Real-time communication protocols: an overview[*]

Ferdy Hanssen and Pierre G. Jansen

October 2003

Abstract

This paper describes several existing data link layer protocols that provide real-time capabilities on wired networks, focusing on token-ring and Carrier Sense Multiple Access based networks. Existing modifications to provide better real-time capabilities and performance are also described. Finally the pros and cons regarding the At-Home Anywhere project are discussed.

1 Introduction

In the past twenty-five years several different, wired networks have been designed and built. But not all of them are dependable, i.e. suitable for accomodating real-time traffic, or traffic with Quality of Service (QoS) requirements. Some were designed with QoS in mind, some had QoS added as an after-thought, and some do not support QoS requirements at all.

The most popular wired network architecture used to date, IEEE 802.3 [46], popularly called Ethernet[1], does not provide any support for real-time traffic. There are no provisions to reserve bandwidth for a certain connection. Nodes access the network using the CSMA/CD (Carrier Sense Multiple Access with Collision Detect) technique [70]. The CSMA/CD algorithm does not define a collision resolvance protocol of its own. Ethernet uses the BEB (Binary Exponential Back-Off) algorithm. This algorithm resolves collisions in a non-deterministic manner, while the first requirement for a network to be called real-time is determinism.

The most popular real-time network architecture, Asynchronous Transfer Mode (ATM), is used mostly for internetwork links these days. ATM provides high bandwidths and QoS guarantees, but has proven to be too expensive for use as a local area network. For industrial applications there are communication networks

[*]This work is sponsored by the Netherlands Organisation for Scientific Research (NWO) under grant number 612.060.111, and this work is supported by the IBM Equinox programme.

[1]This paper will use the term Ethernet throughout for networks adhering to the IEEE 802.3 standard.

with deterministic Medium Access Control (MAC) layers, like Process Field Bus (PROFIBUS) [88], Factory Instrumentation Protocol (FIP) [31], and Controller Area Network (CAN) [47].

The token-bus network IEEE 802.4 [42] is no longer an IEEE standard, but contains some useful concepts, so it will be mentioned briefly. The token-ring network IEEE 802.5 [44] still is an IEEE standard, but its use is declining in favour of IEEE 802.3 based networks. The "improved version", known as Fiber Distributed Data Interface (FDDI) [6–9], provides useful concepts, but is also considered too expensive for use as a local area network. Both provide useful insights and will be discussed in some detail.

In this paper we are going to look at wired local area network architectures, which provide support for real-time traffic. We focus on the medium access layer, where token-ring networks will be discussed, followed by the token-bus network, field control buses and finally various approaches to making Ethernet real-time. All network types will be briefly explained, and their advantages and disadvantages with regard to the At-Home Anywhere project (@HA) will be discussed. Important network aspects with regard to the @HA project are real-time capabilities, component simplicity, and component cost. The higher layers of the protocol stack will also be discussed briefly.

2 Token-ring networks

IEEE 802.5 and FDDI are both ring-shaped networks [38], as depicted in figure 2.1. Both employ a token — a special kind of network frame — to regulate network access of the individual nodes. This token flows from node to node (figure 2.2), and each node may transmit a message only when it has acquired the token. The main difference between IEEE 802.5 and FDDI networks lies in the rules, which dictate when a node may acquire a token, and for how long.

2.1 IEEE 802.5

A token-ring network, as defined in the IEEE 802.5 specification [44], which evolved from the IBM token-ring network [97], makes use of a priority scheme for token acquirement. The token circulates around the ring, and a node which has data to transmit seizes the token. This is done by modifying the token bit, which indicates whether the network packet is a token or a frame. Then the frame control fields, such as source and destination, and the data is appended, together with anything else needed to create a valid network frame. This frame then circulates the ring. The destination node copies the data in the frame, for use by its applications, but still

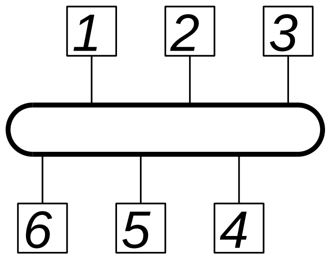

Figure 2.1: Ring architecture

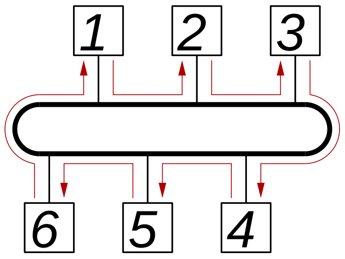

Figure 2.2: Token-ring network with flow of token

forwards the frame. When the frame returns to the initiating node, it is removed from the network. The amount of data which may be transmitted by a node when it has hold of the token, is limited by a token holding timer. If a node's transmission queues do not contain any more data to send, a new token is generated which is then sent onto the ring.

To be able to support different classes of network traffic, IEEE 802.5 has the option of assigning priorities to messages. Each data frame and token contains three bits for priority, thus providing eight levels of priority. A node may only seize the token if it has data to transmit at a priority level, which is at least equal to the priority of the frame currently travelling the network. And all data transmitted during the capture of the token, may only be of equal or higher priority than the priority of the token.

Additionally, each data frame and token have a field indicating the highest priority of any node in the ring. These reservation bits are inspected each time the frame passes a node. If this node has data to transmit with a priority higher than the current reservation, the reservation bits are updated to match the priority of the data the node wants to send. In this way the node requests the next token to be issued with a priority equal to the priority of the data it has waiting at the head of its queue. When the token returns to the node with the highest priority, it may access the network medium and transmit its data.

If a node issues a token with a priority higher than the preceding token, it becomes responsible for making sure that a token with the lower priority of that preceding token will be released on the network at some later point in time. IEEE 802.5 token-ring networks restrict fairness to messages with the same priority. Lower-priority packets may experience starvation, which is caused by large amounts of higher-priority packets waiting in the transmit queues on the various nodes.

IEEE 802.5 token-ring networks are defined for two operating speeds: 4 Mbit/s and 16 Mbit/s. At the higher speed the feature of early token release is supported, which means that a node may release a token as soon as it has transmitted its last bit. It does not have to wait for the last frame to completely traverse the ring. Early token release increases the available bandwidth and improves data transmission efficiency when the frame is shorter than the ring latency. But access delays for higher-priority traffic may increase when a very high load of short frames is placed upon an early-token-release enabled network.

Strosnider et al. [98] proposed a different method for scheduling the IEEE 802.5 token-ring network that not only guarantees deadlines for the synchronous traffic, but also reduces the response time for asynchronous traffic. They apply a scheduling approach which is based on algorithmic techniques, so schedulability decisions can be made on arbitrary message sets. The costly tuning of network parameters is then no longer needed. Their approach allows IEEE 802.5 to be deployed in flexible environments, where the network is both real-time and responsive for non-real-time traffic.

Sathaye and Strosnider [93] have examined scheduling models for both early token release and normal token release on IEEE 802.5. They found that the maximum bandwidth utilization of both protocols is the same when the packet size is not smaller than the time needed for a signal to circulate the ring, but that using early token release can significantly improve bandwidth utilization. But the early token release protocol has a larger blocking component than the other, which can hurt performance when a large number of nodes with relatively long deadlines and a few nodes with short deadlines are present. The large amount of blocking incurred by the early-token-release mechanism then will probably cause deadline misses for packets with a short deadline.

Ng and Liu [77] simulated a token-ring network, where a high-speed network is split into multiple low-speed subnetworks, which should increase the effective network bandwidth and decrease the response time. Also the cost of a network should go down when this is done, as low-speed interface hardware is an order of magnitude less expensive as high-speed interface hardware. Then even several low-speed rings do not cost as much as one high-speed ring. Their topology is built such that every node is connected to a subset of rings: station i is connected to subrings $\{(j-i) \bmod K | j \in [0, 1 \ldots \lceil \frac{K-1}{2} \rceil]\}$, where K is the number of subrings. This so-called staircase topology requires a smaller number of node-network connections, but has a performance which is comparable to the fully connected protocol as long as the source stations are not highly localized and the destinations are equally likely to be in any subnet.

2.2 FDDI

The important difference between IEEE 802.5 token-ring networks and FDDI networks lies in the handling of the token. Where IEEE 802.5 employs a priority token, FDDI makes use of a timed token protocol, derived from the protocol defined by Grow [35, 36] and Ulm [105]. Some introductory texts on FDDI are given by Ross [89–91].

The time it takes a token to circulate the ring is controlled. The token has to adhere to the Target Token-Rotation Time (TTRT), which is defined when the network is initialized, accounting for the maximum propagation time of the entire ring, the time required to transmit a maximum length frame, the time required to transmit a token, and the sum of all the synchronous allocations of all the nodes. Synchronous allocations are the amount of bandwidth a node wishes to use for transmitting synchronous, or stream data. FDDI also supports the asynchronous data type, to be used for bursty, or best-effort data.

The timed token protocol of FDDI makes use of a Token-Rotation Time for each node i (TRT_i), a Token-Holding Time for each node i (THT_i), and a Late Counter for each node i (LC_i). Also known are the Target Token-Rotation Time ($TTRT$) and the synchronous capacity H_i for each node i, which represents the maximum amount of time a node may transmit synchronous traffic.

The protocol operates as follows [18]. When the ring is initialized, THT_i and LC_i are set to 0 and TRT_i is set to $TTRT$ for all nodes i. The TRT_i timer always counts down; as it reaches 0 it is reset to $TTRT$ and LC_i is incremented with 1. Normally, when $LC_i > 1$, the ring recovery process is started [51].

A token is considered early when $LC_i = 0$ when the token arrives, otherwise it is considered late. At early token arrival, the following takes place. THT_i is set to TRT_i, TRT_i is reset to $TTRT$, and synchronous frames can be transmitted for a maximum duration of H_i. Finally timer THT_i is enabled, and it starts counting down. As long as $THT_i > 0 \wedge TRT_i > 0$ asynchronous frames may be transmitted.

