Proc. ISCA 12th International Conference on Parallel and Distributed Computing Systems (PDCS-99), Fort Lauderdale, FL, USA, Aug. 18-20, 1999, pp. 94-101.

Fiber-Ribbon Ring Network with Services for Parallel Processing and Distributed Real-Time Systems

Magnus Jonsson, Carl Bergenhem, and Jörgen Olsson

Centre for Computer Systems Architecture, Halmstad University, Box 823, 301 18 Halmstad, Sweden
{Magnus.Jonsson, Carl.Bergenhem, Jorgen.Olsson}@cca.hh.se, http://www.hh.se/dep/cca

Abstract

In this paper, we present how real-time services are implemented in a control-channel based ring network built up of fiber-ribbon point-to-point links. Services for best effort messages, guarantee seeking messages and real-time virtual channels are supported for single destination, multicast and broadcast transmission by the network. Slot-reservation is used for the implementation of real-time virtual channels. High aggregated throughput can be achieved due to pipelining, i.e., data can be transmitted simultaneously in different segments of the ring. An analysis of worst-case latency and deterministic throughput, which are important measures for real-time service implementation, is provided. The network is analyzed for two variants of time-slot organization, one that offers higher throughput and one that offers lower latency in some situations. We also show how the network offers low-level support for parallel computing, i.e., barrier-synchronization and global reduction. The control channel is used when realizing these functions, which implies no modification of the original network architecture. Low-level support for reliable transmission is also offered in a similar manner. This includes acknowledge / negative acknowledge and flow-control.

Keywords: Real-time communication, fiber-optic interconnection network, parallel processing, barrier synchronization, global reduction.

1 Introduction

Novel optical components result in the possibility of new network solutions for the increasing bandwidth demands of parallel and distributed systems. In this paper, a fiber-optic network with special features for both parallel processing in general and for distributed real-time systems is presented. It is a pipeline ring network based on optical fiber-ribbon point-to-point links. The network uses a MAC-protocol (Medium Access Control) called CC-FPR (Control-Channel based Fiber-ribbon Pipeline Ring) [1]. In a pipeline ring network, several packets can travel through the network simultaneously, thus achieving an aggregated throughput higher than the single-link capacity.

Motorola OPTOBUS™ bidirectional links with ten fibers per direction are used but the links are arranged in a uni-directional ring architecture where only $M/2$ bidirectional links are needed to close a ring of M nodes (assuming that M is an even number). Fiber-ribbon links offering an aggregated bandwidth of several Gb/s have already reached the market [2]. The increasingly good price/performance ratio for fiber-ribbon links indicates a great success potential for the proposed kind of networks.

As shown in Figure 1, the physical ring network is divided into two rings or channels, one data ring and one control ring. In each fiber-ribbon link, eight fibers carry data and one fiber is used to clock the data, byte for byte. Together, these fibers form a data channel in the ring that carries data-packets. The access is divided into slots like in an ordinary TDMA (Time Division Multiple Access) network. The tenth fiber is dedicated for bit-serial transmission of control-packets which are used for the arbitration of data transmission in each slot. The clock signal on the dedicated clock fiber, which is used to

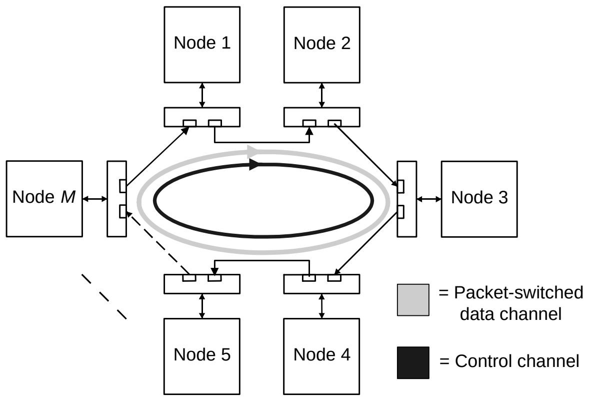

Figure 1: Each link in the the physical ring network is divided into a data channel and a control channel, forming two sub-networks.

clock data, also clocks each bit in the control-packets. Separating clock- and control-fibers simplifies the transceiver hardware implementation, which is verified by the current prototype development. The control-channel is also used for the implementation of low-level support for barrier-synchronization, global reduction, and reliable transmission.

