MODULE ACP_SB

Time-stamp: < 10 *Jun* 2002 at 12:39:50 by charpov on *berlioz.cs.unh.edu* >

Atomic Committment Protocol with Simple Broadcast primitive (ACP-SB)
From:
Sape Mullender , editor. Distributed Systems.
Chapter 6: Non-Blocking Atomic Commitment, by Ö. Babaoğlu and S. Toueg.
1993.

Synchronous communication has been replaced with (implicit) asynchronous communication. Failures are detected "magically" instead or relying on timeouts.

This version of the protocol uses a "simple broadcast" where a broadcast is simply a series of messages sent, possibly interrupted by a failure. Consequently, this algorithm is "non terminating" and property $AC5$ does not hold.

CONSTANTS

participants,	set of participants
yes, *no*,	vote
undecided, *commit*, *abort*,	decision
waiting,	coordinator state wrt a participant
notsent	broadcast state wrt a participant

VARIABLES

participant,	participants (N)
coordinator	coordinator (1)

$$
\begin{array}{l}
TypeInvParticipant \triangleq participant \in [\\
\quad participants \rightarrow [\\
\quad\quad vote : \{yes, no\}, \\
\quad\quad alive : \text{BOOLEAN}, \\
\quad\quad decision : \{undecided, commit, abort\}, \\
\quad\quad faulty : \text{BOOLEAN}, \\
\quad\quad voteSent : \text{BOOLEAN} \\
\quad] \\
]
\end{array}
$$

$$
\begin{array}{l}
TypeInvCoordinator \triangleq coordinator \in [\\
\quad request : [participants \rightarrow \text{BOOLEAN}], \\
\quad vote : [participants \rightarrow \{waiting, yes, no\}], \\
\quad broadcast : [participants \rightarrow \{commit, abort, notsent\}], \\
\quad decision : \{commit, abort, undecided\}, \\
\quad alive : \text{BOOLEAN}, \\
\quad faulty : \text{BOOLEAN} \\
]
\end{array}
$$

$$TypeInv \triangleq TypeInvParticipant \land TypeInvCoordinator$$

Initially:
All the participants:
 have a yes/no vote
 are alive and not faulty
 have not sent in their votes yet
 are undecided about final decision

The coordinator:
 has not sent vote requests yet
 has not recieved votes from any participant
 is alive and not faulty
 has not sent broadcast messages to any participant
 is undecided about final decision

$$
\begin{aligned}
InitParticipant \triangleq\ & participant \in [\\
& \quad participants \rightarrow [\\
& \quad\quad vote : \{yes,\ no\},\\
& \quad\quad alive : \{\text{TRUE}\},\\
& \quad\quad decision : \{undecided\},\\
& \quad\quad faulty : \{\text{FALSE}\},\\
& \quad\quad voteSent : \{\text{FALSE}\}\\
& \quad]\\
&]
\end{aligned}
$$

$$
\begin{aligned}
InitCoordinator \triangleq\ & coordinator \in [\\
& \quad request : [participants \rightarrow \{\text{FALSE}\}],\\
& \quad vote : [participants \rightarrow \{waiting\}],\\
& \quad alive : \{\text{TRUE}\},\\
& \quad broadcast : [participants \rightarrow \{notsent\}],\\
& \quad decision : \{undecided\},\\
& \quad faulty : \{\text{FALSE}\}\\
&]
\end{aligned}
$$

$$Init \triangleq InitParticipant \land InitCoordinator$$

COORDINATOR STATEMENTS

request(i):
IF
coordinator is alive
request for vote has not been sent to participant i
THEN
request for vote is sent to participant i

$$
\begin{aligned}
request(i) \triangleq\ & \land\ coordinator.alive\\
& \land\ \neg coordinator.request[i]\\
& \land\ coordinator' = [coordinator\ \text{EXCEPT}\ !.request =\\
& \quad\quad [@\ \text{EXCEPT}\ ![i] = \text{TRUE}]\\
& \quad]\\
& \land\ \text{UNCHANGED}\ \langle participant \rangle
\end{aligned}
$$

getVote(i):
IF
coordinator is alive
coordinator is still undecided
coordinator has sent request for votes to all participants
coordinator is waiting to receive a vote from participant i
participant i has sent the vote message
THEN
the coordinator can record the vote of participant i