When the token is late, however, the following will occur. LC_i is set to 0, and TRT_i continues to count down to 0, it is not reset to $TTRT$. Now synchronous frames may be transmitted for a maximum duration of H_i, and no asynchronous frames will be transmitted.

FDDI supports the notion of a double-ring network: a primary ring and a secondary ring with counter-rotating traffic (figure 2.3). Contrary to IEEE 802.5, where the secondary, backup ring can only be activated manually, FDDI supports automatic fault recovery using the second ring. When a fault occurs in the main ring, this is detected and the backup ring is used to continue operation (figure 2.4).

Sevcik and Johnson [52,95] first proved that the maximum token rotation time in FDDI is two times the TTRT, later Chen and Zhao [19] proved that the maximum

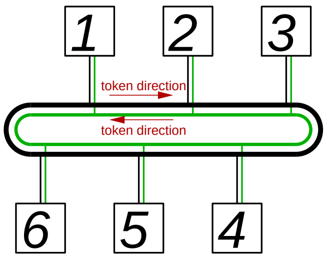

Figure 2.3: Token-ring network with a double ring

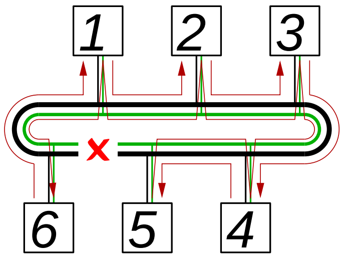

Figure 2.4: Token flow on a broken, double-ring network

time that can elapse between any v token arrivals ($v \geq 2$) at a node i will not exceed $(v-1) \cdot TTRT + \sum_{h=1}^{n} H_h - H_i + a$, where H_j is the synchronous capacity allocated at node j and a is the portion of the TTRT that is unavailable for synchronous messages. Recently Zhang et al. [117] have given a formal proof of an even tighter upper bound of this maximum time between token arrivals at a single node. For hard real-time purposes it is important to know that any periodic stream k with a period P_k, where $P_k \geq 2 \cdot TTRT$, can be served on an FDDI network.

Dykeman and Bux [28] have shown a fundamental property of FDDI: there exists a tradeoff between efficiency and effectiveness. They have shown that the maximum total throughput remains high when the TTRT is large compared to the ring latency. But the total throughput may be severely limited when just one application requires a very short transmission delay. Even if this application is rarely active, the TTRT has to be set to one half of the maximum acceptable delay. This increases the bandwidth needed for circulating the token, and thus decreases the bandwidth avaliable for data.

A useful performance metric for networks is the Worst-Case Achievable Utilization (WCAU): the higher this is, the more useful data can be pumped around the network. For FDDI the WCAU depends on the allocation scheme of the synchronous capacity at each node. Several allocation schemes have been proposed by different groups [1, 18, 116, 122] which provide WCAUs of 33% [1, 122] and 40% [18, 116], all under the assumption that all synchronous traffic uses deadlines equal to its periods. Zhang et al. [118] later proposed an allocation scheme which allows more messages sets to be scheduled than their earlier work [116]. Recently Buzluca and Harmancı [16] have proposed an allocation scheme which adapts better to dynamic environments, but in order to do this they need a small amount of global information, whereas the previously described allocation schemes need only information which is local to the node.

All the allocation schemes mentioned above assume a TTRT is chosen that is as large as possible, to maximize the overall usable bandwidth. Malcolm and Zhao [66, 67] and Hamdaoui and Ramanathan [37] showed that a much higher WCAU can be reached, when the TTRT is chosen to be the optimal value for the synchronous traffic offered to the network, related to the minimum deadline of said traffic. They showed that a higher minimum deadline will lead to a higher WCAU. However, in an environment where the synchronous traffic changes regularly, the optimal TTRT changes with it. But the TTRT is set at network initialization, so this would require the FDDI network to be reinitialized every time a synchronous stream is added, removed or changed.

If an absolute guarantee that all messages deadlines are met is not necessary, Kamat et al. [53] have developed a method of calculating the guarantee probability that all deadlines will be met, given the synchronous message set utilization. If the absolute guarantee that no messages are lost is important, Malcolm et al. [65] developed tests to make sure whether message loss can occur.

2.3 IEEE 802.4

The IEEE 802.4 standard is different from the previously discussed token-ring networks, as this employs a logical ring upon a bus network. It has been withdrawn as an IEEE standard, and is, to our knowledge, no longer or hardly in use today. The protocol used to guide the token is in principle the same as in FDDI: the timed token protocol.

A bus network, as seen in figure 2.5, is transformed into a token-ring network, by letting a token run through the network, cyclically from node to node, in a predetermined order, as shown in figure 2.6. A node may only transmit when it has the token, so collisions on the bus are avoided. When a node gets hold of the token, it may transmit for a specified duration. Each node keeps track of the amount of time elapsed between tokens. This is done by using a timer, which is started when a token is received and keeps running until the next token arrival.

Four data classes are defined, ordered to priority. They are numbered 6, 4, 2, and 0. This scheme was designed to send higher-priority (i.e. class 6) data first and only then lower-priority data, when enough bandwidth was left on the network. With each data class a constant is associated. Class 6 data is associated with THT, the token holding time, this is the maximum duration any node may use to transmit class 6 data. Data in class 4 to 0 is associated with TRT_4, TRT_2, and TRT_0 respectively. It holds for each of these constants that TRT_j is the maximum time

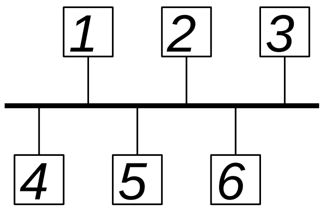

Figure 2.5: Bus architecture

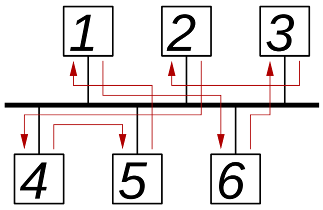

Figure 2.6: Token-bus network with flow of token

that may be used for token rotation, which will still allow transmission of class j data. Pang and Tobagi [81] have shown that this timing mechanism functions correctly under heavy load.

Montuschi et al. [74] have derived the maximum duration of a complete token cycle. They have shown that the mean token rotation time is T_a and that the maximum token rotation time is $T_a + H$. Here T_a is the maximum of all values for TRT_4, TRT_2, and TRT_0 of all nodes, and H is the sum of all values for THT of all nodes. These values are useful to determine the values for the different network parameters that are needed to create a network that provides good responsiveness and throughput.

Moon et al. [76] have studied the IEEE 802.4 network when it is used in a noisy environment. They found that the token is easily lost in certain noisy environments and that the time of recovering the network from such a situation is considerable. Token loss is the major reason for performance degradation of an IEEE 802.4 network.

Cheng et al. [22] enhanced the token bus protocol to make it more suitable for multimedia applications. They proposed not to let the token traverse the network in a fixed order, but to let the token visit certain nodes more than once during a token cycle. This was done to let each node have a token visit cycle which is different from the others. So nodes that have real-time data with a higher periodicity or that need to access the network more frequently for another reason, will also see the token more frequently. A *central network manager* takes care of the token visiting arrangements and makes sure all nodes know about them. Thus each node knows when to expect the token next, and each node can request a different token frequency to account for newly arrived or deleted streams.

2.4 Performance of the token protocols

For any network using the timed token protocol, choosing the proper values for the various TRTs is difficult, and depends on the actual network setup. According to Valenzano et al. [106], it is important to tune these values well, as their results show little difference between the values under worst-case assumptions and under best-case assumptions. Jayasumana [49] developed a model to determine the timer values for the IEEE 802.4 network, and Montuschi et al. [75] presented an analysis to determine the minimum requirements for the values of the THTs and how to use these theoretical results to tune the network performance.

Colvin and Weaver [24] looked at the message delay versus the network utilization for the IEEE 802.4 network, and found that the mean delay increases exponentially with the utilization when the utilization is less than approximately 0.6, and that the mean delay increases more than exponentially when the utilization is larger than this. They also found that the mean delay of the synchronous messages is larger than the mean delay of the asynchronous messages.

Jayasumana and Werahera [50] developed an analytical model to evaluate the throughput of an FDDI network in the presence of multiple classes of traffic. Their simulations coincide with the model, and show that the guaranteed throughputs for each of the classes depend on the chosen TTRTs for each class. It is possible that, given certain TTRTs, the guaranteed throughput of a lower priority class can be higher than that of a higher priority class. They also show that the message delay for each class increases gently up to the overload point, where the delay starts to increase more than exponentially.

Chen and Bhuyan [20] improved performance by building a network of multiple rings. They presented analyses for three kinds of protocols for a multiple ring network: one queue per ring with simultaneous transmissions, a single queue for all rings with simultaneous transmissions, and a single queue for all rings with single transmission. Simultaneous transmissions means that a node may capture more than one token simultaneously, single transmission means that a node may capture only one token at a time. Their simulations agree with their analyses with an error of at most 10%, showing that multiple ring networks provide almost linear performance increases in transfer time with the number of rings, up to four rings, and that their single queue for all rings strategies work better than the one queue per ring strategy, where the single queue with simultaneous transmissions protocol is slightly better than the single queue with single transmission protocol.

Timed token protocols do not guarantee the token to arrive at each node at the beginning of a period. Choo and Kim [23] proposed an approach where the token arrives exactly at the beginning of the requested period, enabling reliable periodic communication. Every node may transmit for at most as long as its THT. When that amount of time is not needed, it adds to the token the amount of time not used, which can then be used by the next token for transmitting non-periodic messages. Their approach guarantees a minimum period of $1 \cdot TTRT$, where $TTRT$ is set to the sum of the bandwidth allocations of all nodes.

3 Broadcast networks

Broadcast networks are typically set up as a shared-bus network, where any node can talk directly to any other node or all other nodes. The best-known example of such a network is Ethernet, which is widely being used in the world today. Current Ethernet networks operate at a speed of 10 Mbit/s or 100 Mbit/s and can typically be found in home and office environments. Ethernet networks which operate at 1000 Mbit/s are beginning to be used in data and network centres. There have been and there still are people working on creating deterministic Ethernet networks, enabling it to be used for real-time purposes [5].

