Deterministic Service Guarantees in 802.12 Networks, Part I : The Single Hub Case

Peter Kim
Network Technology Department
HP Laboratories Bristol
HPL-97-147
December, 1997

LAN, quality of service, deterministic services

In this paper and its sequel [1] we study the problem of allocating resources in single hub and cascaded 802.12 networks. We show that the use of the 802.12 high priority mechanism when combined with admission control, allows the network to provide small, deterministic delay bounds in large, cascaded network topologies with potentially many hundreds of nodes. The allocation scheme proposed is based on a time frame concept that takes advantage of the properties of the Demand Priority medium access protocol to provide much tighter delay bounds than given by the time frame itself. The first part of the work is to analyse relevant network performance parameters and their dependencies. In the second part, we describe the scheduling model and define the admission control conditions used to provide deterministic service guarantees. Experimental results received with a UNIX kernel based implementation in a standard 802.12 test network confirm our theoretical results for network parameters, throughput and delay bounds.

In this paper, the single hub topology is analysed. In the sequel of this paper, the network parameters are derived for cascaded 802. 12 networks which allow the admission control conditions to be applied to the topologies.

Internal Accession Date Only

Deterministic Service Guarantees in 802.12 Networks, Part I: the Single Hub Case

Peter Kim

pk@hplb.hpl.hp.com

HP Technical Report HPL-97-147, April 1997.

Hewlett-Packard Laboratories, Filton Rd, Bristol, U.K.

Abstract

In this paper and its sequel [1] we study the problem of allocating resources in single hub and cascaded 802.12 networks. We show that the use of the 802.12 high priority mechanism when combined with admission control, allows the network to provide small, deterministic delay bounds in large, cascaded network topologies with potentially many hundreds of nodes. The allocation scheme proposed is based on a time frame concept that takes advantage of the properties of the Demand Priority medium access protocol to provide much tighter delay bounds than given by the time frame itself. The first part of the work is to analyse relevant network performance parameters and their dependencies. In the second part, we describe the scheduling model and define the admission control conditions used to provide deterministic service guarantees. Experimental results received with a UNIX kernel based implementation in a standard 802.12 test network confirm our theoretical results for network parameters, throughput and delay bounds.

In this paper, the single hub topology is analysed. In the sequel of this paper, the network parameters are derived for cascaded 802.12 networks which allow the admission control conditions to be applied to those topologies.

1 Introduction

The use of applications with a variety of performance constraints and the widening commercial use of the Internet are driving its migration to an Integrated Services Packet Network (ISPN) [2]. In contrast to the current Internet, which only provides the traditional best-effort service, the new architecture will additionally offer advanced services called Integrated Services. The differentiator of these new services is the quality of service and the diverse service commitments e.g. probabilistic or deterministic performance guarantees which are assured by the network. Quality of service will be required for supporting applications with stringent performance constraints like Internet telephony or distributed virtual reality over the Internet, but will also be useful for ensuring a certain minimum bandwidth for traditional data transfers over congested links.

The key characteristics of the new ISPN are the services offered, the scheduling algorithm applied in routers and switches, the admission control and the reservation setup mechanism. Much research has been performed on each of these areas. The Integrated Services first put forward as draft standards for the new Internet are the *guaranteed-* and the *controlled load* service [3], [4]. Advanced packet scheduling and admission control are used to ensure the service quality specified in these service definitions. A comparative study can be found in [5]. Admission control schemes for a controlled load service are presented in [6], [7]. The reservation setup mechanism requires a protocol which carries reservation requests through the internetwork. It ensures that resources are reserved on all links along the data path between the data source and the receiver. RSVP [8], [9] has been developed for performing this task at the network layer.

Within the ISPN architecture, the service guarantees offered to applications rely on supporting mechanisms at all intermediate layers of the transport system. Applications negotiate the service with the top most management layer e.g. RSVP and specify the service request and traffic characterisation. On each link along the data path, RSVP then requests the service on behalf of the application from the underlying link layer.

LAN technology is typically deployed at the leaves of the Internet where large bridged LANs often interconnect hundreds of users. In order to support end-to-end service guarantees through the Internet, mechanisms which enable these guarantees must also be introduced in switched/bridged LANs. The IETF Integrated Services over Specific Link Layers (ISSLL) working group was chartered with the purpose of exploring the mechanisms required for various link layer technologies. Reference [10] describes the framework for providing the functionality to support Integrated Services on shared and switched IEEE 802 LAN technologies.

There is no standard mechanism for providing service guarantees across existing LANs such as 802.3 Ethernet, 802.5 Token Ring, or 802.12 Demand Priority. This is because the access mechanisms of these technologies differ. Another factor to be considered is the bridged LAN topology which can imply shared, half-duplex- or full-duplex switched links. This is different to the wide-area which usually consists of routers and switches connected by point-to-point links. The packet scheduling and the admission control conditions will thus typically be technology specific,

sometimes even topology dependent, and must be defined separately for each LAN technology.

This paper and its sequel focus on defining the scheduling model and the admission control conditions required for providing deterministic service guarantees across 802.12 networks. Our work consists of two parts. It contains a detailed analysis of the worst-case network performance parameters for single-hop and cascaded topologies. The results from this analysis can be used as the basis for any advanced service to be built on top of the 802.12 high priority access mechanism in cascaded and bridged/switched 802.12 networks. We further define the admission control conditions required for supporting a guaranteed service across single-hub and cascaded networks.

In this paper, we will restrict our attention to the single hub network and leave the analysis of cascaded networks for the sequel. We also do not discuss the scheduling and the admission control conditions applied in bridged networks. This is left for further study.

The remainder of this paper is organized as follows. In section 2, we introduce 802.12 networks and the Demand Priority medium access method used. We then discuss performance parameters such as the available bandwidth and their dependencies. A detailed analysis of the worst-case per-packet overhead and the time it takes to interrupt the low priority service is performed in Appendix A.3 and A.4.

Section 3 first discusses design decisions which we made for our allocation system. We then describe the scheduling model and present the corresponding admission control conditions. The analytical proofs for the bandwidth and the delay bound test are given in Appendix A.1 and A.2. In section 4, we propose a simple time window mechanism that can be used to improve the resource utilization in the case that applications use variable packet sizes, but do not change their packetization process. Section 5 describes our implementation and the test network that was used to experimental confirm analytical results. In section 6, we present measurement results received for network parameters and the end-to-end delay. Resource utilization issues are also discussed. Our conclusions are presented in the sequel after we discussed the results for cascaded 802.12 topologies.

2 IEEE 802.12

IEEE 802.12 [11] is the standard for a shared 100Mbit/s LAN. A simple network consists of a single hub (repeater) and several nodes, each separately connected to the hub creating a star topology. To extend the size of the network, several hubs can be connected to each other. This is called cascading. The shared medium access is controlled by the Demand Priority protocol. Data are transmitted using either IEEE 802.3 or 802.5 frame formats. Several physical layers have been defined. In particular the standard supports Category 3 unshielded twisted pair (UTP) cable, which is the most widely used cabling. The standard also specifies the operation over shielded twisted pair (STP) and multimode fiber.

2.1 Demand Priority

The MAC protocol used in 802.12 is called Demand Priority. Its main characteristics are the support of two priority levels and the service order: data packets from all network nodes are served using a simple round-robin algorithm.

Whenever nodes wish to transmit a packet, they first signal a service request (or demand) to the hub. The request is labelled with either normal or high priority. The hub is continually scanning each of its attached ports and maintains two separate service lists: one for low priority and one for high priority requests. All high priority requests are served first. The hub acknowledges the request of the next node in its current round-robin cycle and grants the transmission of one packet. The selected node then starts sending its packet to the hub. As the hub receives the packet, it decodes the address information, selects the output port, and then only forwards the packet to its destination. This filtering is possible because the hub learned the MAC addresses of all nodes connected to it during a link training process, which is executed when the link to an end-node is setup. Multicast and broadcast frames are send to all nodes. The hub continues the process until the high priority list is empty and then carries on serving demands for normal priority service.

Whenever the hub receives a high priority request while its low priority service list is being served, it completes the processing of the current request before it begins to serve high priority requests. After processing all high priority requests, the hub continues to serve the normal priority list at the last position in the low priority round-robin cycle.

The service policy is unfair if different nodes use different packet sizes. The hub schedules packets according to a simple round robin scheme and does not consider the size of the packets transmitted. Further details and a comparison with the 100BaseT standard (IEEE 802.3u) can be found in [12], [13] and [11].

2.2 Performance Parameters and their Dependencies

The communication between end-nodes and the hub is synchronized by the exchange of link control signals. These are used to signal the local MAC status and to control the medium access. Each packet transmission on Demand Priority networks is associated with a fixed protocol and signalling overhead. This overhead has a significant impact on the performance if small sized data packets are used and, depending on the packet size and the network topology, substantially reduces the data throughput on the network.

To show this important dependency and how it affects the available bandwidth in the network, we have done the following experiment. In a single hub test network, we used 7 computers which we call Traffic Clients to generate multi-

cast traffic with a packet size ranging from 512 bits (64 bytes) to 12000 bits (1500 bytes). All traffic was multicast in conformance with the worst case packet transmission model described in Appendix A.3. Another computer which we call the Controller was used to: (1) control the packet sizes used by the Traffic Clients, and (2) to measure the throughput. All computers were HP 9000/700 workstations connected to a standard hub via 100m of Category 3 UTP cable. The computers used the HP-UX 9.05 operating system and standard EISA 802.12 interface cards. The throughput was measured by periodically reading the MIB counters [14] from the managed hub. This used SNMP *Get-Request* messages [15]. The incremental step of the packet size was 4 bytes, the measurement time interval was 30 seconds. Figure 1 shows the measurement result. One can observe that the achievable throughput varies for different packet sizes and becomes substantially smaller for data transmissions that only use small sized packets.

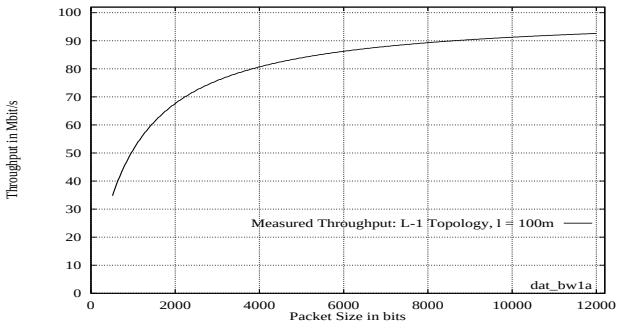

Figure 1. Measured Worst-Case available Bandwidth in a Single Hub 802.12 Network.

The data throughput will be further degraded in higher cascaded 802.12 topologies. This dependency had a strong impact on the design and the complexity of our allocation system and had to be considered in the admission control conditions. Building an efficient allocation system on top of the 802.12 high priority access mechanism thus first requires the computation of the available bandwidth in the network. The result of this computation defines the bandwidth limit up to which a resource allocator may allocate resources. This is not only essential to ensure that allocated resources are actually available on the network, and thus that delay bounds and buffer space requirements are met. More importantly, it enables the resource allocator to guarantee that a certain minimum bandwidth is always free for the best-effort service by accordingly restricting the access to the high priority service.

The maximum bandwidth that could theoretically be allocated while giving deterministic service guarantees depends on two network parameters: the worst-case Demand Priority per-packet signalling overhead and the worst-case time it takes to pre-empt the low priority data transmission (the low priority service interrupt time). Both parameters depend on the cascading level, the physical layer technology and the cable length.

The cascading level has a significant impact because of the increased signalling delay within larger topologies. This is discussed in the sequel. The physical layer can introduce an additional delay when operating in half-duplex mode. This is the case for data transmissions over UTP links. Since data are transmitted on all four pairs across such cables, no link control signals can be exchanged during that time. This has an impact on the low priority service interrupt time in the network and is described in detail in Appendix A.4. The delay is not introduced across STP or fiber optic links since these operate in dual-simplex mode and can exchange data and control signals at the same time.

The dependency of the available bandwidth from the cable length is caused by the propagation delay of control signals and data across the network. This will be significant for long fiber optic links which may have a cable length of up to 2 km. The cable length for UTP and STP links is restricted to 200 m by the standard.

To determine the worst-case per-packet overhead (D_{pp}), and the low priority service interrupt time (D_{it}), the Demand Priority link control signals and the packet transmission model on 802.12 networks must be analysed in great detail. This is done in the appendix for a single hub network. We focused on a UTP physical layer since this is most widely used. The numerical results for D_{pp} and D_{it} are shown in Table 5 and Table 7 in Appendix A.3 and A.4. They include the delay caused by the Demand Priority protocol and by passing data through the protocol stack. In section 6, we compare the measured throughput shown in Figure 1 and the computed allocation limit which was determined by using these results in our admission control conditions.

3 Scheduling Model and Admission Control Conditions

In this section we describe the interaction of the end-nodes with the medium access protocol and how this leads to the admission control conditions. We start by discussing general design decisions and system constraints. We then introduce the traffic characterization used throughout the theoretical analysis and describe the scheduling process. In the last part of this section, we define the admission control conditions.

3.1 Design Decisions and Constraints

Our resource allocation scheme provides a service with an absolute delay bound. This is called a *guaranteed* service [16]. We have first concentrated on this since we believed this to be the more challenging service. 802.12 further only offers two different priority levels. This restricts the number of advanced services that can simultaneously be imple-

mented to just one, since the low priority access mechanism is used for best-effort traffic. A guaranteed service has the advantage that it provides the highest service commitment and can therefore also be used to serve requests for other services e.g. a controlled load service, whereas the opposite case does not hold. In the following, we will call flows using the guaranteed service real-time flows.

