Evaluation of the Communication Latency Over Real-time Channel in HARTS

Seungkweon Jeong, Jaehyun Park [a], Wook Hyun Kwon, and Kang G. Shin [b]

School of Electrical Engineering, Seoul National Univ., Seoul, Korea
[a] Department of Industrial Engineering, Inha Univ., Inchon, Korea
[b] Department of EEECS, The Univ. of Michigan, Ann Arbor, USA

Abstract

In this paper, the end-to-end communication latency in HARTS(Hexagonal Architecture for Real-Time Systems) is evaluated, which is depends on the intra-node delay as well as the inter-node delay. While the inter-node delay is bounded due to the point-to-point real-time channel, the intra-node delay is variable according to the message traffic and the internal structure of a HARTS node. The intra-node delay is evaluated analyically and by computer simulation for the various implementational models, and an arbitration algorithm is proposed to provide real-time scheduling for VMEbus-based HARTS node.

1 Introduction

In a distributed system, real-time applications require predictable communication performance such as bounds on latency or enough average throughput. To make the network predictable, lots of algorithms and architectures have been proposed[1, 2, 3, 5, 6]. However, since the proposed algorithms and architectures are based on the ideal model, they have hardly implemented. HARTS, implemented in the University of Michigan, is an experimental testbed designed to investigate the various issues in distributed real-time computing[7]. The single HARTS node, consisting of multiple processing elements, is connected with each other through the point-to-point real-time channel. While the network topology, routing algorithm, and switching scheme for HARTS were deeply analyzed in the previous works[8, 9, 10], the exact end-to-end latency of a real-time message has not been evaluated yet. Thus, in this paper, the exact communication latency over real-time channel in HARTS is evaluated with considering the hardware parameters as well as the switching algorithm and the node architecture.

The communication latency in HARTS is affected by the application-specific message traffic, the internal structure of each node, and the network strategy such as topology, routing algorithm, switching scheme, and router architecture. In the real-time applications of HARTS, the network strategy is adequately fixed at pre-runtime to remove the stochastic and undeterministic factors[4, 9, 10]. And the influence of the internal structure can be thought of the latency caused by the arbitration policy of VMEbus, since HARTS adopts the VMEbus as the system bus of each node. The application-specific message traffic has influence on the communication latency according to the quantity and the frequency of inter-node communication.

The real-time message can be generated periodically or sporadically. If the periodic message has the deterministic destination and size, its worst-case communication latency depends on scheduling of common resources like system bus and network channel. Though the rate monotonic scheduling algorithm shows a relatively good performance in scheduling this kind of periodic message, it has many difficulties to be used in a real system due to the limitated number of the priority level. Since VMEbus has limited number of signals to represent priority level, the real-time messages with different priority level may be processed as if they have equal priority level in HARTS. To implement fixed priority scheduling algorithm using the limited number of priority signals of VMEbus, a new bus arbitration algorithm is proposed in this paper.

Meanwhile, the sporadic messages may use the common resources by the FIFO policy or the EDF(Earlist Deadline First) scheduling policy. In either case, the bound of the communication latency is not estimated analytically. The communication latency of the sporadic tasks could be estimated by computer simulation. To evaluate the communication latency for sporadic messages with varying message traffic and arbitration algorithm, we models the HARTS node to the hardware-level using *SES/design 3.0*[17, 18].

