MODULE *Broadcast*

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

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

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

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

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

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.

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

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

$RecvAt(m, q) \triangleq$
$\quad \land \exists\, p \in Proc : p \in sent\_by[m]$ Some process has sent the message
$\quad \land q \notin recv\_by[m]$ Not already received by q
$\quad \land recv\_by' = [recv\_by \text{ EXCEPT } ![m] = @ \cup \{q\}]$

$\land$ 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

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

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

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

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

$$\mathit{FairSpec} \triangleq \mathit{Spec} \land \mathit{Liveness}$$

THEOREM $\mathit{Spec} \Rightarrow \Box\, \mathit{TypeInvariant}$

* Modification History
* Last modified Sat *Mar* 11 15:51:20 *CET* 2023 by *kulpreet*
* Created Sun *Mar* 05 15:04:04 *CET* 2023 by *kulpreet*