MODULE *Quorum*

EXTENDS *Naturals*, *Sequences*, *FiniteSets*, *TLC*

CONSTANTS *numClients*, *numReplicas*, *ReadQuorum*, *WriteQuorum*, *MaxClock*

TODO: Verify all uses of *Pid*

```
$Pid \triangleq 1 \,..\, numClients + numReplicas$
$ClientPids \triangleq 1 \,..\, numClients$
$ReplicaPids \triangleq numClients + 1 \,..\, numClients + numReplicas$

$ReadQuorums \triangleq \{x \in \text{SUBSET } ReplicaPids : Cardinality(x) = ReadQuorum\}$
$WriteQuorums \triangleq \{x \in \text{SUBSET } ReplicaPids : Cardinality(x) = WriteQuorum\}$

$ClockVal \triangleq 0 \,..\, MaxClock + 1$
$Message \triangleq [time : ClockVal, type : \{\text{“Read”}, \text{“Write”}, \text{“Ack”}\}, stamp : ClockVal, value : 1 \,..\, 20]$
```

```
  --algorithm Quorum

variables $channel = [source \in Pid \mapsto [destination \in Pid \mapsto \langle\rangle]]$

define
  $GetMsgSrcs(dst, type) \triangleq$
    $\{src \in Pid : \land Len(channel[src][dst]) > 0$
                  $\land Head(channel[src][dst]).type = type$
    $\}$

  $Max(a, b) \triangleq \text{IF } a \leq b \text{ THEN } b \text{ ELSE } a$
end define

macro Receive(type, clock, src, time, stamp, value)begin
  with $s \in GetMsgSrcs(self, type)$ do
    $src := s$ ;
    $time := Head(channel[src][self]).time$ ;
    $stamp := Head(channel[src][self]).stamp$ ;
    $value := Head(channel[src][self]).value$ ;
    $channel := [channel \text{ EXCEPT } ![src][self] = Tail(channel[src][self])]$
  end with
end macro

macro BroadcastTo(dsts, clock, msg)begin
  $channel :=$
    $[channel \text{ EXCEPT } ![self] =$
      $[dst \in Pid \mapsto$
        $\text{IF } dst = self \text{ THEN } channel[self][self]$
                       $\text{ELSE } Append(channel[self][dst], msg)]]$
end macro

macro SendTo(clock, dst, msg)begin
```

```
  $channel :=$
    $[channel$ EXCEPT $![self][dst] = Append(channel[self][dst], msg)]$
end macro

process $Proc \in Pid$
variables
  $clock = 1$,
  $acks = \langle\rangle$,
  $requests = [pid \in Pid \mapsto 0]$,
  $src$,
  $time$,
  $stamper$,
  $value$,

  $TS = 1$,
  $state = 0$,

  $type = 0$

begin
    $loop$: while TRUE do
        if $self \in ClientPids$ then
        Client actions
            either
        Send “Read” or “Write”
                when $requests[self] = 0$;
                    either
             Send “Read”
                        with $quorum \in ReadQuorums$ do
                  print $\langle quorum \rangle$;
                            $BroadcastTo(quorum, clock, [time \mapsto clock, type \mapsto$ “Read”, $stamp \mapsto clock, value \mapsto$
                            $requests := [requests$ EXCEPT $![self] = clock]$;
                            $type := 1$;
                        end with
                    or
             Send “Write
                        with $quorum \in WriteQuorums$ do
                            with $val \in 1 \,..\, 20$ do
                                $BroadcastTo(quorum, clock, [time \mapsto clock, type \mapsto$ “Write”, $stamp \mapsto clock, val$
                                $requests := [requests$ EXCEPT $![self] = clock]$;
                                $type := 2$;
                            end with
                        end with
                    end either ;
            or
        Receive ”$Ack$“
```

```
                $Receive(\text{“Ack”}, clock, src, time, stamper, value)$ ;
                $acks := Append(acks, [source \mapsto src, stamp \mapsto stamper, val \mapsto value])$ ;
                $clock := Max(clock, time)$ ;
            or