The guaranteed service is built on top of the 802.12 high priority access mechanism. No changes to the existing LAN standard are required. Any use of the high priority mechanism is controlled by rate regulators on a per-flow basis at each node in the network. Rate regulation and the Demand Priority protocol thus define the order in which data packets from different nodes are transmitted.

The Demand Priority protocol and the significant per-packet overhead have a strong impact on the scheduling model and the admission control conditions. In contrast to other link technologies, in 802.12 networks, we can not assume that data held in output queues are served with a constant data rate, even though the physical link speed is constant. Instead, the data throughput will depend on the packet sizes used by all nodes in the shared network as could be observed in Figure 1. This provides two problems which we have to solve. The first is that our admission control conditions must consider this dependency and take the per-packet overhead into account. Without this, the admission control conditions would have either provided a low resource utilization or non-deterministic service guarantees. The second problem is that we need a mechanism to find the packet sizes which applications are using, since this enables us to compute the signalling overhead.

Our reservation scheme is based on a time frame concept. It was chosen since this enables us to bind the total packet overhead, provided that the packet sizes are known. A key problem is that in existing systems the link layer cannot negotiate the packet sizes with the upper layers. One could be extremely pessimistic and assume the use of minimum sized packet for all flows, but this reduces the allocatable bandwidth within a single hub network to about 35 Mbit/s. This further decreases in higher cascaded topologies.

Within this section we will assume that packet sizes are fixed or the link layer is able to negotiate them with the application. In section 4, we then propose a simple measurement based algorithm that can be used to find an approximation if the packet sizes are neither negotiatable nor fixed. This algorithm can only be applied for applications which do not change their packetization process over time. This was the case for the multimedia applications *nv, vic, vat, MMC* [17], [18] and the *OptiVision MPEG Communication System* [27] which we tested. Instead of measuring the packet size directly, the algorithm measures the maximum *number of packets* each flow sends in a time frame. This enables us to compute the total packet overhead, but also allows a flow to use a variety of different packet sizes, including minimum sized packets, as long as the number of packet overheads stays below a certain upper bound.

The packet overhead and the simple round-robin service policy differentiate our environment from that of point-to-point links connected e.g. to an ATM switch. In 802.12 networks, hubs are not able to identify and isolate single flows. The service data rate is variable and depends on the packet size used by all end-nodes on the shared single hub or cascaded network segment. This packet size may also be variable within each flow. Further, the queues are distributed and data packets from different hosts can not be scheduled in the order they arrived at the output queue. This makes the analysis of our system more complicated, and is the reason why existing solutions do not apply to our environment.

3.2 Traffic Characterization

To allocate resources for an application, the resource manager needs a traffic characterization that describes the traffic passed to the network by this application. In our analysis, we use the leaky bucket scheme as used in [19], [20]. A leaky bucket filter has two parameters: a token generation rate r and a bucket depth δ. Tokens are generated at rate r and stored in the token bucket. The bucket depth δ limits the maximum number of tokens that can be stored. Sending a packet consumes p tokens from the bucket, where p denotes the packet length in bits. If the bucket is empty or does not contain enough tokens then the packet is stored in a queue until sufficient tokens are available. The maximum size of the queue is bounded.

The leaky bucket enforces the amount of data which can leave the node in any time interval Δt. A traffic source i conforms to the (δ^i, r^i) characterisation if in any existing time interval Δt no more than $b^i(\Delta t)$ bits leave the leaky bucket, where $b^i(\Delta t) \le \delta^i + r^i \Delta t$ is the *traffic constraint function* [19] of source i.

3.3 Packet Scheduling Model

In 802.12 networks, each node maintains two link level output queues: one for normal priority traffic and one for traffic with quality constraints. In our system we add link level rate regulators to control the access to the high priority queue on a per-flow basis on each network node. The number of flows is restricted by admission control. Ill behaved *nodes* can be prevented from using high priority by network management control of the hub.

The link level rate regulators have several functions in our system. We use them (1) to protect the network service from ill behaved applications by controlling the amount of data passed into the network within a time frame, and (2) to limit the number of *data packets* which can leave the regulator within this time interval (packet regulator). If resources are not allocated at peak rate then (3) our rate regulators also smooth out traffic bursts before they can enter the network.

Functions (1) and (3) describe traditional functions of a rate regulator. Feature (2) was added in our design.

The structure of the system is shown in Figure 2. Data packets received from the overlying network layer are first classified. Normal priority data packets are immediately passed to the best-effort output queue. We will not consider them any further in our analysis since their transmission is isolated and pre-emptable. This will be shown in an experiment in section 6. High priority packets are either: (1) sent immediately when a sufficient number of token are available for this flow, (2) are stored in the flow's regulator-queue until they become eligible to send, or (3) are dropped if the regulator-queue has reached its maximum capacity. All output queues are then served in round-robin order as described in Section 2.1.

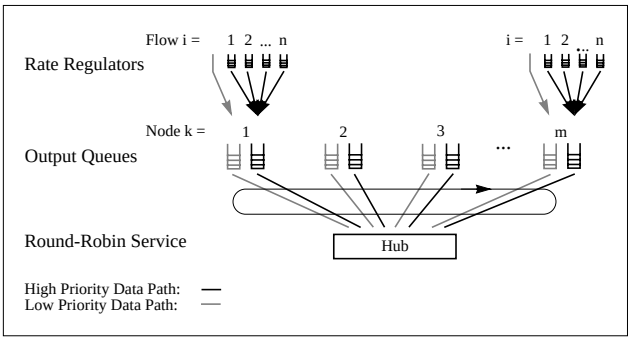

Figure 2. The Packet Scheduling Model.

The time frame concept underlying our resource allocation scheme requires that the total amount of data entering the high priority output queue on each node within a time frame is controlled. This is achieved using the rate regulators which sit above the high priority queue. The parameters of each rate regulator can be set so that they either correspond to the *peak* rate of a flow entering the regulator, or to the *average* rate. If they are set at the peak rate, the regulator does not introduce any delay because there is always a sufficient number of tokens available to pass a packet into the high priority queue. If they are set at the average rate, then the regulator smoothes out traffic peaks. This reduces the bandwidth to be allocated on the network and thus increases the resource utilization. However, delay is additionally introduced by holding packets in the regulator-queue. Smoothing at end-nodes is not a problem because host memory is not a scarce resource. For the sake of simplicity, we assume in the following, that resources are allocated at peak rate and no delay is introduced by the rate regulator.

To see how the time frame concept provides a delay bound, we first assume a time frame of length TF. For each flow i on node k, we define the packet count $pcnt_k^i$ which is the maximum number of packets this flow is allowed to pass into the high priority queue within any interval TF. If we assume that node k has n real-time flows, and sufficient resources are allocated such that any backlog is always cleared within TF, then the maximum number of packets in the high priority queue of node k is bounded by:

$$PCNT_k = \sum_{i=1}^{n} pcnt_k^i \qquad (3.3.1)$$

The simple round-robin service policy of the hub ensures that the $PCNT_k$ packets in the high priority queue at node k will be transmitted within the next $PCNT_k$ high priority round-robin cycles. Since the maximum number of all packets that become eligible within TF on all other nodes is known, the scheme can provide a deterministic delay bound for k. All bounds are in general inversely proportional to the allocated bandwidth: nodes with small reservations receive a smaller delay bound than nodes with large reservations. The delay bounds are further affected by the packet sizes used. The time frame provides the upper bound for all individual node delay bounds. Unlike the time frame in [21] or [22], our time frame is not the *minimum* delay bound that can be guaranteed by the system because we can exploit the round-robin service policy of the Demand Priority access method. For example, with a time frame of 40 ms, our scheme is still able to provide a delay bound of 5 ms or less for a node. Since the 802.12 standard only supports a single high priority level, our system can only provide a single output queueing delay bound per node k. This bound applies to all real-time flows on k. The end-to-end delay of different flows might however vary dependent on the additional delay that is introduced in the flow's rate regulator.

The computation of the packet count for flow i is straightforward when i uses data packets of fixed size. In this case we get:

$$pcnt^i = b^i(TF) / psize^i \qquad (3.3.2)$$

where $b^i(TF)$ is the maximum number of bits which can leave flow i's rate regulator within TF, and $psize^i$ is the packet size used. Equation 3.3.2 also provides a valid bound for a flow which uses variable sized packets, when $psize^i$ is set to the minimum packet size used by the flow (or when set to the minimum link packet size: 64 byte).

In order to provide deterministic service guarantees, all rate regulators must enforce the amount of data which enter the high priority queue in any time interval Δt. In a real implementation, we have to consider the fact that the clocks available to a regulator are granular. With a timer granularity T, where $0 < T \le \Delta t$, all packets which become eligible within the next time tick of length T are instantly granted by the regulator. This increases the burstiness of the traffic output. The traffic constraint function then becomes:

$$b^i(\Delta t) \le \delta^i + r^i \Delta t + r^i T \qquad (3.3.3)$$

This is used in our implementation. Note first, that $b^i(\Delta t)$ describes the traffic output of the rate regulator for flow i

and not the traffic that goes into the regulator. Note further, that we could have retained the traffic constraint function $b^i(\Delta t) \le \delta^i + r^i \Delta t$ and only transmitted packets after they became eligible. But this introduces a delay of T because of the timer granularity.

3.4 Admission Control Conditions

Admission control is the process which determines whether a new flow can be admitted to the network without impairing the service guarantees given to already admitted flows. In our system it consists of two parts: a bandwidth test and a delay bound test. The bandwidth test defined in Theorem 1 proves that the network has sufficient spare bandwidth to support the new request. The theorem checks that all data from all end-nodes can be transmitted within the time frame. The delay bound test is defined in Theorem 2. It takes advantage of the round-robin service policy, which allows us to calculate a delay bound for each individual node that can be considerably lower than the overall time frame. Note that in Theorem 1 and Theorem 2, we use the traffic constraint function $b^i(\Delta t)$ for fixed time intervals $\Delta t = TF$. b^i is equivalent to $b^i(TF)$. The time frames of different nodes are further not synchronized.

3.4.1 Bandwidth Test

Theorem 1 *Consider an 802.12 network with m nodes, where each node k has n real-time flows, which are already admitted. Assume a time frame of TF, a link speed of C_l and that the packet count for flow i on node k is $pcnt_k^i$. Further let P_{min} be the minimum link packet size and D_{pp}, D_{it} be the topology specific worst-case per-packet overhead and low priority service interrupt time, respectively. Assume also, that all flows are rate regulated and that the input traffic obeys the traffic constraint function b_k^i for all intervals TF. Sufficient bandwidth for the new flow ν with b^ν, is available if*

$$b^\nu \le \frac{TF - D_{it} - \frac{1}{C_l}\sum_{k=1}^{m}\sum_{i=1}^{n} b_k^i - \sum_{k=1}^{m}\sum_{i=1}^{n} pcnt_k^i \cdot D_{pp}}{\frac{1}{C_l} + \frac{D_{pp}}{P_{min}}} \qquad (3.4.1.1)$$

The proof can be found in Appendix A.1. The rather complicated structure of Theorem 1 is caused by considering the Demand Priority per-packet overhead. The importance of Theorem 1 is its capability to accurately provide the available link bandwidth for each packet size used (and implicitly for each set of packet counts $pcnt^i$).

Theorem 1 assumes that the new flow only uses minimum sized packets for the data transmission. For each already admitted flow i, the packet count $pcnt^i$ is used during admission control. It represents the number of packet overheads which this flow can consume within a time frame. Since the per-packet overhead is independent of the size of the data packet, flow i may for example use its credit to either send $pcnt^i$ minimum- or maximum sized packets. The sum of the packet counts of all flows is the maximum number of packets that are sent within the interval TF. It corresponds to a *minimum average* packet size P_{min_ave} over the time frame TF. The relation is given by:

$$P_{min\_ave} = \frac{\sum_{k=1}^{m}\sum_{i=1}^{n} b_k^i}{\sum_{k=1}^{m}\sum_{i=1}^{n} pcnt_k^i} \qquad (3.4.1.2)$$

We are able to calculate the total Demand Priority protocol overhead within a time frame since (1) the per-packet overhead is independent of the size of a data packet, and (2) we found an upper bound on the maximum number of packets transmitted in TF. Both is used in Theorem 1.

Note that all the bandwidth unallocated or unused by real-time flows is not wasted. It can be immediately used by the normal priority service.

3.4.2 Delay Bound Test

After testing that the network has sufficient spare resources to admit the new flow, the delay conditions need to be checked. Since the admission of a new flow can change the bounds for all nodes with reservations on the local segment, the verification must be carried out for all of them.

Theorem 2 *Consider an 802.12 network with m nodes, where each node k has n real time flows. Assume a link speed of C_l and that the packet count for flow i on node k is $pcnt_k^i$. Further let P_{max} be the maximum link packet size and D_{pp}, D_{it} be the topology specific worst-case per-packet overhead and low priority service interrupt time, respectively. If Theorem 1 applies, and if all flows are rate regulated and the input traffic passed into the network output queues obeys the traffic constraint function b_k^i for all intervals TF, then the queuing delay d_k for node k is bounded by:*

$$\sum_{j=1,\, j\neq k}^{m} \left(MIN\left(\sum_{i=1}^{n} pcnt_k^i, \sum_{i=1}^{n} \frac{b_j^i}{P_{max}} \right) \cdot \frac{P_{max}}{C_l} + MIN\left(\sum_{i=1}^{n} pcnt_k^i, \sum_{i=1}^{n} pcnt_j^i \right) \cdot D_{pp} \right) +$$

$$\frac{1}{C_l}\sum_{i=1}^{n} b_k^i + \sum_{i=1}^{n} pcnt_k^i \cdot D_{pp} + D_{it} \le d_k \le TF \qquad (3.4.2.1)$$

The proof of Theorem 2 can be found in Appendix A.2. Theorem 2 requires that Theorem 1 applies. Otherwise the condition $d_k \le TF$ is not true for all nodes k on the segment. In such a case, the output queue length and the delay could grow unboundedly since there is not sufficient bandwidth to clear the worst case backlog. If however Theorem 1 applies then data packets on all nodes k in the network will not be queued for longer intervals than the time frame TF.

