–

```
──────────────────────── MODULE PendingKeys ────────────────────────
EXTENDS TLC, Naturals

CONSTANTS Key, Slave, Client, nil

VARIABLES info, pending, pc,
    slave_map,
    client_slave, client_state, client_keys,
    num_action

aux_vars ≜ ⟨client_slave, num_action⟩

vars ≜ ⟨
    info, pending, pc,
    slave_map,
    client_state, client_keys,
    aux_vars
⟩


max_action ≜ 7

Seq ≜ 0 . . 30

NullSlave ≜ Slave ∪ {nil}

SlaveState ≜ [
    running : SUBSET Key,
    latest_seq : Seq,
    wait_list : SUBSET Client
]

Channel ≜ [data : SUBSET Key, status : {"Empty", "Ready", "Consumed"}]

ClientState ≜ [
    chan : Channel,
    consumed_seq : Seq
]


init_slave_state ≜ [
    running ↦ {},
    latest_seq ↦ 0,
    wait_list ↦ {}
]

init_client_state ≜ [
    chan ↦ [data ↦ {}, status ↦ "Consumed"],
    $consumed\_seq \mapsto 0$
$]$

$TypeOK \triangleq$
    $\land \quad info \in [Key \rightarrow NullSlave]$
    $\land \quad pending \subseteq Key$
    $\land \quad pc \in [Client \rightarrow \{\text{“Init”}, \text{“GetRunningKeys”}, \text{“WaitOnChan”}\}]$
    $\land \quad client\_slave \in [Client \rightarrow Slave]$

    $\land \quad slave\_map \in [Slave \rightarrow SlaveState]$
    $\land \quad client\_state \in [Client \rightarrow ClientState]$
    $\land \quad client\_keys \in [Client \rightarrow \textsc{subset}\ Key]$
    $\land \quad num\_action \in 0 \,..\, max\_action$


$Init \triangleq$
    $\land\ info = [k \in Key \mapsto nil]$
    $\land\ pending = \{\}$
    $\land\ pc = [s \in Client \mapsto \text{“Init”}]$
    $\land\ client\_slave \in [Client \rightarrow Slave]$

    $\land\ slave\_map = [s \in Slave \mapsto init\_slave\_state]$
    $\land\ client\_state = [c \in Client \mapsto init\_client\_state]$
    $\land\ client\_keys = [c \in Client \mapsto \{\}]$
    $\land\ num\_action = 0$


$allowAction \triangleq$
    $\land\ num\_action < max\_action$
    $\land\ num\_action' = num\_action + 1$


$pushToClient(client\_set, old\_state) \triangleq$
   $\textsc{let}$
        $get\_slave(c) \triangleq slave\_map'[client\_slave[c]]$

        $curr\_seq(c) \triangleq get\_slave(c).latest\_seq$

        $can\_push(c) \triangleq$
            $\land\ c \in client\_set$
            $\land\ old\_state[c].chan.status = \text{“Empty”}$
            $\land\ old\_state[c].consumed\_seq < curr\_seq(c)$

        $new\_chan(c) \triangleq$
            $[data \mapsto get\_slave(c).running, status \mapsto \text{“Ready”}]$

        $new\_state(c) \triangleq$
            $[chan \mapsto new\_chan(c), consumed\_seq \mapsto curr\_seq(c)]$
        $new\_state\_or\_unchanged(c) \triangleq$
            IF $can\_push(c)$
                THEN $new\_state(c)$
                ELSE $old\_state[c]$ UNCHANGED
    IN
        $[c \in Client \mapsto new\_state\_or\_unchanged(c)]$

$AddKey(k) \triangleq$
    LET
        $added \triangleq$
            IF $k \in pending$ OK
                THEN $\{\}$ OK
                ELSE $\{k\}$ OK

        $add\_one(old) \triangleq$
            IF $k \in pending$
                THEN $old$
                ELSE $old + 1$

        $do\_add\_key(s) \triangleq$
            $\land\ info[k] = nil$ OK
            $\land\ allowAction$ OK
            $\land\ info' = [info$ EXCEPT $![k] = s]$        OK
