MODULE *membership_service*

Specification for maintaining a consistent view of membership list

EXTENDS *Integers*

VARIABLE *node*, *chan*

CONSTANT *IP*, *Data*, *ADD_Request*, *ACK_Response*

$$\begin{aligned}
&\textit{TypeInvariant} \triangleq \ \land\ \textit{node} \in [\textit{type} : \{1, 2\},\ \textit{rdy} : \{0, 1\},\ \textit{ack} : \{0, 1\},\ \textit{ip} : \textit{IP}] \\
&\qquad\qquad\qquad\quad \land\ \textit{IPs} = \textit{IP} \\
&\qquad\qquad\qquad\quad \land\ \textit{chan} \in [\textit{type} : \{1, 2\},\ \textit{rdy} : \{0, 1\},\ \textit{ack} : \{0, 1\},\ \textit{val} : \{\langle\rangle,\ \langle \textit{IPs},\ \textit{IPd}\rangle,\ \langle \textit{IP},\ \textit{MembershipLis}\end{aligned}$$

chan is either of sending type or receiving type
chan carries either empty value (when it's not in use), [*IPs* and *IPs*] (when sending ADD request, source *IP*: *IPs* , destination *IP*: *IPd*), or [*IPs*, *IPd* and Membership list] (when sending regular heartbeats)

$$\begin{aligned}
&\textit{LeaderInit} \triangleq \ \lor\ \land\ \textit{node.type} = 1 \\
&\qquad\qquad\qquad\quad \land\ \textit{node.rdy} = \textit{node.ack} \quad \text{Leader begins in listening mode: Listening for ADD req} \\
&\qquad\qquad\qquad\quad \land\ \textit{node.membership} = \langle\rangle \quad \text{Leader begins with empty membership list}
\end{aligned}$$

$$\begin{aligned}
&\textit{PeerInit} \triangleq \ \lor\ \land\ \textit{node.type} = 2 \\
&\qquad\qquad\quad \land\ \textit{node.rdy} = \textit{node.ack} \quad \text{Peer begins in listening mode: Listening for } \textit{ACK} \text{ res} \\
&\qquad\qquad\quad \land\ \textit{node.membership} = \langle\rangle \quad \text{Peer begins with empty membership list}
\end{aligned}$$

$$\begin{aligned}
&\textit{PeerSendAddReq}(d) \triangleq \ \land\ \textit{PeerInit} \\
&\qquad\qquad\qquad\quad \land\ d = \textit{ADD_Request} \\
&\qquad\qquad\qquad\quad \land\ \textit{chan}' = [\textit{chan} \ \text{EXCEPT}\ !.\textit{val} = \textit{Append}(@,\ \textit{IP}),\ !.\textit{rdy} = 1 - @] \\
&\textit{LeaderReceiveAddEvent} \triangleq \ \lor\ \land\ \textit{LeaderInit} \\
&\qquad\qquad\qquad\qquad\quad \land\ \textit{node}' = [\textit{node} \ \text{EXCEPT}\ !.\textit{ack} = 1 - @] \\
&\textit{LeaderSendAckEvent}(d) \triangleq \ \land\ \textit{LeaderReceiveAddEvent} \\
&\qquad\qquad\qquad\qquad\quad \land\ \text{send } \textit{ack} \text{ to the machine that sent ADD request} \\
&\textit{PeerReceiveAckEvent} \triangleq \ \land\ \textit{PeerSendAddReq} \\
&\qquad\qquad\qquad\qquad\ \land\ \text{peer do something after receiving } \textit{ACK} \text{ response} \\
&\textit{PeerSendRegularHb} \triangleq \ \land\ \textit{PeerReceiveAckEvent} \\
&\qquad\qquad\qquad\qquad\ \land\ \text{what to send in the regular } \textit{Hb}? \\
&\textit{PeerReceiveRegularHb} \triangleq \ \land\ \textit{PeerReceiveAckEvent} \\
&\qquad\qquad\qquad\qquad\ \land\ \text{what to do with received } \textit{Hb}? \\
&\textit{LeaderNext} \triangleq \ \land\ \textit{LeaderReceiveAddEvent} \\
&\qquad\qquad\qquad\quad \land\ \textit{node}' = [\textit{node} \ \text{EXCEPT}\ !.\textit{membership} = \textit{Append}(@,\ \textit{chan.val.IPs})] \\
&\textit{LeaderSendRegularHb} \triangleq \ \land\ (\exists\, d \in \textit{Data} : \textit{LeaderSendAckEvent}(d)) \\
&\qquad\qquad\qquad\qquad\ \land\ \text{What to send in regular } \textit{HB} \text{ as a Leader} \\
&\textit{LeaderReceiveRegularHb} \triangleq \ \land\ (\exists\, d \in \textit{Data} : \textit{LeaderSendAckEvent}(d)) \\
&\qquad\qquad\qquad\qquad\ \land\ \text{What to do with received regular } \textit{HB} \text{ as a Leader}
\end{aligned}$$

How to write a temporal logic for membership service such that

1. There are two types of machines in the system - Leader and Peer
2. Leader begins with a blank membership list
3. Peer knows about the Leader's *Ip* address
4. Peer begins by sending ADD request to Leader, ADD request carries requester's *IP* address.
5. Leader listens for ADD request

6. Leader receives ADD request from Peer i
7. Leader updates its membership list with thie Peer i's *IP* Address and timestamp
8. Leader sends *ACK* with current membership list of *Ip* Addresses and timestamps
9. Leader listens for heartbeat messages from all Peers
10. All Peers send heartbeat messages to Leader
11. All Peers send heartbeat messages to two *Ip* addresses higher than itself, mod calculations is considered
12. All Peers receive heartbeat messages from two peers less than itself, mod calculations is considered

* Modification History
* Last modified *Wed Apr* 11 13:33:58 *PDT* 2018 by *aksin*
* Created *Tue Mar* 27 15:24:27 *PDT* 2018 by *aksin*