Fast, Flexible, and Highly Resilient Genuine Fifo and Causal Multicast Algorithms

Nicolas Schiper[†] Fernando Pedone[†]

[†]Faculty of Informatics
University of Lugano
6904 Lugano, Switzerland

University of Lugano
Faculty of Informatics
Technical Report No. 2009/003
October 2009

Abstract

We study the fifo and causal multicast problem, two group-communication abstractions that deliver messages in an order consistent with their context. With fifo multicast, the context of a message m at a process p is all messages that were previously multicast by m's sender and addressed to p. Causal multicast extends the notion of context to all messages that are causally linked to m by a chain of multicast and delivery events.

We propose multicast algorithms for systems composed of a set of disjoint *groups of processes*: server racks or data centers. These algorithms offer several desirable properties: (i) the protocols are latency-optimal, (ii) to deliver a message m only m's sender and addressees communicate, (iii) messages can be addressed to any subset of groups, and (iv) these algorithms are highly resilient: an arbitrary number of process failures is tolerated and we only require the network to be *quasi-reliable*, i.e., a message m is guaranteed to be received only if the sender and receiver of m are always up. To the best of our knowledge, these are the first multicast protocols to offer all of these properties at the same time.

1 Introduction

Developing dependable distributed applications is not easy. The complexity stems from the asynchrony and unreliability of typical distributed systems: processes execute at different speeds and may abruptly stop executing their code (i.e., crash). Moreover, messages may be arbitrarily delayed, received out-of-order, and even lost, if the sender or receiver is faulty. To ease the development of distributed systems, several group-communication abstractions have been proposed [4, 8]. Two common abstractions are atomic broadcast and atomic multicast. While in the former messages are addressed to all system members, in the latter messages are addressed to subsets of the system members (i.e, *groups*).

Broadcast and multicast abstractions ensure similar *reliability* guarantees—agreement on the set of messages delivered—but offer various message *ordering* properties. Two of these properties, fifo and causal order, are of special interest: they ensure that a message m is not delivered at a process p that does not know m's *context*, where the notion of context is defined differently for each order property. With fifo order, the context of m at p is the messages that were previously broadcast (or multicast) by m's sender and addressed to p. Causal order extends the notion of context to all messages that causally precede m, i.e., messages that are causally linked to m through a chain of broadcast (or multicast) and delivery events. Fifo and causal order help the programming of distributed applications in various domains such as global snapshot construction [2] and fair resource allocation [10]. Causal multicast may also serve as a building block to implement atomic multicast [16].

Fifo and causal broadcast protocols have been largely studied in the literature. In this paper, we propose fifo and causal multicast protocols for systems composed of a set of disjoint groups (e.g., server racks or data centers), each containing several processes. In particular, we show that mechanisms devised for fifo and causal broadcast protocols are not applicable to multicast protocols. As our main contribution, we propose fifo and causal multicast algorithms that offer several desirable properties. To the best of our knowledge, these algorithms are the first to be simultaneously *fast*, *genuine*, *flexible*, and *highly resilient*, in a precise sense, as we now explain.

First, they are *fast*: messages can be delivered in two communication steps; and we further show that this is optimal. Second, these protocols are *genuine* [7]: (i) to deliver a message m only the sender and the addressees of m participate in the protocol. Third, the algorithms are *flexible* in the sense that a process p may multicast messages to groups p does not belong to, that is, groups are "open". Finally, our algorithms are *highly resilient*: they tolerate an arbitrary number of process failures, and can cope with *quasi-reliable* links which guarantee that if both the sender and receiver of a message m are *correct*, i.e., they do not crash, then m is eventually received.

This is in contrast to several multicast protocols [12, 13, 14, 15, 9], which rely on reliable links—message delivery is guaranteed as long as the receiver is correct, regardless of the correctness of the sender. Reliable links are not a realistic assumption: to send a message m, the machine M_p hosting process p typically inserts m into one (or more) local buffer before m is sent over the wire. Hence, even though p *thinks* that m was successfully sent, m may still be lost in case M_p crashes before m hits the wire.

As discussed in [3], devising *open group* multicast protocols that tolerate quasi-reliable links introduce difficulties as we explain next. Figure 1 illustrates the scenario. Consider some process p that multicasts a message m_1 to some group g_2. Later, p multicasts a message m_2 to groups g_1 and g_2 and crashes. Message m_2 is received by processes in g_1, and since m_2 is the first message multicast from p to g_1, m_2 is delivered by processes in g_1. On the contrary, all messages sent from p to members of g_2 are lost. Note that this can happen because p crashes and links are quasi-reliable.

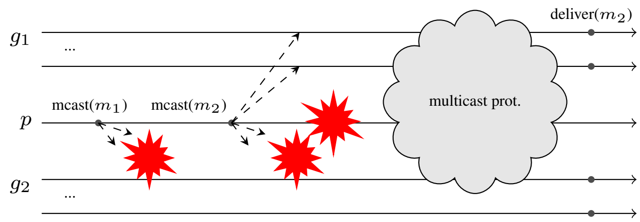

Figure 1: The message delivery order violation problem.

From the reliability guarantees of multicast, correct processes in g_2 must eventually deliver m_2. However, if they do so, the ordering guarantees of fifo and causal multicast will be violated: members of g_2 cannot deliver m_1 before m_2 since m_1 was lost. If messages were broadcast, then m_1 would also be addressed to g_1, and thus, g_1 could help g_2 by forwarding m_1 to g_2. With multicast however, g_1 does not even know about the existence of m_1, since m_1 was not addressed to g_1. In this paper, we propose a mechanism to cope with this problem despite an arbitrary number of process failures and, in contrast to [3], the resulting fifo and causal multicast algorithms are latency-optimal and as latency-efficient as their broadcast counterparts.

The rest of the paper is structured as follows. In Section 2 we discuss the related work. Section 3 presents the system model and some definitions. Sections 4 and 5 respectively provide fifo and causal multicast algorithms; Section 6 shows their latency-optimality. We conclude the paper in Section 7. The correctness proofs of the algorithms can be found in [17].

2 Related Work

Fifo and causal broadcast were originally specified as part of the Isis system [4]. In [8], fifo broadcast is implemented by reliably broadcasting messages along with a sequence number and by delivering messages in the sequence number order.

The first implementation of causal broadcast uses a simple strategy [4]: the causal history of delivered messages is piggybacked on each message to be broadcast. The amount of information contained in messages is thus unbounded. In [14], causal order is ensured differently: messages carry control information in the form of a matrix of counters, where each entry (p, q) denotes the number of messages that were multicast from process p to q in the causal history. This control information is used to know when messages can be safely delivered. This algorithm does not tolerate process failures. A fault-tolerant algorithm that ensures causal order using a similar technique appears in [8]. Although [8] specifies both causal broadcast and multicast, the algorithm given considers the broadcast case only.

