

Ordered End-to-End Multicast for Distributed Multimedia Systems*

H.-Peter Dommel
Santa Clara University
Computer Engineering
Santa Clara, CA 95053
hpdommel@scu.edu

J. J. Garcia-Luna-Aceves
School of Engineering
University of California
Santa Cruz, CA 95064, USA
jj@cse.ucsc.edu

Abstract

We address the problem of message ordering for reliable multicast communication. End-to-end multicast ordering is useful for ensuring the collective integrity and consistency of distributed operations. It is applicable for distributed multiparty collaboration or other multipoint applications, where the ordered reception of messages at all hosts is critical.

Existing reliable multicast protocols largely lack support for ordering. Our novel mechanism can be added to existing reliable multicast services at low cost by performing cascaded total ordering of messages among on-tree hosts en route from senders to receivers. The protocol operates directly on a given end-to-end multicast tree, contrasting other tree-based approaches requiring a separate propagation graph to be built to compute ordering information. For better load distribution, resilience, and ordered subcasting of messages within multicast groups, sequencer nodes are elected dynamically based on address extensions to hosts in the multicast tree.

A taxonomy of broadcast and multicast ordering solutions and comparative cost analysis show that reliable message delivery integrated with staggered ordering in end-to-end multicast trees is more efficient, scalable, and less costly to deploy.

Keywords: Tree-based ordered reliable multicast, group communication support for networked multimedia systems

1 Introduction

IP multicast communication [10] generalizes the point-to-point and broadcast communication model to multipoint dissemination of messages. A source must send a packet only once to the network interface and packets are transparently replicated on their transmission paths to the receivers. This form of communication is indispensable for networked applications with high-volume data transfer, such as distributed software updates, news casts, video-on-demand, and interactive applications, for example distributed simulations or telecollaboration systems. Data handled by these applications fall into two categories, continuous media streams and non-real-time data. Real-time data delivery, e.g., for video or audio streams, is typically best-effort and unordered, but must observe deadlines to be useful for an application. Non-real-time packets carry discrete data, and may require reliable, ordered delivery based on the application semantics.

Changes in datagram routing or transmission errors may cause packets to arrive at their destination out-of-sequence. Disordered delivery of packets in a distributed application may result in different views of the group state at end-hosts. Ordering of messages compensates for the lack of a global system state and the effects of asynchrony, unpredictable network delay, and disparities in host processing in distributed communication. It warrants that destination processes observe the same order of reception of messages. Ordering is complemented by reliability and atomicity. Reliability guarantees that messages eventually arrive correctly at their destinations, and atomicity guarantees that a message is received by all members of a multicast group or none.

Consider a distributed interactive simulation with many moving, interacting entities, where a message m_1 is reliably multicast from source s_1 to receiver group Rec_1 , and m_2 is reliably multicast from s_2 to Rec_2 . A host which belongs to Rec_1 may receive message m_1 before m_2 , while another

*This work was supported by the Defense Advanced Research Projects Agency (DARPA) under grant F19628-96-C-0038.

host belonging to both groups may receive the messages in the opposite order. Correct operation of the simulation system requires not only that the input stream is equivalent for all replicas, but all input events have to be delivered to replicated instances of shared applications in the same order. Some ordering protocol must intercept, or better, be integrated in the delivery process to guarantee such consistency.

The majority of existing reliable multicast solutions [23] lack ordering services. A comparison of the performance characteristics of such protocols [20], entailing sender- or receiver initiated protocols, ring- or tree-based protocols, and tree protocols with negative acknowledgments and periodic polling, showed that the latter protocol type is the most scalable and efficient approach known to date among deployable systems. Based on these observations, our objective is to examine how ordering services can be integrated with reliable multicasting, in particular with tree-based protocols, preserving scalability and efficiency. We describe a solution for this problem using the idea of staggered ordering of messages on their delivery paths from sources to receivers in the reliable multicast tree, which is also used for logical connectivity between hosts for the purpose of error recovery. In contrast to earlier work, our protocol does not require construction of a separate logical propagation graph or global clock synchronization, and ordering is distributed across nodes on the delivery paths between sources and receivers in the multicast tree.

The paper is structured as follows: Section 2 introduces relevant terms and our system model. Section 3 presents a description of the TOM (Tree-based Ordered Multicast) protocol. Section 4 introduces a taxonomy for ordering schemes and discusses performance figures for contending solutions, making the case for tree-based ordered multicast. Section 5 reviews related work, and conclusions are offered in Section 6.

2 System Model and Assumptions

Our network model (H, C) consists of a set of n hosts H and communication links C , communicating via message passing in the absence of physical clock synchronization. A host is equated with the processes running on it. A multicast group is a set of k hosts in a network of H hosts, which is addressable collectively by a unique group address.

Message dissemination is assumed to be genuine multicast, i.e., a source sends a message m once to the network interface in a multicast enabled backbone, which replicates m at multicast enabled routers on its path to $r \leq n$ receivers. This stands in contrast to most prior work on ordered multicasting assuming either unicast, where a message must be sent r times from a source to the network interface to reach $r < n$ receivers, or broadcast, where all n

hosts in the network are addressed and designated receivers must filter out messages targeted at them.

Four cases of group connectivity can be observed: 1) from a single source s to a single group g , denoted as (s, g) , or 2) to multiple groups G , (s, G) , or from multiple sources S to 3) a single group, (S, g) , or 4) to multiple groups, (S, G) . Cases 1) and 2) have a simple solution: sequence numbers fixing the ordering relation are added to outgoing messages at the source and are delivered in that order at the destinations. Cases 3) and 4) are more difficult to implement, because sending messages from one host is independent from other hosts, whereas reception of the same messages may be interdependent and destination groups may overlap. We are interested in totally ordered multicast from multiple sources to multiple receivers or receiver groups. We assume that hosts do not fail and network partitions do not occur. Although very specific, we consider overlapping groups for our protocol, because it was also a focal point in previous work on ordered multicast [13, 14, 16]. Hosts in the intersection of two overlapping multicast groups should receive a messages only once, if this message is sent to both groups.

