MODULE *optishopylist*

EXTENDS TLC, $Integers$, $FiniteSets$, $Sequences$
CONSTANTS $PRODUCTS$, $APPS$, IDs, $GATEAPP$

$PT \triangleq$ INSTANCE PT

$set ++ item \triangleq set \cup \{item\}$
$set -- item \triangleq set \setminus \{item\}$

In shoppying list, the product is in fact the identifier. Any item could have an information for how much of a product one wants to buy (not relevant in this specification).

$ShopyItems \triangleq [id : PRODUCTS, bought : \text{BOOLEAN}]$

$ADD\_ACTION \triangleq$ "add"
$RM\_ACTION \triangleq$ "rm"
$SET\_BOUGHT\_ACTION \triangleq$ "set_bought"
$REQ\_SYNC\_ACTION \triangleq$ "req_sync"
$RESP\_SYNC\_ACTION \triangleq$ "resp_sync"
$END\_SYNC\_ACTION \triangleq$ "end_sync"

Actions is the set of all possible actions in the system.

$$Actions \triangleq \{ ADD\_ACTION, RM\_ACTION, SET\_BOUGHT\_ACTION, REQ\_SYNC\_ACTION, RESP\_SYNC\_ACTION, END\_SYNC\_ACTION \}$$

$SyncActions \triangleq \{REQ\_SYNC\_ACTION, RESP\_SYNC\_ACTION\}$

SyncMsgs is the set of all possible messages sent for synchronisation of shopy lists.

$$SyncMsgs \triangleq [id : IDs, app : APPS, list : \text{SUBSET } ShopyItems, mergedList : \text{SUBSET } ShopyItems, type : SyncActions]$$

The spec now depicts a shopping-list *app* where the server *app* manages several users and hence multiple lists of items that synch eventually.

The list contains unique items, thus we use a set.

--algorithm *OptiShopyList*

variable

one shopping list for all $APPS$
$shopyList = [a \in APPS \mapsto \{\}]$,
sync requests for all $APPS$
$syncReqQueue = [a \in APPS \mapsto \langle\rangle]$,
sync responses for all $APPS$
$syncRespQueue = [a \in APPS \mapsto \langle\rangle]$,
set of taken IDs
$takenIDs = \{\}$;

define

A couple of helpers for shopy-list items

$NewShopyItem(list) \triangleq$
$[id \mapsto (\text{CHOOSE } x \in PRODUCTS : \neg\exists\, i \in list : x = i.id),$
$bought \mapsto \text{FALSE}]$

$ExistingShopyItem(list) \triangleq \text{CHOOSE } x \in list : \text{TRUE}$

$ExistingNotBoughtShopyItem(list) \triangleq \text{CHOOSE } x \in list : x.bought = \text{FALSE}$

Helpers for *Sync* messages request/response.

$NewSyncMsg(id,\ a,\ l,\ ml,\ t) \triangleq$
$[id \mapsto id,$
$app \mapsto a,$
$list \mapsto l,$
$mergedList \mapsto ml,$
$type \mapsto t]$

$NewSyncReqMsg(a,\ l,\ ml,\ t) \triangleq$
$NewSyncMsg($
$(\text{CHOOSE } i \in IDs : \forall\, ti \in takenIDs : i = ti),$
$a,\ l,\ ml,\ t$
$)$

$NewSyncReq(app) \triangleq$
$NewSyncReqMsg(app,\ shopyList[app],\ \{\},\ REQ\_SYNC\_ACTION)$

$NewSyncResp(app,\ mergeResult,\ id) \triangleq$
$NewSyncMsg(id,\ app,\ shopyList[app],\ mergeResult,\ RESP\_SYNC\_ACTION)$

Helpers for the decentralized network features.

