MODULE *skeen_intint_t*

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[1] *Skeen*, *D.*: (1985), Referenced in [1], unpublished communication.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

EXTENDS *Integers*, *FiniteSets*, *Sequences*, *TLC*

CONSTANT

@type: *Int*;
N, the number of processes indexed from 1 to *N*

@type: *Int*;
M, the number of multicast messages indexed from 1 to *M*

@type: *Seq*(*Int*);
Mcaster, an array whose i-th element describes the multicaster of message i

@type: *Seq*(*Set*(*Int*));
GroupDest, an array whose i-th element describes the group of addressees of message i

@type: *Int*;
MaxClock the bound of local clocks

All variables

VARIABLE

@type: *Int* → *Int*;
clock,

@type: ⟨*Int*, *Int*⟩ → *Int*;
phase,

@type: ⟨*Int*, *Int*⟩ → [*t*: *Int*, *g*: *Int*];
localTS,

@type: ⟨*Int*, *Int*⟩ → [*t*: *Int*, *g*: *Int*];
globalTS,

@type: *Int* → *Set*(*Int*);
rcvdMcastID,

@type: *Set*(*Int*);
mcastedID,

@type: ⟨*Int*, *Int*⟩ → *Set*([*type* : *Int*, *t* : *Int*, *id* : *Int*, *source* : *Int*]);
inTransit,

@type: ⟨*Int*, *Int*⟩ → *Bool*;
delivered,

@type: ⟨*Int*, *Int*⟩ → *Set*([*type* : *Int*, *t* : *Int*, *id* : *Int*, *source* : *Int*]);
proposeTS,

@type: ⟨*Int*, *Int*⟩ → *Int*;
dCntr

$vars \triangleq \langle clock, phase, localTS, globalTS, rcvdMcastID, mcastedID, inTransit, delivered, proposeTS, dCntr \rangle$

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

ASSUME
$\land N \in Int$
$\land \forall\, id \in McastID : GroupDest[id] \in \text{SUBSET } Int$
$\land MType \in Int$
$\land PType \in Int$
$\land Start \in Int$
$\land Proposed \in Int$
$\land Committed \in Int$
$\land M = Cardinality(McastID)$

$McastMsgPhase \triangleq \{Start, Proposed, Committed\}$

$McastPhase \triangleq [McastID \to McastMsgPhase]$

TimestampNull: the init value of local timestamps and global timestamps
Type of *TimestampNull* is $[t \mapsto Int, g \mapsto Int]$
$GroupNull \triangleq 0$
$TimeNull \triangleq 0$

type: $[t\text{: } Int, g\text{: } Int]$
$TimestampNull \triangleq [t \mapsto TimeNull, g \mapsto GroupNull]$
type: $Set(Int)$
$Time \triangleq 1 \,..\, MaxClock$
type: $Set(Int)$
$ProcWithNull \triangleq 0 \,..\, N$

The set of all possible in-transit messages
@type: $Set([t : Int, g : Int])$;
$TimestampSet \triangleq [t : Time, g : Proc] \cup \{TimestampNull\}$
@type: $Set([type : Int, t : Int, id : Int, source : Int])$;
$McastMsgSet \triangleq [t : Time, id : McastID, type : \{MType\}, source : Proc]$
@type: $Set([type : Int, t : Int, id : Int, source : Int])$;
$ProposeMsgSet \triangleq [t : Time, id : McastID, type : \{PType\}, source : Proc]$
@type: $Set([type : Int, t : Int, id : Int, source : Int])$;
$InTransitMsgSet \triangleq McastMsgSet \cup ProposeMsgSet$

