Asynchronous Multi-Agent ASMs with real-time constraints

Processor-Group Membership Protocol Specification and Verification

Egon Börger

Dipartimento di Informatica, Universita di Pisa
http://www.di.unipi.it/~boerger
For details see Chapter 6.3 (Time-Constrained Asynchronous ASMs) of:

**E. Börger, R. Stärk**

*Abstract State Machines*

*A Method for High-Level System Design and Analysis*

Springer-Verlag 2003

For update info see AsmBook web page:

http://www.di.unipi.it/AsmBook
Group Membership Problem

• The Context. Fault tolerance for distributed computing services can be provided by
  – letting members of a server group cooperate to provide the services
  – replicating the relevant service state info at all sites

• Group Membership Protocols ensure that despite information propagation delays and server failures, the service state info stored at each group member
  – remains up-to-date
  – is the same for all group members (in the steady state)

• Special Case Processor-Group Membership: via message exchange a global agreement has to be achieved about the set of all correctly functioning processors in the system
The Processor-Group Members

• **PROCESSOR**: finite static domain of processors \( p \) with
  – **Clock\( (p) \)** : REAL strictly increasing monitored fct
  – **Status\( (p) \)** : \{sober, crashed, recovered\} controlled fct
  – **CurProc\( (p) \)** the currently running process, e.g. MS\( (p) \), BS\( (p) \) below, executing tasks from deadline to completion with earliest-deadline-first scheduling
    – **Dline\( (x) \)** = minimum of deadlines of tasks to be handled by \( x \)
  – **Membership Server MS\( (p) \)**
    • handling entry of \( p \) into a new processor-group after recovery from a crash: **Initialize\( (p) \)**, including broadcast of a new-gp msg
    • maintaining processor group info - **Group\( (p) \)** and membership view **Members\( (p) \)** - while \( p \) is alive, by handling incoming
      – messages announcing the constitution of a **new group**
      – messages reporting which processors are **present** in a group
  – **Broadcast Server BS\( (p) \)** sending with period \( d_h \) (**heartbeat**) broadcast msgs to all processors
Fault tolerance lifecycle of a processor $p$

Two reasons for crashing (processor interrupt): $p$ missed the deadline of a broadcast: $BS(p)$ was scheduled too late
- new-group msg because $MS(p)$ was scheduled too late to handle it

Clock($p$) $>$ BCastTime($p$)
Clock($p$) $>$ Timestamp(CurMsg($p$))
Performance Failures

- A **Performance Failure** occurs at p if p is sober but has a pending task whose deadline has been exceeded
  - otherwise p is called **correct**

- It is assumed that performance failures can be detected and turned into crashes
Messages

- **MESSAGE** = MSGTYPE x REAL x 2^PROCESSOR
- For m = (x,y,z) let
  
  Type(m) = x, Timestamp(m) = y, View(m) = z

  Deadline(m) =

  - Timestamp(m) if Type(m) = new-gp
  - Timestamp(m) + d_n if Type(m) = present

new-group msgs are scheduled with delay d_n >
**maximum network propagation delay** so that every
correctly functioning processor has a chance to see them

- Timestamp(m) records the time the group was created by Initialize(p). It is used here as group Id.
- View(m) represents the set of all members of processor group Timestamp(m).
Messages (Cont’d)

- **CurMsg(p)** = the msg currently seen by MS(p). Since CurMsg(p) should always store the msg with earliest deadline, set $Dline(MS(p)) = \text{Deadline}(\text{CurMsg}(p))$

- **InBox(p)** $\subseteq \text{MESSAGE}$ : a controlled set containing the msgs delivered to p from the network, via a monitored fct $\text{InMsg}(p)$, but not yet seen, as CurMsg(p), by MS(p)

- **AptMsg(p)**
  
  = relevant InBox(p)-msgs with Deadline $\geq \text{Clock}(p) - d_u$
  
  = \{m \in \text{InBox}(p) | \text{Deadline}(m) \geq \text{Clock}(p) - d_u \land \text{Timestamp}(m) \geq \text{StartUpTime}(p)\}

where scheduling uncertainty $d_u$ : a task with deadline d may not be scheduled until d - $d_u$

$\text{StartUpTime}(p)$ = recovery time of p from last crash
Message Passing System