The ring can dynamically (for each slot) be partitioned into segments to obtain a pipeline ring network where several transmissions can be performed simultaneously through spatial bandwidth reuse. Even simultaneous multicast transmissions are possible when the multicast segments do not overlap. Real-time services in the form of best effort messages, guarantee seeking messages and real-time virtual channels (RTVC) [3] are supported for single destination, multicast and broadcast transmission by the network.

Other pipeline ring networks are described in [4], [5] and [6] and more references are found in [4]. A fiber-ribbon ring network supporting pipelined circuit-switched traffic on nine fibers and packet-switched traffic on one fiber is described in [7]. Another fiber-ribbon ring network is the USC PONI (formerly called POLO) network, proposed to be used in multimedia applications [8]. The network described in [5] also relies on a separate control channel but needs a central control node that brings additional cost in hardware as well as additional latency when waiting for response from the central control node. The CC-FPR network is insensitive to propagation delay in the sense that no feedback is needed, from other nodes or from a central controller, between control-packet and data-packet transmissions.

The rest of the paper is organized as follows. The protocol is presented in Section 2. In Section 3, user services are described, and in Section 4, an analysis of the network is presented. The paper is then concluded in Section 5.

2 The CC-FPR protocol

Throughout Section 2, slot reserving, as described in the next section, is assumed to be unused. Before explaining the arbitration mechanism we will describe how data-packets travel on the ring.

The access to the network is cyclic and each cycle consists of M time-slots, where M is the number of nodes. Each node is denoted as m_i, $1 \le i \le M$. In each slot there is always one node responsible for initiating the traffic around the ring. This node is called the slot-initiator (SI). Each node is slot-initiator in one slot per cycle of slots. At the end of the slot, the role of being slot-initiator is asynchronously handed over to the next node downstream. This can be done implicitly by sensing the end of the slot.

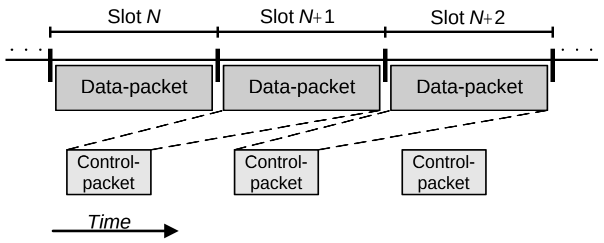

Figure 2: The control-packet is used for arbitration for the next slot.

The CC-FPR medium access protocol is based on the use of a control-packet that, for each slot, travels almost one lap round (over $M-1$ links) the control-channel ring. The node that will be slot-initiator in the next slot initiates the transmission of the control-packet as shown in the figure. We denoted this node as SI+1. In the time domain the control-packet always travels around the ring in the time-slot preceding the time-slot for which it controls the arbitration (see Figure 2). The control-packet will hence always pass each node one time-slot before the data-packet it is related to passes.

The contents of the control-packet is shown in Figure 3. The control-packet consist of a start-bit followed by an M bit long link-reservation field and an M bit long destination field, where M is the number of nodes. Each bit in the link-reservation field tells if the corresponding link is reserved for transmission in the next slot. In the same way each bit in the destination field tells if the corresponding node will have a data-packet destined to it in the next slot. Additional information for, e.g., flow-control is also included in the control-packet but is for clarity not shown in the figure.

Each node succeeding SI+1 checks the control-packet when it passes the node to see: (*i*) if it will receive a data-packet in the next slot, which is indicated by the node's bit in the destination field, and (*ii*) if a data-packet will pass the node in the next slot, which is indicated by the bit in the link-reservation field corresponding to the outgoing link of the node. If no data-packet will pass the node, i.e., the rest of the ring back to the slot-initiator is free, then the node will have the possibility to transmit a data-packet in the next slot in this part of the ring.

If a node has a packet for which transmission is possible, it prepares in advance new link-reservation and destination fields to reserve needed links and notify destination node(s). In this way the node can immediately change the control-packet when it passes if

	Link-reservation field				Destination field			
Bit 0	Bit 1	Bit 2	· · ·	Bit M	Bit M+1	Bit M+2	· · ·	Bit 2M
Start bit	Link 1	Link 2	· · ·	Link M	Node 1	Node 2	· · ·	Node M

Figure 3: A control-packet contains a start bit, a link-reservation field, and a destination field.