0-8186-7626-4/96 $5.00 © 1996 IEEE

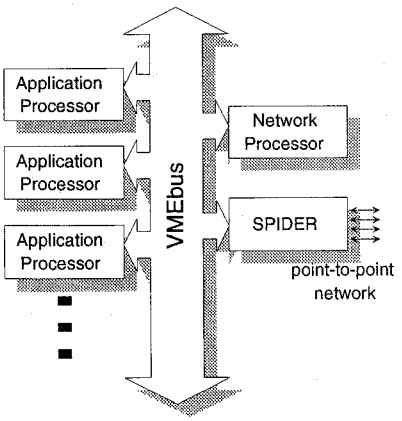

Figure 1: HARTS node block diagram

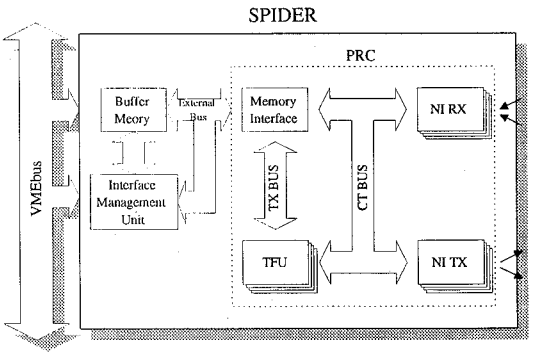

Figure 2: SPIDER block diagram

This paper is organized as follows. Section 2 states the detailed architecture of HARTS and the data transfer mechanism in it. Under the adequate switching scheme and bus arbitration strategy, the end-to-end communication latency in HARTS is analyzed in Section 3. In Section 4, we shows via examples and simulations the effect of the arbitration strategy. The new arbitration algorithm is proposed to implement the best arbitration strategy in Section 5. This paper concludes with Section 6.

2 Communication in HARTS

2.1 Internal structure of HARTS node

Each HARTS node consists of multiple application processors(APs) and single dedicated network processor(NP) in conjuction with custom network adapter(SPIDER), and they are connected through the VMEbus. Application programs are executed on the APs and the NP processes all the high-level communication functions that includes error recovery, message scheduling, and packetization/depacketization. The SPIDER provides lower-level functions including error detection, media access, routing, and switching[11]. At the center of SPIDER is the PRC, custom ASIC, that implements microcode-controlled routing and switching for four physical links with three virtual channels per link[12]. The PRC interacts with the NP in terms of pages, the basic unit of communication in HARTS. The NP transmits a packet by feeding page tags to a transmitter fetch unit(TFU) in the PRC; each page tag includes a memory address and the number of words to transmit. Similarly, the NP supplies each network interface receiver with pointers to free pages in the buffer memory, for arriving packets. PRC has network interface units, RX and TX. Each outbound channel is dedicated to a particular link, and is controlled by a PRC TX, while each inbound channel receives packets from a single link and is controlled by a PRC RX. The PRC supports flexible routing and switching by treating these outbound virtual channels as individually reservable resources; the TFUs and RXs reserves TXs and coordinate word-level data transfer using the cut-through bus(CT bus).

Since the communication devices, NP and SPIDER, are located on the VMEbus, any AP can interact with it. Thus, NP ensures that the channels are serviced in a certain global order as determined by their traffic parameters. Once a real-time channel is successfully established, data transfer occurs only from the source to sink without retransmissions and acknowledgements because real-time channels are unidirectional. On transmission these pages are filled with outgoing data from the source AP via DMA before the corresponding packet is scheduled for transmission. Packet headers, which are placed on separate 256-byte pages, can be examined independently by the NP since it manages all SPIDER pages and has full access to SPIDER's memory. The ability to examine packet headers facilitates reception path optimizations by allowing the NP to make intelligent decisions about the data before the data actually consumes transfer bandwidth within the node. Since SPIDER integrates all the links at the node on a single board, intermediate node traffic can be received and buffered in on-board pages, while the NP examines the headers and schedules the packets for later transmission.

2.2 Scheduling of Common Resources

The actual communication latency depends on the scheduling algorithm of the contended resources. There are two contended resources: the network link and VMEbus. The network link is arbitrated by the NP, because the NP controls the all function of the

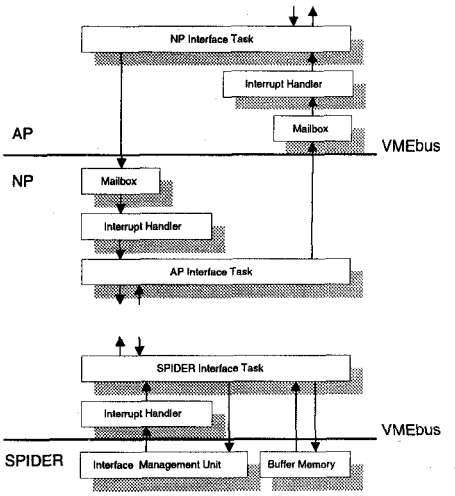

Figure 3: Software model of communication

SPIDER. The NP can break the transmission and allocate the network line for the other message by the scheduling algorithm. Within SPIDER, the worst-case latency of medium access per virtual channel is bounded independent of the number of contending virtual channel. To reduce the medium access latency further and make it more predictable, FIFO queuing inside SPIDER is limited to a depth of six pages for each virtual channel.

