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

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

VARIABLES

$\quad channels,$ All channels between nodes, can be indexed as $channels[from][to]$ and $channels[to][from]$ and has a queue of messages
$\quad sent\_by,$ Function from message to all $Proc$ that have sent it
$\quad sent,$ Same as $P2PBroadcastSpec$
$\quad received\_by$ Same as $P2PBroadcastSpec$

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

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

$Init \triangleq$
$\quad \land sent\_by = [m \in Message \mapsto \{\}]$
$\quad \land received\_by = [m \in Message \mapsto \{\}]$
$\quad \land channels = [\langle p, q \rangle \in Nbrs \mapsto \langle \rangle]$ Messages delivered in order
$\quad \land sent = \{\}$

$TypeInvariant \triangleq$
$\quad \land sent\_by \in [Message \rightarrow \text{SUBSET } Proc]$
$\quad \land received\_by \in [Message \rightarrow \text{SUBSET } Proc]$
$\quad \land channels \in [Nbrs \rightarrow Seq(Message)]$
$\quad \land sent \in \text{SUBSET } Message$

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

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

$SendTo(m, p) \triangleq$
$\qquad \land m.from \notin sent\_by[m]$ Don't send again

```
    $\land \langle m.from, p \rangle \in Nbrs$    Send only to neighbours
    $\land sent\_by' = [sent\_by \text{ EXCEPT } ![m] = @ \cup \{m.from\}]$
    $\land sent' = sent \cup \{m\}$
    $\land channels' = [channels \text{ EXCEPT } ![\langle m.from, p \rangle] = Append(@, m)]$
    $\land \text{UNCHANGED } \langle received\_by \rangle$
```

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

```
$RecvAt(m, p, q) \triangleq$
    $\land \langle p, q \rangle \in Nbrs$    receive only at neighbours
    $\land channels[\langle p, q \rangle] \neq \langle\rangle$    receive if there is a message
    $\land m = Head(channels[\langle p, q \rangle])$    receive the message at head
    $\land \exists r \in Proc : r \in sent\_by[m]$    Some process has sent the message
    $\land q \notin received\_by[m]$    Not already received by $q$
    $\land received\_by' = [received\_by \text{ EXCEPT } ![m] = @ \cup \{q\}]$
    $\land channels' = [channels \text{ EXCEPT } ![\langle p, q \rangle] = Tail(@)]$
    $\land \text{UNCHANGED } \langle sent\_by, sent \rangle$
```

```
$Lose(m, p, q) \triangleq$
    $\land \langle m.from, q \rangle] \neq \langle\rangle$
    $\land m = Head(channels[\langle m.from, q \rangle])$
    $\land channels' =$ [channels $\text{ EXCEPT } ![\langle m.from, q \rangle] = Tail(@)]$
    $\land \text{UNCHANGED } \langle sent\_by, received\_by \rangle$
```

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

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

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

```
$Spec \triangleq \land Init$
```

$\quad\quad \land \Box[Next]_{vars}$

$SendLeadsToRecv \triangleq \forall m \in Message: \forall p \in Proc: \forall q \in Proc: (p \in sent\_by[m]) \leadsto (q \in received\_by[m] \lor q \neq m.from)$

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

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

$FairSpec \triangleq Spec \land Liveness$

THEOREM $Spec \Rightarrow \Box TypeInvariant$

$PBS \triangleq$ INSTANCE $P2PBroadcastSpec$
THEOREM $Spec \Rightarrow PBS!Spec$

\* Modification History
\* Last modified *Fri Apr* 07 09:28:40 *CEST* 2023 by *kulpreet*
\* Created Sun *Mar* 05 15:04:04 *CET* 2023 by *kulpreet*