A Formal Venture into Reliable Multicast Territory

Carolos Livadas and Nancy A. Lynch

Laboratory for Computer Science
Massachusetts Institute of Technology
{clivadas,lynch}@theory.lcs.mit.edu

Abstract. In this paper, we present a formal model of the reliable multicast service that ensures eventual packet delivery with, possibly, some timeliness guarantees. This model dictates precisely what it means to be a member of the reliable multicast group and which packets are guaranteed delivery to which members of the group. Moreover, it is reasonable, implementable, and broad; that is, it captures the intended behavior of numerous reliable multicast protocols. We also present a formal model of the Scalable Reliable Multicast (SRM) protocol [1]. We show that our model of SRM is safe, in the sense that it is a faithful implementation of our model of the reliable multicast service; that is, it may only deliver appropriate packets to each member of the reliable multicast group. We also show that, under certain constraints, the implementation is live, in the sense that it guarantees the timely delivery of the appropriate packets to the appropriate members of the reliable multicast group.

1 Introduction

With the increasing use of the Internet, multi-party communication and collaboration applications are becoming mainstream. Reliable multicast is a communication service that facilitates such applications. In the recent past, a slew of protocols have been proposed to reliably multicast packets efficiently [1–6]. However, reliability in the multicast setting has assumed many meanings, ranging from in-order eventual delivery to timely delivery where a small percentage of packet losses is tolerable. The many notions of reliability stem from the varying assumptions regarding the communication environment and the goals and requirements of the applications to which particular reliable multicast protocols cater.

Most often, the behavior of reliable multicast protocols is described informally. To our surprise, a protocol's description is seldom accompanied by a precise definition of its reliability guarantees. In its simplest form, reliability is informally defined as the eventual delivery of all multicast packets to all group members; other notions of reliability include ordering, no-duplication, and timeliness guarantees. Although intuitive, this simplistic reliability definition does not precisely specify which packets are guaranteed delivery to which members of the group, especially when the group membership is dynamic. Moreover, protocol descriptions put little emphasis on the behavior, or the analysis of the behavior, of the protocol when the group membership is dynamic, either due to failures or frequent joins and leaves. As hosts become more mobile, a better understanding of the behavior of such services and protocols in the context of a dynamic group membership is increasingly important.

In this paper, we present a formal model of the reliable multicast service, which we henceforth refer to as the *reliable multicast specification* (RMS). Specifying the reliable multicast service is not straightforward. The plethora of reliable multicast protocols cater to diverse applications that impose diverse correctness and performance requirements. Clearly, capturing the functionality of all reliable multicast protocols using a single specification would be quite complex and unwieldy. Our reliable multicast service specification formalizes the behavior of a number of protocols, such as SRM [1] and LMS [5], that strive to provide eventual delivery with, possibly, some timeliness guarantees. We stipulate that, in the context of dynamic group membership, membership is intrinsically intertwined with reliability; that is, membership and reliability must be addressed together. Thus, our specification dictates precisely what it means to be a member of a reliable multicast group and which packets are guaranteed delivery to which members of the reliable multicast group. We parameterize our specification with a delivery latency bound, which specifies an upper bound on the latency incurred to reliably deliver multicast packets. This parameterization results in a reliable multicast service specification that encompasses the behavior of a collection of reliable multicast protocols, some with loose and others with potentially stringent timeliness guarantees.

We also present a formal model of the Scalable Reliable Multicast (SRM) protocol [1]. Our model of SRM, which we henceforth refer to as the *reliable multicast implementation* (RMI), involves several components with distinct functionalities, such as the maintenance of the reliable multicast group membership and the packet loss recovery. This decomposition simplifies the reasoning and facilitates future modifications to the implementation. We show that RMI is safe, in the sense that it is a faithful implementation of RMS; that is, it may only deliver appropriate packets to each member of the reliable multicast group. We also show that, under certain constraints, RMI is live, in the sense that it guarantees the timely delivery of the appropriate packets to the appropriate members of the group.

The rest of the paper is organized as follows. Section 2 presents our modeling framework. Section 3 presents the abstract view of the physical system that we adopt in our work. Section 4 presents RMS and its eventual and timely reliability properties. Section 5 presents RMI, derives constraints on RMI's packet loss recovery parameters, and analyzes RMI's safety and liveness with respect to RMS. Finally, Section 6 presents the paper's contributions and future work directions. Due to space constraints, the presentation of the material in this paper is quite terse. A complete presentation of this material appears in Ref. 7.

2 Modeling Framework and Notation

In this paper, we use the *timed input/output (I/O) automaton* (TIOA) modeling framework (introduced as the *general timed automaton* model in Ref. 8); a framework for modeling timed systems. A timed I/O automaton A is a state-machine in which transitions are labeled by *actions*. A's actions ($acts(A)$) are partitioned into *input* ($in(A)$), *output* ($out(A)$), *internal* ($int(A)$), and *time-passage* sets. Time-passage actions model the passage of time. The input and output actions of A are collectively referred to as *external*; denoted $ext(A)$. Input, output, and time-passage actions are collectively referred to as *visible*; denoted $vis(A)$. A timed I/O automaton A is defined by its *signature* (input, output, internal, and time-passage actions), states, start states, and state-transition relation (a cross product of states, actions, and states that dictates A's allowable transitions).

A *timed execution fragment* α of A is a finite or infinite alternating sequence, $\alpha = s_0\pi_1 s_1\pi_2 s_2\ldots$, of states and actions consistent with A's state-transition relation. For any two timed execution fragments α and α' of A, we use the notation $\alpha \leq \alpha'$ to denote that α is a prefix of α'. A timed execution fragment of A is *admissible* if an infinite amount of time elapses within the particular fragment. An admissible timed execution fragment α of A is *fair* when no action is enabled in every state of a suffix of α without appearing in the given suffix. The time of occurrence of an action π_k, for $k \in \mathbb{N}^+$, within a timed execution fragment α of A is the time elapsing within α prior to the occurrence of π_k. The *timed trace* β of a timed execution fragment α of A is the sequence of visible actions in α, each paired with its time of occurrence. For any two timed traces β and β' of A, we use the notation $\beta \leq \beta'$ to denote that β is a prefix of β'.

