MODULE *HKFM*

EXTENDS *Integers*, *Sequences*
CONSTANTS *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

```
    ∧ inbox' = [inbox EXCEPT ![Server] = Append(inbox[Server], msg)]
    ∧ UNCHANGED state

RecvSync(self) ≜
  ∧ inbox[self] ≠ ⟨⟩
  ∧ Head(inbox[self]).action = "sync"
  ∧ LET
      newState ≜ Head(inbox[self]).data
    IN
      ∧ inbox' = [inbox EXCEPT ![self] = Tail(inbox[self])]
      ∧ state' = [state EXCEPT ![self] = newState]

SendSeek(self) ≜
  LET
    playhead ≜ state[self].playhead
    msg ≜ SeekMsg(self, [playhead EXCEPT !.t = playhead.t + 1])
  IN
    ∧ playhead.i ≠ −1
    ∧ inbox' = [inbox EXCEPT ![Server] = Append(inbox[Server], msg)]
    ∧ UNCHANGED state

SendSkip(self) ≜
  LET
    playhead ≜ state[self].playhead
    msg ≜ SkipMsg(self, playhead)
  IN
    ∧ playhead.i ≠ −1
    ∧ inbox' = [inbox EXCEPT ![Server] = Append(inbox[Server], msg)]
    ∧ UNCHANGED state
```

Server Actions

```
BroadcastSync ≜
  ∧ inbox' = [n ∈ Node ↦ IF n = Server
                          THEN Tail(inbox[n])
                          ELSE Append(inbox[n], SyncMsg)]

RecvAdd ≜
  ∧ inbox[Server] ≠ ⟨⟩
  ∧ LET
      server ≜ state[Server]
      msg ≜ Head(inbox[Server])
    IN
      ∧ msg.action = "add"
      ∧ LET
          newPlaylist ≜ Append(server.playlist, msg.data)
```

```
        newPlayhead ≜ IF server.playhead.i = −1
                      THEN [i ↦ Len(server.playlist), t ↦ 0]
                      ELSE server.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
      ∧ msg.data.t > server.playhead.t
      ∧ 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
```

Randomly lose a message from an *inbox*

```
Remove(i, seq) ≜
  [j ∈ 1 .. (Len(seq) − 1) ↦ IF j < i THEN seq[j] ELSE seq[j + 1]]

LoseMsg ≜
  ∃ n ∈ DOMAIN inbox :
    ∃ i ∈ DOMAIN inbox[n] :
```

$\land\ inbox' = [inbox \text{ EXCEPT } ![n] = Remove(i,\ inbox[n])]$
$\land$ UNCHANGED $state$

Spec

$Init \triangleq$
$\land\ inbox = [n \in Node \mapsto \langle\rangle]$
$\land\ state\ = [n \in Node \mapsto InitState]$

$Next \triangleq$
$\lor\ \exists\, self \in Client,\ song \in Song : SendAdd(self,\ song)$
$\lor\ \exists\, self \in Client : RecvSync(self)$
$\lor\ \exists\, self \in Client : SendSeek(self)$
$\lor\ \exists\, self \in Client : SendSkip(self)$
$\lor\ RecvAdd$
$\lor\ RecvSeek$
$\lor\ RecvSkip$
$\lor\ LoseMsg$

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

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
$\lor\ client.i < server.i$
$\lor\ \land\ client.i = server.i$
$\land\ client.t \leq server.t$

$SeekAdvances \triangleq$
$\lor\ state[Server].playhead.i \neq state'[Server].playhead.i$
$\lor\ state[Server].playhead.t \leq state'[Server].playhead.t$

$Synced \triangleq$
$\forall\, c \in Client : state[c] = state[Server]$

THEOREM $Spec \Rightarrow \Box TypeOK$
THEOREM $Spec \Rightarrow \Box PlayheadOK$
THEOREM $Spec \Rightarrow \Box\Diamond Synced$
THEOREM $Spec \Rightarrow \Box[SeekAdvances]_{vars}$