channel[self][dst], msg)]
```

```
end macro

macro $EnterCritSec()$ begin
  $crit := crit \cup \{self\}$
end macro

macro $ExitCritSec()$ begin
  $crit := crit \setminus \{self\}$
end macro

process $Proc \in Pid$
variables
  $clock = 1$,
  $acks = \{\}$,
  $requests = [pid \in Pid \mapsto 0]$,
  $time$,
  $src$

begin
  $loop$: while TRUE do
    either
      when $requests[self] = 0$ ;
        $Broadcast(clock, [time \mapsto clock, type \mapsto$ "Request"$])$ ;
        $requests := [requests$ EXCEPT $![self] = clock]$ ;
        $acks := \{self\}$
    or
      $Receive($"AckReq"$, clock, src, time)$ ;
        $clock := Max(clock, time)$ ;
        $acks := acks \cup \{src\}$
    or
      when $\wedge\ self \notin crit$
           $\wedge\ acks = Pid$
           $\wedge\ \forall p \in Pid : p \neq self \Rightarrow$
                            $LogClockLt(requests, self, p)$ ;
        $EnterCritSec()$ ;
    or
      when $self \in crit$ ;
        $requests := [requests$ EXCEPT $![self] = 0]$ ;
        $ExitCritSec()$ ;
        $acks := \{\}$ ;
        $Broadcast(clock, [time \mapsto clock, type \mapsto$ "Release"$])$

    or
      $Receive($"Request"$, clock, src, time)$ ;
        $requests := [requests$ EXCEPT $![src] = time]$ ;
        $clock := Max(clock, time)$ ;
     $L2$: $SendTo(clock, src, [time \mapsto clock + 1, type \mapsto$ "AckReq"$])$
    or
        Receive("Release", clock, src, time) ;
          clock := Max(clock, time) ;
          requests := [requests EXCEPT ![src] = 0] ;
    end either ;
    tic:   clock := clock + 1
  end while ;
end process

end algorithm
 BEGIN TRANSLATION (chksum(pcal) = "eb5ff142" ∧ chksum(tla) = "bc5fba51")
CONSTANT defaultInitValue
VARIABLES channel, crit, pc

 define statement
LogClockLt(reqs, p, q) ≜
  ∨ reqs[q] = 0
  ∨ reqs[p] < reqs[q]
  ∨ reqs[p] = reqs[q] ∧ p < q

ChanHead(dst, type) ≜
  {src ∈ Pid : ∧ Len(channel[src][dst]) > 0
               ∧ Head(channel[src][dst]).type = type
  }

Max(a, b) ≜ IF a ≤ b THEN b ELSE a

VARIABLES clock, acks, requests, time, src

vars ≜ ⟨channel, crit, pc, clock, acks, requests, time, src⟩

ProcSet ≜ (Pid)

Init ≜ Global variables
        ∧ channel = [source ∈ Pid ↦ [destination ∈ Pid ↦ ⟨⟩]]
        ∧ crit = {}
        Process Proc
        ∧ clock = [self ∈ Pid ↦ 1]
        ∧ acks = [self ∈ Pid ↦ {}]
        ∧ requests = [self ∈ Pid ↦ [pid ∈ Pid ↦ 0]]
        ∧ time = [self ∈ Pid ↦ defaultInitValue]
        ∧ src = [self ∈ Pid ↦ defaultInitValue]
        ∧ pc = [self ∈ ProcSet ↦ "loop"]

loop(self) ≜ ∧ pc[self] = "loop"
             ∧ ∨ ∧ requests[self][self] = 0
                 ∧ channel' = [channel EXCEPT ![self] =
                                  [dst ∈ Pid ↦
```

