Performance Evaluation of Real-Time Message Delivery in RDM Algorithm

Shabnam Mirshokraie[1], Mojtaba Sabeghi[2], Mahmoud Naghibzadeh[1], Koen Bertels[2]
[1]*Computer Engineering Department*
Ferdowsi University of Mashhad
Mashhad, Iran
{mirshokraie, naghib}@um.ac.ir
[2]*Computer Engineering Laboratory*
Delft University of Technology
Delft, the Netherlands
{sabeghi, k.l.m.bertels}@ce.et.tudelft.nl

Abstract

Complexity of distributed real-time applications such as automotive electronics has increased dramatically over the last couple of years. As a result, developing communication protocols to address real-time requirements of these applications such as reliability, in-time delivery of messages, priority support, and fault-tolerance needs more sophisticated techniques. To satisfy these requirements, special communication architectures have been designed. The most important methods presented are TDMA (Time Division Multiple Access), Token Ring and CAN (Controller Area Network). Recently, a new algorithm called RDM (Round Data Mailer) has been presented. The focus of this paper is on the studying of the Round Data Mailer technique and evaluating its message delivery performance by making a comparison of the result of simulated RDM and CAN.

1. Introduction

Improvement in many fields of modern technology like traffic control and atomic reactors depends heavily on accurate operation of fast software and hardware. Therefore all these systems need to provide services and operations with deadline.

A Real-Time System (RTS) is defined as a system in which the time is significant when the outputs are produced [1]. The outputs must be produced within specified time bounds referred to as deadlines so RTS systems have higher reliability and safety requirements than other systems. Fault tolerance which is another important attribute for real-time systems refers to protect and maintaining availability of system services despite occurring errors during its operation.

Nowadays, central real-time systems have been converted to the distributed ones, because many systems which must be controlled are naturally distributed like traffic control or communication systems. TDMA, CAN and IEEE802.5 priority Token ring have been used in such environment to provide safety and reliability in real-time systems [2, 3, 4]. With TDMA and CAN protocols, there is no way to make sure that every message is delivered in time through an efficient global priority-based message delivery mechanism. Although IEEE802.5 priority Token ring supports priorities, it has some other difficulties. The most important problem is about token traveling. To find the next highest priority message a token must travel at least one complete cycle. If there are many nodes in system, it can be a considerable amount of time. In this paper we explain another method called RDM which satisfies the requirements of SSDRT (Small Scale Distributed Real-Time) systems [5]. Although, this method can be used in general distributed systems, our focus is on small-scale real-time ones.

2. RDM Algorithm

In RDM, a logical ring is assumed to connect computers to each other [5, 6, 7]. Most of benefits derived from the ring topology are related to its logical characteristics. For example in a ring connectivity data is quickly transferred without a bottle neck, the transmission of data is relatively simple as packets travel in one direction only and adding additional nodes has very little impact on bandwidth. Because of these benefits, in RDM, ring topology is considered as a logical level like many other network protocols.

Critical message moves clockwise or counterclockwise from one node to its neighbor in assumed logical ring. Critical message is a message that carries real-time data/results which is a datum/result that is captured or produced by a real-time process and there are one or more real-time processes that need this datum/result. In RDM there is only one such a message that travels between nodes. When RDM message reaches to a node, that node has permission to send critical data, so it acts like token for each node but with a small different from known token, because it carries critical data too, which is not common in other algorithms. Structure of this message must be known exactly for all nodes in system.

Each data in the message has only one producer, but it can have one or more consumer. When the critical message reaches to a node, it makes a copy of the whole message, then removes or updates all data/results loaded by this node in the previous round and loads all newly produce real-time data/result on the message. It also searches in whole message in order to find critical data which has been sent for this node. Finally, the node will send the message to its clockwise neighbor. In explained method because of message structure specifications and specific behavior of RDM algorithm there is no problem like token traveling or any other similar difficulties.

