2012

On delay stable communications in asynchronous networks

David Anton Miller
Iowa State University

On delay stable communications in asynchronous networks

by

David Anton Miller

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:

Ahmed E. Kamal, Major Professor

Tom Daniels

Jennifer Davidson

Doug Gemmill

Manimaran Govindarasu

Iowa State University

Ames, Iowa

2012

DEDICATION

I dedicate this dissertation work to my family . . .

to Shannon

To you I lovingly dedicate this thesis. Thank you for your patience, support, long-suffering, understanding and encouragement. You believed in me and helped me through this effort. I love you;

to Grace and Madeleine

I love each of you very much. . .

Always remember you can accomplish anything you set your mind to. . .

whatever that might be. . . ;

to Dad and Mom

I love you both. Thank you for instilling in me the importance of education and hard work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

This work exists as the result of support from many individuals.

First, I would like to express my sincere gratitude to my advisor Dr. Ahmed Kamal for his advice and guidance, and his generous giving of time and expertise to better my work.

I would like to acknowledge the members of my committee, Dr. Kamal, Dr. Daniels, Dr. Davidson, Dr. Gemmill and Dr. Manimaran for their guidance, suggestions and challenging coursework.

I appreciate Dr. Kamal and the College of Engineering at Iowa State University for their willingness to accept a distance student into the Ph.D research program.

I would like to thank my family members, my wife Shannon, and my daughters Grace and Madeleine for supporting and encouraging me to pursue this degree. Without the encouragement and support of my family, I would not have finished the degree.

Above all, I would like to express my thankfulness and appreciation to my wife Shannon for her personal support, encouragement, knowledgeable advice, and patience during this effort.

ABSTRACT

This dissertation defines a frame forwarding technique offering a fixed delay to a subclass of traffic in closed industrial control networks. In these networks bandwidth is dedicated to periodic traffic supporting tight inter-process control and control loop communication. Ideally periodic traffic arrival will have minimal delay-jitter with constant realized delays. This simplifies the implementation of connected control devices. Furthermore networks are simplified with asynchronous node and switch operation. Switch designs are simplified as there is no dependence on adjacent switch clock operation. Correct network function only relies on switches directly traversed by each flow and is not dependent on complex clock synchronization mechanisms. Existing packet scheduling schemes that attempt to minimize delay-jitter, suffer from either requiring inter-switch clock coordination (i.e. RCSP-DJ), or maintain a fixed priority so that the highest priority flows must contend without regard to past frame arrival treatment (i.e. RCSP-RJ). In this dissertation the FlexTDMA protocol is defined which supports closed network communication. FlexTDMA will be enhanced to accommodate real-world networking conditions (FlexTDMA+) and will be enhanced to support simultaneous multicast (FlexTDMA++). The FlexTDMA scheduling algorithm delivers frame data on each flow nearly at the maximal delay bound with minimal delay-jitter in an asynchronous network. Industrial control switching network systems will benefit from FlexTDMA when the complexity of system level synchronization is unacceptable, but the component switches must operate independently. FlexTDMA does not require synchronous network clock coordination and preserves the data content of frames. FlexTDMA+ includes three improvements: baseline preemption, partial baselining and baseline deadline density control, which are used to support real-world conditions of node periodic on-off transmission, clock drift, frame loss and bandwidth load. FlexTDMA++ supports simultaneous multicast under real-world conditions of switch failures, node periodic on-off transmission, clock drift, frame loss and bandwidth load.

CHAPTER 1. OVERVIEW

1.1 Introduction

Closed industrial control networks require that data be delivered from the source, which is typically a central controller, to target nodes with nearly constant delay bounds, and with minimal delay-jitter. The problem with such systems is that networked components are not synchronized, they may be distributed in a wide area, they do not use the same clock, and their clocks may exhibit drifts, in different amounts, and with different polarities. The literature contains a number of solutions to achieve bounded delay periodic traffic, and constant delayed periodic traffic. Bounded delay periodic traffic is supported in (64) using a probabilistic model, and in (56) by exchanging messages for synchronous operation. References (45) (60) (62) (61) support nearly constant delayed traffic in a synchronous network, and require message exchange to maintain a synchronous state. These solutions require a synchronous state, have probabilistic delay bounds, require message exchanges for synchronous operation, or are not suitable for sub millisecond message exchanges.

There is a need, in industrial control, to support constant delay bound communication within an asynchronous network without the use of earliness time stamping. Industrial control systems are often supported with remote computing inter-connected to multiple robots through a closed network of switches (38; 53).

The specific motivation for a constant delay bound in network communication is three fold:

1) Low Delay Bound- A low delay bound is needed for support of control loop stability related applications, and for distributed application logic requiring network frame exchange.

2) Minimal Delay-Jitter - minimal delay-jitter on communicated traffic supports control loop stability. Delay-jitter is defined as the extent of compression between any two arriving frames (7). The phase margin budget of a control loop is influenced by the variability of actual delays. A network offering a low delay-jitter value makes implementation of application control logic more straightforward, and reduces processing and buffering at the receiver.

3) Stable Delay - the realized delay bound should be stable over time, that is, concentrated about a fixed value.

We propose a protocol for constant delay delivery of data in an asynchronous network without the use of clock coordination or message time stamping.

The motivation for asynchronous networking is centered in the elimination of inter-switch clock coordination. Asynchronous switch clock operation has three major advantages. First, the design of each switch is simplified as the switching operation does not need to consider adjacent switch clock operation. Second, the correct function of the network only relies on the switches directly traversed by any flow. This makes the network more robust in that any switch fault will only result in loss of service of flows that directly utilize that switch. Third, the correct operation of the network is not dependent on complex mechanisms needed to maintain clock synchronization between end systems and switches – making the network significantly less complicated and more robust. These properties should contribute to a lower cost closed network solution.

The motivation for avoidance of an 'earliness timestamp' (3) is switch fault isolation preservation. When an 'earliness timestamp' is used, each switch is functionally dependent on the fault-free operation of predecessor switches. A switch is able to provide false earliness indications that cause false preference to frames that may compromise the frame scheduling within subsequent switches. This compromises fault independence within the switching network. Industrial control switching network systems will benefit from the use of FlexTDMA when the complexity of system level synchronization is unacceptable, but the component switches must operate independently. FlexTDMA does not require synchronous clock coordination, and limits the affect of a switch fault within the network.

To provide asynchronous nearly constant delayed communication we introduce the FlexTDMA protocol. This research started with the creation and validation of a protocol called 'FlexTDMA', and continued with validation of mechanisms to improve the constant delay bound performance of the FlexTDMA protocol. Industrial control switching network systems will benefit from the use of FlexTDMA when the complexity of system level synchronization is unacceptable, but the component switches must operate fault independently. FlexTDMA does not require clock coordination, and limits the affect of a switch fault within the network.

The FlexTDMA protocol, described later and detailed in (63), has been developed as a derivative of RCSP-RJ (7; 10) and RCSP-DJ (7; 10). FlexTDMA supports a nearly constant delay bound communication in an asynchronous network without the use of earliness time stamping or inter-switch clock coordination.

The FlexTDMA protocol works by periodically transmitting a maximally delayed frame allowing receivers to maintain a maximal eligibility time for each flow. This allows each switch to periodically hold a frame on each flow until its maximal delay bound. The result is that all frames propagate with very little delay-jitter and with a delay nearly at the delay bound. FlexTDMA requires a periodic maximally delayed frame transmission on each flow. No additional frame transmissions are utilized to support coordination between switches other than user data. Specifically, there are no clock coordination packets, and all user data is forwarded unaltered. Industrial welding robots need very detailed control both in mechanical movement precision, and timing control of the application of the welding process. Robotic control has evolved from integrated control, to remote control, and multiple remote controlled robotic systems (38; 53). Remote robot control is supported through a closed network of switches which carry controller effecter (sensor) data to (from) the robot. Integrated control incorporates the robotic control computation directly into the robot. This has the advantage that the control timing can be done with direct connections having little or no latency to the control actuators, and has the disadvantage that each robot must internally host a computer making maintenance cost higher, environment demands higher for the computer controller, and requires one computer per robot. Remote robot control has the advantage that the control computer does not need to support similar environmental conditions as the robot and is more easily maintained, but has the disadvantage that the control computer to robot communications must

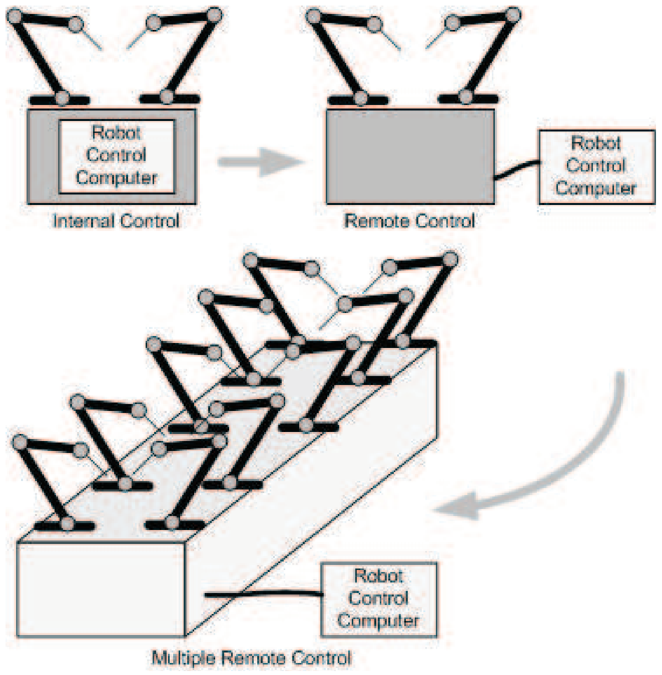

Figure 1.1 Industrial robotic control evolution from internal, to remote, to multiple remote control.

have nearly constant delay and low delay-jitter bounds. Multiple remote robot control has the additional advantage that only a single control computer need be utilized to support many robots. This reduces system cost and reduces maintenance concerns as only a single computer need be maintained.

Our research focuses on improvements to the FlexTDMA protocol to insure flows are maintained with improved delay-jitter characteristics resulting from maximal delayed frame collisions.

1.2 Research Problem

Under FlexTDMA there is no clock coordination between end nodes or switches as they operate asynchronously. Under FlexTDMA there is no clock coordination protocol employed where end nodes and switches exchange data in order to maintain a common concept of relative time. There are many clock synchronization protocols, but are not used here. Under FlexTDMA all user data is transferred unchanged and without appended fields. Under some protocols fields such as transmission time stamping or transmission earliness transmission is appended to the frame.

In order to offer maximal delay bounds each switch should hold each received frame until the arrival time that would occur had the previous switch transmitted the frame with maximal delay. As the nodes and switches operate asynchronously and user data is not modified in the network (i.e. no time stamping applied), offering maximal delay bounds is fundamentally problematic as the age of received frames is unknown.

We develop the FlexTDMA protocol to solve this problem. The FlexTDMA protocol maintains eligibility times based on arrival time of previous frames on each flow. The FlexTDMA protocol requires a baselining process to occur on each flow where a user frame is transmitted nearly at the maximal delay bound for the switch. This allows future computed eligibility times to be based on the maximal delay bound from the previous switch. The baselining process is repeated on each flow to avoid clock drift degradation of the computed eligibility times.

Support of maximal delay bounded traffic in an asynchronous network under FlexTDMA comes at a price. The performance offered under FlexTDMA is dependent on maintaining a baselined state for each flow, and the accuracy of the baselining events occurring on each flow.

In this dissertation we detail the derivation of computing the proper hold time used by each FlexTDMA switch at each frame arrival in order to achieve maximal delay bounds in an asynchronous network. Additionally we evaluate approaches to improve the FlexTDMA maximal delay bound performances.

1.3 Research Contributions

The objective of this work is to study the problem of stable delay bound data delivery in an asynchronous network. We formulate the basis for delay-jitter control in an asynchronous network through the use of a baselining technique derived from RCSP-DJ and RCSP-RJ. First, a frame scheduling protocol called FlexTDMA is defined. Second, we enhance the capabilities of FlexTDMA to accommodate real-world networking conditions in a version we call FlexTDMA+. Third, we enhance the capabilities of FlexTDMA to support simultaneous multicast in a version we call FlexTDMA++.

The scheduling algorithm we call FlexTDMA allows delivery of frame data on each allocated flow nearly at the flow maximal delay bound with minimal delay-jitter. Industrial control switching network systems will benefit from the use of FlexTDMA when the complexity of system level synchronization is unacceptable, but the component switches must operate fault independently. FlexTDMA does not require synchronous clock coordination between end nodes or switches. We evaluate the FlexTDMA protocol demonstrating the maximal delay bound with minimal delay-jitter performance.

FlexTDMA+ enhances FlexTDMA to consider real-world conditions of end node periodic on-off transmission, and network conditions of clock drift, frame loss and network bandwidth load. We propose three improvements 1) baseline preemption, 2) partial baselining and 3) baseline deadline density control. We evaluate the performance of each improvement combination in the presence of network conditions.

FlexTDMA++ expands FlexTDMA to support simultaneous multicast. We evaluate the simultaneous multicast performance under real-world conditions of switch failures, end node periodic on-off transmission, and network conditions of clock drift, frame loss and network bandwidth load. We evaluate the simultaneous multicast performance in the presence of network conditions.

1.4 FlexTDMA Compared To Existing Synchronous TDMA Solutions

FlexTDMA compares favorably to existing closed networking TDMA based communication techniques. The authors of (45) (60) (61) (62) present TDMA based communication techniques using synchronized communication between end nodes and intermediate device switches to support periodic traffic. These approaches require each connected device to maintain a synchronized state, and to transmit periodic traffic using precise timing to respect the TDMA established network scheduling.

FlexTDMA does not require that a synchronized clock state be maintained by connected end nodes. This reduces the processor capacity needs of each connected end node.

FlexTDMA does not require the connected end nodes synchronize transmissions to the schedule of the network. This reduces the needed end node processor capacity and avoids the need to utilize a specialized communication controller to manage the precision of the communication network.

