Message Dissemination Algorithm for Unreliable Broadcast Networks Guaranteeing Causal Order and Deadline Constraints

Jinho Ahn
Department of Computer Science, College of Natural Sciences
Kyonggi University
94-6 Iuidong, Yeongtonggu, Suwon Gyeonggi 443-760, 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.**

Keywords-distributed system; realtime constraint; group communication; broadcast; message delivery order

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. However, the previous deadline-constrained causal order delivery algorithms [2, 3, 8] may not satisfy this important requirement. 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.

II. The Proposed Algorithm

In figure 2, there is a broadcast group consisting of 4 processes, p1, p2, p3 and p4, 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], p3 cannot receive m1 and m2 except for delivering m3 in this example. In order to receive as many messages as possible before their earliest deadline like $deadline_{m3}$, our proposed algorithm allows each member like p1 and p2 to buffer received messages in its memory, $DLVD\_Q_{rcvr}$ (by executing **Module** B-RECV(m, $deadline_m$, $MVector_{sndr}$)). If a member, p3, receives a message like m3, from p2 in this figure, it requests m3's sender, p2, give m3's predecessors, m1 and m2, to itself by sending a solicitation message with m3's dependency vector, $MVector_{rcvr}$, (by executing **Module** SOLICIT-RECV($MVector_{rcvr}$)). After having obtained m1 and m2 from p2, p3 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 algorithm makes each member check deadline violation every time interval (by executing **Module** CHECK-MSGS()).

Module B-SEND(m) OF P_{sndr}
  $MVector_{sndr}[sndr] \leftarrow$ 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}) $\vee$
    ($MVector_{sndr}[sndr] \leq MVector_{rcvr}[sndr]$)) **then**
    discard message m **from** P_{rcvr} ;
  else if(($MVector_{sndr}[sndr] > MVector_{rcvr}[sndr]$) $\wedge$
    ($\forall i \neq sndr$: $MVector_{sndr}[i] \leq MVector_{rcvr}[i]$)) **then**
    $\forall i$: $MVector_{rcvr}[i] \leftarrow$
      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($MVector_{rcvr}$, $MVector_{sndr}$) **to** P_{sndr} ;

// Every time interval, the procedure is executed.
Module CHECK-MSGS() OF PROCESS P_p
  for all e $\in$ $RMSG\_Q_p$ in FIFO order **do**
    if($e.MVector_j[j] > MVector_p[j] \wedge \forall i \neq j$: $e.MVector_j[i] \leq$
      $MVector_p[i]$) **then**

$\forall$i:MVector$_p$[i]$\leftarrow$ **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$_p$;
else if(e.deadline$_m$ = current time value of P$_p$) **then**
for all c $\in$ RMSG_Q$_p$ in FIFO order st
(c.MVector$_k$ $\leq$ e.MVector$_j$) **do**
$\forall$i: MVector$_p$[i] $\leftarrow$
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 $\leftarrow \Phi$;
for all e $\in$ DLVD_Q$_{sndr}$ in FIFO order **st**
(($\forall$i: e.MVector[i] < MVector$_{upper}$[i]) $\wedge$
not($\forall$j: e.MVector[i] $\leq$ MVector$_{rcvr}$[i])) **do**
insert e **into** MSG_Q in e.m's sending time order ;
send reply(MSG_Q) **to** P$_{rcvr}$;

Module RPY-RECV(*MSG_Q*) OF P$_{rcvr}$
for all e $\in$ *MSG_Q* in FIFO order **do**
$\forall$i: MVector$_{rcvr}$[i] $\leftarrow$
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$_p$;

Figure 1. Procedures for each broadcast group member.

ACKNOWLEDGMENT

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant number: 2012R1A1A2044660).

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 Δ-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.

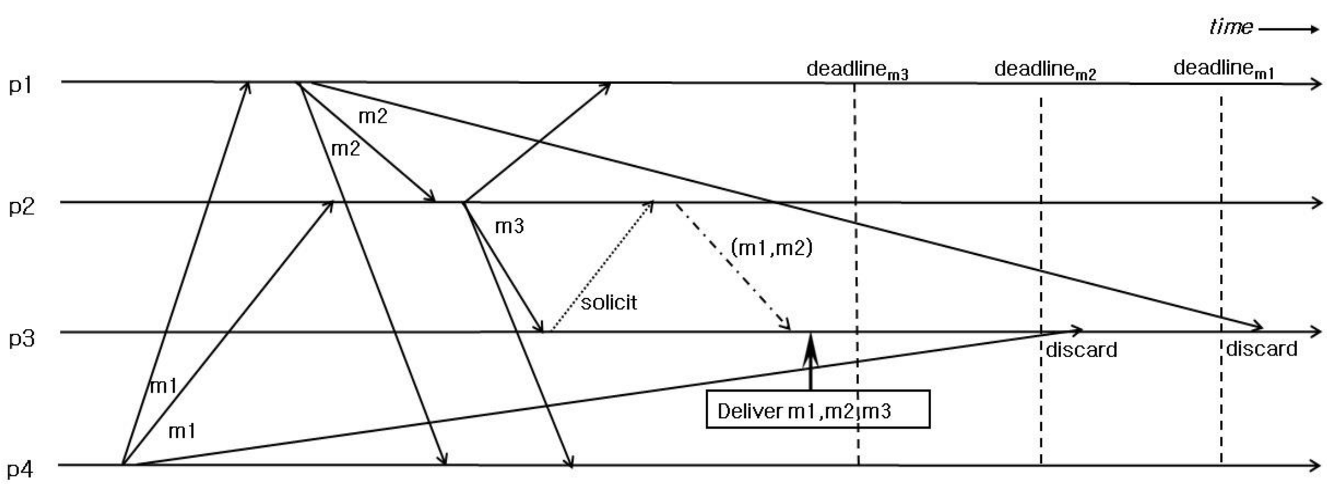

Figure 2. An example of execution of our proposed algorithm supporting its high responsiveness.