* Authors: *Santhosh Raju* < *santhosh@byisystems.com* >
*   Cherry *G. Mathew* < cherry@*byisystems.com* >
*   Fransisca *Andriani* < *sisca@byisystems.com* >
*

MODULE *VoucherTransfer*

The description is based on the "Transfer" operation mentioned in *RFC* 3506. This specification describes the transfer of Voucher between two Holders. It is implemented over the Two-Phase Commit protocol, in which a Voucher Transaction Provider (*VTP*) coordinates the "Source" Voucher Holders (*SHs*) to trade vouchers (Vs) to "Destination" Voucher Holders (*DHs*) described in the *VoucherLifeCycle* specification module. In this specification, *SHs* and *DHs* spontaneously issue *Prepared* messages. We ignore the Prepare messages that the *VTP* can send to the *SHs* and *DHs*.

For simplicity, we also eliminate *Abort* messages sent by an *SHs* and *DHs* when it decides to abort. Such a message would cause the *VTP* to abort the transaction, an event represented here by the *VTP* spontaneously deciding to abort.

Note: The *RFC* does not differentiate between a Holder who is initiating the transfer (*i.e.* the holder of the voucher) and the Holder who is receiving the voucher (*i.e.* the holder who would be the future owner of this voucher). In order to make this distinction we have the "Source" Voucher Holders (*SHs*), a subset of Holders who would like to transfer an existing voucher they are "holding". We also have the "Destination" Voucher Holders (*DHs*), a subset of Holders who are "waiting" to receive the transferred vouchers.

CONSTANT

V,	The set of Vouchers
SH,	The set of "Source" Voucher Holders
DH	The set of "Destination" Voucher Holders

VARIABLES

vState,	*vState*[*v*] is the state of voucher *v*.
vlcState,	*vlcState*[*v*] is the state of the voucher life cycle machine.
shState,	*shState*[*sh*] is the state of "source" voucher holder *sh*.
dhState,	*dhState*[*dh*] is the state of "destination" voucher holder *dh*.
vtpState,	The state of the voucher transaction provider.
vtpTPrepared,	The set of *SHs* and *DHs* from which the *VTP* has received "Prepared for Voucher *Transfer*" messages.
msgs	

In the protocol, processes communicate with one another by sending messages. For simplicity, we represent message passing with the variable *msgs* whose value is the set of all messages that have been sent. A message is sent by adding it to the set *msgs*. An action that, in an implementation, would be enabled by the receipt of a certain message is here enabled by the presence of that message in *msgs*. For simplicity, messages are never removed from *msgs*. This allows a single message to be received by multiple receivers. Receipt of the same message twice is therefore allowed; but in this particular protocol, that's not a problem.

$Messages \ \triangleq$

The set of all possible messages. Messages of type "Prepared" are sent from the *SH* indicated by the message's *vsh* field to the *VTP*. Similar "Prepared" is also sent from *DH* indicated by message's *vdh* field to the *VTP*. Messages of type "Transfer" and "Abort" are broadcast by the *VTPs*, to be received by all *SHs* and *DHs*. The set *msgs* contains just a single copy of such a message.

$$\begin{aligned}
&[type : \{\text{“Prepared”}\},\ vsh : SH] \cup \\
&[type : \{\text{“Prepared”}\},\ vdh : DH] \cup \\
&[type : \{\text{“Transfer”},\ \text{“Abort”}\}]
\end{aligned}$$

$VTPTypeOK \ \triangleq$

The type-correctness invariant

$$\begin{aligned}
&\land\ vState \in [V \rightarrow \{\text{“valid”}\}] \\
&\land\ vlcState \in [V \rightarrow \{\text{“working”}\}] \\
&\land\ shState \in [SH \rightarrow \{\text{“holding”},\ \text{“prepared”},\ \text{“transferred”},\ \text{“aborted”}\}] \\
&\land\ dhState \in [DH \rightarrow \{\text{“waiting”},\ \text{“prepared”},\ \text{“holding”},\ \text{“aborted”}\}] \\
&\land\ vtpState \in \{\text{“init”},\ \text{“done”}\} \\
&\land\ vtpTPrepared \subseteq (SH \cup DH) \\
&\land\ msgs \subseteq Messages
\end{aligned}$$