In [5], processes may belong to several groups at the same time but messages sent from a process p cannot be multicast to groups p is not a member of. Using the terminology of [6], the protocol in [5] is closed-group. In this algorithm, each message carries a vector

Algorithm	order	type	speed	flexibility	resilience	
			latency	open/closed group	processes	requires reliable network?
[8]	fifo	broadcast	2	-	crash-stop	no
$\mathcal{A}_{fifo}$	fifo	multicast	2	open	crash-stop	no
[12]	causal	unicast	1	-	no failures	yes
[8]	causal	broadcast	2	-	crash-stop	no
[13]	causal	multicast	1	closed	no failures	yes
[14]	causal	multicast	1	open	no failures	yes
[15]	causal	multicast	topology dependent	open	no failures	yes
[9][1]	causal	multicast	1	open	crash-stop	yes
[4]	causal	multicast	2	closed	crash-stop	no
[5]	causal	multicast	2	closed	crash-stop	no
[3]	causal	multicast	4	open	crash-stop	no
$\mathcal{A}_{causal}$	causal	multicast	2	open	crash-stop	no

Table 1: Comparison of the fifo and causal multicast algorithms.

of counters, and this for every group in the system. Messages may be large if the number of groups is high. In contrast, [13] only requires processes to append a vector of counters to messages, where the size of the vector is equal to the number of groups. However, this protocol is not fault-tolerant. In [15], causal separators are used to reduce the amount of control information needed in systems that span several network domains. In [3], the authors propose a more general approach that is topology-oblivious.

The necessary conditions on how much information should be stored at processes and appended to messages to ensure causal order are presented in [9]. This paper also provides an information-optimal algorithm that does not need any a priori knowledge of the communication network. The algorithm in [12] does not append any information on messages but only considers the unicast case and postpones the sending of messages until after all the previous messages sent were acknowledged.

In this paper, we present fault-tolerant and latency-optimal fifo and causal multicast protocols, respectively denoted as $\mathcal{A}_{fifo}$ and $\mathcal{A}_{causal}$. To the best of our knowledge, these are the first algorithms that are at the same time open group, latency-optimal, and tolerate quasi-reliable networks.

Table 1 provides a comparison of the algorithms. The last four columns respectively indicate: the best-case latency of the algorithms, measured in the number of communication delays; whether an algorithm $\mathcal{A}$ allows messages to be multicast from a process p to groups p does not belong to, in which case we say that $\mathcal{A}$ is an open group algorithm, or not, in which case we say that $\mathcal{A}$ is a closed-group algorithm; and the process as well as network failure resilience.

3 System Model and Definitions

3.1 Process groups and Communication

We consider a system $\Pi = \{p_1, ..., p_n\}$ of processes which communicate through message passing. We assume the benign crash-stop failure model: processes may fail by crashing, but do not behave maliciously. A process that never crashes is *correct*; otherwise it is *faulty*. The maximum number f of processes that may crash is not bounded, i.e., $f \leq n$.

The system is asynchronous, i.e., messages may experience arbitrarily large (but finite) delays and there is no bound on relative process speeds. Furthermore, the communication links do not corrupt nor duplicate messages and are quasi-reliable, more precisely: (i) *uniform integrity:* For any process p and message m, p receives m at most once, and only if m was previously sent to p, (ii) *quasi-reliability:* For any two *correct* processes p and q, and any message m, if p sends m to q, then q eventually receives m.

Processes have access to the Θ failure detector that gives possibly inaccurate information about process failures [1]. More precisely, at each process, this oracle outputs a list of processes that are trusted to be alive such that: (i) *completeness:* There is a time after which processes do not trust any process that crashes, (ii) *accuracy:* If some process never crashes then, at every time, every process trusts at least one correct process.

We define $\Gamma = \{g_1, ..., g_m\}$ as the set of process groups in the system. Groups are disjoint, non-empty and satisfy $\bigcup_{g \in \Gamma} g = \Pi$. For each process $p \in \Pi$, $group(p)$ identifies the group p belongs to. For any group g, we denote by Θ_g the failure detector Θ whose output is restricted to g's members.

3.2 Fault-tolerant Multicast Specifications

For each message m, $m.sender$ and $m.dst$ respectively denote the process that multicasts m and the groups to which the message is reliably multicast. Let p be a process. By abuse of notation, we write $p \in m.dst$ instead of $\exists g \in \Gamma : g \in m.dst \wedge p \in g$.

Fifo Multicast Fifo multicast ensures that the delivery order of messages multicast from some process q follows the order in which these messages were multicast. More precisely, uniform fifo multicast is defined by primitives *F-MCast(m)* and *F-Deliver(m)*, and satisfies the following properties [8]: (i) *uniform integrity:* For any process p and any message m, p F-Delivers m at most once, and only if $p \in m.dst$ and m was previously F-MCast, (ii) *validity:* If a correct process p F-MCasts a message m, then eventually all correct processes $q \in m.dst$ F-Deliver m, (iii) *uniform agreement:* If a process p F-Delivers a message m, then eventually all correct processes $q \in m.dst$ F-Deliver m, (iv) *uniform fifo order:* If a process p F-MCasts a message m before F-MCasting a message m', then no process in $m.dst \cap m'.dst$ F-Delivers m' unless it has previously F-Delivered m.

Causal Multicast Causal multicast ensures that messages are delivered in an order consistent with causality. Causality between multicast messages is defined by means of Lamport's transitive happened before relation on events [10]. Here, events can be of two types: the *causal multicast* of some message m, C-MCast(m), or its delivery, C-Deliver(m). The relation is defined as follows: $e_1 \rightarrow e_2 \Leftrightarrow e_1, e_2$ are two events on the same process and e_1 happens before e_2 or $e_1 =$ C-MCast(m) and $e_2 =$ C-Deliver(m) for some message m. Uniform causal multicast satisfies the uniform integrity, validity, and uniform agreement property of fifo multicast as well as [8]: *uniform causal order:* For any messages m and m', if C-MCast(m) $\rightarrow$ C-MCast(m'), then no process $p \in m.dst \cap m'.dst$ C-Delivers m' unless it has previously C-Delivered m.

[1] The algorithm in [9] tolerates process crashes and has a latency of 1 message delay. This does not contradict the lower bound of two message delays we show in this paper. Indeed, two message delays is minimal for algorithms that tolerate quasi-reliable links. However, the algorithm in [9] requires reliable links.

Let $\mathcal{A}$ be an algorithm solving fifo/causal multicast. We define $\mathcal{R}(\mathcal{A})$ as the set of all admissible runs of $\mathcal{A}$. We require fifo/causal multicast algorithms to be *genuine* [7]: An algorithm $\mathcal{A}$ solving fifo/causal multicast is said to be *genuine* iff for any run $R \in \mathcal{R}(\mathcal{A})$ and for any process p, if p sends or receives a message then some message m is F-MCast/C-MCast and either p is the process that F-MCasts/C-MCasts m or $p \in m.dst$.

4 Fifo Multicast

In this section, we present a genuine fifo multicast algorithm that tolerates an arbitrary number of failures. This protocol is latency-optimal, as Section 6 shows.

