MODULE *HopProtocol*

EXTENDS *Integers*, *Naturals*, *TLC*, *Sequences*, *FiniteSets*

VARIABLES
l1Chain,
l2Chain,
chains,
pendingTransfers,
commitThreshold,
bondedWithdrawals,
roots

$$SumSeq(S) \triangleq$$
LET $seq \triangleq S$
$Sum[i \in 1 \,..\, Len(seq)] \triangleq$ IF $i = 1$ THEN $seq[i]$ ELSE $seq[i] + Sum[i-1]$
IN IF $seq = \langle\rangle$ THEN 0 ELSE $Sum[Len(seq)]$

RECURSIVE $SeqFromSet(\_)$
$SeqFromSet(S) \triangleq$
IF $S = \{\}$ THEN $\langle\rangle$
ELSE LET $x \triangleq$ CHOOSE $x \in S$: TRUE
IN $\langle x\rangle \circ SeqFromSet(S \setminus \{x\})$

$Range(s) \triangleq \{s[x] : x \in$ DOMAIN $s\}$

$Hash(v) \triangleq$ CHOOSE $n \in 1 \,..\, 5$: TRUE

ASSUME $(Hash(\langle 1\rangle) = Hash(\langle 1\rangle))$
ASSUME $(Hash(\langle\{1, 2, 3\}\rangle) = Hash(\langle\{2, 1, 3, 1\}\rangle))$

$Init \triangleq$
$\land\ l1Chain = 1$
$\land\ l2Chain = 2 \,..\, 3$
$\land\ chains = 1 \,..\, 3$
$\land\ pendingTransfers = [c \in l2Chain \mapsto \langle\rangle]$
$\land\ commitThreshold = 1$
$\land\ roots = [c \in chains \mapsto \langle\rangle]$
$\land\ bondedWithdrawals = [c \in chains \mapsto \{\}]$

$SendTransfer(c) \triangleq$
$\land\ c \neq 1$
$\land\ Len(pendingTransfers[c]) < 5$
$\land\ pendingTransfers' = [pendingTransfers$ EXCEPT $![c] = pendingTransfers[c] \circ \langle[$
$target \mapsto chains,\ amount \mapsto 1,\ id \mapsto Hash(1 \,..\, 2)$
$]\rangle]$
$\land\ Print(\langle$"send", $Len(pendingTransfers[c])\rangle,$ TRUE$)$

$\land$ UNCHANGED $\langle l1Chain,\ l2Chain,\ chains,\ roots,\ commitThreshold,\ bondedWithdrawals\rangle$

$CommitTransfers(c) \triangleq$
$\land\ c \neq 1$
$\land\ Len(roots[c]) < 20$
$\land\ SumSeq([x \in \text{DOMAIN } pendingTransfers[c] \mapsto pendingTransfers[c][x].amount]) > commitThreshold$
$\land\ Print(\langle \text{“commit”} \rangle, \text{TRUE})$
$\land\ pendingTransfers' = [pendingTransfers \text{ EXCEPT } ![c] = \langle\rangle]$
$\land\ roots' = [k \in chains \mapsto roots[k] \circ \langle Hash(pendingTransfers[c])\rangle]$
$\land$ UNCHANGED $\langle l1Chain,\ l2Chain,\ chains,\ commitThreshold,\ bondedWithdrawals\rangle$

$BondWithdrawal(dest) \triangleq$
$\land \exists\, source \in l2Chain :$
$\quad\land\ Len(pendingTransfers[source]) > 0$
$\quad\land \exists\, x \in \text{DOMAIN } pendingTransfers[source] :$
$\quad\quad\land\ pendingTransfers[source][x].id \notin bondedWithdrawals[dest]$
$\quad\quad\land\ bondedWithdrawals' = [bondedWithdrawals \text{ EXCEPT } ![dest] = bondedWithdrawals[dest] \cup \{pending$
$\quad\quad\land\ Print(\langle \text{“bondWithdrawal”}, bondedWithdrawals'\rangle, \text{TRUE})$
$\quad\land$ UNCHANGED $\langle l1Chain,\ l2Chain,\ chains,\ commitThreshold,\ pendingTransfers,\ roots\rangle$

$Next \triangleq$
$\land \exists\, c \in chains :$
$\quad\land \lor SendTransfer(c)$
$\quad\quad\lor CommitTransfers(c)$
$\quad\quad\lor BondWithdrawal(c)$

$AllHaveTransferRoots \triangleq \land \forall\, c \in chains :$
$\land\ Len(roots[c]) > 0$
$\land \forall\, k \in chains : Range(roots[c]) \cap Range(roots[k]) \neq \{\}$

$AllHaveBondedWithdrawals \triangleq \land \forall\, c \in chains :$
$\land\ Len(SeqFromSet(bondedWithdrawals[c])) > 0$

$EventuallyAllHaveTransferRoots \triangleq \Diamond\Box AllHaveTransferRoots$
$EventuallyAllHaveBondedWithdrawals \triangleq \Diamond\Box AllHaveBondedWithdrawals$

$vars \triangleq \langle l1Chain,\ l2Chain,\ chains,\ pendingTransfers,\ commitThreshold,\ roots,\ bondedWithdrawals\rangle$

$Spec \triangleq Init \land \Box[Next]_{vars} \land \text{WF}_{vars}(Next)$
$Live \triangleq EventuallyAllHaveTransferRoots \land EventuallyAllHaveBondedWithdrawals$