FlexTDMA will scale to higher bandwidth rates as no guard band interval is needed to allocate to periodic data submissions into the network. When end node periodic transmissions are synchronized to the network, a guard band interval is provided as a window in which the emission of each periodic frame must occur. The range of this interval must be sufficient to accommodate the error in emission timing of the connected device. As the bit rate capability of the medium increases, the magnitude of the interval represents increasing loss of bandwidth. When using FlexTDMA each connected node simply offers periodic traffic in accordance with the defined traffic envelope for the flow. Any inaccuracies in the emission process by the connected end node is managed in the traffic shaping process of the FlexTDMA switch with an additional delay added to re-shape.

1.5 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we review and discuss the literature that forms the basis for the formulation of the FlexTDMA protocol. In Chapter 3, we introduce the FlexTDMA protocol, the derivation of the hold time computation and validations results of the perfoamce of the protocol. In Chapter 4, we consider enhancements to FlexTDMA to allow performances in real-world operations. In Chapter 5, we consider an enhancement to FlexTDMA to support simultaneous multiccasting. Finally, in Chapter 6, we conclude the thesis and propose potential directions for future work. Appendix A, includes a table of acronyms, and Appendix B, includes a table of symbols and variables.

CHAPTER 2. REVIEW OF LITERATURE

In this chapter we review the key concepts of network calculus that form the basis of the motivation for FlexTDMA. The network calculus scheduling establishes the basis for delay bound determination and the causes of delay-jitter. Network calculus system level issues motivate the contribution of traffic shaping in the network. We review intra-switch issues that motivate the relationship between egress port packet scheduling in a switch and the traffic shaping needed in the subsequent switch. We review the Rate Constrained Static Priority eligibility and arrival time relationship from which we derive the basis for computation of the hold time needed under FlexTDMA in an asynchronous network.

2.1 Introduction

We review key concepts relating to the network calculus analysis applied to closed networks of switches with bounded delay performance when re-shaping is applied to flows. Closed networks are widely used to provide inter-communications between control devices. These control networks are carefully constructed, designed and analyzed to determine the performance provided to each communication flow.

The closed network inter-connects a set of communicating end system applications through a topology of switches. The communication between connected end systems is supported through communication flows. Each network flow has provisioned performance measures: allocated bandwidth and end-to-end delay bound. These performance measures are important to the support of the transmitting and receiving applications hosted in the end systems of the network.

2.2 Network Calculus General Concepts

Network calculus is a form of deterministic queuing theory used to consider the bounding performance and utilization metrics of a configured network. In this section the key properties of network calculus applied to a closed network are considered.

2.2.1 Network Calculus Underlying Assumptions

Network calculus is an analysis technique applied to switching networks in order to determine the expected performance behavior of the data flows supported. Each network supports a set of data flows. The data flows have a characterization of source transmitter, and forwarding in the physical network topology. Additionally, the source transmission behaviors of each flow are characterized. Network calculus techniques are applied to insure the bandwidth of each flow can be supported, and to determine the delay bound performance of each flow to each destination.

Network calculus analysis allows bounding predictions of the actual run time performance. This predicted behavior is based on the behavior of the switches frame processing algorithms, and the transmission behavior of data flows. The application of network calculus is helpful when predictions must be made for a network.

Before introducing network calculus, we review some assumptions which are usually made in order to facilitate the application of network calculus.

2.2.1.1 Delay, Buffer, and Throughput

There are three key properties of each flow supported in a network calculus analysis: delay, buffer, and throughput (4). The delay bound describes the maximal amount of time from when a frame is submitted to the network as part of a flow until the frame arrives to its destination. There may be different delay bounds for each destination when the flow is multiplexed. Network calculus analysis is used to predict the maximal amount of buffer each flow needs at each intermediate system (switch) within a network to insure that frames are not lost due to buffer overflow. Each flow is provided a defined level of burstiness and bandwidth throughput used in the network calculus analysis. The result is that each flow is guaranteed a known level of support.

2.2.1.2 Constant Delay Line

Reference (4; 41) introduce the concept of a constant delay line. A constant delay line is a physical connection path that offers each supported flow an exact fixed delay to all frames transmitted. This is only a theoretical concept, since in practice there are technological delay variations based on design approaches and contention delay variations resulting from inter-operation with other flows. The concept of a constant delay line is useful as it presents an ideal reference point that scheduling approaches can be compared.

2.2.2 Traffic Envelopes

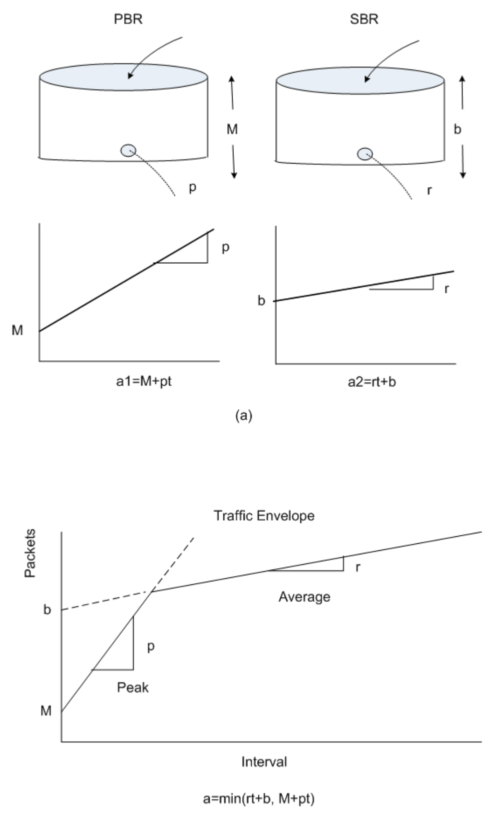

Figure 2.1 Dual token buckets implement Peak and Average traffic envelopes.

A traffic envelope characterizes a bound on traffic that may arrive in any interval of time on a given flow at a given physical point in a network. The time axis is in the interval domain rather than continuous time. That is, the traffic envelope is characterizing a limitation on the amount of arriving traffic in any interval of time, rather than a history of traffic submissions. Figure 2.1.a shows a pair of token buckets used to implement the peak-average traffic constraint envelopes. Tokens are added to the peak token bucket at rate p (average token bucket at rate r), where the peak bucket has volume M (average token bucket has volume b). An arriving frame on a flow is accepted when there are tokens in each token bucket equal to the frame size, otherwise the frame is rejected. When the frame is accepted frame size tokens are removed from each bucket. Figure 2.1.b shows two traffic envelopes that constrain traffic arrivals on a flow. Each arriving frame must conform to both traffic envelopes to be accepted. The peak and average traffic envelopes shown in Figure 2.1.b are represented as $A^*_{peak}(t) = \rho_{peak} \cdot t + \sigma_{peak}$ and $A^*_{ave}(t) = \rho_{ave} \cdot t + \sigma_{ave}$, with ρ_{peak} as the peak rate, ρ_{ave} as the average rate, σ_{peak} as the peak burst, and σ_{ave} as the average burst. When both of these traffic envelopes are enforced the constraining arrival is that shown in Figure 2.1.b. Notice that the peak and average token buckets shown in Figure 2.1.a implement the peak and average traffic envelopes. When both token buckets are active the resulting constraint is that shown in Figure 2.1.b. The peak token bucket has burst $M = \sigma_{peak}$ and rate p $= \rho_{peak}$. The average token bucket has burst $M = \sigma_{ave}$ and rate p $= \rho_{ave}$.

2.2.2.1 Traffic Constraint Envelope

A traffic constraint envelope is a traffic envelope that is established to constrain the amount of traffic accepted on a flow at a given point in a network (11). This establishes the upper limit of frames accepted on a flow regardless of the arrival pattern of frames. The network calculus analysis will be conducted as a function of the traffic constraint envelope configured in the system. This eliminates the need to know the arrival pattern of frames on each flow to each location in the network.

2.2.2.2 Traffic Arrival Envelope

A traffic arrival envelope is a traffic envelope that captures the actual arrival pattern of traffic on a flow. When a transmitter has traffic shaped arriving traffic, any arriving frames that are in excess of an established traffic constraint envelope will be either discarded or delayed until they are eligible (conform to the traffic constraint envelope).

2.2.3 Packet Scheduling

A packet scheduler is an algorithm that stores arriving frames, selects frames for transmission on a medium, and transmits them. The packet scheduler supports a fixed number of ingress flows. Each ingress flow has a traffic constraint envelope established from network calculus analysis of arriving frames. The packet scheduler implements a policy for selection of the frame to transmit from those frames being stored. For example, First In First Out (FIFO), Static Priority (SP) and Earliest Deadline First (EDF) are selection policies to determine the transmission sequence of stored frames (17). The packet scheduler has a given service rate at which frames are capable of being transmitted. This is usually used to model the guaranteed transmission bit rate offered to each of the flows at a switch output port.

2.2.3.1 Stability Function

A stability function is used to determine the 'stability' of a configured packet scheduler (9). Here the term stability refers to the aggregate service needs of supported flows relative to the service rate provided by the packet scheduler. When the aggregate rate allocation to the flows of the packet scheduler exceeds the service rate capability of the packet scheduler the scheduler will be bandwidth 'unstable'. An unstable packet scheduler will accumulate frame arrivals faster than frames are serviced. In this condition the buffer requirements will be unbounded, as frames will simply keep accumulating. Similarly, the delay for any frame arriving to the packet scheduler will be unbounded since there may be an arbitrarily large number of pending frames held in the packet scheduler for transmission.

The general criteria for stability is $\lim_{t\to\infty} \sum_{j=1}^{N} A_j^*(t) \Big/ t < 1$, where $A_j^*(t)$ is the provisioned traffic envelope for flow j (9). This can be interpreted as the time to transmit the workload arriving in time t must be less than the interval t as the duration of t increases. Notionally this means that the bandwidth used by the collection of flows must be less than the transmission capacity of the transmitter.

2.2.3.2 Schedulability Computation

Reference (9; 17; 52) present the concept of schedulability of configured packet schedulers. The concept of packet scheduler schedulability verifies that all configured flows will meet their resource allocation. Each flow is allocated an arrival traffic envelope (bandwidth) and delay bound for each frame.

The schedulability equation has the general form:

(Work that can be done in time t) >= (Work that must be done in time t).

The amount of work that can be done in time t is determined based on the service rate of the packet scheduler. When, for example the transmission capacity is 100 Mbps, then in time t seconds 100t Mbit can be transmitted.

The amount of work that must be done in time t is determined from the amount of arriving traffic on flows to the packet scheduler, the frame selection policy (FIFO, EDF, ...) and the delay bounds of each flow. Consider a flow with a delay bound of d which provides maximal traffic for a period $T = t - d$. All frame data arriving in time T must be transmitted within the interval t. This insures that the last frame will be transmitted within a duration d which is the time delay bound for the flow. This holds for all T values so that each arriving frame is transmitted within the deadline.

Reference (17) gives the schedulability equation for EDF as

$$t \geq \sum_{j \varepsilon N} A_j^* (t - d_j) + \max_{k, d_k > t} (s_k^{\max}) \tag{2.1}$$

for N flows, with a maximum frame size $s_k^{\max}$, delay bound d_k, and traffic envelope $A_k^* (t - d_k)$ for flow k. The left hand side of the equation is simply ‘t’. That is, in t amount of time, t work can be done. Or in 1 second 1 seconds worth of work can be done. The right hand side is the quantity of work that must be done in time t. The value $A_j^*(t - d_j)$ is the amount of data that can arrive on flow j in time period $T = t - d_j$. This data must be transmitted within time d_j which is the delay bound of flow j. The value $\max_{k,d_k>t}(s_k^{\max})$ is included to account for non-preemption of the EDF scheduler. This term is the maximum sized frame from any flow having a delay bound greater (lower priority than flow j) than t. Under EDF the delay bound of a flow is also the priority of the flow. All flows of equal or greater priority are considered in the first term. The maximal work to be done in time t is influenced by the maximum sized frame from lower priority frames. This accounts for the case where a maximal sized lower priority frame has started transmission, when higher priority frames arrive.

2.2.3.3 Delay Bound Calculation

Reference (9; 17) describes the packet scheduler delay bound calculation for a flow. The delay bound of frames on a flow are influenced by the relative priority of the flow, the allocation of network resource to the flow, the frame selection policy (FIFO, EDF, ...), and the transmission capacity of the scheduler. The delay bound for a flow is determined by considering the schedulability equation for the packet scheduler and algebraically solving for the delay d_j for flow j. In order to determine the delay bound equation the arrival traffic envelope equation for $A_j^*()$ must be known. A common definition is the leaky bucket $A_j^*(t) = \rho_j \cdot t + \sigma_j$. For example, the EDF scheduling policy has a schedulability equation of (2.1) which is algebraically transformed into

$$d_j \geq \frac{\sigma_j + \sum_{i=1}^{j-1}(\sigma_i + \rho_i \cdot d_i) + \max_{k>j}(s_k^{\max})}{1 - \sum_{i=1}^{j-1}\rho_i}. \tag{2.2}$$

This fraction can be interpreted as (total work provided by all higher priority flows in their delay bound) / (the residual bandwidth remaining after reduction for the rates allocated to higher priority flows). This defines the maximal delay bound for each flow that leaves the packet scheduler in a schedulable state.

2.2.3.4 Service Discipline Categories

Packet scheduler service disciplines fall into two categories: work conserving and non- work conserving.

Work Conserving Reference (7; 27; 31; 33) details the 'work conserving' property of packet schedulers. Each packet scheduler has a transmission capacity, e.g. the output port bit rate in the case of a switch output port. The term 'work conserving' refers to the conservation of the ability to do work, and a 'work conserving' packet scheduler will always transmit frames when frames are available pending transmission. The advantage of 'work conserving' is that the medium is maximally utilized and the resulting actual delay on each flow is minimized.

non-Work Conserving The term 'non-work conserving' (7; 20; 27; 33) refers to the lack of conservation of the ability to do work. A 'non-work conserving' packet scheduler may delay transmission of a frame, even when frames are available pending transmission at an idle port. The advantage of 'non-work conserving' is that the delay-jitter added to each transmitted flow is minimized. Non-work conserving scheduling policies will increase the average delay to the delay bound for the flow (33). This is acceptable in any system where the delay-jitter is the most important value used when determining system sufficiency. This is true of many real-time systems.