One of the important requirements of real-time systems is reliable and timely data transmission. So, in RDM algorithm, in order to provide reliability for real-time data transmission, each node must recover from data that is damaged or lost by the communication system. This is achieved by requiring an acknowledgment from the receiver. If the acknowledgement is not received with in a predefined period of time, the receiver is checked to distinguish whether the receiver node is active or not, if the node is still active and works properly the data is retransmitted. Whenever a segment containing data is transmitted, a copy is created on a retransmission queue and also a timer will be started, when the acknowledgment for that data is received properly, the segment is deleted from the queue and the timer will reset, but if the acknowledgment is not received before the timer runs out while the receiver is still active the segment will be retransmitted. In the case that the acknowledgement is not received after twice repetition of data retransmission, the receiver will be assumed inactive and the message transmits to the next node in the logical ring.

2.1. RDM Message Structure

Being familiar with the algorithm of RDM, it is time to know about RDM message structure and its specifications. As we described before there is only one RDM message for whole system that travels between nodes and carries critical data. Fig. 1 illustrates critical message structure. Every data which is needed by nodes will be kept in this message. For fields of message a short description follows:

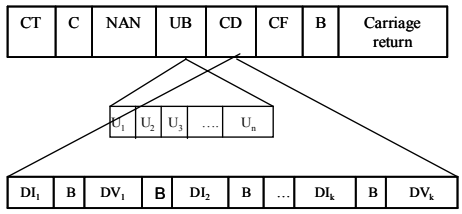
Coordinator Time holds the coordinator's local time which is updated when the critical message passes through the coordinator, just before the message leaves it.

Coordinator contains the current coordinator's identification number.

Updated fields are a collection of Boolean data, one per every node. Every Boolean field shows the corresponding node situation (active or non-active). When the critical message gets to a node, the corresponding field will be set, so the coordinator will recognize inactive nodes. Coordinator's clockwise neighbor will make sure if the coordinator is active and in the case that coordinator is inactive, it will take over the coordinator's responsibility and will update the C field.

Critical data includes the actual critical data. Each critical datum contains data identification and data value. Data value properties will be recognized by its identification number. Also, the critical data will be sent or received based on these numbers.

Check field is used to handle the mistakes, which can be occurred during receiving data. In this structure, we assumed that Data identification and Data value have variable length. The number of subfields may vary as the message passes through different nodes of the system. Because of variable length of different field, blanks are used to distinguish between different parts of message.



- CT:Coordinar Time
- C:Coordinator
- NAN: Number of Active Nodes
- UB: Updated fields
- CD: Critical Data
- CF: Check field
- U_i: User i updated field
- DI_i: Data Identification *i*
- DV_i: Data Value *i*
- B: Blank
- Carriage return: Recognize end of message

Figure 1. Critical message structure

2.2. RDM Fault-Tolerance

Fault tolerance is one of important characteristics of real-time systems that prevent system from destruction. During the system operation, if any changes occur in its status, system must follow some actions and rules in order to prevent system from loosing data or total collapse of mission. In this section, we describe RDM fault tolerance methods which are applied in critical situations.

Software and hardware methods are two main procedures that are used for fault-tolerance in systems, but our focus is on software one. Here we classify system errors in to 3 major groups: Process inactivity, node disablement and coordinator inactivity and for each group we present the fault tolerance mechanisms [8].

In the case that one of system processes became inactive, since all system requirements and produced data are defined exactly, first we have to determine all data and messages which are produced by this process; next step is to determine if there is any other process in system that produces the same data. If there is such a process, we can replace the inactive one with this new process and refer

all processes that need the inactive node data to the selected process. But in the case that there is no such a process that can be replaced with disabled one, we can examine whether it is possible to activate the new copy of inactive process or not. In order to provide this potentiality, it is possible to use software or hardware redundancy in system design [3, 9]. It means that we can make a copy of system resources and processes based on their importance. Naturally, if one of system resources doesn't support redundancy capability and only one copy of this resource is available, in the case that it becomes inactive, it may cause critical situation based on resource effects and its necessity for system. After explained actions, these changes will be announced to the other nodes in a reconfiguration message.