A *timed execution* of A is a timed execution fragment of A that begins in one of A's start states. We let $aexecs(A)$ denote the set of all admissible timed executions of A, $attraces(A)$ denote the timed traces of all executions in $aexecs(A)$, $fair\text{-}aexecs(A)$ denote the set of all fair admissible timed executions of A, and $fair\text{-}attraces(A)$ denote the timed traces of all executions in $fair\text{-}aexecs(A)$.

Two timed I/O automata A_1 and A_2 are *compatible* if $int(A_i) \cap acts(A_j) = \emptyset$ and $out(A_i) \cap out(A_j) = \emptyset$, for $i, j \in \{1, 2\}, i \neq j$. The composition of compatible timed I/O automata yields a timed I/O automaton. The *hiding* operation reclassifies output actions of a timed I/O automaton as internal. Letting A, B be timed I/O automata with the same external interface, B *implements* A, denoted $B \leq A$, when its external behavior is allowed by A; that is, when $attraces(B) \subseteq attraces(A)$. The implementation relation among two timed I/O automata is often shown by defining a *timed simulation relation*; that is, relating states of B to states of A and showing that for any step of B there is a timed execution fragment of A with the same timed trace as the step of B that preserves the state relation.

We use a *precondition-effect* style notation to define the state-transition relations of timed I/O automata. Moreover, we use the notation $S_1 \cup= S_2$, $S_1 \setminus= S_2$, and $s :\in S$ as shorthand for $S_1 := S_1 \cup S_2$, $S_1 := S_1 \setminus S_2$, and the assignment of an arbitrary element of S to the variable s.

3 The Physical System

We assume that the physical system is comprised of an infinite set of hosts that interact through an underlying network. This network involves a set of interconnected routers. Each host is connected to a particular router of the underlying network; for each host, we refer to this particular router as the *gateway router* of the particular host. Hosts and routers are connected among themselves through bi-directional communication links.

We assume that all hosts are of comparable processing power and storage resources. Resident on each host are a set of processes. We assume that hosts are symmetric in the sense that the same set of processes reside on each host. The set of processes on each host consists of a single application process and several additional communication service processes. Henceforth, we refer to the application process at each host as the *client* at the given host. The communication service processes, either individually or collectively, provide the communication services required by the client. For instance, the IP unicast service may be modeled as a set of processes, one such process for each host. Clients may thus exchange IP unicast packets through their respective IP unicast processes; these may in turn interact with the hosts' gateway routers.

In terms of system faults, we consider only host crashes and packet drops on the communication links. Once a host crashes it remains crashed thereafter. A host is said to be *operational* prior to crashing and to have *crashed* thereafter. All the processes on each host are *fate-sharing*; that is, if a host crashes, then all of its processes crash. Router failures and network partitions are assumed to be ephemeral. Such failures are modeled as numerous consecutive packet drops.

Since crashes are assumed to be permanent, we model host restarts implicitly. We think of the restarting of a host as its reincarnation as a completely new host; that is, after crashing, a host may assume the identity of another host that has up to that point in time been idle. This modeling simplification is equivalent to explicitly modeling host restarts and having hosts choose a unique host identifier each time they restart. Such an identifier could involve, for instance, the processor identifier and an infinite reincarnation counter that is stable across crashes.

4 Reliable Multicast Specification (RMS)

We abstractly model the reliable multicast service as a single component that interacts with all client processes. Thus, the reliable multicast service encapsulates the behavior of all communication service processes at all hosts and the underlying network. For simplicity, we assume that there is a single reliable multicast group. Since we assume a single client per host and a single reliable multicast group, we do not distinguish among the client process and the host when considering reliable multicast group membership. In fact, we often use the terms client and host interchangeably.

Throughout our treatment of reliable multicast, we adopt the packet naming scheme used by Floyd *et al.* [1]. In this scheme, clients (applications) assign

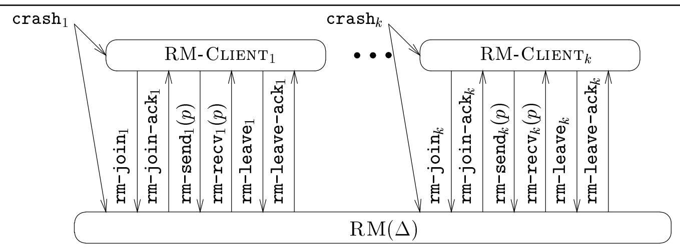
Fig. 1 Reliable Multicast Specification Definitions

H Set of all hosts.

$Status = \{\texttt{idle}, \texttt{joining}, \texttt{leaving}, \texttt{member}, \texttt{crashed}\}$

$P_{\text{RM-CLIENT}}$ = Set of packets such that $\forall\, p \in P_{\text{RM-CLIENT}}$
- $source(p) \in H$
- $seqno(p) \in \mathbb{N}$
- $data(p) \in \{0,1\}^*$
- $id(p) \in H \times \mathbb{N} : id(p) = \langle source(p), seqno(p)\rangle$
- $suffix(p) = \{\langle s,i\rangle \in H \times \mathbb{N} \mid source(p) = s \wedge seqno(p) \leq i\}$

Fig. 2 Reliable Multicast Specification Component Interaction



unique sequence numbers to each packet they multicast. These sequence numbers are assigned in a continuous fashion as hosts join, leave, and rejoin the reliable multicast group; that is, consecutive packets sent by each host are assigned consecutive sequence numbers. Thus, packets are uniquely and persistently identified by a pair involving their source host and their sequence number. Since the clients (applications) are responsible for naming packets, packets are referred to as *application data units* (ADUs).

We formally specify the reliable multicast service and each of the client processes using timed I/O automata. The automaton RM(Δ), for $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$, models the reliable multicast service. RM(Δ) defines what it means to be a member of the reliable multicast group and specifies precisely which packets are guaranteed delivery to each member of the reliable multicast group. The parameter Δ specifies an upper bound on the amount of time required by the reliable multicast service to reliably deliver each packet. The automaton RM-Client$_h$ models the client at the host h. We let RM-Clients denote the composition of all client automata and RM$_S(\Delta)$, for any $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$, denote the composition of the reliable multicast service and all client automata; that is, RM$_S(\Delta)$ = RM(Δ) $\times$ RM-Clients. Figure 1 includes several set definitions pertaining to our reliable multicast service specification. Figure 2 depicts the interaction of the RM(Δ) and RM-Client$_h$, for $h \in H$, automata.