The importance of Theorem 2 is that it allows the allocation system to provide smaller delay bounds than given by the time frame itself. This increases the flexibility of the allocation system and makes mechanisms for negotiating the time frame not stringent.

The delay bound d_k consists of the maximum packet residence time in the output queue, the link delay and the time it takes to interrupt the low priority packet transmission. The residence time depends on the bandwidth share allocated by node k, the total number of nodes that have resources reserved on the segment and their bandwidth share. The link delay is the time that is required for transmitting the data packets queued on node k across the network. Both components consider the corresponding Demand Priority per-packet overhead D_{pp}. The low priority service interrupt time D_{it} represents the difference between the computed minimum available network throughput and the allocation limit. This difference is not significant for single hub networks since D_{it} is small. It however has a larger impact in high cascaded topologies.

4 A Simple Measurement Algorithm

This section describes a simple time-window measurement algorithm. It is used to find a realistic upper bound on the total Demand Priority overhead to be considered for an active application in the admission control. Its development was motivated by the fact that in existing systems, the link layer cannot negotiate the packet size with upper layers or the application. Without such an algorithm, either (1) fixed sized data packet must be used, (2) new mechanisms for negotiating the packet count have to be introduced, or (3) the allocation must be carried out based on the minimum packet size used by the flow. For flows using variable sized packets, this is often the minimum link packet size.

Within this section, we describe the algorithm and discuss its conservativeness and adaptation rate. After reporting implementation issues in section 5, we present measurement results which show that for the applications we tested, the algorithm is able to find an accurate upper bound without impairing the guaranteed service quality.

4.1 The Algorithm

The algorithm is carried out on a per flow basis. It aims to find an upper bound for the number of packets sent by flow i within a time frame TF. This upper bound is denoted with $pcnt^i$. Two parameters are measured at the link layer. The measurement variable $scnt^i$ tracks the number of packets seen from flow i within the current time frame TF. Note that $scnt^i$ is measured after the flow is rate controlled. In $scnt^i_{TW}$, we keep a record of the maximum value observed for $scnt^i$ within the current measurement time window TW. We assume that $TF \ll TW$. The second parameter measured is flow i's data rate r^i_{TW}, averaged over the time window TW. The parameter MAX_PCNT denotes the worst case packet count for the flow. It corresponds to the case when the application only uses minimum sized packets for transmitting its data. MAX_PCNT^i is computed using the minimum link packet size P_{min} in equation 3.3.2.

The measurement process itself is illustrated in Figure 3. A realistic, measured sample pattern is shown later in Figure 12b. In the following, we describe how the measurements are used to estimate an upper bound $pcnt^i$ for flow i.

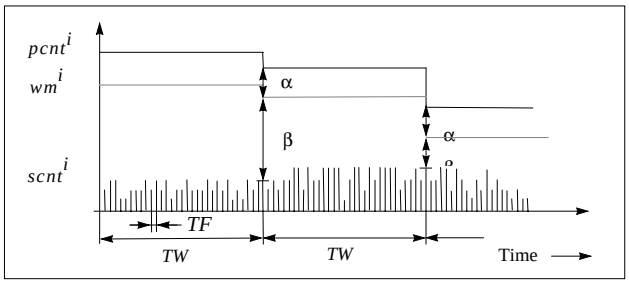

Figure 3. The Measurement Process for Flow *i*.

Initially, $pcnt^i$ is set to MAX_PCNT^i. The value can be changed at two occasions: at the end of each time window TW, and when an individual measurement for $scnt^i$ reaches the high watermark wm^i. The latter case is not illustrated in Figure 3. At the end of each time window, $pcnt^i$ is updated to reflect the measurements taken for the flow in the previous time interval TW. The new value $pcnt'^i$ is the sum of the maximum sample observed and two system parameters: α^i and β^i, which reflect the conservativeness and the level of uncertainty of the sample measured. $pcnt'^i$ however never exceeds MAX_PCNT^i since this is the maximum number of packets which this flow can possibly send in a time frame without violating its allocated data rate. For flow i follows:

$$pcnt'^i = MIN((scnt^i_{TW} + \alpha^i + \beta^i)\ ;\ MAX\_PCNT^i) \qquad (4.1.1)$$

The parameter α^i, where $0 \le \alpha^i \le MAX\_PCNT^i$, allows us to be more conservative by increasing $pcnt'^i$ to a value higher than the measured sample. It is set on a per-flow basis and could be controlled by the application. The parameter β^i reflects the level of uncertainty of the sample measured. It is proportional to the difference between allocated and measured data rate for this flow. β^i is small if the rate measured is close to the allocated rate. If the difference is larger, then β^i also increases. This ensures that the new value $pcnt'^i$ is not decreased when e.g. the data source is switched off or the application temporary generates significant less data than allocated. For flow i we have:

$$\beta^i = \frac{(r^i_{alloc} - r^i_{TW}) \cdot (TF + T)}{P_{min}} + 1 \qquad (4.1.2)$$

where r^i_{alloc} and r^i_{TW} are the allocated and the measured data rate for flow i, respectively. The parameter T is the

timer granularity of the rate regulator. It can be neglected for the case that $TW \gg TF \gg T$ holds. The computation of β is very conservative since we assume the use of minimum sized packets for the data rate unused by flow i. A less conservative approach might instead use an application specific value larger than P_{min}. As illustrated in Figure 3, each $pcnt^i$ has a corresponding high watermark wm^i. For a flow i, the relation is:

$$wm^i = pcnt^i - \alpha^i \quad (4.1.3)$$

Whenever an individual measurement for $scnt^i$ reaches the high watermark wm^i and the existing bound $pcnt^i$ is smaller than MAX_PCNT^i then the present estimation is wrong and we immediately update $pcnt^i$ to be κ times the existing value. The new value $pcnt'^i$ can again not exceed MAX_PCNT^i. Formally, we have:

$$pcnt'^i = MIN((\kappa \cdot pcnt^i);\ MAX\_PCNT^i) \quad (4.1.4)$$

where $pcnt'^i$ and $pcnt^i$ are the new and the old packet count, respectively. The algorithm can be summarized as follows:

1. At the beginning of the measurement process for flow i, set $pcnt^i$ to MAX_PCNT^i.

2. In $scnt^i$, measure the number of packets seen from i within the current time frame TF. In $scnt^i_{TW}$, keep a record of the maximum value observed for $scnt^i$ within the current time window TW. Further measure the data rate r^i_{TW} for the flow and average it over TW.

3. At the end of each time window TW, use equation 4.1.1 and 4.1.2 to compute the new value $pcnt'^i$. If required, replace the existing $pcnt^i$ with the new value and compute the high watermark wm^i using equation 4.1.3.

4. Whenever an individual measurement for $scnt^i$ reaches the high watermark wm^i and $pcnt^i < MAX\_PCNT^i$ then use equation 4.1.4 to compute the new packet count $pcnt'^i$. Update the existing $pcnt^i$ and compute the corresponding high watermark wm^i using equation 4.1.3.

4.2 Admission Control and Service Issues

If the packet count estimation only relies on measured data then any new flow is initially admitted based on the assumption that it will only use minimum sized data packets. Then as the flow starts, the algorithm measures the maximum number of packets used by the flow per time frame and takes a pessimistic maximum higher than the observed value.

The adaptation rate of the algorithm depends on (1) the length of the time window TW and (2) the difference between the allocated bandwidth and the bandwidth actually used by the application. A smaller time window increases the sensitivity of the algorithm since the packet counts are more frequently updated. It however also reduces the averaging interval used to compute the rate parameter r_{TW}, which causes a less conservative uncertainty factor β. If an application only uses a small percentage of the resources allocated then the parameter β ensures that the packet count is not decreased. The application might have stopped the transmission or had temporarily reduced its data output because of e.g. the specific characteristics of the data encoded in the video encoder. In such a case, the algorithm might not be able to find a close approximation within TW since it is uncertain whether the samples observed during that interval actually reflect the characteristic of the packetization process.

The conservativeness of the measurement process is controlled by the length of the time window TW. It could be as pessimistic as required at the expense of utilization. The worst case is an infinite time window which assumes that all data is sent with minimum sized packets as for new flows. This is very pessimistic, especially for realistic flows with a high data rate.

The algorithm relies on the property that the packetization process does not change over time. With the packetization process, we mean the algorithm used to break data e.g. a video frame into single data packets. Video frames of variable length might for example be fragmented by breaking each of them into a number of 1024 byte packets plus one variable sized packet which contains the rest of the frame.

If however the packetization process changes over time and the packet sizes become substantially decreased, then the packet counter $scnt^i$ will hit the high watermark wm^i. This triggers an immediate update of the estimated bound. Note that increasing $pcnt^i$ implies allocating resources on the network. Whenever the high watermark is reached then the flow however can still send α packets within the present time frame TF before a service violation occurs.

We believe that the measurement aspect does not conflict with the requirements of a guaranteed service, because we only apply the algorithm for applications with a constant packetization process. Whenever a service with less stringent commitments is requested e.g. a controlled load service, then the algorithm might also be used for applications which do change their packetization process.

The important advantage of using a measurement based approach is that it can substantially improve the efficiency of the allocation scheme, but does not require mechanisms for negotiating the packet count with upper layers. The disadvantage is that whenever deterministic guarantees are requested, the algorithm can only be used for applications which do not change their packetization process over time. The approach also has a slow adaptation rate which might cause the rejection of a reservation request even though, in reality, sufficient resources are available on the network.

5 Implementation Issues

We implemented and tested our resource allocation scheme in a 802.12 test network which consisted of standard hubs and HP 9000/700 workstations. This section briefly reports some of the design decisions we made and some of the problems we encountered during the implementation. All workstations used the HP-UX 9.05 operating system, standard EISA 802.12 interface cards and were connected to the hubs via Category 3 UTP links.

The rate controller and the classifier are implemented in the device driver of the 802.12 LAN adapter card. The link level signalling and the bandwidth management was performed by the LLRMP protocol [23], [24]. The resource allocation scheme was installed on all workstations that used the 802.12 high priority service. Network nodes that only use the best-effort service do not have to be updated.

5.1 Signalling and Resource Management

The LLRMP is a simple link level signalling protocol that is used to carry the reservation request and the traffic characterisation through shared and switched LANs. The protocol can support a distributed resource management, installs soft-states in end-nodes and bridges, and allows nodes to dynamically change their reservations. The latter property is used by end-nodes to update the resource information e.g. the packet count $pcnt$, which is held for them at the resource arbiter or at other end-nodes. We refer to [24] for any protocol details and the relationship of the LLRMP to the network layer resource management e.g. RSVP.

The host part of the LLRMP is implemented in a user space demon. This demon performs the LLRMP control message processing, the admission control and the time window measurement algorithm. A user interface allows access to the resource data base. The demon runs on top of the 802.12 LAN driver using the Link Level Access (LLA) interface. The LLA is a generalized *ioctl* based interface which provides basic low level access to device drivers in the HP-UX kernel. The LLRMP demon uses this interface for sending and receiving control messages and to control the rate regulators and the packet classifier in the kernel. Application data uses the normal path through the transport and network protocol stack.

We extended the LLA functionality to support asynchronous event notifications and to control the classifier and the rate regulators in the kernel. Asynchronous events are implemented using signals. The control mechanisms for rate regulator and scheduler are based on extended *ioctl* functionality.

The LLRMP protocol was implemented as a user space demon for reasons of simplicity. Only functionality in the data path, like the classifier and the rate regulators were kept in the kernel. However separating these mechanisms also caused a difficulty: context information is basically maintained twice: once in the demon and once in the kernel. Mechanisms were needed to keep both data bases consistent, so an asynchronous event notification mechanism was implemented. Measurement information is collected in the kernel, but all actions are controlled by the user space demon.

5.2 Classifier and Rate Regulator

Data packets are classified in the LAN device driver using the filter information provided by the LLRMP demon. The filter may specify a single or a combination of parameters in the link-level-, the network-, or the transport protocol header of the data packet. The classification can thus e.g. only be based on the MAC multicast destination address, when these addresses are uniquely assigned within the LAN, or can use higher level information like the IP source address and the UDP source port number.

Each rate regulator is able to support the time window algorithm described in the previous section. It counts the number of packets passed into the output queue in each time frame TF and measures the data rate generated by the application over the time window TW. All statistics collected in the kernel are periodically passed to the LLRMP demon which controls the parameter settings for the classifier and all link level rate regulators in the kernel.

Rate regulators also limit the number of packets which can leave the regulator in a single TF interval. The limit is defined by the packet count $pcnt$. If a flow sends more data packets than allowed, then any surplus packets become delayed into the next time frame. This property ensures that the service of other flows is not violated when an application e.g. by mistake passes a different traffic pattern to the network than negotiated.

5.3 Timer Issues

Our reservation scheme assumes time frames TF of $10 - 40\ \mathrm{ms}$ in order to keep the delay bounds low for nodes with large bandwidth requirements e.g. bridges. From Theorem 1 and equation 3.3.3, it follows that only $TF \gg T$, where T is the timer granularity, ensures an efficient use of resources. If the time frame and the timer granularity are of the same order of magnitude, then the result is a poor bandwidth utilization: e.g. for $TF = T = 10ms$, just 50% of the available resources can be reserved for data traffic. The rest must be left unallocated in order to ensure that worst case guarantees are met.