2.2.4 Scheduling Key Concepts

The purpose of a scheduling policy is to manage access to a shared transmission capacity by multiple contending flows. Each flow is offered a transmission service, and deterministic delay bounds. The packet scheduler operates by choosing the frame, of those frames available to the transmission service, to be transmitted in each transmission opportunity.

2.2.4.1 General Processor Sharing (GPS)

The scheduling policy called Generalized Processor Sharing (GPS) (8; 12) serves as a reference service discipline policy. GPS service divides the service capability into infinitely small service elements so that the service offered to any individual connection is smooth. Under GPS when a connection is offered a service capability r of a total service rate R, then that connection is provided at least $r \cdot T$ service in any interval of duration T. This holds independent of the presence of other connections that may also be using the service rate capability R.

GPS then serves as an idealized service discipline in that 1) it provides isolation from other connections, 2) fair service, based on a fairness index of relative provisioned-to-allocated service (31), is provided to each connection, and 3) the delay bounds experienced by a connection are only a function of the properties of that same connection (32).

Since GPS is an abstract service discipline, other service disciplines that attempt to offer service that approximates that of GPS have been introduced in the literature. These service disciplines take into consideration the atomic transmission of frames, rather than transmitting very small data quanta.

GPS – Isolation Reference (8; 12; 37) details the isolation provided by GPS. Each flow i is minimally provided $r_i \cdot T$ service in any interval T. This holds independent of the behavior of other connections also using the same service. Therefore, regardless of behaviors such as burstiness, transmission rates, or frame sizes of other flows the same level of service is always available to the flow of interest.

GPS – Fairness Reference (8; 12; 37) detail the fairness provided by GPS. GPS is ideally fair in that in any interval of time any two flows, i and j, will be provided exactly their allocated service. Reference (37) describes the fairness index $F_{i,j}$ of flows i and j for a service discipline as $\left|\frac{W_i(t_1,t_2)}{r_i} - \frac{W_j(t_1,t_2)}{r_j}\right| \leq F_{i,j}$, when flow i is allocated rate r_i and receives service $W_i(t_1, t_2)$ in interval (t_1, t_2). When the work performed over an interval of time T is always minimally $r_i \cdot T$ for flow i, the two fractions in the above equation will always equal. This follows since ideally the amount of work serviced in any interval is ideally proportional to the allocated service rate for the flow.

GPS - Delay Bounds Reference (37) details the ideal delay bound that GPS provides for any flow. The delay refers to the time from when a frame is offered on a flow, until that same frame has been fully serviced. (37) describes a general class of service disciplines Latency-Rate Servers. Servers of this class have a delay bound of $D \leq \sigma/r + \theta$ where σ is the frame size, r is the allocated rate, and θ is the latency delay to initial service. GPS service policy fits this class of Latency-Rate Servers. GPS offers service to each flow with θ equal to zero. This follows since in the interval σ/r exactly one frame of size σ will be transmitted. Figure 2.2 shows,

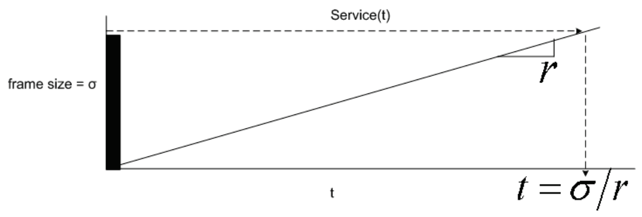

Figure 2.2 GPS service curve with no delay to service.

under GPS, a frame of size σ will be fully serviced in a time period of duration σ/r. Therefore the delay bound D will be σ/r. All other service disciplines have a non-zero θ. Figure 2.3

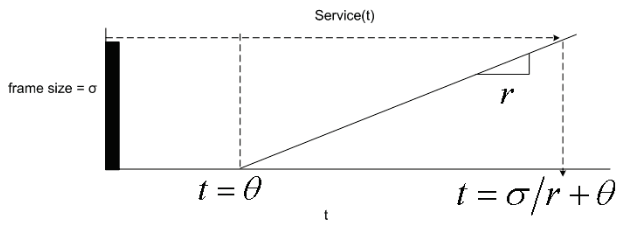

Figure 2.3 GPS service curve with no delay to service equal to θ.

shows that when a service discipline has a θ delay-to-service, the delay bound D will be $\sigma/r+\theta$.

2.2.4.2 GPS Approximation Algorithms

Weighted Fair Queuing (WFQ) Reference (18) details the Weighted Fair Queuing (WFQ) service discipline. Each flow i is configured to have a 'weight', w_i, that gives the proportion of service capacity. When the total service capacity is R then the service rate provided to an individual flow i is given as $r_i = R(w_i/\sum_{k=1}^{n} w_k)$ when there are n flows. Reference (18) describes the workings of the WFQ service policy. The server maintains a list of pending frames to be transmitted. When the next frame is to be selected for transmission, that frame is chosen which would be the first frame to complete service under a GPS service discipline.

The delay bound of WFQ is no more than one frame transmission time larger than GPS (18). The fairness property of GPS must be considered. A common misconception of WFQ is that it provides identical service to GPS that differs by only one frame (18). WFQ has a poor fairness behavior that results from providing one flow more service than GPS would during a given interval of time (18). This inaccuracy is shown to be not one frame, but instead $n/2$ frames where n flows are being supported.

Worst-case Weighted Fair Queuing (WF2Q) Worst-Case Weighted Fair Queuing (WF2Q) is an alternative version of WFQ (18). The WF2Q frame selection policy is very similar to WFQ, expect that it only chooses from those frames which under the GPS service policy would have at least started service (rather than simply the first frame to complete service under a GPS service discipline as in WFQ). From this sub-set of pending frames WF2Q then selects the frame which would be the first frame to complete service under a GPS service discipline (as was done under WFQ).

WF2Q retains the delay bound property of WFQ for any leaky bucket constrained flow. Importantly WF2Q closely approximates the fairness criteria of GPS in that the difference of transmission sequence offered by WF2Q and GPS differs by only a single frame.

Additionally WF2Q closely approximates the isolation criteria of GPS in that the service offered to any flow will very closely follow GPS independent of the behavior of other flows also using the same service.

WF2Q transmits frames maximally 1 frame time different that GPS. Therefore the fairness index of WF2Q differs from GPS by only a single frame.

2.2.4.3 Basic Scheduling Algorithms

Time Division Multiplex (TDM) The Time Division Multiplex (TDM) technique divides the transmission time over a medium into sequential durations called slots, which are fixed in size. A transmission schedule is constructed by assigning slot instances to flows. When a given slot instance is assigned to only a single flow, then that flow can transmit without colliding with transmissions from other flows. TDMA can be used to avoid collisions when using an Ethernet medium (50). This allows the support of time-critical real-time hosting on a medium.

Rate Monotonic Scheduling (RM) The Rate Monotonic Scheduling algorithm which schedules a set of tasks with fixed periods and deadlines equal to the period (1). The RM scheduling algorithm functions by selection of the ready task having the smallest period for execution. The utilization bound for RM is given as $U = n\left(2^{1/n} - 1\right)$. The utilization approaches 0.693 as n increases, which implies that a task set of any size will be schedulable when the utilization is less than this bound. (1) lists several basic advantages of the RM scheduling algorithm. RM can support highest priority tasks in the presence of transient overload conditions, and is easy to implement in software (13). The ease of implementation has made RM a very common choice as a task scheduling algorithm.

In practice the average schedulable space is larger than what the strict utilization equation would predict. Many systems have large quantities of tasks that have a utilization well above 0.693.

Static Priority (SP) Reference (17) describes the Static Priority (SP) non-preemptive scheduling technique. An SP-n scheduler supports a fixed number (n) of priority levels, with each flow statically assigned to a given priority level. A FIFO queue is maintained to store pending frame transmissions for each priority level. The frame selection works by selecting the frame which arrived first in the highest priority non-empty FIFO queue. The number of delay bounds that are available to the flows supported by the SP scheduler is limited to the number of priority levels. The SP scheduling technique is often selected as it is simple to implement, since it only requires maintaining a FIFO queue per priority level, and tracking the maximal priority having data.

The SP schedulability equation for priority p is

$$t \geq \sum_{j \in C_p} A_j^*(t - d_p) + \sum_{q=1}^{p-1} \sum_{j \in C_q} A_j^*(t^-) + \max_{r>p} s_r \qquad (2.3)$$

when there are a fixed number of static priorities, with priority 1 as the highest priority (17). Flows of priority p have delay bound d_p and have a maximal frame size s_p. The term $\max\limits_{r>p} s_r$, the largest frame of any lower priority flow, is included to account for the non-preemptive policy of the scheduler.

2.2.4.4 Network Scheduling Mechanisms

This section explores key concepts describing scheduling algorithms not based on EDF. These scheduling algorithms offer other advantages such as simplicity or delay-bandwidth decoupling.

Fair Service Curve (FSC) Reference (32) introduces the Fair Service Curve (FSC) scheduling algorithm. The primary goal of FSC is to decouple bandwidth allocation and delay requirement for scheduled flows. The WFQ scheduling algorithm couples bandwidth and delay bound as the only configurable parameter is weight. This results in the potential of allocating unneeded bandwidth to a flow in order to satisfy delay bound requirements, or to allocate a required level of bandwidth that offers a smaller delay bound than is required. Either way a resource, bandwidth or delay schedulability, is wasted. The efficient use of the medium under FSC allows higher utilization levels for real-time traffic. Figure 2.4 shows the FCS service

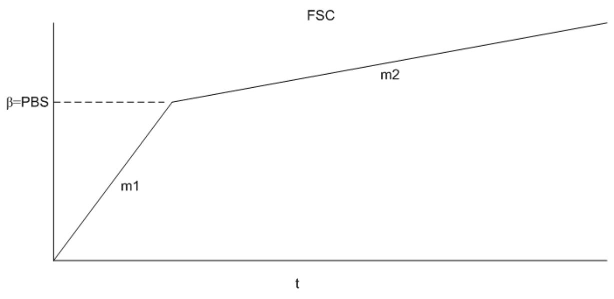

Figure 2.4 FSC service curve.

supporting rates $m1$ and $m2$. Under FSC the ratio $m1/m2$ is called the Burst Performance

Ratio (BPR), and β is the Preferred Burst Size (PBS). Following an idle period, the flow can transmit up to PBS at rate $m1$. When the traffic is bursty with bursts approximately equal to PBS, then most traffic is serviced at rate $m1$ which is BPR times faster than the long term allocated rate $m2$. Thus the performance is generally good, with minimal average bandwidth allocation.

Jitter-EDD The Jitter-EDD (3; 28; 43) traffic shapes (or partially) arriving flows to their negotiated arrival envelope. Arriving frames on each flow are held in a queue and released for EDF transmission scheduling at a time they more closely conform to the expected arrival time of the frame. The result is that delay-jitter is reduced at each scheduler.

Jitter-EDD - Pre-ahead Usage Reference (28) describes the Jitter-EDD ‘pre-ahead’ time. Each time an intermediate system (switch) in a network transmits a frame, the frame is stamped with a ‘pre-ahead’ time. This ‘pre-ahead’ time is the amount of time the frame was transmitted before the deadline of the frame. The following intermediate system (switch) holds each frame for an interval equal to the ‘pre-ahead’ value stamped on the frame. In this way the jitter is removed at each switch along a flow propagation path.

Rate Constrained Static Priority (RCSP) Reference (7; 10) describes the Rate Constrained Static Priority (RCSP) scheduling algorithm. The key features of this scheduling approach are 1) provisioning delay, throughput and loss-free communication for each flow, 2) no coupling between delay bound and bandwidth allocation for each flow, 3) minimal implementation complexity, and 4) simple admission control acceptance tests for frame arrivals on each flow.

There are two classes of RCSP: work conserving and non-work conserving (7). The work conserving RCSP more efficiently utilizes the medium, while the non-work conserving RCSP minimizes and distributes buffer requirements more evenly across a network by reshaping traffic flows.

The RCSP is composed of a rate controller and a static priority scheduler. The rate controller works by computing the eligibility time of each arriving frame, and releasing the frame to the scheduler at the eligibility time. This restores the arriving traffic on each flow to the original service definition of the flow. The reshaped traffic on each flow is then forwarded to the static priority scheduler. The SP scheduler maintains a FIFO queue per priority, and selects the first frame from the maximal priority non-empty queue for transmission. This is easily implemented, and therefore, the RCSP has low operational overhead.

RCSP – Delay Bound Reference (7) gives the RCSP delay bound schedulability equation as

$$\sum_{k=1}^{m}\sum_{j=1}^{i_k}\left(\left\lceil\frac{d_m}{X_{\min_{k,j}}}\right\rceil\cdot P_{k,j}\right)+P_{\max}\leq d_m\cdot l \tag{2.4}$$

The term d_m is the delay bound for any flow having priority level-m. The term l is the line rate transmission capability of the medium. The term $d_m \cdot l$ is the amount of work that can be transmitted on the medium in time d_m. The term $P_{\max}$ accounts for the non-preemption of the packet scheduler. It is possible that a lower priority flow (priority level greater than m) has a frame which is currently in transmission when the frames arrive on flows having level values in the range 1 to m. The term $\left\lceil\frac{d_m}{X_{\min_{k,j}}}\right\rceil \cdot P_{k,j}$ represents the maximal amount of arriving traffic that might occur on flow j within the interval d_m. The ceiling function accounts for the case when the minimum inter-arrival time $X_{\min_{k,j}}$does not divide d_m. The value $P_{k,j}$ is the frame size of flow j on priority level-k.

The left side of the equation therefore represents the maximal amount of work due within d_m, and the right side represents the capability of performing work in the interval d_m. As long as these are in balance, the scheduler is schedulable at priority level-m.

2.2.5 Traffic Envelope Utilization

The flows in a network topology are characterized for traffic utilization in terms of a traffic envelope. This allows network calculus analysis to be performed considering the topology, flow definition, flow routing, and defined traffic arrivals.

