
MODULE *htlc*

Specifications for the *HTLC* sending and forwarding. The protocol is composed of actions like initiate, update, expire. These actions specify how the state of each node and the balance on each channel is allowed to change in response to handling *HTLC* messages

EXTENDS *Integers*,
TLC

CONSTANTS *Node*, *Channel*, *InitialBalance*

Channels are unidirectional in the spec. This helps us track states and balances for the purposes of the specifications. Channel balances are tracked for sender. *htlc* balances are tracked for receiver.

VARIABLES *htcl_states*,
channel_balances,
htlc_balances

vars \triangleq $\langle \textit{htcl_states}, \textit{channel_balances}, \textit{htlc_balances} \rangle$

update_states \triangleq { "ready",
"pending",
"in_latest_commit_tx",
"prev_commit_tx_revoked" }

Initialise channels and *htlc* with a balance and ready state

Init \triangleq
 $\wedge \textit{channel_balances} = [\langle m, n \rangle \in \textit{Channel} \mapsto \text{CHOOSE } b \in \textit{InitialBalance} : \text{TRUE}]$
 $\wedge \textit{htlc_balances} = [\langle m, n \rangle \in \textit{Channel} \mapsto 0]$
 $\wedge \textit{htcl_states} = [\langle m, n \rangle \in \textit{Channel} \mapsto \text{"ready"}]$

TypeInvariant \triangleq
channels are between nodes
 $\wedge \textit{Channel} \in \textit{Node} \times \textit{Node}$
channel balance on the sender side. Balance on $\langle m, n \rangle$ notes outstanding *htlc* balance for *m*.
 $\wedge \textit{channel_balances} \in [\textit{Node} \times \textit{Node} \rightarrow \textit{InitialBalance}]$
outstanding *htlc* balance on receiver side. Balance on $\langle m, n \rangle$ notes outstanding *htlc* balance for *n*.
 $\wedge \textit{htlc_balances} \in [\textit{Node} \times \textit{Node} \rightarrow \textit{InitialBalance}]$
channels *htlc* state
 $\wedge \textit{htcl_states} \in [\textit{Node} \times \textit{Node} \rightarrow \textit{update_states}]$

When invoked on channel $\langle a, b \rangle$. The commit transaction of *b* is affected. We simply track the outstanding *htlc* balance and don't model the entire commit transaction.

update_add_htlc(*m*, *n*, *amount*) \triangleq
Commit tx state can be in any of these states

$$\begin{aligned}
& \wedge \textit{htcl_states}[\langle m, n \rangle] \in \{ \text{“ready”, “in_latest_commit_tx”} \} \\
& \quad \text{Update only if amount is more than zero} \\
& \wedge \textit{amount} > 0 \\
& \quad \text{Update only if there is sufficient balance} \\
& \wedge \textit{channel_balances}[\langle m, n \rangle] - \textit{amount} \geq 0 \\
& \quad \text{Change htlc balance in the commit transaction} \\
& \wedge \textit{htlc_balances}' = [\textit{htlc_balances} \text{ EXCEPT } ![\langle m, n \rangle] = @ + \textit{amount}] \\
& \quad \text{Change channel balance in the commit transaction for sender} \\
& \wedge \textit{channel_balances}' = [\textit{channel_balances} \text{ EXCEPT } ![\langle m, n \rangle] = @ - \textit{amount}] \\
& \quad \text{Keep receiving updates until sender has exhausted channel sender's balance} \\
& \wedge \textit{htcl_states}' = [\textit{htcl_states} \text{ EXCEPT } ![\langle m, n \rangle] = \text{“in_latest_commit_tx”}]
\end{aligned}$$

$\textit{Next} \triangleq$
 $\vee \exists \langle m, n \rangle \in \textit{Channel}, a \in \textit{InitialBalance} :$
 $\quad \wedge \textit{update_add_htlc}(m, n, a)$

$\textit{Spec} \triangleq$
 $\wedge \textit{Init}$
 $\wedge \Box [\textit{Next}]_{\langle \textit{vars} \rangle}$
