ISIS algorithm for total ordering of messages

Let $g=\{p_1,p_2,p_3,...p_n\}$ be a closed group of processes that multicast all messages to the entire group. The following algorithm, executed at each process p_i in g will ensure that all processes in g will deliver all messages in the same order.

On initialization: $s_i = 0$; $counter_i = 0$;

On multicast of message m to all processes in g
  $counter_i := counter_i + 1$;
  B-multicast($<m,counter_i,i>$);

On B-deliver of message $<m,m_{id},j>$ from p_j (where $1 \le j \le n$)
  $s_i := s_i + 1$;
  Send $<m_{id},s_i>$ to p_j;
  Put $<m,m_{id}, j, s_i, i,$ undeliverable> in hold-back queue;

On receive($<m_{id},s_j>$) from p_j (where $1 \le j \le n$)
  Add $<s_j,j>$ to list of suggested sequence numbers for message m_{id};
  if we have received sequence number from all processes in g then
    $<s_k,k>$:= highest sequence number in list (suggested by p_k);
    // choose smallest possible value for k if there are multiple suggesting this sequence #
    B-multicast($<m_{id}, i, s_k, k>$);
  end-if

On B-deliver of message $<m_{id}, i, s_k, k>$ (where $1 \le k \le n$)
  $s_i := \max(s_i,s_k)$
  Modify message with id $<m_{id}, i>$ on hold-back queue as follows:
    change proposed sequence number to s_k;
    change process that suggested sequence number to k;
    change undeliverable to deliverable;

On addition to queue or changing of element in queue
  // note that all elements have the following format: $<m,m_{id},j,s,k,$status>, where
  // m is the message, m_{id} is the message id of m, j is the process that sent m,
  // s is the suggested sequence number, k is the process that suggested s (or k is the
  // suggesting process), and status is either deliverable or undeliverable.
  Sort such that message with smallest sequence number is at the head
  If two sequence numbers are the same then
    place any undeliverable messages at the head
    to break further ties, place message with smallest suggesting process # at the head
  end if
  While message at head of queue has status deliverable do
    deliver the message at the head of the queue
    remove this message from the queue
  end while

This algorithm is based on the description in your text[2] on pages 495 and 496 and on the description of the protocols uses in the ISIS system as described by Birman and Joseph[1].

To show that this algorithms is correct, we must show that all processes order all messages in the same order. To show this, it is sufficient to prove the following claim.

Claim: *For any pair of messages, m and m', which are both multicast to group g, if m is delivered before m' at some process p $\in$ g, then all processes in g deliver m before m'.*

Proof by contradiction. Let process p first deliver m and then m' and let process $q \in g$ first deliver m' and then m. Let a be the actual sequence number for m and a' the actual sequence number for m'. Let s_p and s_p' be the suggested sequence numbers by p for messages m and m' respectively. And let s_q and s_q' be the sequested sequence numbers by q for messages m and m' respectively. This is summarized by the following chart.

Sequence numbers	Suggested by p	Suggested by q	Actual
m	s_p	s_q	a
m'	s_p'	s_q'	a'

Note that at process p m reached the head of the queue in deliverable state first and in process q m' reached the head of the queue in deliverable state first.

There are three possible cases.

1. If $a < a'$ then for q to deliver m' before m it must be case that m' was delivered before it received the actual sequence number of m (otherwise m would be higher in the queue than m' by the time m' was in the state deliverable.) Furthermore, it must be the case that $s_q \geq a'$ (otherwise the undeliverable message m would be higher in the queue than m' and m' could not have been delivered.) But this implies that $s_q > a$ (since $a' > a$). This is not possible, since a's value is the largest suggested value. Thus q could not have suggested a value larger than a and it is not possible that $s_q > a$. Thus, this case is not possible.
2. If $a > a'$ then for p to first deliver m before m' it must be the case that m was delivered before it received the actual sequence number of m'. Furthermore, it must be the case that $s_p' \geq a$. But this implies that $s_p' > a'$ which is not possible by the choice of a'. Hence, this case is also not possible.
3. If $a = a'$ then a and a' must have been suggested by two different processes (since a process never suggests the same sequence number twice.) Let i be the smallest id of all the processes that suggested a and j the smallest id of all the processes that suggested a'. This divides into two subcases:
    a. If $i < j$ then all should have delivered m before m' if both messages were deliverable on the queue at the same time. Hence, process q must have received a' before a and placed m' with a' at the head of the queue before a was received by q. Since m' was placed at the head of the queue, it must be the case that $s_q > a'$. This implies that $s_q > a$ which is not possible by the choice of a. Thus, this case is also not possible

b. If $i > j$ then all should have delivered m' before m if both messages were deliverable on the queue at the same time. Hence, process p must have received a before a' and placed m with a at the head of the queue before a' was received by p. Since m was placed at the head of the queue, it must be the case that $s_p' > a$. This implies that $s_p' > a'$ which is not possible by the choice of a'. Thus, this case is also not possible.

Thus, none of the three cases is possible, which leads us to a contradiction. Hence, it is not possible for two processes to deliver two messages in a different order in the ISIS algorithm. □

References

1. Birman, K. and Joseph, T. Reliable Communication in the Presence of Failures. In *ACM Transactions on Computer Systems 5*, 1 (February 1987), 47-76.
2. Coulouris, G., Dollimore, J. and Kindberg, T. Distributed Systems: Concepts and Design. 4th edition (2005), Addison-Wesley.