Ethernet uses CSMA/CD with BEB to resolve concurrent medium accesses, which makes proper real-time guarantees for network messages impossible. When two or more nodes decide to transmit a message almost simultaneously, i.e. within a slot time[2], a collision occurs on the network. The BEB algorithm can take up to 7151 slots time before the packet is delivered or returned as non-deliverable. This equals 366 ms for a 10 Mbit/s Ethernet network.

The BEB protocol operates as follows. Each node maintains a collision counter c, which is set to zero initially and after each succesful transmission. Whenever a collision is experienced, c is increased with one, until the maximum 16 is reached, when a message is marked non-deliverable. A back-off time b, chosen from the uniformly distributed range $[0, 1 \dots 2^{\min(c,10)} - 1]$, is then used to wait for b slots before the next transmission of the same packet is attempted.

The average message delay increases linearly as the amount of traffic increases, according to the measurements of Boggs et al. [13]. However, to deterministically send messages, one would have to make worst-case assumptions. But the maximum waiting time is so unrealistically large and occurs so rarely that this is not a feasible option. Several techniques have been developed to enable broadcast networks, and Ethernet in particular, to support real-time network traffic.

The most simple approach to have some guarantees on a bus network is by using a TDMA (Time Division Multiple Access) approach. Here every node has a pre-allocated amount of time to transmit its packets. The bus is then only ever in use by at most one node, thus rendering a collision-free network with predictable timing.

[2]A slot time is 512 bit time (e.g. 51.2 µs for a 10 Mbit/s Ethernet and 5.12 µs for a 100 Mbit/s Ethernet) [13].

The main disadvantage is the inflexibility, everything has to be pre-allocated. Nodes that do not have any packets to send, will still reserve the network for their allocated share, bringing down the efficiency of it all. Also it is hard to efficiently accomodate for nodes joining and leaving the network. This technique has been used in practice as the basis for some real-time communication protocols, specifically in the Mars project [56].

3.1 Constraining generated traffic

Providing some guarantees on bandwidth on a network can also be achieved by putting contraints on the traffic generated by the nodes. There are three classes of methods to do this, without making changes to the Ethernet protocols. The traffic smoothing approach [58, 59] uses a statistical method to bound the medium access time by limiting the packet arrival rate at the MAC layer. Thus non-real-time traffic bursts are smoothed, they are spread out over a longer time period, to allow real-time traffic on the network as it arrives. The virtual time and window protocols will be discussed next.

3.1.1 Virtual time CSMA

Virtual time protocols implement a packet release delay mechanism. First proposed by Molle and Kleinrock [73], the protocols work as follows. Each node has two clocks, one gives the real time, the other gives the virtual time. When the channel is idle, the virtual clock is running, and it stops running when the channel is busy. When the virtual clock runs, it runs at a higher rate than the real one if it is behind. A message is sent only when its arrival time is equal to the time of the virtual clock. Zhao and Ramamritham [120] describe several virtual time CSMA (Carrier Sense Multiple Access) protocols, in which the virtual clocks differ. The original virtual time CSMA protocol uses the arrival time to determine when to transmit a message. Zhao and Ramamritham [119] proposed a variant which uses message laxity to determine when to transmit. In their 1987 paper [120] they proposed two other variants: transmitting messages with the earliest deadline first and transmitting messages with the shortest length first.

Zhao and Ramamritham used the following collision resolution algorithm in their simulations. When a collision occurs, the sender node i either retransmits the message immediately with a probability P_i, or modifies the virtual time to start transmission of the message to a random number drawn from a range chosen specifically for each protocol variant, after which the message is put back in the queue of messages waiting to be transmitted. Their simulations showed that all four variants are insensitive to the number of nodes in the network and the protocols perform better than those with a straightforward collision resolution algorithm. All variants are sensitive to the rate of the virtual clock when the load is not very light. But, except for cases of extreme network overload or a very large normalized transmission delay[3], there always exists a wide range of virtual clock rates in which each variant has optimal or close to optimal performance.

With light to medium load all four variants perform similarly, but in high load their performance differs. The original virtual time CSMA protocol performs poorly in the message loss and effective channel utilization ratio, but is not biased towards messages based on length or causes too much collisions. The laxity variant performs best in terms of effective channel utilization, but it performs poorly in terms of message loss. The shortest-length-first variant performs well in terms of message loss, but has the worst channel utilization and the worst collision strategy of all four. The minimum-deadline-first variant has the best performance when it comes to message loss and collisions, but is biased towards short messages, as is the previous one. Therefore its channel utilization is not as good as that of the laxity variant. Zhao and Ramamritham concluded there is no variant that outperforms the others, but that the laxity-based and the minimum-deadline-first variants are to be recommended for hard real-time communications.

Recently Salian et al. [92] improved on the minimum-deadline-first variant to support both hard real-time and soft real-time messages. Their variant reserves time slots for hard real-time messages, and their experiments, carried out with a simulation written in OMNeT^{++} [80], showed that all hard real-time messages are transmitted for all load conditions. Soft real-time messages have to be sacrificed to make this possible, but the percentage of soft real-time messages dropped does not deteriorate as much as compared to the minimum-deadline-first variant.

El-Derini and El-Sakka [29] proposed a virtual time CSMA variant which puts message priorities in the picture. Messages belong to one of two priority classes, where higher-priority messages will be given precedence when sending. A node with high-priority messages sends a reservation signal on the network. If this signal does not cause collisions, it is the only one with high-priority messages and transmits them. If it does collide, all nodes with high-priority packets use the normal virtual time CSMA protocol to transmit only the high-priority messages. When no reservation signal is present, low-priority messages may be transmitted. Their protocol has a lower overall throughput than the original virtual time CSMA protocol, but it behaves better for high-priority packets than the original one.

Molle [71] improved the original protocol, allowing for a certain number of priority classes. He uses a virtual clock for each priority class. Messages of the highest priority class are scheduled to be sent first, until they run out or its virtual clock catches up with real time. Then the next-lower-priority-class messages are sched-

[3]The normalized transmission delay is defined as the maximum end-to-end delay divided by the mean message length.

uled to be sent, and so on. Within a class messages are scheduled according to First Come First Served (FCFS). His simulation results indicate the performance of this prioritized virtual time CSMA protocol compares very favourably to other prioritized CSMA protocols [61, 78, 100].

3.1.2 Window protocols

Gallager [33] first proposed a window protocol, in which the enabled set of stations are those that have messages in the queue which were generated during some interval, or window. All nodes have a network-wide knowledge of the packets waiting to be transmitted. The protocol operates roughly as follows [57, 68]. If only one node has a message in the window, that message is sent. If several nodes have a message in the window, the window size is reduced until only one node with a message in the window remains. Then, if there are no nodes with a message in the window, the window is shifted further along the time axis. Management of the window is the responsibility of the access arbitration process.

The traditional window protocol, as proposed by Gallager, has been modified to allow implementation of Minimum Laxity First (MLF) or other policies, in order to make it suitable for real-time applications. Zhao et al. [121] based the window on the latest time to send a message to make it meet its deadline. Their window protocol immediately considers a newly arriving message for transmission if its latest time to send is less than that of all pending messages in the system. Thus their protocol closely approximates the optimal MLF policy. Simulations showed their protocol to perform well, even under overload.

Znati [124] built a simulation of a window protocol which implements the MLF strategy. He describes what needs to be done when two or more messages have the same laxity. This results in a collision, which is resolved by having each node select a contention parameter, based on its latest time to send and its position in the logical ring, in which all nodes have been placed for this protocol. Each node will select a unique contention parameter, which will be used to determine which node may send its message first. He proved that the contention-parameter selection preserves the MLF policy.

3.2 RETHER

The real-time network RETHER [109–111], developed at the State University of New York at Stony Brook, is based on Ethernet. It uses a token-based technique to regulate access of all the nodes to the network. This circumvents the danger of collisions inherent to the CSMA/CD protocol employed on Ethernet networks. And collisions are resolved in a non-deterministic way on Ethernet, where the BEB algorithm is used for this.

RETHER is essentially a token-bus network. But the token does not follow a simple circular path through the network. The token is guided according to a certain schedule. First the token flows to all nodes which have real-time traffic to send. Next the token will flow from node to node in a circular fassion, to allow each node to transmit its non-real-time packets. This circular path is interrupted as soon as the schedule dictates a node's need for transmitting real-time traffic. After this round of real-time traffic the token is again forwarded to the node which was next in line for transmitting non-real-time traffic, and the token continues its round for non-real-time traffic from there.

When there are no nodes with real-time messages to send, the RETHER network protocol operates in CSMA/CD mode, just like Ethernet. The first node, the so-called initiator, that wants to transmit real-time traffic, broadcasts a special message, indicating it wants to switch to the token mode. It then waits for all nodes to respond to this message, indicating their switch to token mode, before generating a token, transmitting its real-time message and forwarding the token to the next node in the network.

Multiple initiators are handled by letting the initiator with the smaller identification number have priority over the others. Node failures mean the initiator does not receive the expected amount of acknowledgements from the other nodes within a certain, predefined time. This is handled by asking the non-responding nodes to acknowledge the transition to token mode again. Nodes that do not respond then will not be added to the list of live nodes in the token.

When no more real-time traffic is present from any node, RETHER will switch back to CSMA/CD mode. This is done by letting the node which terminates the last real-time stream, broadcast a special message. No acknowledgements are used in this case. Any node which does not receive this broadcast, will eventually switch back to CSMA/CD mode automatically. This is controlled by a timer. This timer counts down from a time value set larger than the longest time between token visits and when zero is reached the switch is made.