2.2.5.1 Regulation

Traffic flow regulation is the process of limiting the number of frames submitted to a physical medium on a flow at a point in the topology so that the total utilization of the flow is constrained by the definition of a traffic envelope (4; 9). This is usually used in transmitting end-systems to establish the initial flow medium utilization definition. The frames from the flow are transmitted within the limitations of the defined traffic constraint envelope for the flow. In this way the traffic utilization of the flow is established at the source end system. Figure 2.5 shows

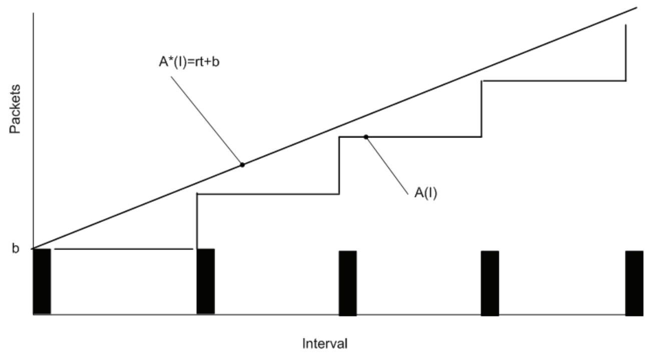

Figure 2.5 Traffic regulation is used to offer traffic for transmission that is compliant with a traffic envelope.

data being offered in transmission in accordance with the constraining traffic envelope. This is accomplished by inserting inter-frame-gaps so that the total quantity of offered data is less than the traffic envelope. This can be expressed as $A(t_1, t_2) \leq A^*(t_2 - t_1)$ for any time interval [t1, t2] (17).

2.2.5.2 Traffic Shaping

The term 'Traffic Shaping' refers to a process of accepting a flow of data and releasing that stream so that the released total amount in any interval is constrained by a defined traffic constraint envelope (4). The process works by determining what bounding traffic envelope would be needed to bound the arriving traffic flow. When a frame of the arrival envelope exceeds the actual expected traffic envelope the frame is delayed a period of time so that the new release time of the frame causes the flow to again be conformant to the intended traffic constraint envelope. Figure 2.6 demonstrates the concept of traffic shaping. When a frame

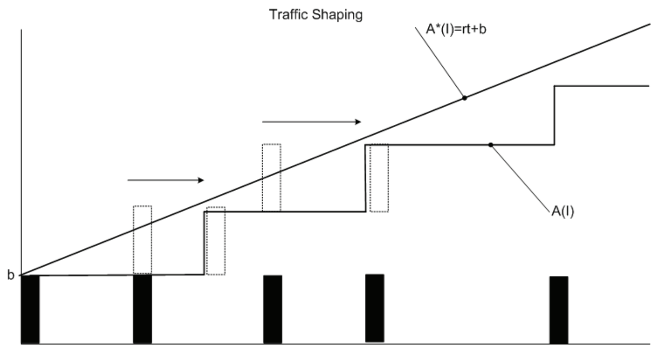

Figure 2.6 Traffic shaping is applied to arriving traffic to insure that the accepted traffic conforms with a defined traffic envelope.

is received so that $A(t_1, t_2) > A^*(t_2 - t_1)$, the frame is held a time period necessary so that $A(t_1, t_2) \leq A^*(t_2 - t_1)$. At the shifted time the frame is released for further processing. Through the process of traffic shaping it is possible to restore an arriving series of frames to the initial traffic envelope definition.

Traffic Shaping at Network Ingress Reference (48) discusses the typical traffic shaping performed at the ingress to the network, either in the transmitting end system or the first switch in the flow propagation path. The application data provided is stored and submitted in accordance with the negotiated differentiated service characterization of the flow traffic envelope. In this way the start of the flow path through the network has a defined arrival characterization. This is used as the starting definition to perform the network analysis of the flow through the network.

2.2.6 Jitter Control

There are two basic goals in jitter control (2). First the jitter control mechanism should reconstruct the original traffic constraint envelope of the flow, and second ensure the frame pattern is not distorted so much that it cannot be restored at the following node, given the nodes functional capability.

Reference (2) shows the delay-jitter results of network simulation for three cases: 1) traditional network calculus with no rate control, 2) rate control but no delay-jitter allocation, and 3) and rate-control with delay-jitter allocation. The results show that each has a normal distribution, but case three has a very small variance. This follows since most of the variance of arrival time is being removed so that each frame arrives to the destination approximately at its expected arrival time.

2.2.6.1 Jitter Control - Requires non-Work Conserving Service

Delay-jitter control within a network usually requires the service discipline used to service flows be non-work conserving (7). This follows as it is fundamentally necessary that the transmission service delay a frame until the jitter-free eligibility time is reached. Under this approach it is possible that the service transmission will be idle when there are pending frames available for transmission. This is the definition of non-work conserving.

2.3 Network Calculus System Level Issues

When network calculus is applied to a system level topology certain analysis issues arise. This section considers literature relating to the topological level analysis of closed networks.

2.3.1 Rate Latency Service Curves

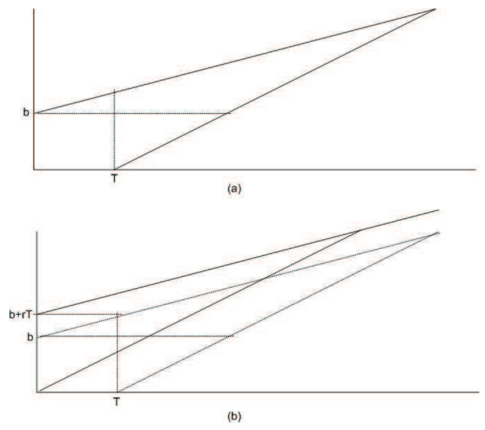

Figure 2.7 The rate latency service shown in (a) has a delay-to-service of T. The rate latency service shown in (b) has a zero, delay-to-service, but a burst of b+rT. The maximal condition at t=T in (a) is equivalent to t=0 in (b).

A rate latency service curve is characterized as $\beta_{R,T}(t) = R \cdot [t - T]^{+}$ with a bandwidth provision of R, and a maximal delay to service of T (46). This can be described graphically as shown in Figure 2.7. Figure 2.7.(a) shows a traffic envelope having a burst of b serviced by a service curve with a delay-to-service of T. The amount of traffic pending transmission at time T is maximally $b + rT$. Figure 2.7.(b) presents an equivalent circumstance at $t = 0$ as exists on (a) at $t = T$. The traffic envelope has a burst of $b + rT$ with a delay-to-service of zero. When a flow is constrained by a leaky bucket traffic envelope of the form $A^{*}(t) = rt + b$, and is serviced by a rate latency service curve of the form $\beta_{R,T}(t) = R \cdot [t - T]^{+}$, the output flow from the service curve is constrained by the traffic envelope $A^{*}out(t) = rt + b + rT$. This follows since during the interval T additional traffic will arrive at rate r. The result is that when the traffic service begins at time T the total burst presented for service will be $b + rT$.

The general delay bound for traffic offered by a flow in the interval [0,t] is given as $d \leq T + b/R$. This follows since the maximal delay to initial service might be as large as T and the delay to completion of the offered burst is b/R. A leaky bucket constrained flow must limit the offering rate of a flow to r after the initial burst is provided. The amount of pending data reduces over time since r must be less than R in order to be bandwidth stable.

This property of a rate latency service of a leaky bucket constrained flow can be utilized to perform iterative network calculus analysis (43). This is performed by progressively using the output traffic envelope of one service element as the input to the next until the propagation path of the flow is considered. In this way a bounding traffic envelope can be determined. Using this approach the burst component of the traffic envelope for each flow continues to compound as each rate latency service is encountered.

2.3.2 Traffic Shaping - Does Not Increase Worse Case Delays

It has been shown that re-shaping arriving traffic will not result in an increase in the end-to-end delay bound (10; 19; 23; 33; 44; 49). When the original traffic envelope of a flow is conformant to a rate − burst traffic envelope, the extent of early arrival to a switch relative to the eligibility time of the leaky bucket constraint is bounded by the early departure of the frame from the packet scheduler in the previous switch. Forcing a reshaping event at each frame arrival will require a holding time bounded by the earliness of the frame. Therefore, the total of the packet scheduler wait time, and the re-shaping hold time in the following switch is bounded by the maximal delay of the packet scheduler of the transmitting switch.

2.3.3 Delay-Jitter is Limited to Maximal Delay Bound

Reference (24) gives a very important but seemingly simple result, in Theorem 1, 'Delay-Jitter Bound'. The bound on the delay-jitter increase on a flow is limited to the bound on the difference between the maximum and minimum delay time through an element that services the flow. This result applies to any scheduling mechanism in an element that can provide a guaranteed upper and lower bound on delay time to service for the supported flows. Therefore elements that schedule flows using a packet scheduler, such as EDF, SPn, and RPQ, support this observation. Additionally this concept can be applied to an unknown scheduling technique that has a requirement to offer a given delay bound.

2.3.4 Output Burstiness of Scheduled Flows Through an Element

Output flow burstiness is directly a function of the maximal delay experienced through the node (24; 33). The 'node' in question can be any sub-element of an implementation that is able to guarantee a delay bound to a flow. This might be an entire switch, a packet scheduler, another sub-part of an implementation, or a sub-network of a larger network. The additional burst component of a flow traffic envelope is based only on the delay bound through a sub-path without regard to the algorithms that generate the delay.

2.3.5 Property of Minimum Service Curves

The 'Property of Minimum Service Curves' states that the maximum and minimum service curve resulting from a flow traversing two elements each guaranteeing a minimum and maximum service curve is the minimum of the two maximum service curves and the minimum of the two minimum service curves respectively (24). This is important when considering the effective end-to-end service provided to a flow. Notice that this end-to-end service is limited by the minimal service at any processing step.

2.4 Intra-Switch Delay Issues

When considering the application of network calculus analysis to a closed switching network intra-switch issues arise. This follows since the flow management applied at one switch affects the resulting behavior at receiving switches. Thus there is a relationship between connected switches. By taking advantage of this relationship system level analysis can be simplified.

2.4.1 Earliness Definition

The term ‘earliness’ is defined as the amount of time between when a frame is selected for transmission at a switch packet scheduler relative to the frame deadline time (33; 47). The concept of earliness is strongly related to the delay-jitter induced in the packet scheduler when transmitting the frame. The concept of earliness is used in consideration of traffic re-shaping at subsequent switches in the propagation path.

2.4.2 Switch Time is Holding and Wait Time

The terms ‘holding’ and ‘wait’ time are defined as components of the total time needed to forward a received frame within a switch (7; 10). The total time through a switch that re-shapes arriving traffic to the flows original traffic envelope definition is the holding time needed to re-shape, and the wait time experienced in the output port scheduler. Figure 2.8

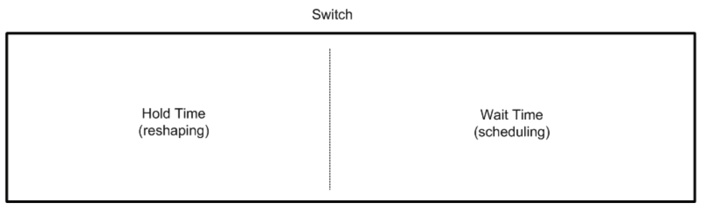

Figure 2.8 Traffic shaping switch implements a Hold Algorithm (for reshaping) and a Wait Algorithm (for scheduling).

shows the two time phases of data traversing a switch. The ‘holding’ time is defined as that period of time an arriving frame is held before the frame is made available to be scheduled in an output port scheduler. The 'wait' time is defined as the period of time an arriving frame resides in the switch after being provided to an output port scheduler for transmission. This 'wait' time is bounded by the packet scheduler delay bound for the flow.

2.4.3 Hold Time Is Limited to "ahead of" Time

Reference (7) establishes the concept that the 'hold' time needed in a switch is limited to the 'ahead of time transmission' in the previous switch. The fundamental purpose of a packet scheduler at the output port of a switch is to multiplex multiple asynchronous flows and transmit each flow within a delay bound. The result is that some flows may be transmitted sooner than the delay bound for the scheduler depending on the arrival sequence of frames from other flows. When holding arriving frames to completely re-shape the arriving traffic stream, and when the traffic stream was fully regulated as it arrived to the previous scheduler using any policy offering bounded delay service, then the time needed to 're-shape' the arriving traffic is exactly the difference between the bounded delay of the previous scheduler and the actual delay in the previous scheduler. This time is sometimes called the "ahead of" time. That is, the amount of time the frame was transmitted before the maximal deadline in the previous scheduler.

2.4.4 Wait{node(i)}+Holding{node(i+1)} = Delay Bound{node(i)}

Reference (7; 10) develops a very important result relating to the wait time of a switch and the hold time needed in the subsequent switch in the flow propagation path. In this context, each switch along a flow propagation path re-shapes the arriving flow to its original traffic constraint envelope. The key observation is that the delay bound from entrance to the packet scheduler in switch(i) to the time the frame leaves the re-shaping function in switch(i+1) is exactly equal to the delay bound of the packet scheduler of switch(i).

Figure 2.9 shows the delay bound from the ingress to the packet scheduler at the output of a switch to the completion of the traffic shaping holding time in the following switch. Notice that

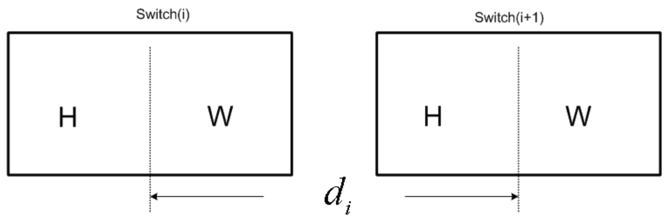

Figure 2.9 The delay bound from the ingress to the packet scheduler in switch (i) to the ingress to the packet scheduler in switch (i+1) is the delay bound of the flow in the packet scheduler of switch(i).

the amount of time switch(i+1) holds the frame on a flow for re-shaping is not dependent on the function of switch(i), but instead only on the arrival time of the frame. Thus the re-shaping function need not be dependent on any frame-stamp applied to the frame by a previous switch. This is important since the re-shaping function is not dependent on the correct function of the previous switch.

2.4.5 Delay-Jitter is Limited to Last Switch Only

Reference (7) makes an important observation that the delay-jitter of traffic offered to the final end system destination is limited to the delay-jitter of the packet scheduler of the final switch preceding the destination end system. This follows from the concept of hold time limited to "ahead of" when applied to a series of switches that re-shape each flow. The traffic envelope of the flow will be fully restored as it arrives to the last switch. The resulting delay-jitter will be limited to the traffic envelope distortion experienced at the transmission scheduler of the last switch. When the scheduling technique in the last switch minimizes the delay-jitter, the destination end system can implement much more straightforward mechanisms to remove delay-jitter of each flow.