	Outgoing control-packet		
Node	Link	Dest.	Transmission allocated
1	1 1 0 0 0	0 0 1 0 0	To Node 3
2	1 1 0 0 0	0 0 1 0 0	Could not allocate
3	1 1 0 0 0	0 0 1 0 0	Could allocate transm. to Node 4, 5, and 1 but had nothing to send
4	0 0 0 1 1	1 0 0 0 1	Multicast to Node 5 & 1
5	0 0 0 1 1	1 0 0 0 1	Could not allocate

Figure 4: A control-packet travels around a network with five nodes. Node 1 is SI+1.

the bit in the link-reservation field, corresponding to the outgoing link of the node, was set to zero. Since there is no data-packet that will pass the node, succeeding nodes have no use of the overwritten information in the control-packet.

Because all the nodes succeeding the slot-initiator repeat the procedure of checking the control-packet for the possibility to send, multiple transmissions in different segments of the ring might be possible in the same slot. An example of how the control-packets travels around a five node network is shown in Figure 4. The arbitration will result in two concurrent data-packet transmissions in the next slot, one single-destination and one multicast packet as shown in Figure 5. Node m_1 is SI+1 in the example and therefore initiates the control-packet transmission described in Figure 4. It reserves Link 1 and Link 2 for transmission to node m_3 and informs node m_3 that it will have a data-packet destined to it in the next slot, by setting the corresponding bits in the link-reservation field and the destination field, respectively. Node m_2 and node m_3 do not change the control-packet but checks it to see if there will be any data-packets destined to them in the next slot. Node m_4 reserves Link 4 and Link 5 for a multicast transmission to node m_5 and node m_1. Node m_5 then receives the control-packet and removes it from the ring.

The reason why the control-packet only travel over the first $M - 1$ links after SI+1 is that the clock signal is interrupted by SI. The node that initiated the transmission of the control-packet, SI+1, will not return the packet. The consequence of this is that the node will

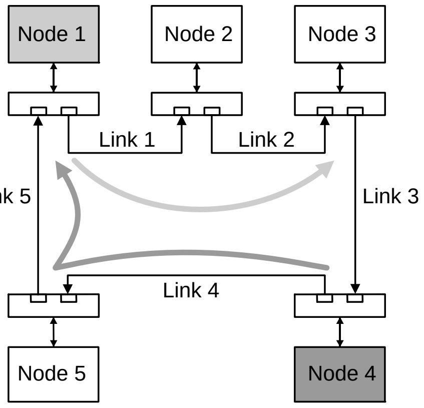

Figure 5: Example where Node 1 sends a single-destination packet to Node 3, and Node 4 sends a multicast packet to Node 5 and Node 1.

not be informed whether there will be a data-packet destined to it or not in the next slot. However, if a packet does arrive then it is also destined to the node.

Essential for the performance is that the delay that the control-packet experience in each node it bypasses is minimal, especially in large networks. One method is to organize the bits in the link-reservation field in the control-packet, for each slot, so that they appear in the same order as the control-packet travels. In other words, the first bit corresponds to the outgoing link from the slot-initiator. In this way a node that wants to change the contents of the control-packet does not have to store the whole packet before checking and possibly overwriting it. Instead it can retransmit the packet bit by bit and exchange the remaining part of the slot, if transmission was possible, after reading the bit in the link-reservation field corresponding to its outgoing link. The node's bit in the destination field in the incoming control-packet must, however, be checked before it is thrown away. Using this method, the delay in each node will only be one or a few bits.

As indicated in Figure 6, the bandwidth utilization depends on the ratio between the total propagation delay around the ring and the cycle length. This is an effect related to the asynchronous passing mechanism of the slot-initiator assignment. Further evaluation of this phenomena is provided in Section 4.

3 User services

The user services described below are: real-time virtual channels (Subsection 3.1), guarantee seeking messages (Subsection 3.2), best effort messages (Subsection 3.3), barrier synchronization (Subsection

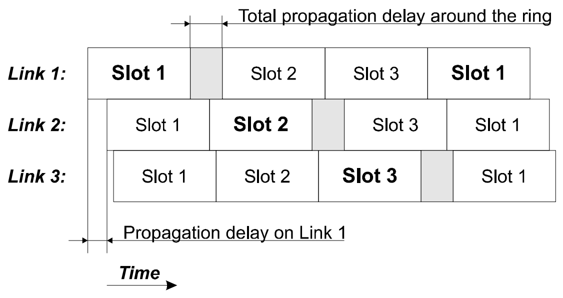

Figure 6: The bandwidth utilization depends on the ratio between the total propagation delay around the ring and the cycle length. The bolded text notify which link each slot first propagates through.