Most operating systems on existing workstations however only provide a timer granularity T of 10 ms. We solved this problem in our prototype by changing the timer granularity used on the test workstations. We implemented a second, fast timer in the HP-UX kernel, which is able to provide granularities of up to 100 µs on a 75 MHz machine. The function of the operating system was not affected since all OS routines are served at their usual times. Only kernel res-

ident modules e.g. LAN device drivers can register for the fast timer and receive service at low kernel priority.

Efficiency and timer granularity are not linearly related. The gain increases slower for smaller T. Throughout the experiments, we used a timer granularity of 1 ms which seems to be a good compromise between efficiency and processing overhead. In the future, a high granularity timer on the LAN adapter card would be an appropriate solution.

6 Measurement Results

In this section we present and discuss experimental results which we received for the throughput, the delay, the time window algorithm and the resource utilization. The results were collected using the implementation and the test network described in the previous section. All workstation were also connected to the site Ethernet and had the usual background processes running.

6.1 Throughput

In our first experiment we measured the maximum throughput on a 802.12 network versus the packet size used for data transmission. This was to experimentally prove Theorem 1 defined in section 3.4. The experiment itself was already described in section 2 to motivate the design decisions we made. In contrast to Figure 1, Figure 4 additionally shows the theoretical minimum network throughput and the allocation limit, both computed from Theorem 1.

The minimum network throughput was computed assuming: (1) there is only one active flow, (2) a time frame of $TF = 20\ \text{ms}$, (3) a single hub topology with 100 m UTP cabling represented in a per-packet overhead of $D_{pp} = 10.109\ \mu\text{s}$, and (4) a low priority service interrupt time of $D_{it} = 0$. The allocation limit differs from the theoretical throughput such that the computation additionally considered the interrupt time for this topology, where $D_{it} = 261.92\ \mu\text{s}$. The computation of both graphs assumed a non-bursty flow and a timer granularity of $T = 0$ to show the accuracy of the admission control.

In Figue_5, one can observe that the measured throughput is always higher than the theoretical throughput computed with Theorem 1. This is important since the computed throughput is the basis for the allocation limit. The difference between the theoretical throughput and the allocation limit thus reflects the minimum capacity that is guaranteed to be available for the low priority service. Some network resources must always be left unallocated since these are required to pre-empt the low priority service. Figure 4 shows the worst case for this and thus the maximum allocation limit. If for example all real-time flows had a minimum average packet size of 512 byte (4096 bit) or more, then bandwidth up to about 79 Mbit/s could theoretically be allocated. The actual available bandwidth however is guaranteed to be slightly higher, which is necessary for providing deterministic service guarantees. Figure 4 also shows that the theoretical and the measured result match closely. This demonstrates the accuracy of the model and of the results computed in Appendix A.3. Resources could potentially be allocated almost up to the actually available network capacity.

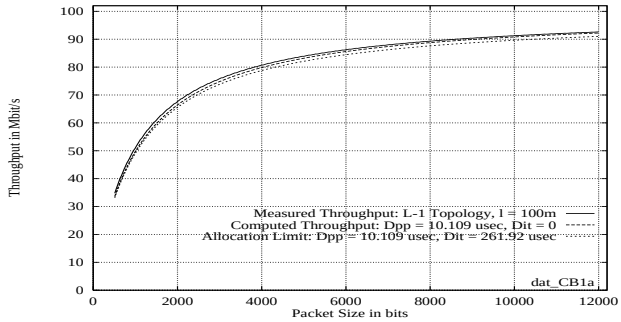

Figure 4. Comparison: Measured Throughput and Computed Allocation Limit in a Single Hub 802.12 Network using 100 m UTP Cabling.

Since the maximum supported UTP cable length for 802.12 networks is 200 m, we also measured the maximum throughput in such a topology. The results shown in Figure 5 are in general similar to the results in Figure 4, except that the throughput and the allocation limit for all packet sizes are decreased by a very small constant offset.

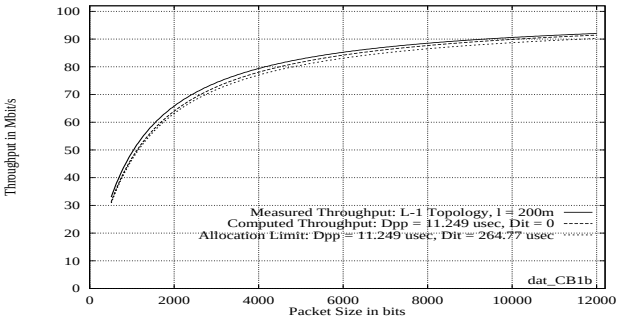

Figure 5. Comparison: Measured Throughput and Computed Allocation Limit in a Single Hub 802.12 Network using 200 m UTP Cabling.

The comparison shows that, despite the signalling overhead, the cable length does not have a significant impact on the worst case network performance when UTP cabling is used. This will be different for fiber optic links because of the long cable length supported for this type of physical layer. The results in Figure 5 were achieved using the same setup as in the previous experiment, except a different cable length. The allocation limit was computed using Theorem 1 with a packet overhead of $D_{pp} = 11.249\ \mu\text{s}$ and an interrupt time of $D_{it} = 264.77\ \mu\text{s}$.

The measured results in Figure 4 and Figure 5 are independent of the number of Traffic Clients used, as long as the Clients can saturate the network for all packet sizes. We

observed the same results as shown in Figure 4 in a configuration with 3 Traffic Clients and one Controller.

6.2 Delay Measurements

In the following experiments, we measured the link level end-to-end delay for data packets using the high and normal priority service. These experiments were carried out to: (1) show the isolation capabilities of 802.12, (2) to experimentally confirm the theoretical results achieved in Appendix A.4 for the worst case low priority service interrupt time, (3) to measure the end-to-end delay in a setup with several high priority data sources, and (4) to experimental determine the operating system overhead which is caused by the DMA-, the interrupt process and the context switch.

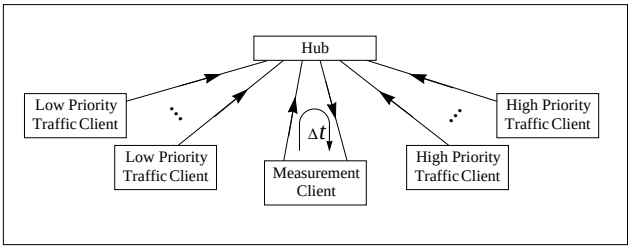

Figure 6. Setup for Measuring End-to-End Delay

Figure 6 illustrates the setup we used. All measurements were taken by a computer which we call the Measurement Client. It had two 802.12 LAN adapter cards, each of them was connected via a separate UTP cable to the hub. One interface was exclusively used for sending data, the second one was used for receiving. All data packets generated by the Measurement Client were addressed to a pre-defined multicast group which was joined with the receive interface. By using the same computer for sending and receiving test packets, we could use the same clock for determining the start and finish time of each measurement. This avoided timing discrepancies that would have occurred if we had used two separate computers. The time was measured using PA-RISC register CR16 [26], which provides a 10 ns tick on a 100 MHz HP 700 workstation. This ensured a high accuracy of the time-stamps. The measured delay Δt is the link layer end-to-end delay. It includes the time for transferring the packet from memory to the adapter card and back again to memory, as well as the relevant operating system overhead. Timing inaccuracies were minimized by ensuring that the workstation encountered no other interrupt e.g. from the Ethernet adapter between sending a test packet and receiving it. Several other computers were used in the different experiments to impose high- and low priority cross traffic. We called these computers High- and Low Priority Traffic Clients, respectively. All packets generated had a length of 1500 bytes to show the worst case effect.

In our first experiment, we measured the end-to-end delay (Δt) for a single high priority data source. The high priority traffic was generated by the Measurement Client. It sent packets at a low mean rate - about 0.56 Mbit/s - corresponding to constant rate compressed video. The experiment further included 10 Low Priority Traffic Clients which imposed low priority traffic at a total load ranging from 0 to 100 Mbit/s. All cross traffic was unicast and rate regulated. Note here that our rate regulators can also regulate normal priority traffic. This was used in this experiment. The measurement interval for each sample was 1 minute which corresponds to about 3000 packets transmitted by the Measurement Client. The incremental step of the low priority network load was 500kbit/s. In contrast to the setup in Figure 6, we did not use High Priority Traffic Clients in this experiment.

Figure 7 shows the results for the maximum-, average- and minimum end-to-end delay measured. The minimum delay is about 300 µs. This consists of 145 µs required for DMA-ing the packet (twice) and flushing the cache, about 25 µs of context switching, and about 130 µs of packet transmission and protocol overhead D_{pp}.

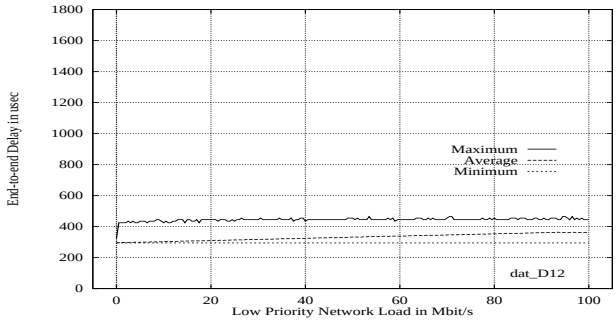

Figure 7. End-to-end Delay using the High Priority Service and Unicast Cross Traffic.

It can be further observed that the maximum delay is bounded and does not increase with higher network loads. This confirms that high priority traffic is isolated on the network. The maximum delay is the minimum plus about 130 µs. This corresponds to one maximum size low priority packet and is the time required in this setup to pre-empt the low priority service. The delay occurs when the a low priority packet just starts before the high priority request was signalled to the hub. There is no further offset because the Measurement Client did not receive any of the cross traffic (since this was unicast). In our experiment, we measured a difference between minimum and maximum delay of about 160 µs. We explain the 30 µs variation because of interference with other DMA operations e.g. a packet output on the Ethernet. A DMA might just been set up when we started the measurement for a test packet.

Figure 8 shows the maximum delay measured in a network with several High- and Low Priority Traffic Clients, as illustrated in Figure 6. Between zero and three computers were used to impose high priority traffic. Each High Priority Client generated data at a rate of 20 Mbit/s. This used a sim-

ple traffic generator. Four other computers were used to impose low priority traffic at a total load ranging from zero to 100 Mbit/s. All cross traffic was unicast and used a packet size of 1500 bytes. The Measurement Client was the same as used in the previous experiment.

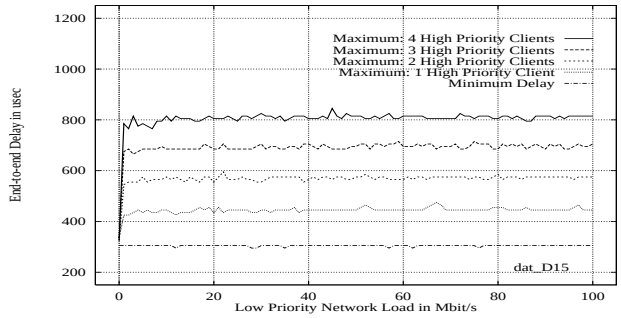

Figure 8. End-to-End Delay in a Setup with several High Priority Traffic Clients.

Figure 8 shows the maximum end-to-end delay (Δt) observed by the Measurement Client while varying the number of High Priority Clients. We can observe that Δt increases with each new High Priority Client by about 130 µs. The maximum delay is encountered when the low priority service is pre-empted and the Measurement Client is the last high-priority node to be served in the round-robin sequence.

In our next test, we repeated the experiment that led to the results in Figure 7. We carried out the same test, which included a setup with 10 Low Priority Clients and one Measurement Client. All Clients now however used the low priority service. This was to observe the average delay versus the total load in the single hub test network. The result in Figure 9 shows that the average delay which is observed by the Measurement Client keeps very small, even when the network load grows up to 80 Mbit/s. The maximum average value of $d_{ave} = (d_{max} - d_{min})/2$ was only measured for network traffic close to the throughput limit of 92.6 Mbit/s.

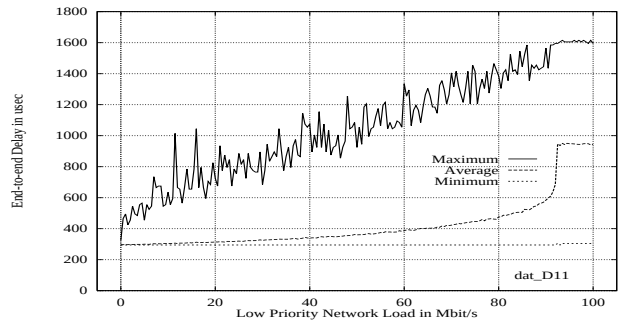

Figure 9. End-to-end Delay in a Setup that only uses the Low Priority Service.

The maximum delay in Figure 9 increases with rising network load and reaches an absolute maximum of 1615 µs. In this test setup, the maximum delay is directly proportional to the number of Low Priority Traffic Clients in the network. The theoretical maximum is 1600 µs assuming no high priority traffic, a minimum delay of 300 µs and 10 network nodes generating cross traffic with data packets of maximum length. The worst case occurs when the data packet from the Measurement Client is delayed by a data packet from each Traffic Client in the network. With longer measurement times of up to 10 min. for each sample, we found that the maximum delay of 1615 µs is also reached for smaller network loads. This is because longer measurement times increase the probability for having a packet transmission with worst case delay in the sample.