Figure 2.10 shows the delay-jitter induced by the final wait time of the packet scheduler of the final switch. The holding functions of each switch fully re-shape each flow, and therefore eliminate all delay-jitter. The final wait function (the egress port packet scheduling algorithm) introduces delay-jitter as pending frames contend for the transmission. The delay-jitter is limited only to the delay-jitter added in the final switch, since there is no means for the

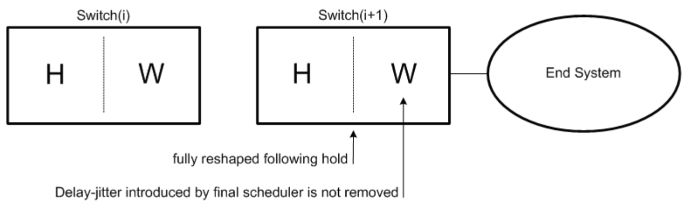
Switch(i)
H
W
Switch(i+1)
H
W
End System
fully reshaped following hold
Delay-jitter introduced by final scheduler is not removed

Figure 2.10 When each switch in the network re-shapes arriving traffic, the delay-jitter on traffic arriving to connected end systems is limited to the delay-jitter introduced in the packet scheduler of the last switch.

switching network to eliminate delay-jitter before the final connected end system receives the frames. As the delay-jitter should be minimal, the end system should be able to efficiently eliminate the delay-jitter.

2.4.6 End to End Delay

The delay bound of a flow from point-of-origin to each destination is called the end-to-end delay bound. Fundamentally, although each switch within a network offers a local switch delay bound, the network analyst is interested in the total end-to-end delay.

2.4.6.1 End to End Delay - Sum of Node Delay Bounds

A simple but very important observation is that traffic shaping a flow at each node along a flow path in a network restores the traffic envelope back to the original definition (14; 19). When all switches receiving a given flow perform network analysis using the original traffic definition there is no sequential dependence on the order in which network analysis is performed. This reduces the complexity of network analysis of a complex network.

Figure 2.11 illustrates the concept of analyzing the required hold and wait time needed for traffic emitted from a switch port. Notice that switch(i) can be analyzed independent of all other switches. Thus, the total delay through $switch(i) = W(i) + H(i+1) = d(i)$. In this way the switches of the network can be considered in any order.

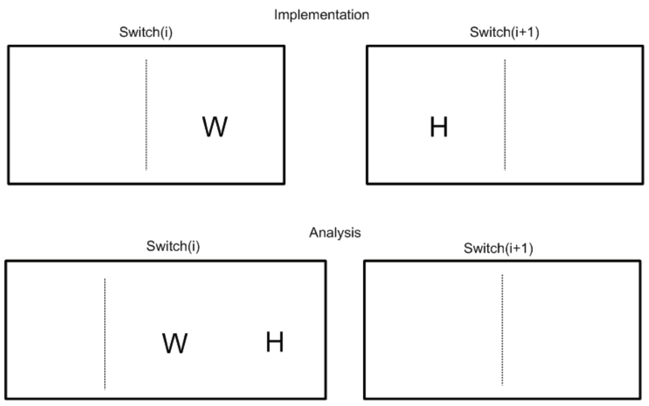

Figure 2.11 The analysis of delay through each switch can be performed as the sum of the wait time in the scheduler plus the hold time needed to fully restore the flow traffic envelope. This insures that the network analysis delay / delay-jitter analysis can be performed iteratively switch-by-switch.

2.4.7 Delay-Jitter Definition

Reference (54) provides a definition of jitter as it relates to analysis of communication in the real-time networks. The delay-jitter is defined as the maximal reduction in the inter-arrival time of transmission occurrences of frames on a periodic flow. (54) defines delay-jitter formally as $J_i = |(T_{i+1} - ET_{i+1}) - (T_i - ET_i)|$, where ET is the eligibility time of the frame and T is the transmission time. Thus the delay-jitter is the compression between the transmission times of any two successive frames relative to the allocated frame period of the flow.

2.4.7.1 Rate-Jitter in Rate Controlled Networks

Reference (7) describes a rate-controlled static priority rate-jitter scheduling algorithm. Switches implementing this policy first rate control arriving frames and then schedule eligible frames for transmission using a static priority scheduling policy. The rate-controlling algorithm maintains a minimum inter-arrival time between received frames equal to the long term rate allocation of the flow. Thus the initial transmitting end system traffic envelope is partially re-constructed. The resulting traffic envelope from the rate-controller regulator will be limited to an envelope of $A^*(t) = 1\,frame + rT + b$. The extra bursting component b is induced by less than minimum inter-arrival time spacing of the rate-jitter policy. This follows as the rate-jitter policy only restores the traffic envelope over a long period of time, rather than on each frame arrival.

2.4.7.2 Delay-Jitter Controlled Networks

A delay-jitter regulator fully restores the arriving flow traffic envelope to the initial traffic constraint envelope of the original transmitting end system at the entrance of the network (2; 7). This can simplify system analysis since the traffic envelope applied against each switch packet scheduler is common throughout the network and does not depend on the processing delays prior to the packet scheduler.

2.5 Chapter Summary

We have reviewed key concepts relating to the network calculus analysis applied to closed networks of switches with bounded delay performance when re-shaping is applied to flows at each switch.

In Section 2.2 we reviewed general network calculus concepts. The network calculus analysis provides performance descriptions of flow end-to-end delay bounds, switch internal buffering needed to avoid frame loss, and validation that throughput bandwidth can be provided to each flow.

Network calculus analysis utilizes traffic envelopes to characterize the amount of traffic that each flow may be using. Using these traffic envelope definitions a characterization of the contention each data flow will encounter at each switch is determined.

Network calculus is an analytic tool to model the transmission behavior of the packet scheduler at each switch output port. From this model the delay performance of data flows at each switch output port is determined. A packet scheduler is characterized by the frame selection policy used, the internal buffering supported, and the frame transmission rate. The goal of packet scheduler analysis is to determine the delay bound of each flow supported, and to insure that adequate packet scheduler buffering is allocated so that no frames are lost due to over-subscription. The use of a packet scheduler allows each switch in the network to offer deterministically bounded delay to service by scheduling arriving flows. This insures that the total end-to-end service is bounded by a delay limit.

The packet scheduler analysis insures that the data flow supported is bandwidth stable. That is, the service rate of the packet scheduler is sufficient to support the data arrival of the data flows insuring finite buffering and delays bounds. Each packet scheduler frame selection policy type has a specific equation used to determine schedulability.

The packet scheduling algorithm called General Processor Sharing (GPS) was characterized for its key performance criteria: isolation, fairness, and delay bounds. The GPS policy was contrasted with Weighted Fair Queuing (WFQ) in terms of these key criteria.

Traffic envelope regulation and traffic shaping were reviewed. Traffic shaping reduces delay-jitter impact while maintaining an end-to-end maximal delay bound.

In Section 2.3 we reviewed network calculus system level issues. The system level impact of switch traffic shaping was evaluated. Traffic shaping reduces buffering, and simplifies network level evaluation as each traffic envelope is restored at each switch.

In Section 2.4 we reviewed inter-switch issues in network calculus analysis of a switching network. The switch traffic shaping processing was characterized as 'hold time' (time needed to reshape the arriving data flow) and 'wait time' (time the frame spends pending in the packet scheduler prior to transmission). The degree of earliness of a frame was shown to relate to the duration of 'wait' time needed in the subsequent switch to reshape the data flow of the frame. The delay bound of a data flow in a switch was shown equal to the wait time in that switch and the hold time in subsequent switch. This fixes the delay bound from the entrance to the packet scheduler in one switch to the departure from the reshaping function in subsequent switch. The delay-jitter was shown to be limited to the delay bound of the packet scheduler in the last switch when a series of switches perform reshaping of a data flow. Thus end-to-end delay is equal to the delay bound through switch n-1 and the actual delay of the final switch n.

The application of traffic shaping to a closed network and associated analysis concepts introduced will be advantageous to networks supporting control loop functions requiring defined delay bounds and minimal delay delay-jitter. Closed networks often use a network of switches supporting hosted applications in end nodes that benefit by low delay-jitter and stable delay.

This minimizes system testing and improves phase-margin of control loops existing in the system.

We characterized the traffic shaping relationship to hold and wait time in connected switches, in that the reduction in hold time in one switch contributes directly to wait time in the next. This fixes the delay bound from traffic shaping completion in one switch to traffic shaping completion in the subsequent switch.

Network analysis can be simplified by avoiding tool management of traffic envelope status at each switch. Instead each traffic envelope delay contribution by each switch is known. When network performance is related to flow delay bounds rather than actual contention the flow behavior is more a function of required delay bound than it is run time contention. This allows system analysis based on the required delay bound rather than the actual delays experience during run-time.

The closed network inter-connects a set of communicating end system applications through a topology of switches. The communication between connected end systems is supported through communication flows having bandwidth and delay bound needs. These are satisfied in the network by bandwidth allocation and network analysis applied to insure that hop-by-hop and end-to-end delay bounds can be supported. Each network flow has provisioned performance measures: allocated bandwidth and end-to-end delay bound. These performance measures are important to the support of the transmitting and receiving applications hosted in the end systems of the network. These performance measures allow end system application designers to insure the supported algorithms will function as required.

CHAPTER 3. FLexTDMA

In this chapter we introduce the protocol we call FlexTDMA. We review the functional performance capabilities of FlexTDMA, derive the hold time formulation needed in an asynchronous network, formally define the protocol and provide an evaluation.

3.1 Introduction

There is a need in closed industrial control networks, which are mission-oriented local networks, for frame forwarding techniques that offer a fixed delay to a subclass of periodic traffic (55). Industrial control networks have a portion of the exchanged bandwidth dedicated to support tight inter-process control and control loop communication.

Utility of Research: Other packet scheduling schemes that attempt to minimize delay-jitter, suffer from either requiring inter-switch clock coordination (i.e. RCSP-DJ) (rather than asynchronous operation), or maintain a fixed priority so that the highest priority flows must contend without regard to past frame arrival treatment (i.e. RCSP-RJ). Techniques utilizing an 'earliness timestamp' (3) are avoided in order to preserve the independence of switches.

Proposed Research: We introduce the FlexTDMA scheduling policy which is intended to offer nearly RCSP-DJ service in an asynchronous network, i.e., without switch coordination. This is accomplished by a periodic maximally delayed frame transmission on each flow (called baselining). This allows an RCSP-RJ policy to closely conform to the RCSP-DJ performance properties. The expectation is that the FlexTDMA protocol, if deployed, would be in the context of a mission oriented industrial control network switch, rather than a more general communication network switch (e.g. Cisco commercial switches). The FlexTDMA function offers value that is probably isolated to limited closed industrial networks with well defined delay requirements established to insure a functional system.

The remainder of this chapter is organized as follows. In Section 3.2, we describe the assumed networking properties and switching traffic regulations employed. In Section 3.3, we detail the proposed FlexTDMA switch frame processing policy. In Section 3.4, we discuss the issues relating to baseline cascading in the FlexTDMA protocol. In Section 3.5, we list observations of FlexTDMA. In Section 3.6, we detail the performance of FlexTDMA. The conclusions of this study are given in Section 3.7.

3.2 Background

In this section we describe the networking model and assumptions, and the switching traffic regulations used to manage network flows.

3.2.1 Network Model and Assumptions

We assume a network of switches that operate asynchronously - that is, independently with no clock coordination. Therefore each switch has no direct knowledge of the internal clock timing of any adjacent switch, and there is no direct way to validate the timing of each received frame other than evaluation of the traffic envelope status of each received flow to the switch. Additionally there is no 'earliness timestamp' exchanged between switches in order to preserve switch independence (33).

We assume each switch is designed to operate with a clock rate that has a bounded part-per-million (ppm) rate deviation from the ideal clock frequency. When a switch operates with a clock rate different from the ideal rate, the regulated transmitted traffic rate is modified, and the perceived arrival rate of traffic is modified.

3.2.2 Policing and Regulation

To provide deterministic switch scheduling guarantees arriving traffic must be constrained to a defined traffic envelope (7; 17). This can be accomplished through policing traffic in excess of the desired traffic envelope, or by restoring the arriving traffic envelope through traffic shaping (22). When a traffic shaping algorithm is used to completely eliminate delay-jitter the traffic pattern of each flow will be fully restored to the initial transmission definition at the network ingress (7; 10). This allows per switch analysis of delay bound and avoids classic cyclic dependency analysis issues (20; 5). A reshaping algorithm can be used to restore a flow so that all frames are separated by the initial minimum frame transmission interval (10). When a flow is bounded by a rate, r, maximum burst size, b, and traffic envelope

$$A^{*}(I) = r \cdot I + b \tag{3.1}$$

within any interval I, the process of reshaping a flow does not increase the maximal delay bound from the ingress of the previous switch packet scheduler to the egress of the reshaping process in the current switch (10; 20). This follows as the holding time in the regulator is limited to the ahead-of-time transmission in the previous switch packet scheduler. A re-shaping regulator is used to determine the eligibility time (ET) of a frame so that the flow conforms to the initial traffic definition (4; 7; 10; 51).

3.2.3 Delay-Jitter Regulator

In (7; 10), the authors establish the definition of a delay-jitter regulator. The values ET_j^k and AT_j^k are the eligibility and arrival times of frame k at switch j, respectively. The initial condition

$$ET_0^k = AT_0^k \tag{3.2}$$

assumes the initial arrival sequence to the first switch is conformant to the delay-jitter free traffic envelope. Under RCSP-DJ ET_j^k is recursively calculated as

$$ET_j^k = ET_{j-1}^k + d_{j-1} + \pi_{j-1,j} \tag{3.3}$$

as shown in Figure 3.1. d$_{j-1}$ is the delay bound of eligible frames on this flow at the previous switch j-1. $\pi_{j-1,j}$ is the transmission time on the medium from switch j-1 to switch j.