3.4), global reduction (Subsection 3.5), and reliable transmission (Subsection 3.6).

3.1 Real-time virtual channels

Many computer systems have real-time demands where the network must offer logical connections with guaranteed bandwidth and bounded latency. This can be done in the network by using slot reserving. We refer to such connections as RTVCs. Either the whole ring is reserved for a specific node in a slot, or several segments of the ring is dedicated to some specific nodes.

When slot reservation is allowed the cycle is prolonged to contain $M(N_{ord} + N_{res})$ slots, where N_{ord} and N_{res} are the number of slots for ordinary use and for reservation, respectively. The values of N_{ord} and N_{res} are chosen at system design and remain unchanged during operation of the network if the system function does not change radically. The N_{ord} ordinary slots cannot be reserved. However, all nodes in the network can try to reserve a segment of the ring in a reservation slot, not only the SI of that slot. In each node, a separate packet-queue is provided for each of its RTVCs.

When a node wants to reserve a slot for an RTVC, it searches for slots where the required links are free, so allocation of a new segment can be done. First, the node's own slots (i.e., where the node itself is the slot-initiator) are searched. If not enough slots (actually only a segment in each slot) could be allocated for the reservation, the search is continued in other slots. In this case, the node broadcasts a packet containing a request to all other nodes to allocate the desired segment in their slots. The packet contains information about the links required and the amount of slots needed. Each node then checks if any of its own slots have the required free links. All nodes sends a packet back to the requesting node to notify which slots, if any, that have been allocated. When the requesting node has received the answers, it decides if it is satisfied with the number of allocated slots. If not, it sends a release packet. Otherwise, it can start using the reserved slots immediately. However, it should still send a release packet if more slots than needed were allocated.

3.2 Guarantee seeking messages

Guarantee seeking messages normally have hard timing constraints. If the communication system cannot guarantee the timing constraints of a guarantee-seeking message, the owner of the message should be aware of it immediately. In the CC-FPR network, a guarantee is only given if enough deterministic bandwidth (slots) owned by the node is free before the deadline of the message. The available deterministic bandwidth corresponds to ordinary slots where the node is the slot-initiator, and which are not already encountered for other guarantee seeking messages queued in the node.

Each transmitter has M – 1 queues for guarantee seeking messages, one for each possible destination (the node itself excluded). When a multicast packet arrives for queuing it is put in the queue corresponding to the destination of the multicast destinations furthest away from the source node downstream. In this way multicast packets are treated in the same way as single-destination packets and multiple multicast packets can be traveling in the network at the same time when possible.

3.3 Best effort messages

Best effort messages can be sent in all ordinary slots where the node is SI and do not have any guarantee seeking messages that can be sent. In competition with the other nodes according to the CC-FPR protocol other ordinary slots can also be used but, again, only as long as no guarantee-seeking messages can be sent. The same method is used for reservation slots that are not (or only partly) reserved for the moment or reserved but not used in the current slot.

As for guarantee seeking messages each transmitter has M – 1 queues. The messages in each of these queues are sorted according to the EDF-algorithm (Earliest Deadline First). If the deadline expires, the sending process is notified.

3.4 Barrier synchronization

Barrier Synchronization (BS) is an operation to control the flow of processes in a distributed processing system. A logical point in the control flow of an algorithm is defined, at which all processes in a processing group must arrive at before any of the

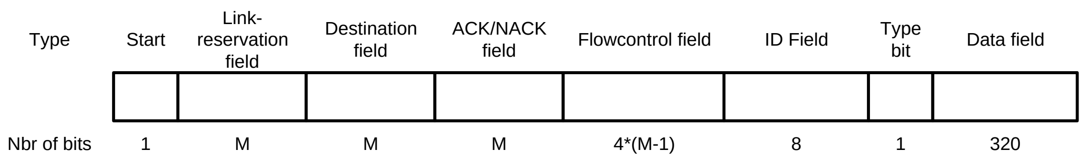

Figure 7: Detailed description of the control packet contents.

processes in the group are allowed to proceed further. When, during execution, a BS point is encountered in the application program, the node broadcasts the encountered BS_id in the control packet when the node is SI+1. In this way all nodes are notified that the node has reached the BS point. Nodes belonging to a different BS group can ignore the broadcast, but nodes belonging to the same group, i.e., has the same id, will make a note in an internal table. The control packet contains a field in which BS_id can be sent (see Figure 7). The id field contains 8 bits, which permits ids ranging from 1 to 255. When the field is zero no BS command is sent.