$VTPInit \ \triangleq$

The initial predicate.

$$\begin{aligned}
&\land\ vState = [v \in V \mapsto \text{“valid”}] \\
&\land\ vlcState = [v \in V \mapsto \text{“working”}] \\
&\land\ shState = [sh \in SH \mapsto \text{“holding”}] \\
&\land\ dhState = [dh \in DH \mapsto \text{“waiting”}] \\
&\land\ vtpState = \text{“init”} \\
&\land\ vtpTPrepared = \{\} \\
&\land\ msgs = \{\}
\end{aligned}$$

We now define the actions that may be performed by the processes, first the *VTP*'s actions, the SHs' actions, then the DHs' actions.

$VTPRcvPrepared(sh,\ dh) \ \triangleq$

The *VTP* receives a "Prepared" message from Source Voucher Holder *sh* and the Destination Voucher Holder *dh*. We could add the additional enabling condition *sh*,*dh* \not in *vtpTPrepared*, which disables the action if the *VTP* has already received this message. But there is no need, because in that case the action has no effect; it leaves the state unchanged.

$$\begin{aligned}
&\land\ vState = [v \in V \mapsto \text{“valid”}] \\
&\land\ vlcState = [v \in V \mapsto \text{“working”}] \\
&\land\ vtpState = \text{“init”}
\end{aligned}$$

$\land [\mathit{type} \mapsto \text{“Prepared”},\ \mathit{vsh} \mapsto \mathit{sh}] \in \mathit{msgs}$
$\land [\mathit{type} \mapsto \text{“Prepared”},\ \mathit{vdh} \mapsto \mathit{dh}] \in \mathit{msgs}$
$\land \mathit{vtpTPrepared}' = \mathit{vtpTPrepared} \cup \{\mathit{sh},\ \mathit{dh}\}$
$\land \text{UNCHANGED}\ \langle \mathit{vState},\ \mathit{vlcState},\ \mathit{shState},\ \mathit{dhState},\ \mathit{vtpState},\ \mathit{msgs} \rangle$

$\mathit{VTPTransfer}(v) \triangleq$

The *VTP* Transfers the voucher; enabled iff the *VTP* is in its initial state and every *SH* and *DH* has sent a “Prepared” message.

$\land \mathit{vState}[v] = \text{“valid”}$
$\land \mathit{vlcState}[v] = \text{“working”}$
$\land \mathit{vtpState} = \text{“init”}$
$\land \mathit{vtpTPrepared} = \mathit{SH} \cup \mathit{DH}$
$\land \mathit{vtpState}' = \text{“done”}$
$\land \mathit{msgs}' = \mathit{msgs} \cup \{[\mathit{type} \mapsto \text{“Transfer”}]\}$
$\land \text{UNCHANGED}\ \langle \mathit{shState},\ \mathit{dhState},\ \mathit{vState},\ \mathit{vlcState},\ \mathit{vtpTPrepared} \rangle$

$\mathit{VTPAbort}(v) \triangleq$

The *VTP* spontaneously aborts the transaction.

$\land \mathit{vState}[v] = \text{“valid”}$
$\land \mathit{vlcState}[v] = \text{“working”}$
$\land \mathit{vtpState} = \text{“init”}$
$\land \mathit{vtpState}' = \text{“done”}$
$\land \mathit{msgs}' = \mathit{msgs} \cup \{[\mathit{type} \mapsto \text{“Abort”}]\}$
$\land \text{UNCHANGED}\ \langle \mathit{vState},\ \mathit{vlcState},\ \mathit{shState},\ \mathit{dhState},\ \mathit{vtpTPrepared} \rangle$

$\mathit{SHPrepare}(\mathit{sh}) \triangleq$

Source Voucher holder *sh* prepares.