In *total order*, two messages m_1 and m_2 are sent to a receiver set Rec in the same relative order. For example, if two sources, A and B, send messages m_1 and m_2 to receiver groups G_1 and G_2 , respectively, then hosts in both groups, in particular in the intersection $G_1 \cap G_2$, should receive both messages either in the order (m_1, m_2) , or (m_2, m_1) . *Atomic order* demands that either all or none of the hosts in Rec receive the messages. A weaker notion of total order is causal order, based on Lamport's "happened before" relation [19]. While a causal precedence relation between two messages preserves their sending order at delivery time, messages without causal linkage may still delivered to different hosts in different order. We assume that all logical point-to-point channels between any pair of hosts are FIFO, which prevents that an earlier message by the same process is overtaken in delivery by a later message. If not provided by the network layer, FIFO-delivery over non-FIFO channels can be implemented by having the source process add a sequence number to its messages and let destinations deliver according to such sequence numbers [4].

Finally, we assume that a reliable, unordered multicast protocol is running at every host providing reliable delivery of a message to all operational hosts in a target multicast group. Ordered multicast should be *host minimal*, i.e., no other hosts should be affected by multicast of a message than the source and receivers, and *message minimal*, i.e., the message size is a function of the size of the receiver set and not of an entire session or network [27]. Total order multicast in a broadcast model is for instance not host minimal. Looking at end-to-end debates [8, 28], we subscribe to the view, that ordering can be provided as middleware com-

plementing reliable multicasting to motivate reusable coding and easier deployment, as shown in Figure 1. We justify this approach based on the observation that many networked multimedia applications are based on similar media characteristics and delivery semantics. In contrast, applications such as the MBone whiteboard tool [12] provide application-level ordering of messages.

application layer services		
TCP	ordered multicast	reliable multicast
IP unicast routing	IP multicast routing	
lower layer network services		

Figure 1. Network protocol stack subsuming ordered multicast.

3 TOM Protocol Description

The Tree-based Ordered Multicast (TOM) protocol relies on an underlying reliable multicast tree for propagation of ordering information besides acknowledgments and retransmissions. This tree is assumed to approximate the underlying multicast routing tree, which for the Internet is built using various protocols such as DVMRP, CBT or PIM-SM (cf. [15] for a general overview). For the following description, we assume that hosts do not fail and network partitions do not occur. Trees can be constructed per source, which amortizes itself only for long-lived or large-volume transmissions, or dissemination can be based on a shared tree, across which (negative) acknowledgments are relayed between hosts. In such a tree, sources may change frequently, only one collective infrastructure must be maintained, and a source need not know the identity of all receivers in the multicast group. However, the paths from sources to receivers may be suboptimal.

It is unimportant for the description of the ordering mechanism, which reliable multicast protocol is used. It is also not crucial, whether the end-to-end multicast tree is source-based or shared, but we will exemplify how TOM operates to provide total order in a shared tree. The key idea in TOM is to multicast a message from a source to a receiver set combined with sending ordering information for the message (sequence numbers or time stamps) to a common node on the tree elected as ordering node for this receiver set (or multicast group). The ordering node sequences messages assigned to it and multicasts binding sequence numbers for final delivery to the receiver set, where pending messages are to be delivered. TOM can be deployed in the form of an API accessible to applications with ordering needs.

3.1 Data Structures

An host in the multicast tree is either a source node (SN), an extra node (EN), a primary node (PN), an ordering node (ON), or a receiver node (RN). Since every host in the multicast session runs the ordering protocol, roles are assumed on-the-fly and no dedicated hardware is needed. SN emit messages to one or more multicast groups in a session. EN are nodes, which are not a member of the receiver set for a message, relaying messages upward or downward in the tree without participation in the ordering process. PN are hosts on the upward ordering path from SN to ON, aggregating the control messages in local order and forwarding revised sequence numbers up in the tree. The ON is the sequencer node for a message, gathering sequence number bids set on route by PN, deciding on a globally valid number, and multicasting the message to the receiver set with a final and binding sequence number directive. Sources can be ON, as well. RN are message recipients, delivering them according to an ON-sanctioned sequence number. Nodes can be SN for their own messages and assume all other roles for other messages. Edges in the acknowledgment tree point from children nodes to their parents.

A TOM message $m = (m^h, m^b)$ consists of a control header m^h and body m^b , with

$$m^h = (SN_id, Rec, seq\#, ts, of)$$

where SN_id is the source identifier, Rec is the target receiver set (which is either a multicast group, or a collection of individual node identifiers), $seq\#$ is the sequence number used for ordering, ts is an optional timestamp for ordering using timing information at nodes, and of is the ordering flag indicating that a binding sequence number for the message has been set. m^b contains the actual data stream.

Each node maintains two message windows for ordering: a window for unordered messages (uw), which have been received, but whose delivery is pending, and an ordered messages window (ow) for messages, which are correctly ordered and can be delivered to local processes. The sizes of these buffers are limited by the number of hosts in the largest multicast group known at the time of buffer allocation. Each host programs its local network interface to subscribe to multicast packets on the same local network, or to receive packets from routers based on IGMP information (cf. [15]).

3.2 Operation

TOM performs message ordering in four steps: 1) a message multicast from each SN to receivers; 2) a control message unicast from SN across PN to the ON for the designated multicast group or transmission, where PN aggregate

messages from their subtrees and hence stagger the ordering process upward in the tree; 3) determination of a binding sequence number for this message and a multicast to the receiver group; and 4) the delivery of messages at end hosts according to the agreed-upon sequence numbers. The goal is to deliver messages consistently in an order all hosts agree to, without requiring sources to know the constituency of the receiver set. Multicast group information is assumed to be available from a session directory service.

