MODULE *P2PBroadcast*

The specification caputers the *DAG* base reliable broadcast to disseminate shares over a peer to peer network.

The broadcast enables nodes to know which nodes have revceived the message by using implicit acknowledgements. The broadcast is not a *BFT* broadcast. We depend on the higher layers to provide that.

Does this open this broadcast to a *DDoS* attack? Yes, and our argument remains that $p2p$ network can resist *DDoS* attacks by other means.

First pass - We assume no processes failures or messages lost.

EXTENDS *Naturals*, *Sequences*

CONSTANT

Proc, Set of processes
Data,
Nbrs

VARIABLES

sent_by, Set of messages sent by processes to their neighbours
recv_by Set of messages received by processes

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

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

$Init \triangleq$
$\land sent\_by = [m \in Message \mapsto \{\}]$
$\land recv\_by = [m \in Message \mapsto \{\}]$

$TypeInvariant \triangleq$
$\land sent\_by \in [Message \to \text{SUBSET } Proc]$
$\land recv\_by \in [Message \to \text{SUBSET } Proc]$

SendTo(m, p) – send message m to neighbour p

Sending to self is required as then the message is in the recv list as well.

$SendTo(m, p) \triangleq$
$\land m.from \notin sent\_by[m]$ Don't send again - we can add decay here
$\land \langle m.from, p \rangle \in Nbrs$ Send only to neighbours
$\land sent\_by' = [sent\_by \text{ EXCEPT } ![m] = @ \cup \{p\}]$
$\land \text{UNCHANGED } \langle recv\_by \rangle$

RecvAt(m, q) – receive message m at q. This can be received from forwards

$RecvAt(m, q) \triangleq$
$\land \exists\, p \in Proc : p \in sent\_by[m]$ Some process has sent the message
$\land q \notin recv\_by[m]$ Not already received by q

$\land\ recv\_by' = [recv\_by \text{ EXCEPT } ![m] = @ \cup \{q\}]$
$\land$ UNCHANGED $\langle sent\_by \rangle$

Forward$(m,\ p,\ q)$ $-$ forward message m from p to q

Enabling condition $-$ m has been sent by some process, q has received the message, q is not the sender

Effect $-$ p forwards the message m to its nbrs

$$
\begin{aligned}
&Forward(m,\ p,\ q) \triangleq \\
&\qquad \land \exists\, r \in Proc : r \in sent\_by[m] && \text{Some process has sent the message} \\
&\qquad \land\ p \neq q && \text{Don't forward to self} \\
&\qquad \land \langle p,\ q \rangle \in Nbrs && \text{Forward only to neighbour} \\
&\qquad \land\ p \in recv\_by[m] && p \text{ has received } m \\
&\qquad \land\ sent\_by' = [sent\_by \text{ EXCEPT } ![m] = @ \cup \{q\}] \\
&\qquad \land \text{ UNCHANGED } \langle recv\_by \rangle
\end{aligned}
$$

$$
\begin{aligned}
&Next \triangleq \exists\, p \in Proc,\ q \in Proc,\ m \in Message : \\
&\qquad \lor\ SendTo(m,\ p) \\
&\qquad \lor\ RecvAt(m,\ p) \\
&\qquad \lor\ Forward(m,\ p,\ q)
\end{aligned}
$$

$$
\begin{aligned}
&Spec \triangleq \land\ Init \\
&\qquad\qquad \land\ \Box[Next]_{vars}
\end{aligned}
$$

Liveness specifies that if a message is enabled to be received at p, it is eventually received at p.

$$Liveness \triangleq \forall\, p \in Proc : \forall\, m \in Message : \mathrm{WF}_{vars}(RecvAt(m,\ p))$$

$$FairSpec \triangleq Spec \land Liveness$$

THEOREM $Spec \Rightarrow \Box\, TypeInvariant$

* Modification History
* Last modified Sun *Mar* 12 07:33:11 *CET* 2023 by *kulpreet*
* Created Sun *Mar* 05 15:04:04 *CET* 2023 by *kulpreet*