MODULE *messaging*

EXTENDS $\mathit{Naturals}, \mathit{PT}, \mathit{FiniteSets}$

The *messaging* in the fridjapp needs to ensure a coherence in the data. This is achieved with version numbers and sending of messages between devices. Here we specify the synchronization of one fridj across various instances hosted by different devices, one being the lead (the creator of the shared fridj).

CONSTANTS $\mathit{DEVICES}$, contains synchronizing devices
DATA, all possible data values
LEAD the creator of the fridj has precedence

ASSUME $\mathit{LEAD} \in \mathit{DEVICES}$

VARIABLES $\mathit{fridjs}, \mathit{msgs}, \mathit{network}$

$\mathit{vars} \triangleq \langle \mathit{fridjs}, \mathit{msgs}, \mathit{network} \rangle$

The fridj's data is represented by some element in the set *DATA*.

$\mathit{FridjValue} \triangleq [v : \mathit{Nat},$ version
$d : \mathit{DATA}]$ some data (could be an ingredient item)

$\mathit{Fridj} \triangleq [\mathit{DEVICES} \rightarrow \mathit{FridjValue}]$

$\mathit{FridjInv} \triangleq \mathit{fridjs} \in \mathit{Fridj}$

All messages ever sent are added to the *msgs*. Messages not yet sent can be modified by the sending device.

$\mathit{Msg} \triangleq [\mathit{to} : \mathit{DEVICES},$ device receiving the message
$\mathit{from} : \mathit{DEVICES},$
$\mathit{sent} : \text{BOOLEAN},$ true iff to and from are in the network
$\mathit{type} : \{\text{“synch”}, \text{“conn”}\},$ either a connection or *synch*
$\mathit{data} : \mathit{FridjValue}]$

$\mathit{MsgInv} \triangleq \mathit{msgs} \subseteq \mathit{Msg}$

The network contains the list of connected devices.

$\mathit{Network} \triangleq \text{SUBSET } \mathit{DEVICES}$

$\mathit{NetworkInv} \triangleq \mathit{network} \in \mathit{Network}$

First version of a fridj is any Natural number.

$\mathit{FirstVersion} \triangleq \exists\, n : n \in \mathit{Nat}$

Compute the maximum version of the fridj by looking at each instance's version.

$\mathit{LattestVersion} \triangleq \mathit{ReduceSet}($
LAMBDA $d, v : \mathit{Max}(\mathit{fridjs}[d].v, v),$
$\mathit{DEVICES}, \mathit{FirstVersion})$

What a *synch* version and data value means: nothing.

$$\begin{array}{l}
\mathit{AnyVersion} \triangleq \mathit{FirstVersion} \\
\mathit{AnyData} \triangleq \textsc{choose}\ d \in \mathit{DATA} : \textsc{true}
\end{array}$$

Following definitions are state functions and actions taken by the user or that represent the management of a transaction across devices.

Device's *conn* messages have been sent

$$\begin{array}{l}
\mathit{AllConnMsgsSent}(\mathit{device}) \triangleq \\
\quad \neg\exists\, \mathit{msg} \in \mathit{msgs} : \begin{array}[t]{l} \land\ \mathit{msg.type} = \text{``conn''} \\ \land\ \mathit{msg.sent} \\ \land\ \mathit{msg.from} = \mathit{device} \end{array}
\end{array}$$

All received *conn* messages have been read.

$$\begin{array}{l}
\mathit{AllConnMsgsRead}(\mathit{device}) \triangleq \\
\quad \neg\exists\, \mathit{msg} \in \mathit{msgs} : \begin{array}[t]{l} \land\ \mathit{msg.type} = \text{``conn''} \\ \land\ \mathit{msg.sent} \\ \land\ \mathit{msg.to} = \mathit{device} \end{array}
\end{array}$$

All received *synch* messages have been read.

$$\begin{array}{l}
\mathit{AllSynchMsgsRead}(\mathit{device}) \triangleq \\
\quad \neg\exists\, \mathit{msg} \in \mathit{msgs} : \begin{array}[t]{l} \land\ \mathit{msg.type} = \text{``synch''} \\ \land\ \mathit{msg.sent} \\ \land\ \mathit{msg.to} = \mathit{device} \end{array}
\end{array}$$

Send a message to notify the other devices of a state change.

$$\begin{array}{l}
\mathit{Notify}(\mathit{devices}, \mathit{version}, \mathit{data}, \mathit{sender}) \triangleq \\
\quad \land\ \mathit{msgs}' = \mathit{msgs} \cup \{[\begin{array}[t]{l} \mathit{to} \mapsto d, \\ \mathit{from} \mapsto \mathit{sender}, \\ \mathit{sent}\ \ \mapsto \begin{array}[t]{l} \land\ d \in \mathit{network} \\ \land\ \mathit{sender} \in \mathit{network}, \end{array} \\ \mathit{type}\ \ \mapsto \text{``synch''}, \\ \mathit{data}\ \ \mapsto [v \mapsto \mathit{version},\ d \mapsto \mathit{data}]] : d \in \mathit{devices}\} \end{array}
\end{array}$$

Send messages not yet sent through network.

$$\begin{array}{l}
\mathit{SendAll}(\mathit{device}) \triangleq \\
\quad \land\ \ \mathit{device} \in \mathit{network} \\
\quad \land\ \ \mathit{msgs}' = \{m \in \mathit{msgs} : m.\mathit{from} \neq \mathit{device} \lor m.\mathit{sent} = \textsc{true}\} \cup \\
\qquad\qquad\quad \{[\mathit{dm}\ \textsc{except}\ !.\mathit{sent} = \mathit{dm.to} \in \mathit{network}] : \\
\qquad\qquad\qquad \mathit{dm} \in \{m \in \mathit{msgs} : m.\mathit{from} = \mathit{device} \land m.\mathit{data}.v \geq \mathit{fridjs}[\mathit{device}].v\}\} \\
\quad \land\ \ \textsc{unchanged}\ \langle \mathit{fridjs}, \mathit{network} \rangle
\end{array}$$