To allow for selective addressing of hosts and dynamic election of an ON we introduce a labeling mechanism known from multiprocessor routing and recently proposed for reliable multicast in the tree-based protocol Lorax [20], and for multicast routing. Labels allow for open ordered multicast, i.e., addressing of specific nodes in the tree without the need to manifest a separate multicast group or revealing IP-addresses, and facilitate self-routing of messages to their destinations based on prefix comparison. Each node i in the acknowledgment tree is labeled with a unique label $l(i)$, which is the prefix of all children of i . The label alphabet is a set of symbols with a defined order, such as integers or letters with lexicographic order, with the alphabet cardinality corresponding to the tree branching factor B . The heuristics to select an ON is as follows: for each set of messages destined to a particular multicast group or set of hosts, elect as ON the node, whose label is the longest common prefix among all node labels in the receiver set. Each ON gathers sequence number bids set en route by PNs, deciding on a globally valid number, and multicasts the respective message to the receiver set with a final and binding sequence number directive. Figure 2 illustrates the mechanics of TOM.

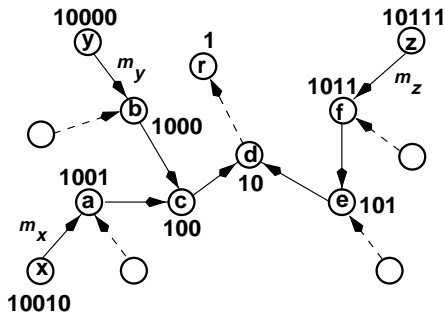


Figure 2. Ordered multicast on acknowledgment tree using address labels (node labels are only depicted if nodes are involved in transmission.)

Node r , as the root of the tree, carries label 1. Node d is the only child in this multicast session, carrying the prefix of its parent r concatenated with its own index 0. All three sources of messages, nodes x , y and z have labels of length 5, being positioned at depth 5 in the tree. The key

idea with using labels for the ordering procedure is to create a confluence of messages at strategically optimal nodes in the tree for ordering a number of messages arriving in the same time window. Rather than depending on a statically assigned ordering node, ON is dynamically selected per transmission as the node with the longest common prefix among the sources of pending messages in the targeted multicast group, without the need to pass an election token among nodes.

Consider the case that x, y and z want to multicast messages to a multicast group $Rec = \{x, y, z, a, b, c, d, e, f\}$. Each source multicasts its message to Rec , where it is entered in the order of collective arrival into uw . Control messages m_x^h and m_y^h are routed from SN x and y , respectively, across their parents to the first common prefix node c , are intermittently ordered at c and, with revised sequence numbers, percolated up in the tree to node d , where message header m_z^h is also arriving. At any node on the path, a bitmask operation on the matching prefix indicates, which messages must be up-routed or handled locally. At d it is determined that its label 10 matches the longest common prefix of SN labels $l(x), l(y)$ and $l(z)$. Hence, $ON(m_x, m_y, m_z) = d$ and node d sequences and multicasts updated message headers to Rec to signal that the associated messages can be delivered. Once each receiver in Rec receives the ordering information per message m with $of = true$ from ON, it shifts m into the ow , where the heading element is delivered to end-processes first.

Similarly, messages to a multicast group located in a left subbranch of the acknowledgment tree can be handled locally by the ON of that group, without affecting any nodes in other segments of the tree. The only overhead incurred in the ordering process is the control message unicast from SNs to some ON, plus one multicast to the receiver set. Total order is hence achieved in a diffusing computation, where the ordering process is carried out along with the message multicast, but neither are receiver nodes burdened with sorting out messages, nor do they have to know the identity of ON. Through the percolation process from SN to ON, usage of the same sequence number for a specific message to all receivers in a multicast group is guaranteed.

Labels allow open ordered multicast, i.e., addressing of specific nodes in the tree with an ordered message sequence without the need to manifest a separate multicast group, and for self-routing of messages to their destinations based on prefix comparison. Figure 3 specifies the ordering algorithm of $TOM()$ that an ontree host i may use to send a message m totally ordered to a receiver set Rec (hosts are assumed to carry prefix labels). Procedure $TOM_send()$ multicasts a message to the receiver set and unicasts the control header towards the dynamically elected ON; $TOM_cast()$ self-routes messages to a receiver based on prefix labels; and $TOM_receive()$ checks, whether a node is EN, PN, ON,

or RN and takes according actions:

```

proc TOM (node i)
Cobegin
  TOM_send(); TOM_receive()
Coend
proc TOM_send (message  $m^b$ , receivers Rec)
Begin /* i is SN */
  If  $m \neq 0$  And  $i = \text{SN}(m)$ 
    Then  $m^h = (l(i), \text{Rec}, \text{seq\#}, \_ , \text{false})$ 
       $m = (m^h, m^b)$ 
      reliable_multicast(m, Rec)
      TOM_cast( $m^h$ , parent(i))
End
proc TOM_cast (message m, receiver rec)
Begin /* self-route from node i to rec */
  If  $|l(i)| > |l(\text{receiver})|$ 
    OrIf  $l(i) \neq \text{prefix}(\text{rec})$ 
      Then If  $\exists$  parent(i)
        Then unicast(m) to parent(i)
      Else If  $\exists$  children(i)
        Then unicast(m) to child(i)
          where  $l(\text{child}(i)) = \text{prefix}(l(\text{rec}))$ 
End
proc TOM_receive (message m, receivers Rec)
Begin
  If  $i \notin \text{Rec}(m)$  /* i is EN */
    Then unicast(m) to parent(i);
  ElseIf  $\text{of}(m) = \text{false}$  And  $m^b = 0$  /* i is PN */
    Then tag m with new seq#;
      TOM_cast(m, parent(i))
  ElseIf  $l(i) \in \text{Rec}$  /* i is RN */
    Then If  $m^b \neq 0$  And  $\text{of}(m) = \text{false}$ 
      Then insert ( $m^b, uw$ )
      Else shift m from  $uw$  to  $ow$ 
      deliver(head( $ow$ ), local processes)
  Else compute longest common prefix lcp = (m, pm)
    If  $lcp = l(i)$  /* i is ON */
      And (query parent(i) for pending msgs = 0)
      Then Forall msgs in  $uw$  select seq#s
        shift msgs with set seq#s into  $ow$ 
        TOM_cast(head( $ow$ , Rec))
End

```

Figure 3. TOM procedures for ordered multicast from host i to other hosts and for processing received messages from other nodes for ordered local delivery.

