MODULE *StateSync*

EXTENDS *TLC*, *Integers*, *Sequences*

CONSTANTS *Key*, *Client*, *nil*

VARIABLES *server_state*, *wait_list*, *push_back_list*,
client_keys, *client_states*,
next_val, *server_pc*, *client_pc*, *locked*,
channels, *client_channel*, *client_queue*,
consume_channel, *outer_states*

$vars \triangleq \langle server\_state, wait\_list, push\_back\_list,$
$client\_keys, client\_states,$
$next\_val, server\_pc, client\_pc, locked,$
$channels, client\_channel, client\_queue,$
$consume\_channel, outer\_states\rangle$

$client\_vars \triangleq \langle$
$client\_keys, client\_states, client\_pc,$
$consume\_channel, outer\_states\rangle$

$min\_val \triangleq 21$
$max\_val \triangleq 23$

$Value \triangleq min\_val \,..\, max\_val$

$NullValue \triangleq Value \cup \{nil\}$

$KevVal \triangleq [Key \to NullValue]$

$emptyKV \triangleq [k \in Key \mapsto nil]$

$Pair \triangleq Key \times Value$

$NullPair \triangleq Pair \cup \{nil\}$

$Channel \triangleq [data : NullPair, status : \{\text{“Empty”}, \text{“Ready”}, \text{“Consumed”}\}]$

$ClientState \triangleq \{\text{“Init”}, \text{“ClientCheckQueue”}, \text{“GetFromQueue”}, \text{“WaitOnChan”}\}$

$TypeOK \triangleq$
- $server\_state \in KevVal$
- $wait\_list \in [Key \to \text{SUBSET } Client]$
- $push\_back\_list \in [Key \to \text{SUBSET } Client]$
- $next\_val \in (min\_val - 1) \,..\, max\_val$
- $client\_keys \in [Client \to \text{SUBSET } Key]$
- $client\_states \in [Client \to KevVal]$
- $server\_pc \in \{\text{“Init”}, \text{“CheckWaitList”}, \text{“SetBackWaitList”}\}$
- $client\_pc \in [Client \to ClientState]$

$\land$ $locked \in \text{BOOLEAN}$
$\land$ $channels \in Seq(Channel)$
$\land$ $client\_channel \in [Client \to 1 \,..\, Len(channels) \cup \{nil\}]$
$\land$ $client\_queue \in [Client \to \text{SUBSET } Key]$
$\land$ $consume\_channel \in [Client \to 1 \,..\, Len(channels) \cup \{nil\}]$
$\land$ $outer\_states \in [Client \to KevVal]$

$Init \triangleq$
$\land\ server\_state = emptyKV$
$\land\ wait\_list = [k \ \in Key \mapsto \{\}]$
$\land\ push\_back\_list = [k \in Key \mapsto \{\}]$
$\land\ client\_keys \in [Client \to \text{SUBSET } Key]$
$\land\ \forall\, c \in Client : client\_keys[c] \neq \{\}$ client keys should not be empty
$\land\ client\_states = [c \in Client \mapsto emptyKV]$
$\land\ next\_val = min\_val - 1$
$\land\ server\_pc =$ "Init"
$\land\ client\_pc\ = [c \in Client \mapsto$ "Init"$]$
$\land\ locked = \text{FALSE}$
$\land\ channels = \langle\rangle$
$\land\ client\_channel = [c \in Client \mapsto nil]$
$\land\ client\_queue = [c \in Client \mapsto client\_keys[c]]$
$\land\ consume\_channel = [c \in Client \mapsto nil]$
$\land\ outer\_states = [c \in Client \mapsto emptyKV]$

$waitListEmpty \triangleq$
$\forall\, k \in Key : wait\_list[k] = \{\}$

$SetServerState(k) \triangleq$
$\land\ next\_val < max\_val$
$\land\ \neg locked$

$\land\ server\_pc =$ "Init"
$\land$ IF $waitListEmpty$
THEN
$\land$ UNCHANGED $locked$
$\land$ UNCHANGED $server\_pc$
ELSE
$\land\ locked' = \text{TRUE}$
$\land\ server\_pc' =$ "CheckWaitList"

$\land\ next\_val' = next\_val + 1$
$\land\ server\_state' = [server\_state \text{ EXCEPT } ![k] = next\_val']$

$\land$ UNCHANGED $\langle wait\_list, push\_back\_list\rangle$
$\land$ UNCHANGED $channels$

$\land$ UNCHANGED $client\_channel$
$\land$ UNCHANGED $client\_queue$
$\land$ UNCHANGED $client\_vars$

$setOutputChan(c,\ k) \triangleq$
LET
$index \triangleq client\_channel[c]$
$oldState \triangleq channels[index]$
$val \triangleq server\_state[k]$
$newState \triangleq [oldState$ EXCEPT $!.data = \langle k,\ val\rangle,\ !.status =$ "Ready"$]$
IN
$\land\ channels' = [channels$ EXCEPT $![index] = newState]$
$\land\ client\_channel' = [client\_channel$ EXCEPT $![c] = nil]$
$\land\ client\_states' = [client\_states$ EXCEPT $![c][k] = server\_state[k]]$

