REDUCING LATE MESSAGE DISCARD RATE OF SOFT REAL-TIME CONSTRAINED CAUSAL ORDER DELIVERY

JINHO AHN

Dept. of Computer Science, Kyonggi University, Republic of Korea
E-mail: jhahn@kgu.ac.kr

Abstract- Most of existing deadline constrained causal order broadcast algorithms force any group member to drop late messages received before the expiration of their deadlines, but not respecting causal order condition. However, their users want to see as many messages as possible in their cause-effect order within the earliest deadline among them. In this paper, we propose a highly efficient real-time constrained causal order broadcast algorithm to highly improve responsiveness and minimize the number of late messages discarded. In order to satisfy these features, the protocol should maintain each message broadcast to a group into the volatile storage of every group member. Also, it allows in-transit predecessors of each received message to be obtained from another group member as fast before its deadline as possible

Keywords- Distributed System, Group Communication, Message Delivery Order, Real-Time Constraint.

I. INTRODUCTION

Causal order delivery to a broadcast group is a very important issue in the fields of sensor networks, video conferencing, stock trading, auction sales and so on [6, 7]. This message ordering condition can be satisfied if any two message sending events have cause-effect relation and the same destination, their corresponding delivery events should occur on the destination in their sending order. In order to ensure this ordering constraint, two approaches may generally be used as follows. First, if a group member receives a message capable of violating the constraint, the message delivery to the application is forced to wait for releasing the restriction caused by its predecessors [1, 4, 5]. Second, if deadline-constrained causal order requirement should be guaranteed, late messages, whose deadlines have passed or whose successors already received have exceeded their deadlines, are discarded. In the latter case, their users want to see as many messages as possible in their cause-effect order within the earliest deadline among them. Many researches on developing application-level broadcast communication algorithms have been performed to try to satisfy one of the three properties such as causal message ordering, message deadline assurance and high responsiveness.But, to the best of our knowledge, there exists no algorithm to satisfy all the three properties so far [2, 3, 8]. In this paper, we propose a highly efficient real-time constrained causal order broadcast algorithm to highly improve responsiveness and minimize the number of late messages discarded like in Fig. 1.This feature may help each group member locally make correct decisions with consistent information obtained from the messages even in soft real-time constrained environments.

II. SYSTEM MODEL

Deadline-constrained message delivery requires existence of a global clock where all agents can get their clock values and sequence causally dependent events by assigning the values to them. For this purpose, several global clock synchronization protocols whose clock drift values range from 5 to 10 ms have been developed. The deadline of a message m is measured as the global clock value before which message m must be delivered at least, including its local clock drift time. Therefore, if the deadlines of all messages scheduled to be sent are reasonably assigned, most of them can be satisfied for delivering their corresponding messages to the applications. But, message lost or congestion may incur message transfer delay, resulting in exceeding the deadline(s) of itself or its successors, which should drop the message in the soft real-time environments.



Fig. 1. The three properties of the proposed protocol.

III. PRELIMINARY

A. Causal Order Delivery to a Broadcast Group

First of all, in order to ensure the first property called causal message ordering, when a group member receives a message capable of violating the constraint, the delivery of the message to the corresponding application is forced to wait for releasing the restriction caused by its predecessors.

The previous algorithms can be divided into the following two approaches.The first approach is

designed for optimizing the amount of the control information piggybacked on each sent message.

The second one is to try to reduce the delivery time of the message to the corresponding application by piggybacking all of its predecessors on each sent message.But, generally the first one is preferred.Let me show you an example in Fig. 2for this approach.In this example, there is a broadcast group consisting of four processes p, q, r and s.The three processes p, q and s broadcast three messages m1, m2 and m3 to their own group in order.But, in this case, as the process r receives the third message m3 from s, r hasn't received m3's predecessors, m1 and m2, yet.Therefore, m3's delivery should be delayed until m1 and m2 are received and delivered at this point.Like in this example, the casual order delivery algorithm with vector timestamp piggybacked can ensure the first constraint.

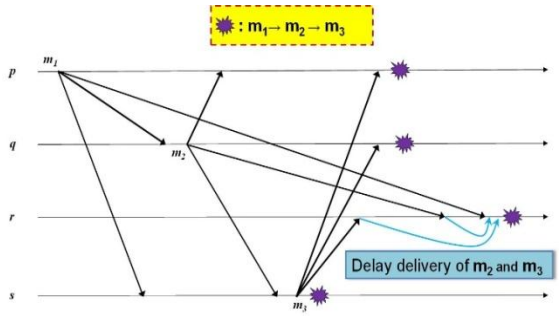


Fig. 2. An example showing how to ensureinter-message causality relation in message delivery delaying approach.