Node disablement is another type of change that can be occurred during the system activity. In this case, all actions have been described previously must be done for each process in the disabled node. If it is necessary, a new system setup will be done to renew nodes connectivity.
And finally, if coordinator becomes inactive, since each node has one inactive copy of coordinator process, it will be recognized by its clockwise neighbor and the neighbor node will be introduced as a new coordinator. It also takes over the coordinator responsibility and the reconfiguration process will be done.

3. Theoretic Comparison

In previous section we describe RDM algorithm and explain how it works. In this part after we compare RDM with three important protocols including Token Ring, TDMA and CAN, we refer to the most important advantages of RDM algorithm and some of its limitations, in order to obtain general conclusion of RDM behavior.

3.1. Comparison with Token Ring

1) In Token ring, complex operations must be done in order to manage and control the media especially when we have priorities [10]. In this case token must travel at least one complete cycle in order to identify the next highest priority sender [11, 12], but in RDM without any complex operation, in each moment the bus owner is defined [5, 6, 7, 13].

2) In a SSDRT system, correct execution of a request often depends on the freshness of the data that are received from other nodes. Freshness of data is the Data Life-Time which is the time interval between data production in source node and data consumption in destination node. with the Token ring every data is delivered separately, while with the RDM a collection of data are encapsulated in one message and then it is ready to send, so data delivered by the RDM is fresher than data delivered by Token ring. This is a major advantage of the RDM over the Token ring [5, 13].

3) Unlike RDM, in Token Ring the exact time of send and receive is not computable, which is really important for hard real-time systems and could be a major deficiency for Token Ring, but in RDM it is possible to present an exact formula for computing these times [5, 6, 13].

3.2. Comparison with CAN

1) In CAN, exact computation of response time is difficult and depends on many environmental factors, but in RDM it is possible to compute accurate request and response time by presenting exact formulas [14].

2) In CAN because of using message broadcast mechanism, all nodes on the network receive the source node message [14, 15, 16] which strongly eliminates data security. But in RDM we can establish some limitative rules to provide secrecy [5]. The explanation of RDM security scheme is out of the paper scope and can be described in details in future works.

3) In RDM, add or remove nodes increase the system complexity which makes it more complicated than CAN [14].

4) In RDM practically there is no limitation on message length as far as its round trip doesn't exceed maximum circulation period, but in CAN message length is bounded to 8 bytes and fragmentation is required for messages longer than 8 bytes [5, 15].

3.3. Comparison with TDMA

1) In TDMA time span allocation for each process is based on worst state of message transmission and because of this it is not possible to use the band width efficiently [5, 17], however in RDM because of special mechanism which is used in send and receives (just one node uses media in each moment) optimal use of band width is possible.

2) Operation of TDMA is based on time accuracy. Clock synchronization to provide time accuracy increases system cost and overhead, while in RDM there is no need for clock synchronization [5, 18].

3) Unlike RDM, TDMA is a static method and doesn't support dynamic models [5, 19].

4) Because of time accuracy in TDMA start and end of each process job can be measured exactly and this is a major advantage of TDMA toward other protocols [20].

3.4. RDM characteristics

After comparison of RDM with other protocols, we summarize most important specifications of RDM. The followings are advantages of using the RDM:

- The types of real-time data that are transferred between processes are known in advance and the RDM makes use of this information to design a specific message structure with capability of saving message delivery time.
- The RDM technique can be used in small-scale distributed real-time systems which contain independent data and control sharing processes [13].
- RDM uses special message structure in order to keep all critical data in one message, so there is no need to deliver each data separately like Token ring [21]. Because of this, data delivered by RDM is fresher than Token ring.
- One node is the coordinator and it has the responsibility of system establishment and timing considerations. When the system is going to become active, the coordinator will introduce all real-time data to all nodes. It will also recognize inactive nodes. If the coordinator becomes inactive, it will be recognized in the shortest possible time and replacement will be done.
- Because of special design of RDM, There is no need for clock synchronization. This will save tremendous amount of overhead.
- Because of special manner of RDM, in each moment the bus owner is known, therefore there is no problem to identify the next sender.