• Message Passing System modeled by abstract propagation of messages to the network & two intermediary agents:
  - **Broadcast** $m$, say by setting $\text{InTransit}(m):=\text{true}$ for some predicate $\text{InTransit}$ over messages, shared by all $\text{MsgCarrier}(p)$
  - $\text{MsgCarrier}(p)$ delivers to $\text{InBox}(p)$ every message $\text{InMsg}(p)$ which is incoming from a Broadcast (i.e. from $\text{InTransit}$)
  - $\text{Custodian}(p)$ delivers from the subset $\text{AptMsg}(p)$ of $\text{InBox}(p)$ a message with the minimal timestamp to $\text{MS}(p)$, namely as $\text{CurMsg}(p)$

• **Reliability Assumption**: every message which has been broadcast by a processor will be delivered to every processor

• **Delivery Bound** relating different processor clocks: a msg sent (which in this protocol means broadcast) by $p$ at $\text{Clock}(p)$-time $t_1$ will be received by $q$ at $\text{Clock}(q)$-time $t_2$ within a carry delay $d_c$

  $0 < t_2 - t_1 \leq d_c$
Scheduling Requirements

- **earliest-deadline-first:**
  - Scheduler\((p)\) provides for the next process CurProc\((p)\) the enabled process among MS\((p)\) or BS\((p)\) with minimal deadline
  - Custodian\((p)\) provides for CurMsg\((p)\) always a message with the earliest deadline

- **non-preemptive:**
  - Scheduler\((p)\) runs only when CurProc\((p)\) has been switched from MS\((p)\) or BS\((p)\), after their completion, to undef. This realizes the requirement that only processors can be interrupted, whereas tasks run until their completion.

- **scheduling uncertainty:** a task with deadline \(d\) may not be scheduled until \(d-d_u\) where \(d_u\) is the system wide upper uncertainty bound. Therefore processes on \(p\) are called enabled only when their deadline is \(\geq\)Clock\((p)\)- \(d_u\):

  \[
  \text{EnabledBS}(p) \iff \text{DlineBS}(p) \geq \text{Clock}(p) - d_u \\
  \text{Dline}(\text{BS}(p)) = \text{BCastTime}(p)
  \]

  \[
  \text{EnabledMS}(p) \iff \text{CurMsg}(p) \neq \text{undef} \& \text{DlineMS}(p) \geq \text{Clock}(p) - d_u
  \]
Agents of the processor-group membership protocol

- **Five subagents** for each processor p
  - **Scheduler(p)** assigns as next CurProc(p) the enabled process among MS(p) or BS(p) with minimal deadline, including scheduling MS(p) upon crash recovery of p
    - a lower recovery bound $d_r$ is assumed for the time bw a crash and a subsequent recovery
  - **Membership Server MS(p)**
    - initializes p upon recovery
    - maintains processor group info, reacting to CurMsg(p) by MS(p)-crash, or synchronizing p with new-gp, or updating Members(p)
  - **Broadcast Server BS(p)** periodically broadcasts presence msgs for p & detects broadcast crashes at p
  - **MsgCarrier(p)** collects for p msgs incoming from the network, adding them to InBox(p)
  - **Custodian(p)** delivers a msg with earliest deadline from InBox(p) to CurMsg(p) of MS(p)
6 Goals of the processor-group membership protocol

- **Stability of local views.** Once a processor joins a group \( g \), it stays in \( g \) until either a processor fails or one recovers and attempts to rejoin.

- **Agreement on history.** If two processors are joined to a common group \( g \) and none of them crashes between joining \( g \) and joining the next group, then that next group is the same for both processors.

- **Agreement on group membership.** Alive processors in a group have the same membership view.

- **Reflexivity.** The membership view of an alive processor in a group contains that processor.

- **Bounded join delay.** There is a constant join delay \( d_j \) s.t. if a processor becomes sober at time \( t \) then, by time \( t + d_j \), it will join a new group along with every other processor that stays correct in the time interval \([t, t + d_j]\).

- **Bounded failure detection delay.** There is a constant failure delay \( d_f \) s.t. if a processor belonging to a group \( g \) fails at time \( t \) then, by time \( t + d_f \), all the members of \( g \) that stay correct in \([t, t + d_f]\) will join a new group \( g' \) not containing \( p \).
Teaming member subagents for processor-group membership Protocol Runs

- We assume the **five subagents** for each processor $p$ to be **teamed as a basic ASM** $M(p)$
  - e.g. choosing non-deterministically for each move of $M(p)$ one enabled subagent to fire, defining $M(p)$ as
    choose $R \in \{\text{Scheduler}(p), \text{MS}(p), \text{BS}(p), \text{MsgCarrier}(p), \text{Custodian}(p)\}$ in $R$