Consider the special case of ordering with this mechanism, when messages must be sent to two different, but overlapping multicast groups, e.g., $G_1 = \{a, b, c\}$ and $G_2 = \{c, d, e, f\}$, with $G_1 \cap G_2 = c$. Nodes in each group must receive a given message sequence in total order, and node c shall not receive contradictorily ordered messages. This case can be solved, if individual membership in target groups is known. Instead of choosing the node with the longest common prefix as ON, the nodes with multiple membership will then be the ordering cores for a transmis-

sion, prescribing their sequencing decisions to their respective ON. In our case, c will be instrumental in informing d about the sequence in group G_1 , such that d can construct a sequence from this that is compatible with G_2 .

While total order of messages within one or more destination multicast groups is ensured, causal order among messages is not preserved in the above algorithm. To provide causality, the sequence numbers of messages to be ordered must encode causal dependency information before reaching ON. This can be achieved for instance by Lamport clocks maintained by all nodes belonging to a multicast group, and updating sequence numbers in the staggered ordering process to preserve the causal relations. To implement atomicity in delivery, that is, either all RN in $\text{Rec}(m)$ receive message m or none, another message exchange between all RN and ON must be introduced, where all RN signal reception of m and m^h to ON, and ON must send another `ok_to_deliver(m)` signal for RN to collectively proceed with delivery.

Resilience is another important aspect in TOM operation that we only briefly discuss for space reasons. Ordering can be linked with several types of reliability [13], including 1) giving no guarantee on the reliability of ordered deliveries, 2) assuming only inconsistent deliveries with failed hosts, 3) inciting roll-backs at operational hosts to repair inconsistent deliveries, and 4) the assumption that inconsistencies never happen. Furthermore, another set of choices addresses the time it takes to deliver a message, and to which recipients the delivery guarantee extends. In the event of host or link failures, the ordering tree may be partitioned into subtrees, each of which may continue to run TOM. A vanished ON will be replaced by the next common node in the destination set according to the label semantics. In operational subgroups, the semantics of reliable delivery is preserved for all multicast operations. Failure and recovery events must be made known to all operational hosts in an ordered fashion. Partitioned subbranches of the ordering tree may rejoin as soon as communication paths between them are reestablished. A link failure is detected, when a host fails to probe a neighbor node on the tree before expiration of a local timer. A host failure is detected, when a host with a pending queue of messages does not receive an expected message within a timeout period.

4 Taxonomy and Performance Comparison

We classify predominant ordering paradigms using reliable broadcast or multicast into two main classes, as depicted in Fig. 4: 1) geometry-independent protocols such as *symmetric*, *two-phase* or *centralized* solutions, and 2) geometry-dependent protocols such as *ring-based* and *tree-based* solutions. Some schemes may involve all hosts in the ordering process in a decentralized way, using message

stability properties, versus solutions that burden one or a few hosts with the responsibility to order messages on behalf of the hosts in a multicast group. The main problem in the first case is to reach consensus among hosts on ordering patterns, the problem in the second case is to elect sequencer nodes. Our taxonomy contrasts the distinction between symmetric and token-site algorithms proposed by Rodrigues *et al.* [26], which does not accommodate methods that are neither symmetric nor based on token-passing, such as tree-based ordering.

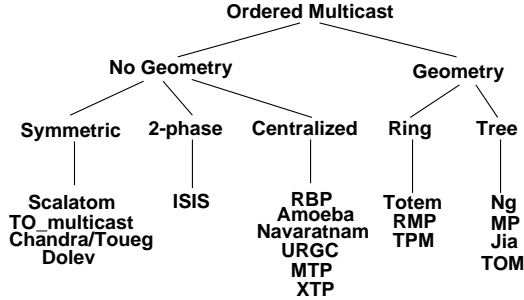


Figure 4. Taxonomy of ordered multicast solutions.

We evaluate the processing load X at involved hosts and the message overhead M required to successfully multicast a message in order from a source to all receivers. We assume IP-multicast as the dissemination model for all schemes, although all schemes except TOM have been proposed in broadcast systems. The goal of this comparison is not an elaborate modeling of the many possible nuances and optimizations of ordering schemes in conjunction with reliable multicast, but rather a plain comparison of the fundamental working structure of ordering solutions. To this end we do not include loss probabilities and assume that all schemes consistently use *sender-initiated* or *receiver-initiated* error recovery [9]. Sender-initiated models place the burden for processing acknowledgments and requests for corrupt or lost packets on the transmission source, opposite to receiver-initiated solutions, where retransmissions are handled in local groups among receivers and sources are contacted only in cases of unrecoverable packet loss. Receiver-initiated protocols achieve better scalability because a source is likely only contacted in case of packet loss.

