```
──────────────────── MODULE skeen ────────────────────

 Skeen's protocol [1] is encoded in TLA+.

 [1] Skeen, D.: (1985), Referenced in [1], unpublished communication.

 [2] Birman, K.P., Joseph, T.A.: Reliable communication in the presence of failures. ACM
     Transactions on Computer Systems (TOCS) 5(1), 47–76 (1987)

  Thanh-Hai Tran, Igor Konnov, Josef Widder, 2021 This file is a subject to the license that is
  bundled together with this package and can be found in the file LICENSE.
EXTENDS Integers,  FiniteSets, Sequences, TLC


 Type operator for APALACHE
a <: b ≜ a

 All variables
VARIABLE clock, phase, localTS, globalTS, rcvdMcastID, mcastedID,
           inTransit, delivered, proposeTS, dCntr

vars ≜ ⟨clock, phase, localTS, globalTS, rcvdMcastID, mcastedID,
          inTransit, delivered, proposeTS, dCntr⟩

CONSTANT N,              the number of processes indexed from 1 to N
         M,              the number of multicast messages indexed from 1 to M
         GroupDest,      an array whose i-th element describes the group of addressees of message i
         Mcaster,        an array whose i-th element describes the multicaster of message i
         MaxClock        the bound of local clocks


Proc ≜ 1 . . N
McastID ≜ 1 . . M
MType ≜ 10              type of multicast messages
PType  ≜ 11             type of proposed messages
Start   ≜ 12
Proposed ≜ 13
Committed ≜ 14

ASSUME
  ∧ N ∈ Int
  ∧ ∀ id    ∈ McastID : GroupDest[id] ∈ SUBSET Int
  ∧ MType ∈ Int
  ∧ PType  ∈ Int
  ∧ Start   ∈ Int
  ∧ Proposed ∈ Int
  ∧ Committed ∈ Int
  ∧ M = Cardinality(McastID)

McastMsgPhase ≜ {Start, Proposed, Committed}
```

$McastPhase \triangleq [McastID \rightarrow McastMsgPhase]$

TimestampNull: the init value of local timestamps and global timestamps

Type of *TimestampNull* is $[t \mapsto Int,\ g \mapsto Int]$

$GroupNull \triangleq 0$

$TimeNull \triangleq 0$

$TimestampNull \triangleq [t \mapsto TimeNull,\ g \mapsto GroupNull] <: [t \mapsto Int,\ g \mapsto Int]$

$Time \triangleq 1 \,..\, MaxClock$

$ProcWithNull \triangleq 0 \,..\, N$

The set of all possible in-transit messages

$$\begin{aligned}
TimestampSet &\triangleq [t : Time,\ g : Proc] \cup \{TimestampNull\} \\
&\qquad\qquad <: \{[t \mapsto Int,\ g \mapsto Int]\} \cup\ (\{TimestampNull\} <: \{[g : Int,\ t : Int]\}) \\
McastMsgSet &\triangleq ([t : Time,\ id : McastID,\ type : \{MType\},\ source : Proc] \\
&\qquad\qquad <: \{[type \mapsto Int,\ t \mapsto Int,\ id \mapsto Int,\ source \mapsto Int]\}) \\
ProposeMsgSet &\triangleq ([t : Time,\ id : McastID,\ type : \{PType\},\ source : Proc] \\
&\qquad\qquad <: \{[type \mapsto Int,\ t \mapsto Int,\ id \mapsto Int,\ source \mapsto Int]\}) \\
InTransitMsgSet &\triangleq McastMsgSet \cup ProposeMsgSet
\end{aligned}$$

The initialized states
- clock: local clocks
- $phase[p][m]$: stores the status of message m at process p
- $localTS[p][m]$: stores the local timestamp issued by process p for message m
- $globalTS[p][m]$: stores the global timestamp issued by process p for message m
- $delivered[p][m]$: refers to whether process p has delivered message m
- $rcvdMcastID[p][m]$: a set of multicast messages that process p has received
- $proposeTS[p][m]$: stores a set of proposals for messages m
- $mcastedID$: a set of messages that were multicast
- $inTransit[p][q]$: a set of in-transit messages from process p to process q
- $dCntr[p][m]$ to keep trach of how many times process p has delivered message m.

