─────────────── 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
$\quad\quad$ *Proc*, $\quad$ Set of processes
$\quad\quad$ *Data*,
$\quad\quad$ *Nbrs*

VARIABLES
$\quad\quad$ *sent_by*, $\quad$ Set of messages sent by processes to their neighbours
$\quad\quad$ *recv_by* $\quad$ Set of messages received by processes

───

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

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

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

───

$SendTo(m,\ p)$ - send message m to neighbour p

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

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

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

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

1

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

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

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

$Spec \triangleq\ \wedge Init$
$$\wedge \square[Next]_{\langle sent\_by,\ recv\_by \rangle}$$

THEOREM $Spec \Rightarrow \square\, TypeInvariant$

\ * Modification History
\ * Last modified *Thu Mar* 09 22:47:00 *CET* 2023 by *kulpreet*
\ * Created Sun *Mar* 05 15:04:04 *CET* 2023 by *kulpreet*