MODULE *P2PBroadcastSpec*

Spec for a reliable broadcast. This captures the requirement that if any processor sends a message then eventually all other processes receive the message.

EXTENDS *Naturals*, *Sequences*, *FiniteSets*

CONSTANT
Proc, Set of processes
Data

VARIABLES
sent, All messages sent by all processors
received_by All messges received. Function from message to receiving processors

---

$vars \triangleq \langle sent, received\_by \rangle$

Message is a record including the sending proc and a data.

$Message \triangleq [from : Proc, data : Data]$

$$\begin{aligned} Init \triangleq\ & \land sent = \{\} \\ & \land received\_by = [m \in Message \mapsto \{\}] \end{aligned}$$

$$\begin{aligned} TypeOK \triangleq\ & \land sent \in \text{SUBSET } Message \\ & \land received\_by \in [Message \to \text{SUBSET } Proc] \end{aligned}$$

Send message m.

$$\begin{aligned} Send(m) \triangleq\ & \land m \notin sent && \text{Message is sent only once by the original sender} \\ & \land sent' = sent \cup \{m\} \\ & \land \text{UNCHANGED } \langle received\_by \rangle \end{aligned}$$

Receive a message m at proc p

$$\begin{aligned} Recv(m, p) \triangleq\ & \land m \in sent && \text{receive only if } m \text{ was sent first} \\ & \land p \notin received\_by[m] && \text{receieve only once} \\ & \land received\_by' = [received\_by \text{ EXCEPT } ![m] = @ \cup \{p\}] \\ & \land \text{UNCHANGED } \langle sent \rangle \end{aligned}$$

$Next \triangleq \exists\, m \in Message, p \in Proc : Send(m) \lor Recv(m, p)$

$Spec \triangleq Init \land \Box[Next]_{vars}$

FairSpec is *Spec* with the addition requirement that it keeps taking steps.

$FairSpec \triangleq Spec \land \mathrm{WF}_{vars}(Next)$

* Modification History
* Last modified *Fri Apr* 07 08:46:38 *CEST* 2023 by *kulpreet*
* Created *Wed Apr* 05 09:47:12 *CEST* 2023 by *kulpreet*