$$\begin{aligned}
&Init \triangleq \\
&\quad \land\ clock = [p \in Proc \mapsto 0] \\
&\quad \land\ phase = [p \in Proc \mapsto [m \in McastID \mapsto Start]] \\
&\quad \land\ localTS = [p \in Proc \mapsto [m \in McastID \mapsto TimestampNull]] \\
&\quad \land\ globalTS = [p \in Proc \mapsto [m \in McastID \mapsto TimestampNull]] \\
&\quad \land\ delivered = [p \in Proc \mapsto [m \in McastID \mapsto \text{FALSE}]] \\
&\quad \land\ rcvdMcastID = [p \in Proc \mapsto (\{\} <: \{Int\})] \\
&\quad \land\ proposeTS = [p \in Proc \mapsto [id \in McastID \mapsto (\{\} <: \{[type \mapsto Int,\ t \mapsto Int,\ id \mapsto Int,\ source \mapsto Int]\})]] \\
&\quad \land\ mcastedID = (\{\} <: \{Int\}) \\
&\quad \land\ inTransit = [p \in Proc \mapsto [q \in Proc \mapsto (\{\} <: \{[type \mapsto Int,\ t \mapsto Int,\ id \mapsto Int,\ source \mapsto Int]\})]] \\
&\quad \land\ dCntr = [p \in Proc \mapsto [id \in McastID \mapsto 0]]
\end{aligned}$$