We proceed by describing the functionality of the RM(Δ) and RM-Client$_h$ automata. We then present some preliminary properties and definitions pertaining to RM(Δ), RM-Client$_h$, and RM$_S(\Delta)$, for $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$ and $h \in H$. We conclude this section by presenting the reliability properties of RM$_S(\Delta)$.

4.1 The RM(Δ) Automaton

Figure 3 presents the signature, the variables, and the discrete transitions of RM(Δ).

Membership and Crashing The RM(Δ) automaton maintains the set of members of the reliable multicast group. Hosts initiate the process of joining and leaving the reliable multicast group by issuing join and leave requests to the reliable multicast service. A request to join the reliable multicast group is effective only when the host is *idle* with respect to the reliable multicast group; that is, it is operational and neither a member of nor in the process of joining or leaving the reliable multicast group. A host becomes a member of the reliable multicast group upon the acknowledgment of an earlier join request. Hosts may only send and receive packets through the reliable multicast service while they are both operational and members of the reliable multicast group. Once a host issues a request to leave the reliable multicast group, it ceases to be a member of the reliable multicast group and, thus, relinquishes its right to receive any more reliable multicast packets. Leave requests overrule join requests in the sense that if the client is already in the process of joining the group while it issues a leave request, then the process of joining is aborted and the process of leaving is initiated. Once a host leaves the reliable multicast group, it may later rejoin the reliable multicast group by re-issuing a join request. Hosts may crash at any point in time. Once a host has crashed, the reliable multicast service ignores all events pertaining to the crashed host. Recall that host restarts are treated implicitly by thinking of host restarts as host reincarnations.

Multicast Reliability We say that a member h of the reliable multicast group has *delivered* the packet p if it has either sent or received the packet p. We say that a member h of the reliable multicast group is *aware of* a packet p, or is *expecting* p, if it has delivered either p or an earlier packet p' from the source of p. Moreover, we say that a packet p is *active* if at least one member of the reliable multicast group that has become aware of p since last joining the reliable multicast group, has also delivered it since last joining the reliable multicast group.

Once a host joins the reliable multicast group, the issue of catching up on any of the packets multicast earlier is orthogonal to the transmission of future packets using the reliable multicast service. Thus, once a host joins the reliable multicast group, the first packet it receives from a particular source dictates the set of packets that are guaranteed delivery to the given host. In particular, none of the earlier packets and any of the later packets that remain active after being sent are guaranteed delivery, provided the host remains a member of the reliable multicast group. The host may catch up on earlier packets from the given source through a separate service. For example, earlier packets may be requested directly from the source through a unicast communication channel. The rationale behind this modeling choice is that the recovery of a large number

Fig. 3 The RM(Δ) Automaton

Parameters:

$\Delta \in \mathbb{R}^{\geq 0} \cup \{\infty\}$

Actions:

Input:
- $\texttt{crash}_h$, for $h \in H$
- $\texttt{rm-join}_h$, for $h \in H$
- $\texttt{rm-leave}_h$, for $h \in H$
- $\texttt{rm-send}_h(p)$, for $h \in H, p \in P_{\text{RM-CLIENT}}$

Output:
- $\texttt{rm-join-ack}_h$, for $h \in H$
- $\texttt{rm-leave-ack}_h$, for $h \in H$
- $\texttt{rm-recv}_h(p)$, for $h \in H, p \in P_{\text{RM-CLIENT}}$

Time Passage:
- $\nu(t)$, for $t \in \mathbb{R}^{\geq 0}$

Variables:

$now \in \mathbb{R}^{\geq 0}$, initially $now = 0$
$status(h) \in Status$, for all $h \in H$, initially $status(h) = \texttt{idle}$, for all $h \in H$
$trans\text{-}time(p) \in \mathbb{R}^{\geq 0} \cup \perp$, for all $p \in P_{\text{RM-CLIENT}}$, initially $trans\text{-}time(p) = \perp$, for all $p \in P_{\text{RM-CLIENT}}$
$expected(h, h') \subseteq H \times \mathbb{N}$, for all $h, h' \in H$, initially $expected(h, h') = \emptyset$, for all $h, h' \in H$
$delivered(h, h') \subseteq H \times \mathbb{N}$, for all $h, h' \in H$, initially $delivered(h, h') = \emptyset$, for all $h, h' \in H$

Derived Variables:

$idle = \{h \in H \mid status(h) = \texttt{idle}\}$
$joining = \{h \in H \mid status(h) = \texttt{joining}\}$
$leaving = \{h \in H \mid status(h) = \texttt{leaving}\}$
$members = \{h \in H \mid status(h) = \texttt{member}\}$
$intended(p) = \{h \in H \mid id(p) \in expected(h, source(p))\}$, for all $p \in P_{\text{RM-CLIENT}}$
$completed(p) = \{h \in H \mid id(p) \in delivered(h, source(p))\}$, for all $p \in P_{\text{RM-CLIENT}}$
$sent\text{-}pkts = \{p \in P_{\text{RM-CLIENT}} \mid trans\text{-}time(p) \neq \perp\}$
$active\text{-}pkts = \{p \in P_{\text{RM-CLIENT}} \mid p \in sent\text{-}pkts \wedge intended(p) \cap completed(p) \neq \emptyset\}$