- The choice is bw one of Scheduler, MS, BS – which have disjoint guards - and one of MsgCarrier, Custodian.
  - This avoids possibly conflicting updates of CurMsg (by MS and Custodian) and of InBox (by MsgCarrier and Custodian)
Protocol Runs for processor-group membership

• The monitored fcts of $M(p)$ are $\text{Clock}(p)$, $\text{InMsg}(p)$, all other dynamic fcts are controlled or derived

• Initial states: $\text{Status}(p)=\text{crashed}$, $\text{InBox}(p)=\emptyset$, all of $\text{CurMsg}(p)$, $\text{Group}(p)$, $\text{Members}(p)$, $\text{StartUpTime}(p)$, $\text{BCastTime}(p)$ have value undef

• Protocol Runs are defined as multi-agent asynchronous runs of the basic machines $M(p)$, $p \in \text{PROCESSOR}$, which (are assumed to) respect
  – the lower $\text{Recovery Bound } d_r$ on recovery time
  – the upper $\text{Delivery Bound } d_c$ on message delivery time over the network

  • These constraints on runs abstract from explicit lower/upper time bounds one could incorporate into the related rules of $M(p)$

• NB. Passing of time can be taken as reflecting not furthermore specified moves of other processes
Scheduler(p) rules assigning values to CurProc(p)

- **Recover(p) ≡**
  upon recovery of p, MS(p) has to initialize p
  If Status(p) = crashed then Status(p) := recovered
  CurProc(p) := MS(p)

- **Schedule(p) ≡**
  scheduling protocol processes MS(p), BS(p)
  If CurProc(p) = undef & Status(p) = sober
  & (AptMsg(p) = ∅ or CurMsg(p) ≠ undef )
  then CurProc(p) := MinEnabled(p)
  earliest-deadline-first
  where **MinEnabled(p) =**
  \[
  \begin{align*}
  x & \text{ if } (\text{Enabledx} \& \text{not Enabledy}) \\
  & \text{or } (\text{Enabledx} \& \text{Enabledy} \& \text{Dline}(x) < \text{Dline}(y)) \\
  y & \text{ if } (\text{Enabledx} \& \text{Enabledy} \& \text{Dline}(x) \geq \text{Dline}(y))
  \end{align*}
  \]
  for \((x,y) \in \{(\text{MS}(p),\text{BS}(p)), (\text{BS}(p),\text{MS}(p))\}\)

NB. CurMsg(p) is defined when via the Custodian(p)-rule below it has been set to a msg from InBox(p) with minimal deadline
MS(p)-Initial rule to initialize recovered processors

• **MS(p)-Initialize**
  If \( \text{Status}(p) = \text{recovered} \) & \( \text{CurProc}(p) = \text{MS}(p) \)
  then
  Reset\( (p) \)
  ResetGroupInfo\( (p) \)
  StartUpTime\( (p) := \text{Clock}(p) + d_n \)
  BroadcastNewGroup\( (p) \)
  Status\( (p) := \text{sober} \)
  CurProc done

• **Reset\( (p) \)**
  \( B\text{CastTime}(p) := \text{undef}, \text{CurMsg}(p) := \text{undef} \)

• **ResetGroupInfo\( (p) \)**
  \( \text{Group}(p) := \text{undef}, \text{Members}(p) := \text{undef} \)

• **BroadcastNewGroup\( (p) \)**
  Broadcast \((\text{new-gp}, \text{Clock}(p) + d_n, \{p}\})

• **CurProc done**
  \( \text{CurProc}(p) := \text{undef} \)
\textbf{MS}(p)-\textbf{Crash rule}

handling miss of a new-gp msg

\begin{itemize}
  \item \textbf{MS}(p)-\textbf{Crash} $\equiv$
  \begin{itemize}
    \item \( p \) crashes upon deadline miss of a new-gp msg
  \end{itemize}
  \begin{itemize}
    \item If \( \text{Status}(p) = \text{sober} \) \& \( \text{CurProc}(p) = \text{MS}(p) \) \& \( \text{Type}(\text{CurMsg}(p)) = \text{new-gp} \)
    \item then If \( \text{Clock}(p) > \text{Timestamp}(\text{CurMsg}(p)) \)
    \item then \( \text{Status}(p) := \text{crashed} \)
  \end{itemize}
\end{itemize}
MS(p)-NewGroup rule for timely handling of new-gp msg

MS(p)-NewGroup \equiv
\begin{align*}
\text{If } & \text{Status}(p) = \text{sober} \land \text{CurProc}(p) = \text{MS}(p) \land \\
& \text{Type}(\text{CurMsg}(p)) = \text{new-gp} \\
\text{then } & \text{If } \text{Clock}(p) \leq \text{Timestamp}(\text{CurMsg}(p)) \text{ then} \\
& \text{BroadcastRegistration}(p) \\
& \text{SynchronizeWithNewGp}(p) \\
& \text{CurMsg done} \\
& \text{CurProc done}
\end{align*}