From the view of an application programmer, the message delivery sequence inside a HARTS node can be modeled as in Fig. 3, From the Fig. 3, the total latency to transmit a message from AP to SPIDER can be thought of the sum of the software execution time(in AP and NP), and the data transfer time over VMEbus. Since the software execution time in AP or NP mainly depends on the operating system or other software issues, it could be assumed the predetermined delay independent of the network condition. However, since the data transfer time over VMEbus depends on the arbitration policy of VMEbus, that is usually controlled by the priority of the message, the priority of each message should be well adjusted by the programmer. In other words, to guarantee the real-time bound of each message over VMEbus, the system should provide the predictable latency bound of each message to the given priority.

In message transfer between APs and NP, or between NP and SPIDER over VMEbus, the system controller arbitrates the bus mastership. When the system controller detects a bus request signal from a requester, it grants the bus mastership to the requester unless the bus is being used by another master module or several master modules request the bus mastership at the same time. The system controller should select one requester based on its pre-defined arbitration strategy. The VMEbus standard defines three types of arbitration: priority-based, round- robin selection, and single-level. In addition to these standard arbitration strategies, the IEEE standard permits user-defined bus arbitration strategy[13, 14]. In a VMEbus-based system, because there is no separate backplane bus for each prioritized data, the timing constraints should be satisfied by scheduling the usage of the backplane bus, which is determined by bus arbitration.

With considering the nature of VMEbus mentioned above and HARTS architecture, following conditions are assumed to analyze the latency over VMEbus: (1) data transfer is assumed to be burst operation, because the communication latency in HARTS is only meaningful by the unit of message, (2) each module has enough memory to receive the data transferred burst, (3) there is no overload such as context switching in bus preemption because the preempted DMA stalls the transfer during the use of VMEbus by other DMA, (4) VMEbus mastership is arbitrated word by word.

3 Communication Latency of Periodic Message

3.1 Local bus based model

The end-to-end communication in HARTS can be divided into three stages, AP to NP via VMEbus, NP to NP via network through switching at intermediate nodes, NP to AP via VMEbus. The data transfer time over VMEbus, t_v, can be defined as the time from the instant that a bus master, one of the communication interface tasks in APs or NP, initiates a data transfer cycle to the instant that all data are loaded on the destination memory. If VMEbus is not used by any other master, t_v is defined as

$$t_v = t_{aq} + t_{dt}. \quad (1)$$

t_{aq} is the bus acquisition time, the time required for a requester to acquire the right to use VMEbus and t_{dt} is the data transfer time, which is proportional to the size of the transferred data[15]. Compared with t_{dt}, the t_{aq} is small enough to be ignored in the case of burst data transfer.

Suppose the period of the generated message M_n, which has the n-th priority, is T_n and the size of M_n is S_n. If the VMEbus bandwidth is B_{vme} $word/s$, the time taken to transmit M_n is $t_{aq} + \frac{S_n}{B_{vme}}$. But the communication can be preempted by the message

Table 1: Symbol expression

M_n	message of n-th priority
T_n	period of M_n
S_n	size of M_n
S_P	size of a page
B_{vme}	VMEbus bandwidth
B_{nt}	network transmitter bandwidth
R	a page processing time in NP3

with the higher-level priority. Because the preemption occurs only when a word transferred completely and the next word is to be transferred, the blocking time, the time blocked by lower priority data before preemption, can be thought of the time taken for a word to transfer, $\frac{1}{B_{vme}}$. The communication latency over VMEbus of M_n, $t_{v,n}$ in a HARTS node S is derived in Eq.(2) like as the task completion time in rate monotonic scheduling test algorithm[16].

$$\begin{aligned} L_n(0) &= 0 \\ L_n(k+1) &= t_{aq} + \frac{S_n}{B_{vme}} + \frac{1}{B_{vme}} \\ &\quad + \sum_{i<n, M_i \in N_s} \lceil \frac{L_n(k)}{T_i} \rceil \frac{S_i}{B_{vme}} \\ t_{v,n} &= L_n(\infty), \end{aligned} \tag{2}$$

where $\lceil \cdot \rceil$ gives the ceiling integer of $\cdot$, and N_s is the set of the messages which are generated in node S or aimed to node S.