The notation used is as follows: s is the number of sources transmitting a message m destined to the same receiver(s) at a given time (each sender is assumed to be receiver); r is the number of receivers of m in the receiver set $Rec(m)$; X_f is the time to feed a packet from a higher protocol layer; X_p is the time to process the transmission of a packet (including retransmissions); $X_{\#}$ is the time to process a sequence number check; Y_p is the time to process

a newly received packet; Y_f is the time to deliver a packet to an end process; X^ω is the processing overhead per message in protocol $\omega = \{S, 2P, C, R, T^{MP}, T^{MG}, T^{TOM}\}$. Source nodes are denoted as SN, ordering nodes as ON, and receiver nodes as RN (the detailed node semantics may vary among protocols). M is the number of transmissions for all receivers to receive a message orderly.

4.1 Geometry-Independent Protocols

Reliable broadcast solutions are largely designed for fault-tolerant, asynchronous distributed systems. Such protocols are geometry-independent, i.e., all hosts are assumed to be fully connected with each other, and routing between hosts does not presume any prearranged host geometry. We subsume symmetric, two-phase and centralized solutions under this paradigm. Centralized ordering could also be classified as a star-geometry, but the central node is typically chosen *ad hoc* based on some election or token-passing scheme among all nodes.

4.1.1 Symmetric Ordering

In *symmetric* schemes (S) [6, 11, 27], all hosts partake in the ordering process in a decentralized way, analogous to a voting process, using message stability properties. SN disseminate messages reliably to all hosts, which assign a timestamp to each message and place it in a pending buffer; for each message m , participant hosts (SN and RN) agree on a unique order number using timestamp information by running a consensus protocol; a message with an assigned order number is shifted to the delivery queue and delivered to end processes in the globally binding order. Thus the number of messages to be exchanged is a function of the hosts in the system involved in the ordering process. With X_c denoting the extra cost for the consensus protocol, the expected overhead of a generic symmetric protocol at SN and RN is

$$\begin{aligned}
 X_{SN}^S &= X_f + rX_p & (1) \\
 X_{RN}^S &= s(Y_p + X_{\#} + rX_c + Y_f)
 \end{aligned}$$

With broadcast communication, a source node sends a message to $r - 1$ receivers, which in turn send $r - 1$ messages to agree on the final sequence number, i.e., $M_{BC} = s((r - 1) + r(r - 1))$, that is $O(sr^2)$ for s sources. With multicast and $r < n$ receivers, $M = s(1 + 2r)$, that is one multicast message to all receivers, one multicast per each of the r receivers to each other, and one timestamp sweep from all receivers to the source. Protocols with fault-tolerance measures may incur significantly higher cost [27].

4.1.2 Two-phase Ordering

In *2-phase* ordering (2P) [5], four communication steps are required: a source sends a message m to a multicast group, where each receiver assigns a priority number to the message, places m as pending in its local queue, and returns the priority number to the source. The source selects the highest number and sends it to all receivers, which replace the original number with the new one, tag the message as deliverable, reorder the queue and deliver messages heading the queue. The expected overhead at SN and RN is

$$\begin{aligned} X_{SN}^{2P} &= X_f + r(Y_p + X_{\#} + 2X_p) \\ X_{RN}^{2P} &= s(2Y_p + X_{\#} + X_p + Y_f) \end{aligned} \quad (2)$$

If we assume $r \geq s$, then $X^{2P} = \max(X_{SN}^{2P}, X_{RN}^{2P}) = O(r)$. With multicast, one message from s sources to r receivers, r control messages with priority numbers back to each source, and one final control message multicast from the source to the receiver set for each message are required, i.e., $M = s(1 + r)$.

4.1.3 Centralized Ordering

In *centralized* ordering (C) [3, 7, 21], a source SN transmits a message m to a sequencer host, which assigns a unique number to m and forwards it to the receiver set $Rec(m)$, where it is ultimately delivered to end processes in the order prescribed by sequence numbers. The sequencer role may rotate among hosts. The expected overhead at SN, ON, and RN is

$$\begin{aligned} X_{SN}^C &= X_f + X_p \\ X_{ON}^C &= s(Y_p + X_{\#} + rX_p) \\ X_{RN}^C &= s(Y_p + Y_f) \end{aligned} \quad (3)$$

Hence $X^C = O(sr)$, and $M = s + r$, consisting of s messages from sources to ON, and one multicast per message from ON to all receivers. If SN = ON, we spare one step.

4.2 Geometry-Dependent Protocols

Geometry-dependent protocols presume a specific host topology to route ordering information.

4.2.1 Ring-based Ordering

In *ring-based* ordering (R) [2, 25, 31], a logical ring imposes a transmission path between hosts, where each host needs only communicate with its predecessor and successor in the ring. To multicast a message, a host must possess the token; the token contains requests for messages to be resent and the highest sequence number for any message broadcast on the ring; each host maintains an input buffer containing pending messages with assigned sequence numbers; on receipt of the token, the host completes processing of messages in its buffer by adjusting sequence numbers,

resends messages requested in the token, updates the token information and forwards the token; messages are sent to end processes, when marked as deliverable. Each SN, as token-site, assumes the role of ON. With X_{tk} indicating the token transfer time, the expected overhead at SN and RN in a single ring is

$$\begin{aligned} X_{SN}^R &= X_f + X_p + r(Y_p + X_{\#} + X_p) + X_{tk} \\ X_{RN}^R &= s(Y_p + X_{\#} + Y_f) \end{aligned} \quad (4)$$

Hence $X^R = O(r)$, if $r > s$, and the message overhead is at best $M = 2n/k$, where $2n$ is the number of token transfers required to accept k multicast messages in a ring of n nodes [25]. With $k = 1$, s sources, and despite $r < n$ receivers, $M = 2sn$.

4.2.2 Tree-based Ordering

For *tree-based* ordering (T), we compare the MP protocol by Garcia-Molina and Spauster [13], and the metagroup approach (MG) by Jia [16, 29] with TOM. Known tree-based reliable multicast protocols [20, 24, 32] do not feature ordering. Common to MP, MG and TOM is the idea of distributing ordering responsibility and load across several nodes on the tree. While MP and MG use group membership information to cluster nodes for optimized message delivery, TOM uses the end-to-end multicast topology.