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

$$
\begin{aligned}
&Init \triangleq \\
&\quad \land clock = [p \in Proc \mapsto 0] \\
&\quad \land phase = [\langle p, m\rangle \in Proc \times McastID \mapsto Start] \\
&\quad \land localTS = [\langle p, m\rangle \in Proc \times McastID \mapsto TimestampNull] \\
&\quad \land globalTS = [\langle p, m\rangle \in Proc \times McastID \mapsto TimestampNull] \\
&\quad \land delivered = [\langle p, m\rangle \in Proc \times McastID \mapsto \text{FALSE}] \\
&\quad \land rcvdMcastID = [p \in Proc \mapsto \{\}] \\
&\quad \land proposeTS = [\langle p, id\rangle \in Proc \times McastID \mapsto \{\}] \\
&\quad \land mcastedID = \{\} \\
&\quad \land inTransit = [\langle p, q\rangle \in Proc \times Proc \mapsto \{\}] \\
&\quad \land dCntr = [\langle p, id\rangle \in Proc \times McastID \mapsto 0]
\end{aligned}
$$

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

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*.

$$
\begin{aligned}
&Multicast(mid) \triangleq \\
&\quad \text{LET } snder \triangleq Mcaster[mid] \\
&\quad \text{IN} \quad \land mid \notin mcastedID \\
&\qquad \land clock[snder] < MaxClock \qquad \text{Only for bounded model checking} \\
&\qquad \land snder \in GroupDest[mid] \\
&\qquad \land mcastedID' = mcastedID \cup \{mid\} \qquad \text{Marks that message mid is multicast} \\
&\qquad \land \text{LET } time \triangleq clock[snder] + 1 \\
&\qquad\quad \text{@type: [type: } Int\text{, } t\text{: } Int\text{, } id\text{: } Int\text{, source: } Int\text{];} \\
&\qquad\quad msg \triangleq [type \mapsto MType,\ id \mapsto mid,\ t \mapsto time,\ source \mapsto snder] \\
&\qquad\quad \text{IN} \quad \land inTransit' = [\langle p, q\rangle \in Proc \times Proc \mapsto \\
&\qquad\qquad\qquad\qquad \text{IF } p = snder \land q \in GroupDest[mid] \\
&\qquad\qquad\qquad\qquad \text{THEN } inTransit[\langle p, q\rangle] \cup \{msg\} \\
&\qquad\qquad\qquad\qquad \text{ELSE } inTransit[\langle p, q\rangle]] \\
&\qquad\qquad\quad \land clock' = [clock \text{ EXCEPT } ![snder] = time] \\
&\qquad\qquad\quad \land \text{UNCHANGED } \langle phase, proposeTS, rcvdMcastID, localTS, globalTS, delivered, dCntr\rangle
\end{aligned}
$$

Pick the in-transit message with the smallest timestamp from process *snder* to process *rcver*
@type: (*Int*, *Int*, [type: *Int*, *t*: *Int*, *id*: *Int*, source: *Int*]) ⇒ *Bool*;

```
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
         @type: [type: Int, t: Int, id: Int, source: Int];
        newMsg ≜ [type ↦ PType, id ↦ mid, source ↦ rcver, t ↦ time]   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, q⟩ ∈ Proc × 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, q⟩ ∈ Proc × 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
@type: ([t : Int, g : Int], [t : Int, g : Int]) ⇒ Bool;
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
@type: (Int, Int) ⇒ Bool;
```

```
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.
        ∧ IF HasAllProposes(rcver, id)
          THEN LET m ≜ PickMsgWithMaxTS(rcver, id)
                   maxTS ≜ [g ↦ m.source, t ↦ m.t]
               IN      Set the global timestamp for message msg.id
                  ∧ globalTS' = [globalTS EXCEPT ![⟨rcver, id⟩] = maxTS]
                    Synchronizes the local clocks
                  ∧ clock' = [clock EXCEPT ![rcver] = Max(clock[rcver], maxTS.t)]
                  ∧ phase' = [phase EXCEPT ![⟨rcver, id⟩] = Committed]
                  ∧ delivered' = [⟨p, mid⟩ ∈ Proc × McastID ↦
                                    IF p ≠ rcver
                                    THEN delivered[⟨p, mid⟩]
                                    ELSE IF ¬delivered[⟨rcver, mid⟩]
                                         THEN CanDeliver(rcver, mid)
                                         ELSE delivered[⟨rcver, mid⟩]]
                    Update how many times p has delivered message mid
                  ∧ dCntr' = [⟨p, mid⟩ ∈ Proc × McastID ↦
                                IF p ≠ rcver
                                THEN dCntr[⟨p, mid⟩]
```