NP processes the message into pages, moves a page to the buffer memory of SPIDER, and writes a page tag to SPIDER. Then TFU transfers the data to NI TX. Suppose the bandwidth of the memory interface of PRC is B_{em} *words/sec*, and the bandwidth of NI TX is B_{nt}. The data throughput of network is limited as $min(B_{em}, B_{nt})$, where $B_{em} > B_{nt}$ is assumed as to SPIDER. The maximum page size is bounded by S_P and the time taken for NP to make a page and transfer it to SPIDER is assumed to be R. The transfer time of the message n from NP to SPIDER is

$$R\lceil \frac{S_n}{S_P} \rceil.$$

The transfer of a page to the buffer of SPIDER and the transfer of a page to the NI TX are processed pipelined. The time taken for the message of size S_n to transfer to network, $t_{tx,n}$ is

$$t_{tx,n} = MAX(R, B_{nt}S_P) \cdot (\lceil \frac{S_n}{S_P} \rceil - 1) + R + B_{nt}S_P \tag{3}$$

The network line can be shared with other transferred messages. For two messages M_a, M_b generated from a single node, let the notation \$ be defined such that $M_a\$M_b = 1$ if they share a network line and $M_a\$M_b = 0$ otherwise. The period of the bypassed message through the same network line with priority i is $T_{B,i}$, the time of use is $t_{BP,i}$. The communication latency over network line, t_{tl} is derived in Eq.(4).

$$\begin{aligned} W_n(0) &= 0 \\ W_n(k+1) &= t_{tx,n} + B_{nt}S_P + \sum_j \lceil \frac{W_n(k)}{T_{B,i}} \rceil t_{BP,i} \\ &\quad + \sum_{i<n, M_i\$M_n=1} \lceil \frac{W_n(k)}{T_i} \rceil t_{tx,i} \\ t_{tl,n} &= W_n(\infty) \end{aligned} \tag{4}$$

where $B_{nt}S_P$ is a blocking time in Eq.(4), since the use of the network line can be switched when a page is transferred completely. Eq.(5) shows the time taken for a message to move from AP to the network, $t_{vtx,n}$.

$$\begin{aligned} V_n(0) &= 0 \\ V_n(k+1) &= t_{aq} + \frac{S_n}{B_{vme}} + t_{tx,n} + \frac{1}{B_{vme}} + B_{nt}S_P \\ &\quad + \sum_{i<n, M_i\$M_n=1} \lceil \frac{V_n(k)}{T_i} \rceil (t_{aq} + \frac{S_i}{B_{vme}} + t_{tx,i}) \\ &\quad + \sum_{i<n, M_i\$M_n=0} \lceil \frac{V_n(k)}{T_i} \rceil (t_{aq} + \frac{S_i}{B_{vme}}) \\ &\quad + \sum_j \lceil \frac{V_n(k)}{T_{B,j}} \rceil t_{BP,j} \\ t_{vtx,n} &= V_n(\infty) \end{aligned} \tag{5}$$

Since a low priority message is not preempted twice at VMEbus and at the network line by the same message, $t_{vtx,n}$ is not the sum of $t_{v,n}$ and $t_{tl,n}$. The Eq.(5) assumes there is no contention in NP processing. The condition of the assumption is $R \geq B_{vme}S_P$.

Suppose the message passes through u intermediate nodes to the destination node. At each intermediate node, packet switching is occurred by the unit of page. The passing time at an intermediate node is the time taken for a message to be transferred from the buffer memory to PRC TX. The loading time to the buffer memory from network is included in the passing time at the previous node. Message $M_{q,j}$ is assumed to share the same network line at the q-th intermediate node with priority j. The period of $M_{q,j}$ is $T_{B,q,j}$ and the size is $S_{B,q,j}$. The passing time at each intermediate node is shown in Eq.(6).

$$U_n(0) = 0$$

$$
\begin{aligned}
U_n(k+1) &= S_n B_{nt} + S_P B_{nt} + \sum_{j<n} \lceil \frac{U_n(k)}{T_{B,q,j}} \rceil S_{B,q,j} B_{nt} \\
t_{ps,q,n} &= U_n(\infty) \\
t_{ps,n} &= \sum_{q \in (intermediate\ nodes)} t_{ps,q,n}
\end{aligned} \tag{6}
$$

