MODULE *P2PBroadcast*

The specification caputers the *DAG* based reliable broadcast to disseminate messages 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
- $channels$, — All channels between nodes, can be indexed as $channels[from][to]$ and $channels[to][from]$ and has a queue of messages
- $sent\_by$, — Set of messages sent by processes to their neighbours
- $recv\_by$ — Set of messages received by processes

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

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

$Init \triangleq$

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

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

$\quad \land\ channels = [\langle p,\ q \rangle \in Nbrs \mapsto \langle\rangle]$ — Messages delivered in order

$TypeInvariant \triangleq$

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

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

$\quad \land\ channels \in [Nbrs \rightarrow Seq(Message)]$

$SendTo(m,\ p) -$ 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

$\quad \land\ sent\_by' = [sent\_by \text{ EXCEPT } ![m] = @ \cup \{m.from\}]$

$\quad \land\ channels' = [channels \text{ EXCEPT } ![\langle m.from,\ p \rangle] = Append(@,\ m)]$

$\land \text{UNCHANGED } \langle \mathit{recv\_by} \rangle$

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

$\mathit{RecvAt}(m, q) \triangleq$
$\land \langle m.\mathit{from}, q \rangle \in \mathit{Nbrs}$ receive only at neighbours
$\land \mathit{Len}(\mathit{channels}[\langle m.\mathit{from}, q \rangle]) > 0$
$\land m = \mathit{Head}(\mathit{channels}[\langle m.\mathit{from}, q \rangle])$
$\land \exists\, p \in \mathit{Proc} : p \in \mathit{sent\_by}[m]$ Some process has sent the message
$\land q \notin \mathit{recv\_by}[m]$ Not already received by q
$\land \mathit{recv\_by}' = [\mathit{recv\_by} \text{ EXCEPT } ![m] = @ \cup \{q\}]$
$\land \mathit{channels}' = [\mathit{channels} \text{ EXCEPT } ![\langle m.\mathit{from}, q \rangle] = \mathit{Tail}(@)]$
$\land \text{UNCHANGED } \langle \mathit{sent\_by} \rangle$

$\mathit{Lose}(m, p, q) \triangleq$
$\land \mathit{Len}(\mathit{channels}[\langle m.\mathit{from}, q \rangle]) > 0$
$\land m = \mathit{Head}(\mathit{channels}[\langle m.\mathit{from}, q \rangle])$
$\land \mathit{channels}' =$ [channels $\text{EXCEPT } ![\langle m.\mathit{from}, q \rangle] = \mathit{Tail}(@)]$
$\land \text{UNCHANGED } \langle \mathit{sent\_by}, \mathit{recv\_by} \rangle$

$\mathit{Forward}(m, p, q) -$ 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

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

$\mathit{Next} \triangleq \exists\, p \in \mathit{Proc}, q \in \mathit{Proc}, m \in \mathit{Message} :$
$\lor \mathit{SendTo}(m, p)$
$\lor \mathit{RecvAt}(m, p)$
$\lor \mathit{Lose}(m, p, q)$
$\lor \mathit{Forward}(m, p, q)$

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

$\mathit{SendLeadsToRecv} \triangleq \forall\, m \in \mathit{Message} : \forall\, p \in \mathit{Proc} : \forall\, q \in \mathit{Proc} :$
$(p \in \mathit{sent\_by}[m] \land p \neq q) \leadsto (q \in \mathit{recv\_by}[m])$

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

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

$FairSpec \triangleq Spec \land Liveness$

THEOREM $Spec \Rightarrow \Box TypeInvariant$

\* Modification History
\* Last modified *Tue Mar* 28 11:34:28 *CEST* 2023 by *kulpreet*
\* Created Sun *Mar* 05 15:04:04 *CET* 2023 by *kulpreet*