A. Garcia, L. Johansson, M. Jonsson, and M. Weckstén, "Guaranteed periodic real-time communication over wormhole switched networks," Proc. ISCA 13th International Conference on Parallel and Distributed Computing Systems (PDCS-00), Las Vegas, NV, USA, Aug. 8-10, 2000, pp. 632-639.

Guaranteed Periodic Real-Time Communication over Wormhole Switched Networks

Alejandro Garcia, Lisbeth Johansson, Magnus Jonsson, and Mattias Weckstén

School of Information Science, Computer and Electrical Engineering,
Halmstad University, Halmstad, Sweden

Magnus.Jonsson@ide.hh.se, http://www.hh.se/ide

Abstract

In this paper, we investigate how to efficiently implement TDMA (Time Division Multiple Access) on a wormhole switched network using a pure software solution in the end nodes. Transmission is conflict free on the time-slot level and hence deadlock free. On the sub-slot level, however, conflicts are possible when using "early sending," a method we propose in order to reduce latency while still not hazarding the TDMA schedule. We propose a complete system to offer services for dynamic establishment of guaranteed periodic real-time virtual channels. Two different clock synchronization approaches for integration into the TDMA system are discussed. Implementation and experimental studies have been done on a cluster of PCs connected by a Myrinet network. Also, a case study with a radar signal processing application is presented to show the usability. A best-case reduction of the latency of up to to 37 percent for 640 Byte messages by using early sending in Myrinet is shown in the case study. Source routed wormhole switching networks are assumed in the work but the results are applicable on some other categories of switched networks too.

1 Introduction

Switched high-performance networks are commonly used for local area networks and interconnection networks for parallel and distributed computing systems of today and tomorrow. Examples include clusters of workstations or PCs running multimedia applications, and parallel computers for radar signal processing applications. A number of networks with a competitive price/performance ratio have appeared on the market, e.g., Myrinet [1] and Gigabit Ethernet [2]. However, these networks typically have no or very little support for real-time traffic, especially hard real-time traffic which is required in applications like those mentioned above. Networks like ATM are available but less complex affordable alternatives are needed where each node can be connected directly to the switched network.

In this paper, we present work done on time-deterministic communication to support cyclic traffic in a class of switched networks. By using TDMA (Time Division Multiple Access), the access to each link in the network is divided into time-slots. When the traffic is changed (e.g., a new real-time virtual channel, RTVC, between two nodes is requested), the mapping of traffic onto links and time-slots is rescheduled in a distributed manner. Since clock synchronization messages are scheduled onto the same network and no scheduling is done in the switches (only in the end nodes), the real-time support can be implemented purely in software. Also worth mentioning is that the network becomes totally deadlock free since the whole path between source and destination is reserved in the same time-slot.

We assume wormhole switched networks in the paper but the concept holds for cut through and store-and-forward switching too. However, the overhead can become rather high in store-and-forward networks due to high latency compared to the effective sending time. This latency must be encountered before a new time-slot and the sending of a new message can begin. Moreover, source routing or another deterministic routing method is assumed. In this way, it is possible to reserve the corresponding links of a path between source and destination. Since switched systems allow for concurrent transmissions, multiple such paths can

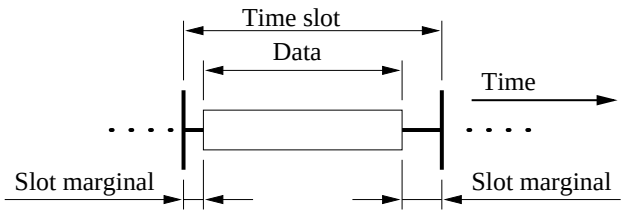

Figure 1: The TDMA slot.