The MP protocol has two work phases: 1) the transmission from the source to a primary host, and 2) the transmission from this host to the receivers. It builds a forest of propagation trees, where hosts in the intersections of multicast groups are chosen as hop nodes, i.e., roots of subtrees. A message is first sent to these primary hosts, and then propagated downward in the tree toward the receiver hosts, being ordered on their propagation path, and finally unicast to the receiver hosts. The MG protocol clusters hosts from overlapping multicast groups into metagroups, which do not overlap. Each group has a primary metagroup (PM), and in each metagroup one member is assigned to be manager. Metagroups are organized in a forest of propagation trees, such that the PM of a group is the ancestor of all other metagroups of the same group in the tree. Messages destined to multicast group G are first sent to $PM(G)$, which propagates the messages along the tree to all other metagroups, which are subsets of G . The manager of a metagroup broadcasts a message to other members in its metagroup.

The drawback with MP and MG is the need to compute a logical propagation or metagroup tree per source as overlays to the end-to-end geometry, which means that in order to construct such a tree, the computation host(s) must know the membership of all groups. This approach works only for closed multicast and static groups, and amortizes itself only for long-lived transmissions between hosts. The processing

overhead common to all tree-based schemes is

$$\begin{aligned} X_{SN}^T &= X_f + X_p \\ X_{ON}^T &= B(Y_p + X_{\#} + X_p) \\ X_{RN}^T &= Y_p + Y_f \end{aligned} \quad (5)$$

Hence generally $X^T = O(B)$, where B indicates the branching factor of the tree. With multicast, $M^{MP} = s(1 + d)$ messages are required - one message from each of the s sources to the primary destination in the subtree, and one broadcast at each level of the subtree, where d is the subtree depth [13]. MG has three work phases and requires one message to PM(G), d messages to the managers of the deepest metagroups at depth d in the subtree, and another k messages to the members of the k metagroups containing the target multicast group, i.e., $M^{MG} = s(1 + d + k)$ [29]. TOM requires a multicast from s SNs to the receiver set, and p unicasts from the SN to the ON, where p is the average path length, and one final multicast from ON to RN, i.e., $M^{TOM} = s(2 + p)$.

4.3 Results

Table 1 summarizes expected message costs and delays. Centralized and two-phase approaches incur only two respectively three message exchange phases, but messaging is concentrated on specific hosts in the session, which may become a bottleneck or fail. Rings engage all hosts in a session in the transmission process, even when a source and multicast receiver group constitute only a small portion of the entire session. Trees allow for selective engagement of hosts on those subbranches or local groups, which are actually affected by the message processing.

Protocol	X	M
Symmetric	$O(sr^2)$	$s(1 + 2r)$
Two-Phase	$O(r)$	$s(1 + r)$
Centralized	$O(sr)$	$s + r$
Ring-based	$O(r)$	$2sn$
Tree-based: MP	$O(B)$	$s(1 + d)$
Tree-based: MG	$O(B)$	$s(1 + d + k)$
Tree-based: TOM	$O(B)$	$s(2 + p)$

Table 1. Average processing overhead X and multicast message cost M .

We assume that there are as many sources as receivers, $r = n$ and $s = 1$. In the graph, we neglect the cost to compute and maintain the propagation infrastructure, which may be substantial for MG and MP in comparison to TOM, which simply relies on a given acknowledgment tree. We vary the session size between $n = [1, 1000]$, with $r = n/10$

as the average size of a receiver multicast group. The MP tree depth has been projected between $d = [1, 8]$ for simulations with $n = 200$ and average group size $g = [5, 40]$ [13]. The tree depth for a metagroup tree has been projected between $d = [1, 5]$ for up to 40 metagroups with $g = 50$, and an overlapping degree of 10. We also assume that each source sends only one multicast message per transmission cycle. Simulations for the Lorax protocol have indicated that optimal ack trees are built when each nodes supports at least $B = 5$ neighbors [20]. As a baseline comparison, we hence choose the average depth of a subbranch in a MP and MG tree as $d = \log_B r$, where $B = 5$ is the average node degree. The average path length for TOM is chosen as $p = h/2$, because roughly half of the height h of the tree needs to be traversed to converge on a ON. Note that a message comparison provides a limited view on the relative performance of the protocols, because parallelism in message processing, the processing overhead at various nodes, and the shape of the tree would need to be considered in a more precise way. However, concentrating on M alone is sufficient to express fundamental differences between the approaches. Figure 5 plots the multicast message cost of the various schemes under given assumptions.

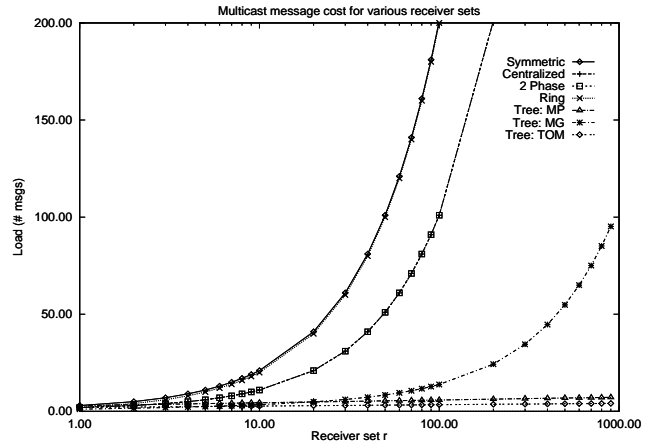


Figure 5. Average message cost with multicast.