$$
\begin{array}{l}
\quad\quad\quad\quad\quad\quad\quad\quad \text{ELSE IF } \neg delivered[\langle rcver, mid\rangle] \land CanDeliver(rcver, mid) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{THEN } dCntr[\langle rcver, mid\rangle] + 1 \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{ELSE } dCntr[\langle rcver, mid\rangle]] \\
\quad\quad \text{ELSE UNCHANGED } \langle phase, globalTS, clock, delivered, dCntr\rangle
\end{array}
$$

Only to avoid deadlock checking

$$
\begin{array}{l}
Done \triangleq \\
\quad \land \forall id \in McastID : \forall p \in GroupDest[id] : delivered[\langle p, id\rangle] \\
\quad \land \text{UNCHANGED } vars
\end{array}
$$

$$
\begin{array}{l}
Next \triangleq \\
\quad \lor \exists m \in McastID : Multicast(m) \\
\quad \lor \exists snder, rcver \in Proc : \exists msg \in inTransit[\langle snder, rcver\rangle] : ReceiveMulticast(snder, rcver, msg) \\
\quad \lor \exists snder, rcver \in Proc : \exists msg \in inTransit[\langle snder, rcver\rangle] : ReceivePropose(snder, rcver, msg) \\
\quad \lor Done
\end{array}
$$

$$
\begin{array}{l}
Spec \triangleq \\
\quad \land Init \\
\quad \land \Box[Next]_{vars} \\
\quad \land \text{WF}_{vars}(\lor \exists m \in McastID : Multicast(m) \\
\quad\quad\quad\quad \lor \exists snder, rcver \in Proc : \exists msg \in inTransit[\langle snder, rcver\rangle] : \\
\quad\quad\quad\quad\quad ReceiveMulticast(snder, rcver, msg) \\
\quad\quad\quad\quad \lor \exists snder, rcver \in Proc : \exists msg \in inTransit[\langle snder, rcver\rangle] : \\
\quad\quad\quad\quad\quad ReceivePropose(snder, rcver, msg))
\end{array}
$$

- Total Order: There exists a total order $<$ on all messages that are multicast in an execution trace such that, if process p delivers message m, then for all messages $m' < m$ such that p is one of addresses of message m', p delivers m' before m.
- Total Order can be formalized as the following formula

$$
\begin{array}{l}
GlobalTotalOrdering \triangleq \\
\quad \exists \text{ ordering } \in [McastID \to 1 \,..\, M] : \\
\quad\quad \land \forall p \in Proc : \forall id1, id2 \in McastID : \\
\quad\quad\quad (\land globalTS[\langle p, id1\rangle] \neq TimestampNull \\
\quad\quad\quad\quad \land globalTS[\langle p, id2\rangle] \neq TimestampNull \\
\quad\quad\quad\quad \land ordering[id1] < ordering[id2]) \\
\quad\quad\quad\quad\quad \Rightarrow Less(globalTS[\langle p, id1\rangle], globalTS[\langle p, id2\rangle])
\end{array}
$$

- However, *APALACHE* cannot verify *GlobalTotalOrdering* because the initialization of ordering and its corresponding quantifiers.