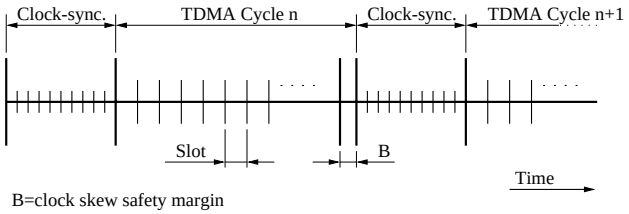

Figure 2: TDMA cycle when the clock synchronization is separated from the rest of the data traffic.

be reserved in the same time-slot.

We propose a method called early sending which can be used in, e.g., wormhole networks. By this method, a node with a scheduled slot S_{i+1} is allowed to initiate sending already in slot S_i. Expression for the exact time in slot S_i where it is allowed to initiate sending is given in Section 2.3. In a case study based on a radar signal processing application on a system with Myrinet, we show that the latency in the best-case can be reduced by up to 37 percent by using early sending. By doubling the message size from 640 Byte to 1280 Byte, the best-case improvement is 90 percent. For the early sending method, we assume some form of low-level flow control as used in wormhole networks.

Some work has been done in the field of switched networks with support for hard real-time traffic. Examples of such work are discussed below. RACEway is a switched network primarily developed for embedded systems [3] [4]. It has support for real-time traffic by the use of priorities but dynamic establishment of RTVCs with guaranteed performance is not supported. A similar system as the one discussed in this paper, but on a circuit-switched HIPPI network, is presented in [5]. In this paper however, we focus on more fine grained TDMA schedules and investigate how, e.g., clock synchronization accuracy influence on performance and other parameters.

There are a lot of work reported on how to support real-time traffic by modifying the hardware and/or software in the switches (see, e.g., [6] [7] [8] [9]). In contrast, in our work we have assumed no changes to either software or hardware in the switches. Instead, it is a pure software solution which only affects the end nodes. Instead of reserving access to the network, as in our case, one approach to get real-time services over a standard switched network is to calculate the worst-case latency. However, the worst-case throughput can be very low when a high worst-case latency separates each guaranteed access to the network [10] [11].

The rest of the paper is organized as follows. TDMA, clock synchronization, and early sending are presented and discussed in Section 2. In Section 3, our Myrinet implementation is described, and a case study is presented in Section 4. The paper is then concluded in Section 5.

2 Time deterministic communication concept

To pass messages with hard real-time constrains over a generic switched network, a method is needed to guarantee bandwidth. In order to allow transmission of multiple data-streams over a shared media, it is possible to use time domain multiplexing combined with reservation of every single network link in the system. This works only if all nodes has an unified apprehension of the time. In the following sections we will discuss the support for periodic traffic with hard real-time constraints. Further information related to Section 2 and 3 can be found in [12].

2.1 TDMA and clock synchronization

If the nodes in the network have large clock drifts the margins in the slots (Figure 1) need to be large in order to prevent blockages, but large slot margins gives a low network utilization. The alternative is a more frequent clock synchronization to keep down the clock drift. The margins can be reduced or totally removed if the switches are able to handle blocking situations without removing any message from the network. A message that starts its transmission a short time (relatively to the slot length) before it is allowed to, will be held up if the needed links are occupied with packets belonging to the previous TDMA slot (see Section 2.3).

Two different approaches for the creation of the TDMA cycle have been considered in this work. The first approach have the clock synchronization part separated from the rest of the TDMA cycle. However, this leads to a minimum length of the TDMA cycle (Figure 2) in order for the clock synchronization to reappear

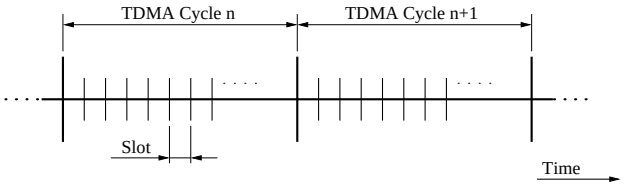

Figure 3: TDMA cycle when the clock synchronization is scheduled among with other data packets.

at certain intervals. As the clock synchronization period occupies a continuous period of time, when no other traffic is allowed, the minimum time period for data packets will be affected. The time period has to be larger than the total duration of clock synchronization traffic.