        Do work: ”Read“
                when $(type = 1 \land Len(acks) = ReadQuorum)$ ;
                    $acks := \langle\rangle$ ;
                    $requests := [requests \text{ EXCEPT } ![self] = 0]$ ;
                    $type := 0$ ;
            or
        Do work: ”Write“
                when $(type = 2 \land Len(acks) = WriteQuorum)$ ;
                    $acks := \langle\rangle$ ;
                    $requests := [requests \text{ EXCEPT } ![self] = 0]$ ;
                    $type := 0$ ;
            end either ;
         else
        Replica actions
            either
        Receive ”Read“
                $Receive(\text{“Read”}, clock, src, time, stamper, value)$ ;
                $clock := Max(clock, time)$ ;
                $L2$: $SendTo(clock, src, [time \mapsto clock + 1, type \mapsto \text{“Ack”}, stamp \mapsto TS, value \mapsto state])$
            or
        Receive ”Write
                $Receive(\text{“Write”}, clock, src, time, stamper, value)$ ;
                if $(stamper > TS) \lor (stamper = TS \land src > self)$ then
                    $TS := stamper$ ;
                    $state := value$ ;
                    $clock := Max(clock, time)$ ;
                    $L3$: $SendTo(clock, src, [time \mapsto clock + 1, type \mapsto \text{“Ack”}, stamp \mapsto TS, value \mapsto state])$
                 else
                    $clock := Max(clock, time)$ ;
                    $L4$: $SendTo(clock, src, [time \mapsto clock + 1, type \mapsto \text{“Ack”}, stamp \mapsto TS, value \mapsto 0])$
                end if  ;
            end either ;
        end if ;
        $tic$:   $clock := clock + 1$
    end while ;
end process

end algorithm
```

BEGIN TRANSLATION $(chksum(pcal) = \text{“}85c970db\text{”} \land chksum(tla) = \text{“}b3dc63aa\text{”})$

CONSTANT $defaultInitValue$

VARIABLES $channel, pc$

```
define statement
$GetMsgSrcs(dst, type) \triangleq$
  $\{src \in Pid : \land Len(channel[src][dst]) > 0$
                 $\land Head(channel[src][dst]).type = type$
  $\}$

$Max(a, b) \triangleq$ IF $a \leq b$ THEN $b$ ELSE $a$

VARIABLES $clock, acks, requests, src, time, stamper, value, TS, state, type$

$vars \triangleq \langle channel, pc, clock, acks, requests, src, time, stamper, value, TS,$
          $state, type \rangle$

$ProcSet \triangleq (Pid)$

$Init \triangleq$ Global variables
        $\land channel = [source \in Pid \mapsto [destination \in Pid \mapsto \langle\rangle]]$
        Process $Proc$
        $\land clock = [self \in Pid \mapsto 1]$
        $\land acks = [self \in Pid \mapsto \langle\rangle]$
        $\land requests = [self \in Pid \mapsto [pid \in Pid \mapsto 0]]$
        $\land src = [self \in Pid \mapsto defaultInitValue]$
        $\land time = [self \in Pid \mapsto defaultInitValue]$
        $\land stamper = [self \in Pid \mapsto defaultInitValue]$
        $\land value = [self \in Pid \mapsto defaultInitValue]$
        $\land TS = [self \in Pid \mapsto 1]$
        $\land state = [self \in Pid \mapsto 0]$
        $\land type = [self \in Pid \mapsto 0]$
        $\land pc = [self \in ProcSet \mapsto$ "loop"$]$

$loop(self) \triangleq \land pc[self] =$ "loop"
              $\land$ IF $self \in ClientPids$
                    THEN $\land \lor \land requests[self][self] = 0$
                               $\land \lor \land \exists\, quorum \in ReadQuorums :$
                                          $\land channel' = [channel$ EXCEPT $![self] =$
                                                         $[dst \in Pid \mapsto$