The conjunction of *ConsistentGlobalTS* and *AsymmetricOrdering* implies Total Order

$$
\begin{array}{l}
AsymmetricOrdering \triangleq \\
\quad \forall id1, id2 \in McastID : \forall p, q \in Proc : \\
\quad\quad (\land globalTS[\langle p, id1\rangle] \neq TimestampNull \\
\quad\quad\quad \land globalTS[\langle p, id2\rangle] \neq TimestampNull \\
\quad\quad\quad \land globalTS[\langle q, id1\rangle] \neq TimestampNull \\
\quad\quad\quad \land globalTS[\langle q, id2\rangle] \neq TimestampNull
\end{array}
$$

$\land\ id1 \neq id2)$
$\quad \Rightarrow \neg(Less(globalTS[\langle p,\ id1\rangle],\ globalTS[\langle p,\ id2\rangle]) \land Less(globalTS[\langle q,\ id2\rangle],\ globalTS[\langle q,\ id1\rangle]))$

$ConsistentGlobalTS \triangleq$
$\land \forall\, id \in McastID : \forall\, p,\ q \in Proc :$ All addressees of message id must agree on its global timestamp.
$\quad (\land\ globalTS[\langle p,\ id\rangle] \neq TimestampNull$
$\quad\ \land\ globalTS[\langle q,\ id\rangle] \neq TimestampNull)$
$\quad\quad \Rightarrow globalTS[\langle p,\ id\rangle] = globalTS[\langle q,\ id\rangle]$
$\land \forall\, id1,\ id2 \in McastID : \forall\, p \in Proc :$ Every message has a unique global timestamp.
$\quad (\land\ id1 \neq id2$
$\quad\ \land\ globalTS[\langle p,\ id1\rangle] \neq TimestampNull$
$\quad\ \land\ globalTS[\langle p,\ id2\rangle] \neq TimestampNull)$
$\quad\quad \Rightarrow globalTS[\langle p,\ id1\rangle] \neq globalTS[\langle p,\ id2\rangle]$

$Validity \triangleq \forall\, p \in Proc : \forall\, id \in McastID : delivered[\langle p,\ id\rangle] \Rightarrow id \in mcastedID$

If process p is not an addressee of message id, p never issues a local timestamp for id.
Process p issues a local timestampe for message id if and only if it receive a multicast message for id.
The time in every local timestamp cannot greater than the current value of the local clock.
Never issues a local timestamp with $GroupNull$ or $TimestampNull$.
The owner of the local timestamp $localTS[\langle p,\ id\rangle]$ must be process p.
Never issues two local timestapms at one time point.
$ValidOwnedLocalTS \triangleq$

$\land (\forall\, id \in McastID : \forall\, p \in Proc \setminus GroupDest[id] : localTS[\langle p,\ id\rangle] = TimestampNull)$
$\land (\forall\, id \in McastID : \forall\, p \in GroupDest[id] :$
$\quad\ \land\ localTS[\langle p,\ id\rangle] = TimestampNull \equiv id \notin rcvdMcastID[p]$
$\quad\ \land\ localTS[\langle p,\ id\rangle].t \leq clock[p]$
$\quad\ \land\ (localTS[\langle p,\ id\rangle].g \neq GroupNull \Rightarrow localTS[\langle p,\ id\rangle].t \neq TimeNull)$
$\quad\ \land\ (localTS[\langle p,\ id\rangle] \neq TimestampNull$
$\quad\quad \Rightarrow (\land\ id \in mcastedID$
$\quad\quad\quad \land\ localTS[\langle p,\ id\rangle].g = p)))$
$\land \forall\, id1,\ id2 \in McastID : \forall\, p \in Proc :$
$\quad ((\ \land\ p \in GroupDest[id1]$
$\quad\quad \land\ p \in GroupDest[id2]$
$\quad\quad \land\ id1 \neq id2$
$\quad\quad \land\ localTS[\langle p,\ id1\rangle] \neq TimestampNull$
$\quad\quad \land\ localTS[\langle p,\ id2\rangle] \neq TimestampNull)$
$\quad\quad\quad \Rightarrow localTS[\langle p,\ id1\rangle].t \neq localTS[\langle p,\ id2\rangle].t)$

Every in-transit message in $inTransit[\langle snder,\ rcver\rangle]$ was sent by process $snder$.
The in-transit proposed message for message id must be sent after the multicast message for message id.