A similar idea was already proposed by Gopal and Wong [34] earlier. They proposed a hybrid token-CSMA/CD network. In this network collisions are resolved as usual by all nodes on the network, except the node which has the token. This node disables its collision resolution algorithm, and retransmits its packet immediately when the network becomes available, i.e. when the collision detection signal has disappeared. It does not wait for a random time as all the other nodes do, so it gets priority in the collision resolution. The token travels between the nodes on a logical ring. Their protocol, however, does not provide any real-time guarantees. But their hybrid protocol is better than plain CSMA/CD, and it is better than the token-ring protocol with light load. With heavy load their hybrid protocol approaches the performance of the token-ring protocol.

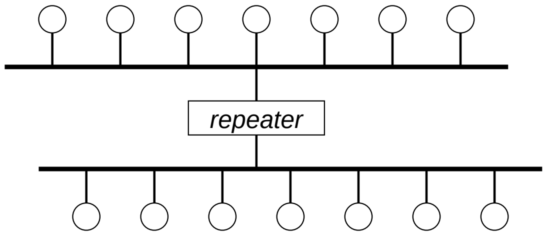

Figure 3.1: Ethernet bus architecture

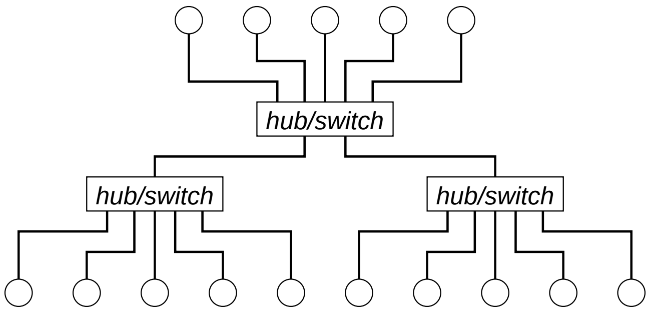

Figure 3.2: Ethernet star architecture

3.3 Switched real-time Ethernet networks

Modern Ethernet networks are more and more being built using switches instead of hubs or coax cabling. Coax cabling is used to build a traditional bus network, with repeaters interconnecting two buses, as in figure 3.1. Hubs and switches create a star-like network, as in figure 3.2. The difference between switches and hubs is the intelligence. Hubs simply pass on incoming traffic on any port to all other ports, whereas switches learn the topology of the network, i.e. where all the nodes are situated. When a packet, destined for a certain known Ethernet address, arrives, it is sent only to the port behind which the switch knows that Ethernet address to be residing.

In a star-like network layout every node is connected with a private cable to a switch. When Universal Twisted Pair (UTP) cabling is used, there is a private cable for each direction. Then collisions can no longer occur on any network cable. This can be used to an advantage when providing QoS guarantees. The real-time capabilities can now be embedded inside the switch. Present-day switches employ a technique called store-and-forward to transfer packets from one port to another, using per-port buffers for packets waiting to be sent on that port. But congestion may occur here, when one node is suddenly receiving a lot of packets from the other nodes. Current switches do not provide any guarantees as to which packets will be sent first. Recent research by Pedreiras et al. [85] shows that switches may behave erratically when high-priority packets are dropped due to a lack of memory capacity.

The EtheReal switch from Varadarajan and Chiueh [107] solves this by including a bandwidth reservation scheme inside the switches. Another design goal of them was to not change the standard operating systems in the nodes. They use a specialized library which the applications needing real-time services can use. This library connects to a special *daemon* process running on the node, which communicates with the switch to set up real-time connections. Special proxy IP (Internet Protocol) and Ethernet addresses of the forms `1.1.`x`.`y and `ff:ff:ff:ff:`xx`:`yy, respectively, are used for real-time streams on all links from source to destination node, except the final link from the last switch to the destination. The receiving nodes are not involved in the connection set-up with EtheReal, and standard UDP (User Datagram Protocol) is used to transmit any real-time stream data.

EtheReal supports automatic fault detection and recovery. Varadarajan and Chiueh [108] have shown an approach in which the effect of network faults is localized to only the affected nodes and switches, without service interruption in other, non-affected parts of the network. They use periodic polls to check for availability of switches. When a network topology change is detected, the switches build a new spanning tree of the network, after which all network traffic to reachable nodes is continued. Their performance measurements show acceptable detection and recovery times for a real-time network.

Another, similar, approach is being taken by Hoang et al. [39–41]. They also provide a switch with bandwidth reservation capabilities. But they do not build their node software layer on top of IP, but underneath it, on top of the Ethernet MAC layer. And they use Earliest Deadline First (EDF) [63] scheduling inside the switch, to make decisions as to which packets are forwarded first. This provides guarantees for both bit rates and delivery deadlines. The added real-time layer in the nodes enables applications to reserve real-time channels on the network, subject to a feasibility analysis. In this way they guarantee bandwidth for real-time traffic. Unfortunately the paper is not clear whether their approach supports a network consisting of multiple switches, as EtheReal does.

3.4 Other approaches

Court [25] created a deterministic Ethernet network by using a collision avoidance technique. He classifies collisions into two categories. The first type occurs when two or more nodes simultaneously check the bus, find it idle, and transmit a packet. The second type occurs when two or more nodes detect a busy network, wait until it is idle, and then start transmitting their packets. The second type of collision occurs much more frequently than the first type, and this property is used by Court to create a collision avoidance technique, by having each node access the network only at a specific time window, which is chosen relative to a busy/idle transition of the network. The first type of collision thus never occurs, and the second type is avoided by using a delay arbitration protocol. Each nodes waits for a period of time that is proportional to its priority. Any network activity will force the network to wait until the next busy/idle transition, but when the network is idle during the allocated time window for a node, it may transmit. This provides a distributed solution to making Ethernet deterministic, but a small hardware modification is necessary to implement this protocol.

Yavatkar et al. [114] changed the Ethernet protocol, by using a reservation-based scheme for real-time traffic. Their algorithm, called PCSMA (Predictable Carrier Sense Multiple Access), assumes all real-time traffic to be periodic. It requires these periodic source nodes to reserve transmission slots before they may begin transmitting. Non-real-time traffic still uses the normal CSMA/CD protocol. They claim that all periodic messages meet their deadlines, provided the channel utilization stays below 100%.

Instead of avoiding collisions, one can also modify the collision resolution algorithm. The CSMA/DCR (Carrier Sense Multiple Access with Deterministic Collision Resolution) protocol, proposed by Le Lann and Rivierre [60], replaces the probabilistic BEB algorithm with a deterministic binary tree search algorithm. When no collisions occur, the bus is accessed randomly, just like in Ethernet. Collisions are resolved by scheduling messages with a policy based on the node addresses. This policy may turn out to be inadequate for real-time systems, so a deadline-oriented approach has also been proposed: DOD/CSMA/CD (Deadline Oriented Deterministic Carrier Sense Multiple Access with Collision Detect) [60]. This algorithm is similar to CSMA/DCR, but it can handle real-time messages with arbitrary and strict deadlines, making it more suitable to a wider range of real-time applications.

Pérez-Turiel et al. [86] combined the PCSMA and CSMA/DCR protocols, creating their CSMA/PDCR (Carrier Sense Multiple Access with Priority Deterministic Collision Resolution) approach. Unlike PCSMA, CSMA/PDCR only comes to action at the occurrence of a collision, which is then resolved using a binary tree algorithm, like that proposed by Capetanakis [17].

Jan and Yeh [48] proposed a dynamic p_i-persistent CSMA/CD protocol, where p_i is the transmission probability of a packet with laxity i. The transmission probability depends on both the packet laxity and a time window. The size of the time window is chosen by each node initially as some reasonable value. In each slot each node halves the size of the window following a collision, not allowing it to drop below the duration of one slot, and doubles the size of the window otherwise, not allowing it to grow beyond the initial value. The transmission policy is used to implement the MLF policy. Their simulation results show their protocol to have almost the same performance as the window protocol of Zhao et al. [121], but their protocol is simpler than the window protocol.

Li [62] proposed a priority-driven protocol to be applied on top of the IEEE 802.3

network, enabling support for 256 priority levels. They divide the time axis in slots, with a length equal to the maximum end-to-end delay on the network. Nodes may only start message transmission at the start of a slot. They use the window protocol approach, so each node has a notion of a global priority queue of all messages waiting to be transmitted on the network. Priority contention is solved by broadcasting a so-called status packet, indicating the priority and length of the message a node wishes to transmit. The node with the highest priority may then transmit the message. Collisions between two or more status messages are resolved using a window protocol. The protocol can only have collisions for the status messages, not for the actual data packets.

Foh and Zukerman [32] approached the problem of bandwidth reservation by allowing ongoing packet transmissions to be interrupted. Their CSMA/RI (Carrier Sense Multiple Access with Reservations by Interruptions) protocol allows nodes, which are ready and have a high-priority packet in the queue, to interrupt the ongoing transmission of a packet from another node on the network. Only one such interruption may take place at any given time. When a high-priority packet is being sent which interrupted another one, this may not be interrupted again. Foh and Zukerman showed their CSMA/RI protocol to have better performance than the standard CSMA/CD protocol. Unfortunately they still cannot give any hard real-time guarantees.

Wang et al. [112] proposed a protocol, called RTCC (Real-Time Communication Control), which enables hard real-time communication on Ethernet. RTCC sits on top of Ethernet. One node is designated bus controller, all others are called remote terminals. A remote terminal has to receive permission from the controller before it may send anything. This centralized approach obviously has the disadvantage that failure in the master renders the entire network useless, unless some sort of recovery protocol is implemented that enables the slaves to choose a new master among them. Kerkes [55] developed an arbitration mechanism to create a real-time Ethernet network which is very similar.

Molle [72] proposed an alternative to the BEB protocol which eliminates the performance anomalies of this protocol. Furthermore, his BLAM (Binary Logarithmic Arbitration Method) does not introduce any new problems of its own. BLAM shows a large decrease in mean delay over BEB under moderate to high network load, and it is backwards compatible, as nodes employing both BLAM as BEB may coexist in the same network. Unfortunately, BLAM does not provide any timing guarantees, so it does not make Ethernet deterministic.