When a node participating in the BS group has received the correct BS_id from all the participants it knows that all the other nodes are at the same executing point and may proceed. The worst case latency, for a node that reaches the BS point until it can broadcast this to the other nodes, is one cycle. This assumes one slot per node and that each node is SI+1 only once per cycle. Clearly the implications of sending BS information in the control channel is both bounded latency and better bandwidth utilization in the data channel. The whole BS mechanism is handled by the communication interface, transparent to the calling user processes.

In the description above, static allocation of barrier synchronization BS_ids is assumed. The programmer (or the compiler) allocates the required parameters for BS and GR off-line, before runtime. With minor adjustments, dynamic allocation is also possible but is not investigated further in this paper.

3.5 Global reduction

Global Reduction (GR) is similar to barrier synchronization where data is collected from distributed processes when they signal their arrival at the synchronization point. A global operation, e.g., sum or product, is performed on the collected data so that only a single value is returned. At the end of the GR all participating nodes have access to the same data. As in the case of BS we assume that the programmer (or the compiler) statically allocates the necessary parameters, off-line, before runtime.

The GR command requires the following parameters: operation, length, data type, GR_id, and the ids of the participating nodes. The last two parameters are similar to the parameters for the BS command. The operation parameter tells the nodes what operation (sum, product, max, min, etc.) should be performed on the received data. The data type parameter indicates how the data field in the control packet (see Figure 7) is to be interpreted and the length parameter tells the length of the data field. In the studied case, the data field is 320 bits long and may facilitate the transfer of, e.g., up to five double precision floating point numbers (IEEE-754). Other data types may also be distributed. Except for the additional fields and the global function, the nodes treat GR commands in the same way as BS commands. Further on the same reasoning of performance advantages also holds for GR.

The type bit tells whether the control-packet contains a BS or a GR command, and hence whether data is contained in the data field or not (see Figure 7). The data field is currently only used for data reduction.

3.6 Low-level support for reliable transmission

The proposed network has low level support for reliable transmission [9]. Network control information, acknowledge/negative acknowledge and flow control, is sent in the control channel instead of in ordinary data packets. This results in less or no overhead in the data channel, i.e., better bandwidth utilization. The field in the control packet named ACK/NACK contains bits that correspond to the M packets that may have been received by the current SI+1 during the previous M slots. The ACK/NACK information is therefore always sent when a node is SI+1. If a packet was correctly received (correct checksum) a “1” is written in the position of the ACK/NACK field that corresponds to the slot that the packet was received. If a faulty packet or no packet was received a “0” is written. All nodes must keep track of their transmissions and can therefore resolve the meaning of the bits in the ACK/NACK field. In this way the nodes can be notified if their packet was correctly received or has to be retransmitted. The latency for a node to send and receive ACK/NACK is bounded which is desirable.

The $4(M - 1)$ bits in the flow control field relate to independent logical connections. Put simply, each node

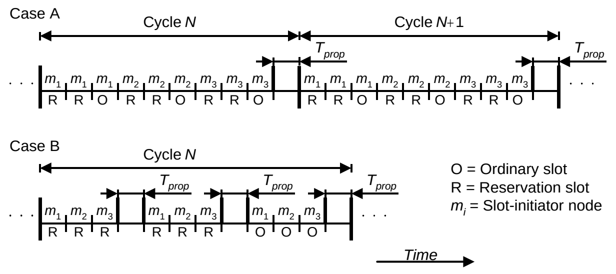

Figure 8: Slot-distribution according to Case A and B. In the example, N_r = 2, N_o = 1, and M = 3.

can have up to four logical connections, with low-level support for flow control, from each other node. The SI+1 sets the bit corresponding to a logical connection to "1" if it is to be halted temporarily, else the bit is set to "0".