It's required to read messages to update the data to a new version.

```
$ReadSynchMsg(device) \triangleq \exists\, msg \in \{m \in msgs : m.type = \text{“synch”}\} :$
    $\land\ msg.to = device$
    $\land\ msg.sent$
    $\land\ msg.data.v \geq fridjs[device].v$
    $\land\ fridjs' = [fridjs\ \text{EXCEPT}\ ![device] = msg.data]$
    $\land\ msgs' = msgs \setminus \{msg\}$
    $\land\ \text{UNCHANGED}\ network$
```

When a device connects to the network, it sends a notification to the connected devices. So we need to read these messages too. There are two cases:
- the sent version is newer so we update on our side
- the sent version is older and we send an update

```
$ReadConnMsg(device) \triangleq \exists\, msg \in \{m \in msgs : m.type = \text{“conn”}\} :$
    $\land\ msg.to = device$
    $\land\ msg.sent$
    $\land\ AllSynchMsgsRead(device)$
    $\land\ \text{IF}\ msg.data.v \geq fridjs[device].v$
        THEN  $\land\ fridjs' = [fridjs\ \text{EXCEPT}\ ![device] = msg.data]$
              $\land\ msgs' = msgs \setminus \{msg\}$
        ELSE  $\land\ \text{LET}\ msg\_sent \triangleq device \in network \land msg.from \in network$
                     $new\_msg \triangleq [to \mapsto msg.from,$
                                  $from \mapsto device,$
                                  $sent \mapsto msg\_sent,$
                                  $type \mapsto \text{“synch”},$
                                  $data \mapsto fridjs[device]]$
                 IN  $msgs' = (msgs \setminus \{msg, [new\_msg\ \text{EXCEPT}\ !.sent = \text{FALSE}]\})$
                            $\cup \{new\_msg\}$
              $\land\ \text{UNCHANGED}\ fridjs$
    $\land\ \text{UNCHANGED}\ network$
```

Update the fridj after the user's input. Here implemented by a simple increment of the counter.

```
$FridjUserInput(device) \triangleq \exists\, dt \in DATA :$
   LET $new\_version \triangleq fridjs[device].v + 1$
   IN
        no *synch* to do or new connections to handle
        $\land\ \neg\exists\, msg \in msgs : msg.to = device \land msg.sent$
        wait for *synch* on device's connection
        $\land\ AllConnMsgsSent(device)$
        $\land\ LEAD \in network \land device \in network$
        $\land\ fridjs' = [fridjs\ \text{EXCEPT}\ ![device] = [d \mapsto dt,$
                                                  $v \mapsto new\_version]]$
        $\land\ Notify(\{d \in DEVICES : d \neq device\},$
                 $new\_version,$
                 $dt,$
                 $device)$
```

```
    $\land$ UNCHANGED $network$
```

Devices must connect to the network to be able to share messages.

```
$Connect(device) \triangleq$
    $\land \quad device \notin network$
    $\land \quad network' = network \cup \{device\}$
    $\land \quad AllSynchMsgsRead(device)$
    $\land \quad AllConnMsgsRead(device)$
    send $msg$ to other connected devices
    $\land\ msgs' = msgs \cup \{[to \mapsto d,$
                           $from \mapsto device,$
                           $sent \mapsto d \in network,$
                           $type \mapsto \text{“conn”},$
                           $data \mapsto fridjs[device]] : d \in network\}$
    $\land$ UNCHANGED $\langle fridjs \rangle$
```

Every device can loose its internet connection at some point.

```
$Disconnect(device) \triangleq$
    $\land\ device \in network$
    $\land\ AllConnMsgsRead(device)$
    $\land\ network' = network \setminus \{device\}$
    $\land$ UNCHANGED $\langle msgs,\ fridjs \rangle$
```

Temporal formula definition of the specification.

```
$Init \triangleq \exists\, dt \in DATA :$
    $\land\ fridjs = [d \in DEVICES \mapsto [v \mapsto FirstVersion,$
                                          $d \mapsto dt]]$
    $\land\ msgs = \{\}$
    $\land\ network = \{\}$

$Next \triangleq \exists\, d \in DEVICES :$
    $\lor\ FridjUserInput(d)$
    $\lor\ ReadSynchMsg(d)$
    $\lor\ ReadConnMsg(d)$
    $\lor\ Connect(d)$
    $\lor\ Disconnect(d)$
    $\lor\ SendAll(d)$

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

Invariant: For every fridj instance, version equality implies data equality.

```
$SynchronizedFridjs \triangleq \forall\, d1,\ d2 \in DEVICES :$
   $fridjs[d1].v = fridjs[d2].v \Rightarrow fridjs[d1].d = fridjs[d2].d$
```

Associated theorems for previously defined type and safety invariants.

$$\text{THEOREM } \mathit{Spec} \Rightarrow \begin{array}{l} \land \Box \mathit{FridjInv} \\ \land \Box \mathit{MsgInv} \\ \land \Box \mathit{NetworkInv} \\ \land \Box \mathit{SynchronizedFridjs} \end{array}$$

* Modification History
* Last modified Sun *Oct* 27 01:09:52 *CEST* 2024 by *davd*33
* Created Sat *Oct* 26 14:17:55 *CEST* 2024 by *davd*33