Discrete Transitions:

input $\texttt{crash}_h$
eff $status(h) := \texttt{crashed}$
  foreach $h' \in H$ **do:**
    $expected(h, h') := \emptyset$
    $delivered(h, h') := \emptyset$

input $\texttt{rm-join}_h$
eff **if** $h \in idle$ **then**
    $status(h) := \texttt{joining}$

input $\texttt{rm-leave}_h$
eff **if** $h \in joining \cup members$ **then**
    $status(h) := \texttt{leaving}$
    foreach $h' \in H$ **do:**
      $expected(h, h') := \emptyset$
      $delivered(h, h') := \emptyset$

input $\texttt{rm-send}_h(p)$
eff **if** $h \in members \cap \{source(p)\}$ **then**
    if $expected(h, h) = \emptyset$ **then**
      $expected(h, h) := suffix(p)$
    if $id(p) \in expected(h, h)$ **then**
      $trans\text{-}time(p) := now$
      $delivered(h, h) \cup= \{id(p)\}$

output $\texttt{rm-join-ack}_h$
pre $h \in joining$
eff $status(h) := \texttt{member}$

output $\texttt{rm-leave-ack}_h$
pre $h \in leaving$
eff $status(h) := \texttt{idle}$

output $\texttt{rm-recv}_h(p)$
pre $h \in members \backslash \{source(p)\}$
  $\wedge p \in sent\text{-}pkts$
  $\wedge (expected(h, source(p)) = \emptyset$
    $\Rightarrow now \leq trans\text{-}time(p) + \Delta)$
  $\wedge (expected(h, source(p)) \neq \emptyset$
    $\Rightarrow id(p) \in expected(h, source(p)))$
eff **if** $expected(h, source(p)) = \emptyset$ **then**
    $expected(h, source(p)) := suffix(p)$
  $delivered(h, source(p)) \cup= \{id(p)\}$

time-passage $\nu(t)$
pre $\forall\, p \in active\text{-}pkts$,
    $now + t \leq trans\text{-}time(p) + \Delta$
    $\vee intended(p) \subseteq completed(p)$
eff $now := now + t$

of earlier packets may strain the reliable multicast service and wastefully expose the recovery of earlier packets to all or a subset of the reliable multicast group.

If Δ is equal to infinity, then RM(Δ) guarantees that if a packet p remains active forever after its transmission then any member that becomes aware of p and remains a member of the reliable multicast group thereafter, delivers p. Equivalently, if two members become aware of a packet p, remain members forever thereafter, and one member delivers p, then the other member delivers p

Fig. 4 The RM-CLIENT$_h$ Automaton

Parameters:

$h \in H$

Actions:

Input:
`crash`$_h$
`rm-join-ack`$_h$
`rm-leave-ack`$_h$
`rm-recv`$_h(p)$, for all $p \in P_{\text{RM-CLIENT}}$

Output:
`rm-join`$_h$
`rm-leave`$_h$
`rm-send`$_h(p)$, for all $p \in P_{\text{RM-CLIENT}}$
Time Passage:
$\nu(t)$, for $t \in \mathbb{R}^{\geq 0}$

Variables:

$now \in \mathbb{R}^{\geq 0}$, initially $now = 0$
$status \in Status$, initially $status =$ `idle`
$seqno \in \mathbb{N}$, initially $seqno = 0$

Discrete Transitions:

input `crash`$_h$
eff $status :=$ `crashed`

input `rm-join-ack`$_h$
eff **if** $status =$ `joining` **then**
$status :=$ `member`

input `rm-leave-ack`$_h$
eff **if** $status =$ `leaving` **then**
$status :=$ `idle`

input `rm-recv`$_h(p)$
eff None

output `rm-join`$_h$
pre $status =$ `idle`
eff $status :=$ `joining`

output `rm-leave`$_h$
pre $status =$ `member`
eff $status :=$ `leaving`

output `rm-send`$_h(p)$
pre $status =$ `member`
$\wedge source(p) = h \wedge seqno(p) = seqno$
eff $seqno := seqno + 1$

time-passage $\nu(t)$
pre None
eff $now := now + t$

also. It is important to note that a host is not required to remain a member of the reliable multicast group indefinitely in order for the packets it multicasts to be received by hosts that become aware of them; the eventual reception of packets is guaranteed to all hosts that become aware of them provided the packets remain active forever after they are sent.

If Δ if finite, then RM(Δ) guarantees that if a packet remains active for Δ time units past its transmission, then it is delivered to all hosts that become aware of it within these Δ time units and, subsequently, remain members of the reliable multicast group for the remaining duration of these Δ time units elapse.

4.2 The RM-CLIENT$_h$ Automata

Figure 4 presents the signature, the variables, and the discrete transitions of RM-CLIENT$_h$. The RM-CLIENT$_h$ automaton models a *well-behaved* client; that is, a client that: i) transmits packets only when it is a member of the reliable multicast group, ii) transmits packets in ascending and contiguous sequence number order, iii) issues join requests only when it is idle with respect to the reliable multicast group, and iv) issues leave requests only when it is a member of the reliable multicast group.

4.3 Preliminary Properties and Definitions

The automaton RM-$\textsc{Client}_h$, for any $h \in H$, satisfies *transmission correctness* and *transmission uniqueness*. Transmission correctness is the property that clients only transmit packets for which they are actually the source. Transmission uniqueness is the property that no two packets transmitted by a client share the same identifier.

Lemma 1 (Transmission Correctness). *Let β be any timed trace of the automaton* RM-$\textsc{Client}_h$*, for any $h \in H$. If β contains the action* $\texttt{rm-send}_h(p)$*, for some $p \in P_{\text{RM-}\textsc{Client}}$, then the host h is the source of p; that is, $h = source(p)$.*

Lemma 2 (Transmission Uniqueness). *Let β be any timed trace of the automaton* RM-$\textsc{Client}_h$*, for any $h \in H$. For any packet identifier $\langle s, i\rangle \in H \times \mathbb{N}$, β contains at most one action* $\texttt{rm-send}_h(p)$*, for $p \in P_{\text{RM-}\textsc{Client}}$, such that $id(p) = \langle s, i\rangle$.*

The automaton $\mathrm{RM}_S(\Delta)$, for any $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$ satisfies *transmission integrity*. Transmission integrity it the property that, within a timed trace of $\mathrm{RM}_S(\Delta)$, the reception of a packet must be preceded by the particular packet's transmission.