The followings are limitations of using the RDM [5]:

- If message length becomes very long, it will have negative effect on message round time and triggers to increase it.
- Increment of message length leads to message buffering in nodes and it can increase sending time.
- Automatic adding nodes may cause system complexity which is against RDM nature, so manual adding must be done.

4. Simulation

In this section, first we describe simulation of RDM and CAN algorithms over TCP/IP. It should be mentioned that we just considered the message passing characteristics of this two. And because of that we call them algorithm here rather than network protocol. For the purpose of system setup first step is to specify coordinator

After defining coordinator, system processes and their important parameters like execution time, period and deadline must be defined exactly. After process definition, each process specifies data that consumes or produces. These data and information organize in a source list which is kept in coordinator. Based on this source list, coordinator makes a system source table which controls access authorities. In this stage Coordinator distributes all processes in to different work stations. It also makes a logical circular ordering of nodes that connects stations to each other and assigns data movement direction. Based on the nodes location, an identification number assigns to them and by this assigning, all nodes communicate with each other according to id numbers. In last step of system setup, all these organized information and details which are needed for activities such as sending, receiving and allocating of data during processes operation, will be sent to all nodes.

In previous descriptions we referred to system setup and configuration, now we want to explain software environment and processes specifications in our simulation. In this simulation we assumed eight real time processes with different specifications in execution time and deadline (deadline is the time in which the process should be finished). The characteristics of each process are selected by a program and completely random. Table 1 shows the list of these processes and their details.

In RDM we assumed three main nodes and different processes were distributed on these nodes. Process 1, 2 and 3 are placed in the first node, process 4 and 5 are placed in the second node and process 6, 7 and 8 are distributed in the third node.

In CAN all processes were placed on a common bus and their settlement ordering is based on their numbers. In CAN another process is also assumed which is called Monitor. This process has the responsibility of system controlling during the system operation.

TABLE 1. System Processes Specifications

Process No	Execution Time(s)	Deadline(s)	Time interval between two requests(ms)
1	6	8	11
2	8	11	16
3	13	16	18
4	9	14	22
5	11	15	25
6	16	18	28
7	12	16	34
8	15	18	39

5. Simulation Results

We simulated RDM and CAN with explained specifications. We also considered 100 requests for each process during processes execution. In this simulation our main goal is to compute average Response Time and Average Data Life-Time in RDM and CAN and then compare the results. Response Time is the amount of time that is passed, since the request is sent, until the response is received. Data Life-Time is another factor which means time interval between data production in source node and data consumption in destination node. After data storage during simulation, we also compute maximum Data Life-Time and maximum Response Time by statistical operations.