It is straightforward to see that the results in Figure 9 are only valid in the absence of any high priority traffic on the network. In a setup where high and low priority data packets are transmitted, low priority packets become delayed and will be served according to the mechanisms described in section 2.1. The delay distribution in Figure 9 for a load of 80 Mbit/s is shown in Figure 10. It shows a long tail distribution with a maximum of about 1400 µs.

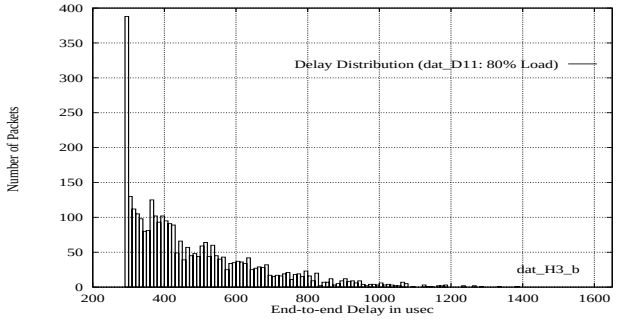

Figure 10. Delay Distribution in Figure 9 for 80Mbit/s Load.

In all our previous experiments within this section, all cross traffic used the unicast addressing mechanism. The data packets were sent to a single node that was not further involved in the measurements. This ensured that all other network nodes could signal their service request to the hub immediately after DMA-ing the packet onto the LAN adapter card. In more realistic environments however, when multicast and unicast are used and data packets are simultaneously sent and received, the request-signalling can be blocked by e.g. the transmission of a multicast packet. This can lead to an increased overhead whenever UTP cabling is used, as discussed in Appendix A.4. We measured this overhead in order to confirm the worst-case model used in Theorem 1 and 2. The result is shown in Figure 11.

For this measurement, we used exactly the same setup that led to the results in Figure 7, except that all cross traffic was now addressed with multicast. This forced the hub to repeat all data packets towards all nodes on the segment.

The result in Figure 11 is similar to the one observed for the unicast case. However the maximum delay has increased by another packet transmission time. This is the time the measurement node sometimes has to wait before it can sig-

nal its high priority request. The probability of waiting for an inter-packet-gap increases for higher loads. If the system runs close to its capacity limit then each packet sent by the Measurement Client is delayed, which causes the step increase of the minimum delay that can be observed in Figure 11.

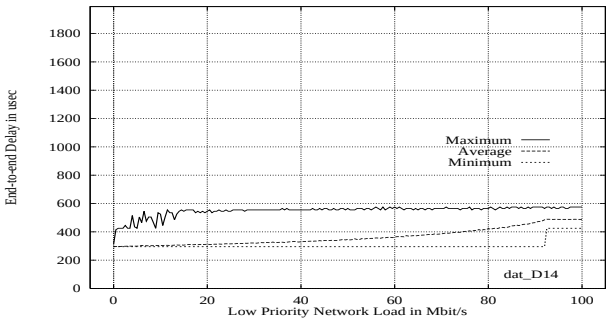

Figure 11. End-to-end Delay using the High Priority Service and Multicast Cross Traffic.

The difference between the maximum- and the minimum delay in Figure 11 is the worst case time it takes to pre-empt the low priority service in a single hub 802.12 network. We measured a maximum of 275 µs . This confirms the theoretical result of D_{it} = 261.92µs that is computed in Appendix A.4.

6.3 Results for the Time Window Algorithm

We implemented and tested the measurement algorithm on a HP 9000/725 workstation as part of our allocation system. All measurements are taken in the device driver of the 802.12 LAN adapter card and are evaluated by the LLRMP demon, just as described in section 5. The tests reported in this section had two goals: (1) to experimentally show that the algorithm can find an accurate upper bound for the packet count and thus for the Demand Priority overhead, and (2) to show that the algorithm is sufficiently conservative such that no service violation occurs.

So far we tested the algorithm using the applications: *vic*, *vat*, *nv*, *MMC* [17], [18] and the *OptiVision MPEG Communication System* [27]. In each test, we recorded the data rate generated by the application, the packet size distribution and the estimation process for the packet count *pcnt* over a measurement time interval of 15 min. During the tests, we varied the data rate of the input source e.g. by changing the camera position and temporary switching off the source. This caused large scale data rate variations.

We further restricted the estimation process. At the end of each time window *TW*, we only updated *pcnt* when the new value *pcnt'* was *smaller* than the existing estimation. This reduced the number of updates and minimized the LLRMP signalling overhead on the network. *pcnt* could have been only increased if a sample had reached the corresponding high watermark. This however never happened in any of the tests. The system parameters of the measurement algorithm, which were used in all experiments are provided in Table 2.

Measurement Time Window *TW*	40 s
Allocation Time Frame *TF*	20 ms
Timer Granularity *T*	1 ms
α	1
κ	2
Minimum Link Packet Size P_{min}	64 byte

Table 1. System Parameters used while Testing the Time Window Measurement Algorithm.

In our first experiment (Test 1), we used *vic* version v2.7b2 as test application. It generated a motion jpeg compressed video data stream with a rate of about 1 Mbit/s. Hardware support was given by a parallax card [25]. The data source was a video camera. We used the following *vic* specific parameters that can be adjusted by the user: normal size (resolution: 368 x 276 pixel), ordered, jpeg, 22 frames/s. All data packets were sent using IP multicast. At the link layer, we allocated 1 Mbit/s for application data using the LLRMP. The video camera was switched off during the time intervals: 0 - 120 s, 480 - 540 s and 780 - 840 s. The measurement results are shown in Figure 12a and Figure 12b.

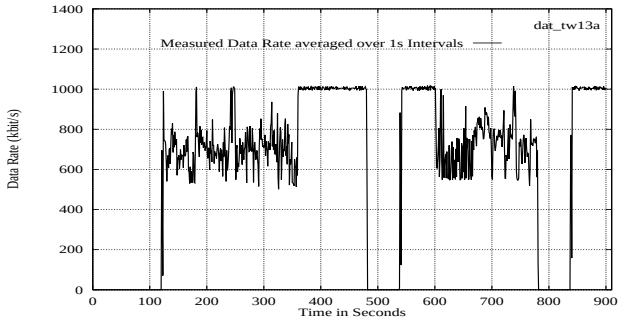

Figure 12a:Data Rate generated by *vic* during Test 1.

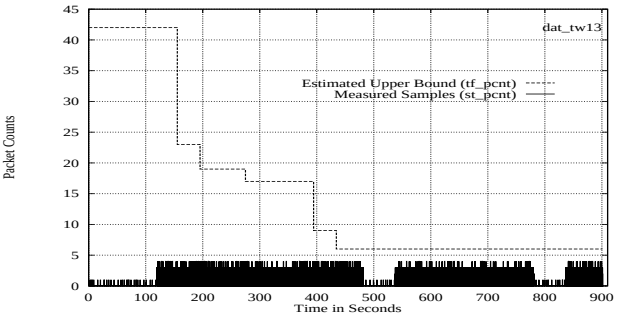

Figure 12b:Packet Count Estimation Process for *vic* in Test 1.

Figure 12a shows the measured data rate, Figure 12b the packet count estimation process. The upper curve in Figure 12b represents the upper bound for the packet count (*pcnt*) that was estimated by the algorithm. The lines at the bottom

of the diagram show the maximum samples ( ) measured during the test. For the sake of brevity, we omitted the result received for the packet size distribution.

The estimation process starts after the flow is admitted. This is at time 0. The initial value for the packet count $pcnt$ is *MAX_PCNT*, which is 42 in the setup for Test 1. It reflects the worst case in which the algorithm assumes that *vic* only sends minimum sized data packets. The initial value for $pcnt$ does not change until *vic* starts sending video data (at time 120 s) because the parameter β in equation 4.1.1 causes any new estimate to be *MAX_PCNT*. As the data rate approaches the allocation limit of 1 Mbit/s, the algorithm is able to find more accurate estimations for the maximum packet count actually used by this application. The most accurate bound in Test 1 is found after about 430 seconds. It is retained despite the fact that the data rate changes later since we only increase $pcnt$ when an individual measurement sample ($scnt$) reaches the high watermark. This however never occurs in Test 1 as can be observed in Figure 12b.

In Test 2, we tested the measurement algorithm with *MMC* version v4.0 as test application. The results for the data rate and the packet count estimation process are shown in Figure 13a and Figure 13b.

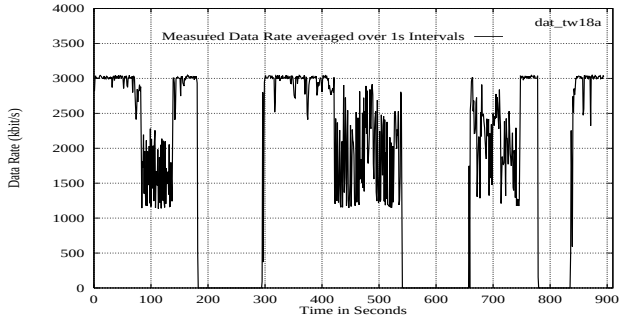

Figure 13a: Data Rate generated by *MMC* during Test 2.

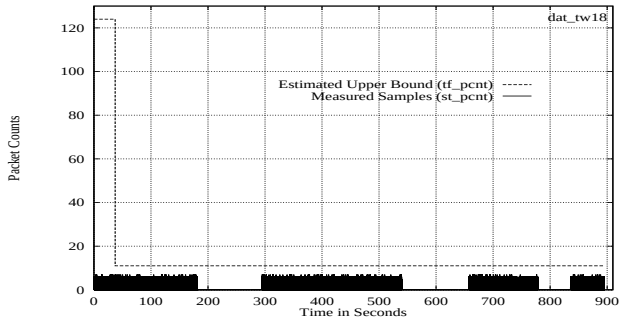

Figure 13b: Packet Count Estimation Process for *MMC* in Test 2.

MMC generated a motion jpeg compressed video data stream of about 3 Mbit/s. This was based on the same parallax card as used in Test 1. The size of the video was 720 x 540 pixel. The application generated about 11 frames/s with an average of about 32 kbytes per frame. All data were sent unicast and used UDP as transport protocol. We allocated a bandwidth of 3 Mbit/s at the link layer. The maximum packet count *MAX_PCNT* was 124 as can be observed in Figure 13b. The video camera was switched off during the time intervals: 180 -300 s, 540 - 660 s and 780 - 840 s.

In contrast to Test 1, the algorithm finds an accurate estimation for the packet count $pcnt$ within a single *TW* interval. This is because MMC instantly used all the resources reserved for it. The estimation is retained through the entire test since there is again no measurement sample that reaches the high watermark. Figure 13b shows an estimation process which is desired for each flow since a maximum upper bound is found quickly and then retained until the end of the session. This ensures minimal LLRMP signalling overhead since the resources reserved for this flow had to be updated at the resource arbiter only once during the test.

Similar experiments as reported in Test 1 and Test 2 were also carried out for *vat*, *nv* and the *OptiVision MPEG Communication System*. For all applications, we repeated the test and varied, where possible, the data rate generated and the data encoding scheme used. All measurement results are similar to the ones discussed for Test 1 and Test 2. They only differ in respect to: (1) the traffic pattern and the samples measured, (2) the adaptation rate and (3) the difference between the worst-case packet count and the estimated upper bound.

The experiments showed that if an application generates data with a rate close to the resources allocated for it, then the measurement algorithm is able to find an accurate upper bound for the packet count actually used. This significantly reduces the number of packet overheads to be considered for an existing application during the admission control of a new flow. The difference (estimation gain) between the worst-case packet count (*MAX_PCNT*) and the final estimated upper bound ($pcnt$) depends on the packet sizes used and on the data rate.

The gain achieved in Test 1 and Test 2 was large because *vic* and *MMC* generated data at a high rate and mainly used large sized data packets. No benefit will be achieved when applications use small sized packets or only generate a low bitrate data stream. No gain was for example observed for *vat* generating: (1) an audio data stream of about 20 kbit/s using GSM encoding, and (2) an audio stream of about 75 kbit/s using PCM2 encoding. In both tests vat used the built-in audio device of the HP 9000/725 workstation. In case (1) no gain could have possibly been achieved since the worst case packet count is already one for a flow with such a low data rate ($MAX\_PCNT = 1$). This however is the minimum number of packet overheads to be reserved for an application in a time frame. It can not be decreased. We did not observe an estimation gain in case (2) due to the conservativeness of the algorithm and the small difference between the worst case overhead ($MAX\_PCNT = 4$) and the maximum sample measured ($scnt_{TW} = 2$).

In all measurements carried out so far, we did not detect a service violation for a single data packet. This could be

observed despite that all applications changed their data rate in a large scale. We also did not observe the case that an individual measurement sample ($scnt$) reached the high watermark and caused the reallocation of resources. We thus believe that the algorithm can be used to estimate the packet overhead for applications using the guaranteed service, provided that the packetization process is constant and does not change over time. So far, we have only tested a very small number of existing applications which might use this service. The test of other applications is left for the future. Further generalizations can be made within the bounds of the Controlled Load service due to the weaker service commitment.

6.4 Resource Utilization Issues

Table 3 shows the maximum number of *vat*, *nv*, *vic*, *OptiVision* and *MMC* flows which our allocation scheme was able to simultaneously admit while guaranteeing a certain queuing delay bound. All results are based on the use of the time window measurement algorithm. Since the number of flows that can be admitted depends on the characteristics of the flow, in particular the packet size distribution, we used the measured characteristics of our test applications for admission control and not an artificially generated traffic pattern.

The goal of this section is to show the maximum high priority resource utilization that can be achieved for a set of test applications by using (1) the allocation scheme in a realistic setup and (2) the time window algorithm proposed in this paper. A generalization of the results for other applications can not easily be made since these applications may have different traffic characteristics e.g. use smaller packet sizes, which then requires the allocation of additional network resources. A higher utilization can however always be achieved when the packet sizes are fixed or can be negotiated since this removes the overhead introduced by the measurement approach.