$$
\begin{aligned}
getVote(i) \triangleq\ & \land\ coordinator.alive\\
& \land\ coordinator.decision = undecided\\
& \land\ \forall j \in participants : coordinator.request[j]\\
& \land\ coordinator.vote[i] = waiting\\
& \land\ participant[i].voteSent\\
& \land\ coordinator' = [coordinator\ \text{EXCEPT}\ !.vote =\\
& \quad\quad [@\ \text{EXCEPT}\ ![i] = participant[i].vote]\\
& \quad]\\
& \land\ \text{UNCHANGED}\ \langle participant \rangle
\end{aligned}
$$

detectFault(i):
IF
coordinator is alive

coordinator is still undecided
coordinator has sent request for votes to all participants
coordinator is waiting for vote from participant i
participant i has died without sending its vote
THEN
coordinator times out on participant i and decides to abort

$$
\begin{aligned}
\mathit{detectFault}(i) \triangleq\ & \land\ \mathit{coordinator.alive} \\
& \land\ \mathit{coordinator.decision} = \mathit{undecided} \\
& \land\ \forall j \in \mathit{participants} : \mathit{coordinator.request}[j] \\
& \land\ \mathit{coordinator.vote}[i] = \mathit{waiting} \\
& \land\ \neg \mathit{participant}[i].\mathit{alive} \\
& \land\ \neg \mathit{participant}[i].\mathit{voteSent} \\
& \land\ \mathit{coordinator}' = [\mathit{coordinator}\ \text{EXCEPT}\ !.\mathit{decision} = \mathit{abort}] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{participant} \rangle
\end{aligned}
$$

makeDecision:
IF
coordinator is alive
coordinator is undecided
coordinator has received votes from all participants
THEN
IF
all votes are yes
THEN
coordinator decides commit
ELSE
coordinator decides abort

$$
\begin{aligned}
\mathit{makeDecision} \triangleq\ & \land\ \mathit{coordinator.alive} \\
& \land\ \mathit{coordinator.decision} = \mathit{undecided} \\
& \land\ \forall j \in \mathit{participants} : \mathit{coordinator.vote}[j] \in \{\mathit{yes}, \mathit{no}\} \\
& \land\ \lor\ \land\ \forall j \in \mathit{participants} : \mathit{coordinator.vote}[j] = \mathit{yes} \\
& \qquad\ \ \land\ \mathit{coordinator}' = [\mathit{coordinator}\ \text{EXCEPT}\ !.\mathit{decision} = \mathit{commit}] \\
& \quad\ \lor\ \land\ \exists j \in \mathit{participants} : \mathit{coordinator.vote}[j] = \mathit{no} \\
& \qquad\ \ \land\ \mathit{coordinator}' = [\mathit{coordinator}\ \text{EXCEPT}\ !.\mathit{decision} = \mathit{abort}] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{participant} \rangle
\end{aligned}
$$

coordBroadcast(i) (simple broadcast):
IF
coordinator is alive
coordinator has made a decision
coordinator has not sent the decision to participant i
THEN
coordinator sends its decision to participant i

$$
\begin{aligned}
\mathit{coordBroadcast}(i) \triangleq\ & \land\ \mathit{coordinator.alive} \\
& \land\ \mathit{coordinator.decision} \neq \mathit{undecided} \\
& \land\ \mathit{coordinator.broadcast}[i] = \mathit{notsent} \\
& \land\ \mathit{coordinator}' = [\mathit{coordinator}\ \text{EXCEPT}\ !.\mathit{broadcast} = \\
& \qquad [@\ \text{EXCEPT}\ ![i] = \mathit{coordinator.decision}] \\
& \quad\ ] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{participant} \rangle
\end{aligned}
$$

coordDie:
IF
coordinator is alive
THEN
coordinator dies
coordinator is now faulty

$$
\begin{aligned}
coordDie \triangleq\ & \land\ coordinator.alive \\
& \land\ coordinator' = [coordinator \text{ EXCEPT } !.alive = \text{FALSE}, !.faulty = \text{TRUE}] \\
& \land\ \text{UNCHANGED } \langle participant \rangle
\end{aligned}
$$