The other approach considered, schedules the clock synchronization messages among with all other real-time messages in the network (i.e., logical channels are established for clock synchronization in the same way as for normal data). This method allows concurrent transfers in the rest of the network (see Figure 3).

Regarding the second approach, problems occur when a master node have many clock synchronization messages to send. Assume a slot length of 30 μs and with a clock synchronization message of 5 μs (enough in Myrinet), the utilized part of every slot used for clock synchronization is $\frac{1}{6}$ only. The two methods are exemplified in the next subsection.

2.2 Clock Synchronization Example

A common real-time traffic example, e.g., in telecommunication applications, have a period of 125 μs. Assuming a data size of 1300 bytes per transmission, the necessary slot size for this message size is approximately 12.5 μs according to

$$T_{setup} + T_{maxdrift} + \frac{M}{c} \qquad (1)$$

in Myrinet. The 12.5 μs is calculated using a measured setup time of 3 μs (T_{setup}) for a zero copy message, a maximal clock synchronization difference between two clocks in the network of 1 μs ($T_{maxdrift}$) (see Section 3), and the size of the message (M) in bytes divided by the transmission rate (c) for Myrinet in bytes per second resulting in $3\,\mu s + 1\,\mu s + \frac{1300}{160}\,\mu s = 12.125\,\mu s$.

Consider a network consisting of a 4×4 mesh of switches, where a group of end-nodes are connected to each of the 16 switches. Every group consist of 16 end nodes, one synchronization sub-master and 15 slaves (see Figure 4). If the network is only allowed to have

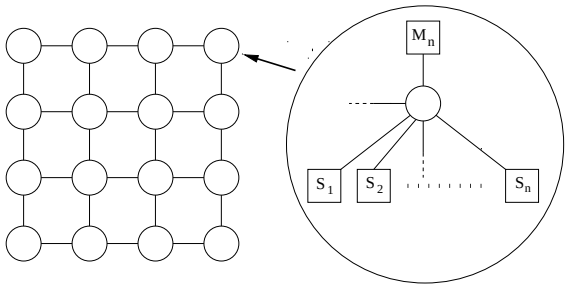

Figure 4: Clock synchronization in a 4×4 mesh.

a maximum clock drift of 1μs, a clock synchronization period of 5000 μs is needed as described in Section 3.

Previous in this section two different approaches for creating the TDMA-cycle where discussed. Using the first method (i.e., the method with the clock synchronization separated from the TDMA-cycle) the time to synchronize the whole network is calculated to be as follows. All sub-master nodes need $15 \cdot 5\,\mu s = 75\mu s$ (5 μs for each node) to synchronize to the master, while all the nodes in the network need $75\,\mu s + 75\,\mu s = 150\,\mu s$ (i.e., 75 μs is the time for all sub-clusters to synchronize their slaves, all clusters in parallel). With a margin of 12.5μs (one slot length), this gives a total time of $150\,\mu s + 12.5\,\mu s = 162.5\mu s$. This results in a need of $\frac{162.5\,\mu s}{5000\,\mu s} = 3\%$ of the total bandwidth for clock synchronization purposes.

Using the second method (i.e., with the clock synchronization packets scheduled among ordinary traffic) the total number of slots needed in order to synchronize the whole network are as follows. To synchronize all the sub-master nodes 15 slots are needed, plus 15 slots for each sub-master cluster (i.e., all sub-clusters synchronize their slaves in parallel) which gives a total of 30 slots ($15 + 15 = 30$). With a slot-length of 12.5 μs the total time is $30 \cdot 12.5\,\mu s = 375\,\mu s$. This results in a need of $\frac{375\,\mu s}{5000\,\mu s} = 7.5\%$ of the total bandwidth for clock synchronization purposes. However, ordinary traffic is allowed in this case, and shorter time periods and deadlines for the traffic are allowed. The second method is assumed in the rest of the paper.