Following the worst-case model, each flow was first admitted assuming the use of only minimum sized packets, where $P_{min} = 64byte$. For all existing flows, the admission control used the application characteristics measured during the experiments in section 6.3. Note that flow arrival and lifetime statistics were not considered in this test since we focused on determining the highest utilization in a predefined setup. The packet counts shown in Table 2 were measured in the tests listed below. Note that all application parameters e.g. the data rate were measured at link layer.

1. *vat* version v3.2: *vat* generated an audio data stream of about 75 kbit/s. The test used the default application setup for PCM2 audio encoding. The data source was the built-in audio device of the HP 9000/725 workstation. All data packets were sent using IP multicast. 75 kbit/s were allocated at the link layer.

2. *nv* version v3.3beta: *nv* generated a video data stream of about 128 kbit/s. Hardware support was provided by an HP A.B9.01.3A frame grabber card. The test used the default setup for *nv* with a medium picture size. All data packets were sent using IP multicast. We allocated 128 kbit/s at the link layer.

3. *vic*: this used the same test setup as described for Test 1 in section 6.3. The data rate and an example for the packet count estimation process are shown in Figure 12a and Figure 12b.

4. *OptiVision* version 1.2f: the OptiVision system generated an MPEG-1 encoded video stream with an average rate of about 1.2 Mbit/s. The video source was a video player playing the adventure movie *Jurassic Park*. The picture resolution was 704 x 480 pixel. 25 frames per second were generated. All data packets were transmitted using IP multicast. We allocated 1.8 Mbit/s at the link layer for each flow.

5. *MMC*: this used the same test setup as described for Test 2 in section 6.3. The data rate and an example for the packet count estimation process are shown in Figure 13a and Figure 13b.

Table 2 shows the maximum number of flows (N_{max}) that could be admitted for three different time frames: 10 ms, 20 ms and 40 ms. For the sake of simplicity, the queuing delay bound requested for all flows was always equal to the time frame. The timer granularity T was 1 ms (see equation 3.3.3). The rate regulators allowed an initial burst of $\delta^i = 12000\text{bits}$, which corresponds to one maximum size data packet. We further always admitted homogeneous flows. Each row in Table 2 provides the result for one application in a given setup: e.g. for a time frame of $TF = 20ms$, a maximum of 49 *vic* flows, each generating data at a rate of 1 Mbit/s, can be simultaneously admitted while providing a deterministic delay bound of 20 ms for each of them.

After admitting all flows, a total bandwidth of about 49 Mbit/s is transmitted using the 802.12 high priority mechanism. The maximum high priority network utilization is computed by relating the allocated bandwidth to the maximum allocation limit. The maximum allocation limit is the maximum capacity that can be allocated when all data is sent with maximum sized packets. It is fixed for each topology and can thus be used as reference value for computing the network utilization. For a single hub network and a time frame of 20 ms, the maximum allocation limit is 91.02 Mbit/s. This corresponds to a maximum high priority network utilization of 53.83% for the 49 1 Mbit/s *vic* flows. The allocation limit was computed using Theorem 1 and the network parameters for 100 m UTP cabling provided in Table 5 and Table 7 in Appendix A.3 and A.4.

Time frame TF	Delay Bound	Application	Data rate allocated per flow.	Max. number of flows admitted (N_{max})	$pcnt$ measured	Bandwidth allocated (Mbit/s)	Maximum high priority network utilization (%)
10 ms	10 ms	vat	75 kbit/s	65	2	4.88	5.43
	10 ms	nv	128 kbit/s	59	3	7.55	8.41
	10 ms	vic	1 Mbit/s	34	5	34.00	37.86
	10 ms	OptiVision	1.8 Mbit/s	24	7	43.20	48.10
	10 ms	MMC	3 Mbit/s	17	8	51.00	56.78
20 ms	20 ms	vat	75 kbit/s	112	4	8.40	9.23
	20 ms	nv	128 kbit/s	105	4	13.44	14.77
	20 ms	vic	1 Mbit/s	49	6	49.00	53.83
	20 ms	OptiVision	1.8 Mbit/s	32	9	57.60	63.28
	20 ms	MMC	3 Mbit/s	21	11	63.00	69.21
40 ms	40 ms	vat	75 kbit/s	197	5	14.78	16.13
	40 ms	nv	128 kbit/s	170	6	21.76	23.75
	40 ms	vic	1 Mbit/s	61	10	61.00	66.58
	40 ms	OptiVision	1.8 Mbit/s	37	16	66.60	72.69
	40 ms	MMC	3 Mbit/s	24	17	72.00	78.58

Tabelle 2: High Priority Network Utilization in a Single Hub 802.12 Network.

In Table 2, several observations can be made. The maximum high priority network utilization achieved when only low bitrate (*vat*) flows are admitted is low. This has two reasons. First, *vat* only uses small sized data packets which reduces the available bandwidth on the network. The utilization is further decreased by the allocation overhead. For each flow, the allocation scheme reserves resources for at least one maximum size data packet in each time frame to ensure that deterministic service guarantees are met. This is required since the time frames of different nodes in the network are not synchronized. The allocation overhead could be reduced, at the expense of a more complicated allocation system, by (1) introducing synchronization mechanisms between high priority network nodes, and (2) by determining a lower bound for the maximum packet size used by each flow. We however believe that the utilization in the existing scheme is sufficient so that such mechanisms are not necessary.

For higher bitrate streams e.g. 1 Mbit/s *vic* flows, a much higher utilization can be achieved because of a smaller overhead and a larger allocation limit. An increase of N_{max} can further be observed for all applications in Table 2 when larger delay bounds and time frames become used in the allocation system.

Remaining work in this context includes the comparison between the deterministic delay bound provided by the allocation system and the maximum end-to-end delay measured for several applications in our test network. This is performed in the sequel for the level-2 cascaded topology.

6.5 Conclusions from our Implementation

The importance of the measurement results reported in this section is that they confirm the basics of our reservation scheme. These are: (1) the 802.12 high priority access mechanism is sufficiently isolated such that an advanced service can be built on top of it. If admission control is applied then packet delays are predictable and, apart from an initial interrupt time, independent of the low priority traffic. (2) Theorem 1 can be used to accurately calculate the minimum available bandwidth on the network. The experiments further confirmed the theoretical result for the low priority service interrupt time. Both are used to compute the allocation limit, up to which resources can be reserved. (3) For all applications tested, the time window algorithm could find an accurate upper bound for the total Demand Priority overhead. We observed that the algorithm is sufficiently conservative such that no service violation occurs. (6) The implementation further allowed us to experimentally determine the operating system overhead caused by the DMA- and the interrupt process. (7) Our measurements have also demonstrated the basic operation of our link level signalling protocol (LLRMP).

A Appendices

A.1 Proof of Theorem 1

Theorem 1 is based on a simple sum approach which also includes the Demand Priority protocol overhead. To prove the theorem, we first define the time frame TF as the busy period interval as used in [19]. This period is the maximum interval of time for which high priority data is sent on the network at link speed C_l. The idea is that during the busy period, the amount of traffic that enters the system is equal to the amount of data that is served. This is ensured by allocating resources for all data which can leave the link-level rate regulators at all nodes in the network within the time frame TF.

The busy period also includes a time offset required at the start of the interval to pre-empt the low priority service. We denote this offset D_{it}. It follows that, if the amount of data that is passed in the high priority output queue on each node k is bounded by the traffic constraint function $b_k^i(\Delta t)$ for all flows i on node k and all intervals $\Delta t = TF$, then TF is the busy period of the system if:

$$D_{it} + \frac{1}{C_l} \cdot \sum_{k=1}^{m} \sum_{i=1}^{n} b_k^i(TF) \leq TF \qquad \text{(A.1.1)}$$

applies, where m, n denote the number of network nodes and the number of flows with reservations on each node, respectively. If used for admission control, then equation A.1.1 restricts the amount of data that can use the high priority access mechanism. In an overhead free network, this would ensure that any backlog of high priority packets is cleared in a time interval smaller or equal to TF. Since the Demand Priority protocol overhead however has a significant impact on the network performance, it needs to be considered in the admission control and must therefore be added to equation A.1.1.

In order to bound the overhead, we consider the number of packets sent by each flow i in every time frame TF. This number is denoted $pcnt^i$. In can be the exact number of packets sent by flow i, or an upper bound if packet sizes are neither fixed nor negotiatable. An upper bound can always be computed by assuming that the flow uses minimum sized packets for the data transmission. Since (1) $pcnt^i$ exists for all real-time flows, and (2) the per-packet overhead is independent of the length of a data packet, the total transmission overhead within the time frame TF can be computed.

If we assume that the worst case per-packet overhead is D_{pp} and that $pcnt_k{}^i$ denotes the maximum number of packets sent by flow i on node k, then by adding D_{pp} for each data packet served, we get from equation A.1.1:

$$D_{it} + \frac{1}{C_l} \cdot \sum_{k=1}^{m} \sum_{i=1}^{n} b_k^i(TF) + \sum_{k=1}^{m} \sum_{i=1}^{n} pcnt_k^i \cdot D_{pp} \le TF \qquad \text{(A.1.2)}$$

It follows that, a new flow ν with a traffic constraint function b^ν can be accepted if:

$$D_{it} + \frac{1}{C_l} \cdot \sum_{k=1}^{m} \sum_{i=1}^{n} b_k^i + \sum_{k=1}^{m} \sum_{i=1}^{n} pcnt_k^i \cdot D_{pp} + \frac{b^\nu}{C_l} + \frac{b^\nu}{P_{min}} \cdot D_{pp} \le TF \qquad \text{(A.1.3)}$$

holds, where b^i is equivalent to $b^i(TF)$. The new flow is first admitted assuming a data transmission that uses packets of minimum size P_{min}. If the packet size is fixed then we can replace P_{min} in equation A.1.3 with the actual packet size $psize^\nu$ used by the new flow ν. If ν uses variable packet sizes and the actual number of data packets transmitted is known or can be negotiated then the term b^ν / P_{min} in equation A.1.3 can be replaced by flow ν's packet count $pcnt^\nu$. Theorem 1 follows directly from re-arranging equation A.1.3. □

A.2 Proof of Theorem 2

Network node k passes a maximum of

$$PCNT_k = \sum_{i=1}^{n} pcnt_k^i \qquad \text{(A.2.1)}$$

packets into its high priority output queue within a time frame TF. This is enforced by using a rate regulator for each real-time flow i on node k. If Theorem 1 applies for all high priority traffic in the network, then the delay for all data packets is bounded by TF. This ensures that at any time, there are never more than $PCNT_k$ packets in node k's high priority queue.

The worst case delay d_k for the last packet in the output queue of node k consists of: (1) the interrupt time required to signal the high priority service request and to pre-empt the low priority packet transmission, (2) the transmission delay: defining the time it takes to transmit all packets through the network stack and over the physical medium, and (3) the queuing delay: packets on node k might have to wait until high priority requests on other nodes have been served according to the round-robin service policy. We get:

$$D_{it} + dT_k + dQ_k \le d_k \le TF \qquad \text{(A.2.2)}$$

where D_{it}, dT_k, dQ_k denote the interrupt time, the transmission- and the queuing delay, respectively. We now provide bounds for all three components. The worst case low priority service interrupt time D_{it} is constant and mainly depends on the cascading level. The transmission of a maximum of $PCNT_k$ data packets queued at node k is bounded by:

$$dT_k \le \frac{1}{C_l} \cdot \sum_{i=1}^{n} b_k^i(TF) + \sum_{i=1}^{n} pcnt_k^i \cdot D_{pp} \qquad \text{(A.2.3)}$$

This follows from the considerations in A.1. The queuing delay dQ_k on node k depends on the number of high priority packets queued on all other nodes during the interval TF. This number however is bounded by $PCNT_j$ for each node j on the network due to the packet regulating mechanisms in the rate regulator.

The service of packets from node k is most delayed by node j, when node j has at least as many packets queued as node k. In general, two cases can be identified. If we first assume that node j has more than $PCNT_k$ maximum size packets in its output queue, then the hub serves the same number of packets from node j and k until all packets on k have been transmitted. Some packets are still in the queue on j, but they do not have to be considered for the delay computation on k. Thus we have the relation:

$$if \quad PCNT_k \le \sum_{i=1}^{n} \frac{b_j^i(TF)}{P_{max}} \quad then \quad dQ_{k,j} \le PCNT_k \cdot \frac{P_{max}}{C_l} \qquad \text{(A.2.4)}$$

If, in contrast, node j has less packets to send than node k, then all packets on j become served during the transmission of $PCNT_k$ packets from node k. This is due to the round-robin policy. For this case, we receive the relation:

$$if \quad PCNT_k > \sum_{i=1}^{n} \frac{b_j^i(TF)}{P_{max}} \quad then \quad dQ_{k,j} \le \sum_{i=1}^{n} \frac{b_j^i(TF)}{P_{max}} \cdot \frac{P_{max}}{C_l} \qquad \text{(A.2.5)}$$

If we now consider all nodes j in the network with $j \neq k$, we have from relation A.2.4 and A.2.5:

$$dQ_k \le \sum_{j=1,\, j \neq k}^{m} \left(MIN\left(PCNT_k, \sum_{i=1}^{n} \frac{b_j^i}{P_{max}} \right) \cdot \frac{P_{max}}{C_l} \right) \qquad \text{(A.2.6)}$$

Equation A.2.6 provides an upper bound on the service time required to serve all nodes j, while the maximum of $PCNT_k$ packets is served from k.

