MODULE *Agreement*

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

CONSTANT *NPROCESSES*
CONSTANT *NGROUPS*
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 $Processes \triangleq \{i : i \in 1 \,..\, NPROCESSES\}$
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.

$Algorithm \triangleq$ INSTANCE *GenericMulticast2* WITH
$INITIAL\_MESSAGES \leftarrow [g \in Groups \mapsto$
$PartiallyOrdered($
$MessagesCombinations[(g \% NMESSAGES) + 1], CONFLICTR)]$

Weak fairness is necessary.

$Spec \triangleq Algorithm!SpecFair$

If a correct process deliver a message m , then all correct processes in $m.d$ eventually delivers m .

We verify that all messages in *AllMessages*, for all the processes that delivered a message, eventually, all the correct members in the destination will deliver.

$$
\begin{array}{l}
Agreement \triangleq \\
\quad \forall m \in AllMessages : \\
\quad\quad \forall g\_i \in Groups : \\
\quad\quad\quad \exists p\_i \in ProcessesInGroup[g\_i] : \\
\quad\quad\quad\quad Algorithm!WasDelivered(g\_i, p\_i, m) \\
\quad\quad\quad\quad\quad \leadsto \forall g\_j \in m.d : \\
\quad\quad\quad\quad\quad\quad \exists p\_j \in ProcessesInGroup[g\_j] : \\
\quad\quad\quad\quad\quad\quad\quad p\_j \in Processes \land Algorithm!WasDelivered(g\_j, p\_j, m)
\end{array}
$$