            $\land\ slave\_map' = [slave\_map$ EXCEPT    OK
                    $![s].latest\_seq = add\_one(@),$    OK
                    $![s].running = @ \cup added]$
            $\land\ client\_state' = pushToClient(slave\_map[s].wait\_list,\ client\_state)$
            $\land$ UNCHANGED $pending$
            $\land$ UNCHANGED $pc$
            $\land$ UNCHANGED $client\_slave$
            $\land$ UNCHANGED $client\_keys$
    IN
        $\exists\, s \in Slave : do\_add\_key(s)$

$RemoveKey(k) \triangleq$
    LET
        $do\_remove\_key(s) \triangleq$
            $\land\ info[k] \neq nil$
            $\land\ info[k] = s$
            $\land\ allowAction$
            $\land\ info' = [info$ EXCEPT $![k] = nil]$

            $\land\ slave\_map' = [slave\_map$ EXCEPT
                    $![s].running = @ \setminus \{k\},$
                    $![s].latest\_seq = @ + 1]$
            $\land\ client\_state' = pushToClient(slave\_map[s].wait\_list,\ client\_state)$

            $\land$ UNCHANGED $pending$
            $\land$ UNCHANGED $pc$
            $\land$ UNCHANGED $client\_slave$
            $\land$ UNCHANGED $client\_keys$
   IN
      $\exists\, s \in Slave : do\_remove\_key(s)$

$addPendingKeyUpdateSlaveMap(k) \triangleq$
   LET
      $s \triangleq info[k]$
   IN
      $\land\ slave\_map' = [slave\_map$ EXCEPT
            $![s].running = @ \setminus \{k\},$
            $![s].latest\_seq = @ + 1$
         $]$
      $\land\ client\_state' = pushToClient(slave\_map[s].wait\_list,\ client\_state)$

$AddPendingKey(k) \triangleq$
   $\land\ k \notin pending$
   $\land\ allowAction$
   $\land\ pending' = pending \cup \{k\}$
   $\land$ IF $info[k] \neq nil$
      THEN $addPendingKeyUpdateSlaveMap(k)$
      ELSE
         $\land$ UNCHANGED $slave\_map$
         $\land$ UNCHANGED $client\_state$
   $\land$ UNCHANGED $info$
   $\land$ UNCHANGED $pc$
   $\land$ UNCHANGED $client\_slave$
   $\land$ UNCHANGED $client\_keys$

$removePendingKeyUpdateSlaveMap(k) \triangleq$
   LET
      $s \triangleq info[k]$
   IN
      $\land\ slave\_map' = [slave\_map$ EXCEPT
            $![s].running = @ \cup \{k\},$
            $![s].latest\_seq = @ + 1$
         $]$
      $\land\ client\_state' = pushToClient(slave\_map[s].wait\_list,\ client\_state)$

$RemovePendingKey(k) \triangleq$
```

```
    $\land k \in pending$
    $\land allowAction$
    $\land pending' = pending \setminus \{k\}$

    $\land$ IF $info[k] \neq nil$
        THEN $removePendingKeyUpdateSlaveMap(k)$
        ELSE
            $\land$ UNCHANGED $slave\_map$
            $\land$ UNCHANGED $client\_state$

    $\land$ UNCHANGED $info$
    $\land$ UNCHANGED $pc$
    $\land$ UNCHANGED $client\_slave$
    $\land$ UNCHANGED $client\_keys$

$InitClient(c) \triangleq$
    LET
        $s \triangleq client\_slave[c]$
    IN
        $\land pc[c] =$ "Init"
        $\land pc' = [pc$ EXCEPT $![c] =$ "GetRunningKeys"$]$
        $\land slave\_map' = [slave\_map$ EXCEPT $![s].wait\_list = @ \cup \{c\}]$
        $\land$ UNCHANGED $client\_state$
        $\land$ UNCHANGED $client\_keys$
        $\land$ UNCHANGED $info$
        $\land$ UNCHANGED $pending$
        $\land$ UNCHANGED $aux\_vars$

$init\_channel \triangleq [data \mapsto \{\}, status \mapsto$ "Empty"$]$

$GetRunningKeys(c) \triangleq$