4 Performance analysis

To be able to give guaranteed real-time services, worst-case performance must be determined. An analysis of how worst-case latency and deterministic throughput vary with the network design parameters is given below. Each node is assumed to have N_{ord} ordinary slots and N_{res} slots for which reservation is allowed. Two ways of circulating the role of being slot-initiator are treated and compared to each other (see Figure 8): Case A where each node is slot-initiator in N_{ord} + N_{res} slots in sequence, and Case B where the slots are interleaved in a way that each cycle is in practice divided into N_{ord} + N_{res} sub-cycles, where each node is slot-initiator once per sub-cycle. Throughout the section, the total propagation time around the ring is denoted as T_{prop}, the skew between a control-packet and the data-packet it arbitrates for as T_{skew}, while M is the number of nodes in the network. The value of T_{skew} is assumed to be equal to the duration of one slot, T_{slot}, which is set to 1 μs in this analysis.

The worst-case latency a node experience for guarantee-seeking traffic is when all reservation-slots are reserved for other traffic and the only ordinary slots it gets access to are those where it is the slot-initiator. If a packet is generated just after the node had the chance to send, the node has to wait until the next ordinary slot it is slot-initiator in. The latency for Case A will then be:

$$T_{lat} = T_{slot}(M(N_{ord} + N_{res}) - N_{ord} + 1) + T_{prop} + T_{skew}. \qquad (1)$$

The equation holds for best-effort traffic too if no other (best-effort or guarantee-seeking) traffic is queued in the node to be sent before. The latency is plotted in Figure 9 against the number of nodes, for different combinations of N_{ord} and N_{res}, and for different values of T_{prop}. The same worst-case latency but for Case B is:

$$T_{lat} = (T_{prop} + T_{slot}M)(1 + N_{res}) + T_{skew} \qquad (2)$$

and is plotted in Figure 10. As seen in the figures, Case B can be more sensitive to large propagation delays. With the same assumptions as above, further comparison between Case A and B is provided in Figure 11. Case B can offer lower latency in some situations where N_{ord} > 1, especially for large values of M.

Even if worst-case parameters for guaranteed services must be determined, the worst-case latency at no other traffic in the network is an essential parameter to get a feeling of the more general performance. It is, however,

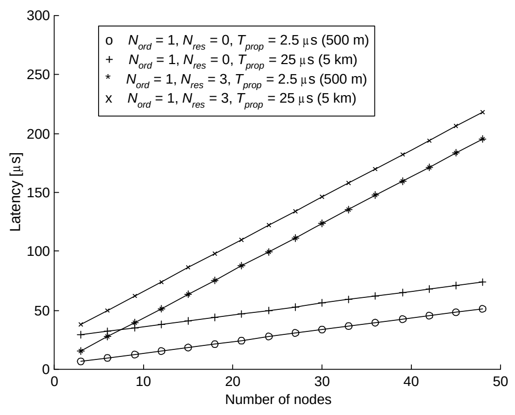

Figure 9: Worst-case latency when the only slots a node gets are those ordinary slots where it is the slot-initiator. Case A is assumed.

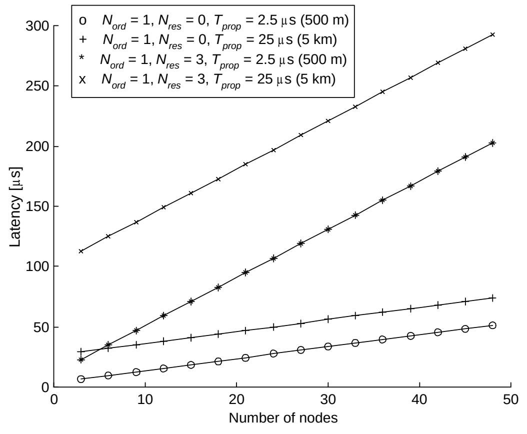

Figure 10: Worst-case latency when the only slots a node gets are those ordinary slots where it is the slot-initiator. Case B is assumed.