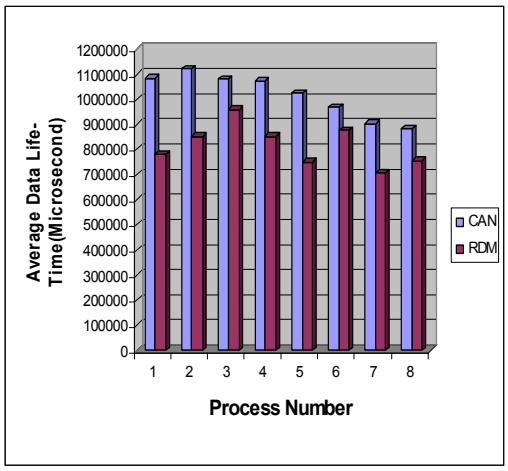


Figure 2. RDM and CAN comparative chart for average data life tim

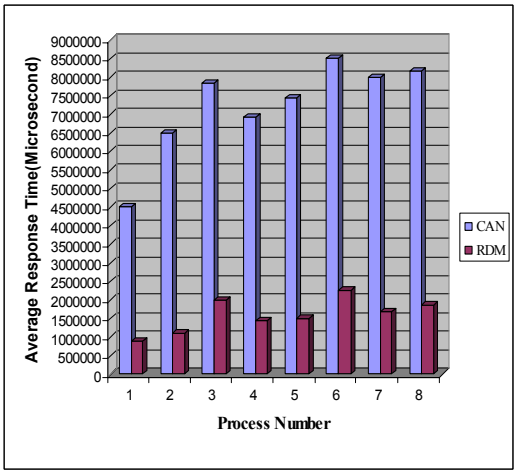


Figure 3. RDM and CAN comparative chart for average response time

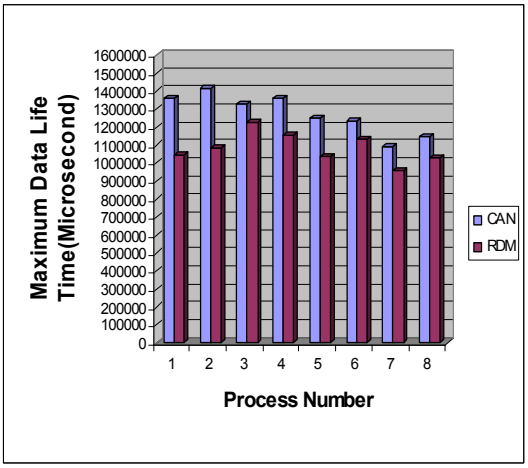


Figure 4. RDM and CAN comparative chart for maximum data life-time

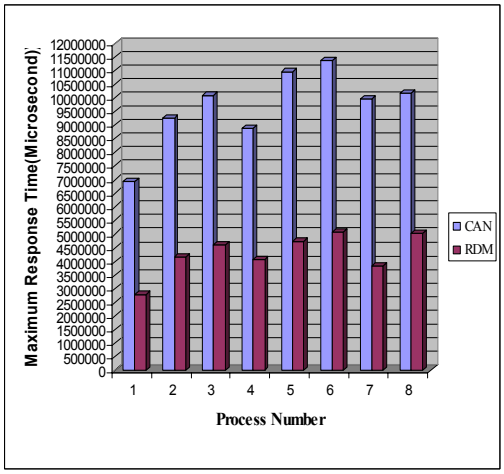


Figure 5. RDM and CAN comparative chart for maximum response time

The average Data-Life Time and average Response Time results are shown in Fig. 2 and 3 and the maximum Data-Life Time and maximum Response Time results are shown in Fig. 4 and Fig. 5 (Time measurement unit is microsecond).

As we can see in these figures RDM acts better than CAN based on all measured factors, however there is no big difference between these two algorithms in Average and maximum Data Life-Time, it is obvious that there is a considerable difference between RDM and CAN in average and maximum Response Time. The main reason behind this difference is RDM special message structure and the way that each node sends its real-time data to the destination. In fact in CAN each message is delivered separately, while in RDM there is only one message that circulates around the logical ring and carries all real-time data. In other words, a collection of data is encapsulated in one message and then it is made ready to send, so in RDM we can achieve better use of bandwidth than CAN.

Just as we can see in the obtained results from average and maximum Data-Life Time, RDM and CAN algorithms have similar efficiency in this factor with a little improvement in RDM in comparison with CAN. But as it is obvious in Response Time results, RDM with its short Response Time can be an ideal choice for systems with short intervals. Furthermore RDM algorithm seems much more suitable for systems with short deadline time, especially in hard real-time systems. Its special mechanisms in message passing decreases the probability of deadline expiration and falling system in a critical situation, so it increases data freshness (freshness is Data Life-Time which is the time interval between data production in source node and data consumption in destination node), security and generally system safety which is really important for real-time systems. With comparing results, as a general conclusion in real-time systems with few numbers of nodes and longer deadlines,

RDM and CAN have approximately similar efficiency, but in distributed real-time systems with short deadlines specially heard real-time types RDM is a very good choice with much more efficiency and can guarantee to transmit data to the proper destination in appropriate time without deadline expiration.

6. Conclusion

In this paper, we explained the RDM algorithm, its message passing mechanism and its message structure. We weight up the pros and cons of the RDM technique by making a comparisons between RDM and other algorithms. We also referred to our simulation results for RDM and CAN which shows the better performance of RDM than CAN in hard real-time systems with short deadlines.