Finally, we have to add the worst case Demand Priority per-packet overhead, denoted by D_{pp}. It follows from A.2.4 and A.2.5 that this delay is bounded by the minimum of $PCNT_k$ and $PCNT_j$. Thus, we have from equation A.2.7:

$$dQ_k \le \sum_{j=1,\, j\neq k}^{m} \left(MIN\left(PCNT_k, \sum_{i=1}^{n} \frac{b_j^i}{P_{max}} \right) \cdot \frac{P_{max}}{C_l} + MIN(PCNT_k, PCNT_j) \cdot D_{pp} \right) \quad \text{(A.2.7)}$$

If we now substitute equation A.2.1 in equation A.2.7 and insert A.2.3 and A.2.7 in equation A.2.2, then we receive for the maximum delay on node k:

$$\sum_{j=1,\, j\neq k}^{m} \left(MIN\left(\sum_{i=1}^{n} pcnt_k^i, \sum_{i=1}^{n} \frac{b_j^i}{P_{max}} \right) \cdot \frac{P_{max}}{C_l} + MIN\left(\sum_{i=1}^{n} pcnt_k^i, \sum_{i=1}^{n} pcnt_j^i \right) \cdot D_{pp} \right) +$$

$$\frac{1}{C_l} \cdot \sum_{i=1}^{n} b_k^i + \sum_{i=1}^{n} pcnt_k^i \cdot D_{pp} + D_{lt} \quad \le \quad d_k \le \quad TF \qquad \text{(A.2.8)}$$

This is Theorem 2. □

A.3 The Worst-Case Packet Transmission Model and Per-Packet Signalling Overhead

Within this appendix, we describe the details of the data transmission on a 802.12 network and discuss our worst case model to compute the packet overhead caused by: (1) the Demand Priority protocol itself and (2) by passing a data packet through the protocol stack. We consider a single hub topology using Category 3 UTP non-bundled cabling as the physical links.

As with other network technologies standardized within the IEEE, 802.12 is structured in a Media Access Control (MAC) sublayer, a Physical Medium Independent (PMI) sublayer, a Medium Independent Interface (MII), and a Physical Medium Dependent (PMD) sublayer. The MAC controls the access to the medium and carries out the frame preparation and link training. The PMI sublayer performs the quartet channelling, data encoding, and adds the preamble-, fill pattern, starting- and ending delimiter. The data encoding uses a 5B6B block coding scheme. The PMD performs the NRZ encoding and controls the link status. We refer to [11] for the details.

The medium access in 802.12 networks is centralized and controlled by the hub. An end-node wishing to transmit a data packet must wait until it receives permission from the hub. The communication between nodes and the hub is based on the exchange of link control signals. There are 6 primary control signals that are relevant for the packet transmission in single hub networks. The Idle signal (*Idle*) indicates that the sender e.g. the end-node currently has no request pending for the hub connected at the other end of the link. The Request signal (*Req_H*, *Req_L*) is used by end-nodes to demand the transmission of a normal (*Req_L*) or high priority (*Req_H*) data packet. The Grant signal (*Grant*) indicates that the end-node has been given permission to send a packet. *Incoming* will be signalled by the hub in order to inform end-nodes that a packet may soon be sent to them. This allows them to prepare themselves for receipt.

In our packet transmission model, we make worst case assumptions for the signalling delay and the packet processing in order to comply with the requirements for a deterministic service. The model describes the case when the lowest network throughput is achieved with a hub that never runs idle. The worst case is reached in two configurations: (1) when two network nodes are switching between sending and receiving unicast data packets, or (2) when two or more nodes send data packets using the multicast or broadcast addressing mechanism. In both cases the receiver of the last data packet is also the receiver of the next grant. This forces the hub to add an extra time offset, which is called *SEND_IDLE_BURST* (*I_BST*), before the Grant is signalled to the end-node.

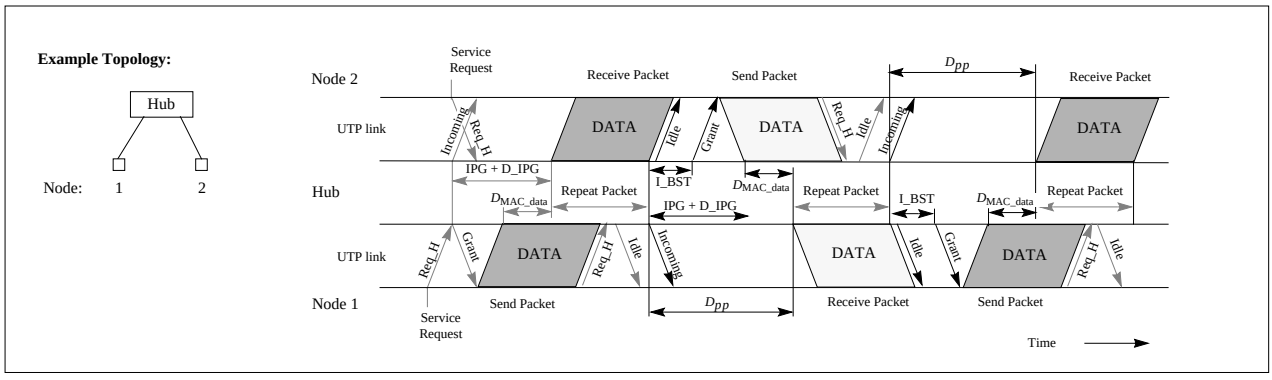

Figure 14. Worst Case Signalling and Data Transmission on a single Hub 802.12 Network.

Figure 14 shows the packet transmission and the signalling required for transmitting three data packets using the high priority service. The example topology consists of a single hub and two end-nodes sending multicast data packets. The worst case per-packet signalling overhead is denoted by D_{pp}.

The data flow starts when the upper layer of Node 1 passes a data packets to the 802.12 MAC layer. After receiving the packet, the MAC at Node 1 signals Req_H to the hub demanding the transmission of the high priority data packet. If the hub is idle, as assumed at the beginning of the data flow in Figure 14, then the hub immediately acknowledges the request and returns a Grant signal to Node 1.

At the same time, the hub signals Incoming to all other nodes on the network e.g. to Node 2. After detecting the Grant, Node 1 starts transmitting the data packet to the hub, which then forwards the packet to Node 2. The packet processing in the hub introduces a small delay which is denoted with D_{MAC_data} in Figure 14. While the rest of the packet is repeated, the hub signals Idle to all nodes other than the destination e.g. to Node 1. This allows them to signal their next service request (Req_H, Req_L) or idle (Idle) to the hub. In Figure 14, Node 1 demands the transmission of another high priority packet by signalling Req_H. This assumes that another data packet was passed into the output queue at Node 1 while the first packet was transmitted to the hub.

In the meantime, the hub in Figure 14 has also received a transmission request from Node 2. This request is granted after the data packet from Node 1 has been fully repeated. The corresponding Grant signal however is not signalled before the *SEND_IDLE_BURST* (*I_BST*) timer has expired. This potentially allows Node 2 to signal a service request to the hub. The transmission of the data packet from Node 2 requires the same signalling as described for the previous data packet. After this packet has been repeated, the hub continues and serves the next packet from Node 1 and so on, until all requests have been served.

The medium access mechanism defines that the gap between two subsequent packet transmissions is always larger than a certain defined time interval called the *Inter-Packet Gap* (*IPG*). This is enforced by a timer mechanism at the hub. The corresponding timer is called the *IPG* timer. If the packet was received from an end-node, then the inter packet gap is increased by an additional time offset of length *D_IPG*. It accounts for clock differences between different hubs in the shared network. The packet overhead D_{pp} is thus at least as big as *IPG* plus *D_IPG*. The worst-case however is determined by the maximum signalling-, packet-processing and propagation delay as illustrated in Figure 14. This includes the worst case delay for: (1) signalling Grant from the hub to the end-node, (2) passing the data packet through the 802.12 protocol stack, (3) sending the data packet over the link, (4) receiving the packet at the hub and passing it to the MAC layer, and (5) decoding the address information and passing the data packet to the PMI of the outgoing port. The precise breakdowns for these operations are given in Table 3 and Table 4.

All delays are worst case delays and are based on references in the standard. Note that a Bit Time (BT) corresponds to $33.\bar{3}$ ns, e.g. $D_{PMD\_Rx\_grant}$ in Table 3 is equal to 200 ns. The propagation delays on the physical medium are provided for 100 m Category 3 UTP cable. Further, we assume in our analysis, that the Medium Independent Interface (MII) itself does not introduce any significant delay.

Sublayer	Comments	Worst Case Delay	Reference in [11]:
RMAC (Hub)		-	12.6.3.4 12.6.4.1
PMI	$D_{PMI\_Tx\_ctr}$ Control signal encoding, (control signals do not have a preamble).	4 BT	14.3.1
PMD	$D_{PMD\_Tx\_ctr}$ Max. propagation delay within the PMD.	20 BT	16.5.3.2
PHY (Link)	D_{PHY_UTP} Prop. delay on 100 m UTP or STP cable.	570 ns	16.9.1.3
PMD	$D_{PMD\_Rx\_grant}$ Grant signal detection.	6 BT	16.6.5
PMI	$D_{PMI\_Rx\_ctr}$ Control signal mapping.	4 BT	14.3.2 14.3.3
MAC (Receiver)		-	

Table 3. Breakdown of the Grant-Signalling Delay.

Sublayer	Comments	Worst Case Delay	Reference in [11]:
MAC (Source)		-	12.6.3.4 12.6.4.1
PMI	$D_{PMI\_Tx\_data}$ Addition of preamble pattern (48 BT): Addition of starting delimiter (12 BT): Propagation delay for data (3 BT):	63 BT	 14.4.2.3.2 14.4.2.3.3 14.3.4
PMD	$D_{PMD\_Tx\_data}$ Max. propagation delay within PMD.	8 BT	16.5.2
PHY (Link)	D_{PHY_UTP} Prop. delay on 100 m UTP or STP cable.	570 ns	16.5.3.2
PMD	$D_{PMD\_Rx\_data}$ Data recovery delay.	10 BT	16.6.4
PMI	$D_{PMI\_Rx\_data}$ Synchronization, data decoding (8 BT): Propagation delay within the PMI (3 BT):	11 BT	 14.4.4 as 14.3.4
MII -> MII (Hub)	$D_{MII\_Rx\_Tx\_data}$, Transmit delay MII -> MII in the RMAC:	4.5 µsec	12.9.7.2

Table 4. Breakdown of the Data Transmission Delay.

Using the transmission model in Figure 14 and the worst case delays given in Table 3 and Table 4, we are able to compute the worst case per-packet overhead D_{pp}. In the following, we will denote the overhead caused by the data transmission over a single link e.g. from the hub to the end-node by D_{Tx_Data}. The parameter D_{Signal_Grant} is the worst-case time it takes to signal *Grant* from the hub to the end-

node. Both parameters includes the overhead which is introduced in the 802.12 protocol stack e.g. while detecting or decoding the link signal. They are computed later.

Under idle conditions, the Grant signal can travel much faster than the data signal due to a smaller overhead in the sending and receiving 802.12 PMDs and PMIs. We can however observe in Figure 14 that the Grant always travels behind a data packet. Node 2 thus cannot detect the Grant signal in $I\_BST + D_{Signal\_Grant}$ time units after the hub has made its decision to serve this node. Instead, Node 2 first has to receive the data packet. We assume in our model that the Grant has been detected *I_BST* time units after the last bit of the data packet has been received at Node 2. The resulting delay is thus: $D_{Tx\_Data} + I\_BST$. When detecting the Grant, Node 2 instantly sends the data packet. It takes not more than $D_{Tx\_Data} + D_{MAC\_data} + P_{max}/C_l$ time units until the hub has fully repeated the packet from Node 2, where D_{MAC_data} denotes the worst case time, the packet is delayed in the Repeater MAC (RMAC) of the hub. P_{max}/C_l is the transmission time for a data packet of maximum size. If we now consider that the per-packet overhead is always larger than the Inter-packet Gap then we get for the worst case per-packet overhead D_{pp} :

$$D_{pp} \leq MAX((IPG + D\_IPG);\ (D_{Tx\_Data} + I\_BST + D_{Tx\_Data} + D_{MAC\_data})) \quad \text{(A.3.1)}$$

The timer values for the *IPG*- and *D_IPG* window, and the *I_BST* offset are defined in the standard (see section 12.5.1). The numerical results for D_{Signal_Grant} and D_{Tx_Data} immediately follow from Table 3 and Table 4 by adding up the delay components introduced in each layer of the protocol stack. We thus have:

$$D_{Signal\_Grant} = D_{PMI\_Tx\_ctr} + D_{PMD\_Tx\_ctr} + D_{PHY\_UTP} + D_{PMD\_Rx\_grant} + D_{PMI\_Rx\_ctr} \quad \text{(A.3.2)}$$

$$D_{Tx\_Data} = D_{PMI\_Tx\_data} + D_{PMD\_Tx\_data} + D_{PHY\_UTP} + D_{PMD\_Rx\_data} + D_{PMI\_Rx\_data} \quad \text{(A.3.3)}$$

The delay in the RMAC sublayer (D_{MAC_data}) is computed using the delay bounds given in Table 4. Since the standard provides the worst case delay between the receiving and transmitting MII of the RMAC, we receive D_{MAC_data} by taking off the delays added by the PMIs:

$$D_{MAC\_data} = D_{MII\_Rx\_Tx\_data} - D_{PMI\_Rx\_data} - D_{PMI\_Tx\_data} \quad \text{(A.3.4)}$$

This provides a delay of 2.033 µs for D_{MAC_data}. The value for D_{MAC_data} is fixed, the results for D_{Signal_Grant} and D_{Tx_Data} however depend on the cable length.