$waitListEmptyNew \triangleq$
$\forall\, k \in Key : wait\_list'[k] = \{\}$

$handleWaitEntryNoChange(c,\ k) \triangleq$
$\land\ push\_back\_list' = [push\_back\_list$ EXCEPT $![k] = @ \cup \{c\}]$
$\land$ UNCHANGED $channels$
$\land$ UNCHANGED $client\_states$
$\land$ UNCHANGED $client\_queue$
$\land$ UNCHANGED $client\_channel$

$handleWaitEntryChanged(c,\ k) \triangleq$
IF $client\_channel[c] \neq nil$
THEN
$\land\ setOutputChan(c,\ k)$
$\land\ client\_queue' = [client\_queue$ EXCEPT $![c] = @ \cup \{k\}]$
$\land$ UNCHANGED $push\_back\_list$
ELSE
$\land\ client\_queue' = [client\_queue$ EXCEPT $![c] = @ \cup \{k\}]$
$\land$ UNCHANGED $channels$
$\land$ UNCHANGED $client\_channel$
$\land$ UNCHANGED $client\_states$
$\land$ UNCHANGED $push\_back\_list$

$ServerCheckWaitList(k,\ c) \triangleq$
$\land\ server\_pc =$ "CheckWaitList"
$\land\ c \in wait\_list[k]$
$\land\ wait\_list' = [wait\_list$ EXCEPT $![k] = @ \setminus \{c\}]$

$\land$ IF $client\_states[c][k] = server\_state[k]$
THEN $handleWaitEntryNoChange(c,\ k)$

```
        ELSE  handleWaitEntryChanged(c, k)

    ∧ IF waitListEmptyNew
        THEN
            ∧ server_pc' = "SetBackWaitList"
            ∧ UNCHANGED locked
        ELSE
            ∧ UNCHANGED server_pc
            ∧ UNCHANGED locked

    ∧ UNCHANGED server_state
    ∧ UNCHANGED ⟨client_keys, client_pc, consume_channel, outer_states⟩
    ∧ UNCHANGED next_val

ServerSetBackWaitList ≜
    ∧ server_pc = "SetBackWaitList"
    ∧ server_pc' = "Init"
    ∧ locked' = FALSE
    ∧ wait_list' = push_back_list
    ∧ push_back_list' = [k ∈ Key ↦ {}]
    ∧ UNCHANGED server_state
    ∧ UNCHANGED channels
    ∧ UNCHANGED ⟨client_channel, client_queue⟩
    ∧ UNCHANGED client_vars
    ∧ UNCHANGED next_val


clientGoto(c, state) ≜ client_pc' = [client_pc EXCEPT ![c] = state]

newChannel ≜
    ∧ channels' = Append(channels, [data ↦ nil, status ↦ "Empty"])

newChannelIndex ≜ Len(channels')

GetState(c) ≜
    ∧ client_pc[c] = "Init"
    ∧ ¬locked
    ∧ locked' = TRUE
    ∧ newChannel
    ∧ client_channel' = [client_channel EXCEPT ![c] = newChannelIndex]
    ∧ clientGoto(c, "ClientCheckQueue")
    ∧ UNCHANGED ⟨client_keys, client_states, client_queue⟩
    ∧ UNCHANGED next_val
    ∧ UNCHANGED ⟨server_pc, server_state⟩
    ∧ UNCHANGED ⟨consume_channel, outer_states⟩
    ∧ UNCHANGED ⟨wait_list, push_back_list⟩
```

$ClientCheckQueue(c) \triangleq$
$\land\ client\_pc[c] = \text{“ClientCheckQueue”}$
$\land$ IF $client\_queue[c] = \{\}$
THEN
$\land\ clientGoto(c, \text{“WaitOnChan”})$
$\land\ consume\_channel' = [consume\_channel$ EXCEPT $![c] = client\_channel[c]]$
$\land\ locked' =$ FALSE
$\land$ UNCHANGED $client\_channel$
ELSE
$\land\ clientGoto(c, \text{“GetFromQueue”})$
$\land$ UNCHANGED $locked$
$\land$ UNCHANGED $client\_channel$
$\land$ UNCHANGED $consume\_channel$
$\land$ UNCHANGED $channels$
$\land$ UNCHANGED $\langle client\_queue, client\_states\rangle$
$\land$ UNCHANGED $\langle client\_keys\rangle$
$\land$ UNCHANGED $\langle server\_pc, server\_state\rangle$
$\land$ UNCHANGED $next\_val$
$\land$ UNCHANGED $\langle outer\_states\rangle$
$\land$ UNCHANGED $\langle wait\_list, push\_back\_list\rangle$