                                                          IF $dst = self$ THEN $channel[self][self]$
                                                                          ELSE $Append(channel[self][dst], ([tim$
                                          $\land requests' = [requests$ EXCEPT $![self] = [requests[self]$ EXCEPT $![se$
                                          $\land type' = [type$ EXCEPT $![self] = 1]$
                                  $\lor \land \exists\, quorum \in WriteQuorums :$
                                          $\exists\, val \in 1 \,..\, 20 :$
                                            $\land channel' = [channel$ EXCEPT $![self] =$
                                                           $[dst \in Pid \mapsto$
                                                            IF $dst = self$ THEN $channel[self][self]$
                                                                            ELSE $Append(channel[self][dst], ([t$
                                            $\land requests' = [requests$ EXCEPT $![self] = [requests[self]$ EXCEPT $![s$
```

```
                      $\land\ type' = [type\ \text{EXCEPT}\ ![self] = 2]$
                  $\land\ \text{UNCHANGED}\ \langle clock,\ acks,\ src,\ time,\ stamper,\ value\rangle$
              $\lor\ \land\ \exists\, s \in GetMsgSrcs(self,\ \text{“Ack”}):$
                      $\land\ src' = [src\ \text{EXCEPT}\ ![self] = s]$
                      $\land\ time' = [time\ \text{EXCEPT}\ ![self] = Head(channel[src'[self]][self]).time]$
                      $\land\ stamper' = [stamper\ \text{EXCEPT}\ ![self] = Head(channel[src'[self]][self]).st$
                      $\land\ value' = [value\ \text{EXCEPT}\ ![self] = Head(channel[src'[self]][self]).value]$
                      $\land\ channel' = [channel\ \text{EXCEPT}\ ![src'[self]][self] = Tail(channel[src'[self]]$
                  $\land\ acks' = [acks\ \text{EXCEPT}\ ![self] = Append(acks[self],\ [source \mapsto src'[self],\ stan$
                  $\land\ clock' = [clock\ \text{EXCEPT}\ ![self] = Max(clock[self],\ time'[self])]$
                  $\land\ \text{UNCHANGED}\ \langle requests,\ type\rangle$
              $\lor\ \land\ (type[self] = 1 \land Len(acks[self]) = ReadQuorum)$
                  $\land\ acks' = [acks\ \text{EXCEPT}\ ![self] = \langle\rangle]$
                  $\land\ requests' = [requests\ \text{EXCEPT}\ ![self] = [requests[self]\ \text{EXCEPT}\ ![self] = 0]]$
                  $\land\ type' = [type\ \text{EXCEPT}\ ![self] = 0]$
                  $\land\ \text{UNCHANGED}\ \langle channel,\ clock,\ src,\ time,\ stamper,\ value\rangle$
              $\lor\ \land\ (type[self] = 2 \land Len(acks[self]) = WriteQuorum)$
                  $\land\ acks' = [acks\ \text{EXCEPT}\ ![self] = \langle\rangle]$
                  $\land\ requests' = [requests\ \text{EXCEPT}\ ![self] = [requests[self]\ \text{EXCEPT}\ ![self] = 0]]$
                  $\land\ type' = [type\ \text{EXCEPT}\ ![self] = 0]$
                  $\land\ \text{UNCHANGED}\ \langle channel,\ clock,\ src,\ time,\ stamper,\ value\rangle$
          $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“tic”}]$
          $\land\ \text{UNCHANGED}\ \langle TS,\ state\rangle$
  ELSE    $\land\ \lor\ \land\ \exists\, s \in GetMsgSrcs(self,\ \text{“Read”}):$
                      $\land\ src' = [src\ \text{EXCEPT}\ ![self] = s]$
                      $\land\ time' = [time\ \text{EXCEPT}\ ![self] = Head(channel[src'[self]][self]).time]$