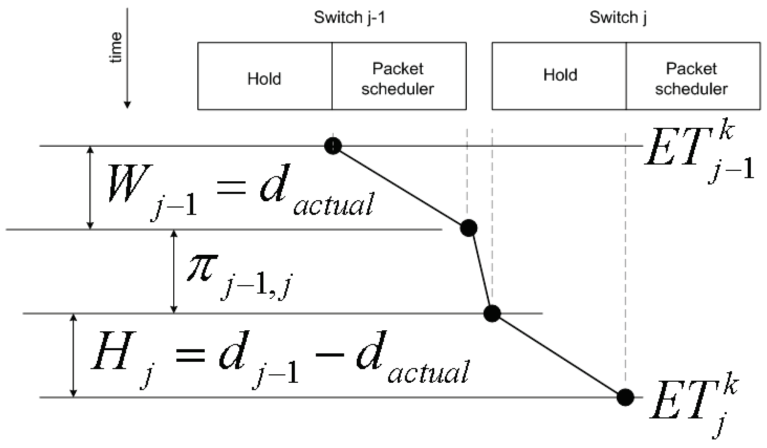

Figure 3.1 Frame k switch j-1 eligibility to switch j eligibility. The time between eligibility times in successive switches is the sum of the wait time, line time, and hold time.

3.2.4 Full Restoration Following Regulation

In (7; 10), the authors establish the eligibility times, under RCSP-DJ, of two successive frames arriving to switch j which can be represented as $ET_j^{k+1} - ET_j^k = ET_{j-1}^{k+1} - ET_{j-1}^k$ by rearranging equation (3.3). By inductive application of (3.2) $ET_j^{k+1} - ET_j^k = AT_0^{k+1} - AT_0^k$. Therefore the initial arrival pattern to the first switch is fully restored at each switch. This is only possible assuming that the eligibility time in the previous switch ET_{j-1}^k is known. This is not possible in an asynchronous system where switch clock timing is not exchanged. In (7; 10), the authors describe a rate-jitter reduction, under RCSP-RJ, that depends only on the eligibility times of the current switch. While rate-jitter restoration will improve the delay and delay-jitter, it will not fully restore the traffic envelope of each flow to the original definition.

3.3 FlexTDMA

In this section we introduce the FlexTDMA protocol. We explain the derivation of flow regulation within the asynchronous network, the baselining process, and provide a formal description of FlexTDMA.

3.3.1 Restoration Based on ET in Previous Switch

Direct application of equation (3.3) is not possible in an asynchronous network. Notice that the regulation equation (3.3) is based on knowing the three right side terms of the equation. The value $\pi_{j-1,j}$ is fixed at $\mathrm{P}_k/\mathrm{l}_j$, the frame k size divided by the line rate of switch j. The value d_{j-1} is fixed in a static configuration based on the schedulability delay bound of the packet scheduler supporting the flow containing frame k in the previous switch j-1. This is determined based on the off-line schedulability delay bound analysis of the previous switch configuration. ET_{j-1}^{k} is not directly known to switch j. The value ET_{j-1}^{k} is exactly known only within the clock domain of switch j-1. Recall that each switch in the network operates asynchronously with no clock coordination with other switches.

3.3.2 Reimplementation of Regulation Using Token Bucket

Here we show the derivation of flow regulation in an asynchronous network. Recall that $\pi_{j-1,j}$, the transmission time from switch j-1 to switch j, is fixed. From this we know that all the bits of the frame will arrive to switch j exactly $\pi_{j-1,j}$ after transmission was initiated at switch j-1 so $ET_j^k - ET_{j-1}^k = d_{j-1} + \pi_{j-1,j}$. The time from ET_{j-1}^k to ET_j^k is composed of W_{j-1}^k, the actual wait time in the packet scheduler of switch j-1, the transmission time $\pi_{j-1,j}$, and H_j^k, the actual holding time within the regulator of switch j so that $ET_j^k - ET_{j-1}^k = W_{j-1}^k + \pi_{j-1,j} + H_j^k$. Thus $H_j^k = d_{j-1} - W_{j-1}^k$.

When using a token bucket as a regulator (*limits arrivals to* (3.1) *during any interval* I), the value W_{j-1}^k can only be detected as the difference in arrival times of two successive frames to switch j of AT_j^k and AT_j^{k-1}. The exact jitter applied to frame k can be detected when frame k-1 was delayed the maximal amount (the delay bound of the flow). Once a maximally delayed frame is received from switch j-1, the ET_j^k will be computed as $\max\left(AT_j^k, ET_j^{k-1} + X_{\min} \cdot (1 - clockDrift_{ppm})\right)$, where X_{min} is the minimum frame inter-arrival time on the flow. FlexTDMA maximally delays frames on each flow through a process called *flow baselining*.

3.3.3 Formal Description of FlexTDMA

```
switch switch frame event on flow k do
    case Frame Arrival
        frame ← retrieve frame from input port
        AT[k] ← now
        ET[k] ← max(AT[k], ET[k] + $X_{\min} \cdot (1 - clockDrift_{ppm})$)
        frame deadline ← ET[k] + flow(k) delay bound
        enqueue(Eligibility Queue, frame)
    case Frame Eligibility
        frame ← dequeue(Eligibility Queue)
        store frame in FIFO or $\text{Flow}_{01}$
    case Frame Transmission Completion
        if $flow_{01}$ queue head scheduled time > now then
            frame ← dequeue($\text{Flow}_{01}$ Queue)
            Baselined[flow k] ← true
            transmit frame
        else if not empty(FIFO Queue) then
            frame ← dequeue(FIFO Queue)
            transmit frame
        end
    endsw
endsw
```

Algorithm 1: FlexTDMA

The FlexTDMA algorithm has three processing stages in the switch, as shown in Algorithm 1: 1) frame arrival to the switch, 2) frame eligibility, and 3) frame transmission. When a frame arrives at the input port of a switch, the arrival time (AT) is recorded, and the eligibility time (ET) of the flow is determined. The frame transmission deadline is determined from the eligibility time and the flow deadline bound in the switch. Finally the frame is stored pending eligibility. A frame, held until its eligibility time, is scheduled for transmission on flow_{01} at its deadline time (flow baselining) when either the flow is not currently baselined or the baseline deadline has been exceeded, and flow_{01} is available. Otherwise the frame is placed into the appropriate FlexTDMA priority FIFO queue pending transmission. When a switch output port becomes idle, the switch selects a frame from flow_{01} queue when there is a frame present which is scheduled in the window [now, now + frame_{time}]. Alternatively, a frame is selected from the head of a FIFO queue (if any).

3.3.4 FlexTDMA Frame Eligibility Queuing Decision

The actions taken by the FlexTDMA protocol at frame eligibility are characterized in Table 3.1. This table is used to determine which queue an eligible frame will be stored in pending transmission (either the flow$_{01}$ or the FIFO queue). This table shows the frame eligibility actions for all 3 flow baselined states of the FlexTDMA Protocol.

When the flow is not baselined or has a baseline deadline exceeded and flow$_{01}$ has a transmission opportunity at the flow deadline the frame will be scheduled in flow$_{01}$ at the deadline time. Otherwise the flow will be placed in the FIFO queue.

Table 3.1 FlexTDMA Queuing Decision Table

	Flow Baseline Status	**Flow01 Availability**	**Queue**
1	Not Baselined	**Yes**	Flow01 at deadline
2	Not Baselined	No	FIFO
3	Baseline Deadline Exceeded	**Yes**	Flow01 at deadline
4	Baseline Deadline Exceeded	No	FIFO
5	Baselined	NA	FIFO

3.3.5 Flow Baselining

Flow baselining is included in FlexTDMA to insure the actual wait time W of a frame is equal to the delay bound for a few sparse frames on each flow. As each switch operates with an independent clock, when a token bucket is used at each switch (for independence) there is no way to know the delay time through the scheduler of the previous switch. Here we introduce the concept of a ‘baselined’ flow.

Definition *Baselined flow: A flow on which a frame has been recently transmitted at the delay bound, and each frame has been received before its eligibility time.*

A flow is considered ‘baselined’ since subsequent switches experience the maximal relative receipt time. A baselined frame, transmitted at the maximal delay bound, cause a maximal series of eligibility times in the downstream switch. When a flow is baselined, most of the frames on the flow can be transmitted with any delay less than the flow delay bound. When switch i transmits a frame on a flow j at the deadline time for the packet scheduling algorithm the flow is considered baselined at switch i. Subsequent frames transmitted by switch i will be held at switch i+1 a time period so that the maximum delay bound of switch i is enforced (traffic shaping).

When a flow is not baselined by transmitting with a maximum delay at the scheduler of a switch, the delay bound can range from Π to $\sum_{i=1}^{n-1} d_i + \Pi$, where Π is the sum of the transmission times at all switches. The delay-jitter is limited to the wait time in the packet scheduler of the final switch (7; 10). When a frame is received to a baselined flow, the frame can be transmitted at any time between the eligibility time of the flow and the transmission deadline.

A minimum baseline interval (BI) is enforced for each active flow. The acceptable interval is a function of the delay-jitter performance need of each flow given the clock drift rate capability of the switch.

A baseline deadline (BD) is established for each active flow. When a flow is baselined the baseline deadline for that flow is set to the current time plus the baseline interval. The flow will be re-baselined before the baseline deadline to limit the effect of clock drift on the flow.

Frames selected for baselining, using flow$_{01}$, are stored in a sorted queue of scheduled frame transmissions. Flow$_{01}$ is considered available when no frame is currently scheduled for transmission within one flow$_{01}$ transmission period of the desired deadline time in order to respect the minimum transmission period of flow$_{01}$. Each element has a frame pending transmission, and a scheduled transmission time. The selection logic used to determine which arriving frame to baseline must be carefully considered given the limited transmission opportunities of flow$_{01}$. A FIFO queue will be used to store pending frames which are not selected for baselining.

Figure 3.2 demonstrates the baselining process. Frame j (fr$_j$) through fr$_{j+3}$ arrive to switch 1 (sw$_1$) at a period P. Sw$_1$ will consider the flow non-baselined at the arrival of fr$_j$ (assumes a long silent period preceding fr$_j$), and will therefore attempt to transmit fr$_j$ at the deadline time. However, due to contention sw$_1$ is unable to transmit the frame at the deadline time, thus the frame is scheduled for transmission in a standard FIFO queue and will be transmitted at a delay less than the maximal delay bound. Fr$_j$ is then received to sw$_2$ which considers the flow non-baselined and therefore establishes the eligibility time as the fr_j receipt time. Assume that sw_2 was able to transmit fr_j at the deadline time so that sw_2 will then consider the flow baselined.

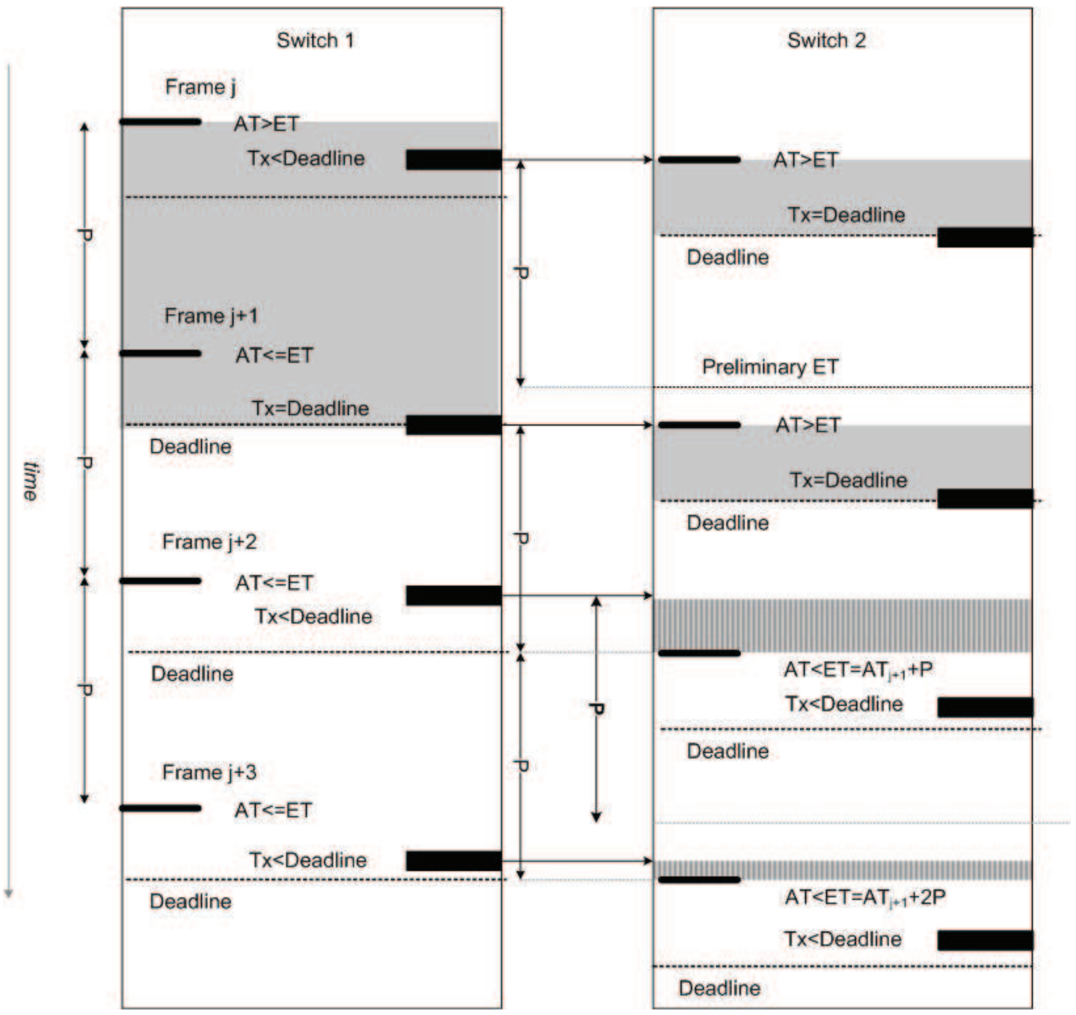

Figure 3.2 FlexTDMA uses a per switch baselining strategy in which flows are periodically baselined by maximum delay transmission. Transmissions on baselined flows can occur at any time up to the deadline of the flow. Non-baselined time periods are shown in grey, and time periods in which frames are held for eligibility are shown in stripped black.