In Algorithm $\mathcal{A}_{\mathit{fifo}}$, every message m is tagged with a sequence number, denoted as $m.seq$. Messages multicast by some process q are F-Delivered in the sequence number order. To do so, every process p keeps track of the next message F-MCast by q to be F-Delivered by p. This information is stored in a variable denoted as $nextFDel[q]_p$. So far, this is like the fifo broadcast algorithm in [8]. We now explain how $\mathcal{A}_{\mathit{fifo}}$ differs from [8]. First, since messages may be addressed to a subset of the system's groups, messages do not carry a single sequence number, as in [8], but an array of sequence numbers, one for each group (see Algorithm $\mathcal{A}_{\mathit{fifo}}$, lines 5-9).

Algorithm $\mathcal{A}_{\mathit{fifo}}$

Genuine Fifo Multicast - Code of process p

```
 1: Initialization
 2:    nbCast[g] ← 0, for each group g
 3:    nextFDel[q] ← 1, for each process q
 4:    msgSet ← ∅
 5: To F-MCast message m                                        { Task 1 }
 6:    foreach g ∈ m.dst do
 7:       nbCast[g] ← nbCast[g] + 1
 8:    m.seq ← nbCast
 9:    send(m) to m.dst

10: When receive(m) or receive(m, OK)
11:    if m ∉ msgSet then
12:       if m.seq[group(p)] = nextFDel[m.sender] then
13:          send(m, OK) to m.dst
14:       else
15:          send(m) to m.dst
16:       msgSet ← msgSet ∪ {m}

17: When ∃m ∈ msgSet:
       ∀g ∈ m.dst : received (m, OK) from all processes in Θ_g
          ∧ m.seq[group(p)] = nextFDel[m.sender]
18:    F-Deliver(m)
19:    nextFDel[m.sender] ← nextFDel[m.sender] + 1
20:    if ∃m′ ∈ msgSet :
          m′.seq[group(p)] = nextFDel[m′.sender] then
21:       send(m′, OK) to m′.dst
```

Second, recall the aforementioned problematic scenario specific to multicast: some process p F-MCasts a message m_1 to some group g_2; later, p F-MCasts a message m_2 to groups g_1 and g_2 and crashes. Message m_2 is received by processes in g_1, and since m_2 is the first message multicast from p to g_1, m_2 is delivered by processes in g_1. On the contrary, all messages sent from p to members of g_2 are lost. Note that this can happen because p crashes and links are quasi-reliable. From the uniform agreement property of fifo multicast, correct processes in g_2 must eventually deliver m_2. However, if they deliver m_2, the fifo order property of fifo multicast will be violated: members of g_2 cannot deliver m_1 before m_2 since m_1 was lost.

To solve this problem, before F-Delivering a message m, a process $p \in m.dst$ announces the addressees of m that it F-Delivered all messages $m.sender$ F-MCast before m by sending them an *OK* message (lines 13 or 21). Message m is then F-Delivered by p when p received an *OK* message from at least one correct process of every correct destination group of m. This is implemented by relying on failure detector Θ.

To ensure that p received an *OK* message from at least one correct process of every correct destination group g of m, for every such group g, p waits to receive an *OK* message from all processes trusted by Θ_g, i.e., the failure detector Θ whose output is restricted to members of g (line 17).

This mechanism is also used to ensure uniform agreement: if there exists a correct addressee of m, when p received an *OK* message from all processes trusted by Θ_g, and this for every group g in $m.dst$, process p knows m was received by at least one correct addressee of m. Hence, all correct processes in $m.dst$ will eventually receive m.

5 Causal Multicast

We now present the first open-group causal multicast algorithm that tolerates quasi-reliable communication links. This algorithm tolerates an arbitrary number of failures and is latency-optimal (c.f. Section 6).

The causal multicast algorithm $\mathcal{A}_{causal}$ relies on fifo multicast and is blocking, that is, to ensure causal order, processes may delay the delivery of a message m for a later time even though all the protocol messages to deliver m have been received.

In Algorithm $\mathcal{A}_{causal}$, every process p keeps track of how many messages, multicast by some process q, it has C-Delivered. This bookkeeping is done for every process q of the system. At p, this information is stored in a vector denoted as $nbDel_p$, indexed by process id. This is like in the causal broadcast algorithm in [8]. In this algorithm, to broadcast a message m, p F-BCasts m along with $nbDel_p$. Upon F-Delivering m, p inserts m in a list of messages $msgLst_p$ and C-Delivers m as soon as it is the first message in $msgLst_p$ such that $nbDel_p \geq m.nbDel$.[2] It is not hard to see why this algorithm works: since $m.nbDel[q]$ denotes the number of messages originating from q that causally precede the multicast of m, C-Delivering m when it is the first message in $msgLst_p$ such that $nbDel_p \geq m.nbDel$, ensures that causal order will not be violated.

In the multicast case, however, this algorithm does not work. Consider the following causal relation between two messages m and m', C-MCast$(m) \rightarrow$ C-MCast(m'), both addressed to some group g, that we denote as *blind* for g. Figure 2 illustrates the scenario. Messages m and m' are such that the causal chain linking the events C-MCast(m) and C-MCast(m') does not contain any events of type C-Deliver of some message addressed to g, and let m' be C-MCast by a process different from $m.sender$. Intuitively, this causal relation is problematic because processes in g may C-Deliver m and m' in different orders.

[2]Given any two vectors v_1 and v_2, we write $v_1 \geq v_2$ instead of $\forall q \in \Pi : v_1[q] \geq v_2[q]$ for simplicity.

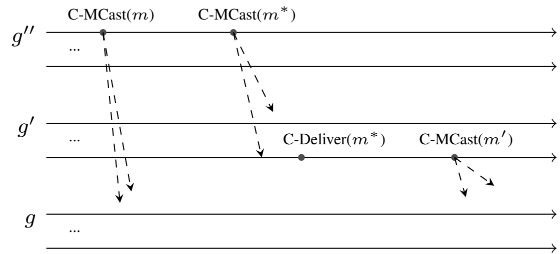

Figure 2: A causal relation between m and m' that is blind for g.

Indeed, since the causal chain linking the events C-MCast(m) and C-MCast(m') does not contain any events of type C-Deliver of some message addressed to g, it is impossible to distinguish m from m' by only comparing the number of messages addressed to g that were C-Delivered in the causal history of events C-MCast(m) and C-MCast(m').[3]

To be able to distinguish messages m and m' in the example above, processes keep track of the number of messages C-MCast in the causal history instead of the number of C-Delivered messages. This accounting is done on a group basis.