IV. THE LATE MESSAGE DISCARDING RATE REDUCING PROTOCOL

Their users want to see as many messages as possible in their cause-effect order within the earliest deadline among them. Thus, our proposed protocol intends to be designed having the following features.

- Minimize the number of late messages discarded.
- Improve responsiveness highly.

Therefore, in order to satisfy these features, the protocol should maintain each message broadcast to a group into the volatile storage of every group member. Also, it allows in-transit predecessors of each received message to be obtained from another group member as fast before its deadline as possible.

Let us show how our protocol execute to achieve this goal using an example.In Fig. 3 and Fig. 4, there is a broadcast group consisting of 4 processes, p, q, r and s, sending 3 messages, m1, m2 and m3, to all members in order (by executing Module B-SEND(m)), whose deadlines are $deadline_{m1}$, $deadline_{m2}$ and $deadline_{m3}$, respectively. In the previous deadline constrained algorithms[2, 3, 8], r cannot receive m1 and m2 except for delivering m3 in Fig. 3. In order to receive as many messages as possible before their earliest deadline like $deadline_{m3}$, our proposed protocol allows each member like p and q to buffer received messages in its memory, $DLVD_Q_{rcvr}$ (by executing Module B-RECV(m, $deadline_m$, $MVector_{sndr}$)) like in Fig. 4. If a member, r, receives a message like m3, from s in this figure, it requests m3's sender, s, gives m3's predecessors, m1 and m2, to itself by sending a solicitation message with m3's dependency vector, $MVector_{rcvr}$, like in Fig. 5 (by executing Module SOLICIT-RECV($MVector_{rcvr}$)). After having obtained m1 and m2 from s, r can deliver all three messages to their corresponding application (by executing Module RPY-RECV(MSG_Q)). In order to keep the deadline-constrained causal order requirement, our protocol makes each member check deadline violation every time interval (by executing Module CHECK-MSGS()).

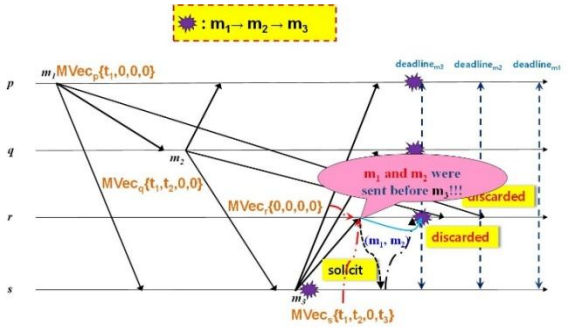


Fig. 3. An example showinglimitation ofthe previous approach.

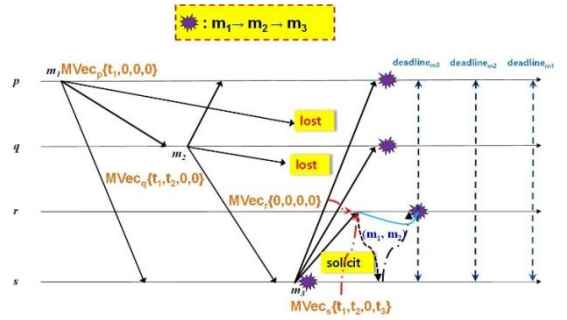


Fig.4. An example showinghow our protocol supports high responsiveness.

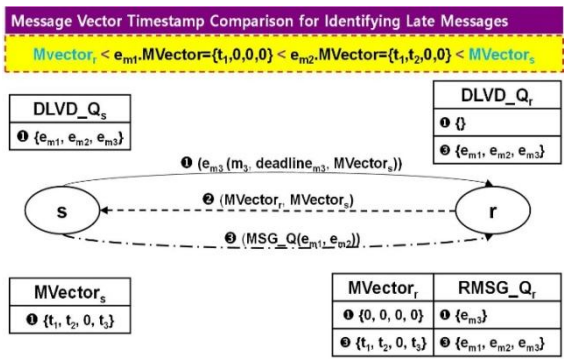


Fig.5. Interaction procedure of our proposed protocol.

CONCLUSION

In this paper, we presentedeffective deadline-constrained broadcast protocol to ensure

inter-message causality and highly reduce late message discarding rate with high responsiveness.The protocol can make every group member keep each message broadcast to its group on its volatile storage. Also, it enables late messages preceding each received message to be given to its receiver from another group member as quickly before its deadline as possible.
For future works, we intends to adapt it to our soft real-time system to run deadline-constrained collaborative applications to measure how much better this work performs compared to the existing protocols.