2.3 Early Sending in TDMA

In networks that use a low-level flow control as in wormhole networks it is possible to utilize the network better by taking away all margins (see Figure 1). In this way, a message belonging to slot $n+1$ and for which sending is initiated already in slot n, can be halted in the network due to an already occupied path. The transmission will be resumed as soon as the path becomes free. The margins before and after the

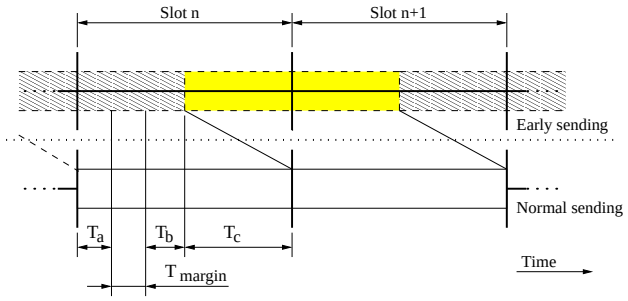

Figure 5: Early sending example.

message actually starts and ends are not needed in a such networks.

If the transmission of the message to be sent in time slot $n+1$ is initiated before the end of time slot n (Figure 5), spare time in time slot n can be used. However, the transmission of the message belonging to time slot $n+1$ need to be delayed until it is certain that all message heads belonging to time slot n has staked their way through all switches between source and destination (delayed time: T_a). The clock synchronization drift in the network must also be taken into account before allowing early sending initiation. Using a margin (T_{margin}) that is larger or equal to the maximal difference ($T_{maxdrift}$) between two clocks in the network will solve the problem. In other words, the early sending can be initiated when all trunks and ports used for the transmission in time slot n are reserved, i.e., the header has reached the last switch on the path. The early sending of a message belonging to slot $n+1$ is not allowed to be initiated until a delay of T_{early} has passed from slot-start of slot n, where $T_{early} = T_a + T_{margin} \geq T_a + T_{maxdrift}$.

In some cases, a switch will stop the message belonging to time slot $n+1$ because of elements in the network already being utilized by message belonging to the previous time slot (blocking time: T_b). As soon as messages belonging to time slot n complete their transmission, the resources will be released and the blocked message can start its transmission (early sending time: T_c). The latest time for the blocked message to start its transmission will be when time slot $n+1$ start, clock synchronization drift not encountered. The early sending method can not be used for clock synchronization messages as their total transmission time must be deterministic.

Another network where the early sending method together with TDMA is applicable is the circuit switched network HIPPI (High-Performance Parallel Interface) described in [5], if using the camp-on feature. With that feature, a request for connection establishment can be temporary halted in the same way as in wormhole networks.

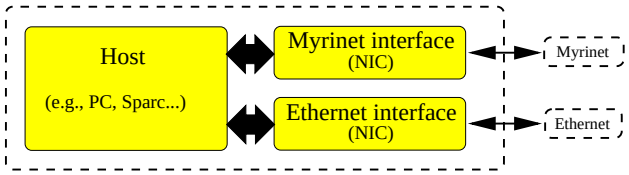

Figure 6: A system overview

3 Implementation on Myrinet

The work has been focused on supporting periodic traffic; however, it is possible to support aperiodic or isochronous traffic by using periodic reservations. The reservation of channels can be altered during runtime (i.e., dynamic allocation of logical channels with real-time support, RTVCs, is possible). In addition to Myrinet another network with broadcast functionality is assumed in our implementation. The implementation is not limited by any particular topology, since resource allocation in a TDMA schedule avoids deadlock. Figure 6, shows a system overview with the blocks of a single node which consists of a host computer (e.g., Pentium PC) with a Ethernet Network Interface Card (NIC), and a Myrinet NIC. Functions as time slotting, clock synchronization between nodes, and receive initialization will be handled by the onboard processor on the Myrinet NIC.

In the tests, we have measured the maximum drift between two nodes to less than 100 μs/s. Given an allowed drift of 500 ns the necessary synchronization period is less than 5 ms. The master node time is sampled with a possible error of 500 ns which gives a total maximum drift ($T_{maxdrift}$) of 1 μs.