Algorithm $\mathcal{A}_{causal}$

Genuine Causal Multicast - Code of process p

Initialization
  $nbCast[g][q] \leftarrow 0$, for each group g and process q
  $nbDel[q] \leftarrow 0$, for each process q
  $msgLst \leftarrow \epsilon$

To C-MCast message m { Task 1 }
  foreach $g \in m.dst$ **do**
    $nbCast[g][p] \leftarrow nbCast[g][p] + 1$
  $m.nbCast \leftarrow nbCast$
  F-MCast (m) to $m.dst$

Function IsDeliverable(m)
  return $\forall q \in \Pi \setminus \{m.sender\}$:
    $m.nbCast[group(p)][q] \leq nbDel[q]$

When F-Deliver(m)
  $msgLst \leftarrow msgLst \oplus m$ ▷ add m at the tail of $msgLst$
  while $\exists m' \in msgLst$: IsDeliverable(m')
    Let m' be the first message in $msgLst$
      s.t. IsDeliverable(m')
    C-Deliver(m')
    $nbDel[m'.sender] \leftarrow m'.nbCast[group(p)][m'.sender]$
    foreach $g \in \Gamma$ **do**
      max applied per vector entry
      $nbCast[g] \leftarrow \max(m'.nbCast[g], nbCast[g])$
    $msgLst \leftarrow msgLst \ominus m'$

Hence, in addition to maintaining vector $nbDel$, each process p keeps track of the number of messages addressed to any group g, originating from any process q, that were C-MCast in its causal history, denoted as $nbCast[g][q]_p$. This variable is piggybacked on every C-MCast message m. Message m is then C-Delivered at p as soon as it is the first message in

[3] An event e is in the causal history of an event e' iff $e \rightarrow e'$.

$msgLst_p$ such that for all processes q different from $m.sender$, $m.nbCast[group(p)][q] \leq nbDel[q]$, i.e., p C-Delivered all messages addressed to $group(p)$ that were C-MCast in the causal history of event C-MCast(m). The delivery condition does not involve $m.sender$ since fifo multicast ensures that messages multicast by the same process will be delivered in the order they were multicast.

We now present the causal multicast algorithm $\mathcal{A}_{causal}$ in detail. To C-MCast a message m, for any group $g \in m.dst$, p increments $nbCast[g][p]_p$ and F-MCasts m along with the $nbCast$ variable (lines 6-9). As soon as some process q F-Delivers this message, q adds m at the end of $msgLst$ (line 13) and checks whether a message can be C-Delivered (line 14). If it is the case, the first C-Deliverable message of $msgLst_p$, m', is C-Delivered. Before removing m' from $msgLst$, $nbDel[m'.sender]$ is updated and for all group g and processes q of the system, $nbCast[g][q]$ is set to the maximum between $m'.nbCast[g][q]$ and $nbCast[g][q]$ so that $nbCast[g][q]$ represents the number of messages originating from q and addressed to g that were C-MCast in the causal history of C-MCast(m') (line 19).

6 Latency Optimality

We show that for any message m there exists no uniform reliable multicast algorithm $\mathcal{A}$ that tolerates quasi-reliable links and delivers m in one message delay, whatever the destination groups of m are.[4] This result is independent of the genuineness of $\mathcal{A}$ and shows the optimality of our uniform fifo and causal multicast algorithms. Indeed, if these algorithms were not optimal, we could get a more efficient uniform reliable multicast algorithm by reducing it to causal or fifo multicast, a contradiction. Moreover, this result also applies to uniform reliable broadcast. To see why, suppose there would exist a uniform reliable broadcast algorithm $\mathcal{A}_{urb}$ that could deliver messages in one message delay. We could then devise a non-genuine uniform reliable multicast algorithm that could deliver messages in one message delay by relying on $\mathcal{A}_{urb}$, a contradiction.

We show this result in the synchronous round-based model which we briefly recall now (see Chapter 2 in [11] for a formal description). Processes may fail by crashing and up to f of them may be faulty. Furthermore, each process p has a buffer, $buffer_p$, that represents the set of messages that have been sent to p but not yet received; p receives the message when it removes it from its buffer. In any run of an algorithm, until it crashes, each process p repeatedly performs the following two steps, which define one round:
-In the first step, p generates the (possibly $null$) messages to be sent to each process based on its current state, and puts these messages in the appropriate process buffers. If p crashes in round r, only a subset of the messages created in r by p are put in the buffers.
-In the second step, p determines its new state based on its current state and on the messages received, and removes all messages from its buffer.

Proposition 6.1 *In any system with $n \geq 3$, $f \geq 2$, and quasi-reliable links, for any uniform reliable multicast algorithm $\mathcal{A}$ and any message m addressed to at least two processes, there does not exist a run R of $\mathcal{A}$ in which m is R-MCast in some round r and R-Delivered by some process q at the end of r.*

Proof: Assume to the contrary that such an algorithm $\mathcal{A}$ and run R of $\mathcal{A}$ exist. In some round r of run R, some process p R-MCasts m and q R-Delivers m at the end of round

[4] Uniform reliable multicast satisfies all the properties of uniform fifo multicast except uniform fifo order.

r. We build a run R' that is indistinguishable from R to q up to and including round r. In R', p crashes in r and m is only received by q. Moreover, q crashes just after R-Delivering m. Hence, in run R', no correct process in $m.dst$ R-Delivers m, violating the uniform agreement property of $\mathcal{A}$. $\square$

7 Conclusion

In this paper, we proposed fifo and causal multicast algorithms that are open-group. These protocols tolerate an arbitrary number of process failures, tolerate quasi-reliable networks, and we showed that they are latency-optimal.

As future work, we intend to study the minimum message complexity of these two problems and characterize how the amount of information about process failures affects this complexity.

References

[1] M. K. Aguilera, S. Toueg, and B. Deianov. Revising the weakest failure detector for uniform reliable broadcast. In *Proceedings of DISC'99*, pages 19–33. Springer-Verlag, 1999.

[2] S. Alagar and S. Venkatesan. An optimal algorithm for distributed snapshots with causal message ordering. *Information Processing Letters*, 50(6):311–316, 1994.

[3] R. Baldoni, R. Friedman, and R. van Renesse. The hierarchical daisy architecture for causal delivery. *Distributed Computing Systems, International Conference on*, 0:570–577, 1997.

[4] K. P. Birman and T. A. Joseph. Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5(1):47–76, 1987.

[5] K. P. Birman, A. Schiper, and P. Stephenson. Lightweight causal and atomic group multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, August 1991.

[6] X. Défago, A. Schiper, and P. Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys*, 36(4):372–421, 2004.

[7] R. Guerraoui and A. Schiper. Genuine atomic multicast in asynchronous distributed systems. *Theoretical Computer Science*, 254(1-2):297–316, 2001.

[8] V. Hadzilacos and S. Toueg. Fault-tolerant broadcasts and related problems. In Sape J. Mullender, editor, *Distributed Systems*, chapter 5, pages 97–145. Addison-Wesley, 1993.

[9] A. D. Kshemkalyani and M. Singhal. Necessary and sufficient conditions on information for causal message ordering and their optimal implementation. *Distributed Computing*, 11(2):91–111, 1998.

[10] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[11] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[12] F. Mattern and S. Fünfrocken. A non-blocking lightweight implementation of causal order message delivery. In *Selected Papers from the International Workshop on Theory and Practice in Distributed Systems*, pages 197–213, London, UK, 1995. Springer-Verlag.

