MODULE *skeen_int_int_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[1] *Skeen*, *D.*: (1985), Referenced in [1], unpublished communication.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
- $phase[p][m]$: 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.

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

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

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*.

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

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*;

$isYoungestMsg(snder,\ rcver,\ msg) \triangleq$
$\forall\, m \in inTransit[snder][rcver] : msg.t \leq m.t$

Receives a multicast message
$ReceiveMulticast(snder,\ rcver,\ msg) \triangleq$
$\land\ clock[rcver] < MaxClock$
$\land\ msg.type = MType$
$\land\ isYoungestMsg(snder,\ rcver,\ msg)$ *msg* must have the smallest timestamp in $inTransit[snder][rcver]$
$\land\ rcvdMcastID' = [rcvdMcastID \text{ EXCEPT } ![rcver] = rcvdMcastID[rcver] \cup \{msg.id\}]$
$\land$ UNCHANGED $\langle proposeTS,\ globalTS,\ delivered,\ mcastedID,\ dCntr\rangle$
$\land$ LET $mid \triangleq msg.id$
$time \triangleq clock[rcver] + 1$
$newTS \triangleq [t \mapsto time,\ g \mapsto rcver]$ the local timestamp for message *msg.id*
@type: [type: *Int*, *t*: *Int*, *id*: *Int*, source: *Int*];
$newMsg \triangleq [type \mapsto PType,\ id \mapsto mid,\ source \mapsto rcver,\ t \mapsto time]$ the proposal for message *msg.id*
IN $\land\ clock' = [clock \text{ EXCEPT } ![rcver] = clock[rcver] + 1]$
$\land\ localTS' = [localTS \text{ EXCEPT } ![rcver][mid] = newTS]$
$\land\ phase' = [phase \text{ EXCEPT } ![rcver][mid] = Proposed]$
Sends its proposal to every addressee of message *msg.id*
$\land$ IF $snder \neq rcver$
THEN $inTransit' = [p \in Proc \mapsto [q \in Proc \mapsto$
IF $p = rcver \land q \in GroupDest[mid]$
THEN $inTransit[p][q] \cup \{newMsg\}$
ELSE IF $p = snder \land q = rcver$
THEN $inTransit[p][q] \setminus \{msg\}$
ELSE $inTransit[p][q]]]$
ELSE $inTransit' = [p \in Proc \mapsto [q \in Proc \mapsto$
IF $p = rcver \land q = rcver$
THEN $(inTransit[p][q] \cup \{newMsg\}) \setminus \{msg\}$
ELSE IF $p = rcver \land q \in GroupDest[mid]$
THEN $inTransit[p][q] \cup \{newMsg\}$
ELSE $inTransit[p][q]]]$

Compare two timestamps based on lexicographical order
@type: $([t : Int,\ g : Int],\ [t : Int,\ g : Int]) \Rightarrow Bool$;
$Less(ts1,\ ts2) \triangleq$
$\lor\ ts1.t < ts2.t$
$\lor\ \land\ ts1.t = ts2.t$
$\land\ ts1.g < ts2.g$

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: $(Int,\ Int) \Rightarrow Bool$;

```
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' = [delivered EXCEPT ![rcver] = [mid ∈ McastID ↦
                                    IF ¬delivered[rcver][mid]
                                    THEN CanDeliver(rcver, mid)
                                    ELSE delivered[rcver][mid]]]
                    Update how many times p has delivered message mid
                  ∧ dCntr' = [dCntr EXCEPT ![rcver] = [mid ∈ McastID ↦
                                    IF ¬delivered[rcver][mid] ∧ CanDeliver(rcver, mid)
                                    THEN dCntr[rcver][mid] + 1
                                    ELSE dCntr[rcver][mid]]]
          ELSE UNCHANGED ⟨phase, globalTS, clock, delivered, dCntr⟩
```

Only to avoid deadlock checking

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

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

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

- 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

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

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

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

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

$$
\begin{array}{l}
ConsistentGlobalTS \triangleq \\
\quad \land \forall\, id \in McastID : \forall\, p,\, q \in Proc :
\end{array}
$$

All addressees of message id must agree on its global timestamp.

$$
\begin{array}{l}
\quad (\ \wedge\ globalTS[p][id] \neq TimestampNull \\
\quad\ \ \wedge\ globalTS[q][id] \neq TimestampNull) \\
\qquad \Rightarrow globalTS[p][id] = globalTS[q][id] \\
\wedge\ \forall\, id1,\ id2 \in McastID : \forall\, p \in Proc : \\
\quad (\ \wedge\ id1 \neq id2 \\
\quad\ \ \wedge\ globalTS[p][id1] \neq TimestampNull \\
\quad\ \ \wedge\ globalTS[p][id2] \neq TimestampNull) \\
\qquad \Rightarrow globalTS[p][id1] \neq globalTS[p][id2]
\end{array}
$$

Every message has a unique global timestamp.