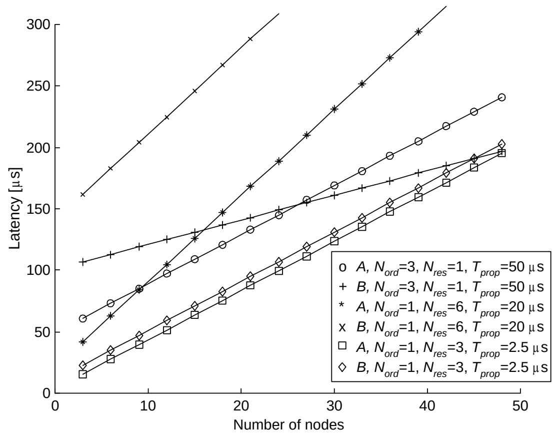

Figure 11: Comparison of worst-case latency for Case A and B.

still assumed that all reservation slots are reserved for other traffic. For communication to the nearest neighbor down-stream, it is possible to send in any ordinary slots. If, instead, the packet must travel over several links, the slots where any of the intermediate nodes is slot-initiator cannot be used. The reason is that the slot-initiator always terminates both clock and data. The worst-case latency for Case A is:

$$T_{lat} = T_{slot}(Q(N_{ord} + N_{res}) - N_{ord} + 1) + T_{prop} + T_{skew} \tag{3}$$

where Q is the number of links a packet has to pass to come to the destination node. The gap between each cycle (see Figure 6) is always experienced by a node when it is handing over the role of being slot-initiator to the next node. Because the worst-case latency appears when a node just misses a slot where it is the slot-initiator (last of them if there are several in a sequence), the whole T_{prop} is always a part of the latency, even if $Q < M$. Because of the similarity between Equation 1 and 3, a plot of Equation 3 would look the same as the plot of Equation 1 in Figure 9, except that Q is represented on the x-axis. The same worst-case latency but for Case B is:

$$T_{lat} = T_{slot}(N_{res}M + Q) + T_{skew} + T_{prop}(1 + N_{res}) \tag{4}$$

and varies with respect to both Q and M but not with respect to N_{ord}.

For the ability to give a bounded latency coupled to an RTVC, the placement of the reserved slots, for the RTVC, in the cycle must be concidered. The N slots reserved for an RTVC are denoted as s_i, $1 \le i \le N$. The worst-case latency is:

$$T_{lat} = \max\{N_{gap\_i}T_{slot} + N_{SI\_pass\_i}T_{prop} + T_{skew} \,|\, 1 \le i \le N\} \tag{5}$$

where N_{gap_i} is the amount of slots between the start of slot s_i and the start of slot s_{i+1} (s_0 in next cycle if $i = N$). The number of slots for which the source node is handing over the role of being slot-initiator to the next node during this time is denoted as $N_{SI\_pass\_i}$.

The maximum aggregated throughput in the network is:

$$S_{\max} = \frac{(N_{ord} + N_{res})MPT_{slot}}{(N_{ord} + N_{res})MT_{slot} + T_{prop}} \tag{6}$$

where P is the average number of transmissions possible per slot due to spatial slot-reuse, and Case A is assumed. The maximum aggregated throughput is plotted in Figure 12, for $P = 1$. When $P > 1$, the throughput plotted in the figure is simply multiplied by P. At low values of T_{prop}, throughputs near P can be achieved even for low values of M. The corresponding throughput for Case B is:

$$S_{\max} = \frac{MPT_{slot}}{MT_{slot} + T_{prop}} \tag{7}$$

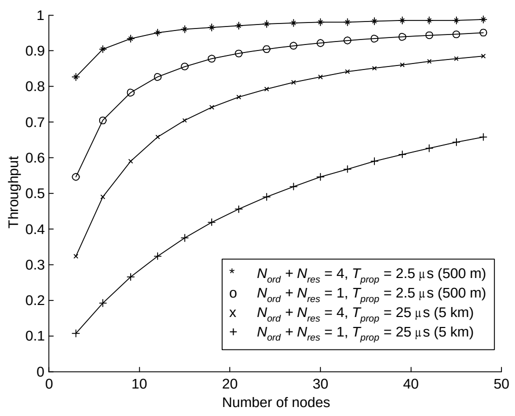

Figure 12: Maximum aggregated throughput. Case A is assumed.

and is plotted in Figure 13. As seen, the maximum throughput for Case B does not vary with respect to N_{ord} and N_{res}. When $N_{ord} + N_{res} = 1$, Case A and Case B gives the same aggregated throughput. If $N_{ord} + N_{res} > 1$, however, Case A gives better aggregated throughput.