Lemma 3 (Transmission Integrity). *Let β be any timed trace of* $\mathrm{RM}_S(\Delta)$*, for any $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$. For $h, h' \in H$ and $p \in P_{\text{RM-}\textsc{Client}}$, such that $h \neq h'$ and $h = source(p)$, it is the case that any* $\texttt{rm-recv}_{h'}(p)$ *action is preceded in β by a* $\texttt{rm-send}_h(p)$ *action.*

Let β be a timed trace of $\mathrm{RM}_S(\Delta)$, for any $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$, and $p \in P_{\text{RM-}\textsc{Client}}$ be any packet transmitted within β using the reliable multicast service. We proceed by defining the set of *members* of β, the *intended and completed delivery* sets of p within β, and the set of *active packets* within β. The members of a timed trace β are the hosts that have neither crashed nor left the reliable multicast group since last joining the reliable multicast group within β. The intended delivery set of p within β is the set of intended recipients of p within β; that is, the set of members of β that have become aware of p since last joining the reliable multicast group. The completed delivery set of p within β is the set of members of β that have delivered p since last joining the reliable multicast group. The set of active packets within β is comprised of the packets whose intended and completed delivery sets within β intersect.

Definition 1 (Membership). *Let β be any timed trace of* $\mathrm{RM}_S(\Delta)$*, for any $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$. We define the* members of β*, denoted $members(\beta)$, to be the set of hosts $h \in H$ such that β contains a* $\texttt{rm-join-ack}_h$ *action that is not succeeded by either a* $\texttt{rm-leave}_h$ *or a* $\texttt{crash}_h$ *action.*

Definition 2 (Intended Delivery Set). *Let β be any timed trace of the automaton* $\mathrm{RM}_S(\Delta)$*, for any $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$, containing the transmission of a packet $p \in P_{\text{RM-}\textsc{Client}}$. We define the* intended delivery set of p within β*, denoted $intended(p, \beta)$, to be the members of β that have delivered either the packet* *p or an earlier packet from the source of p since they last joined the reliable multicast group; that is,* $h \in$ *intended*(p, β) *if and only if* $h \in$ *members*(β) *and the last* `rm-join-ack`$_h$ *action in* β *is succeeded by either a rm-send*$_h(p')$ *or a rm-recv*$_h(p')$ *action, where source*$(p') =$ *source*(p) *and seqno*$(p') \leq$ *seqno*(p)*.*

Definition 3 (Completed Delivery Set). *Let* β *be any timed trace of the automaton* $\mathrm{RM}_S(\Delta)$*, for any* $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$*, containing the transmission of a packet* $p \in P_{\text{RM-CLIENT}}$*. We define the* completed delivery set of p within β*, denoted completed*(p, β)*, to be the members of* β *that have delivered the packet* p *since they last joined the reliable multicast group; that is,* $h \in$ *completed*(p, β) *if and only if* $h \in$ *members*(β) *and the last* `rm-join-ack`$_h$ *action in* β *is succeeded by either a rm-send*$_h(p)$ *or a rm-recv*$_h(p)$ *action.*

Definition 4 (Active Packets). *Let* β *be any finite timed trace of the automaton* $\mathrm{RM}_S(\Delta)$*, for any* $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$*. We define the set of* active packets within β*, denoted active-pkts*(β)*, to be the set of all packets* $p \in P_{\text{RM-CLIENT}}$ *such that intended*$(p, \beta) \cap$ *completed*$(p, \beta) \neq \emptyset$*. If* $p \in$ *active-pkts*(β)*, then we say that* p *is active within* β*.*

4.4 Reliability Properties

The $\mathrm{RM}_S(\Delta)$ automaton, for any $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$, satisfies the *eventual delivery* and, equivalently, *pairwise eventual delivery*, properties. Eventual delivery is the property that if a host h is a member of the reliable multicast group, becomes aware of a packet p, remains a member of the group thereafter, and p remains active thereafter, then h delivers p since last joining the reliable multicast group. Its pairwise counterpart is the property that if two hosts are members of the reliable multicast group, become aware of the packet p, remain members of the group thereafter, and one of them delivers p since last joining the reliable multicast group, then so does the other. The eventual and pairwise eventual delivery properties are equivalent.

Theorem 1 (Eventual Delivery). *Let* β *be any fair admissible timed trace of* $\mathrm{RM}_S(\Delta)$*, for any* $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$*, containing the transmission of a packet* $p \in P_{\text{RM-CLIENT}}$*. If* $p \in$ *active-pkts*(β)*, then it is the case that intended*$(p, \beta) \subseteq$ *completed*(p, β)*.*

Corollary 1 (Pairwise Eventual Delivery). *Let* β *be any fair admissible timed trace of the* $\mathrm{RM}_S(\Delta)$ *automaton, for any* $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$*, containing the transmission of a packet* $p \in P_{\text{RM-CLIENT}}$*. For any hosts* $h, h' \in$ *intended*(p, β)*, it is the case that* $h \in$ *completed*$(p, \beta) \Rightarrow h' \in$ *completed*(p, β)*.*

Finally, the $\mathrm{RM}_S(\Delta)$ automaton, for any $\Delta \in \mathbb{R}^{\geq 0}$, satisfies the *time-bounded delivery* property. Time-bounded delivery is the property that, at any point in time Δ time units past the transmission of a packet, if the given packet is active, then is has been delivered to all members of the reliable multicast group that are aware of it.

Theorem 2 (Time-Bounded Delivery). *Let β be any timed trace of the automaton* $\mathrm{RM}_S(\Delta)$*, for any $\Delta \in \mathbb{R}^{\geq 0}$, that contains the transmission of a packet $p \in P_{\text{RM-CLIENT}}$. Let β' be the finite prefix of β ending with the transmission of p; that is, the last action contained in β' is the action* $\texttt{rm-send}_h(p)$*, for $h \in H, h = source(p)$. Let $t, t' \in \mathbb{R}^{\geq 0}$ be the time of occurrence of the last actions of β and β', respectively. For any $h' \in H$, if $t' + \Delta < t$, $p \in active\text{-}pkts(\beta)$ and $h' \in intended(p, \beta)$, then it is the case that $h' \in completed(p, \beta)$.*

5 Reliable Multicast Implementation (RMI)

In this section, we present RMI — a formal model of the Scalable Reliable Multicast (SRM) protocol [1]. RMI precisely specifies the behavior of the basic version of SRM — more sophisticated versions involve adaptive and local recovery schemes [1,9]. We proceed by describing RMI's architecture, briefly present RMI's packet loss recovery scheme, and derive some constraints on RMI's parameters. Finally, we state the safety and liveness properties of RMI with respect to RMS.