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

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 $localTS[p][id]$ must be process p.
Never issues two local timestapms at one time point.

$$
\begin{array}{l}
ValidOwnedLocalTS \triangleq \\
\quad \wedge\ (\forall\, id \in McastID : \forall\, p \in Proc \setminus GroupDest[id] : localTS[p][id] = TimestampNull) \\
\quad \wedge\ (\forall\, id \in McastID : \forall\, p \in GroupDest[id] : \\
\qquad \wedge\ localTS[p][id] = TimestampNull \equiv id \notin rcvdMcastID[p] \\
\qquad \wedge\ localTS[p][id].t \leq clock[p] \\
\qquad \wedge\ (localTS[p][id].g \neq GroupNull \Rightarrow localTS[p][id].t \neq TimeNull) \\
\qquad \wedge\ (localTS[p][id] \neq TimestampNull \\
\qquad\quad \Rightarrow (\ \wedge\ id \in mcastedID \\
\qquad\qquad \wedge\ localTS[p][id].g = p))) \\
\quad \wedge\ \forall\, id1,\ id2 \in McastID : \forall\, p \in Proc : \\
\qquad ((\ \wedge\ p\ \ \in GroupDest[id1] \\
\qquad\quad \wedge\ p\ \ \in GroupDest[id2] \\
\qquad\quad \wedge\ id1 \neq id2 \\
\qquad\quad \wedge\ localTS[p][id1] \neq TimestampNull \\
\qquad\quad \wedge\ localTS[p][id2] \neq TimestampNull) \\
\qquad\qquad \Rightarrow localTS[p][id1].t \neq localTS[p][id2].t)
\end{array}
$$

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

$$
\begin{array}{l}
ValidInTransitMsg \triangleq \\
\quad \wedge\ \forall\, snder,\ rcver \in Proc : \forall\, m \in inTransit[snder][rcver] : m.source = snder \\
\quad \wedge\ \forall\, snder,\ rcver \in Proc : \forall\, m1,\ m2 \in inTransit[snder][rcver] : \\
\qquad (\quad (\ \wedge\ m1.id = m2.id \\
\qquad\quad\ \ \wedge\ m1.type = MType \\
\qquad\quad\ \ \wedge\ m2.type = PType)
\end{array}
$$

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

- 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.

$$
\begin{array}{l}
ValidInTransitProposeTS \triangleq \\
\quad \land (\forall\, id \in McastID : \forall\, rcver \in Proc \setminus GroupDest[id] : \forall\, snder \in Proc : \\
\qquad \forall\, m \in inTransit[snder][rcver] : m.id \neq id) \\
\quad \land (\forall\, id \in McastID : \forall\, snder \in Proc \setminus GroupDest[id] : \forall\, rcver \in Proc : \\
\qquad \forall\, m \in inTransit[snder][rcver] : m.id \neq id) \\
\quad \land (\forall\, id \in McastID : \forall\, rcver \in GroupDest[id] : \forall\, snder \in Proc : \forall\, m \in inTransit[snder][rcver] : \\
\qquad\quad (\land\, m.id = id \\
\qquad\qquad \land\, m.source = snder \\
\qquad\qquad \land\, m.type = PType) \\
\qquad\quad \Rightarrow\ (\land\, m.t = localTS[snder][id].t \\
\qquad\qquad \land\, id \quad \in rcvdMcastID[snder] \\
\qquad\qquad \land\, id \quad \in mcastedID)) \\
\quad \land (\forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[snder][rcver] : m.t \leq clock[m.source] \land m.t > TimeNull) \\
\quad \land (\forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[snder][rcver] : \\
\qquad m.t = PType \Rightarrow (\land\, globalTS[rcver][m.id] = TimestampNull \\
\qquad\qquad\qquad\qquad \land\, localTS[m.source][m.id] \neq TimestampNull \\
\qquad\qquad\qquad\qquad \land\, m.id \in rcvdMcastID[m.source])) \\
\quad \land (\forall\, snder,\, rcver \in Proc : \forall\, m \in inTransit[snder][rcver] : \\
\qquad m.t = PType \Rightarrow (\land\, localTS[m.source][m.id].g = m.source \\
\qquad\qquad\qquad\qquad \land\, localTS[m.source][m.id].t = m.t)) \\
\quad \land (\forall\, id \in McastID : \forall\, snder,\, rcver \in GroupDest[id] : \forall\, m \in inTransit[snder][rcver] : \\
\qquad ((m.t = PType \land m.id = id) \\
\qquad\quad \Rightarrow\ (\forall\, m1 \in inTransit[Mcaster[id]][snder] : m1.type = MType \Rightarrow m1.id \neq id)))
\end{array}
$$

- 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.

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

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.

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

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.

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

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.

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

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.

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

- 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.

$$
\begin{array}{l}
ValidInTransitMcast \triangleq \\
\quad \wedge\ \forall\, snder,\ rcver \in Proc : \forall\, id \in McastID : \forall\, m \in inTransit[snder][rcver] :
\end{array}
$$

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

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

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

$$
\begin{array}{l}
IndInv \triangleq \\
\quad \land\, TypeOK \\
\quad \land\, Integrity \\
\quad \land\, Validity \\
\quad \land\, ValidInTransitMsg \\
\quad \land\, AsymmetricOrdering \\
\quad \land\, ConsistentGlobalTS \\
\quad \land\, ValidOwnedLocalTS \\
\quad \land\, ValidInTransitProposeTS \\
\quad \land\, ValidRcvdProposeTS \\
\quad \land\, BoundedClock
\end{array}
$$

$\land$ *ValidGlobalTS*
$\land$ *ValidDelivery*
$\land$ *ValidPhase*
$\land$ *ValidInTransitMcast*
$\land$ *UniqueMsg*

* Modification History
* Last modified *Wed Oct* 06 11:57:13 *CEST* 2021 by *tran*
* Created *Tue Mar* 16 08:59:43 *CET* 2021 by *tran*