\text{BroadcastRegistration}(p) \equiv \text{reporting } p \text{ as present in new-gp}
\text{Broadcast} \left( \text{present}, \text{Timestamp}(\text{CurMsg}(p)), \{p\} \right)

\text{SynchronizeWithNewGp}(p) \equiv
\text{BCastTime}(p) := \text{Timestamp}(\text{CurMsg}(p)) + d_h \\
\text{synchronize heartbeat of } p \text{ to new-gp canceling any pending heartbeat}

\text{CurMsg done} \equiv \text{CurMsg}(p) := \text{undef}
**MS(p)-ChangeGroup** rule handling arrival of presence reports

**MS(p)-ChangeGroup** ≡ If \( \text{Status}(p) = \text{sober} \) & \( \text{CurProc}(p) = \text{MS}(p) \) & type(\( \text{CurMsg}(p) \)) = \text{present} 
then \( \text{CurMsg} \) done 
CurProc done

If \( \text{Members}(p) \neq \text{View}(\text{CurMsg}(p)) \)
then UpdateGroupMembership(p)

View(m) yields the processors reported as present in m’s group

NB. Groups have as identifiers the timestamp of their creation
here Timestamp(m)

No group change if p’s group has not changed

**UpdateGroupMembership**(p) ≡ change to new group

Members(p) := View(\( \text{CurMsg}(p) \))
Group(p) := Timestamp(\( \text{CurMsg}(p) \))
BS(p)-Crash rule
handling miss of a heartbeat

• BS(p)-Crash  \equiv
  p \text{ crashes upon deadline miss of a broadcast}
  \text{If } \text{Status}(p) = \text{sober} \& \text{CurProc}(p) = \text{BS}(p)
  \text{then } \text{If } \text{Clock}(p) > \text{BCastTime}(p)
  \text{then } \text{Status}(p) := \text{crashed}
BS(p)-Presence rule for timely handling of heartbeats

BS(p)-Presence $\equiv$ within broadcast time, p reports its presence & updates its next heartbeat deadline

If Status(p) = sober & CurProc(p) = BS(p)
then If $\text{Clock}(p) \leq \text{BCastTime}(p)$ then
  BroadcastPresence(p)
  $\text{BCastTime}(p) := \text{BCastTime}(p) + d_h$
CurProc done

BroadcastPresence(p) $\equiv$ Broadcast (present, BCastTime(p), {p}) reporting p as present in its group BCastTime(p)
**MsgCarrier(p) rule** transfering incoming msgs to InBox(p)

\[ MsgCarrier(p) \equiv \]

If InMsg(p) \neq \text{undef} then

Let \((a,b,c) = \text{InMsg(p)}\) in

\begin{align*}
\text{If } a &= \text{new-gp} \text{ then} & \text{add new-gp msg} \\
\text{InBox(p):} &= \text{InBox(p)} \cup \{\text{InMsg(p)}\} \\
\text{elseif InBox(p) has no msg with Timestamp b then} & \\
\text{InBox(p):} &= \text{InBox(p)} \cup \{\text{InMsg(p)}\} \\
& \hspace{1cm} \text{add presence msg from newly constituted group} \\
\text{else let } m &= \exists x(x \in \text{InBox(p): Timestamp(x)=b}) \text{ in} & \\
\text{InBox(p):} &= (\text{InBox(p)} \setminus \{m\}) \cup \{(a,b,\text{View(m)} \cup \{c\})\} \\
& \hspace{1cm} \text{bundle msgs into one per group (i.e. with same timestamp)}
\end{align*}
Custodian\( (p) \) rule assigning CurMsg\( (p) \) to MS\( (p) \)

\[
\text{Custodian}(p) \equiv \\
\text{If Status}(p) = \text{sober} \& \text{CurMsg}(p) = \text{undef} \& \\
\text{AptMsg}(p) \text{ contains a msg with minimal deadline} \\
\text{then deliver } m \text{ from InBox}(p) \text{ to MS}(p) \\
\text{where } m = \varepsilon x \ ( x \text{ has minimal deadline in AptMsg}(p) ) \\
\text{deliver } m \text{ from InBox}(p) \text{ to MS}(p) \equiv \\
\text{InBox}(p):= \text{InBox}(p)-\{m\} \\
\text{CurMsg}(p):= m
\]