$GetFromQueue(c, k) \triangleq$
$\land\ client\_pc[c] = \text{“GetFromQueue”}$
$\land\ k \in client\_queue[c]$
$\land$ IF $client\_states[c][k] = server\_state[k]$
THEN
$\land\ client\_queue' = [client\_queue$ EXCEPT $![c] = @ \setminus \{k\}]$
$\land\ clientGoto(c, \text{“ClientCheckQueue”})$
$\land\ wait\_list' = [wait\_list$ EXCEPT $![k] = @ \cup \{c\}]$
$\land$ UNCHANGED $channels$
$\land$ UNCHANGED $client\_channel$
$\land$ UNCHANGED $client\_states$
$\land$ UNCHANGED $locked$
$\land$ UNCHANGED $consume\_channel$
$\land$ UNCHANGED $push\_back\_list$
ELSE
$\land\ clientGoto(c, \text{“WaitOnChan”})$
$\land\ locked' =$ FALSE
$\land\ setOutputChan(c, k)$
$\land\ consume\_channel' = [consume\_channel$ EXCEPT $![c] = client\_channel[c]]$
$\land$ UNCHANGED $client\_queue$
$\land$ UNCHANGED $wait\_list$
$\land$ UNCHANGED $push\_back\_list$
$\land$ UNCHANGED $\langle server\_pc, server\_state\rangle$

```
    ∧ UNCHANGED client_keys
    ∧ UNCHANGED next_val
    ∧ UNCHANGED outer_states

ConsumeFromChan(c) ≜
    LET
        index ≜ consume_channel[c]
        old_state ≜ channels[index]
        k ≜ old_state.data[1]
        val ≜ old_state.data[2]
        new_state ≜ [old_state EXCEPT !.data = nil, !.status = "Consumed"]
    IN
        ∧ client_pc[c] = "WaitOnChan"
        ∧ channels[index].status = "Ready"
        ∧ clientGoto(c, "Init")
        ∧ channels' = [channels EXCEPT ![index] = new_state]
        ∧ outer_states' = [outer_states EXCEPT ![c][k] = val]
        ∧ UNCHANGED ⟨client_keys, client_states, client_queue, client_channel⟩
        ∧ UNCHANGED consume_channel
        ∧ UNCHANGED locked
        ∧ UNCHANGED ⟨server_pc, server_state, next_val, wait_list⟩
        ∧ UNCHANGED push_back_list

TerminateCond ≜
    ∧ server_pc = "Init"
    ∧ ∀ c ∈ Client :
        ∧ client_pc[c] = "WaitOnChan"
        ∧ channels[consume_channel[c]].status = "Empty"
    ∧ next_val = max_val

Terminated ≜
    ∧ TerminateCond
    ∧ UNCHANGED vars

Next ≜
    ∨ ∃ k ∈ Key :
        ∨ SetServerState(k)
    ∨ ∃ c ∈ Client :
        ∨ GetState(c)
        ∨ ClientCheckQueue(c)
        ∨ ∃ k ∈ Key :
            ∨ GetFromQueue(c, k)
            ∨ ServerCheckWaitList(k, c)
        ∨ ConsumeFromChan(c)
```

$\lor\ ServerSetBackWaitList$
$\lor\ Terminated$

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

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

$Inv \triangleq$
$TerminateCond \Rightarrow$
$\forall\, c \in Client : \forall\, k \in client\_keys[c] :$
$\land\ client\_states[c][k] = server\_state[k]$
$\land\ outer\_states[c][k] = server\_state[k]$

$AlwaysTerminate \triangleq \Diamond TerminateCond$

$ChannelInv \triangleq$
$\forall\, index \in 1 \,..\, Len(channels) :$
LET
$ch \triangleq channels[index]$
IN
$\lor\ ch.data = nil \land ch.status =$ "Empty"
$\lor\ ch.data = nil \land ch.status =$ "Consumed"
$\lor\ ch.data \neq nil \land ch.status =$ "Ready"

$LockedCorrectly \triangleq$
$(server\_pc =$ "Init" $\land \forall\, c \in Client : client\_pc[c] =$ "Init" $) \Rightarrow \neg locked$

$allChannelConsumedExceptWaiting \triangleq$
$\forall\, i \in$ DOMAIN $channels :$
$(\forall\, c \in Client : consume\_channel[c] \neq i) \Rightarrow channels[i].status =$ "Consumed"

$AllChannelConsumed \triangleq$
$TerminateCond \Rightarrow allChannelConsumedExceptWaiting$

$channelPushOrRecv \triangleq$
$\forall\, index \in 1 \,..\, Len(channels) :$
LET
$before \triangleq channels[index]$
$after \triangleq channels'[index]$
IN
$\lor\ \land\ before.status =$ "Empty"
$\land\ after.status =$ "Ready"
$\lor\ \land\ before.status =$ "Ready"
$\land\ after.status =$ "Consumed"

$$\lor\ before = after$$

$$channelPushRecvOrAppend \triangleq$$
$$\quad \lor\ channelPushOrRecv$$
$$\quad \lor\ Len(channels') = Len(channels) + 1$$

$$ChannelPushInv \triangleq$$
$$\quad \Box[channelPushRecvOrAppend]_{channels}$$

$$Symm \triangleq Permutations(Key) \cup Permutations(Client)$$