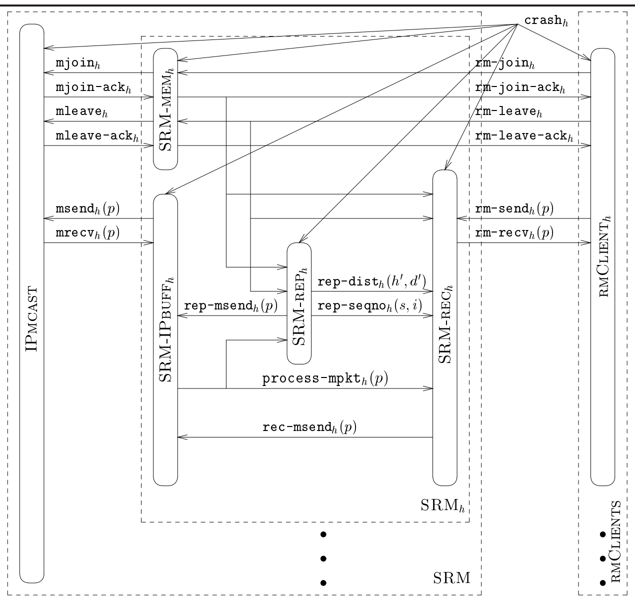
5.1 RMI's Architecture

Presuming the abstract view of the physical system introduced in Section 3, RMI involves the interaction of a set of client processes, one process per host, a set of reliable multicast processes, one process per host, and an IP multicast service component. The client processes are identical to those presented in Section 4. The reliable multicast processes execute the SRM protocol. The IP multicast service component encapsulates the behavior of all communication processes at all hosts and the underlying network and provides the best-effort multicast primitive.

We model each reliable multicast process as four interacting components, each with distinct functionalities. The *membership component* manages the reliable multicast group membership of the host. It handles the join and leave requests of the client process and issues join and leave requests to the underlying IP multicast service. The *IP buffer component* buffers all packets either received from or to be transmitted using the underlying IP multicast service. The *recovery component* incorporates all the functionality pertaining to the detection and recovery of missing packets. Finally, the *reporting component* incorporates all the functionality pertaining to the exchange of session messages among the members of the reliable multicast group. Session messages are used to exchange transmission state and inter-host round-trip-time (RTT) information. This information aids the detection of losses, in particular during transmission gaps, and the calculation of inter-host round-trip-time estimates, which are required by the recovery component.

Figure 5 depicts the interaction of the various components of RMI. The reliable multicast process SRM_h at each host h is the composition of the automata SRM-MEM$_h$, SRM-IPBUFF$_h$, SRM-REC$_h$, and SRM-REP$_h$. The reliable multicast implementation as a whole, denoted SRM, is the composition

Fig. 5 Reliable Multicast Implementation Component Interaction



of the SRM processes and the underlying IP multicast service after hiding all output actions that are not output actions of the specification $\mathrm{RM}(\Delta)$, for any $\Delta \in \mathbb{R}^{\geq 0} \cup \infty$; that is, $\mathrm{SRM} = \mathit{hide}_{\Phi}(\prod_{h \in H} \mathrm{SRM}_h \times \mathrm{IP}\textsc{mcast})$, with $\Phi = \mathit{out}(\prod_{h \in H} \mathrm{SRM}_h \times \mathrm{IP}\textsc{mcast}) \setminus \mathit{out}(\mathrm{RM}(\Delta))$. Finally, we define RM_I to be the composition of the reliable multicast implementation with all the client automata; that is, $\mathrm{RM}_I = \mathrm{SRM} \times \mathrm{RM}\text{-}\textsc{Clients}$. Ref. 7 contains complete timed I/O automaton models of each of the components depicted in Figure 5.

5.2 Overview of RMI's Packet Loss Recovery

Receivers detect packet losses by identifying sequence number gaps in the stream of packets received from each source. Upon detecting the loss of a packet p, a host h initiates a new recovery round for p by scheduling a retransmission *request* for p. This request is scheduled for transmission at a point in time in the future that is uniformly chosen within the interval $[C_1 \hat{d}_{hs}, (C_1 + C_2)\hat{d}_{hs}]$, where $C_1, C_2 \in \mathbb{R}^{\geq 0}$ are request scheduling parameters and $\hat{d}_{hs}$ is half of h's round-trip-time (RTT) estimate to the source s of the packet p.

Upon either the transmission of a request for p or the reception of a request for p while a request for p is pending transmission, the host h initiates a new recovery round for p by rescheduling the request for p for transmission at a point in time in the future that is uniformly chosen within the interval $2^{k-1}[C_1 \hat{d}_{hs}, (C_1 + C_2)\hat{d}_{hs}]$, where $k \in \mathbb{N}^{+}$ is the number of recovery rounds for p that h has already initiated. In effect, the request for p is rescheduled by performing an exponential back-off. If h receives p while a request for p is pending transmission, then the request for p is canceled.

Once h reschedules its request for p, it observes a *back-off abstinence period.* During this period, it refrains from backing-off its request for p. Any requests for p received during this period are considered to pertain to prior recovery rounds and are discarded. Thus, back-off abstinence periods prevent requests from being backed-off multiple times by requests pertaining to the same recovery round. The back-off abstinence period for p expires at the point in time that is $2^{k-1}C_3\hat{d}_{hs}$ time units in the future, where $k \in \mathbb{N}^+$ is the number of recovery rounds for p that h has already initiated and $C_3 \in \mathbb{R}^{\geq 0}$ is the back-off abstinence parameter.

