MODULE *HKFM*

EXTENDS *Integers*, *Sequences*
CONSTANT *Client*, *Song*
VARIABLES *inbox*, *state*

Definitions

$$vars \triangleq \langle inbox,\ state \rangle$$

$$Server \triangleq \text{CHOOSE } x : x \notin Client$$
$$Node \triangleq Client \cup \{Server\}$$

$$Idx \triangleq Nat \cup \{-1\}$$
$$Playlist \triangleq Seq(Song)$$
$$Playhead \triangleq [i : Idx,\ t : Nat]$$
$$Stopped \triangleq [i \mapsto -1,\ t \mapsto 0]$$
$$State \triangleq [playlist : Playlist,\ playhead : Playhead]$$
$$InitState \triangleq [playlist \mapsto \langle\rangle,\ playhead \mapsto Stopped]$$

$$\begin{aligned} Message \triangleq\ & [action : \{\text{“sync”}\},\ data : State] \cup \\ & [action : \{\text{“add”}\},\ data : Song,\ sender : Client] \cup \\ & [action : \{\text{“seek”},\ \text{“skip”}\},\ data : Playhead,\ sender : Client] \end{aligned}$$

$$\begin{aligned} TypeOK \triangleq\ & \land\ inbox \in [Node \to Seq(Message)] \\ & \land\ state\ \in [Node \to State] \end{aligned}$$

Message Constructors

$$SyncMsg \triangleq$$
$$[action \mapsto \text{“sync”},\ data \mapsto state'[Server]]$$

$$AddMsg(client,\ song) \triangleq$$
$$[action \mapsto \text{“add”},\ data \mapsto song,\ sender \mapsto client]$$

$$SeekMsg(client,\ playhead) \triangleq$$
$$[action \mapsto \text{“seek”},\ data \mapsto playhead,\ sender \mapsto client]$$

$$SkipMsg(client,\ playhead) \triangleq$$
$$[action \mapsto \text{“skip”},\ data \mapsto playhead,\ sender \mapsto client]$$

Client Actions

$$SendAdd(self,\ song) \triangleq$$
LET
$$msg \triangleq AddMsg(self,\ song)$$
IN

$\land\ inbox' = [inbox \text{ EXCEPT } ![Server] = Append(inbox[Server],\ msg)]$
$\land$ UNCHANGED $state$

$RecvSync(self) \triangleq$
$\land\ inbox[self] \neq \langle\rangle$
$\land\ Head(inbox[self]).action =$ "sync"
$\land$ LET
$newState \triangleq Head(inbox[self]).data$
IN
$\land\ inbox' = [inbox \text{ EXCEPT } ![self] = Tail(inbox[self])]$
$\land\ state' = [state \text{ EXCEPT } ![self] = newState]$

$SendSeek(self) \triangleq$
LET
$playhead \triangleq state[self].playhead$
$msg \triangleq SeekMsg(self,\ [playhead \text{ EXCEPT } !.t = playhead.t + 1])$
IN
$\land\ playhead.i \neq -1$
$\land\ inbox' = [inbox \text{ EXCEPT } ![Server] = Append(inbox[Server],\ msg)]$
$\land$ UNCHANGED $state$

$SendSkip(self) \triangleq$
LET
$playhead \triangleq state[self].playhead$
$msg \triangleq SkipMsg(self,\ playhead)$
IN
$\land\ playhead.i \neq -1$
$\land\ inbox' = [inbox \text{ EXCEPT } ![Server] = Append(inbox[Server],\ msg)]$
$\land$ UNCHANGED $state$

---

Server Actions

$BroadcastSync \triangleq$
$\land\ inbox' = [n \in Node \mapsto \text{IF } n = Server$
$\text{THEN } Tail(inbox[n])$
$\text{ELSE } Append(inbox[n],\ SyncMsg)]$

$RecvAdd \triangleq$
$\land\ inbox[Server] \neq \langle\rangle$
$\land\ Head(inbox[Server]).action =$ "add"
$\land$ LET
$song \triangleq Head(inbox[Server]).data$
$playlist \triangleq state[Server].playlist$
$playhead \triangleq state[Server].playhead$
$newPlaylist \triangleq Append(playlist,\ song)$
$newPlayhead \triangleq \text{IF } playhead.i = -1$

```
                    THEN [i ↦ Len(playlist), t ↦ 0]
                    ELSE playhead
    IN
      ∧ state' = [state EXCEPT ![Server] = [playlist  ↦ newPlaylist,
                                            playhead ↦ newPlayhead]]
      ∧ BroadcastSync

RecvSeek ≜
  ∧ inbox[Server] ≠ ⟨⟩
  ∧ LET
      server ≜ state[Server]
      msg ≜ Head(inbox[Server])
    IN
      ∧ msg.action = "seek"
      ∧ msg.data.i = server.playhead.i
      ∧ state' = [state EXCEPT ![Server].playhead.t = msg.data.t]
      ∧ BroadcastSync

RecvSkip ≜
  ∧ inbox[Server] ≠ ⟨⟩
  ∧ LET
      server ≜ state[Server]
      msg ≜ Head(inbox[Server])
    IN
      ∧ msg.action = "skip"
      ∧ msg.data.i = server.playhead.i
      ∧ LET
          newIndex ≜ server.playhead.i + 1
          newPlayhead ≜ IF newIndex < Len(server.playlist)
                        THEN [i ↦ newIndex, t ↦ 0]
                        ELSE Stopped
        IN
          ∧ state' = [state EXCEPT ![Server].playhead = newPlayhead]
          ∧ BroadcastSync
```

Spec

```
Init ≜
  ∧ inbox = [n ∈ Node ↦ ⟨⟩]
  ∧ state = [n ∈ Node ↦ InitState]

Next ≜
  ∨ ∃ self ∈ Client, song ∈ Song : SendAdd(self, song)
  ∨ ∃ self ∈ Client : RecvSync(self)
  ∨ ∃ self ∈ Client : SendSeek(self)
```

$\lor \exists\, self \in Client : SendSkip(self)$
$\lor\ RecvAdd$
$\lor\ RecvSeek$
$\lor\ RecvSkip$

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

---

Invariants

$PlayheadOK \triangleq$
LET
$server \triangleq state[Server].playhead$
IN
$\lor\ server = Stopped$
$\lor \forall\, c \in Client :$
LET
$client \triangleq state[c].playhead$
IN
$\lor\ client.i < server.i$
$\lor\ \land\ client.i = server.i$
$\land\ client.t \leq server.t$

THEOREM $Spec \Rightarrow \Box TypeOK$
THEOREM $Spec \Rightarrow \Box PlayheadOK$

---