[13] A. Mostefaoui and M. Raynal. Causal multicasts in overlapping groups: Towards a low cost approach. In *Proceedings of the 4th IEEE International Conference on Future Trends in Distributed Computing Systems*, pages 136–142, 1993.

[14] M. Raynal, A. Schiper, and S. Toueg. The causal ordering abstraction and a simple way to implement it. *Information Processing Letters*, 39:343–350, 1991.

[15] L. Rodrigues and P. Verissimo. Causal separators for large-scale multicast communication. In *Proceedings of ICDCS '95*, page 83, Washington, DC, USA, 1995. IEEE Computer Society.

[16] N. Schiper and F. Pedone. Solving atomic multicast when groups crash. In *Proceedings of OPODIS'08*, pages 481–495, 2008.

[17] N. Schiper and F. Pedone. Fast, scalable, flexible, and highly resilient fifo and causal multicast algorithms. Technical Report 2009/003, University of Lugano, 2009.

A Proofs of Correctness

In the proofs below, we denote the value of a variable V on a process p at time t as V_p^t. Furthermore, for events of the type C-MCast and C-Deliver, we sometimes add a subscript to denote on which process this event occurred.

A.1 The Proof of Algorithm $\mathcal{A}_{fifo}$

Proposition A.1 (Uniform Integrity) *For any process p and any message m, (a) p F-Delivers m at most once, and (b) only if $p \in m.dst$ and (c) m was previously F-MCast.*

Proof:

- (a) After p F-Delivers m, p increments $nextFDel[m.sender]$. Thus, the condition of line 17 can never evaluate to true for m anymore.
- (b) Follows directly from the uniform integrity property of links and the algorithm.
- (c) Process p F-Delivers m only if p received m. From the uniform integrity property of links, m was sent by some process. Consequently, m was F-MCast. □

Proposition A.2 Uniform Fifo Order *If a process p F-MCasts a message m before F-MCasting a message m', then no process in $m.dst \cap m'.dst$ F-Delivers m' unless it has previously F-Delivered m.*

Proof: Let q be any process in $m.dst \cap m'.dst$ that F-Delivers m', we show that q F-Delivers m before. If q F-Delivers m', then there is a time t before q F-Delivers m' at which $nextFDel[m.sender]_q^t = m'.seq[group(q)]$. From the definition of m, $m.seq[group(q)] < m'.seq[group(q)]$. From lines 17-19, q must have F-Delivered m before t, and thus before q F-Delivers m'. □

Definition A.1 *We define the binary relation $pred$ on messages as follows, $m_1 \; pred \; m_2$ iff:*

1. *$m_1.sender = m_2.sender$,*
2. *$m_1.sender$ F-MCasts m_1 before m_2, and*
3. *There exists at least one correct process in $m_1.dst \cap m_2.dst$*

Moreover, let $\mathcal{G}_{pred(m)} = (V, E)$ be a finite DAG constructed as follows:

1. *add vertex m to V*
2. *while $\exists m_1 \in V : \exists m_2 \notin V : m_2 \; pred \; m_1$ do:*
   add m_2 to V and add directed edge $m_2 \rightarrow m_1$ to E

For any message m' in $\mathcal{G}_{pred(m)}$, we say that m' is at distance k of m iff the longest path from m' to m is of length k. We let $\mathcal{M}_k$ be the subset of messages in $\mathcal{G}_{pred(m)}$ that are at distance k of m.

Lemma A.1 *For any message m, if for all messages m' in $\mathcal{G}_{pred(m)}$ all correct processes in $m'.dst$ receive m', then all correct processes $p \in m.dst$ eventually F-Deliver m.*

Proof: Assume that for all messages m' in $\mathcal{G}_{pred(m)}$ all correct processes in $m'.dst$ receive m'. We prove that, for any $k \geq 0$, all messages in $\mathcal{M}_k$ are eventually F-Delivered by all their correct addressees. Since $\mathcal{M}_0 = \{m\}$, this shows the claim. Let x be the largest integer such that $\mathcal{M}_x \neq \emptyset$. We proceed by induction on k, starting from $k = x$.

- Base step ($k = x$): Let m_x be any message in $\mathcal{M}_x$ and q be any correct process in $m_x.dst$. From the definition of x, (*) there exists no message m_{x+1} such that $m_{x+1}\, pred\, m_x$. Since for all messages m' in $\mathcal{G}_{pred(m)}$, all correct processes in $m'.dst$ eventually receive m', q eventually receives m_x. By (*), m_x is the first message F-MCast by $m.sender$ such that $q \in m_x.dst$, and hence, $m_x.seq[group(q)] = 1$. Therefore, all correct processes in $m_x.dst$ eventually send(m_x, OK) and by the reliability property of links, q eventually receives these messages. By the completeness property of Θ, there exists a time after which q does not trust any process that crashes. Hence, by the condition of line 17, q eventually F-Delivers m_x.

- Induction step: Suppose the claim holds for k ($0 < k \leq x$), we show it holds for $k-1$. Let m_{k-1} be any message in $\mathcal{M}_{k-1}$, g be any correct group in $m_{k-1}.dst$, and q be any correct process in g. We first show that (*) there exists a time t at which $nextFDel[m.sender]_q^t = m_{k-1}.seq[group(q)]$. Either (a) m_{k-1} is the first message F-MCast to g or (b) not.

  - In case (a), $m_{k-1}.seq[g] = 1$. Since $nextFDel[m.sender]$ is initialized to 1, (*) holds.
  - In case (b), there exists a message $m_{k'}$ in $\mathcal{M}_{k'}$ ($x \geq k' > k-1$) such that $m_{k'}\, pred\, m_{k-1}$, $g \in m_{k'}.dst \cap m_{k-1}.dst$, and $m_{k'}.seq[g] = m_{k-1}.seq[g] - 1$. By the induction hypothesis, q F-Delivers $m_{k'}$. Therefore, (*) holds.

  From the algorithm, since for all messages m' in $\mathcal{G}_{pred(m)}$, all correct processes in $m'.dst$ eventually receive m', all correct processes r in $m_{k-1}.dst$ eventually receive m_{k-1}. Consequently, from (*), all r send (m_{k-1}, OK), either at line 13 or at line 21. By the quasi-reliability property of links, q eventually receive these messages. By the completeness property of Θ, there exists a time after which q does not trust any process that crashes. Consequently, by the condition of line 17, q eventually F-Delivers m_{k-1}. □

Lemma A.2 *For any message m and any process p, if p sends (m, OK), then p F-Delivered all messages m' such that $p \in m'.dst$ and $m.sender$ F-MCast m' before m.*

Proof: If m is the first message $m.sender$ F-MCasts to $group(p)$, the claim holds trivially. Otherwise, let m_x be the message such that $p \in m_x.dst$ and $m.sender$ F-MCasts m_x just before m. Since p sends (m, OK), there exists a time t at which $nextFDel[m.sender]_p^t = m.seq[group(p)]$. From lines 17-19, p must have F-Delivered m_x before t. By applying Proposition A.2 multiple times, before t, p also F-Delivered all messages addressed to $group(p)$ that $m.sender$ F-MCast before m_x. □

Proposition A.3 (Uniform Agreement) *If a process p F-Delivers a message m, then all correct processes $q \in m.dst$ eventually F-Deliver m.*

Proof: Let $\mathcal{M}_k$ be the subset of messages in $\mathcal{G}_{pred(m)}$ that are at distance k of m. We first show that, for any $k \geq 0$ and any message m' in $\mathcal{M}_k$ such that $\mathcal{M}_k \neq \emptyset$: (1) m' is received by all correct processes in $m'.dst$ and (2) for each correct group $g \in m'.dst$, there is a correct process q in g that sends (m', OK). We proceed by simultaneous induction on (1) and (2).