3.1 TDMA Implementation and Performance

In the conversion from data stream into packages, control information has to be added. This control information (the overhead) consists of the routing information and the packet length (i.e., the packet header). Not only the control information cause overhead, the transmission time also increases because of DMA bottlenecks for each packet. Utilization is plotted against packet length for both the theoretical and measured case, i.e., excluding and including, respectively, DMA bottlenecks.To model the optimal utilization of a channel (η) using a certain message length (M) at a certain

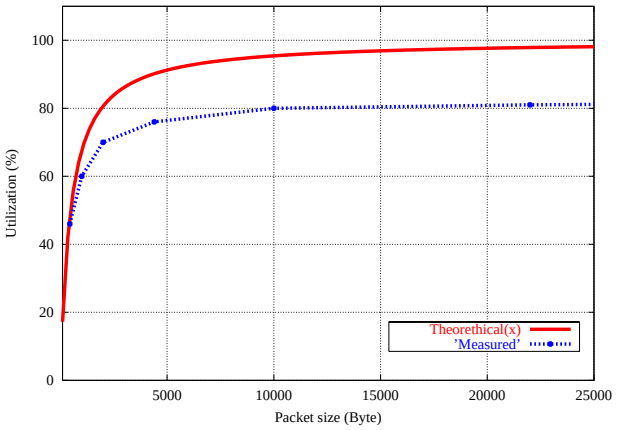

Figure 7: The channel utilization for different packet sizes. The theoretical curve describes the optimal performance while the measured curve describes the implemented solution.

transmission rate (c), the overhead time has been measured (τ) by transmitting a packet with zero bytes of data (see Figure 7). The optimal utilization is:

$$\eta = \frac{1}{\frac{\tau}{\frac{M}{c}} + 1} = \frac{1}{\frac{c\tau}{M} + 1} \quad (2)$$

3.2 The Scheduler and Connection Establishment

Some definitions used in this section are: S_i: Source host of message stream i. D_i: Destination host for message stream i. p_i: Period time of message stream i. d_i: Separate deadline (incremented by the period of the traffic stream) associated with message stream i, i.e., the latest time for the whole message stream i to reach D_i. s_i : Minimum number of time slots required for message stream i.

The traffic over Myrinet using TDMA has to be scheduled in some way. In order to release the burden of the LANai (RISC processor on the Myrinet NIC) the schedule is scheduled in the host. By using distributed scheduling all nodes runs the same algorithm with the same input, i.e., every node's altered bandwidth demand. By using Ethernet multicasting for this, the burden on the Myrinet network is reduced. The traffic over Myrinet will only consist of clock synchronization and data messages.

The routing information is statically added in advance for simplicity. However, it is possible to let a program determine the network topology with the help of the characteristic behavior of the switches and the nodes. The chosen TDMA approach (see Section 2.1) is the one where the clock is scheduled among with all other data packets. All traffic handled is periodic and a separate deadline is associated with each message stream. At the start of every new period the message has to be available for sending. Every message stream i is characterized by the following tuple: $\{S_i, D_i, p_i, d_i, s_i\}$. The shortest deadline for messages with the size of one slot is three slot lengths, while the shortest period is two slot lengths (e.g., if the slot-length is $30\,\mu s$ the shortest period is $2 \cdot 30\,\mu s$) since the clock synchronization messages requires at least one slot per node. The longest allowed period is as long as the TDMA cycle length (e.g., $40 \cdot 30\,\mu s$). Before being scheduled all bandwidth demands are sorted according to their individual deadlines. This is an established method for making a good schedule. However, the algorithm is not designed to deliver an optimized schedule (not focused on in this work). The created schedule runs repeatedly until a new schedule is available.

To generate a new schedule the scheduler first gather all bandwidth demands for periodic real-time traffic and then determines the schedule. The output information from the scheduler contains information about which slots that belongs to each task and the path that is allowed when transmitting the message.