At the destination node, the message is loaded on the buffer memory and most of them are transferred to the NP during the passing time at the previous node. The transfer time from NI RX to AP, t_{rx} is defined as the time taken for NP to get one remaining page and to concatenate all pages. Let the notation # be defined for received messages from NI RX, M_c, M_d such that $M_c \# M_d = 1$ if both M_c and M_d destine the current node, and $M_c \# M_d = 0$ otherwise. The latency of the message M_n fed into the RX to move to the AP in the destination node D, $t_{vrx,n}$, is derived in Eq.(7).

$$
\begin{aligned}
V_n(0) &= 0 \\
V_n(k+1) &= t_{aq} + \frac{S_n}{B_{vme}} + t_{rx,n} + \frac{1}{B_{vme}} \\
&\quad + \sum_{i<n, M_i \# M_n = 1} \lceil \frac{V_n(k)}{T_i} \rceil (t_{aq} + \frac{S_i}{B_{vme}} + t_{rx,i}) \\
&\quad + \sum_{i<n, i \in N_d} \lceil \frac{V_n(k)}{T_i} \rceil (t_{aq} + \frac{S_i}{B_{vme}}) \\
t_{vrx,n} &= V_n(\infty)
\end{aligned} \tag{7}
$$

The total communication latency from a source node to the destination node, t_l is achieved as Eq.(8)

$$
t_{l,n} = t_{vtx,n} + t_{ps,n} + t_{vrx,n} \tag{8}
$$

3.2 VMEbus based model

In the VMEbus system, each module has a constraint in the size. NP and SPIDER are located on VMEbus independently, for the convenience of implementing the HARTS node. In this configuration, a message uses VMEbus twice in communication between AP and the network driver. The communication latency between AP and NP, $t_{v,n}$, is changed as Eq.(9).

$$
\begin{aligned}
L_n(0) &= 0 \\
L_n(k+1) &= t_{aq} + \frac{S_n}{B_{vme}} + \frac{1}{B_{vme}} + 2 \sum_{i<n} \lceil \frac{L_n(k)}{T_i} \rceil \frac{S_i}{B_{vme}} \\
t_{v,n} &= L_n(\infty)
\end{aligned} \tag{9}
$$

In NP, after the message are made into pages, they are transferred to SPIDER. The time taken for NP to make message into pages is αS_n, where α is assume to be constant. The message transfer time from NP to the SPIDER buffer is same as t_v, and the message transfer time from SPIDER buffer to NI TX, t_{stx} is achieved as Eq.(10).

$$
\begin{aligned}
W_n(0) &= 0 \\
W_n(k+1) &= B_{nt} S_n + \sum_j \lceil \frac{W_n(k)}{T_{B,i}} \rceil t_{BP,i} \\
t_{stx,n} &= W_n(\infty)
\end{aligned} \tag{10}
$$

The page data is transferred from NP to SPIDER by DMA, and then NP commands the PRC by writing the page tag, while SPIDER transmits the previous page data to the NI TX. The data transfer from NP to SPIDER and from SPIDER to NI TX are pipelined. Thus, the total transfer time, $t_{sl,n}$ is derived in Eq.(11).

$$
t_{sl,n} = 2(2t_{v,n} + QS_n - \frac{1}{B_{vme}}(\lceil \frac{S_n}{S_P} \rceil - 1)) + t_{stx,n} + t_{ps,n} \tag{11}
$$

where $t_{ps,n}$ is same as the one in Eq.(6) and $\frac{1}{B_{vme}}(\lceil \frac{S_n}{S_P} \rceil - 1)$ is the overlapped time during the pipelined processing.

Eq.(8) and Eq.(11) assume that a low-priority message be preempted instantly by a high-priority message both in VMEbus and in the network line. While the multiple levels of priority can be implemented in network line arbitration, the priority level of VMEbus is limited by IEEE standard. Due to this limitation, the messages with different priority level should share the same priority level of VMEbus. The latency evaluated analytically Eq.(8) and Eq.(11) cannot be guaranteed. Roughly speaking, the worst-case latency of the high priority message is prolonged by the sum of the data transfer time of lower priority message which is represented as the same priority level.