Pedreiras et al. [83, 84] proposed the FTT-Ethernet protocol, using the flexible time-triggered paradigm, developed for FTT-CAN [3, 4] in the same group. Unsatisfied with existing real-time Ethernet solutions, they needed a protocol that supports both event- and time-triggered communication, in hard and soft real-time and non-real-time types. Their protocol divides the time axis into an infinite amount of so-called elementary cycles. Within each cycle there can exist several windows, reserved for different types of messages. There are currently two windows, one for synchronous, or time-triggered traffic, and one for asynchronous, or event-triggered traffic. Synchronous traffic is subjected to admission control, thus guaranteeing its timeliness, and asynchronous traffic is based on a best-effort paradigm. Different from normal Ethernet is the addressing scheme. All packets are source addressed and broadcast to all nodes. Nodes interested in data from this particular source can process the data, the other nodes simply discard the packet. There is one master node which contains all the configuration and management data in the network, and is responsible for the admission control for synchronous network traffic.

4 Other networks

The 100VG-AnyLAN network [2, 113], originally developed by HP, uses a demand-priority scheme. It was standardized by the IEEE as IEEE 802.12 [45]. The network topology is a hierarchical structure of switches, to which the nodes are connected. The structure is hierarchical, because there is one switch which controls the entire network. All lower-level switches simply relay the requests to the master switch. All switches together act as one big logical switch.

The protocol operates roughly as follows. A node issues a request whenever it wants to transmit a packet. The master switch honours these requests in Round Robin (RR) fashion. Messages can have one of two priorities. All higher-priority messages are granted access before the lower-priority messages may take their turn. Higher-priority messages may also pre-empt lower-priority messages. A more detailed description is given by Barilovits and Kadambi [12].

Support for real-time traffic, where multimedia traffic is mentioned mostly, is given by 100VG-AnyLAN. The protocol guarantees access delay and throughput for streams using high-priority packets. Unfortunately, the protocol has disadvantages. Martini and Ottensmeyer [69] showed high sensitivity to the traffic characteristics of the high-priority packets. While high-priority traffic is honoured to the requested limits, low-priority traffic regularly suffers from very poor efficiency. The efficiency deteriorates drastically with decreasing frame size.

Another area where real-time communication has been necessary for many years is industry. E.g. production lines, where several automated procedures have to be carried out with precise timing differences, real-time networks have been in use for several years. These industrial networks are commonly known with the term fieldbuses. Examples are PROFIBUS [88, 104], FIP [31], and CAN [47]. PROFIBUS is based around a Timed Token Protocol (TTP) [102, 103]. FIP uses a bus master which grants publishers access to the bus, whose packets then get picked up by nodes designated to be a consumer of that producer's data. CAN uses a priority-driven scheme, where the device having the message with the highest priority is granted access to the bus. Unfortunately, these networks seldomly provide large amounts of bandwidth, as they are typically used to transmit control messages, which tend to be small. The CAN bus was developed for use inside automobiles, the others for use in factories. Tindell et al. [99] derived an idealized scheduling analysis for CAN, and showed, by studying two actual CAN interface controllers, that in practice the ideal situation is not always reached: one controller lives up to the model, the other does not, and needs an adapted model.

However, due mostly to cost, people are researching the use of Ethernet in such application areas. At the end of the last section FTT-Ethernet was already mentioned, as well as FTT-CAN, to enhance CAN with support for both flexible operation under guaranteed timeliness and joint support of both time-triggered and event-triggered traffic. Already in 1985 Shimokawa and Shiobara [96] developed a way of creating a real-time Ethernet for use in industry. They gave all nodes a unique sequence number. A master node periodically broadcasts a synchronization frame, after which each node may transmit, according to its sequence number in its assigned time slot. Then the entire process starts again with a new synchronization frame. The TEMPRA system by Pritty et al. [87] follows more or less the same principle, only they make sure the node emitting the synchronization is at the far end of an Ethernet bus network, so the synchronization frame travels the bus unidirectionally. But their protocol is unfair, in that nodes closer to the synchronization frame-emitting node will be given precedence.

Chen et al. [21] describe a protocol, based on a switched Ethernet network, which uses the native priority queueing support [43] available in certain switches. This enables nodes to provide Ethernet packets with a priority in the range $[0, 1 \ldots 7]$. Switches give precedence to higher-priority packets over lower-priority packets. They assign periodic real-time frames the highest priority (7), aperiodic real-time frames the next lower priority (6), and the other priorities are used for soft real-time and best-effort data. The periodic real-time frames are then scheduled using Rate Monotonic (RM).

5 Real-time support in the higher layers

Up to now real-time transmission policies for use in a Local Area Network (LAN) have been discussed. But, if it is necessary to have real-time communication between two or more LANs, with several networks of all kinds inbetween, these policies are not enough. Naturally, when a real-time connection is to be set up and whatever

QoS parameters that were agreed upon have to be kept, all involved networks have to be able to provide such guarantees. When one network in the chain does not provide QoS guarantees, the entire link cannot be guaranteed.

But it is not practical to set up a real-time connection by first making arrangements in the first network on the route, then with the second, with the third, and so on, until the final network, in which the destination node is situated, is reached. So it is desirable to use a single protocol, known to all internetworking nodes or gateways, that connect two networks together, with which a node can make appropriate QoS arrangements along the entire route.

Applications do not talk directly to the data link layer. Applications use the transport layer to provide them with network services. And this transport layer uses the network layer to talk to the data link layer. The network layer provides addressing, which is unique for every node, and preferably for every application on each node. Network layer addressing is also unique in the entire internetwork, whereas the addressing used in the data link layer needs only to be unique for LAN in which the node resides. The transport layer is responsible for providing concepts as channels, which can be used to provide guarantees that all packets will actually be delivered, in order and without errors. For real-time services the transport layer needs to be able to provide timeliness and throughput guarantees to the application, and it needs support from the underlying layers to be able to do so.

Andrews and Cyganski [10] developed a transport layer protocol, called XUDP (eXtended User Datagram Protocol), that provides an alternative, semi-reliable, method for transmitting real-time, multimedia data over an internet, without direct real-time support from the underlying layers. They claim that XUDP works on top of any IP network, which, in general, do not support any QoS parameters at all. It is semi-reliable, as data that is detected to be too late by the sending node, is thrown away to make room for following data that is on time. For multimedia data it is not too bad to throw away a frame now and then, as long as the latency remains constant. XUDP is not able to provide hard real-time guarantees, soft real-time is the best that can be provided.

Long et al. [64] developed a transport layer, called Swift, which can provide guarantees for a minimum throughput, a maximum delay, and a maximum jitter when using FDDI for the data link layer. The protocol provides streams which will be transmitted with a nearly constant data rate, for which messages will be kept in sequence. The application may choose whether the protocol should attempt to guarantee delivery, the requested data rate is always guaranteed. Swift is meant for multimedia data and cannot provide hard real-time guarantees. Also it is suitable only for use on FDDI networks.

Zheng and Shin [123] have given a mathematical basis for the problem of establishing real-time channels in packet-switched networks, where real-time schedulers in the switches make decisions which packets are transferred first over the links. Ferrari and Verma [30] obtained a solution under the assumption that the summation of the maximum packet transmission times over all real-time channels passing through a certain link is not larger than the minimum packet inter-arrival times of these channels. Without this assumption Kandlur et al. [54] established a sufficient condition to check the schedulability of channels. Zheng and Shin [123] provide a necessary and sufficient condition for the schedulability of a set of channels over a link and developed an efficient method for computing the minimum delay bound for each channel. Algorithms like these are necessary if gateways are to be built that can provide guarantees on internetwork links.

The Tenet group has built a cooperating set of network and transport layer protocols that provide real-time internetworking channels [11]. They provide two transport layer protocols, called RMTP (Real-time Message Transport Protocol) and CMTP (Continuous Media Transport Protocol), which work on top of the RTIP (Real-Time Internetwork Protocol) network layer protocol. These are managed by RCAP (Real-time Channel Administration Protocol). The RTCMP (Real-Time Control Message Protocol) is used to deal with failures. RTIP is connection-oriented, but does not provide reliable data delivery. RMTP provides a message-based service on top of RTIP, CMTP provides a periodic, time-driven service.

The ST-II protocol [101] was proposed as a network layer protocol to use for bandwidth reservation for packetized audio and video communication on the Internet. ST-II tries to forward data packets quickly and uses a small header containing only the necessary components to build a connection using virtual circuit addressing, similar to ATM addressing. It has been implemented by Delgrossi et al. [26], together with the Heidelberg multimedia transport protocol to be used on top of ST-II.

RSVP (ReSerVation Protocol) [14, 15, 115] is the protocol that is most in use in the Internet today. RSVP takes care only of the reservation of resources along a route from node to node, the gateways and routers inbetween reserve whatever is needed to accomodate the link, after which a protocol such as IP is used for transmitting the actual data. Therefore, on any network the directly connecting gateways or routers can make local reservations, using whatever facilities are available on that link, which can be anything of the protocols discussed earlier. When all reservations are made, the source can be notified using RSVP messages that the connection is set up, with bandwidth reservations in place, and that data transfer can begin.

This data transfer is mostly done using the transport-layer protocol RTP (Real-time Transport Protocol) [94]. RTP provides applications with synchronization and sequencing services, framing, multiplexing, and unreliable data transfer. It is independent of the lower-layer protocols (UDP, IP) and features a separate control protocol for QoS information and retransmission requests.

Inside the Internet routers employ different techniques to ensure QoS parameters on the traffic flowing through them. To schedule the packets on their outgoing interfaces algorithms such as First In First Out (FIFO), Priority Queueing, RR, or Weighted Fair Queueing (WFQ) [27, 82] are used. To regulate the rate at an interface routers commonly use the leaky bucket algorithm [79].

6 Suitability for the At-Home Anywhere project

The network that is to be used in the At-Home Anywhere project needs to be multi-purpose. It needs to support real-time streams, to be used for both entertainment (audio, video) and Command and Control (C&C). It also needs to support best-effort traffic. Furthermore, the network components must be inexpensive and, when possible, made of existing, unmodified hardware, which can also be incorporated in small devices. And the protocol must be able to deal with nodes coming and going at all times, i.e. it must support a form of plug-and-play.