- Base step ($k = 0$):
  - (1) Since p F-Delivers m, from the condition of line 17, p received an (m, OK) message from all processes trusted by Θ_g, and this for every group $g \in m.dst$. If there are no correct processes in $m.dst$, then the base step of (1) holds trivially. Otherwise, by the accuracy property of Θ, p received an (m, OK) message from a correct addressee q of m. Since q is correct, by the quasi-reliability property of links, every correct process in $m.dst$ eventually receives m from q.
  - (2) Since p F-Delivers m, from the condition of line 17, for every group g in $m.dst$, p received a message (m, OK) from all processes trusted by Θ_g. By the accuracy property of Θ, for all correct group $g \in m.dst$, p received message (m, OK) from a correct process q in g. Hence, by the uniform integrity property of links, q sent (m, OK).

- Induction step: Suppose that (1) and (2) hold for $k - 1$ ($k > 0$), we show that (1) and (2) also hold for k. Let m_{k-1} be any message in $\mathcal{M}_{k-1}$ and let m_k be any message in $\mathcal{M}_k$ such that $m_k \; pred \; m_{k-1}$.
  - (1) Because $k > 0$, from the definition of m_k and the definition of the $pred$ relation, $m_k.sender$ F-MCasts m_k before m_{k-1} and there exists a correct process in $m_k.dst \cap m_{k-1}.dst$. By the induction hypothesis, for each group g in m_{k-1} containing at least one correct process, there exists a correct process q in g that sends (m_{k-1}, OK). Hence, there exists a correct process q in $m_{k-1}.dst \cap m_k.dst$ such that q sends (m_{k-1}, OK). By Lemma A.2, q F-Delivered m_k. If there are no correct processes in $m_k.dst$, then the induction step of (1) holds trivially. Otherwise, from the condition of line 17, q received an (m_k, OK) message from all processes trusted by Θ_g, and this for every group $g \in m_k.dst$. Hence, from the accuracy property of Θ, q received (m_k, OK) from a correct process $r \in m_k.dst$. Therefore, by the quasi-reliability property of links, every correct process in $m_k.dst$ eventually receives m_k from r.
  - (2) From the definition of m_k and the definition of the $pred$ relation, $m_k.sender$ F-MCasts m_k before m_{k-1} and there exists a correct process in $m_k.dst \cap m_{k-1}.dst$. By the induction hypothesis, there exists a correct process $r \in m_k.dst \cap m_{k-1}.dst$ such that r sends (m_{k-1}, OK). By Lemma A.2, r F-Delivered m_k. From the condition of line 17, r received an (m_k, OK) message from all processes trusted by Θ_g, and this for every group $g \in m_k.dst$. Hence, by the accuracy property of Θ, for all correct group $g \in m_k.dst$, r received (m_k, OK) from a correct process q in g. Therefore, By the uniform integrity property of links, q sent (m_k, OK).

Hence, from (1), all messages $m' \in \mathcal{G}_{pred(m)}$ are received by all correct processes in $m'.dst$. Therefore, by Lemma A.1, all correct processes in $m.dst$ F-Deliver m. □

Proposition A.4 (Validity) *If a correct process p F-MCasts a message m, then eventually all correct processes $q \in m.dst$ F-Deliver m.*

Proof: Since p is correct, by the quasi-reliability property of links, for all messages $m' \in \mathcal{G}_{pred(m)}$, all correct processes in $m'.dst$ receive m. By Lemma A.1, all correct processes $q \in m.dst$ eventually F-Deliver m. □

A.2 The Proof of Algorithm $\mathcal{A}_{causal}$

Proposition A.5 (Uniform Integrity) *For any process p and any message m, (a) p C-Delivers m at most once, and (b) only if $p \in m.dst$ and (c) m was previously C-MCast.*

Proof:

- (a) Follows directly from the uniform integrity property of fifo multicast and from the fact that a message is removed from $msgLst_p$ after it is C-Delivered.
- (b) Follows directly from the algorithm.
- (c) Process p C-Delivers m only if p F-Delivered m. From the uniform integrity property of fifo multicast, m was F-MCast. Consequently, m was C-MCast. □

Lemma A.3 *For any message m such that $m.nbDel$ is defined, any group g, and any integer k,$m.nbCast[g][m.sender] = k$ iff m is the k-th message $m.sender$ C-MCasts to g.*

Proof:

- (⇒): From the algorithm, $m.sender$ increments $nbCast[g][m.sender]_{m.sender}$ at line 7 only ($m.sender$ does not update $nbCast[g][m.sender]_{m.sender}$ at line 19). Moreover, $m.sender$ does so before every message C-MCast to g. Therefore, since $nbCast[g][m.sender]$ is initialized to 0, m is the k-th message $m.sender$ C-MCasts to g.
- (⇐): The same argument as in (⇒) is used to show that $m.nbCast[g][m.sender] = k$. □

Lemma A.4 *For any two messages m and m' such that $m.nbCast$ and $m'.nbCast$ are defined, and any group g, if C-MCast(m) → C-MCast(m'), then $m.nbCast[g] \leq m'.nbCast[g]$.*

Proof: From the definition of the causal precedence relation, it is easy to see that there exist processes $p_1, p_2, .., p_k$ and messages $m_1, m_2, .., m_k = m'$ ($k \geq 2$) such that:

- $p_1 = m.sender$
- p_i C-MCasts m_i for all $1 \leq i \leq k$
- either (a) $m = m_1$ or (b) p_1 C-MCasts m before m_1 and

- p_i C-Delivers m_{i-1} before it C-MCasts m_i, for all $2 \leq i \leq k$

- In case (a), we show that for any $1 \leq i < k$, $m_i.nbCast[g] \leq m_{i+1}.nbCast[g]$. Process p_{i+1} C-Delivers m_i before C-MCasting m_{i+1}. Thus, from line 19 and because for any process q $nbCast[g][q]_{p_{i+1}}$ is monotonically increasing with time, $m_i.nbCast[g] \leq m_{i+1}.nbCast[g]$.