The same performance difference, as in the case of aggregated throughput, is found when looking at the throughput corresponding to one slot per cycle. The

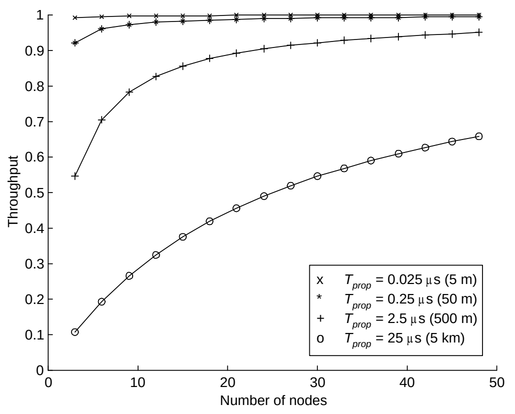

Figure 13: Maximum aggregated throughput. Case B is assumed.

single-slot throughput is relevant information for all kind of traffic, but especially when calculating guaranteed bandwidth for an RTVC. It also determines the minimum throughput a node gets if it only can send in a single ordinary slot per cycle, i.e., the only ordinary slot for which the node is slot-initiator if $N_{ord} = 1$. The single-slot throughputs are:

$$S_{\max} = \frac{T_{slot}}{(N_{ord} + N_{res})MT_{slot} + T_{prop}} \tag{8}$$

and

$$S_{\max} = \frac{T_{slot}}{(N_{ord} + N_{res})(MT_{slot} + T_{prop})} \tag{9}$$

for Case A and B, respectively.

5 Conclusions

We have presented a fiber-ribbon based ring network with services for parallel and distributed real-time systems. A key component of the network architecture is the flexible control channel that can be configured to be used for different types of network control. Examples of this is the low-level support for barrier synchronization, global reduction, and reliable transmission. A big advantage is that the low-level support can be implemented with little or no modifications to existing hardware. High throughputs can be achieved in the network, especially in systems where some kind of pipelined dataflow between the nodes exists. The network offers real-time services both for logical connections with guaranteed performance, RTVCs, and for separate messages, best effort and guarantee seeking messages. An analysis of worst-case latency and deterministic throughput has been provided for two variants of time-slot organization. One offers higher throughput while the other offers lower latency in some situations. Also worth mentioning is that the network can be built today using fiber-optic off-the-shelf components and that this is ongoing work.

6 Acknowledgement

This work is part of the REMAP project, financed by NUTEK, the Swedish National Board for Industrial and Technical Development, and the PARAD project, financed by the KK Foundation in cooperation with Ericsson Microwave Systems AB. OPTOBUS components for the bulding of a prototype has been sponsored by Motorola.

References

[1] M. Jonsson, “Two fiber-ribbon ring networks for parallel and distributed computing systems,” *Optical Engineering*, vol. 37, no. 12, pp. 3196-3204, Dec. 1998.

[2] D. Bursky, "Parallel optical links move data at 3 Gbits/s," *Electronic Design*, vol. 42, no. 24, pp. 79-82, Nov. 21, 1994.

[3] K. Arvind, K. Ramamritham, and J. A. Stankovic, “A local area network architecture for communication in distributed real-time systems,” *Journal of Real-Time Systems*, vol. 3, no. 2, pp. 115-147, May 1991.

[4] P. C. Wong and T.-S. P. Yum, "Design and analysis of a pipeline ring protocol," *IEEE Transactions on communications*, vol 42, no. 2/3/4, pp. 1153-1161, Feb./Mar./Apr. 1989.

[5] H. Jafari, T. G. Lewis, and J. D. Spragins, “Simulation of a class of ring-structured networks,” *IEEE Transactions on Computers*, vol. 29, no. 5, pp. 385-392, May 1980.

[6] M. Xu and J. H. Herzog, “Concurrent token ring protocol,” *Proc. INFOCOM’88*, pp. 145-154, 1988.

[7] M. Jonsson, B. Svensson, M. Taveniku, and A. Åhlander, “Fiber-ribbon pipeline ring network for high-performance distributed computing systems,” *Proc. International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN’97)*, Taipei, Taiwan, Dec. 18-20, 1997, pp. 138-143.

[8] B. J. Sano and A. F. J. Levi, "Networks for the professional campus environment," in *Multimedia Technology for Applications.* B. Sheu and M. Ismail, Eds., McGraw-Hill, Inc., pp. 413-427, 1998, ISBN 0-7803-1174-4.

[9] C. Bergenhem and J. Olsson, “Protocol suite and demonstrator for a high performance real-time network,” *Master thesis, Centre for Computer Architecture (CCA), Halmstad University, Sweden*, Jan. 1999.