4 Case study and experiments

Typical real-time applications with high throughput requirements and a pipelined dataflow between the computational modules include future radar signal processing systems [13] [14]. In Figure 8, a signal processing chain, similar to the one described in [15], is shown together with its bandwidth demands. As a feasibility study and example of applicability of our results, we will show how all RTVCs in the chain (the arrows) can be guaranteed over the network shown in Figure 9. We derive some important parameters of the system based on experiences with an implementation on a Myrinet-based cluster of PCs.

The indexes of the nodes in Figure 9 corresponds to those in the chain in Figure 8. The linear array of switches topology is chosen for the case study but can, e.g., be easily extended with additional switches to form a ring. The distribution module is assumed to have two network interfaces to reach the needed throughput demand. We denote the bidirectional links as L_i, $1 \leq i \leq 18$ and put an additional letter for the direction (N, E, S, and W) when referring to a one way part of a link.

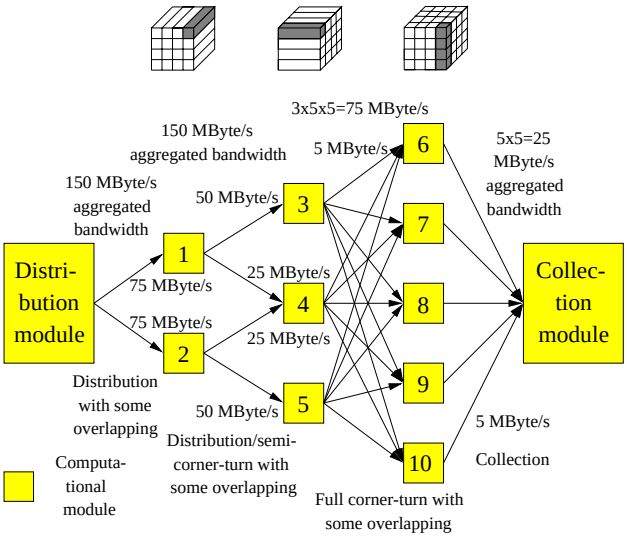

Figure 8: Data flow between the computational modules in the case study.

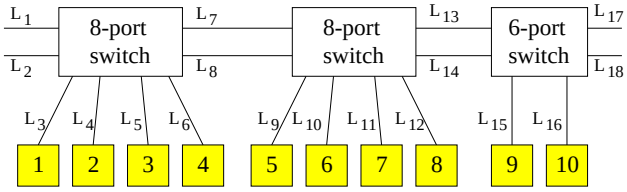

Figure 9: A linear array of switches connects the computational modules. The collection and distribution modules are not shown.

We assume 1.28 + 1.28 Gbit/s (160 + 160 MByte/s) full duplex Myrinet links and denote the link capacity as $R = 160$ MByte/s. Further on, we have measured a maximum bandwidth utilization of 80% (at continuous traffic) and a start-up overhead of up to $T_{setup} = 3\mu s$ for each message. With a slot length of $T_{slot} = 8$ μs, this corresponds to an efficient payload of $0.8R(T_{slot}$ - 3 μs) = 640 Byte/slot and a link bandwidth utilization of 50%:

$$0.8\frac{T_{slot} - 3\,\mu s}{T_{slot}} = 0.5 \qquad (3)$$

However, the slot length can be increased to get a higher bandwidth utilization.

A feasible schedule of the slots is shown in Table 1. Both data and clock synchronization messages are scheduled in those slots. There are 16 slots in a cycle, S_i, $1 \leq i \leq 16$, where one slot per cycle corresponds to a throughput of 5 MByte/s. Even though the example is hand made to get a clear example schedule, our scheduler also managed to schedule the traffic in 16 slots.