PARTICIPANT STATEMENTS

sendVote(i):
IF
participant is alive
participant has received a request for vote
THEN
participant sends vote

$$
\begin{aligned}
sendVote(i) \triangleq\ & \land\ participant[i].alive \\
& \land\ coordinator.request[i] \\
& \land\ participant' = [participant \text{ EXCEPT } ![i] = \\
& \qquad [@ \text{ EXCEPT } !.voteSent = \text{TRUE}] \\
& \quad] \\
& \land\ \text{UNCHANGED } \langle coordinator \rangle
\end{aligned}
$$

abortOnVote(i):
IF
participant is alive
participant is undecided
participant has sent its vote to the coordinator
participant's vote is no
THEN
participant decides (unilaterally) to abort

$$
\begin{aligned}
abortOnVote(i) \triangleq\ & \land\ participant[i].alive \\
& \land\ participant[i].decision = undecided \\
& \land\ participant[i].voteSent \\
& \land\ participant[i].vote = no \\
& \land\ participant' = [participant \text{ EXCEPT } ![i] = \\
& \qquad [@ \text{ EXCEPT } !.decision = abort] \\
& \quad] \\
& \land\ \text{UNCHANGED } \langle coordinator \rangle
\end{aligned}
$$

abortOnTimeoutRequest(i):
IF
participant is alive
participant is still undecided
coordinator has died without sending request for vote
THEN
participant decides (unilaterally) to abort

$$
\begin{aligned}
abortOnTimeoutRequest(i) \triangleq\ & \land\ participant[i].alive \\
& \land\ participant[i].decision = undecided \\
& \land\ \neg coordinator.alive \\
& \land\ \neg coordinator.request[i] \\
& \land\ participant' = [participant \text{ EXCEPT } ![i] = \\
& \qquad [@ \text{ EXCEPT } !.decision = abort] \\
& \quad] \\
& \land\ \text{UNCHANGED } \langle coordinator \rangle
\end{aligned}
$$

decide(i):
IF
participant is alive
participant is undecided

participant has recieved decision from the coordinator
THEN
participant decides according to decision from coordinator

$$
\begin{aligned}
\mathit{decide}(i) \triangleq\ & \land\ \mathit{participant}[i].\mathit{alive} \\
& \land\ \mathit{participant}[i].\mathit{decision} = \mathit{undecided} \\
& \land\ \mathit{coordinator}.\mathit{broadcast}[i] \neq \mathit{notsent} \\
& \land\ \mathit{participant}' = [\mathit{participant}\ \text{EXCEPT}\ ![i] = \\
& \qquad [@\ \text{EXCEPT}\ !.\mathit{decision} = \mathit{coordinator}.\mathit{broadcast}[i]] \\
& \quad] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{coordinator} \rangle
\end{aligned}
$$

$\mathit{parDie}(i)$:
IF
participant is alive
THEN
participant dies and is now faulty

$$
\begin{aligned}
\mathit{parDie}(i) \triangleq\ & \land\ \mathit{participant}[i].\mathit{alive} \\
& \land\ \mathit{participant}' = [\mathit{participant}\ \text{EXCEPT}\ ![i] = \\
& \qquad [@\ \text{EXCEPT}\ !.\mathit{alive} = \text{FALSE},\ !.\mathit{faulty} = \text{TRUE}] \\
& \quad] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{coordinator} \rangle
\end{aligned}
$$

FOR N PARTICIPANTS

$$\mathit{parProg}(i) \triangleq \mathit{sendVote}(i) \lor \mathit{abortOnVote}(i) \lor \mathit{abortOnTimeoutRequest}(i) \lor \mathit{decide}(i)$$

$$\mathit{parProgN} \triangleq \exists\, i \in \mathit{participants} : \mathit{parDie}(i) \lor \mathit{parProg}(i)$$

$$\mathit{coordProgA}(i) \triangleq \mathit{request}(i) \lor \mathit{getVote}(i) \lor \mathit{detectFault}(i) \lor \mathit{coordBroadcast}(i)$$

$$\mathit{coordProgB} \triangleq \mathit{makeDecision} \lor \exists\, i \in \mathit{participants} : \mathit{coordProgA}(i)$$

$$\mathit{coordProgN} \triangleq \mathit{coordDie} \lor \mathit{coordProgB}$$

$$\mathit{progN} \triangleq \mathit{parProgN} \lor \mathit{coordProgN}$$

Death transitions are left outside of fairness

$$
\begin{aligned}
\mathit{fairness} \triangleq\ & \land\ \forall\, i \in \mathit{participants} : \mathrm{WF}_{\langle \mathit{coordinator},\, \mathit{participant} \rangle}(\mathit{parProg}(i)) \\
& \land\ \mathrm{WF}_{\langle \mathit{coordinator},\, \mathit{participant} \rangle}(\mathit{coordProgB})
\end{aligned}
$$

$$\mathit{Spec} \triangleq \mathit{Init} \land \Box[\mathit{progN}]_{\langle \mathit{coordinator},\, \mathit{participant} \rangle} \land \mathit{fairness}$$