NB. This rule realizes the protocol requirement that the membership servers see the msgs in their deadline order
Notations for States of Protocol Runs

- $\Sigma(p)$: the signature of $M(p)$ without InTransit
- $\Sigma^-(p) = \Sigma(p)$ without monitored $\text{Clock}(p)$, $\text{InMsg}(p)$
- For states $S$ of $p$ (really of $\Sigma(p)$) let $S^-$ be its restr to $\Sigma^-(p)$
- $S_n$: the $n$-th state of $p$, $t_n$: the value of $t$ in state $S_n$
- $d_r$-recovery bound & $d_c$-delivery bound constraints on protocol runs can be made precise as follows, for all $p,q$:
  - If $p$ gets crashed in $S_n$ and recovered in $S_{n+k}$ for some $k>0$, then $\text{Clock}(p)_{n+k} - \text{Clock}(p)_n \geq d_r$
  - If $p$ sends msg=$(\text{type},\text{time},\text{view})$ to $q$ (i.e. sets $\text{InTransit}(\text{msg})$ to true) in state $S_n$ of $p$, then there is a unique state $S_k$ of $q$ such that
    - $\text{InMsg}(q)_k = \text{msg}$
    - $\text{InBox}(q)_{k+1}$ contains some $\langle \text{type},\text{time},\text{view}' \rangle$ with $\text{view}' \supseteq \text{view}$
    - $0 < \text{Clock}(q)_k - \text{Clock}(p)_n \leq d_c$
Correctness of processors in Protocol Runs

• Correct(p) in state $S_n$ iff no pending task has exceeded deadline:
  - $\text{Clock}(p)_n \geq \text{StartUpTime}(p)_n$
  - $\forall m \in \text{InBox}(q)_n : \text{if } \text{Timestamp}(m) \geq \text{StartUpTime}(p)_n \text{ then } \text{Deadline}(m) \geq \text{Clock}(p)_n$
  - $\text{BCastTime}(p)_n \geq \text{Clock}(p)_n$

• Correct(p) in real-time interval $[t,t']$ iff Correct(p) in every state $S_n$ with $k \leq n \leq k'$ where $\text{Clock}(p)_k \leq t < \text{Clock}(p)_{k+1}$ & $\text{Clock}(p)_{k'-1} < t' \leq \text{Clock}(p)_{k'}$

• Following these lines to make intuitive protocol related notions rigorous, the above mentioned six goals of the group membership protocols can be turned into precise statements on protocol runs, becoming theorems which have been proved in [GurevichMani1995].
Exercises to the Processor-Group Membership Protocol

• Define the \textit{<-relation for moves} of agents $M(p)$ in (asynchronous multi-agent) protocol runs.
  – Hint: use the broadcast reliability assumption.

• Prove the \textbf{finite history property} for protocol runs.
  – Hint: use the finiteness of \textsc{PROCESSOR}, the monotonicity of every \textsc{Clock}(p), the positive heartbeat interval $d_h > d_u$, the lower recovery bound $d_r > d_h + d_u$, the new-gp timestamp increment $d_n > d_c + d_u$, and the upper carry-over bound $d_c$.

• Prove the \textbf{coherence property} for protocol runs.

• Modify the related rules of $M(p)$ to include the lower \textbf{Recovery Bound $d_r$} on recovery time and the upper \textbf{Delivery Bound $d_c$} on message delivery time over the network, so that the two constraints on these bounds can be deleted from the notion of protocol runs.
Exercises to the Processor-Group Membership Protocol

• Modify Broadcast m to keep the size of InTransit bounded.
  – Hint: Instead of sending a single copy of m to the entire group, make InTransit binary to send one copy to each processor p, to be deleted by the MsgCarrier(p) when put into InBox(p).

• Modify M(p) to keep the number of messages in InBox(p) bounded, deleting unselectable messages, e.g. those arriving while p is crashed so that they may never be seen by MS(p) and never be removed from InBox(p).
  – Hint: Modify the Custodian(p)-rule to delete messages
    • with timestamp < Clock(p) if p is crashed
    • with timestamp < StartUpTime(p) if p is alive
References

- F. Cristian: Reaching Agreement on Processor-Group Membership in Synchronous Distributed Systems.
  - In: Distributed Computing, 6: 175-187, April 1991

- Y. Gurevich and R. Mani: Group Membership Protocol: Specification and Verification

- E. Börger, R. Stärk: Abstract State Machines. A Method for High-Level System Design and Analysis
  Springer-Verlag 2003, see http://www.di.unipi.it/AsmBook