$IsGate(app) \triangleq app = GATEAPP$

end define ;

This proccess represents a shopy-list running in one of the several offline clients. Several shopy-lists are running on various *app* processes. Actually, we specify what happens in the system when 2 client apps want to synch their shopy-list. We assume that every client *app* has only one such list since the user is able to synch any 2 lists together.

fair process $ClientApp \in APPS$
variables $gossipFriends$;

begin *AppLoop*:
  while TRUE **do**

    either
      ADD
      await $Cardinality(shopyList[self]) < Cardinality(PRODUCTS)$;
      $shopyList[self] := shopyList[self] \mathbin{++} NewShopyItem(shopyList[self])$;
    or
      REMOVE
      await $shopyList[self] \neq \{\}$;
      $shopyList[self] := shopyList[self] \mathbin{--} ExistingShopyItem(shopyList[self])$;
    or
      set to BOUGHT
      await $shopyList[self] \neq \{\}$;
      await $\exists\, item \in shopyList[self] : \neg item.bought$;
      with $modifiedItem = ExistingNotBoughtShopyItem(shopyList[self])$ **do**
        $shopyList[self] := shopyList[self] \mathbin{--} modifiedItem \mathbin{++} [modifiedItem$ EXCEPT $!.bought = \text{TRUE}$
      end with ;
    or
      request a sync with another *app*
      with $a \in (APPS \mathbin{--} self)$,
           $newRequest = NewSyncReq(self)$ **do**

        $syncReqQueue[a] := Append(syncReqQueue[a],\ newRequest)$;
      end with ;
    or
      receive a sync request from another *app*
      await $syncReqQueue[self] \neq \langle\rangle$;
      with $syncRequest = Head(syncReqQueue[self])$,
           $mergeResult = shopyList[self] \cup syncRequest.list$,
           $newResp = NewSyncResp(self,\ mergeResult,\ syncRequest.id)$ **do**

        $syncReqQueue[self] := Tail(syncReqQueue[self])$;
        merge from request *app*
        $shopyList[self] := mergeResult$;
        $syncRespQueue[syncRequest.app] := Append(syncRespQueue[syncRequest.app],\ newResp)$;
      end with ;
    or
      receive a sync response
      await $syncRespQueue[self] \neq \langle\rangle$;

```
            with syncResponse = Head(syncRespQueue[self]),
                 mergeResult = shopyList[self] ∪ syncResponse.list do

                shopyList[self] := mergeResult ;
                syncRespQueue[self] := Tail(syncRespQueue[self]) ;
            end with ;
        end either ;
    end while ;
end process ;

end algorithm ;
```

BEGIN TRANSLATION ($chksum(pcal) =$ "$f4ae31c$" $\wedge\ chksum(tla) =$ "$894de1a$")

CONSTANT *defaultInitValue*

VARIABLES *shopyList*, *syncReqQueue*, *syncRespQueue*, *takenIDs*

define statement

$$NewShopyItem(list) \triangleq$$
$$[id \mapsto (\text{CHOOSE } x \in PRODUCTS : \neg\exists\, i \in list : x = i.id),$$
$$\ bought \mapsto \text{FALSE}]$$

$$ExistingShopyItem(list) \triangleq \text{CHOOSE } x \in list : \text{TRUE}$$

$$ExistingNotBoughtShopyItem(list) \triangleq \text{CHOOSE } x \in list : x.bought = \text{FALSE}$$

$$NewSyncMsg(id,\ a,\ l,\ ml,\ t) \triangleq$$
$$[id \mapsto id,$$
$$\ app \mapsto a,$$
$$\ list \mapsto l,$$
$$\ mergedList \mapsto ml,$$
$$\ type \mapsto t]$$

$$NewSyncReqMsg(a,\ l,\ ml,\ t) \triangleq$$
$$NewSyncMsg($$
$$(\text{CHOOSE } i \in IDs : \forall\, ti \in takenIDs : i = ti),$$
$$a,\ l,\ ml,\ t$$
$$)$$

$$NewSyncReq(app) \triangleq$$
$$NewSyncReqMsg(app,\ shopyList[app],\ \{\},\ REQ\_SYNC\_ACTION)$$

$$NewSyncResp(app,\ mergeResult,\ id) \triangleq$$
$$NewSyncMsg(id,\ app,\ shopyList[app],\ mergeResult,\ RESP\_SYNC\_ACTION)$$

$IsGate(app) \triangleq app = GATEAPP$

VARIABLE $gossipFriends$

$vars \triangleq \langle shopyList,\ syncReqQueue,\ syncRespQueue,\ takenIDs,\ gossipFriends \rangle$