4 Communication Latency of Sporadic Message

If the sporadic message is generated in APs or other network nodes, the worst-case communication latency cannot be estimated analytically. As a special, if the minimum bound of the interarrival time and the maximum bound of the size are given, the sporadic message can be regarded as the periodic message and the worst-case communication latency can be achieved as like Eq.(8) or Eq.(11). However, these parameters may not be given or the worst-case assumption may be pessimistic not to be scheduled by the fixed priority scheduling method. If the message has a soft real-time

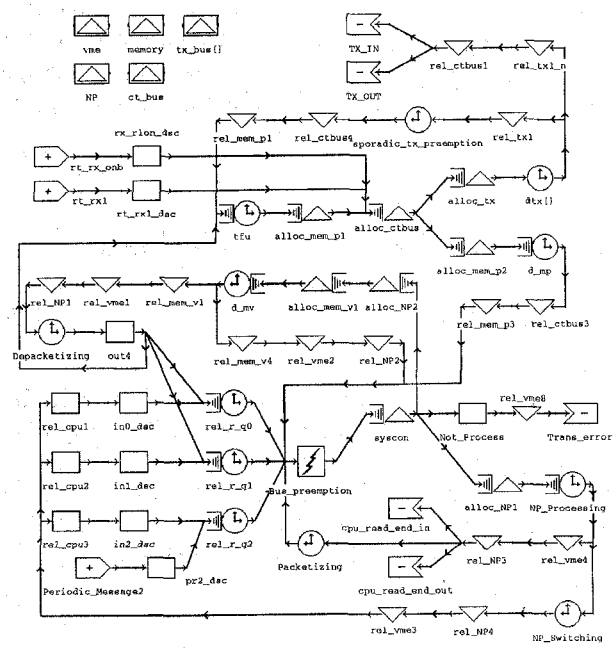

Figure 4: The *SES* model of a HARTS node

constraint, EDF scheduling policy can be used. To use EDF for arbitration of VMEbus, the local deadline, the deadline for local resources, rather than end-to-end deadline should be used. The local deadline can be computed as the time difference between end-to-end deadline and the estimated time from the next stage to final destination. To simplify the computation, it can be assumed that message is not blocked or preempted by other sporadic message. To implement EDF scheduling on the VMEbus arbitration, local deadline for VMEbus should be mapped onto the priority level of VMEbus request line. However, the 4 priority levels of VMEbus is not enough to schedule sporadic message based on the EDF scheduling because the local deadline is represented in real value. Moreover, if periodic messages are mixed with sporadic messages, some priority levels should be reserved for the periodic messages. Hence, it is more difficult to schedule the sporadic messages by EDF policy.

By computer simulation, the effect of the VMEbus limitation can be evaluated. The HARTS node is modeled by *SES/design 3.0* as shown in Fig.4. In the model, one AP generates a message randomly and it is transferred to PRC NI TXs through NP processing. The random-generated messages, destined to the other APs, are fed into the PRC NI RXs. The messages have a regular deadline and interarrival time, but the data size has normal distribution bounded by a certain maximum value. It is a feasible model in real

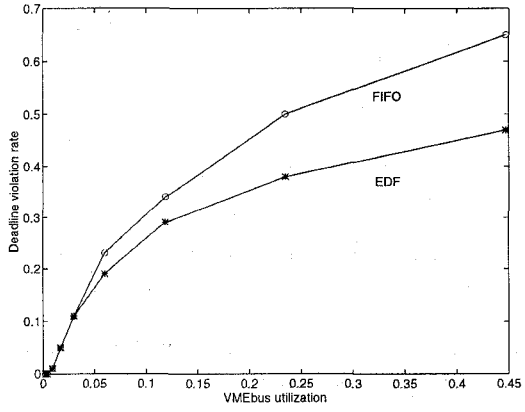

Figure 5: Deadline violation rate in VMEbus(1)

applications such as a video conference system. The data frame is generated periodically, the size of the data is irregular since the data are compressed with a irregular compression rate. We simulate two VMEbus arbitration strategies. First one is a round-robin used in HARTS currently. Second one is the EDF, which is ideal but cannot be implemented in standard VMEbus system. If the utilization of VMEbus is low, the performances are similar. As the utilization of VMEbus becomes high, the deadline-violation rate is higher under the round-robin policy than the EDF policy. The comparison is shown in Fig.5.

5 Proposed Arbitration Algorithm

To implement EDF policy without any additional bus arbitration time, the VMEbus should support enough priority levels that the local deadline can be expressed with bus-request priority level. It is difficult that new signals are allocated to the standard VMEbus. We propose a developed arbitration algorithm that the bus arbitration is operated by the interrupter and the interrupt handler as well as the bus arbiter. Generally, a bus arbiter in the system controller is a fixed state machine, which generates bus grant signal from the bus request signal according to a predefined arbitration scheme. Since the arbiter cannot manage the bus mastership dynamically, the bus requester controls the request by itself in the proposed algorithm.