$ValidInTransitMsg \triangleq$

$\land \forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] : m.source = snder$

$\land \forall\, snder,\, rcver \in Proc : \forall\, m1,\, m2 \in inTransit[\langle snder,\, rcver\rangle] :$

$(\quad (\land m1.id = m2.id$

$\land m1.type = MType$

$\land m2.type = PType)$

$\Rightarrow m1.t < m2.t)$

- If process p is not an addressee of message id, no processes send a proposal for message id to process p.
- If process p is not an addressee of message id, it never sends a proposal for message id.
- If there exists a proposal for message id from process $snder$, process $snder$ has issued a local timestamp for message m. These timestamps must be the same.
- If there exists a proposal for message id, message id must be multicast before.
- The time in an issued timestamp by process $snder$ cannot greater than the current value of the clock of process $snder$.
- If there exists an in-transit proposed message for message id that is sent to process $rcver$, process $rcver$ has not issued a global timestamp for message id.
- If there exists an in-transit proposed message for message id that is sent from process $snder$, process $snder$ has issued a local timestamp for message id.
- If there exists an in-transit proposed message for message id, message id must be multicast before.
- If there exists an in-transit proposed message for message id that is sent from process $snder$, there exists no in-transit multicast message to process $snder$ such that this multicast message is for message id.

$ValidInTransitProposeTS \triangleq$

$\land (\forall\, id \in McastID : \forall\, rcver \in Proc \setminus GroupDest[id] : \forall\, snder \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] : m.i$

$\land (\forall\, id \in McastID : \forall\, snder \in Proc \setminus GroupDest[id] : \forall\, rcver \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] : m.i$

$\land (\forall\, id \in McastID : \forall\, rcver \in GroupDest[id] : \forall\, snder \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] :$

$(\land m.id = id$

$\land m.source = snder$

$\land m.type = PType)$

$\Rightarrow\ (\land m.t = localTS[\langle snder,\, id\rangle].t$

$\land id \in rcvdMcastID[snder]$

$\land id \in mcastedID))$

$\land (\forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] : m.t \leq clock[m.source] \land m.t > TimeNull)$

$\land (\forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] :$

$m.t = PType \Rightarrow (\land globalTS[\langle rcver,\, m.id\rangle] = TimestampNull$

$\land localTS[\langle m.source,\, m.id\rangle] \neq TimestampNull$

$\land m.id \in rcvdMcastID[m.source]))$

$\land (\forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] :$

$m.t = PType \Rightarrow (\land localTS[\langle m.source,\, m.id\rangle].g = m.source$

$\land localTS[\langle m.source,\, m.id\rangle].t = m.t))$

$\land (\forall\, id \in McastID : \forall\, snder,\, rcver \in GroupDest[id] : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] :$

$((m.t = PType \land m.id = id)$

$\Rightarrow\ (\forall\, m1 \in inTransit[\langle Mcaster[id],\, snder\rangle] : m1.type = MType \Rightarrow m1.id \neq id)))$

- If process p is not an addressee of message id, it never receives a proposal for message id.
- Every received proposed message for message id must be grouped correctly.
- Every received proposed message for message id from process $snder$ must propose the local timestamp that is issued by process $snder$ for message id.

$$
\begin{array}{l}
ValidRcvdProposeTS \triangleq \\
\quad \land (\forall\, id \in McastID : \forall\, rcver \in Proc \setminus GroupDest[id] : \forall\, snder \in Proc : \\
\qquad proposeTS[\langle rcver,\ id\rangle] = \{\}) \\
\quad \land (\forall\, id \in McastID : \forall\, rcver \in GroupDest[id] : \forall\, msg \in proposeTS[\langle rcver,\ id\rangle] : \\
\qquad \land\ msg.t = localTS[\langle msg.source,\ msg.id\rangle].t \\
\qquad \land\ msg.id = id \\
\qquad \land\ (\forall\, m \in inTransit[\langle msg.source,\ rcver\rangle] : m.id \neq id)) \\
\quad \land (\forall\, id \in McastID : \forall\, rcver \in GroupDest[id] : \forall\, msg \in proposeTS[\langle rcver,\ id\rangle] : \\
\qquad \land\ msg.t = localTS[\langle msg.source,\ msg.id\rangle].t \\
\qquad \land\ msg.source = localTS[\langle msg.source,\ msg.id\rangle].g \\
\qquad \land\ msg.id = id)
\end{array}
$$