References

[1] M. Joseph, "*Real-time Systems: Specification, Verification and Analysis*", Prentice Hall International, London, 1996.

[2] ANSI X3T9.5, "*FDDI Token Ring Media Access Control*", May 1987.

[3] H. Kopetz, A. Damm, C. Koza. M. Mulazzani, W. Schwabl, C. Senft, and R. Zainlinger, "Distributed Fault-Tolerant Real-Time System," *The Mars Approach, IEEE Macro*, Feb. 1989, pp25-40.

[4] L. M. Pinho, F. Vasques and L. Ferreira, "Programming Atomic Multicasts in CAN," *10th International Real-Time Ada Workshop (IRTAW'2000)*, Avila, Spain, Sept. 2000, pp18-22.

[5] M. Naghibzadeh, "Round Data Mailer Message," *IEEE International Conference on Systems, Man and Cybernetics*, Hammamet, Tunisia, Oct. 2002, pp586- 591.

[6] M. Sabeghi, M. Naghibzadeh "Performance Assessment of a Distributed Real-Time Control System Utilizing RDM and RDM+ Protocols for Communication," *2nd International Conference on Future Networking Technologies (CoNEXT06),* ACM SIGCOMM, December 2006.

[7] M. Sabeghi, M. Naghibzadeh, K.L.M. Bertels, "RDM+: A New Mac Layer Real-Time Communication Protocol," *IEEE Sarnoff Symposium*, April 2007., in press.

[8] S. Punnekkat, Schedulability Analysis of Fualt Tolerant Real-Time Systems, University of York, UK, June 1997.

[9] M. Baleani, A. Ferrari, L. Mangeruca, M. Peri, S. Pezzini, and A. Sangiovanni-Vincentelli, "Fault-tolerant platforms for automotive safety-critical applications," *International Conference on Compilers, Architectures and Synthesis for Embedded Systems, CASES'03*, San Jose, USA, November 2003.

[10] S. D. Cote, CCP, CNE, "*Token-Ring Architecture (The Details Count When Choosing a Topology),"* 1996, http://www.bralyn.net/ techpages/ papers/ token.ring.html.

[11] The Institute of Electrical and Electronics Engineers, IEEE Standards for Local Area Networks: *Token-passing Bus Access Method and Physical Layer Specification*, IEEE, New York, 1985.

[12] The Institute of Electrical and Electronics Engineers, IEEE Standards for Local Area Networks: *Token Ring Access Method and Physical Layer Specifications*, 1985.

[13] H. Abachi and M. Naghibzadeh, "A message-passing protocol for small-scale distributed real-time systems," *World Automation Congress*, Spain, 2004.

[14] R. Bosch, *CAN specifications* ver 2.0, Postfach 50, 1991.

[15] L. Feng-Li, J. R Moyne, D.M Tilbury, "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet, " *Control Systems Magazine, IEEE* Volume 21, Issue 1, Feb 2001, pp. 66 – 83.

[16] M. Schofield, Controller Area Network - *How CAN Works*, 1996.

[17] S. Tanenbaum, "*Computer Networks*", Prentice Hall, Fourth edition, 2003.

[18] V. Claesson, H. Lnn, N. Suri, "An Efficient TDMA Synchronization Approach for Distributed Embedded Systems," *International Symposium on Reliable and Distributed Systems (SRDS*), 2001, pp. 198-201.

[19] L. Farman, U. Sterner, O. Tronarp, "Analysis of capacity in ad hoc networks with variable data rates," *Vehicular Technology Conference 2004 IEEE 59th, VTC 2004-Spring 2004*, Milan, Italy, May 2004, pp. 2101-2105.

[20] G. Grunsteidle, H. Kopetz, "A reliable Multicast Protocol for Distributed Real-Time Systems," *8th IEEE workshop on real-time operating systems*, Atlanta, GA, USA, 1991.

[21] D. E. Commer, "*Internetworking with TCP/IP*", Prentice-Hall, 1996.