```
                    $\text{IF } dst = self \text{ THEN } channel[self][self]$
                                      $\text{ELSE } Append(channel[self][dst], ([time \mapsto clock[self], type \mapsto$
   $\land requests' = [requests \text{ EXCEPT } ![self] = [requests[self] \text{ EXCEPT } ![self] = clock[self]]]$
   $\land acks' = [acks \text{ EXCEPT } ![self] = \{self\}]$
   $\land pc' = [pc \text{ EXCEPT } ![self] = \text{"tic"}]$
   $\land \text{UNCHANGED } \langle crit, clock, time, src\rangle$
$\lor \land \exists s \in ChanHead(self, \text{"AckReq"}) :$
         $\land src' = [src \text{ EXCEPT } ![self] = s]$
         $\land time' = [time \text{ EXCEPT } ![self] = Head(channel[src'[self]][self]).time]$
         $\land channel' = [channel \text{ EXCEPT } ![src'[self]][self] = Tail(channel[src'[self]][self])]$
   $\land clock' = [clock \text{ EXCEPT } ![self] = Max(clock[self], time'[self])]$
   $\land acks' = [acks \text{ EXCEPT } ![self] = acks[self] \cup \{src'[self]\}]$
   $\land pc' = [pc \text{ EXCEPT } ![self] = \text{"tic"}]$
   $\land \text{UNCHANGED } \langle crit, requests\rangle$
$\lor \land \land self \notin crit$
      $\land acks[self] = Pid$
      $\land \forall p \in Pid : p \neq self \Rightarrow$
                    $LogClockLt(requests[self], self, p)$
   $\land crit' = (crit \cup \{self\})$
   $\land pc' = [pc \text{ EXCEPT } ![self] = \text{"tic"}]$
   $\land \text{UNCHANGED } \langle channel, clock, acks, requests, time, src\rangle$
$\lor \land self \in crit$
   $\land requests' = [requests \text{ EXCEPT } ![self] = [requests[self] \text{ EXCEPT } ![self] = 0]]$
   $\land crit' = crit \setminus \{self\}$
   $\land acks' = [acks \text{ EXCEPT } ![self] = \{\}]$
   $\land channel' = [channel \text{ EXCEPT } ![self] =$
                  $[dst \in Pid \mapsto$
                    $\text{IF } dst = self \text{ THEN } channel[self][self]$
                                      $\text{ELSE } Append(channel[self][dst], ([time \mapsto clock[self], type \mapsto$
   $\land pc' = [pc \text{ EXCEPT } ![self] = \text{"tic"}]$
   $\land \text{UNCHANGED } \langle clock, time, src\rangle$
$\lor \land \exists s \in ChanHead(self, \text{"Request"}) :$
         $\land src' = [src \text{ EXCEPT } ![self] = s]$
         $\land time' = [time \text{ EXCEPT } ![self] = Head(channel[src'[self]][self]).time]$
         $\land channel' = [channel \text{ EXCEPT } ![src'[self]][self] = Tail(channel[src'[self]][self])]$
   $\land requests' = [requests \text{ EXCEPT } ![self] = [requests[self] \text{ EXCEPT } ![src'[self]] = time'[self]]]$
   $\land clock' = [clock \text{ EXCEPT } ![self] = Max(clock[self], time'[self])]$
   $\land pc' = [pc \text{ EXCEPT } ![self] = \text{"L2"}]$
   $\land \text{UNCHANGED } \langle crit, acks\rangle$
$\lor \land \exists s \in ChanHead(self, \text{"Release"}) :$
         $\land src' = [src \text{ EXCEPT } ![self] = s]$
         $\land time' = [time \text{ EXCEPT } ![self] = Head(channel[src'[self]][self]).time]$
         $\land channel' = [channel \text{ EXCEPT } ![src'[self]][self] = Tail(channel[src'[self]][self])]$
   $\land clock' = [clock \text{ EXCEPT } ![self] = Max(clock[self], time'[self])]$
   $\land requests' = [requests \text{ EXCEPT } ![self] = [requests[self] \text{ EXCEPT } ![src'[self]] = 0]]$
```

$$\begin{array}{l}
\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"tic"}] \\
\land\ \text{UNCHANGED } \langle crit,\ acks \rangle
\end{array}$$

$$tic(self) \triangleq \begin{array}{l}
\land\ pc[self] = \text{"tic"} \\
\land\ clock' = [clock \text{ EXCEPT } ![self] = clock[self] + 1] \\
\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"loop"}] \\
\land\ \text{UNCHANGED } \langle channel,\ crit,\ acks,\ requests,\ time,\ src \rangle
\end{array}$$

$$L2(self) \triangleq \begin{array}{l}
\land\ pc[self] \ \ = \text{"L2"} \\
\land\ channel' = [channel \text{ EXCEPT } ![self][src[self]] = Append(channel[self][src[self]], ([time \mapsto clock \\
\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"tic"}] \\
\land\ \text{UNCHANGED } \langle crit,\ clock,\ acks,\ requests,\ time,\ src \rangle
\end{array}$$

$$Proc(self) \triangleq loop(self) \lor tic(self) \lor L2(self)$$

$$Next \triangleq (\exists\, self \in Pid : Proc(self))$$

$$Spec \triangleq Init \land \Box[Next]_{vars}$$

END TRANSLATION

$$View \triangleq \langle channel,\ crit,\ clock,\ acks,\ requests,\ pc \rangle$$

$$MutualExclusion \triangleq Cardinality(crit) < 2$$