APPENDIX: PROOF OF CORRECTNESS

Proposition 1. *(Uniform integrity) For any message m, every process p delivers m at most once, and only if p is a destination of m and m was previously multicast.*

Proof: Process p delivers m at Task 6 if m's state is ORDERED. After delivering m, p sets m's state to DONE, and thus m cannot be delivered more than once.

Let c be the client that multicasts m to groups in dst, and let p be in group g. From Task 6, p only delivers m if it is in p's M buffer and m's state is ORDERED. Message m's state is set to ORDERED in Task 5 if its current state is MCAST. A message's state is set to MCAST in procedure *Relay*, which is invoked in two cases: (a) by client c upon multicasting m (Task 1) to groups in dst, in which case $g \in dst$; or (b) by some process q that suspects c (Task 10), has m in its buffer in state MCAST, and g is a destination of m. In case (b), m was written in q's buffer either (b.1) directly by c or (b.2) indirectly by some other process. In any case, there is some process r such that m is included in r's buffer by c. It follows from Task 1 that p is a destination of m and m was multicast by client c. □

Lemma 1. *If all correct processes in the destination of an atomically multicast message m have m in their M buffer in the MCAST state, then they eventually set m to the ORDERED state.*

Proof: Let m be addressed to groups in dst and q be a correct process addressed by m. We claim that for each $h \in dst$, q will have a timestamp for h that is acknowledged by a quorum of processes in h. By the leader election oracle and the fact that each group has a majority of correct processes, group h eventually has a stable correct leader l. Either (a) l executes Task 2 and proposes its clock value as h's timestamp or (b) l executes Task 7 to replace a suspected leader. In (b), l sends a CATCH_UP message to all processes and will receive for each group $g \in dst$ the timestamp proposed in g, if any, and the corresponding acknowledgements from processes in g (Task 8). For the case where $h = g$, l will pick the timestamp decided by a previous leader or choose one if no timestamp has been decided (Task 9). Thus, in both cases (a) and (b), the leader writes the chosen timestamp in the M buffer of each process in h and in the leaders of other groups in dst. From Task 3, every follower in h will acknowledge this timestamp in the buffer of each process in the destination of m. From Task 4, when l has a timestamp from $g \neq h$, l writes the timestamp in the buffer of its followers, which concludes the claim. Therefore, eventually q has a timestamp for every group in dst, can compute m's final timestamp, and set m's state as ORDERED.

Lemma 2. *If a correct process p has an atomically multicast message m in its M buffer in the ORDERED state, then p eventually delivers m.*

Proof: Assume for a contradiction that q does not deliver m. Thus, there is some message m' in the buffer such that $m \neq m'$, m''s timestamp is smaller than m's timestamp, and m''s state is not DONE.

We first show that any message added in the buffer after m becomes ORDERED has a timestamp bigger than m's timestamp. Message m only becomes ordered after it has timestamps from all groups in m's destinations dst. When q reads a timestamp x for m from some group in dst, q updates its clock such that it contains the maximum between its current value and x. Since the next event that q handles for a message m'' will increment its clock, it follows that m'' will have a timestamp bigger than x.

We now show that every message that contains a timestamp smaller than m's final timestamp ts is eventually delivered and its state set to DONE. To see why, let m' be the message with the smallest timestamp in the buffer. Thus, such a message is eventually delivered and its state set to ORDERED. Eventually, m will be the message in the buffer with smallest timestamp and therefore delivered, a contradiction. We conclude then that q eventually delivers m. □

Proposition 2. *(Validity) If a correct client c multicasts a message m, then eventually every correct process p in m's destination dst delivers m.*

Proof: Upon multicasting m, c relays m to groups in dst (see Task 1). The Relay procedure then copies m to the M buffer of every correct process p in groups in dst and sets its state to MCAST. From Lemma 1, it follows that every correct process p set m's state to ORDERED. From Lemma 2, p eventually delivers m. □

Proposition 3. *(Uniform agreement) If a process p delivers a message m, then eventually all correct processes q in m's destination dst deliver m.*

Proof: For process p to deliver m, from Task 6, p has a timestamp for every group h in dst in the M buffer such that ts is the largest among these timestamps. Moreover, there is no message m' in the buffer such that $m \neq m'$, $ts < y$, where y is a timestamp assigned to m', and m' is not ordered.

We first show by contradiction that q eventually has m in its M buffer. Let c be the client that multicasts m. If c is correct then, c writes m in q's buffer, so consider that c fails before it can write m in q's buffer. Since p delivers m, it has a quorum of acknowledges from each group in dst. Any quorum includes at least one correct process, which from

Task 10, eventually suspects c and relays m to all processes in dst, including q, a contradiction.

It follows from Lemma 1 that q eventually sets the state of m to ORDERED in its buffer, and from Lemma 2 that q eventually delivers m. □

PROPOSITION 4. *(Uniform prefix order) For any two messages m and m' and any two processes p and q such that $\{p, q\} \subseteq dst \cap dst'$, where dst and dst' are the groups addressed by m and m', respectively, if p delivers m and q delivers m', then either p delivers m' before m or q delivers m before m'.*

PROOF: The proposition trivially holds if p and q are in the same group, so assume p is in group g and q is in group h and suppose, by way of contradiction, that p does not deliver m' before m nor does q deliver m before m'. Without loss of generality, suppose that m's timestamp ts is smaller than m''s timestamp ts'.

We claim that q inserts m into the M buffer before delivering m'. In order for m (respectively, m') to be delivered by p (resp., q), p's (resp., q's) M buffer must contain a timestamp ts_g from group g and ts_h from group h (resp., ts'_g from group g and ts'_h from group h).

From Task 2 (or Task 9 if some process has suspected the leader), the leader l in group g must have included the timestamp ts_g for message m and ts'_g for message m' in p's M buffer and both timestamps have been acknowledged by a quorum of processes in group g. Assume that the leader l has written ts_g before ts'_g to the M buffer of every follower in group g and the leader l_h in group h. From Task 2, we have $ts_g < ts'_g$. Therefore, from Task 4, l_h will write to the M buffer of every follower in group h, including q, both ts_g for message m and ts'_g for message m'.

Consequently, from the claim, q delivers m before m' since $m.ts < m'.ts$, a contradiction that concludes the proof. □

PROPOSITION 5. *(Uniform acyclic order) Let relation $<$ be defined such that $m < m'$ iff there exists a process that delivers m before m'. The relation $<$ is acyclic.*

PROOF: Suppose, by way of contradiction, that there exist messages $m_1, ..., m_k$ such that $m_1 < m_2 < ... < m_k < m_1$. From Task 6, processes deliver messages following the order of their final timestamps. Thus, there must be processes p and q such that the final timestamps they assign to m_1, ts_p and ts_q, satisfy $ts_p < ts_q$, a contradiction since both p and q have the same timestamps for each group in dst in Task 6. □

THEOREM 1. *RamCast implements atomic multicast.*

PROOF: This follows directly from Propositions 1 through 5. □