MODULE *Collision*

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

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

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 1. The *INITIAL_MESSAGES* is a sequence, totally ordered within a group, wherein the elements are tuples with the message, state, and timestamp.

Algorithm $\triangleq$ INSTANCE *GenericMulticast*1 WITH
INITIAL_MESSAGES $\leftarrow$ [
$g \in Groups \mapsto$
TotallyOrdered(*MessagesCombinations*[$(g\%NMESSAGES) + 1$])]

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.

Collision $\triangleq$
$\Box\forall\, g \in Groups :$
$\forall\, p \in ProcessesInGroup[g] :$
$\forall\, m1,\, m2 \in AllMessages : m1.id \neq m2.id$
$\land$ *Algorithm*!*WasDelivered*(g, p, $m1$)
$\land$ *Algorithm*!*WasDelivered*(g, p, $m2$)
$\land$ *CONFLICTR*($m1$, $m2$)
$\Rightarrow$ *Algorithm*!*DeliveredInstant*(g, p, $m1$) $\neq$
Algorithm!*DeliveredInstant*(g, p, $m2$)