The token-ring networks have a few drawbacks which make them unsuitable for use in the At-Home Anywhere project (@HA). IEEE 802.4 has the main drawback that it has been withdrawn as a standard by the IEEE. There is no hardware for sale any more that uses this network protocol.

IEEE 802.5 is still being used, but its use is declining. Personally, we have only seen it being used in office environments, and most of these are switching to Ethernet. There are two main reasons for this: performance and cost. Modern Ethernet networks provide up to six times the bandwidth of IEEE 802.5 networks, and cost only a fraction of them. As real-time capabilities are generally not important in an office environment, Ethernet usually more than meets its needs. These two reasons are also important in @HA: IEEE 802.5 is too expensive and rather limited in bandwidth if one has to transport several combined video and audio streams across the network.

FDDI is being used mostly on a larger scale. FDDI networks can span many kilometers and make use of high-bandwidth, expensive fiber optic cabling. Cost is the main reason why FDDI is not suitable for the @HA project.

Fieldbuses have the general disadvantage of not being built to support high-bandwidth traffic. Some also use source addressing, which is less useful for building a multi-purpose network. As these networks are mostly used for industrial applications or in expensive consumer products, such as automobiles, these devices generally are not small or inexpensive. These protocols also do not support plug-and-play techniques, as they are designed for networks with a static layout.

Ethernet is the most promising network technology. It is in wide use today, generally available and inexpensive to procure and install. It does, however, need some higher-layer protocol to make it suitable for real-time traffic. There are several protocols available, but some, such as the virtual time and window protocols, are very complex, making it hard to implement them in small devices that are low on processing and memory capacity. The master-slave protocols fail immediately when the master is taken out of the network, therefore making the incorporation of plug-and-play difficult. Protocols which put additional intelligence into the switches require firmware, possibly even hardware, changes in them. A protocol, such as RETHER, looks quite promising, but it uses a long switching time to enter the real-time mode for the first real-time stream, and its real-time scheduling strategy is less than optimal.

Within the @HA project we intend to work on a network protocol based on token passing. We will be using standard real-time scheduling algorithms to steer the token through the network, and in this way allocate bandwidth to the nodes according to the needs of the real-time streams originating from the nodes. Our current protocol needs a fully-connected network with broadcast and busy-detection capabilities. We intend to use a distributed scheduler, i.e. each node makes its own decisions about the global schedule, so we will also need some form of clock synchronization, to make sure that all nodes will make the same decisions in comparable situations.

For the implementation we will be using Ethernet, in its 10 Mbit and 100 Mbit variants, because of its large installment today and its cost. We will use IP on top of our protocol, which we named RTnet, to enable us to build applications which make use of the network. We will address network scheduling issues and clock synchronization issues. We will also make sure our protocol functions properly in a dynamic network configuration, where nodes may appear and disappear at any time.

Bibliography

[1] G. Agrawal, B. Chen, W. Zhao, and S. Davari. Guaranteeing synchronous message deadlines with the timed token medium access control protocol. *IEEE Transactions on Computers*, 43(3):327–339, Mar. 1994.

[2] A.R. Albrecht and P.A. Thaler. Introduction to 100VG-AnyLAN and the IEEE 802.12 local area network standard. *Hewlett-Packard Journal: technical information from the laboratories of Hewlett-Packard Company*, 46(4):6–12, Aug. 1995.

[3] L. Almeida, J.A.G. Fonseca, and P. Fonseca. A flexible time-triggered communication system based on the Controller Area Network: Experimental results. In *Proceedings of the Fieldbus Conference (FeT '99)*, pages 342–350, Magdeburg, Germany, Sept. 1999. ISBN 3-211-83394-3.

[4] L. Almeida, P. Pedreiras, and J.A.G. Fonseca. The FTT-CAN protocol: Why and how. *IEEE Transactions on Industrial Electronics*, 49(6):1189–1201, Dec. 2002.

[5] M. Alves, E. Tovar, and F. Vasques. Ethernet goes real-time: a survey on research and technological developments. Technical Report HURRAY-TR-0001, Polytechnic Institute of Porto — School of Engineering (ISEP-IPP), Jan. 2000.

[6] American National Standards Institute. *ANSI FDDI Standard — Media Access Control (MAC)*, 1987. ANSI Std. X3.139-1987, ISO 9314-2:1989.

[7] American National Standards Institute. *ANSI FDDI Standard — Physical Layer, Medium Dependent (PMD)*, 1990. ANSI Std. X3.166-1990, ISO 9314-3:1990.

[8] American National Standards Institute. *ANSI FDDI Standard — Station Management (SMT)*, 1994. ANSI Std. X3.229-1994, ISO DIS 9314-6*.

[9] American National Standards Institute. *ANSI FDDI Standard — Physical Layer Protocol (PHY)*, 1998. ANSI Std. X3.148-1998, ISO 9314-1:1989.

[10] M.J. Andrews and D. Cyganski. The XUDP protocol for timely multimedia transport. In *Proceedings 3rd Technical Conference on Telecommunications R&D*, Worcester, MA, USA, Nov. 1997.

[11] A. Banerjea, D. Ferrari, B.A. Mah, M. Moran, D.C. Verma, and H. Zhang. The Tenet real-time protocol suite: Design, implementation, and experiences. *IEEE/ACM Transactions on Networking*, 4(1):1–10, Feb. 1996.

[12] S. Barilovits and J. Kadambi. 100VG-AnyLAN: Network operation and real-time capabilities. In *Proceedings 19th annual IEEE conference on Local Computer Networks*, pages 120–128, Minneapolis, MN, USA, Oct. 1994. ISBN 0-8186-6680-3.

[13] D.R. Boggs, J.C. Mogul, and C.A. Kent. Measured capacity of an ethernet: Myths and reality. In *Symposium proceedings on Communications architectures and protocols (ACM SIGCOMM '88)*, pages 222–234, 1988. ISBN 0-89791-279-9.

[14] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. *Resource ReSerVation Protocol (RSVP) — Version 1 Functional Specification*. RFC Editor, Sept. 1997. RFC 2205.

[15] R. Braden, D. Estrin, S. Berson, S. Herzog, and D. Zappala. The design of the RSVP protocol. Technical report, USC/Information Sciences Institute, July 1996.

[16] F. Buzluca and E. Harmancı. Dynamic synchronous bandwidth allocation scheme for hard real-time communication in FDDI networks. *IEE Proceedings — Computers and Digital Techniques*, 148(1):15–21, Jan. 2001.

[17] J.I. Capetanakis. Generalized TDMA: The multi-accessing tree protocol. *IEEE Transactions on Communications*, 27(10):1476–1484, Oct. 1979.

[18] B. Chen, G. Agrawal, and W. Zhao. Optimal synchronous capacity allocation for hard real-time communications with the timed token protocol. In *Proceedings 13th IEEE Real-Time Systems Symposium*, pages 198–207, Phoenix, AZ, USA, Dec. 1992. IEEE Computer Society Press. ISBN 0-8186-3195-3.

[19] B. Chen and W. Zhao. Properties of the timed token protocol. Technical Report 92-038, Texas A&M University, Oct. 1992.

[20] C.H. Chen and L.N. Bhuyan. Design and analysis of multiple token ring networks. In *Proceedings 7th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '88)*, pages 477–486, New Orleans, LA, USA, Mar. 1988. ISBN 0-8186-0833-1.

[21] J. Chen, Z. Wang, and Y. Sun. Real-time capability analysis for switch industrial Ethernet traffic priority-based. In *Proceedings of the 2002 IEEE International Conference on Control Applications*, pages 525–529, Glasgow, UK, Sept. 2002. ISBN 0-7803-7386-3.

[22] T. Cheng, J.Y. Chung, and C.J. Georgiou. Enhancement of token bus protocols for multimedia applications. In *Digest of papers, IEEE COMPCON Spring*, pages 30–36, 1993.

[23] Y. Choo and C. Kim. Guaranteeing periodic communication services in a multiple access network using token with timer. Technical Report TP01ISA1109, The Instrumentation, Systems, and Automation Society, Research Triangle Park, NC, USA, 2001.

[24] M.A. Colvin and A.C. Weaver. Performance of single access classes on the IEEE 802.4 token bus. *IEEE Transactions on Communications*, 34(12):1253–1256, Dec. 1986.

[25] R. Court. Real-time Ethernet. *Computer Communications*, 15(3):198–201, Apr. 1992.

[26] L. Delgrossi, R.G. Herrtwich, and F.O. Hoffmann. An implementation of ST-II for the Heidelberg transport system. *Internetworking: Research and Experience*, 5(2):43–69, June 1994.

[27] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, 1(1):3–26, Sept. 1990.

[28] D. Dykeman and W. Bux. Analysis and tuning of the FDDI media access control protocol. *IEEE Journal on Selected Areas in Communications*, 6(6):997–1010, July 1988.

[29] M.N. El-Derini and M.R. El-Sakka. A CSMA protocol under a priority time constrained for real-time communication. In *Proceedings 2^{nd} IEEE Workshop on Future Trends of Distributed Computing Systems*, pages 128–134, Cairo, Egypt, Sept./Oct. 1990. IEEE Computer Society Press. ISBN 0-8186-2088-9.

[30] D. Ferrari and D.C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, Apr. 1990.

[31] FIP web site. `http://www.worldfip.org/`.

[32] C.H. Foh and M. Zukerman. CSMA with reservations by interruptions (CSMA/RI): A novel approach to reduce collisions in CSMA/CD. *IEEE Journal on Selected Areas in Communications*, 18(9):1572–1580, Sept. 2000.

[33] R.G. Gallager. Conflict resolution in random access broadcast networks. In *Proceedings of the AFOSR Workshop in Communication Theory and Applications*, pages 74–76, Provincetown, MA, USA, 1978.

[34] P.M. Gopal and J.W. Wong. Analysis of a hybrid token-CSMA/CD protocol for bus networks. *Computer Networks & ISDN Systems*, 9(2):131–141, Feb. 1985.