Our modeling of back-off abstinence periods departs slightly from SRM. Floyd *et al.* [1] propose two schemes for ensuring that requests are backed off only once per recovery round. The first scheme involves back-off abstinence periods that expire once half the time to the transmission time of the respective request has elapsed. Our use of a parameter for specifying how long to abstain from backing off allows more tuning freedom. Moreover, having back-off abstinence periods expire once half the time to the transmission time of the respective request has elapsed allows for the back-off abstinence period to overlap the interval within which requests are scheduled. This seems to go against the intention of the abstinence period. Requests received within the interval within which the current request was scheduled, should be considered to be requests of the current round and, thus, should result in the rescheduling of the current request. The second scheme annotates requests with their recovery round and backs off requests only upon receiving a request pertaining to the same or, presumably, a later round.

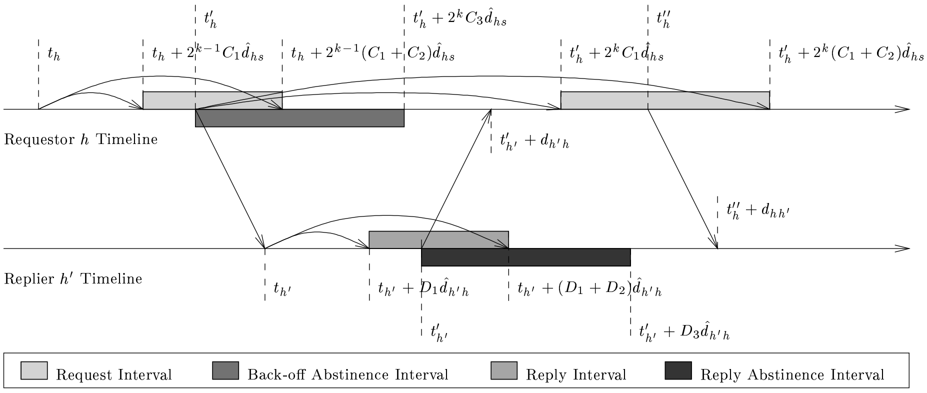
If a host h' receives a request for the packet p from the host h and it has already either sent or received p, then it schedules a *reply* for (retransmission of) p. This reply is scheduled for transmission at a point in time in the future that is uniformly chosen within the interval $[D_1\hat{d}_{h'h}, (D_1 + D_2)\hat{d}_{h'h}]$, where $D_1, D_2 \in \mathbb{R}^{\geq 0}$ are reply scheduling parameters and $\hat{d}_{h'h}$ is half of h''s RTT estimate to h (the requestor of p). If h' receives a reply for p while its own reply for p is pending transmission, then h' cancels its own reply for p.

Once h' either receives a reply for p or retransmits p itself, it observes a *reply abstinence period*; a period during which it refrains from scheduling replies to requests for p. The reply abstinence period for p expires at the point in time that is $D_3\hat{d}_{hh'}$ time units in the future, where $D_3 \in \mathbb{R}^{\geq 0}$ is the reply abstinence parameter. The reply abstinence period prevents multiple requests pertaining to a given recovery round from generating multiple replies.

5.3 Constraints on RMI's Parameters

Figure 6 illustrates the behavior of RMI's packet loss recovery scheme. In particular, for any $k \in \mathbb{N}^+$, it depicts the transmission of a k-th round request by h, the scheduling of a $k+1$-st round request by h, and the scheduling of a reply to h's k-th round request by a host h'. t_h is the point in time at which h schedules its k-th round request, t'_h is the point in time for which h schedules its k-th round request, $t_{h'}$ is the point in time h' receives h's k-th round request, and $t'_{h'}$ is the point in time for which h' schedules its reply to h's k-th round request.

Fig. 6 Timing Diagram of SRM's Loss Recovery Scheme



$\hat{d}_{hs}$ is half of h's RTT estimate to the source s of the packet being recovered, $d_{hh'}$ and $d_{h'h}$ are the actual transmission latencies between h and h', and $\hat{d}_{h'h}$ is half of h' RTT estimate to the host h.

RMI must ensure that the back-off abstinence intervals do not overlap with request intervals. From Figure 6, this requirement is enforced by imposing the parameter constraint $C_3 < C_1$. Moreover, RMI must ensure that requestors schedule their retransmission requests such that they succeed the reception of replies pertaining to prior recovery rounds. Prematurely transmitting requests would result in wasteful recovery traffic. From Figure 6, this requirement corresponds to the satisfaction of the inequalities $d_{hh'} + (D_1 + D_2)\hat{d}_{h'h} + d_{h'h} < 2^k C_1 \hat{d}_{hs}$, for $k \in \mathbb{N}^+$. Presuming that inter-host transmission latencies are fixed and symmetric and that RMI's inter-host RTT estimates are accurate, these inequalities are satisfied if $D_1 + D_2 + 2 < 2C_1$. Finally, RMI must also ensure that a particular round's requests are not discarded by potential repliers because they are received during the repliers' abstinence periods pertaining to the prior recovery round. From Figure 6, this requirement corresponds to the satisfaction of the inequalities $d_{hh'} + (D_1 + D_2)\hat{d}_{h'h} + D_3\hat{d}_{h'h} < 2^k C_1 \hat{d}_{hs} + d_{hh'}$, for $k \in \mathbb{N}^+$. Presuming that inter-host transmission latencies are fixed and symmetric and that RMI's inter-host RTT estimates are accurate, these inequalities are satisfied if $D_1 + D_2 + D_3 < 2C_1$.

The following assumption summarizes the constraints on RMI's parameters.

Assumption 1 RM_I*'s parameters* C_1, C_2, C_3, D_1, D_2, *and* D_3 *satisfy the following constraints:* $C_3 < C_1$, $D_1 + D_2 + 2 < 2C_1$, *and* $D_1 + D_2 + D_3 < 2C_1$.

To our knowledge, these constraints on SRM's request/reply scheduling parameters, or even similar ones, have not been expressed to date. In fact, most analyses and simulations presume that no recovery packets are lost; that is, they presume that the initial recovery round is always successful. Our timing analysis illustrates that if the parameters are chosen arbitrarily it is possible to cause either superfluous requests and replies or the failure of a recovery round due to replier abstinence. Although in practice, due to inaccurate inter-host RTT estimates and varying and non-symmetric inter-host transmission latencies, superfluous traffic and/or recovery round failure may indeed be unavoidable, it is still important to realize their tie to SRM's parameters.

