MODULE *IdemProxy*

Expectations :
- Every request must hit the server exactly once
- For each try of a request, the response must be the same
- 2 requests must be processable in parallel

EXTENDS *Integers*, *FiniteSets*

CONSTANTS
_ReqTokens, the request tokens used as idempotent keys
_MaxTries how many times a request is retried

ASSUME $\_MaxTries < 10$

VARIABLES
requests, the state of all requests and their corresponding tries
locks locks on specific requests

$vars \triangleq \langle requests, locks \rangle$

$tryKeys \triangleq 1 \,..\, \_MaxTries$ simple helper providing the set of try keys

$TypeInvariants \triangleq$
$\land \forall\, req \in \_ReqTokens,\, x \in tryKeys :$
$requests[req][x] \in \{$
"pending",
"inProxy",
"lock",
"processed",
"cached",
"fromCache"
$\}$
$\land \forall\, req \in \_ReqTokens : locks[req] \in$ BOOLEAN

$Init \triangleq$
requests is a struct with *_ReqTokens* as keys associated with structs with *tryKeys* as keys associatied with the given try current status
$\land\ requests = [req \in \_ReqTokens \mapsto [st \in tryKeys \mapsto$ "pending"$]]$
$\land\ locks = [req \in \_ReqTokens \mapsto$ FALSE$]$

Actions

$HitProxy(r,\, i) \triangleq$
$\land\ requests[r][i] =$ "pending"

$\land\ requests' = [requests \text{ EXCEPT } ![r][i] = \text{“inProxy”}]$
$\land$ UNCHANGED $\langle locks \rangle$

$Lock(r,\ i) \triangleq$
check lock and set lock is atomic here
$\land\ requests[r][i] = \text{“inProxy”}$
$\land\ locks[r] = \text{FALSE}$
$\land\ Cardinality(\{x \in \text{DOMAIN } requests[r] : requests[r][x] = \text{“cached”}\}) = 0$
$\land\ requests' = [requests \text{ EXCEPT } ![r][i] = \text{“lock”}]$
$\land\ locks' = [locks \text{ EXCEPT } ![r] = \text{TRUE}]$

$HitServer(r,\ i) \triangleq$
$\land\ requests[r][i] = \text{“lock”}$
$\land\ requests' = [requests \text{ EXCEPT } ![r][i] = \text{“processed”}]$
$\land$ UNCHANGED $\langle locks \rangle$

$GetCache(r,\ i) \triangleq$
$\land\ requests[r][i] \quad = \text{“inProxy”}$
$\land\ Cardinality(\{x \in \text{DOMAIN } requests[r] : requests[r][x] = \text{“cached”}\}) = 1$
$\land\ requests' = [requests \text{ EXCEPT } ![r][i] = \text{“fromCache”}]$
$\land$ UNCHANGED $\langle locks \rangle$

$Cache(r,\ i) \triangleq$
$\land\ requests[r][i] = \text{“processed”}$
$\land\ requests' = [requests \text{ EXCEPT } ![r][i] = \text{“cached”}]$
$\land\ locks' = [locks \text{ EXCEPT } ![r] = \text{FALSE}]$

Spec

$Next \triangleq$
$\lor\ \exists\, r \in \_ReqTokens,\ i \in tryKeys :$
$\lor\ HitProxy(r,\ i)$
$\lor\ HitServer(r,\ i)$
$\lor\ Lock(r,\ i)$
$\lor\ Cache(r,\ i)$
$\lor\ GetCache(r,\ i)$

$Fairness \triangleq \forall\, r \in \_ReqTokens,\ i \in tryKeys :$
$\land\ \text{WF}_{vars}(HitServer(r,\ i))$
$\land\ \text{WF}_{vars}(HitProxy(r,\ i))$
$\land\ \text{WF}_{vars}(Lock(r,\ i))$
$\land\ \text{WF}_{vars}(Cache(r,\ i))$
$\land\ \text{WF}_{vars}(GetCache(r,\ i))$

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

$RequestIsProcessedOnlyOnce \triangleq$
$\Box(\forall\, req \in \text{DOMAIN } requests :$
$Cardinality(\{x \in \text{DOMAIN } requests[req] : requests[req][x] \in \{\text{“processed”}, \text{“cached”}\}\}) < 2)$

$EveryReqFinishAsCachedOrFromCache \triangleq$
$\Diamond\Box(\forall\, req \in \text{DOMAIN } requests :$
$\forall\, x \in \text{DOMAIN } requests[req] :$
$(requests[req][x] = \text{“cached”}) \lor (requests[req][x] = \text{“fromCache”}))$

$AttemptsCanBeProcessedConcurrently \triangleq$
$\Box(\forall\, req \in \text{DOMAIN } requests,\ x \in tryKeys : requests[req][x] = \text{“pending”} \Rightarrow \text{ENABLED } HitProxy(req,\ x))$

THEOREM $Spec \Rightarrow RequestIsProcessedOnlyOnce$
THEOREM $Spec \Rightarrow EveryReqFinishAsCachedOrFromCache$
THEOREM $Spec \Rightarrow AttemptsCanBeProcessedConcurrently$