$Max(a,\ b) \triangleq \text{IF } a > b \text{ THEN } a \text{ ELSE } b$

```
Process snder multicasts the message whose identifier is mid.
The multicast message for message mid is tag with a local timestamp issued by process snder.
Multicast(mid) ≜
  LET snder ≜ Mcaster[mid]
  IN   ∧ mid ∉ mcastedID
       ∧ clock[snder] < MaxClock                  Only for bounded model checking
       ∧ snder ∈ GroupDest[mid]
       ∧ mcastedID′ = mcastedID ∪ {mid}         Marks that message mid is multicast
       ∧ LET time ≜ clock[snder] + 1
             msg ≜  ([type ↦ MType, id ↦ mid, t ↦ time, source ↦ snder]
                         <: [type ↦ Int, t ↦ Int, source ↦ Int, id ↦ Int])
         IN      ∧ inTransit′ = [p ∈ Proc ↦ [q ∈ Proc ↦
                                     IF p = snder ∧ q ∈ GroupDest[mid]
                                      THEN inTransit[p][q] ∪ {msg}
                                      ELSE  inTransit[p][q]]]
                 ∧ clock′ = [clock EXCEPT ![snder] = time]
                 ∧ UNCHANGED ⟨phase, proposeTS, rcvdMcastID, localTS, globalTS, delivered, dCntr⟩


Pick the in-transit message with the smallest timestamp from process snder to process rcver
isYoungestMsg(snder, rcver, msg) ≜
  ∀ m ∈ inTransit[snder][rcver] : msg.t ≤ m.t


Receives a multicast message
ReceiveMulticast(snder, rcver, msg) ≜
  ∧ clock[rcver] < MaxClock
  ∧ msg.type = MType
  ∧ isYoungestMsg(snder, rcver, msg)     msg must have the smallest timestamp in inTransit[snder][rcver]
  ∧ rcvdMcastID′ = [rcvdMcastID EXCEPT ![rcver] = rcvdMcastID[rcver] ∪ {msg.id}]
  ∧ UNCHANGED ⟨proposeTS, globalTS, delivered, mcastedID, dCntr⟩
  ∧ LET mid ≜ msg.id
        time ≜ clock[rcver] + 1
        newTS ≜ [t ↦ time, g ↦ rcver]          the local timestamp for message msg.id
        newMsg ≜ ([type ↦ PType, id ↦ mid, source ↦ rcver, t ↦ time]
                     <: [type ↦ Int, t ↦ Int, source ↦ Int, id ↦ Int])     the proposal for message msg.id
    IN   ∧ clock′ = [clock EXCEPT ![rcver] = clock[rcver] + 1]
         ∧ localTS′ = [localTS EXCEPT ![rcver][mid] = newTS]
         ∧ phase′ = [phase EXCEPT ![rcver][mid] = Proposed]
          Sends its proposal to every addressee of message msg.id
         ∧ IF snder ≠ rcver
             THEN inTransit′ = [p ∈ Proc ↦ [q ∈ Proc ↦
                                  IF p = rcver ∧ q ∈ GroupDest[mid]
                                   THEN inTransit[p][q] ∪ {newMsg}
                                   ELSE IF p = snder ∧ q = rcver
                                          THEN inTransit[p][q] \ {msg}
```

```
                                    ELSE  inTransit[p][q]]]
                ELSE  inTransit′ = [p ∈ Proc ↦ [q ∈ Proc ↦
                                   IF p = rcver ∧ q = rcver
                                    THEN (inTransit[p][q] ∪ {newMsg}) \ {msg}
                                    ELSE  IF p = rcver ∧ q ∈ GroupDest[mid]
                                           THEN inTransit[p][q] ∪ {newMsg}
                                           ELSE  inTransit[p][q]]]


Compare two timestamps based on lexicographical order
Less(ts1, ts2) ≜
  ∨ ts1.t < ts2.t
  ∨ ∧ ts1.t = ts2.t
    ∧ ts1.g < ts2.g



Check whether message id can be delivered to process p
The local timestamps of all committed messages must be greater than the global timestamp of message id
CanDeliver(p, id) ≜
  ∧ ¬delivered[p][id]
  ∧ phase′[p][id] = Committed
  ∧ ∀ mid ∈ rcvdMcastID′[p] :
       phase′[p][mid] = Proposed ⇒ Less(globalTS′[p][id], localTS′[p][mid])

Process rcver has received the proposals from all addressees of message id.
HasAllProposes(rcver, id) ≜
  ∀ p ∈ GroupDest[id] : ∃ m ∈ proposeTS′[rcver][id] : m.source = p

Pick a proposed message with the greatest local timestamp for message id
PickMsgWithMaxTS(rcver, id) ≜
  CHOOSE m ∈ proposeTS′[rcver][id] :
    ∀ m1 ∈ proposeTS′[rcver][id] :
       ∨ m1.t < m.t
       ∨ ∧ m1.t = m.t
         ∧ m1.source ≤ m.source

Process rcver has received a proposed message from process snder
ReceivePropose(snder, rcver, msg) ≜
  ∧ msg.type = PType
  ∧ isYoungestMsg(snder, rcver, msg)      msg must have the smallest timestamp in inTransit[snder][rcver]
  ∧ inTransit′ = [inTransit EXCEPT ![snder][rcver] = inTransit[snder][rcver] \ {msg}]
  ∧ LET ts ≜ [t ↦ msg.t, g ↦ msg.source]
        id ≜ msg.id
     IN    ∧ UNCHANGED ⟨localTS, mcastedID, rcvdMcastID ⟩
           ∧ proposeTS′ = [proposeTS EXCEPT ![rcver][id] = proposeTS[rcver][id] ∪ {msg}]
              Whether process rcver has received the proposals from all addressees of message id.
```

```
$\land$ IF $HasAllProposes(rcver, id)$
   THEN LET $m \triangleq PickMsgWithMaxTS(rcver, id)$
            $maxTS \triangleq [g \mapsto m.source, t \mapsto m.t]$
        IN     Set the global timestamp for message $msg.id$
          $\land$  $globalTS' = [globalTS$ EXCEPT $![rcver][id] = maxTS]$
                 Synchronizes the local clocks
          $\land$  $clock' = [clock$ EXCEPT $![rcver] = Max(clock[rcver], maxTS.t)]$
          $\land$  $phase' = [phase$ EXCEPT $![rcver][id] = Committed]$
          $\land$  $delivered' = [delivered$ EXCEPT $![rcver] = [mid \in McastID \mapsto$
                              IF $\neg delivered[rcver][mid]$
                               THEN $CanDeliver(rcver, mid)$
                               ELSE $delivered[rcver][mid]]]$
                 Update how many times $p$ has delivered message mid
          $\land$  $dCntr' = [dCntr$ EXCEPT $![rcver] = [mid \in McastID \mapsto$
                               IF $\neg delivered[rcver][mid] \land CanDeliver(rcver, mid)$
                                THEN $dCntr[rcver][mid] + 1$
                                ELSE $dCntr[rcver][mid]]]$
   ELSE UNCHANGED $\langle phase, globalTS, clock, delivered, dCntr \rangle$

Only to avoid deadlock checking
$Done \triangleq$
   $\land$ $\forall id \in McastID : \forall p \in GroupDest[id] : delivered[p][id]$
   $\land$ UNCHANGED $vars$

$Next \triangleq$
   $\lor \exists m \in McastID : Multicast(m)$
   $\lor \exists snder, rcver \in Proc : \exists msg \in inTransit[snder][rcver] : ReceiveMulticast(snder, rcver, msg)$
   $\lor \exists snder, rcver \in Proc : \exists msg \in inTransit[snder][rcver] : ReceivePropose(snder, rcver, msg)$
   $\lor$ $Done$

$Spec \triangleq$
   $\land$ $Init$
   $\land \Box[Next]_{vars}$
   $\land \mathrm{WF}_{vars}( \lor \exists m \in McastID : Multicast(m)$
              $\lor \exists snder, rcver \in Proc : \exists msg \in inTransit[snder][rcver] : ReceiveMulticast(snder, rcver, msg)$
              $\lor \exists snder, rcver \in Proc : \exists msg \in inTransit[snder][rcver] : ReceivePropose(snder, rcver, msg))$
```

* Modification History
* Last modified *Mon Sep* 20 16:46:38 *CEST* 2021 by *tran*
* Created *Tue Mar* 16 08:59:43 *CET* 2021 by *tran*