The worst-case latency is two slot (16 μs) for an RTVC with 15 slots per cycle, and one cycle (128 μs) for an RTVC with one slot per cycle. This is low enough for the radar system since the total communication latency through the four communication steps is allowed to be 10 ms. The average latency for the case of 15 slots per cycle is

$$T_{lat} = \frac{15 \cdot 0.5 + 1 \cdot 1.5}{16} T_{slot} + T_{setup} = 7.5\,\mu s \qquad (4)$$

where the two terms corresponds to the case during a slot where the next slot is owned (average latency of $0.5\,T_{slot}$), and the case during the slot before the slot which is not owned by the node (average latency of $1.5\,T_{slot}$). The average latency for the case of one slot per cycle is a half cycle (64 μs).

When using early sending, transmission can start during the previous slot after a delay of $T_{early} = T_a + T_{margin}$ according to Figure 5. We assume a maximum clock drift of $T_{margin} = 1\,\mu s$ and a worst-case latency through a switch of 600 ns including 10 meters of cable. With a maximum delay of $T_a = 3 + 0.6\,(N-1)\ \mu s$ before the last of N switches is reached, we get $T_{early} = 5.2\,\mu s$ if a path can traverse a maximum of three switches. The decreased latency is $T_{slot} - T_{early} = 2.8\,\mu s$ which, for the case of 15 slots per cycle, gives

$$\frac{T_{slot} - T_{early}}{T_{slot}} = 37\% \qquad (5)$$

in relative improvement, while the relative improvement for the case of one slot per cycle is 4.4%. These figures are best-case improvements, i.e., assuming the whole path is free when the transmission is initiated in the previous slot. If the previous slot is occupied, but only partly, the latency can still be improved by a lower amount. With a longer slot duration, the best-case improvements is even higher, e.g., 10.8 μs when $T_{slot} = 16\,\mu s$, which gives best-case relative improvements of 90% and 8.4% for the cases of 15 and one slot per cycle, respectively.

5 Conclusion

We have evaluated a software based method to implement hard real-time services over a class of generic switched networks. By using resource allocation combined with TDMA, blockages and consequently deadlocks are avoided in the network. We have discussed

Links	Time slots S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
L_{1E}	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	
L_{2E}	→2	→2	→2	→2	→2	→2	→2	→2	→2	→2	→2	→2	→2	→2		→2
L_{3S}	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	→1	4→▶1
L_{3N}	1→3	1→3	1→3	1→3	1→3	1→3	1→3	1→3	1→3	1→3	1→4	1→4	1→4	1→4	1→4	
L_{4S}	→2	→2	→2	→2	→2	→2	→2	→2	→2	→2	→2	→2	→2	→2	4→▶2	→2
L_{4N}	2→4	2→4	2→4	2→4	2→4	2→5	2→5	2→5	2→5	2→5	2→5	2→5	2→5	2→5	2→5	
L_{5S}	1→3	1→3	1→3	1→3	1→3	1→3	1→3	1→3	1→3	1→3				4→▶3		
L_{5N}	3→6	3→7	3→8	3→9	3→10											
L_{6S}	2→4	2→4	2→4	2→4	2→4					10→▶4	1→4	1→4	1→4	1→4	1→4	
L_{6N}						4→6	4→7	4→8	4→9	4→10				4→▶3	4→▶2	4→▶1
L_{7E}	3→6	3→7	3→8	3→9	3→10	4→6	4→7	4→8	4→9	4→10						
L_{8E}						2→5	2→5	2→5	2→5	2→5	2→5	2→5	2→5	2→5	2→5	
L_{8W}										10→▶4						
L_{9S}						2→5	2→5	2→5	2→5	2→5	2→5	2→5	2→5	2→5	2→5	8→▶5
L_{9N}											5→6	5→7	5→8	5→9	5→10	
L_{10S}	3→6					4→6					5→6				8→▶6	
L_{10N}	6→															
L_{11S}		3→7					4→7					5→7		8→▶7		
L_{11N}		7→														
L_{12S}			3→8					4→8				10→▶8	5→8			
L_{12N}			8→											8→▶7	8→▶6	8→▶5
L_{13E}	6→	7→	8→	3→9	3→10				4→9	4→10				5→9	5→10	
L_{13W}										10→▶4		10→▶8				
L_{15S}				3→9					4→9					5→9		10→▶9
L_{15N}				9→												
L_{16S}					3→10					4→10					5→10	
L_{16N}					10→					10→▶4		10→▶8				10→▶9
L_{17E}	6→	7→	8→	9→	10→											