ACKNOWLEDGEMENT

This work was supported by Kyonggi University Research Grant 2014(Project Number: 2014-012).

```
Module B-SEND(m) OF P_sndr
MVector_sndr[sndr] ← current time value of P_sndr ;
broadcast (m, deadline_m, MVector_sndr) to all the other members ;
Module B-RECV(m, deadline_m, MVector_sndr) OF P_rcvr
if((deadline_m < current time value of P_rcvr ) ∨
  (MVector_sndr[sndr] ≤ MVector_rcvr[sndr])) then
  discard message m from P_rcvr ;
elseif((MVector_sndr[sndr] > MVector_rcvr[sndr])∧
  (∀i≠sndr: MVector_sndr[i] ≤ MVector_rcvr[i])) then
  ∀i: MVector_rcvr[i] ← max(MVector_rcvr[i], MVector_sndr[i]) ;
  deliver m to its corresponding application ;
  insert (m, deadline_m, MVector_sndr) into DLVD_Q_rcvr
  in m's sending time order ;
  call CHECK-MSGS() ;
else
  insert (m, deadline_m, MVector_sndr) into RMSG_Q_rcvr
  in m's sending time order ;
  send solicitation(MVector_rcvr, MVector_sndr) toP_sndr ;

// Every time interval, the procedure is executed.
Module CHECK-MSGS() OF PROCESS P_p
for all e ∈ RMSG_Q_p in FIFO order do
  if(e.MVector_j[j] > MVector_p[j]∧∀i≠j: e.MVector_j[i] ≤
  MVector_p[i]) then
  ∀i:MVector_p[i]←max(MVector_p[i], e.MVector_j[i]) ;
  deliver e.m to its corresponding application ;
  insert e into DLVD_Q_p in e.m's sending time order ;
  remove e from RMSG_Q_p ;
elseif(e.deadline_m = current time value of P_p) then
  for all c ∈ RMSG_Q_p in FIFO order st (c.MVector_k≤
    e.MVector_j) do
    ∀i: MVector_p[i] ← max(MVector_p[i], c.MVector_k[i]) ;
  deliver c.m to its corresponding application ;
  insert c into DLVD_Q_p in c.m's sending time order ;
  remove c from RMSG_Q_p ;

Module SOLICIT-RECV(MVector_rcvr, MVector_upper) OF P_sndr
MSG_Q ←Φ ;
for all e ∈ DLVD_Q_sndr in FIFO order st
  ((∀i: e.MVector[i] < MVector_upper[i])∧
  not(∀j: e.MVector[i] ≤ MVector_rcvr[i])) do
  insert e into MSG_Q in e.m's sending time order ;
  send reply(MSG_Q) toP_rcvr ;

Module RPY-RECV(MSG_Q) OF P_rcvr
for all e ∈MSG_Q in FIFO order do
  ∀i: MVector_rcvr[i] ← max(MVector_rcvr[i], e.MVector_j[i]) ;
  deliver e.m to its corresponding application ;
  insert e into DLVD_Q_p in e.m's sending time order ;
  remove e from RMSG_Q_p ;
```

Fig. 6. Modules for our real-time constrained broadcast protocol ensuring message causality.

REFERENCES

[1] R Baldoni, "A Positive Acknowledgment Protocol for Causal Broadcasting", IEEE Transactions on Computers, vol. 47, no. 12, pp. 1341-1350, 1998.

[2] R. Baldoni, A. Mostefaoui and M. Raynal, "Causal Delivery of Messages with Real-Time Data in unreliable Networks", Real-Time Systems Journal, vol. 10, no. 3, pp. 245 -262, 1996.

[3] R. Baldoni, R. Prakash, M. Raynal and M. Singhal, "Efficient Delta-Causal Broadcasting", Journal of Computer Systems Science and Engineering, vol. 13, no. 5, pp. 263-270, 1998.

[4] K. Birman and T. Joseph, "Reliable Communication in the Presence of Failures", ACM Transactions on Computer Systems, vol. 5, no. 1, pp. 47-76, 1987.

[5] C. Kim and J. Ahn, "Causal order multicast protocol using minimal message history information", the 12th international conference on Algorithms and Architectures for Parallel Processing, vol. 1, pp. 546-559, 2012.

[6] J. Fanchon, K. Drira, S. P. Hernandez, "Abstract channels as connectors for software components in group communication services", Computer Science, ENC 2004. Proceedings of the Fifth Mexican International Conference in, pp. 88-95, 2004.

[7] C. Plesca, R. Grigoras, P. Queinnec, G. Padiou, and J. Fanchon, "A coordination-level middleware for supporting flexible consistency in CSCW", 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2006.

[8] L. Rodrigues, R. Baldoni, E. Anceaume, and M. Raynal, "Deadline-constrained causal order", Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 234-241, 2000.

★★★