- In case (b), since for any process q $nbCast[g][q]_{p_1}$ is monotonically increasing with time, $m.nbCast[g][q] \leq m_1.nbCast[g][q]$. To conclude the proof, the same argument as in (a) is used to show that for any $1 \leq i < k$, $m_i.nbCast[g] \leq m_{i+1}.nbCast[g]$.

Lemma A.5 *For any processes p and q, any integer k, and any time t at which p evaluates the condition of line 11, $nbDel[group(p)][q]^t_p = k$ iff before t, p C-Delivered the first k messages q C-MCasts to $group(p)$.*

Proof:

- ($\Rightarrow$) : We proceed by induction on k.
  - Base step ($k = 0$): Since $nbDel[group(p)][q]_p$ is initialized to zero and monotonically increasing with time, if at the time t at which p evaluates line 11, $nbDel[group(p)][q]^t_p = 0$ then p did not C-Deliver any message C-MCast from q before t.
  - Induction step: Suppose that the claim holds for any l such that $0 \leq l \leq k-1$, we show that it also holds for k. Process p sets $nbDel[group(p)][q]_p$ to k just after C-Delivering a message m_k originating from q such that $m.nbCast[group(p)][q] = k$. From Lemma A.3, m_k is the k-th message q C-MCasts to $group(p)$. Let t' be the latest time before t at which p evaluates line 11 such that $nbDel[group(p)][q]^{t'}_p \neq nbDel[group(p)][q]^t_p$. We show that (*) $nbDel[group(p)][q]^{t'}_p = k-1$.

    Suppose, by way of contradiction, that $nbDel[group(p)][q]^{t'}_p \neq k-1$. Let k' be the value of $nbDel[group(p)][q]^{t'}_p$. Either (a) $k' < k-1$ or (b) $k' > k-1$. We show that both (a) and (b) lead to a contradiction.
    * In case (a), from the induction hypothesis, before t' p C-Delivered the first k' messages C-MCast from q. Since $k' < k-1$, either (a-i), p does not C-Deliver the k-1-th message m_{k-1} C-MCast from q or (a-ii) p C-Delivers m_{k-1} after m_k.
      · In case (a-i), since p C-Delivers m_k, from the uniform fifo order property of fifo multicast, p F-Delivers and inserts m_{k-1} before m_k in $msgLst_p$. From Lemma A.4, $m_{k-1}.nbCast[group(p)][q] \leq m_k.nbCast[group(p)][q]$. From the condition of line 11 and since $nbDel[group(p)][q]_p$ is monotonically increasing with time, p must have C-Delivered m_{k-1} before m_k, a contradiction to the fact that p does not C-Deliver m_{k-1}.
      · In case (a-ii), the same argument as in (a-i) is used to obtain a contradiction.

* In case (b), since $k' \neq k$ and $k' > k-1$, $k' > k$. Process p sets $nbDel[group(p)][q]_p$ to k' just after C-Delivering a message $m_{k'}$ originating from q such that $m.nbCast[group(p)][q] = k'$. From Lemma A.3, $m_{k'}$ is the k'-th message q C-MCasts to $group(p)$. Since $t' < t$, (**) p C-Delivers $m_{k'}$ before m_k. Since $k < k'$, from the uniform fifo order property of fifo multicast, p F-Delivers and inserts m_k before $m_{k'}$ in $msgLst_p$. From Lemma A.4, $m_k.nbCast[group(p)][q] \leq m_{k'}.nbCast[group(p)][q]$. From the condition of line 11 and since $nbDel[group(p)][q]_p$ is monotonically increasing with time, p C-Delivers m_k before $m_{k'}$, a contradiction to (**).

  From (*), $nbDel[group(p)][q]_p^{t'} = k-1$. From the induction hypothesis, before t' p C-Delivered the first $k-1$ messages C-MCast from q. Therefore, before t, p C-Delivered the first k messages C-MCast from q.

- ($\Leftarrow$): Either (a) $k = 0$ or (b) $k > 0$.

  - In case (a), if p did not C-Deliver any message C-MCast from q, since $nbDel[group(p)][q]_p$ is initialized to zero, then $nbDel[group(p)][q]_p^t = 0$.
  - In case (b), let m_k be the k-th message C-MCast from q that p C-Delivers. We show that (*) the last message C-MCast from q that p C-Delivers before t is m_k. Suppose, by way of contradiction, that the last message $m_{k'}$ C-MCast from q that p C-Delivers before t is not m_k. If before t, p C-Delivers the first k messages C-MCast from q, then m'_k is the k'-th message C-MCast from q to $group(p)$ such that $k' < k$. From the uniform fifo order property of fifo multicast, p F-Delivers and inserts $m_{k'}$ before m_k in $msgLst_p$. From Lemma A.4, $m_{k'}.nbCast[group(p)][q] \leq m_k.nbCast[group(p)][q]$. Since p C-Delivers m_k, from the condition of line 11 and since $nbDel[group(p)][q]_p$ is monotonically increasing with time, p C-Delivers $m_{k'}$ before m_k. Since p C-Delivers m_k before t, $m_{k'}$ is not the last message C-MCast from q that p C-Delivers before t, a contradiction.

    From Lemma A.3, m_k is such that $m_k.nbCast[group(p)][q] = k$. Therefore, from (*) and line 17, $nbDel[group(p)][q]_p^t = k$. □

Proposition A.6 Uniform Causal Order *For any messages m and m', if C-MCast(m) $\rightarrow$ C-MCast(m'), then no process $p \in m.dst \cap m'.dst$ C-Delivers m' unless it has previously C-Delivered m.*

Proof: Let q be any process in $m.dst \cap m'.dst$ that C-Delivers m', we show that q C-Delivered m before. Either (a) $m.sender = m'.sender$ or (b) not.

- In case (a), since q C-Delivers m', q F-Delivers m'. From the uniform fifo order property of fifo multicast, (*) q F-Delivers m before F-Delivering m'. Either (a-i) there exists a time at which m and m' are in $msgLst_q$ or (a-ii) not.

  - In case (a-i), from (*) m can only appear before m' in $msgLst_q$. From Lemma A.4, $m.nbCast[group(q)] \leq m'.nbCast[group(q)]$. Since at line 15, processes C-Deliver the first message in $msgLst$ such that the condition of line 11 is satisfied, q C-Delivers m before m'.

- In case (a-ii), from (*) m is inserted in $msgLst_q$ before m'. Since processes remove messages from $msgLst_q$ only after C-Delivering them (line 20), q C-Delivers m before m'.

