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*, *ChannelId*, *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 *htlc_states*,
channel_balances,
htlc_balances

$vars \triangleq \langle htlc\_states,\ channel\_balances,\ htlc\_balances \rangle$

$update\_states \triangleq \{$"ready",
"pending",
"in_latest_commit_tx",
"prev_commit_tx_revoked"$\}$

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

$Init \triangleq$
$\land\ channel\_balances = [c \in Channel \times ChannelId \mapsto \text{CHOOSE } b \in InitialBalance : \text{TRUE}]$
$\land\ htlc\_balances = [c \in Channel \times ChannelId \mapsto 0]$
$\land\ htlc\_states = [c \in Channel \times ChannelId \mapsto$ "ready"$]$

$TypeInvariant \triangleq$
channel balance on the sender side. Balance on c notes outstanding *htlc* balance for m.
$\land\ channel\_balances \in [Channel \times ChannelId \to InitialBalance]$
outstanding *htlc* balance on receiver side. Balance on c notes outstanding *htlc* balance for n
$\land\ htlc\_balances \in [Channel \times ChannelId \to InitialBalance]$
channels *htlc* state
$\land\ htlc\_states \in [Channel \times ChannelId \to update\_states]$

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

$update\_add\_htlc(c,\ amount) \triangleq$
Commit tx state can be in any of these states
$\land\ htlc\_states[c] \in \{$"ready", "in_latest_commit_tx"$\}$
Update only if amount is more than zero

$\land\ amount > 0$

Update only if there is sufficient balance

$\land\ channel\_balances[c] - amount \geq 0$

Change *htlc* balance in the commit transaction

$\land\ htlc\_balances' = [htlc\_balances \text{ EXCEPT } ![c] = @ + amount]$

Change channel balance in the commit transaction for sender

$\land\ channel\_balances' = [channel\_balances \text{ EXCEPT } ![c] = @ - amount]$

Keep receiving updates until sender has exhausted channel sender's balance

$\land\ htlc\_states' = [htlc\_states \text{ EXCEPT } ![c] = \text{“in\_latest\_commit\_tx”}]$

Commit all the updates received so far for a channel. Moves the channel *htlc* to the next state − *prev_commit_tx_revoked*.

$commitment\_signed(c) \triangleq$

$\land\ htlc\_states[c] \in \{\text{“in\_latest\_commit\_tx”}\}$

$\land\ htlc\_states' = [htlc\_states \text{ EXCEPT } ![c] = \text{“prev\_commit\_tx\_revoked”}]$

$\land\ \text{UNCHANGED } \langle channel\_balances,\ htlc\_balances\rangle$

In a channel $\langle m, n\rangle$ once n received *commitment_signed* from m and moved the *htlc* to *prev_commit_tx_revoked*. This action updates the state at m.

$revoke\_and\_ack(c) \triangleq$

Update the state at n, the receiver end of the unidirectional channel

$\land\ htlc\_states[\langle c[1],\ c[0],\ c[2]\rangle] \in \{\text{“prev\_commit\_tx\_revoked”}\}$

$\land\ \text{UNCHANGED } \langle channel\_balances,\ htlc\_balances\rangle$

$Next \triangleq$

$\lor\ \exists\, c \in Channel \times ChannelId :$

$\lor\ \exists\, a \in InitialBalance : update\_add\_htlc(c,\ a)$

$\lor\ commitment\_signed(c)$

$Spec \triangleq$

$\land\ Init$

$\land\ \Box[Next]_{\langle vars\rangle}$