Fr_{j+1} arrives to sw_1 one period P following fr_j. Recall, the flow is yet non-baselined in sw_1 as it was not possible to transmit the fr_j at the deadline. Therefore, fr_{j+1} is scheduled for transmission at the deadline time (it was possible to schedule fr_{j+1} at the deadline), and the flow will be considered baselined by sw_1. Sw_2 receives fr_{j+1} more than one period P following the eligibility time of fr_j. Therefore, sw_2 will consider the flow non-baselined and the eligibility time of the flow will be adjusted to the arrival time of fr_{j+1}. Assume that sw_2 was able to transmit fr_{j+1} at the deadline time so that sw_2 will then consider the flow baselined.

Fr_{j+2} arrives to sw_1 one period P following fr_{j+1}. Recall, the flow is considered baselined in sw_1. Therefore, fr_{j+2} can be scheduled for transmission any time up to the deadline time, since the transmission of fr_{j+1} baselined sw_1. Sw_1 transmits fr_{j+2} with a small delay following the eligibility time. Sw_2 receives fr_{j+2} before the eligibility time of fr_{j+2}. Therefore, sw_2 holds fr_{j+2} until the eligibility time and transmits fr_{j+2} as early as possible before the deadline time.

Fr_{j+3} arrives to sw_1 one period P following fr_{j+2}. Recall, the flow is considered baselined in sw_1. Therefore, fr_{j+3} can be scheduled for transmission any time up to the deadline time. Sw_1 transmits fr_{j+3} as early as possible following the eligibility time, but with a relatively large delay due to contention with other flows. Sw_2 receives fr_{j+3} before the eligibility time of fr_{j+3}. Notice that sw_2 receives fr_{j+3} more than one period after fr_{j+2}. However, no baseline establishment is needed as fr_{j+3} arrives to sw_2 before the eligibility time of fr_{j+3}. This follows as sw_1had previously established a baseline for the flow by transmitting fr_{j+1} at the deadline time causing sw_2 to baseline given the new eligibility time. Therefore, sw_2 holds fr_{j+3} until the eligibility time and transmits fr_{j+3} as early as possible before the deadline time.

3.3.6 Flow Transition to a Non-baselined State

When a frame k arrives on flow n with an arrival time of $AT_n^k > ET_n^k$ where ET_n^k is the eligibility time of frame k, the flow will be considered non-baselined. The frame will arrive with $AT_n^k > ET_n^k$ only when the actual delay achieved in the packet scheduler of the previous switch was nearly equal to the maximum delay bound, and the relative clock error was sufficient to cause the receiving switch to perceive that the frame was received beyond its eligibility time. This occurrence will be very rare, other than an intentional baselining event in the previous switch since in practice most frames are transmitted with a delay much less than the maximal computed delay bound for the output port.

3.3.7 Updating Baseline Deadline

When a baselining transmission occurs that precedes the ideal baseline time (the delay bound of the flow) the baseline deadline will be reduced by the time needed to accumulate the error in baselining time. The error in baselining time is limited to the drift error that may occur in the time remaining to the baseline deadline, computed as driftRate * ((now+BI) - BD). The error in baseliniing time is determined as, d_{bound} - d_{actual}, where d_{bound} is the flow delay bound and d_{actual} is the actual delay of the baselining frame. Thus early baselines are limited by the constraint $(d_{bound} - d_{actual}) >= (driftRate * ((now + BI) - BD))$.

3.3.8 Virtual Flow for Baselining

A virtual flow is added to the packet scheduler at priority level-0 (highest) called flow$_{01}$ for the purpose of allocating high priority transmission opportunities for baselining of flows utilizing the packet scheduler. There is only a single flow at this priority level with a flow number of 1. When flow$_{01}$ is included in the flow set of the packet scheduler the RCSP schedulability equation becomes:

$$\left\lceil \frac{d_m}{X_{\min_{0,1}}} \right\rceil \cdot P_{0,1} + \sum_{k=1}^{m} \sum_{j=1}^{i_k} \left(\left\lceil \frac{d_m}{X_{\min_{k,j}}} \right\rceil \cdot P_{k,j} \right) + P_{\max} \leq d_m \cdot l \tag{3.4}$$

The first term of (3.4) represents the amount of workload that is added by the inclusion of the baselining flow flow$_{01}$.

When B is the lowest (minimal) priority level allowed to utilize flow$_{01}$ baselining opportunities, we can draw the following 3 conclusions.

1) The delay bound for priority levels-m in the range $[1,B-1]$ must consider the existence of flow$_{01}$ in the delay bound schedulability equation. All priority levels in the range $[m+1,B]$ (those priority levels allowed to baseline and at lower priority than the priority level m) may fully utilize the flow$_{01}$ transmission opportunities for baselining purposes. The resulting delay bound schedulability equation for priority level-m will be (3.4). Here it is assumed that the summation of the bandwidth allocated to priority levels in the range of $[m+1,B]$ is sufficient to utilize all transmission opportunities of flow$_{01}$.

2) The delay bound for priority level-m $= B$ can be scheduled as if flow$_{01}$ had no bandwidth allocation at all. The rate at which frames arrive to flows within the priority level range 1 to B is restricted based on the minimal inter-arrival time of each flow. Recall that each flow is completely restored to its initial traffic envelope. Therefore, the workload of flows having frames that become eligible is constrained by the second term of (3.4). Therefore, considering the non-preemption characteristic of the scheduling policy the delay bound schedulability equation for priority level-B is the original RCSP defined as

$$\sum_{k=1}^{m}\sum_{j=1}^{i_k}\left(\left\lceil \frac{d_m}{X_{\min_{k,j}}}\right\rceil \cdot P_{k,j}\right) + P_{\max} \leq d_m \cdot l \qquad (3.5)$$

3) The delay bound for priority levels-m in the range $[B+1, M]$, where M is the lowest (minimal) level can be computed as if flow$_{01}$ had no bandwidth allocation at all. This follows as priority levels in the range $[1, B]$ all have higher priority and are limited in arrival rate by the fully restored traffic envelope on each flow. Equation (3.5) establishes the delay bound for these flow levels, which is the original RCSP. Flows supporting periodic traffic are occasionally transmitted as frames on the high priority virtual flow flow$_{01}$. In this way the frame can be scheduled for transmission at a certain time and will not have to contend with other traffic. This allows a schedule to be made for the transmission time of the frame, while additional traffic may yet arrive in the time remaining before the scheduled transmission time.

3.3.9 Virtual Flow Allocation Considerations

In order to support sufficient baselining epochs during the packet scheduling process, virtual flow flow$_{01}$ must be allocated a minimum inter-arrival time to support the baseline intervals of the FlexTDMA flows. This time (3.6) should be sufficient to allow a baseline event on each flow once each Baseline Interval (BI) period.

$$1 \Big/ \sum_{k=1}^{n} \frac{1}{\min BI_{flow_k}} \qquad (3.6)$$

The minimum BI for a flow (3.7) is a function of the clock drift rate and the maximum tolerable clock drift error.

$$\frac{\max ErrorPerBI_{flow_k}}{clockDrift_{ppm}} \qquad (3.7)$$

3.4 Baseline Cascading

In this section we describe the phenomenon of switch-to-switch baseline cascading and explain the details needed to avoid a *baseline cascade transmission inversion. Baseline Cascading* occurs when a baseline event on flow n produces a new base eligibility reference time by transmitting frame k at the deadline. The downstream switch receives the baselining frame k later than the existing eligibility time ($AT_n^k > ET_n^k$), sets flow n to a non-baselined state, sets the eligibility time of flow n to the arrival time of frame k ($ET_n^k = AT_n^k$), and will re-baseline flow n by transmitting frame k (or a later frame) at the deadline time. In this way, re-baselining events propagate through the network.

The FlexTDMA protocol is designed to avoid a *Baseline Cascade Transmission Inversion* which occurs when a frame is scheduled for baseline transmission at its deadline and a subsequent frame on the same flow is transmitted before the baselining frame. In this case the downstream switch may update the eligibility time of the flow at the reception of the second frame and therefore not treat the baselining frame as a baselining event. The result is that the downstream switch may not update the eligibility time basis for the flow. When this occurs, the end-to-end delay bound is shortened by the reduction in eligibility time basis.

A *Baseline Cascade Transmission Inversion* is caused by multiple frames on a flow arriving to a switch within the delay bound of a frame. This occurs when either the delay bound is larger than the flow period of the flow, or there is sufficient delay-jitter so that two or more frames arrive on a flow within the delay bound of the flow.

To illustrate this concept, Figure 3.3 shows a *Baseline Cascade Transmission Inversion* that would occur if the FlexTDMA protocol did not avoid this case by insuring a frame selected for baselining is transmitted prior to any subsequent frame on the flow. In Figure 3.3 three frames arrive to a FlexTDMA switch. Frame 1 arrives with $AT > ET$. The flow baseline state is set to non-baselined, Frame 1 is selected as a baselining frame and is scheduled for transmission at its deadline in flow$_{01}$ queue. The AT of Frame 2 is less than the current ET (as the ET was updated by Frame 1), and is scheduled for transmission in the FIFO queue. As there is no contending traffic in the FIFO queue (in this example), Frame 2 is selected for transmission

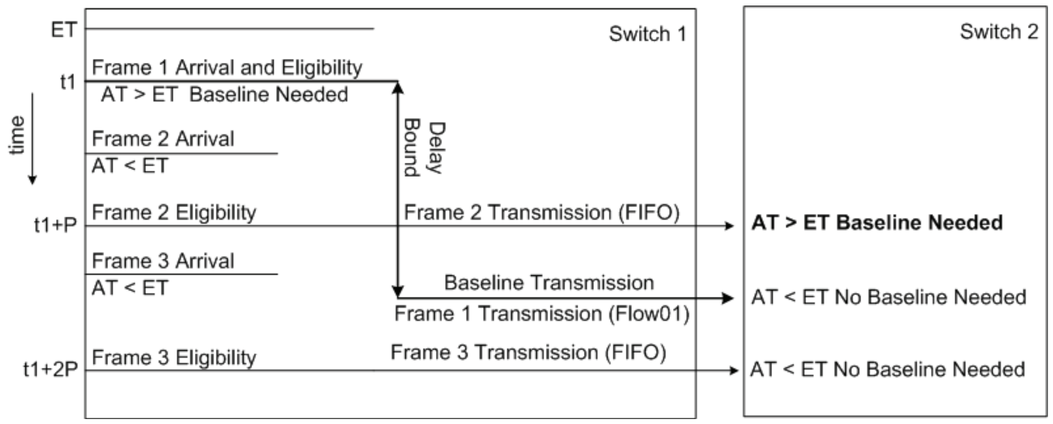

Figure 3.3 Baseline cascade transmission inversion may occur if it were not avoided by FlexTDMA

which precedes Frame 1 located in flow$_{01}$ queue. Frame 2 is received by the downstream switch and the ET for the flow is updated to $max(AT, ET) + flow\,period$. Notice the ET of the downstream switch is established based on Frame 2 rather than the baselining frame Frame 1. When Frame 1 becomes ready in flow$_{01}$ it is transmitted at its deadline as an attempted baselining transmission. The current switch will mark the flow as baselined but the downstream switch will receive the frame with $AT < ET = prev\,ET + flow\,period$, thus considering the frame not yet eligible leaving the flow baselined at the previous ET basis. Frame 3 arrives after Frame 1 transmission and is scheduled in the FIFO queue for transmission. The result is that the flow remains baselined using the previous ET basis in the downstream switch, and the baseline cascade attempt has failed.

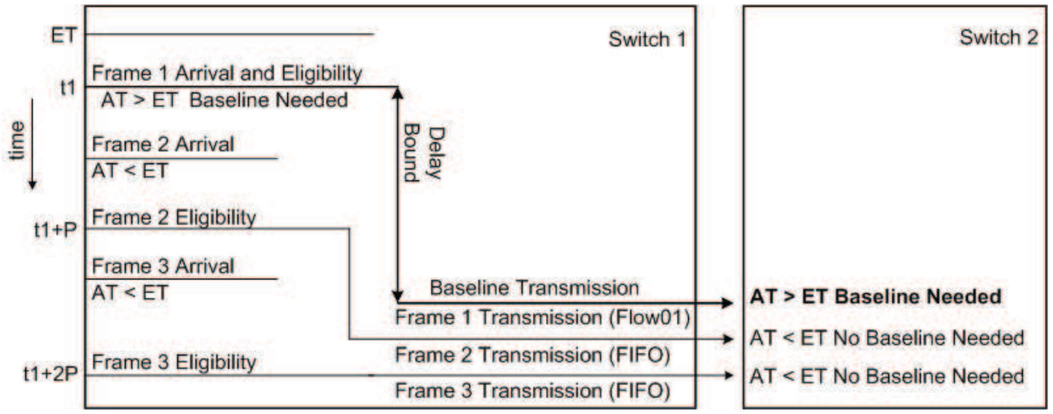

Figure 3.4 Baseline cascade transmission inversion is avoided by FlexTDMA

Figure 3.4 shows how the FlexTDMA protocol avoids a *Baseline Cascade Transmission Inversion* by insuring a frame selected for baselining is transmitted prior to any subsequent frame on the flow. In Figure 3.4 three frames arrive to a FlexTDMA switch. Frame 1 arrives with an $AT > ET$. The flow baseline state is set to non-baselined, Frame 1 is selected as a baselining frame and is scheduled for transmission at its deadline in flow$_{01}$ queue. Frame 2 arrives with an AT less than the current ET (as ET was updated by Frame 1), and is scheduled for transmission in the FIFO queue, but with the constraint that Frame 2 not be transmitted before the scheduled time of Frame 1 in flow$_{01}$ queue. When Frame 1 becomes ready in flow$_{01}$ queue it is transmitted at its deadline as a baselining transmission. The current switch will mark the flow as baselined and the downstream switch will receive Frame 1 with AT $>$ ET, as Frame 1 deadline represents an updated ET basis. The result is that the flow will be baselined at the downstream switch using the updated ET basis of Frame 1. Frame 2 is selected for transmission from the FIFO queue as it was delayed until the transmission of Frame 1. Frame 2 is received by the downstream switch with AT $<$ ET as the ET was updated to reflect the ET basis of Frame 1. Frame 2 will be placed in the downstream switch FIFO queue and transmitted. Frame 3 arrives after Frame 1 transmission and is scheduled in the FIFO queue for transmission. Few frames on a flow will be delayed since the flow period is typically larger than the delay-jitter on the flow. The baseline cascade attempt is successful as the downstream switch re-baselined the flow using the updated eligibility time basis from frame 1.