The obtained results picture only a special scenario, under which the discussed protocols perform, namely genuine multicast with one source transmitting. The multiple source case would reinforce that the throughput of a generic tree-based protocol for ordered reliable multicast scales better with receiver set due to locus and execution of sequencing. Symmetric methods exhibit the least scalability, because all nodes are involved in processing messages from all other nodes. If all nodes broadcast at the same time, latency may be low, but a consensus protocol must be run. Two-phase, centralized and ring solutions have similar message overhead, however, rings may permit higher concurrency, with

the drawback that latency increases in large sessions. The centralized ordering method is reasonable for few hosts, but is a potential bottleneck and single point of failure, particularly for large sessions. A logical hop between hosts in MP and MG may be multiple hops across long distances in the multicast routing tree, in contrast to TOM, which operates under the assumption that the structure of the ack tree mirrors the path information in the multicast routing tree, rather than using separate propagation graphs. Comparing the three tree-based methods, TOM performs equal or better than MG and MP, spreads the computational load of ordering over multiple nodes in the tree, and is suited well for dynamically changing multicast groups, rather than catering to static membership and long-lived transmissions.

5 Related Work

Much work on total and causal ordering for multicast is centered around fault tolerance or consistency issues in distributed systems. Chandra and Toueg [6] have shown that total order broadcast and consensus are equivalent problems in asynchronous systems. Many protocols from this field suffer from high overhead to achieve fault tolerance by introducing messaging to implement a failure detector and consensus mechanism.

Symmetric or decentralized approaches are based on the total order solution by Lamport [19], where timestamps are assigned to messages at broadcast time. The algorithm by Chandra and Toueg [6] executes reliable broadcast in four communication steps with a weak failure detector, and the solution by Dolev *et al.* [11] is based on a majority consensus protocol. Both schemes have $O(n^2)$ message complexity. Newer approaches such as the TO_multicast protocol [17] or Scalatom [27] incur message complexity $O(r^2)$. Hybrid algorithms using distributed messaging in conjunction with centralized sequencing [26] have been proposed to achieve better scalability.

The ISIS system [5] implemented two-phase ordering using vector clocks for logical time keeping, an explicit membership model and layered microprotocols such as CB-CAST for causal ordering and ABCAST for total ordering. Later versions of ISIS have adopted a ring-based approach.

Centralized reliable broadcast approaches have been proposed in the RBP protocol [7] with a rotating token to identify the sequencer, the Amoeba system [18], where the kernel passes messages to a dedicated sequencing processor, or the protocol by Navaratnam *et al.* [21], where a primary manager orders messages and forwards them to a secondary manager heading each local multicast group delivering to application processes. Centralized reliable multicast protocols such as URGC [1], MTP [3] or XTP [30] follow a similar principle.

Protocols of the ring type, such as Totem [2], RMP [31]

or TPM [25] implement total ordering and feature resilience toward network partitions and process failures. A related approach is the token-bus protocol MLMO [33], which ensures causal and total ordering in a hierarchical infrastructure to connect internetworks.

The tree protocol by Ng [22] implements reliable broadcast with source-ordered delivery to a single group allowing up to k host failures, building a minimum spanning tree between hosts. Total ordering is achieved using two vectors of time-stamps at each host to keep track of the last message sent to and received from each neighbor. A broadcast message with timestamp t is delivered to end processes only if all messages with smaller timestamps have already been received. The MP protocol by Garcia-Molina and Spauster [13] solves single source, multiple source, and multiple group total ordering, including the case that messages are addressed to different, overlapping groups. However, delivery from the last intermediate host to final destinations is unicast, causal ordering is not supported, and reliable failure detection is required for the protocol to operate resiliently. The improvement by Jia [16, 29] clusters hosts into metagroups representing intersections of overlapping groups and forming propagation graphs based on metagroup relations to minimize duplicate deliveries, enhance parallelism in delivery, and shorten propagation graphs. Various tree-based reliable multicast protocols, TMTP [32], RMTP [24], or Lorax [20], have been proposed in recent years. TMTP and RMTP build a source-based tree for flow and error control, and Lorax features positional labels in a shared ack tree for concurrent multicast, but all three protocols lack support for multi-source and multi-group ordering. TOM is geared towards adding ordering to reliable concurrent multicast as in Lorax, but it could also be deployed in TMTP with domain managers, and in RMTP with designated receivers as intermediate ordering nodes.

6 Conclusion

This paper makes the case for adding ordering services to tree-based concurrent reliable multicast, based on the notion that ordered delivery of multimedia data is essential to a growing number of Internet applications supporting telepresence and near-synchronous information sharing. Looking at reliable multicasting for such applications, we observed that ordering services have not been considered as an integrated component in data dissemination. The TOM protocol stands in contrast to previous reliable broadcast solutions tailored to local area networks, where ordering was performed assuming symmetric communication, centralized, ring-based or propagation graph schemes.

The TOM protocol is a first solution working directly on reliable multicast trees, using staggered ordering of messages on their paths from sources to the receivers. Work-

load is hence distributed, the infrastructure used for ordering is cohesive with the one for reliability provision, and the addition of address labels yields efficient ordering for multiple groups and subgroups. Opposite to other prominent solutions, TOM does not require computation of separate graphs for propagating ordering information. It implements ordering in a diffusing computation, where messages are ordered on their delivery paths from sources to receivers, and each node deals only with its children and parent node instead of the entire multicast group. We also proposed a taxonomy for ordering schemes integrating reliable broadcast and multicast solutions. A simple performance comparison showed that ordering in trees surpasses contending solutions in terms of scalability, efficiency and practicality. Extension of this work include an analysis of host processing costs with packet loss probabilities and a closer look at resiliency issues.

