MODULE *Collision*

EXTENDS *Naturals*, *FiniteSets*, *Commons*

CONSTANT *NGROUPS*
CONSTANT *NPROCESSES*
CONSTANT *NMESSAGES*
CONSTANT $CONFLICTR(\_, \_)$

This algorithm works in an environment with crash-stop failures, but we do not model processes failing. The set of all processes contains all correct ones.

LOCAL $Processes \triangleq 1 \,..\, NPROCESSES$
LOCAL $Groups \triangleq 1 \,..\, NGROUPS$
LOCAL $ProcessesInGroup \triangleq [g \in Groups \mapsto Processes]$

LOCAL $AllMessages \triangleq CreateMessages(NMESSAGES, Groups, Processes)$
LOCAL $MessagesCombinations \triangleq CreatePossibleMessages(AllMessages)$

VARIABLES *K*, *PreviousMsgs*, *Delivered*, *Votes*, *MemoryBuffer*,
QuasiReliableChannel, *AtomicBroadcastBuffer*

Initialize the instance for the Generic Multicast 2. The *INITIAL_MESSAGES* is a sequence, partially ordered. The sequence elements are sets of messages, messages that commute can share a set.

$$
\begin{aligned}
&Algorithm \triangleq \text{INSTANCE } GenericMulticast2 \text{ WITH} \\
&\quad INITIAL\_MESSAGES \leftarrow [g \in Groups \mapsto \\
&\qquad PartiallyOrdered(\\
&\qquad\quad MessagesCombinations[(g\%NMESSAGES) + 1], CONFLICTR)]
\end{aligned}
$$

$Spec \triangleq Algorithm!Spec$

If a correct process p delivers messages m and n, p is in the destination of both messages, m and n do not commute. Then, p delivers either m and then n or n and then m.

$$
\begin{aligned}
&Collision \triangleq \\
&\quad \Box \forall g \in Groups : \\
&\qquad \forall p \in ProcessesInGroup[g] : \\
&\qquad\quad \forall m1, m2 \in AllMessages : m1.id \neq m2.id \\
&\qquad\qquad \land Algorithm!WasDelivered(g, p, m1) \\
&\qquad\qquad \land Algorithm!WasDelivered(g, p, m2) \\
&\qquad\qquad \land CONFLICTR(m1, m2) \\
&\qquad\qquad\quad \Rightarrow Algorithm!DeliveredInstant(g, p, m1) \neq \\
&\qquad\qquad\qquad Algorithm!DeliveredInstant(g, p, m2)
\end{aligned}
$$