

EXTENDS *Naturals, FiniteSets, Sequences, TLC*

Is leader *ALIVE* or *CRASHED*

VARIABLE *leaderState*

Helper variable tracking the number of times the leader node has failed

VARIABLE *nodedownIndex*

A collection of 'nodedown' messages the leader has sent.

In *Erlang*, when a monitored node crashes, the 'nodedown' message is sent to the follower.

The value of a message is not a part of the implementation and represents the number of times the node has failed.

VARIABLE *nodedownMessages*

$nodedownInfo \triangleq \langle nodedownMessages, nodedownIndex \rangle$

Heartbeat messages the leader has sent

The value of a message is not a part of the implementation and represents the number of times a heartbeat message was sent.

VARIABLE *heartbeatMessages*

Helper variable tracking the number of sent heartbeat messages

VARIABLE *heartbeatIndex*

$heartbeatInfo \triangleq \langle heartbeatMessages, heartbeatIndex \rangle$

Whether the follower timed out

VARIABLE *isTimeout*

$vars \triangleq \langle leaderState, nodedownInfo, heartbeatInfo, isTimeout \rangle$

The leader crashes and doesn't recover

$CrashLeader \triangleq$

$\wedge leaderState = \text{"ALIVE"}$
 $\wedge leaderState' = \text{"CRASHED"}$

Erlang sends the *NODEDOWN* message if the leader was monitored

$\wedge nodedownMessages' = Append(nodedownMessages, nodedownIndex)$
 $\wedge nodedownIndex' = nodedownIndex + 1$
 $\wedge UNCHANGED \langle heartbeatInfo, isTimeout \rangle$

Helper function to remove a message from a sequence of messages

$RemoveMessage(i, seq) \triangleq$

$[j \in 1 \dots Len(seq) - 1 \mapsto \text{IF } j < i \text{ THEN } seq[j] \text{ ELSE } seq[j + 1]]$

The network drops a heartbeat message

$DropHeartbeat \triangleq$

$$\begin{aligned}
& \wedge \text{Len}(\text{heartbeatMessages}) > 0 \\
& \wedge \exists i \in 1 \dots \text{Len}(\text{heartbeatMessages}) : \\
& \quad \text{heartbeatMessages}' = \text{RemoveMessage}(i, \text{heartbeatMessages}) \\
& \wedge \text{UNCHANGED } \langle \text{leaderState}, \text{heartbeatIndex}, \text{nodedownInfo}, \text{isTimeout} \rangle
\end{aligned}$$

The follower receives a heartbeat message from the leader.

$$\begin{aligned}
\text{ReceiveHeartbeat} & \triangleq \\
& \wedge \text{Len}(\text{heartbeatMessages}) > 0 \\
& \wedge \exists i \in 1 \dots \text{Len}(\text{heartbeatMessages}) : \\
& \quad \text{heartbeatMessages}' = \text{RemoveMessage}(i, \text{heartbeatMessages}) \\
& \wedge \text{UNCHANGED } \langle \text{leaderState}, \text{heartbeatIndex}, \text{nodedownInfo}, \text{isTimeout} \rangle
\end{aligned}$$

The leader sends a heartbeat message to the follower.

$$\begin{aligned}
\text{SendHeartbeat} & \triangleq \\
& \wedge \text{leaderState} = \text{"ALIVE"} \\
& \wedge \text{heartbeatMessages}' = \text{Append}(\text{heartbeatMessages}, \text{heartbeatIndex}) \\
& \wedge \text{heartbeatIndex}' = \text{heartbeatIndex} + 1 \\
& \wedge \text{UNCHANGED } \langle \text{leaderState}, \text{nodedownInfo}, \text{isTimeout} \rangle
\end{aligned}$$

The network drops a nodedown message.

$$\begin{aligned}
\text{DropNodedown} & \triangleq \\
& \wedge \text{Len}(\text{nodedownMessages}) > 0 \\
& \wedge \exists i \in 1 \dots \text{Len}(\text{nodedownMessages}) : \\
& \quad \text{nodedownMessages}' = \text{RemoveMessage}(i, \text{nodedownMessages}) \\
& \wedge \text{UNCHANGED } \langle \text{leaderState}, \text{nodedownIndex}, \text{heartbeatInfo}, \text{isTimeout} \rangle
\end{aligned}$$

The follower receives a nodedown message from the leader which causes it time out immediately.

$$\begin{aligned}
\text{ReceiveNodedown} & \triangleq \\
& \wedge \text{Len}(\text{nodedownMessages}) > 0 \\
& \wedge \exists i \in 1 \dots \text{Len}(\text{nodedownMessages}) : \\
& \quad \text{nodedownMessages}' = \text{RemoveMessage}(i, \text{nodedownMessages}) \\
& \wedge \text{isTimeout}' = \text{TRUE} \\
& \wedge \text{UNCHANGED } \langle \text{leaderState}, \text{nodedownIndex}, \text{heartbeatInfo} \rangle
\end{aligned}$$

$$\begin{aligned}
\text{Timeout} & \triangleq \\
& \wedge \text{isTimeout} = \text{FALSE} \\
& \wedge \text{isTimeout}' = \text{TRUE} \\
& \wedge \text{UNCHANGED } \langle \text{leaderState}, \text{heartbeatInfo}, \text{nodedownInfo} \rangle
\end{aligned}$$

Initial state of model

$$\begin{aligned}
\text{Init} & \triangleq \wedge \text{leaderState} = \text{"ALIVE"} \\
& \wedge \text{nodedownIndex} = 0 \\
& \wedge \text{nodedownMessages} = \langle \rangle \\
& \wedge \text{heartbeatIndex} = 0 \\
& \wedge \text{heartbeatMessages} = \langle \rangle \\
& \wedge \text{isTimeout} = \text{FALSE}
\end{aligned}$$

Next state function

$$\begin{aligned} Next &\triangleq \vee DropHeartbeat \\ &\vee SendHeartbeat \\ &\vee ReceiveHeartbeat \\ &\vee DropNodedown \\ &\vee ReceiveNodedown \\ &\vee CrashLeader \\ &\vee Timeout \end{aligned}$$

$$Spec \triangleq Init \wedge \Box[Next]_{vars} \wedge WF_{vars}(Next)$$

Invariant that helps make sure we haven't stepped out of bounds

$$\begin{aligned} TypeOK &\triangleq \wedge leaderState \in \{\text{"ALIVE"}, \text{"CRASHED"}\} \\ &\wedge nodedownIndex \in Nat \\ &\wedge nodedownMessages \in Seq(Nat) \\ &\wedge heartbeatMessages \in Seq(Nat) \\ &\wedge heartbeatIndex \in Nat \\ &\wedge isTimeout \in \text{BOOLEAN} \end{aligned}$$

Properties of the system

$$LeaderFailureDetected \triangleq leaderState = \text{"CRASHED"} \leadsto isTimeout = \text{TRUE}$$

$$\text{THEOREM } Correctness \triangleq Spec \Rightarrow \Box LeaderFailureDetected$$