    LET
         new channel and assign to $client\_state$
        $updated\_chan \triangleq [client\_state$ EXCEPT $![c].chan = init\_channel]$
    IN
        $\land pc[c] =$ "GetRunningKeys"
        $\land pc' = [pc$ EXCEPT $![c] =$ "WaitOnChan"$]$

        $\land$ UNCHANGED $slave\_map$
        $\land client\_state' = pushToClient(\{c\}, updated\_chan)$

        $\land$ UNCHANGED $client\_keys$
        $\land$ UNCHANGED $info$
        $\land$ UNCHANGED $pending$
        $\land$ UNCHANGED $aux\_vars$
```

```
$ConsumeChan(c) \triangleq$
    $\land\ pc[c] = \text{"WaitOnChan"}$
    $\land\ client\_state[c].chan.status = \text{"Ready"}$
    $\land\ pc' = [pc \text{ EXCEPT } ![c] = \text{"GetRunningKeys"}]$
    $\land\ client\_state' = [client\_state \text{ EXCEPT } ![c].chan.status = \text{"Consumed"}]$
    $\land\ client\_keys' = [client\_keys \text{ EXCEPT } ![c] = client\_state[c].chan.data]$
    $\land\ \text{UNCHANGED } slave\_map$
    $\land\ \text{UNCHANGED } info$
    $\land\ \text{UNCHANGED } pending$
    $\land\ \text{UNCHANGED } aux\_vars$

$clientWaitOnChan(c) \triangleq$
    $\land\ pc[c] = \text{"WaitOnChan"}$
    $\land\ client\_state[c].chan.status = \text{"Empty"}$

$TerminateCond \triangleq$
    $\land\ \forall c \in Client : clientWaitOnChan(c)$
    $\land\ num\_action = max\_action$

$Terminated \triangleq$
    $\land\ TerminateCond$
    $\land\ \text{UNCHANGED } vars$

$Next \triangleq$
    $\lor\ \exists k \in Key :$
        $\lor\ AddKey(k)$
        $\lor\ RemoveKey(k)$
        $\lor\ AddPendingKey(k)$
        $\lor\ RemovePendingKey(k)$
    $\lor\ \exists c \in Client :$
        $\lor\ InitClient(c)$
        $\lor\ GetRunningKeys(c)$
        $\lor\ ConsumeChan(c)$
    $\lor\ Terminated$

$Spec \triangleq Init \land \Box[Next]_{vars}$

$FairSpec \triangleq Spec \land \mathrm{WF}_{vars}(Next)$

$AlwaysTerminate \triangleq \Diamond TerminateCond$

$running\_keys(s) \triangleq \{k \in Key : info[k] = s\} \setminus pending$

$ClientKeysMatchSharedState \triangleq$
    $\forall c \in Client :$
```

```
        $clientWaitOnChan(c) \Rightarrow$
            $\land\ client\_keys[c] = running\_keys(client\_slave[c])$

$SlaveMapRunningMatchSharedState \triangleq$
    $\forall\, s \in Slave :$
        $slave\_map[s].running = running\_keys(s)$

$channelAction(c) \triangleq$
    $\lor\ \land\ client\_state[c].chan.status = \text{“Consumed”}$
       $\land\ client\_state'[c].chan.status = \text{“Empty”}$
    $\lor\ \land\ client\_state[c].chan.status = \text{“Consumed”}$
       $\land\ client\_state'[c].chan.status = \text{“Ready”}$
    $\lor\ \land\ client\_state[c].chan.status = \text{“Empty”}$
       $\land\ client\_state'[c].chan.status = \text{“Ready”}$
    $\lor\ \land\ client\_state[c].chan.status = \text{“Ready”}$
       $\land\ client\_state'[c].chan.status = \text{“Consumed”}$
    $\lor\ client\_state'[c].chan = client\_state[c].chan$

$allChannelAction \triangleq$
    $\forall\, c \in Client : channelAction(c)$

$ChannelSpec \triangleq$
    $\Box[allChannelAction]_{client\_state}$

$ReadyAlwaysConsumed \triangleq$
    $\forall\, c \in Client :$
        $client\_state[c].chan.status = \text{“Ready”}$
            $\leadsto client\_state[c].chan.status = \text{“Consumed”}$
```