- In case (b), from Lemma A.4, $m.nbCast[group(q)] \leq m'.nbCast[group(q)]$. From the condition of line 11, there exists a time t before q C-Delivers m' at which q evaluates line 11 such that for any process $r \neq m'.sender$ $nbDel[r]_q^t \geq m'.nbCast[group(q)][r]$. Hence, since $m.sender \neq m'.sender$, $nbDel[m.sender]_q^t \geq m.nbCast[group(q)][m.sender]$. Let k_1 and k_2 be $nbDel[m.sender]_q^t$ and $m.nbCast[group(q)][m.sender]$ respectively. From Lemma A.3, m' is the k_2-th message $m.sender$ C-MCasts to $group(q)$. From Lemma A.5, before t, q C-Delivers the first k_1 messages $m.sender$ C-MCasts to $group(q)$. Therefore, since $k_1 \geq k_2$, q C-Delivers m before m'. □

Definition A.2 *Let m be a message, we define the finite DAG $\mathcal{G}_{pred(m)} = (V, E)$ as follows:*

1. *add vertex m to V*
2. *while $\exists m_1, m_2$ s.t. $m_1 \in V \wedge$ C-MCast(m_2) $\rightarrow$ C-MCast(m_1) $\wedge\ m_1.dst \cap m_2.dst \neq \emptyset$ do:*
   add m_2 to V and add directed edge $m_2 \rightarrow m_1$ to E

For any message m' in $\mathcal{G}_{pred(m)}$, we say that m' is at distance k of m iff the longest path from m' to m is of length k. We let $\mathcal{M}_k$ be the subset of messages in $\mathcal{G}_{pred(m)}$ that are at distance k of m.

Lemma A.6 *For any message m, if for all messages $m' \in \mathcal{G}_{pred(m)}$ all correct processes in $m'.dst$ F-Deliver m', then all correct processes in $m.dst$ eventually C-Deliver m.*

Proof: Assume that for all messages m' in $\mathcal{G}_{pred(m)}$ all correct processes in $m'.dst$ F-Deliver m'. We prove that, for any $k \geq 0$, all messages in $\mathcal{M}_k$ are eventually C-Delivered by their correct addressees. Since $\mathcal{M}_0 = \{m\}$, this shows the claim. Let x be the largest integer such that $\mathcal{M}_x \neq \emptyset$. We proceed by induction on k, starting from $k = x$.

- Base step ($k = x$): Let m_x be any message in $\mathcal{M}_x$ and q be any correct process in $m_x.dst$. From the definition of m_x, (*) there exists no message m' such that C-MCast(m') $\rightarrow$ C-MCast(m_x) and $m'.dst \cap m_x.dst \neq \emptyset$. Let g be any group in $m_x.dst$ and let r be any process different from $m_x.sender$. From (*), $m_x.sender$ never updated $nbCast[g][r]_{m_x.sender}$ at line 19. Therefore, $m.nbCast[g][r] = 0$. From the algorithm, $nbDel[g][r]_q$ is monotonically increasing with time and hence $nbDel[g][r]_q \geq m.nbCast[g][r]$ is always true. Since all correct processes in $m_x.dst$ eventually F-Deliver m_x, from the condition of line 11, all correct processes in $m_x.dst$ eventually C-Deliver m_x.

- Induction step: Suppose that for any l such that $x \geq l > k \geq 0$ the claim holds, we show the claim holds for k. Let m_k be any message in $\mathcal{M}_k$ and q be any correct process in $m_k.dst$ (if there exists no correct process in $m_k.dst$, then the claim holds trivially). We show that (*) for any process r different from $m_k.sender$ there exists a time t at which $nbDel[group(q)][r]_q^t \geq m_k.nbCast[group(q)][r]$.

If $m_k.nbCast[group(q)][r] = 0$, then (*) holds trivially. Otherwise, from line 19, there exists a message m_r such that $m_r.sender = r$, $group(q) \in m_r.dst$, C-MCast(m_r) $\rightarrow$ C-MCast(m_k), and $m_r.nbCast[group(q)][r] = m_k.nbCast[group(q)][r]$. From the induction hypothesis, q eventually C-Delivers m_r and sets $nbDel[group(q)][r]_q$ to $m_r.nbCast[group(q)][r]$. Since $m_r.nbCast[group(q)][r] = m_k.nbCast[group(q)][r]$, (*) holds.

Since all correct processes in $m_k.dst$ eventually F-Deliver m_k, from (*) and since $nbDel[group(q)][r]_q$ is monotonically increasing with time, from the condition of line 11, all correct processes in $m_k.dst$ eventually C-Deliver m_k. □

Proposition A.7 (Uniform Agreement) *If a process p C-Delivers a message m, then all correct processes $q \in m.dst$ eventually C-Deliver m.*

Proof: Let $\mathcal{M}_k$ be the subset of messages in $\mathcal{G}_{pred(m)}$ that are at distance k of m. We first show that (*) for any k, all messages in $\mathcal{M}_k$ are eventually F-Delivered by all of their correct addressees.

- Base step ($k = 0$): Since p C-Delivers m, p F-Delivers m. From the uniform agreement property of fifo multicast, all correct processes in $m.dst$ eventually F-Deliver m.
- Induction step: Suppose the claim holds for any l such that $k \geq l \geq 0$, we show that the claims holds for $k + 1$. Let m_{k+1} be any message in $\mathcal{M}_{k+1}$ and g be any correct group in $m_{k+1}.dst$. Furthermore, let k' be the largest integer smaller than $k + 1$ such that there exists a message $m_{k'}$ in $\mathcal{M}_{k'}$, $g \in m_{k'}.dst \cap m_{k+1}.dst$, and C-MCast($m_{k+1}$) $\rightarrow$ C-MCast($m_{k'}$). Either (a) $m_{k+1}.sender = m_{k'}.sender$ or (b) not.
  - In case (a), by the induction hypothesis, all correct processes in g eventually F-Deliver $m_{k'}$. Therefore, from the uniform fifo order property of fifo multicast, all correct processes in g F-Deliver m_{k+1} before $m_{k'}$.
  - In case (b), since C-MCast(m_{k+1}) $\rightarrow$ C-MCast($m_{k'}$) and $m_{k+1}.sender \neq m_{k'}.sender$, from the definition of $m_{k'}$, $m_{k'}.sender$ C-Delivers m_{k+1} before C-MCasting $m_{k'}$. Hence, from the algorithm, $m_{k'}.sender$ F-Delivers m_{k+1}. Therefore, from the uniform agreement property of fifo multicast, all correct processes in g eventually F-Deliver m_{k+1}.

By (*) and Lemma A.6, all correct processes $q \in m.dst$ eventually C-Deliver m. □

Proposition A.8 (Validity) *If a correct process p C-MCasts a message m, then eventually all correct processes $q \in m.dst$ C-Deliver m.*

Proof: Let $\mathcal{M}_k$ be the subset of messages in $\mathcal{G}_{pred(m)}$ that are at distance k of m. We show that (*) for any k, all messages in $\mathcal{M}_k$ are eventually F-Delivered by all of their correct addressees.

- Base step ($k = 0$): From the algorithm, p F-MCasts m. Since p is correct, from the validity property of fifo multicast, all correct processes in $m.dst$ eventually F-Deliver m.

- Induction step: Suppose the claim holds for any l such that $k \geq l \geq 0$, we show that the claims holds for $k + 1$. The same argument as in the induction step of Proposition A.7 is used.

By (*) and Lemma A.6, all correct processes $q \in m.dst$ eventually C-Deliver m. $\square$