$\land \mathit{vState} = [v \in V \mapsto \text{“valid”}]$
$\land \mathit{vlcState} = [v \in V \mapsto \text{“working”}]$
$\land \mathit{shState}[\mathit{sh}] = \text{“holding”}$
$\land \mathit{shState}' = [\mathit{shState}\ \text{EXCEPT}\ ![\mathit{sh}] = \text{“prepared”}]$
$\land \mathit{msgs}' = \mathit{msgs} \cup \{[\mathit{type} \mapsto \text{“Prepared”},\ \mathit{vsh} \mapsto \mathit{sh}]\}$
$\land \text{UNCHANGED}\ \langle \mathit{vState},\ \mathit{vlcState},\ \mathit{vtpState},\ \mathit{dhState},\ \mathit{vtpTPrepared} \rangle$

$\mathit{SHChooseToAbort}(\mathit{sh}) \triangleq$

Source Voucher holder *sh* spontaneously decides to abort. As noted above, *sh* does not send any message in our simplified spec.

$\land \mathit{vState} = [v \in V \mapsto \text{“valid”}]$
$\land \mathit{vlcState} = [v \in V \mapsto \text{“working”}]$
$\land \mathit{shState}[\mathit{sh}] = \text{“holding”}$
$\land \mathit{shState}' = [\mathit{shState}\ \text{EXCEPT}\ ![\mathit{sh}] = \text{“aborted”}]$
$\land \text{UNCHANGED}\ \langle \mathit{vState},\ \mathit{vlcState},\ \mathit{vtpState},\ \mathit{dhState},\ \mathit{vtpTPrepared},\ \mathit{msgs} \rangle$

$\mathit{SHRcvTransferMsg}(\mathit{sh}) \triangleq$

Source Voucher holder *sh* is told by the *VTP* to *Transfer*.

$$\begin{array}{l}
\land\ vState = [v \in V \mapsto \text{“valid”}] \\
\land\ vlcState = [v \in V \mapsto \text{“working”}] \\
\land\ shState[sh] = \text{“holding”} \\
\land\ [type \mapsto \text{“Transfer”}] \in msgs \\
\land\ shState' = [shState\ \text{EXCEPT}\ ![sh] = \text{“transferred”}] \\
\land\ \text{UNCHANGED}\ \langle vtpState, vlcState, vState, dhState, vtpTPrepared, msgs\rangle
\end{array}$$

$SHRcvAbortMsg(sh) \triangleq$

Source Voucher holder *sh* is told by the *VTP* to abort.

$$\begin{array}{l}
\land\ vState = [v \in V \mapsto \text{“valid”}] \\
\land\ vlcState = [v \in V \mapsto \text{“working”}] \\
\land\ shState[sh] = \text{“holding”} \\
\land\ [type \mapsto \text{“Abort”}] \in msgs \\
\land\ shState' = [shState\ \text{EXCEPT}\ ![sh] = \text{“aborted”}] \\
\land\ \text{UNCHANGED}\ \langle vState, vlcState, vtpState, dhState, vtpTPrepared, msgs\rangle
\end{array}$$

$DHPrepare(dh) \triangleq$

Destination Voucher holder *dh* prepares.

$$\begin{array}{l}
\land\ vState = [v \in V \mapsto \text{“valid”}] \\
\land\ vlcState = [v \in V \mapsto \text{“working”}] \\
\land\ dhState[dh] = \text{“waiting”} \\
\land\ dhState' = [dhState\ \text{EXCEPT}\ ![dh] = \text{“prepared”}] \\
\land\ msgs' = msgs \cup \{[type \mapsto \text{“Prepared”}, vdh \mapsto dh]\} \\
\land\ \text{UNCHANGED}\ \langle vState, vlcState, vtpState, shState, vtpTPrepared\rangle
\end{array}$$

$DHChooseToAbort(dh) \triangleq$

Destination Voucher holder *dh* spontaneously decides to abort. As noted above, *dh* does not send any message in our simplified spec.

$$\begin{array}{l}
\land\ vState = [v \in V \mapsto \text{“valid”}] \\
\land\ vlcState = [v \in V \mapsto \text{“working”}] \\
\land\ dhState[dh] = \text{“waiting”} \\
\land\ dhState' = [dhState\ \text{EXCEPT}\ ![dh] = \text{“aborted”}] \\
\land\ \text{UNCHANGED}\ \langle vState, vlcState, vtpState, shState, vtpTPrepared, msgs\rangle
\end{array}$$

$DHRcvTransferMsg(dh) \triangleq$

Destination Voucher holder *dh* is told by the *VTP* to *Transfer*.