                      $\land\ stamper' = [stamper\ \text{EXCEPT}\ ![self] = Head(channel[src'[self]][self]).st$
                      $\land\ value' = [value\ \text{EXCEPT}\ ![self] = Head(channel[src'[self]][self]).value]$
                      $\land\ channel' = [channel\ \text{EXCEPT}\ ![src'[self]][self] = Tail(channel[src'[self]]$
                  $\land\ clock' = [clock\ \text{EXCEPT}\ ![self] = Max(clock[self],\ time'[self])]$
                  $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“L2”}]$
                  $\land\ \text{UNCHANGED}\ \langle TS,\ state\rangle$
              $\lor\ \land\ \exists\, s \in GetMsgSrcs(self,\ \text{“Write”}):$
                      $\land\ src' = [src\ \text{EXCEPT}\ ![self] = s]$
                      $\land\ time' = [time\ \text{EXCEPT}\ ![self] = Head(channel[src'[self]][self]).time]$
                      $\land\ stamper' = [stamper\ \text{EXCEPT}\ ![self] = Head(channel[src'[self]][self]).st$
                      $\land\ value' = [value\ \text{EXCEPT}\ ![self] = Head(channel[src'[self]][self]).value]$
                      $\land\ channel' = [channel\ \text{EXCEPT}\ ![src'[self]][self] = Tail(channel[src'[self]]$
                  $\land\ \text{IF}\ (stamper'[self] > TS[self]) \lor (stamper'[self] = TS[self] \land src'[self] > self$
                        THEN  $\land\ TS' = [TS\ \text{EXCEPT}\ ![self] = stamper'[self]]$
                              $\land\ state' = [state\ \text{EXCEPT}\ ![self] = value'[self]]$
                              $\land\ clock' = [clock\ \text{EXCEPT}\ ![self] = Max(clock[self],\ time'[self])]$
                              $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“L3”}]$
                        ELSE  $\land\ clock' = [clock\ \text{EXCEPT}\ ![self] = Max(clock[self],\ time'[self])]$
                              $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“L4”}]$
```

```
                                                    ∧ UNCHANGED ⟨TS, state⟩
                                   ∧ UNCHANGED ⟨acks, requests, type⟩

tic(self) ≜ ∧ pc[self] = "tic"
            ∧ clock′ = [clock EXCEPT ![self] = clock[self] + 1]
            ∧ pc′ = [pc EXCEPT ![self] = "loop"]
            ∧ UNCHANGED ⟨channel, acks, requests, src, time, stamper,
                         value, TS, state, type⟩

L2(self) ≜ ∧ pc[self]   = "L2"
           ∧ channel′ = [channel EXCEPT ![self][src[self]] = Append(channel[self][src[self]], ([time ↦ clock
           ∧ pc′ = [pc EXCEPT ![self] = "tic"]
           ∧ UNCHANGED ⟨clock, acks, requests, src, time, stamper, value,
                        TS, state, type⟩

L3(self) ≜ ∧ pc[self]   = "L3"
           ∧ channel′ = [channel EXCEPT ![self][src[self]] = Append(channel[self][src[self]], ([time ↦ clock
           ∧ pc′ = [pc EXCEPT ![self] = "tic"]
           ∧ UNCHANGED ⟨clock, acks, requests, src, time, stamper, value,
                        TS, state, type⟩

L4(self) ≜ ∧ pc[self]   = "L4"
           ∧ channel′ = [channel EXCEPT ![self][src[self]] = Append(channel[self][src[self]], ([time ↦ clock
           ∧ pc′ = [pc EXCEPT ![self] = "tic"]
           ∧ UNCHANGED ⟨clock, acks, requests, src, time, stamper, value,
                        TS, state, type⟩

Proc(self) ≜ loop(self) ∨ tic(self) ∨ L2(self) ∨ L3(self) ∨ L4(self)

Next ≜ (∃ self ∈ Pid : Proc(self))

Spec ≜ Init ∧ □[Next]_vars

 END TRANSLATION

ReplicaConsistency ≜ ∀ p1, p2 ∈ ReplicaPids : (TS[p1] = TS[p2] ⇒ state[p1] = state[p2])

 Essential variables to be monitored by TLC
View ≜ ⟨channel, TS, state, clock, acks, requests, pc⟩
```