References

- [1] R. Aiello, E. Pagani, and G. P. Rossi. Causal ordering in reliable group communication. In *Proc. ACM SIGCOMM*, pages 106–115, San Francisco, CA, Sept. 1993.
- [2] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, and P. Ciarfella. Fast message ordering and membership using a logical token-passing ring. In *Proc. Int. Conf. on Dist. Comp. Sys. (ICDCS)*, pages 551–560, Pittsburgh, PA, May 1993. IEEE.
- [3] S. Armstrong, A. Freier, and K. Marzullo. Multicast transport protocol. RFC 1301, Feb. 1992.
- [4] O. Babaoglu and K. Marzullo. *Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms*, chapter 4, pages 55–96. In: S. Mullender (Ed.) - *Distributed Systems*, Addison-Wesley, 1993.
- [5] K. Birman, A. Schiper, and P. Stephenson. Lightweight causal and atomic group multicast. *ACM Trans. on Computer Systems*, 9(3):272–314, Aug. 1991.
- [6] T. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.
- [7] J. M. Chang and N. F. Maxemchuck. Reliable broadcast protocols. *ACM Transactions on Computer Systems*, 2(3):251–273, August 1984.
- [8] D. R. Cheriton and D. Skeen. Understanding the limitations of causally and totally ordered communication. *Operating Systems Review*, 27(5):44–57, Dec. 1993.
- [9] D. D. Towsley, J. Kurose, and S. Pingali. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *IEEE Journal on Sel. Areas in Comm.*, 15(3):398–406, Apr. 1997.
- [10] S. Deering. Host extensions for IP multicasting. RFC-1112, August 1989.
- [11] D. Dolev, S. Kramer, and D. Malki. Early delivery totally ordered multicast in asynchronous environments. In *Int. Symposium on Fault-Tolerant Computing*, pages 544–553, Toulouse, France, June 1993. IEEE Comput. Soc.
- [12] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level-framing. In *Proc. ACM SIGCOMM*, pages 342–356, Cambridge, MA, Aug./Sep. 1995.
- [13] H. Garcia-Molina and A. Spauster. Ordered and reliable multicast communication. *ACM Trans. on Comp. Sys.*, 9(3):242–271, Aug. 1991.
- [14] R. Guerraoui and R. Schiper. Total order multicast to multiple groups. In *Proc. 17th Int. Conf. on Distributed Computing Systems*, pages 578–585, Baltimore, MD, May 1997. IEEE.
- [15] C. Huitema. *Routing in the Internet*. Prentice Hall, Englewood Cliffs, NJ, 1995.
- [16] X. Jia. A total ordering multicast protocol using propagation trees. *IEEE Trans. Parallel and Distrib. Sys.*, 6(6):617–627, June 1995.
- [17] U. F. Jr., P. Ingels, A. Mostefaoui, and M. Raynal. Fault-tolerant total order multicast to asynchronous groups. In *Proc. 17th IEEE Sympos. on Reliable Distributed. Sys.*, pages 228–234, West Lafayette, IN, Oct. 1998. IEEE.
- [18] M. F. Kaashoek, A. S. Tanenbaum, S. F. Hummel, and H. E. Bal. An efficient reliable broadcast protocol. *Operating Systems Review*, 23(4):5–19, Oct. 1989.
- [19] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Comm. ACM*, 21(7):558–565, July 1978.
- [20] B. N. Levine, D. Lavo, and J. J. Garcia-Luna-Aceves. The case for reliable concurrent multicasting using shared ack trees. In *Proc. ACM Multimedia*, pages 365–376, Boston, MA, Nov. 1996.
- [21] S. Navaratnam, S. Chanson, and G. Neufeld. Reliable group communication in distributed systems. In *Proc. 8th Int. Conf. on Distributed Computing Systems*, pages 439–446, San Jose, CA, June 1988. IEEE.
- [22] T. P. Ng. Ordered broadcasts for large applications. In *Proc. IEEE 10th Symp. Reliable Dist. Sys.*, pages 188–197, Pisa, Italy, Sep. 1991.
- [23] K. Obraczka. Multicast transport protocols: a survey and taxonomy. *IEEE Communications Magazine*, 36(1):94–102, Jan. 1998.
- [24] S. Paul, K. K. Sabnani, J. C.-H. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). *IEEE J. on Sel. Areas in Comm.*, 15(3):407–421, April 1997.
- [25] B. Rajagopalan and P. K. McKinley. A token-based protocol for reliable, ordered multicast communication. In *Proc. of the 8th Symposium on Reliable Distributed Systems*, pages 84–93, Seattle, WA, Oct. 1989.
- [26] L. Rodrigues, H. Fonseca, and P. Verissimo. Totally ordered multicast in large-scale systems. In *Proc. of the 16th Int. Conf. on Distributed Computing Systems*, pages 503–510, Hong Kong, May 1996. IEEE.
- [27] L. Rodrigues, R. R. Guerraoui, and A. Schiper. Scalable atomic multicast. In *Proc. 7th Int. Conf. on Computer Comm. and Networks*, pages 840–847, Lafayette, LA, Oct. 1998. IEEE.
- [28] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Trans. on Comp. Sys.*, 2(4):277–288, Nov. 1984.
- [29] S.-P. Shieh and F.-S. Ho. A comment on ‘A total ordering multicast protocol using propagation trees’. *IEEE Trans. Parallel. and Distrib. Sys.*, 8(10):1084, Oct. 1997.
- [30] W. T. Strayer, B. Dempsey, and A. Weaver. *XTP: The Xpress Transfer Protocol*. Addison Wesley, 1992.
- [31] B. Whetten, T. Montgomery, and S. Kaplan. A high performance totally ordered multicast protocol. In *Theory and Practice in Distributed Systems, LNCS 938*, pages 33–57, Berlin, Sept. 1994. Springer.
- [32] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proc. ACM Multimedia*, pages 333–344, San Francisco, Nov. 1995.
- [33] O. ZeinEldine and H. Abdel-Wahab. Multicasting in interconnected networks. In *Proc. Sympos. Comp. and Comm.*, pages 313–319, Alexandria, Egypt, June 1995. IEEE.