Every local clock is bounded with $MaxClock$

$$BoundedClock \triangleq \forall\, p \in Proc : clock[p] \leq MaxClock$$

- The global timestamp for message m cannot be less than any proposed local timestamp for message m.
- The global timestamp for message m must equal some local timestamp for message m.
- If process p is not an addressee of message id, it never issues a global timestamp for message id.
- There exists no global timestamp with $GroupNull$ or $TimeNull$.
- The time in every global timestamp cannot greater than the current value of the clock.
- The global timestamp for message id is issued if and only if message id is committed.

$$
\begin{array}{l}
ValidGlobalTS \triangleq \\
\quad \land \forall\, id \in McastID : \forall\, rcver \in GroupDest[id] : \\
\qquad (globalTS[\langle rcver,\ id\rangle] \neq TimestampNull \\
\qquad\quad \equiv (\ \land \forall\, snder \in GroupDest[id] : \exists\, m \in proposeTS[\langle rcver,\ id\rangle] : \\
\qquad\qquad\qquad (\ \land\ m.source = snder \\
\qquad\qquad\qquad\quad \land\ \lor\ m.t < globalTS[\langle rcver,\ id\rangle].t \\
\qquad\qquad\qquad\qquad \lor\ \land\ m.t = globalTS[\langle rcver,\ id\rangle].t \\
\qquad\qquad\qquad\qquad\quad \land\ m.source \leq globalTS[\langle rcver,\ id\rangle].g)) \\
\qquad\qquad\quad \land \exists\, snder \in GroupDest[id] : \exists\, m \in proposeTS[\langle rcver,\ id\rangle] : \\
\qquad\qquad\qquad (\ \land\ globalTS[\langle rcver,\ id\rangle].t = m.t \\
\qquad\qquad\qquad\quad \land\ globalTS[\langle rcver,\ id\rangle].g = m.source)) \\
\quad \land \forall\, id \in McastID : \forall\, rcver \in Proc \setminus GroupDest[id] : globalTS[\langle rcver,\ id\rangle] = TimestampNull \\
\quad \land \forall\, id \in McastID : \forall\, rcver \in GroupDest[id] : \\
\qquad globalTS[\langle rcver,\ id\rangle].g \neq GroupNull \Rightarrow globalTS[\langle rcver,\ id\rangle].t \neq TimeNull \\
\quad \land \forall\, id \in McastID : \forall\, p \in Proc \setminus GroupDest[id] : globalTS[\langle p,\ id\rangle]\ = TimestampNull \\
\quad \land \forall\, id \in McastID : \forall\, p \in GroupDest[id] : \\
\qquad globalTS[\langle p,\ id\rangle]\quad \neq TimestampNull \Rightarrow \exists\, q \in GroupDest[id] : localTS[\langle q,\ id\rangle] = globalTS[\langle p,\ id\rangle] \\
\quad \land \forall\, id \in McastID : \forall\, rcver \in GroupDest[id] : \\
\qquad globalTS[\langle rcver,\ id\rangle].g \neq GroupNull \Rightarrow globalTS[\langle rcver,\ id\rangle].t \leq clock[rcver]
\end{array}
$$

$$\land \forall\, id \in McastID : \forall\, p \in Proc : globalTS[\langle p,\, id\rangle] \neq TimestampNull \equiv phase[\langle p,\, id\rangle] = Committed$$

Process p sets the status of message id to *Start* iff it has not issued a local timestamp for message id.
If process p commits message id, it has received at least one proposal for message id.
If process p commits message id, it has not issued any global timestamp for message id.