CORRECTNESS SPECIFICATION

This specification follows the original paper, except that $AC3$ is stronger: It forces participants to abort if one vote at least is NO (in the absence of failure).

The specification is split between safety and liveness.

SAFETY

All participants that decide reach the same decision

$$
\begin{aligned}
AC1 \triangleq\ & \Box \forall\, i, j \in \mathit{participants} : \\
& \quad \lor\ \mathit{participant}[i].\mathit{decision} \neq \mathit{commit} \\
& \quad \lor\ \mathit{participant}[j].\mathit{decision} \neq \mathit{abort}
\end{aligned}
$$

If any participant decides commit, then all participants must have votes YES

$$AC2 \triangleq \Box((\exists i \in participants : participant[i].decision = commit) \Rightarrow (\forall j \in participants : participant[j].vote = yes))$$

If any participant decides abort, then:
at least one participant voted NO, or
at least one participant is faulty, or
coordinator is faulty

$$AC3\_1 \triangleq \Box((\exists i \in participants : participant[i].decision = abort) \Rightarrow \lor (\exists j \in participants : participant[j].vote = no) \lor (\exists j \in participants : participant[j].faulty) \lor coordinator.faulty)$$

Each participant decides at most once

$$AC4 \triangleq \Box \land (\forall i \in participants : participant[i].decision = commit \Rightarrow \Box(participant[i].decision = commit)) \land (\forall j \in participants : participant[j].decision = abort \Rightarrow \Box(participant[j].decision = abort))$$

LIVENESS
(stronger for $AC3$ than in the original paper)

$$AC3\_2 \triangleq \Diamond \lor \forall i \in participants : participant[i].decision \in \{abort, commit\} \lor \exists j \in participants : participant[j].faulty \lor coordinator.faulty$$

(SOME) INTERMEDIATE PROPERTIES USED IN PROOFS

$$FaultyStable \triangleq \land \forall i \in participants : \Box(participant[i].faulty \Rightarrow \Box participant[i].faulty) \land \Box(coordinator.faulty \Rightarrow \Box coordinator.faulty)$$

$$VoteStable \triangleq \forall i \in participants : \lor \Box(participant[i].vote = yes) \lor \Box(participant[i].vote = no)$$

$$StrongerAC2 \triangleq \Box((\exists i \in participants : participant[i].decision = commit) \Rightarrow \land (\forall j \in participants : participant[j].vote = yes) \land coordinator.decision = commit)$$

$$StrongerAC3\_1 \triangleq \Box((\exists i \in participants : participant[i].decision = abort) \Rightarrow \lor (\exists j \in participants : participant[j].vote = no) \lor \land \exists j \in participants : participant[j].faulty \land coordinator.decision = abort \lor \land coordinator.faulty \land coordinator.decision = undecided)$$

($AC1$ follows from $StrongerAC2 \land StrongerAC3\_1$)

$$NoRecovery \triangleq \Box \land \forall i \in participants : participant[i].alive \equiv \neg participant[i].faulty \land coordinator.alive \equiv \neg coordinator.faulty$$

(SOME) INVALID PROPERTIES

$$DecisionReachedNoFault \triangleq (\forall i \in participants : participant[i].alive) \leadsto (\forall k \in participants : participant[k].decision \neq undecided)$$

$$AbortImpliesNoVote \triangleq \Box((\exists i \in participants : participant[i].decision = abort) \Rightarrow (\exists j \in participants : participant[j].vote = no))$$

The following is the termination property that this SB algorithm doesn't have

$$AC5 \triangleq \Diamond \forall\, i \in participants : \begin{array}{l} \lor\ participant[i].decision \in \{abort,\ commit\} \\ \lor\ participant[i].faulty \end{array}$$