$$\begin{array}{l}
\land\ vState = [v \in V \mapsto \text{“valid”}] \\
\land\ vlcState = [v \in V \mapsto \text{“working”}] \\
\land\ dhState[dh] = \text{“waiting”} \\
\land\ [type \mapsto \text{“Transfer”}] \in msgs \\
\land\ dhState' = [dhState\ \text{EXCEPT}\ ![dh] = \text{“holding”}] \\
\land\ \text{UNCHANGED}\ \langle vtpState, vState, vlcState, shState, vtpTPrepared, msgs\rangle
\end{array}$$

$DHRcvAbortMsg(dh) \triangleq$

Destination Voucher holder *dh* is told by the *VTP* to abort.

$$\begin{aligned}
&\land\ vState = [v \in V \mapsto \text{“valid”}]\\
&\land\ vlcState = [v \in V \mapsto \text{“working”}]\\
&\land\ dhState[dh] = \text{“waiting”}\\
&\land\ [type \mapsto \text{“Abort”}] \in msgs\\
&\land\ dhState' = [dhState \ \textsc{except}\ ![dh] = \text{“aborted”}]\\
&\land\ \textsc{unchanged}\ \langle vState, vlcState, vtpState, shState, vtpTPrepared, msgs\rangle
\end{aligned}$$

$$\begin{aligned}
&VTPNext \triangleq\\
&\quad \lor\ \exists\, v \in V :\\
&\qquad\quad VTPTransfer(v) \lor VTPAbort(v)\\
&\quad \lor\ \exists\, sh, dh \in SH \cup DH :\\
&\qquad\quad VTPRcvPrepared(sh, dh)\\
&\quad \lor\ \exists\, sh \in SH :\\
&\qquad\quad SHPrepare(sh) \lor SHChooseToAbort(sh)\\
&\qquad\quad \lor SHRcvAbortMsg(sh) \lor SHRcvTransferMsg(sh)\\
&\quad \lor\ \exists\, dh \in DH :\\
&\qquad\quad DHPrepare(dh) \lor DHChooseToAbort(dh)\\
&\qquad\quad \lor DHRcvAbortMsg(dh) \lor DHRcvTransferMsg(dh)
\end{aligned}$$

$VTPConsistent \triangleq$

A state predicate asserting that a *SH* and an *DH* have not reached conflicting decisions. It is an invariant of the specification.

$$\land\ \forall\, sh \in SH, dh \in DH : \quad \begin{array}{l} \land \neg \land\ shState[sh] = \text{“transferred”}\\ \qquad \land\ dhState[dh] = \text{“aborted”}\\ \land \neg \land\ shState[sh] = \text{“aborted”}\\ \qquad \land\ dhState[dh] = \text{“holding”} \end{array}$$

$$VTPVars \triangleq \langle shState, dhState, vState, vlcState, vtpState, vtpTPrepared, msgs\rangle$$

$$VTPSpec \triangleq VTPInit \land \Box[VTPNext]_{VTPVars}$$

The complete spec of the a Voucher *Transfer* using Two-Phase Commit protocol.

$$\textsc{theorem}\ VTPSpec \Rightarrow \Box(VTPTypeOK \land VTPConsistent)$$

This theorem asserts the truth of the temporal formula whose meaning is that the state predicate $VTPTypeOK \land VTPConsistent$ is an invariant of the specification *VTPSpec*. Invariance of this conjunction is equivalent to invariance of both of the formulas *VTPTypeOK* and *VTPConsistent*.

We now assert that the Voucher *Transfer* specification implements the Voucher Life Cycle specification of a voucher mentioned in module *VoucherLifeCycle*. The following statement imports all the definitions from module *VoucherLifeCycle* into the current module.

$$\textsc{instance}\ VoucherLifeCycle$$

$$\textsc{theorem}\ VTPSpec \Rightarrow VSpec$$

This theorem asserts that the specification *VTPSpec* of the Two-Phase Commit protocol implements the specification *VSpec* of the Voucher life cycle specification.

* Modification History

\ * Last modified *Tue Jun* 12 13:15:55 *IST* 2018 by Fox
\ * Created *Fri Mar* 16 17:45:37 *SGT* 2018 by Fox