[35] R.M. Grow. A timed-token protocol for local area networks. In *Proceedings Electro '82, Token Access Protocols*, pages 17/3:1–7, May 1982.

[36] R.M. Grow. *Timed token protocol for local area networks*, July 1984. Patent US4459588.

[37] M. Hamdaoui and P. Ramanathan. Selection of timed token protocol parameters to guarantee message deadlines. *IEEE/ACM Transactions on Networking*, 3(3):340–351, June 1995.

[38] G. Held. *Token-ring networks: characteristics, operation, construction and management.* John Wiley & Sons, 1994. ISBN 0-471-94041-0.

[39] H. Hoang and M. Jonsson. Switched real-time Ethernet in industrial applications—asymmetric deadline partitioning scheme. In *Proceedings 2^{nd} International Workshop on Real-Time LANs in the Internet Age*, Porto, Portugal, July 2003. To appear.

[40] H. Hoang, M. Jonsson, U. Hagström, and A. Kallerdahl. Switched real-time Ethernet with earliest deadline first scheduling — protocols and traffic handling. In *Proceedings Workshop on Parallel and Distributed Real-Time Systems (WPDRTS 2002) in conjunction with International Parallel and Distributed Processing Symposium (IPDPS '02)*, Fort Lauderdale, FL, USA, Apr. 2002.

[41] H. Hoang, M. Jonsson, A. Larsson, R. Olsson, and C. Bergenhem. Deadline first scheduling in switched real-time Ethernet — deadline partitioning issues and software implementation experiments. In E. Tovar, T. Sauter, and L.M. Pinho, editors, *Proceedings 1st International Workshop on Real-Time LANs in the Internet Age*, pages 68–71, Vienna, Austria, June 2002. Instituto Politécnico do Porto. ISBN 972-8688-06-7.

[42] Institute of Electrical and Electronics Engineers. *IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 4: Token-Passing Bus Access Method and Physical Layer Specifications*, 1990. IEEE Std. 802.4-1990, ISO/IEC 8802-4:1990.

[43] Institute of Electrical and Electronics Engineers. *IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Networks — Common Specifications — Part 3: Media Access Control (MAC) Bridges*, 1998. IEEE Std. 802.1D-1998, ISO/IEC 15802-3:1998.

[44] Institute of Electrical and Electronics Engineers. *IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 5: Token Ring Access Method and Physical Layer Specifications*, 1998. IEEE Std. 802.5-1998, ISO/IEC 8802-5:1998E.

[45] Institute of Electrical and Electronics Engineers. *IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 12: Demand-Priority Access Method, Physical Layer and Repeater Specifications*, 1998. IEEE Std. 802.12-1998, ISO/IEC 8802-12:1998.

[46] Institute of Electrical and Electronics Engineers. *IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, 2002. IEEE Std. 802.3-2002.

[47] International Standards Organisation. *Road Vehicles — Interchange of Digital Information — Controller Area Network (CAN) for High-Speed Communication*, 1993. ISO 11898.

[48] R.H. Jan and Y.J. Yeh. CSMA/CD protocol for time-constrained communication on bus networks. *IEE Proceedings I — Communication, Speech and Vision*, 140(3):197–202, June 1993.

[49] A.P. Jayasumana. Throughput analysis of the IEEE 802.4 priority scheme. *IEEE Transactions on Communications*, 37(6):565–571, June 1989.

[50] A.P. Jayasumana and P.N. Werahera. Performance of fibre distributed data interface network for multiple classes of traffic. *IEE Proceedings E — Computers and Digital Techniques*, 137(5):401–408, Sept. 1990.

[51] M.J. Johnson. Reliability mechanisms of the FDDI high bandwidth token ring protocol. *Computer Networks & ISDN Systems*, 11(2):121–131, Feb. 1986.

[52] M.J. Johnson. Proof that timing requirements of the FDDI token ring protocol are satisfied. *IEEE Transactions on Communications*, 35(6):620–625, June 1987.

[53] S. Kamat, N. Malcolm, and W. Zhao. Performance evaluation of a bandwidth allocation scheme for guaranteeing synchronous messages with arbitrary deadlines in an FDDI network. In *Proceedings 14th IEEE Real-Time Systems Symposium*, pages 34–43, Raleigh-Durham, NC, USA, Dec. 1993. IEEE Computer Society Press. ISBN 0-8186-4480-X.

[54] D.D. Kandlur, K.G. Shin, and D. Ferrari. Real-time communication in multi-hop networks. In *Proceedings 11th International Conference on Distributed Computing Systems (ICDCS '91)*, pages 300–307, Arlington, TX, USA, May 1991. IEEE Computer Society Press.

[55] J. Kerkes. Real-time Ethernet. *Embedded Systems Programming*, 14(1):43–54, Jan. 2001.

[56] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft, and R. Zainlinger. Distributed fault-tolerant real-time systems: The Mars approach. *IEEE Micro*, 9(1):25–40, Jan./Feb. 1989.

[57] J.F. Kurose, M. Schwartz, and Y. Yemini. Multiple-access protocols and time-constrained communication. *ACM Computing Surveys*, 16(1):43–70, Mar. 1984.

[58] S.K. Kweon, K.G. Shin, and G. Workman. Achieving real-time communication over Ethernet with adaptive traffic smoothing. In *Proceedings 6th IEEE Real-Time Technology and Applications Symposium*, pages 90–100, Washington, DC, USA, May/June 2000. ISBN 0-7695-0713-1.

[59] S.K. Kweon, K.G. Shin, and Q. Zheng. Statistical real-time communication over Ethernet for manufacturing automation systems. In *Proceedings 5th IEEE Real-Time Technology and Applications Symposium*, pages 192–202, Vancouver, Canada, June 1999. ISBN 0-7695-0194-X.

[60] G. Le Lann and N. Rivierre. Real-time communications over broadcast networks: the CSMA-DCR and the DOD-CSMA-CD protocols. Research report 1863, INRIA, Mar. 1993.

[61] C.T. Lea and J.S. Meditch. A channel access protocol for integrated voice/data applications. *IEEE Journal on Selected Areas in Communications*, 5(6):939–947, July 1987.

[62] M. Li. A priority-based protocol for the 802.3 network. In J.A. de la Puente and M.G. Rodd, editors, *Proceedings of the 12th IFAC Workshop on Distributed Computer Control Systems*, pages 19–22, Toledo, Spain, Sept. 1994. ISBN 0-08-042237-3.

[63] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, Jan. 1973.

[64] D.D.E. Long, C. Osterbrock, and L.F. Cabrera. Providing performance guarantees in an FDDI network. In *Proceedings 13th International Conference on Distributed Computing Systems (ICDCS '93)*, pages 328–336, Pittsburgh, PA, USA, May 1993. IEEE Computer Society Press. ISBN 0-8186-3770-6.

[65] N. Malcolm, S. Kamat, and W. Zhao. Real-time communication in FDDI networks. *Real-Time Systems*, 10(1):75–107, Jan. 1996.

[66] N. Malcolm and W. Zhao. Guaranteeing synchronous messages with arbitrary deadline constraints in an FDDI network. In *Proceedings 18th annual IEEE conference on Local Computer Networks*, pages 186–195, Minneapolis, MN, USA, Sept. 1993. ISBN 0-8186-4510-5.

[67] N. Malcolm and W. Zhao. The timed-token protocol for real-time communications. *IEEE Computer*, 27(1):35–41, Jan. 1994.

[68] N. Malcolm and W. Zhao. Hard real-time communication in multiple-access networks. *Real-Time Systems*, 8(1):35–77, Jan. 1995.

[69] P. Martini and J. Ottensmeyer. Real-time communication in the demand-priority LAN, the effects on normal priority traffic. In *Proceedings 20th annual IEEE conference on Local Computer Networks*, pages 350–357, Minneapolis, MN, USA, Oct. 1995. ISBN 0-8186-7162-9.

[70] R.M. Metcalfe and D.R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, July 1976.

[71] M.L. Molle. Prioritized-virtual-time CSMA: Head-of-the-line priority classes without added overhead. *IEEE Transactions on Communications*, 39(6):915–927, June 1991.

[72] M.L. Molle. A new binary logarithmic arbitration method for Ethernet. Technical Report CSRI-298, Computer Systems Research Institute, University of Toronto, Toronto, Canada, July 1994.

[73] M.L. Molle and L. Kleinrock. Virtual time CSMA: Why two clocks are better than one. *IEEE Transactions on Communications*, 33(9):919–933, Sept. 1985.

[74] P. Montuschi, L. Ciminiera, and A. Valenzano. Time characteristics of IEEE 802.4 token bus protocol. *IEE Proceedings E — Computers and Digital Techniques*, 139(1):81–87, Jan. 1992.

[75] P. Montuschi, A. Valenzano, and L. Ciminiera. Selection of token holding times in timed-token protocols. *IEEE Transactions on Industrial Electronics*, 37(6):442–451, Dec. 1990.

[76] H. Moon, H.S. Park, S.C. Ahn, and W.H. Kwon. Performance degradation of the IEEE 802.4 token bus network in a noisy environment. *Computer Communications*, 21(6):547–557, May 1998.

[77] J.K.Y. Ng and J.W.S. Liu. Performance of multiple-ring networks for real-time communications. In *Proceedings 17th annual IEEE conference on Local Computer Networks*, pages 426–435, Minneapolis, MN, USA, Sept. 1992. ISBN 0-8186-3095-7.

[78] I.G. Niemegeers and C.A. Vissers. TWENTENET: a LAN with message priorities, design and performance considerations. In *Proceedings of the ACM SIGCOMM symposium on communications architectures and protocols*, pages 178–185, 1984. ISBN 0-89791-136-9.

[79] G. Niestegge. The 'leaky bucket' policing method in the ATM (Asynchronous Transfer Mode) network. *International Journal of Digital and Analog Communication Systems*, 3(2):187–197, Apr./May/June 1990.