$$\begin{array}{l}
ValidPhase \triangleq \\
\quad \forall\, p \in Proc : \forall\, id \in McastID : \\
\quad\quad (\land\, p \notin GroupDest[id] \Rightarrow phase[\langle p,\, id\rangle] = Start \\
\quad\quad\ \land\, phase[\langle p,\, id\rangle] = Start \equiv localTS[\langle p,\, id\rangle] = TimestampNull \\
\quad\quad\ \land\, phase[\langle p,\, id\rangle] = Proposed \Rightarrow (localTS[\langle p,\, id\rangle] \neq TimestampNull \land id \in rcvdMcastID[p]) \\
\quad\quad\ \land\, phase[\langle p,\, id\rangle] = Committed \equiv (\forall\, q \in GroupDest[id] : \exists\, m \in proposeTS[\langle p,\, id\rangle] : m.source = q) \\
\quad\quad\ \land\, ((localTS[\langle p,\, id\rangle] \neq TimestampNull \land id \in rcvdMcastID[p]) \\
\quad\quad\quad\quad \Rightarrow phase[\langle p,\, id\rangle] \in \{Proposed,\, Committed\}))
\end{array}$$

Message id can be delivered to process p if and only if process p has issued a global timestamp for message id and the local timestamps of all proposed message at process p must be greater than the global timestamp of message id.

$$\begin{array}{l}
ValidDelivery \triangleq \\
\quad \forall\, p \in Proc : \forall\, id \in McastID : \\
\quad\quad delivered[\langle p,\, id\rangle] \\
\quad\quad \Rightarrow (\land\, globalTS[\langle p,\, id\rangle] \neq TimestampNull \\
\quad\quad\quad\ \land\, phase[\langle p,\, id\rangle] = Committed \\
\quad\quad\quad\ \land\, \forall\, mid \in rcvdMcastID[p] : \\
\quad\quad\quad\quad phase[\langle p,\, mid\rangle] = Proposed \Rightarrow Less(globalTS[\langle p,\, id\rangle],\, localTS[\langle p,\, mid\rangle]))
\end{array}$$

Every in-transit message has an unique timestamp.
If process *snder* has sent a proposal for message id, no in-transit message to process p is a multicast message for message id.

$$\begin{array}{l}
UniqueMsg \triangleq \\
\quad \land\, (\forall\, snder,\, rcver \in Proc : \forall\, m1,\, m2 \in inTransit[\langle snder,\, rcver\rangle] : \\
\quad\quad\quad \land\, (m1.type\ \ = m2.type \land m1.id = m2.id) \Rightarrow m1.t = m2.t \\
\quad\quad\quad \land\, (m1.type\ \ = m2.type \land m1.t\ \ = m2.t)\ \ \Rightarrow m1.id = m2.id \\
\quad\quad\quad \land\, (m1.id\ \ \ = m2.id\ \ \ \land m1.t\ \ = m2.t)\ \ \Rightarrow m1.type = m2.type) \\
\quad \land\, (\forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] : \\
\quad\quad\quad\quad m.type = PType \Rightarrow \neg(\exists\, m1 \in inTransit[\langle Mcaster[m.id],\, snder\rangle] : \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \land\, m1.id = m.id \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \land\, m1.source = Mcaster[m.id] \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \land\, m1.type = MType))
\end{array}$$

- If process p is not an addressee of message id, it never receives a multicast message for message id.
- Every multicast message for message id must be multicast by its multicaster.
- If there exists a multicast message for message id, message id must be multicast before.
- The timestamp of every proposed message from process *snder* cannot be greater the local clock of process *snder*, and must be not 0.