When a message is generated in any bus master, the master interrupts the system controller. The system controller keeps up the global message deadline table, which the interrupt handler updates by scanning the id. and the deadline of the generated message in the interrupter module. The system controller broadcasts the table to all masters and let

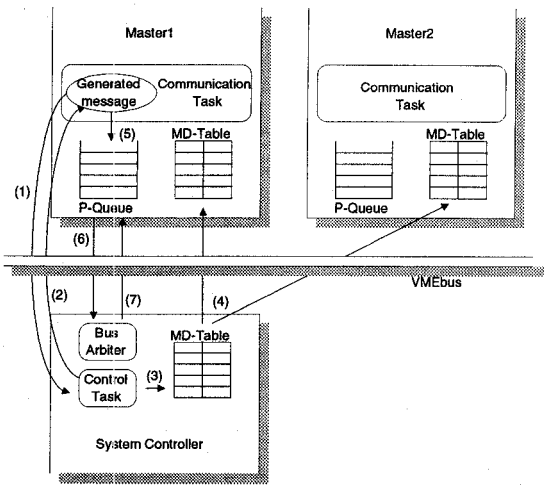

Figure 6: Flow of the proposed arbitration

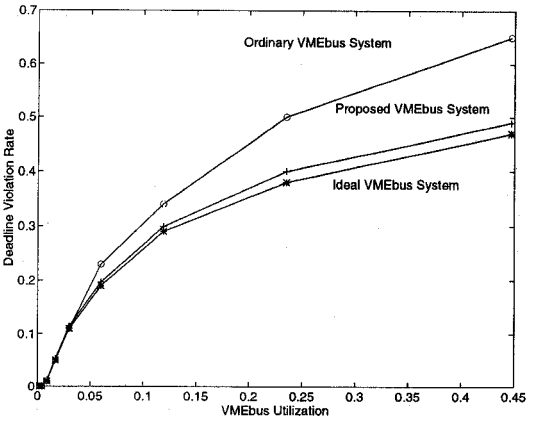

Figure 7: Deadline violation rate in VMEbus(2)

the generated message be inserted to the message queue, in which the earlist-deadline message goes out first. Only when the message out of the queue is same as the earlist-deadline message in the global table, the master request the bus mastership. The bus operation of the system controller has the highest priority in use of bus, and the other bus masters request the bus mastership with the same bus request signal of low priority. The transmitted message is preempted by the interrupt handler and it is relocated in message queue during the interrupt handling.

The proposed bus arbitration algorithm(Fig.6)

(1) *Communication task* informs *control task* of the message generation by interrupt.

(2) *Control task* fetches the id. and the deadline of the generated message in handling the interrupt.

(3) *Control task* updates the *MD(Message Deadline)-Table.*

(4) System controller broadcasts the *MD-Table.*

(5) The generated message inserted into the *P(Priority)-Queue.*

(6) If the earlist deadline message in *P-Queue* is globally earlist in *MD-Table*, *P-Queue* requests the bus mastership to *bus arbiter* in system controller.

(7) *Bus arbiter* grant the bus mastership to the requester.

◇ System controller acquires the bus with the highest priority during the operation (2), (3).

This algorithm gives the overhead, the interrupt handling time, to the communication. The interrupt handling time, t_{ih}, is derived as

$$t_{ih} = t_{int-pre} + \frac{2}{B_{vme}} + \beta M + \frac{M}{B_{vme}} + \beta Q + t_{int-post}, \quad (12)$$

where M is a maximum *MD-Table* size, Q is a maximum *P-Queue* size, β is a CPU processing rate for searching and sorting, $t_{int-pre}$ is a context saving time, and $t_{int-post}$ is a context restoring time. In the case that the HARTS adopts the proposed algorithm, the end-to-end communication latency derived in Eq.(8) and Eq.(11) grows longer as Eq.(13) and Eq.(14),

$$t'_{l,n} = t_{l,n} + t_{ih}\Big(\sum_{i,M_i \in N_s} \lceil \frac{t_{v,n}}{T_i} \rceil + \sum_{j,M_j \in N_d} \lceil \frac{t_{v,n}}{T_j} \rceil\Big), \quad (13)$$