[80] OMNeT^{++} web site. `http://www.omnetpp.org/`.

[81] J.W.M. Pang and F.A. Tobagi. Throughput analysis of a timer controlled token passing protocol under heavy load. *IEEE Transactions on Communications*, 37(7):694–702, July 1989.

[82] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[83] P. Pedreiras and L. Almeida. Flexibility, timeliness and efficiency over Ethernet. In E. Tovar, T. Sauter, and L.M. Pinho, editors, *Proceedings 1st International Workshop on Real-Time LANs in the Internet Age*, pages 53–56, Vienna, Austria, June 2002. Instituto Politécnico do Porto. ISBN 972-8688-06-7.

[84] P. Pedreiras, L. Almeida, and P. Gai. The FTT-Ethernet protocol: Merging flexibility, timeliness and efficiency. In *Proceedings 14th Euromicro Conference on Real-Time Systems*, pages 152–160, Vienna, Austria, June 2002. ISBN 0-7695-1665-3.

[85] P. Pedreiras, R. Leite, and L. Almeida. Characterizing the real-time behavior of prioritized switched-Ethernet. In *Proceedings 2nd International Workshop on Real-Time LANs in the Internet Age*, Porto, Portugal, July 2003. To appear.

[86] J. Pérez-Turiel, J.C. Fraile-Marinero, and J.R. Perán-González. CSMA/PDCR: A random access protocol without priority inversion. In *Proceedings IEEE IECON 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, volume 2, pages 910–915, Taipei, Taiwan, Aug. 1996. ISBN 0-7803-2776-4.

[87] D.W. Pritty, J.R. Malone, D.N. Smeed, S.K. Banerjee, and N.L. Lawrie. A real-time upgrade for Ethernet based factory networking. In *Proceedings IEEE IECON 21st International Conference on Industrial Electronics, Control, and Instrumentation*, volume 2, pages 1631–1637, Orlando, FL, USA, Nov. 1995. ISBN 0-7803-3027-7.

[88] PROFIBUS web site. `http://www.profibus.com/`.

[89] F.E. Ross. FDDI — a tutorial. *IEEE Communications Magazine*, 24(5):10–17, May 1986.

[90] F.E. Ross. Rings are 'round for good! *IEEE Network*, 1(1):31–38, Jan. 1987.

[91] F.E. Ross. An overview of FDDI: The fiber distributed data interface. *IEEE Journal on Selected Areas in Communications*, 7(7):1043–1051, Sept. 1989.

[92] S.F. Salian, A.Y.M. Shakaff, and R.B. Ahmad. Improvement of virtual-time CSMA protocol for distributed hard and soft real-time systems on the Ethernet. In *Proceedings 2002 Student Conference on Research and Development (SCOReD 2002)*, pages 128–131, Shah Alam, Malaysia, July 2002. IEEE. ISBN 0-7803-7565-3.

[93] S.S. Sathaye and J.K. Strosnider. Conventional and early token release scheduling models for the IEEE 802.5 token ring. *Real-Time Systems*, 7(1):5–32, July 1994.

[94] H. Schulzrinne, S.L. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications.* RFC Editor, Jan. 1996. RFC 1889.

[95] K.C. Sevcik and M.J. Johnson. Cycle time properties of the FDDI token ring protocol. *IEEE Transactions on Software Engineering*, 13(3):376–385, Mar. 1987.

[96] Y. Shimokawa and Y. Shiobara. Real-time Ethernet for industrial applications. In P.P. Fasang, editor, *Proceedings IEEE IECON 11th International Conference on Industrial Electronics, Control, and Instrumentation*, pages 829–834, San Francisco, CA, USA, Nov. 1985.

[97] N.C. Strole. The IBM token-ring network — a functional overview. *IEEE Network*, 1(1):23–30, Jan. 1987.

[98] J.K. Strosnider, T. Marchok, and J. Lehoczky. Advanced real-time scheduling using the IEEE 802.5 token ring. In *Proceedings 9th IEEE Real-Time Systems Symposium*, pages 42–52, Huntsville, AL, USA, Dec. 1988. IEEE Computer Society Press.

[99] K.W. Tindell, H. Hansson, and A.J. Wellings. Analysing real-time communications: Controller Area Network (CAN). In *Proceedings 15th IEEE Real-Time Systems Symposium*, pages 259–263, San Juan, Puerto Rico, Dec. 1994. IEEE Computer Society Press. ISBN 0-8186-6600-5.

[100] F.A. Tobagi. Carrier sense multiple access with message-based priority functions. *IEEE Transactions on Communications*, 30(1):185–200, Jan. 1982.

[101] C. Topolcic. *Experimental Internet Stream Protocol, Version 2 (ST-II).* RFC Editor, Oct. 1990. RFC 1190.

[102] E. Tovar and F. Vasques. Analysis of the worst-case real token rotation time in PROFIBUS networks. In *Proceedings of the Fieldbus Conference (FeT '99)*, pages 359–366, Magdeburg, Germany, Sept. 1999. ISBN 3-211-83394-3.

[103] E. Tovar and F. Vasques. Cycle time properties of the PROFIBUS timed-token protocol. *Computer Communications*, 22(13):1206–1216, Aug. 1999.

[104] E. Tovar and F. Vasques. Real-time fieldbus communications using Profibus networks. *IEEE Transactions on Industrial Electronics*, 46(6):1241–1251, Dec. 1999.

[105] J.M. Ulm. A timed token ring local area network and its performance characteristics. In *Proceedings 7th IEEE conference on Local Computer Networks*, pages 50–56, 1982.

[106] A. Valenzano, P. Montuschi, and L. Ciminiera. Some properties of timed token medium access protocols. *IEEE Transactions on Software Engineering*, 16(8):858–869, Aug. 1990.

[107] S. Varadarajan and T. Chiueh. EtheReal: A host-transparent real-time fast Ethernet switch. In *Proceedings 6th IEEE International Conference on Network Protocols*, pages 12–21, Austin, TX, USA, Oct. 1998.

[108] S. Varadarajan and T. Chiueh. Automatic fault detection and recovery in real time switched Ethernet networks. In *Proceedings 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99)*, volume 1, pages 161–169, New York, NY, USA, Mar. 1999.

[109] C. Venkatramani. *The Design, Implementation and Evaluation of RETHER: a Real-Time Ethernet Protocol.* PhD thesis, State University of New York at Stony Brook, Jan. 1997.

[110] C. Venkatramani and T. Chiueh. Supporting real-time traffic on Ethernet. In *Proceedings 15th IEEE Real-Time Systems Symposium*, pages 282–286, San Juan, Puerto Rico, Dec. 1994. IEEE Computer Society Press. ISBN 0-8186-6600-5.

[111] C. Venkatramani and T. Chiueh. Design, implementation and evaluation of a software based real-time Ethernet protocol. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication (ACM SIGCOMM '95)*, pages 27–37, Cambridge, MA, USA, Aug./Sept. 1995.

[112] Z.P. Wang, G.Z. Xiong, J. Luo, M.Z. Lai, and W. Zhou. A hard real-time communication control protocol based on the Ethernet. In *Proceedings 7th Australasian Conference on Parallel and Real-Time Systems (PART 2000)*, pages 161–170, Sydney, Australia, Nov. 2000. Springer-Verlag. ISBN 962-430-134-4.

[113] G. Watson, A. Albrecht, J. Curcio, D. Dove, S. Goody, J. Grinham, M.P. Spratt, and P.A. Thaler. The demand priority MAC protocol. *IEEE Network*, 9(1):28–34, Jan./Feb. 1995.

[114] R. Yavatkar, P. Pai, and R. Finkel. A reservation-based CSMA protocol for integrated manufacturing networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1247–1258, Aug. 1994.

[115] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource reservation protocol. *IEEE Network*, 7(5):8–18, Sept. 1993.

[116] S. Zhang and A. Burns. An optimal synchronous bandwidth allocation scheme for guaranteeing synchronous message deadlines with the timed-token MAC protocol. *IEEE/ACM Transactions on Networking*, 3(6):729–741, Dec. 1995.

[117] S. Zhang, A. Burns, and T.H. Cheng. Cycle-time properties of the timed token medium access control protocol. *IEEE Transactions on Computers*, 51(11):1362–1367, Nov. 2002.

[118] S. Zhang, A. Burns, and A.J. Wellings. An efficient and practical local synchronous bandwidth allocation scheme for the timed-token MAC protocol. In *Proceedings 15th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '96)*, volume 2, pages 920–927, San Francisco, CA, USA, Mar. 1996. ISBN 0-8186-7292-7.

[119] W. Zhao and K. Ramamritham. A virtual time CSMA protocol for hard real time communication. In *Proceedings 7th IEEE Real-Time Systems Symposium*, pages 120–127, New Orleans, LA, USA, Dec. 1986. IEEE Computer Society Press. ISBN 0-8186-0749-1.

[120] W. Zhao and K. Ramamritham. Virtual time CSMA protocols for hard real-time communication. *IEEE Transactions on Software Engineering*, 13(8):938–952, Aug. 1987.

[121] W. Zhao, J.A. Stankovic, and K. Ramamritham. A window protocol for transmission of time-constrained messages. *IEEE Transactions on Computers*, 39(9):1186–1203, Sept. 1990.

[122] Q. Zheng and K.G. Shin. Synchronous bandwidth allocation in FDDI networks. In *Proceedings first ACM international conference on Multimedia*, pages 31–38, Anaheim, CA, USA, Aug. 1993. ACM Press. ISBN 0-89791-596-8.

[123] Q. Zheng and K.G. Shin. On the ability of establishing real-time channels in point-to-point packet-switched networks. *IEEE Transactions on Communications*, 42(2/3/4):1096–1105, Feb./Mar./Apr. 1994.

[124] T. Znati. A deadline-driven window protocol for transmission of hard real-time traffic. In *Proceedings 10th annual IEEE Conference on Computers and Communications*, pages 667–673, Scottsdale, AZ, USA, Mar. 1991. ISBN 0-8186-2133-8.