Table 1: One cycle of 16 slots. Each entry in the table indicates the two nodes between which the link is part of the path in the slot. Unused directions of the bidirectional links are omitted in the table. Two different arrows are used:"→" for ordinary data messages and "▶" for clock synchronization messages.

clock synchronization aspects and shown how the latency can be significantly reduced by using the early sending method.

References

[1] N. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic, and Wen-King Su. MyriNET — a gigabit-per-second local-area network. *IEEE-Micro*, 15(1):29–36, February 1995.

[2] H. Frazier and H. Johnson. Gigabit ethernet: from 100 to 1,000 mbps. *IEEE Internet Computing*, 3(1):24–31, January 1999.

[3] B.C. Kuszmaul. The RACE network architechture. In *Proc. 9th Int. Parallel Processing Symposium (IPPS'95)*, pages 508–513, April 1995.

[4] T. Einstein. RACEway interlink — a real-time multicomputing interconnect fabric for high-performance VWEbus-systems. *VMEbus Systems*, Spring 1996.

[5] R. Bettati and A. Nica. Real-time networking over HIPPI. In *Proc. of the Fourth Workshop on Parallel and Distributed Real-Time Systems*, Santa Barbara, CA, USA, April 1995.

[6] B. Kim, J. Kim, S. Hong, and S. Lee. A real-time communication method for wormhole switching networks. In *Proc. of the Int. Conference on Parallel Processing*, 1998.

[7] H. Song, B. Kwon, and H. Yoon. Throttle and preempt: A new flow control for real-time communications in wormhole networks. In *Proc. of the 1997 Int. Conference on Parallel Processing (ICPP'97)*, pages 198–202, August 1997.

[8] J. Jonsson and J. Vasell. Implementation of a time-deterministic communication chip. Technical Report 206, CTH, Dept. of Computer Engineering, Computer Architecture Laboratory (CAL), MMP, 1995.

[9] J.-P. Li and M.W. Mutka. Priority based real-time communication for large scale wormhole networks. In *Proc. of the IEEE 8th Int. Parallel Processing Symposium (IPPS'94)*, pages 433–438, April 1994.

[10] K.H. Connelly and A.A. Chien. FM-QoS: Real-time communication using self-synchronizing schedules. In *High Performance Networking and Computing: Proc. of the 1997 ACM/IEEE SC97*, November 1997.

[11] S. Sundaresan and R. Bettati. Distributed connection management for real-time communication over wormhole-routed networks. In *Proc. of the 17th Int. Conference on Distributed Computing Systems*, pages 209–216, May 1997.

[12] A. Garcia, L. Johansson, and M. Weckstén. Real-time services in myrinet based clusters of PCs. Master's thesis, Halmstad University, January 2000. Research Report CCA - 0003.

[13] M. Jonsson, A. Åhlander, M. Taveniku, and B. Svensson. Time-deterministic WDM star network for massively parallel computing in radar systems. In *Proc. Massively Parallel Processing using Optical Interconnections (MPPOI'96)*, pages 85–93, October 1996.

[14] M. Taveniku, A. Ahlander, M. Jonsson, and B. Svensson. The VEGA moderately parallel MIMD, moderately parallel SIMD, architecture for high performance array signal processing. In *Proc. 12th Int. Parallel Processing Symposium & 9th Symposium on Parallel and Distributed Processing (IPPS/SPDP'98)*, pages 226–232, April 1998.

[15] M. Jonsson. Comments on interconnection networks for parallel radar signal processing systems. Technical report, Centre for Computer Systems Architecture (CCA), May 1999. Research Report CCA - 9911.