$$t'_{sl,n} = t_{sl,n} + 2t_{ih}\Big(\sum_{i,M_i \in N_s} \lceil \frac{t_{v,n}}{T_i} \rceil + \sum_{j,M_j \in N_d} \lceil \frac{t_{v,n}}{T_j} \rceil\Big), \quad (14)$$

where $t_{v,n}$ is defined in Eq.(2). When the size of message is so large that the interrupt handling time is smaller than the transmission time of the message, the real-time performance is raised by priority-based scheduling though the interrupt handling time is added to the communication latency. In scheduling the sporadic real-time messages by the proposed algorithm, the performance is showed to be near to the performance of the ideal EDF scheduling(Fig.7).

6 Conclusion

In this paper, the communication latency is evaluated in HARTS for a real-time application. The worst-case communication latency is derived analytically for the periodic hard real-time messages, based on the real-implemented system with considering hardware-level parameters. Computer simulations are used to evaluate the communication latency for the sporadic soft real-time messages, and the new priority-based bus arbitration algorithm is proposed to bound the

deadline efficiently. The evaluation results in this paper can be used for real-time scheduling in HARTS and other distributed systems.

References

[1] J. Rexford, J. Dolter, W. Feng, and K. Shin, "PP-MESH-SIM: A simulator for evaluating multicomputer interconnection networks," *Proc. Annal Simulation Symposium*, April 1995, pp. 84-93.

[2] R. S. Raji, "Smart networks for control," *IEEE Spectrum*, vol. 31, pp 49-55, June 1994.

[3] C. M. Aras, J. F. Kurose, D. S. Reeves, and H. Schulzrinne, "Real-time communication in packet-switched networks," *Proc. of IEEE*, vol. 82, pp. 122-139, January 1994.

[4] D. D. Kandlur, K. G. Shin, and D. Ferrari, "Real-time communication in multi-hop networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, pp. 1044-1056, October 1994.

[5] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Computers*, vol. C-36, no. 5, pp.547-553, May 1987.

[6] L. Ni and P. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, pp. 62-76, February 1993.

[7] K. G. Shin, "HARTS: A distributed real-time architecture," *IEEE Computer*, pp. 25-35, May 1991.

[8] M. S. Chen, K. G. Shin, and D. D. Kandlur, "Addressing, routing, and broadcasting in hexagonal mesh multiprocessors," *IEEE Trans. on Computers*, pp.10-18, January 1990.

[9] J. W. Dolter, P. Ramanathan, and K. G. Shin, "Performance analysis of virtual cut-through switching in HARTS: A hexagonal mesh multicomputer," *IEEE Trans. on Computers*, pp. 669-680, June 1991.

[10] J. Rexford, J. Hall, and K. Shin, "A router architecture for real-time point-to-point networks," *Proc. International Symposium on Computer Architecture*, pp. 237-246, May 1996.

[11] J. Dolter, S. Daniel, A. Mehra, J. Rexford, W. Feng, and K. Shin, "SPIDER: Flexible and efficient communication support for point-to-point distributed systems," *Proc. Inter. Conference on Distributed Computing Systems*, pp. 574-580, June 1994.

[12] S. Daniel, J.Rexford, J. Dolter, and K. Shin, "A programmable routing controller for flexible communication in point-to-point networks," *Proc. International Conference on Computer Design*, pp.320-325, October 1995.

[13] Peterson, and Wade D, *The VMEbus Handbook, 3rd ed..*, VMEbus International Trade Association. Scottsdale, AZ, 1993.

[14] VMEbus International Trade Association, *The VMEbus specification*, VMEbus International Trade Association. Scottsdale, AZ, 1987.

[15] J. Park, and K. Shin, "Evaluation of communication latency in VMEbus-based real-time control systems," *Proc. IFAC Workshop on AARTC*, May 1995, pp.

[16] J. Lehoczky, L.Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," *Proc. IEEE Real-Time Systems Symposium*, pp. 166-171, December 1989.

[17] SES, *SES/workbench Creating Models*, Release 3.0, SES 4301 Westbank Drive, Bldg. A Austin, TX 78746, 1995.

[18] SES, *SES/workbench Sim Language Reference*, Release 3.0, SES 4301 Westbank Drive, Bldg. A Austin, TX 78746, 1995.