5.4 Safety and Liveness Analysis of RMI

The following theorem states that the reliable multicast implementation, RM_I, is safe, in the sense that it implements $\mathrm{RM}_S(\infty)$; that is, it may only deliver appropriate packets to each member of the reliable multicast group.

Theorem 3. $\mathrm{RM}_I \leq \mathrm{RM}_S(\infty)$

We let $aexecs_k(\mathrm{RM}_I)$, for $k \in \mathbb{N}$, be the subset of admissible timed executions of RM_I within each of whose elements the number of packet drops suffered collectively by all packets pertaining to the transmission and, potentially, the recovery of any packet p is at most k. Finally, we let $attraces_k(\mathrm{RM}_I)$ be the traces of all executions of RM_I in $aexecs(\mathrm{RM}_I)$. Let $C\text{-}aexecs_k(\mathrm{RM}_I)$, for $k \in \mathbb{N}^+$, be the subset of $aexecs_k(\mathrm{RM}_I)$ comprised of all admissible timed executions of RM_I that satisfy the following constraints:

1. hosts neither leave the reliable multicast group nor crash,
2. the transmission latency incurred by any packet multicast using the IP multicast service by h and received by h' lies in the interval $[\underline{d}, \overline{d}]$, for $\underline{d}, \overline{d} \in \mathbb{R}^{\geq 0}$, such that $\underline{d} > 0$, $\overline{d} > 0$, and $\underline{d} \leq \overline{d}$,
3. the fate of any packet transmitted using the IP multicast service is resolved within $\overline{d}$ time units; that is, for any member h' of the IP multicast group when any packet p is transmitted, either h' delivers p or p is dropped on its way to h' within $\overline{d}$ time units of the transmission of p,
4. the inter-host transmission latency estimates of the recovery component of each reliable multicast process of RM_I lie in the interval $[\underline{d}, \overline{d}]$, and
5. the delay in detecting losses is bounded by $\texttt{DET-BOUND}$, for $\texttt{DET-BOUND} \in \mathbb{R}^{\geq 0}$, such that $\overline{d} \leq \texttt{DET-BOUND}$.

Let $C\text{-}attraces_k(\mathrm{RM}_I)$ be the traces of all executions in $C\text{-}aexecs_k(\mathrm{RM}_I)$. Finally, let $k^* = \lceil \log_2[(D_1+D_2+D_3+2)\overline{d}-\underline{d}] - \log_2(C_3\underline{d}) \rceil$ and $\texttt{REC-BOUND}(k') = [(2^{k'}-1)(C_1+C_2)+D_1+D_2+2]\overline{d}$, for any $k' \in \mathbb{N}^+$.

The following theorem states that, under the aforementioned constraints, RM_I implements the timely reliable multicast service specification $\mathrm{RM}_S(\Delta)$, for $k \in \mathbb{N}^+$ and $\Delta = \texttt{DET-BOUND} + \texttt{REC-BOUND}(k^*+k)$.

Theorem 4. *For $k \in \mathbb{N}^+$ and $\Delta = \texttt{DET-BOUND} + \texttt{REC-BOUND}(k^*+k)$, it is the case that $C\text{-}attraces_k(\mathrm{RM}_I) \subseteq attraces(\mathrm{RM}_S(\Delta))$.*

6 Contributions & Future Work

The contributions of this paper are several. First, we present a timed I/O automaton model of the reliable multicast service. This model formally specifies the behavior of several reliable multicast protocols that strive to provide eventual delivery with, possibly, some timeliness guarantees. In particular, it dictates what it means to be a member of a reliable multicast group and which packets are guaranteed delivery to which members of the reliable multicast group. Moreover, we present a timed I/O automaton model of the SRM protocol. This model decomposes the functionality of the reliable multicast service, thus facilitating reasoning and the future modeling of either variations and extensions to SRM's recovery scheme, or other reliable multicast protocols altogether. We show that our model of SRM is safe, in the sense that it may only deliver appropriate packets to each member of the reliable multicast group. We also show that, under certain constraints, our implementation is live, in the sense that it guarantees the timely delivery of the appropriate packets to each member of the reliable multicast group.

In the future, we intend to relax the constraints used in our liveness analysis of SRM and to analyze the performance of SRM in the context of a dynamic group membership. We also intend to model, analyze, and compare the performance of extensions to SRM and other reliable multicast protocols. The safety analysis of each such protocol will guarantee that the protocols are compared on an equal footing; something rarely done precisely when comparing protocols.

Acknowledgments We thank Idit Keidar and the reviewers for helpful comments and suggestions.

References

[1] Floyd, S., Jacobson, V., McCanne, S., Liu, C.G., Zhang, L.: A Reliable Multicast Framework For Light-Weight Sessions And Application Level Framing. IEEE/ACM Transactions on Networking **5** (1997) 784–803
[2] Holbrook, H.W., Singhal, S.K., Cheriton, D.R.: Log-Based Receiver-Reliable Multicast For Distributed Interactive Simulation. In: Proc. ACM/SIGCOMM'95. (1995) 328–341
[3] Lin, J.C., Paul, S.: RMTP: Reliable Multicast Transport Protocol. In: Proc. IEEE/INFOCOM'96. Volume 3. (1996) 1414–1424
[4] Paul, S., Sabnani, K.K., Lin, J.C., Bhattacharyya, S.: Reliable Multicast Transport Protocol (RMTP). IEEE Journal on Selected Areas in Communications **15** (1997) 407–421
[5] Papadopoulos, C., Parulkar, G., Varghese, G.: An Error Control Scheme For Large-Scale Multicast Applications. In: Proc. IEEE/INFOCOM'98. Volume 3. (1998) 1188–1196
[6] Li, D., Cheriton, D.R.: OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol. In: Proc. IEEE/ICNP'98. (1998) 237–245
[7] Livadas, C., Lynch, N.A.: A Formal Venture into Reliable Multicast Territory. Technical Report, Lab. for Computer Science, MIT (2002)
[8] Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann Publishers, Inc. (1996)
[9] Liu, C.G., Estrin, D., Shenker, S., Zhang, L.: Local Error Recovery in SRM: Comparison of Two Approaches. IEEE/ACM Transactions on Networking **6** (1998) 686–692