From equations A.3.1 - A.3.4 and the values in Table 3 and Table 4, we computed a worst case per-packet overhead for the single hub 802.12 network. The results for 100 m and 200 m UTP cabling are shown in Table 5. This is what we used throughout our theoretical analysis.

UTP cable length	D_{pp}
100 m	10.109 µs
200 m	11.249 µs

Table 5. The Worst-Case Packet Overhead *Dpp* for the Single Hub 802.12 Network using UTP Cabling.

A.4 Worst-Case Low Priority Service Interrupt Time

Within this appendix, we describe the model used to compute the worst case time it takes to interrupt the low priority service in single hub 802.12 networks. The model is illustrated in Figure 15. It shows the worst case signalling required for pre-empting the low priority service and for transmitting a single high priority packet. The analysis is focused on the use of non-bundled Category 3 UTP cabling as physical links.

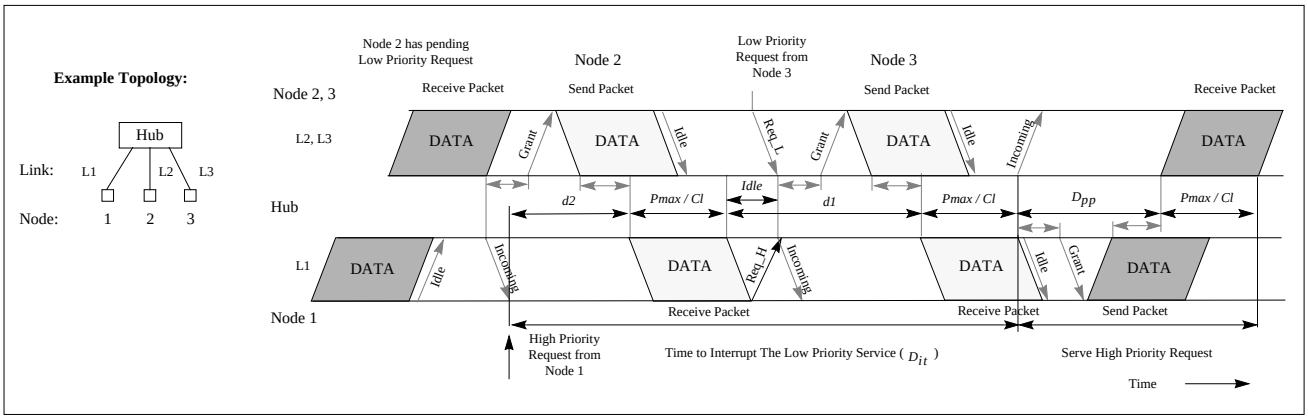

Figure 15. Model for Computing the Worst Case Interrupt Time.

The example topology shown in Figure 15 consists of a single hub and three nodes. These might for example be end-hosts or bridges. We describe the worst case interrupt time in respect to Node 1 which is requesting the transmission of a high priority data packet. The two other nodes in the setup, Node 2 and Node 3, only use the low priority service. Similar to the packet transmission model discussed in Appendix A.3, the worst case delay occurs when Node 2 and Node 3 send data packets using the multicast or broadcast mechanism, while Node 1 is requesting the high priority service. Note that the hub forwards multicast data packets to all network nodes, regardless of whether they have joined the corresponding multicast group. Multicast data packets will thus always be forwarded to Node 1. We further assume all data packets in Figure 15 to be of maximum size.

In Figure 15, the worst case low priority service interrupt time is denoted by D_{it} . The worst case occurs when the signalling of the high priority request (Req_H) from Node 1 to the hub is delayed by the transmission of low priority data packets on the network. In a single hub topology, a maximum of two data packets can be served by the hub before the low priority service is pre-empted. This is caused by the properties of the UTP physical layer which operates in half duplex mode. When used over UTP cable, packet data is transmitted on all four pairs in order to reach the desired physical link throughput. Control signals between end-nodes and the hub can thus only be exchanged during the inter packet gap and not while data are transmitted over the link.

The data flow in Figure 15 starts when Node 1 sends a multicast data packet. This is forwarded towards Node 2 and Node 3. At the same time, we assume that Node 2 has a pending low priority service request. Following the worst case model discussed in the previous appendix, Node 2 must first receive the data packet from Node 1, before it can detect the Grant signal from the hub. Instantly after the hub has decided to serve Node 2, it also signals Incoming to all other nodes on the network e.g. to Node 1 which is running idle. The worst case condition for D_{it} occurs if a high priority request is made at Node 1 instantly after the Incoming signal was detected. In this case, the physical layer at Node 1 does not signal Req_H to the hub because it must prepare itself for receiving the data packet from Node 2. As shown in Figure 15, the Req_H signal is not transmitted before the low priority data packet from Node 2 has been fully received at Node 1.

After the hub repeated the packet from Node 2, it runs idle until it receives a demand for transmitting a low priority data packet from Node 3. The worst case occurs when the high priority request from Node 1 arrives at the hub just after the low priority request from Node 3 has been acknowledged. The hub then first grants the transmission of the packet from Node 3. After forwarding this packet, the low priority service is pre-empted and the hub starts to serve the high priority packet from Node 1. Once pre-empted, the low priority service is only resumed after all high priority packets have been served. Note that even though the low priority request arrives later at Node 3, it is served earlier than the high priority data packet from Node 1.

Assuming that both nodes, Node 2 and Node 3, send a maximum size data packet, we can observe in Figure 15 that the worst case interrupt time D_{it} is given by:

$$D_{it} = 2 \cdot \frac{P_{max}}{C_l} + d_2 + d_1 \tag{A.4.1}$$

where P_{max}/C_l is the time it takes to transmit one data packet of maximum size. The two constants d_2 and d_1 contain the overhead for the two low priority data packets. The overhead for the data packet from Node 2 is the worst case overhead D_{pp} as determined for the data packets in Figure 14 in Appendix A.3. For the interrupt time, we however only have to consider:

$$d_2 = D_{pp} - D_{Incom} \tag{A.4.2}$$

where D_{Incom} is the time it takes to signal Incoming across a single UTP link. The overhead for the low priority packet from Node 3 follows from Figure 15:

$$d_1 = D_{Tx\_Data} + D_{Req\_H} + I\_BST + D_{Signal\_Grant} + D_{Tx\_Data} + D_{MAC\_data} \tag{A.4.3}$$

where D_{Tx_Data}, D_{Signal_Grant}, D_{MAC_data} and I_BST are the parameters discussed in the previous appendix. D_{Req_H} is the times it takes to signal Req_H across link L1. Both parameters, D_{Req_H} and D_{Incom}, have the same numeric value which we denote with D_{Signal_Ctrl}:

$$D_{Incom} = D_{Req\_H} = D_{Signal\_Ctrl} \tag{A.4.4}$$

A precise breakdown for D_{Signal_Ctrl} is provided in Table 6. Using these components, we get:

$$D_{Signal\_Ctrl} = D_{PMI\_Tx\_ctr} + D_{PMD\_Tx\_ctr} + D_{PHY\_UTP} + D_{PMD\_Rx\_ctr} + D_{PMI\_Rx\_ctr} \tag{A.4.5}$$

D_{Signal_Ctrl} is larger than D_{Signal_Grant} since the PMD can detect a Grant signal faster than any other link control signal. One can further observe, the worst case for D_{it} is achieved when the network is not fully loaded since the hub in Figure 15 runs idle for a short time after serving the low priority data packet from Node 2. The packet overhead d_1 is thus larger than the worst-case overhead D_{pp} since it includes the idle time: $D_{Tx\_Data} + D_{Req\_H}$. This however does not have any significant impact on the result for D_{it}. In a fully loaded network, d_1 is equal to D_{pp}.

Sublayer	Comments	Worst Case Delay	Reference in [11]:
RMAC (Hub)		-	12.6.3.4 12.6.4.1
PMI	$D_{PMI\_Tx\_ctr}$ Control signal encoding, (control signals do not have a preamble).	4 BT	14.3.1
PMD	$D_{PMD\_Tx\_ctr}$ Max. propagation delay within the PMD.	20 BT	16.5.3.2
PHY (Link)	D_{PHY_UTP} Prop. delay on 100 m UTP or STP cable.	570 ns	16.9.1.3
PMD	$D_{PMD\_Rx\_ctr}$ Control signal recovery and decoding.	48 BT	16.6.1
PMI	$D_{PMI\_Rx\_ctr}$ Control signal mapping.	4 BT	14.3.2
MAC (Receiver)		-	

Table 6. Breakdown of the Delay required for Signalling the Control Signals *Req_H, Req_L* and *Incoming*.

From equations A.4.1 - A.4.5, and the results in Appendix A.3, we were able to compute the worst case interrupt time D_{it} to be considered in a single hub 802.12 network. The result for a UTP cable length of 100 m and 200 m are shown in Table 7.

UTP cable length	D_{it}
100 m	261.92 µs
200 m	264.77 µs

Table 7. The Worst-Case Low Priority Service Interrupt Time *Dit* for a Single Hub 802.12 Network using UTP Cabling.

References

[1] P. Kim, *Deterministic Service Guarantees in 802.12 Networks, Part II: the Cascaded Network Case*, HP Technical Report HPL-97-148, April 1997.

[2] D. Clark, S. Shenker, L. Zhang, *Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism*, in Proc. ACM SIGCOMM '92, pp. 14 - 26, Aug. 1992.

[3] S. Shenker, G. Partridge, R. Guerin, *Specification of the Guaranteed Quality of Service*, Internet Draft draft-ietf-intserv-guaranteed-svc-06.txt, August 1996.

[4] J. Wroclawski, *Specification of the Controlled-Load Network Element Service*, Internet Draft draft-ietf-intserv-ctrl-load-svc-03.txt, August 1996.

[5] H. Zhang, S. Keshav, *Comparison of Rate-Based-Service Disciplines*, in Proc. of ACM SIGCOMM '91, pp. 113 - 121, September 1991.

[6] S. Jamin, *A Measurement-based Admission Control Algorithm for Integrated Services Packet Networks*, PhD Dissertation, University of Southern California, August 1996.

[7] S. Floyd, *Comments on Measurement-based Admission Control for Controlled-Load Services*, pre-released Draft, submitted to CCR, July 1996.

[8] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, *RSVP: A New Resource ReSerVation Protocol*, IEEE Networks, September 1993.

[9] R. Braden, L. Zhang, S. Berson, S. Herzog, S.Jamin, *Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification*, Internet Draft draft-ietf-rsvp-spec-14.ps, October 1996.

[10] A. Ghanwani, J. W. Pace, V. Srinivasan, *A Framework for Providing Integrated Services Over Shared and Switched LAN Technologies*, Internet Draft draft-ietf-issll-is802-framework-01.txt, April 1997.

[11] *IEEE 802.12, IEEE Standard for Local and Metropolitan Area Networks: Demand-Priority Access Method, Physical Layer and Repeater Specification for 100Mb/s Operation*, IEEE, November 1995.

[12] G. Watson, A. Albrecht, J. Grinham, J. Curcio, D. Dove, S. Goody, M. Spratt, P. Thaler, *The Demand Priority MAC Protocol*, IEEE Network Vol. 9, No. 1, pp. 28 - 34, Jan. 1995.

[13] M. Molle, G, Watson, *100Base-T/IEEE 802.12/Packet Switching*, IEEE Communications Magazine, pp. 64 - 73, August 1996.

[14] J. Flick, *Definitions of Managed Objects for IEEE 802.12 Repeater Devices*, Internet Draft, June 1995.

[15] J. Case, M. Fedor, M. Schoffstall, C. Davin, *Simple Network Management Protocol (SNMP)*, RFC 1157, May 1990.

[16] D. Ferrari, D. Verma, *A Scheme for Real-Time Channel Establishment in Wide-Area Networks*, IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, pp. 368 - 279, April 1990.

[17] Mbone Tools, Online Software: http://www-nrg.ee.lbl.gov/.

[18] N. Leymann, *Eine Videokomponente fuer das Videokonferenzsystem Multimedia Collaboration*, Diploma Thesis, in German, Technical University Berlin, 1996.

[19] R. L. Cruz, *A Calculus for Network Delay, Part I: Network Elements in Isolation*, IEEE Transactions on Information Theory, Vol. 37(1), pp. 114 - 131, Jan. 1991.

[20] R. L. Cruz, *A Calculus for Network Delay, Part II: Network Analysis*, IEEE Transactions on Information Theory, Vol. 37(1), pp. 132 - 141, Jan. 1991.

[21] G. Agrawal, B. Chen, W. Zhao, *Local Synchronous Capacity Allocation Schemes for Guaranteeing Message Deadlines with the timed Token Protocol*, in Proc. of INFOCOM '93, pp.186-193, 1993.

[22] K. Shin, Q. Zheng, *Mixed Time-Constrained and Non-Time-Constrained Communications in Local Area Networks*, IEEE Transaction on Communications, Vol. 41, No. 11, Nov. 1993.

[23] P. Kim, *LLRMP: a Signalling Protocol for Reserving Resources in Bridged Networks*, in Proc. of OPENSIG '96, October 1996.

[24] P. Kim, *Link Level Resource Management Protocol (LLRMP), Protocol Specification - Version 1*, Internet Draft draft-kim-llrmp-01.ps, December 1996.

[25] Parallax Graphics, *PowerVideo700 Board*, hhtp://www.parallax.com/products/hp/xvideo700.html.

[26] Hewlett-Packard, *PA-RISC 1.1 Architecture and Instruction Set*, Reference Manual, Manual Part No: 09740-90039, September 1992.

[27] OptiVision Inc., *OptiVision Live MPEG Communication System*, User's Guide, Version 1.2f, September 1996.