- If there exists an in-transit multicast message for message id to process $rcver$, process $rcver$ has not issued neither local timestamp nor global timestamp for message id.

$$
\begin{aligned}
&ValidInTransitMcast \triangleq \\
&\quad \land \forall\, snder,\, rcver \in Proc : \forall\, id \in McastID : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] : \\
&\qquad (m.type = MType \land m.id = id) \Rightarrow (snder = Mcaster[id] \land id \in mcastedID) \\
&\quad \land \forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] : \\
&\qquad m.type = MType \Rightarrow m.source = Mcaster[m.id] \\
&\quad \land \forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] : \\
&\qquad m.type = MType \Rightarrow m.id \in mcastedID \\
&\quad \land \forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[\langle snder,\, rcver\rangle] : \\
&\qquad m.t \leq clock[m.source] \land m.t > TimeNull \\
&\quad \land \forall\, mcaster,\, rcver \in Proc : \forall\, m \in inTransit[\langle mcaster,\, rcver\rangle] : \\
&\qquad m.type = MType \Rightarrow (\ \land \neg(\exists\, q \in Proc : \exists\, m1 \in inTransit[\langle rcver,\, q\rangle] : \\
&\qquad\qquad\qquad\qquad \land\, m1.source = rcver \\
&\qquad\qquad\qquad\qquad \land\, m1.id = m.id \\
&\qquad\qquad\qquad\qquad \land\, m1.type = PType) \\
&\qquad\qquad\qquad \land\, localTS[\langle rcver,\, m.id\rangle] = TimestampNull \\
&\qquad\qquad\qquad \land \forall\, p \in GroupDest[m.id] : globalTS[\langle p,\, m.id\rangle] = TimestampNull)
\end{aligned}
$$

$$
\begin{aligned}
&TypeOK \triangleq \\
&\quad \land\, clock \in [Proc \to Time \cup \{TimeNull\}] \\
&\quad \land\, localTS \in [(Proc \times McastID) \to TimestampSet] \\
&\quad \land\, globalTS \in [(Proc \times McastID) \to TimestampSet] \\
&\quad \land\, phase \in [(Proc \times McastID) \to \{Start,\, Proposed,\, Committed\}] \\
&\quad \land\, rcvdMcastID \in [Proc \to \text{SUBSET } McastID] \\
&\quad \land\, mcastedID \in \text{SUBSET } McastID \\
&\quad \land\, inTransit \ \in [(Proc \times Proc) \to \text{SUBSET } InTransitMsgSet] \\
&\quad \land\, delivered \ \in [(Proc \times McastID) \to \text{BOOLEAN }] \\
&\quad \land\, proposeTS \in [(Proc \times McastID) \to \text{SUBSET } ProposeMsgSet] \\
&\quad \land\, dCntr \in [Proc \times McastID \to \{0,\, 1\}]
\end{aligned}
$$

$$
\begin{aligned}
&Integrity \triangleq \\
&\quad \forall\, id \in McastID : \forall\, p \in Proc : \\
&\qquad \land\, delivered[\langle p,\, id\rangle] \equiv dCntr[\langle p,\, id\rangle] = 1 \\
&\qquad \land \neg delivered[\langle p,\, id\rangle] \equiv dCntr[\langle p,\, id\rangle] = 0
\end{aligned}
$$

$$
\begin{aligned}
&IndInv \triangleq \\
&\quad \land\, TypeOK \\
&\quad \land\, Integrity \\
&\quad \land\, Validity \\
&\quad \land\, ValidInTransitMsg \\
&\quad \land\, AsymmetricOrdering \\
&\quad \land\, ConsistentGlobalTS \\
&\quad \land\, ValidOwnedLocalTS
\end{aligned}
$$

$\land$ *ValidInTransitProposeTS*
$\land$ *ValidRcvdProposeTS*
$\land$ *BoundedClock*
$\land$ *ValidGlobalTS*
$\land$ *ValidDelivery*
$\land$ *ValidPhase*
$\land$ *ValidInTransitMcast*
$\land$ *UniqueMsg*

* Modification History
* Last modified *Wed Oct* 06 11:52:32 *CEST* 2021 by *tran*
* Created *Mon Oct* 04 16:37:11 *CEST* 2021 by *tran*