$ProcSet \triangleq (APPS)$

$Init \triangleq$ Global variables
$\land\ shopyList = [a \in APPS \mapsto \{\}]$
$\land\ syncReqQueue = [a \in APPS \mapsto \langle\rangle]$
$\land\ syncRespQueue = [a \in APPS \mapsto \langle\rangle]$
$\land\ takenIDs = \{\}$
Process *ClientApp*
$\land\ gossipFriends = [self \in APPS \mapsto defaultInitValue]$

$ClientApp(self) \triangleq$
$\land \lor \land Cardinality(shopyList[self]) < Cardinality(PRODUCTS)$
$\land\ shopyList' = [shopyList \text{ EXCEPT } ![self] = shopyList[self] ++ NewShopyItem(shop$
$\land$ UNCHANGED $\langle syncReqQueue,\ syncRespQueue \rangle$
$\lor \land shopyList[self] \neq \{\}$
$\land\ shopyList' = [shopyList \text{ EXCEPT } ![self] = shopyList[self] -- ExistingShopyItem(s$
$\land$ UNCHANGED $\langle syncReqQueue,\ syncRespQueue \rangle$
$\lor \land shopyList[self] \neq \{\}$
$\land \exists\, item \in shopyList[self] : \neg item.bought$
$\land$ LET $modifiedItem \triangleq ExistingNotBoughtShopyItem(shopyList[self])$IN
$shopyList' = [shopyList \text{ EXCEPT } ![self] = shopyList[self] -- modifiedItem ++ [n$
$\land$ UNCHANGED $\langle syncReqQueue,\ syncRespQueue \rangle$
$\lor \land \exists\, a \in (APPS -- self) :$
LET $newRequest \triangleq NewSyncReq(self)$IN
$syncReqQueue' = [syncReqQueue \text{ EXCEPT } ![a] = Append(syncReqQueue[a],\ ne$
$\land$ UNCHANGED $\langle shopyList,\ syncRespQueue \rangle$
$\lor \land syncReqQueue[self] \neq \langle\rangle$
$\land$ LET $syncRequest \triangleq Head(syncReqQueue[self])$IN
LET $mergeResult \triangleq shopyList[self] \cup syncRequest.list$IN
LET $newResp \triangleq NewSyncResp(self,\ mergeResult,\ syncRequest.id)$IN
$\land\ syncReqQueue' = [syncReqQueue \text{ EXCEPT } ![self] = Tail(syncReqQueue[se$
$\land\ shopyList' = [shopyList \text{ EXCEPT } ![self] = mergeResult]$
$\land\ syncRespQueue' = [syncRespQueue \text{ EXCEPT } ![syncRequest.app] = Append$
$\lor \land syncRespQueue[self] \neq \langle\rangle$
$\land$ LET $syncResponse \triangleq Head(syncRespQueue[self])$IN
LET $mergeResult \triangleq shopyList[self] \cup syncResponse.list$IN
$\land\ shopyList' = [shopyList \text{ EXCEPT } ![self] = mergeResult]$
$\land\ syncRespQueue' = [syncRespQueue \text{ EXCEPT } ![self] = Tail(syncRespQueue[$
$\land$ UNCHANGED $syncReqQueue$
$\land$ UNCHANGED $\langle takenIDs,\ gossipFriends \rangle$

$Next \triangleq (\exists\, self \in APPS : ClientApp(self))$

$$Spec \triangleq \land Init \land \Box[Next]_{vars} \\ \land \forall self \in APPS : \mathrm{WF}_{vars}(ClientApp(self))$$

END TRANSLATION

$$TypeOK \triangleq \\ \land \quad \forall a \in APPS : \\ \land shopyList[a] \subseteq ShopyItems \\ \land PT!Range(syncReqQueue[a]) \subseteq SyncMsgs \\ \land PT!Range(syncRespQueue[a]) \subseteq SyncMsgs \\ \land \quad takenIDs \subseteq IDs$$

* Modification History
* Last modified Sun *Mar* 07 16:22:38 *CET* 2021 by *davd*
* Created *Tue Mar* 02 12:33:43 *CET* 2021 by *davd*