3.4.1 FlexTDMA Baseline Cascading Support Details

To avoid a *Baseline Cascading Transmission Inversion* the FlexTDMA protocol tracks two values per flow. The frame id (AtGTEtFrId) of the last arriving frame where the $AT > ET$, and the minimum transmit time (minTxTime) allowed for frames on the flow. The AtGTEtFrId parameter is updated to the frame id of the latest arriving frame with AT $>$ ET. In this way the latest frame id of each flow establishing an updated eligibility time baseline is known.

At frame eligibility time the frame is excluded from the flow$_{01}$ queue when the frame id $<$ AtGTEtFrId. This insures a baseline attempt is not performed with a frame having an ET basis less than the most recent. When a frame is placed in flow$_{01}$ queue the value minTxTime is updated to the scheduled transmission time and all frames on the flow are restricted to transmission times no sooner than minTxTime. This insures that no frame on a flow is transmitted prior to the delay bound of the frame being used to establish a baseline cascading event in the downstream switch.

A frame in flow$_{01}$ queue will be selected for transmission at its scheduled transmission time. If the frame id of the selected frame is less than AtGTEtFrId the flow baseline state is set to non-baselined, otherwise to baselined. This occurs when a frame is received with AT > ET while a previous frame selected for baselining is pending transmission. Notice that any ready frame from flow$_{01}$ queue having a frame id in the interval [AtGTEtFrId, maximum received id] is acceptable to use for baselining as all of the frames in this range have a deadline based on the latest ET basis.

3.4.2 Example of Inversion Avoidance

Table 3.2 shows an example series of frame arrivals and the resulting values of AtGTEtFrId, ET, and NextET. The flow period is assumed to be 10 ms and the flow delay bound is 25 ms.

Table 3.2 Baselining Parameters - an Example

Frame Id	**Arrival Time (ms)**	**At>Et FrId**	**Baseline Candidate**	**ET (ms)**	**Next ET (ms)**	**Deadline (ms)**
		0			7	
1	10	1	Yes	10	20=10+10	35=10+25
2	17	Same	No	20	30=20+10	45=20+25
3	31	3	Yes	31	41=31+10	56=31+25
4	36	Same	No	41	51=41+10	66=41+25
5	37	Same	No	51	61=51+10	76=51+25

Frame 1 arrives at time 10 ms which is greater than the current NextET of 7 ms, thus the AtGTEtFrId is set to the frame id of the frame = 1, the flow is marked as non-baselined, ET is set to max(AT=10 ms, nextET=7 ms), and the NextET is set to ET + flow period = 20 ms. Frame 1 is selected as a baseline candidate frame and will be placed into flow$_{01}$ queue and scheduled for baselining transmission at the deadline of 35 ms.

Frame 2 arrives at time 17 ms which is less than the current NextET of 20 ms, thus the AtGTEtFrId remains the same, ET is set to max(AT=17 ms, nextET=20 ms), and the NextET is set to ET + flow period = 30 ms.

Frame 3 arrives at time 31 ms which is greater than the current NextET of 30 ms, thus the AtGTEtFrId is set to the frame id of the frame = 3, the flow is marked as non-baselined, ET is set to max(AT=31 ms, nextET=30 ms), and the NextET is set to ET + flow period = 41 ms. Frame 3 will be placed into flow_{01} queue and scheduled for baselining transmission at the deadline of 56 ms. When Frame 1 is ready, at time 35 ms, in flow_{01} queue it will not be used to set the flow to a baselined state as the frame id of 1 is less than AtGTEtFrId of 3. Instead when Frame 3 is transmitted at the deadline time the flow will be baselined which insures the baselining event establishes the eligibility time basis of Frame 3 in the downstream switch.

Frame 4 arrives at time 36 ms which is less than the current NextET of 41 ms, thus the AtGTEtFrId remains the same, ET is set to max(AT=35 ms, nextET=41 ms), and the NextET is set to ET + flow period = 51 ms.

Frame 5 arrives at time 37 ms which is less than the current NextET of 51 ms, thus the AtGTEtFrId remains the same, ET is set to max(AT=37 ms, nextET=51 ms), and the NextET is set to ET + flow period = 61 ms.

Following the receipt of Frame 3, frame ids 3, 4, and 5 are in the range of [AtGTEtFrId, maximum received id] = [3, 5]. Therefore frames 3, 4 and 5 can be used for baselining since their eligibility times all share the same Frame 3 eligibility time basis. In this example Frame 3 was used to baseline the flow. In practice flow_{01} may have been unavailable for Frame 3 and a subsequent frame would have been selected for baselining.

In any case, when a frame arrives with an arrival time greater than eligibility time ($AT > ET$) a baseline will occur which is based on the new eligibility time basis. The flow is set to a non-baselined state so a future baseline is needed on the flow. The flow baselining event will occur using a frame with a frame id >= AtGTEtFrId since any frames with a frame id < AtGTEtFrId have an eligibility time not based on the latest eligibility time basis.

3.4.3 Proof of Correctness

Here we show that, in any case, once a frame is received with an arrival time greater than the eligibility time of the flow, a baseline event will occur using the updated eligibility time basis insuring proper baseline cascading to downstream switches. There are three possibilities for the status of the previous frame which was selected as a baselining frame when the current frame arrives with $AT > ET$: pending eligibility, pending transmission in flow$_{01}$ queue, or fully processed (already transmitted).

1) Pending Eligibility - When the previous frame becomes eligible it will be scheduled using the FIFO queue since the frame id is less than the frame id of the more recent frame received with $AT > ET$. Thus the previous frame will not be used as a baselining frame. Instead the most recent frame received with $AT > ET$ will establish the eligibility time basis for the flow.

2) Selected For Baselining Transmission In flow$_{01}$ Queue - When the previous frame is selected for transmission from flow$_{01}$ queue the flow is set to a non-baselined state since the frame id of the previous frame is less than the frame id of the more recent frame received with $AT > ET$. A later frame (possibly the current frame) will establish a baseline using the updated eligibility time basis.

3) Fully Processed - When the previous baselining frame is fully processed there is no conflict with the current frame as the flow was transitioned to a non-baselined baseline state when the current frame was received with AT > ET. Therefore, the pending baseline will not utilize the eligibility time basis of the previous frame.

3.5 Observations

In this section we make observations about the FlexTDMA protocol.

3.5.1 FlexTDMA Properties

FlexTDMA provides a level of service between asynchronous end systems and switches similar to a synchronized system supported with a TDMA scheduled interconnecting bus. The key criteria is: *constant delay bound, a low delay bound, and very low delay-jitter.*

Delay Bound: FlexTDMA offers deterministically bounded delay as all frames are transmitted at or below the delay bound of the flow at each switch, regardless of the baselined status of the flow. The end-to-end delays will be nearly equal to the end-to-end delay bound as each switch holds frames until eligibility once a flow is baselined.

Delay Stability: The inability of FlexTDMA to instantly establish a baselined status on each activated flow compromises the delay stability and delay-jitter performance relative to RCSP-DJ. A baselining opportunity must be used to establish a baseline for each flow. FlexTDMA provides fault isolation with no clock synchronization while RCSP-DJ provides more stable delay-jitter bounds.

Delay-Jitter: Once baselined, delay-jitter is limited to the final switch as previous switches remove delay-jitter by holding frames to the eligibility time. Delay-jitter is larger prior to baseline by contention in the switch path.

Constant Delay Bound: The FlexTDMA constant delay bound, based on packet scheduler delay, is larger than the synchronous TDMA tight delay bound. Synchronous TDMA systems offer a delay bound of 10's of μs relating to the slot timing precision.

Contention Management: FlexTDMA manages asynchronism by sharing transmissions at the ideal reserved slot time when contention occurs on baselined flows, as baseline events are infrequent.

Clock Drift: Baselined flows are re-baselined infrequently to compensate for intra-switch clock-drift at an interval that limits the maximal error. Baselining occurs frequently enough to limit clock-drift induced delay-jitter.

3.5.2 Complexity

FlexTDMA is more complex than RCSP. FlexTDMA and RCSP (7; 10) both require full reshaping prior to scheduling insertion at an output port scheduler. Both share a similar FIFO queuing mechanism to post frames pending transmission.

FlexTDMA must additionally track the baseline status of each flow, and selectively chose frames for baselining to insure that the each switch in the network has proper understanding of the maximal delay bound of each flow.

The relative complexity of FlexTDMA and RCSP differ by the flow$_{01}$ sorting process. At each frame arrival the correct queue location must be determined. The complexity will be O(n) scaled with the size of the queue. When implemented in software a for-loop is needed to traverse the queue, followed by pointer updates for insertion. When implemented in hardware parallel comparators can be used.

3.5.3 Implementability

FlexTDMA can be implemented with the addition of a virtual flow that holds non-work conserving frame transmissions that have been scheduled for transmission at the deadline (baselining). The implementation must sort the flow$_{01}$ based on the scheduled time, and determine if the element at the head of the list is due for transmission at each frame transmission completion. This is not significantly more effort than management of multiple FIFO queues in RCSP.

3.6 Performance

In this section we characterize key performances of the FlexTDMA protocol.

3.6.1 Performance Comparison Criteria

The two key performance characteristics that differentiate FlexTDMA from TDMA and RCSP are stable delay bounds at the end-to-end bounding value, and minimal delay-jitter. TDMA offers both minimal delay bounds and minimal delay-jitter service but only when the communicating elements are synchronized so that the common TDMA framing structure is respected.

The RCSP-DJ policy offers log-normal distributed delay bounds that are centered on the delay bound, with a corresponding small delay-jitter bound (7). The FlexTDMA will offer nearly the same delay and delay-jitter performance as the RCSP-DJ policy. This is because FlexTDMA establishes the eligibility time computation based on the maximal delay in the previous switch, just as the RCSP-DJ. The difference is that FlexTDMA is able to accomplish this in an asynchronous network, while RCSP-DJ requires inter-switch clock timing.

RCSP-RJ offers a low delay-jitter, and nominally low delay which is dependent on the actual delay bound at each switch in the network (7). The FlexTDMA policy will offer a nearly constant end-to-end delay equal to the delay bound with minimal delay-jitter (until the final switch) in an asynchronous network of switches with no clock coordination. When RCSP-RJ is used in an asynchronous network, the actual end-to-end delay bound is dependent on the frequency of occurrence of local maximal delay bounds at each individual switch. Maximal delay bounds are very rare because of network under-utilization, and the low probability of maximal frame arrival alignment among flows. Although RCSP-RJ will limit the delay-jitter, the end-to-end delay bound is not stable.

FlexTDMA offers a delay distribution that is concentrated at the upper bound of the end-to-end flow path, and has a very small delay-jitter value. Additionally FlexTDMA can be configured with a delay that significantly exceeds the bounding delay value. This allows the end-to-end delay bound to be based on a required value rather than an artifact of the as-built configuration. This maintains the specified end-to-end delay bound independent of other network utilizations, as long as the bounding delay value does not grow to the required value.

3.6.2 Simulation Results

Figure 3.5 shows the network topology used to demonstrate the relative advantages of FlexTDMA. The topology has 6 switches connected in series. The traffic from node 0 is forwarded to node 10 where the end-to-end delay and delay-jitter are monitored. The nodes 1 to 9 introduce cross traffic which is forwarded to switch 10 where it is discarded. Each node transmits frames on 50 flows at a period of 2050 μs with 0 to 50 μs of jitter added to each period time. This topology and switching function were simulated using OPNET. Each switch can be configured to support FlexTDMA, RCSP-RJ or RCSP-DJ. The bounding delays were computed off-line and were configured into each switch. This allows the FlexTDMA and RCSP-DJ policies to target the bounding delays at each switch. Additionally an artificially high delay bound was selected to simulate a requirement driven delay target for FlexTDMA.

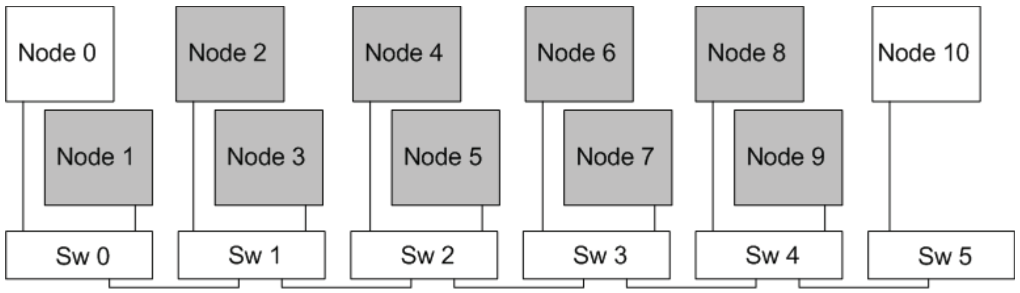

Figure 3.5 The OPNET simulation network used a 6-switch serial topology. The traffic generated by node 0 was accepted at node 10. Nodes 1 to 9 were used to generate cross-traffic forwarded to switch 5. The switches simulate RCSP-RJ, RCSP-DJ, and FlexTDMA.

Figure 3.6 shows the Cumulative Distribution Function (CDF) for end-to-end delay under each of the scheduling policies. The delay bounds for RCSP-RJ were nearly zero as there was little contention and therefore minimal actual delay at each switch. RCSP-DJ delay bounds were centered on the bounding delay of 4176 μs. RCSP-DJ has direct knowledge of the delay in the previous switch, and can simply delay each frame to the bounding delay of the previous switch. FlexTDMA suffers from baseline contention that causes some degree of delay less than the maximal delay bound. FlexTDMA had a few delay bounds ranging 3954 to 4176 μs due to baseline contention (this can be seen in the graph as the lower foot of the CDF for FlexTDMA), with the majority at the bounding delay of 4176 μs. Under FlexTDMA most frames are forwarded in a baselined condition so that no baseline contention is encountered. FlexTDMA was also tested with an artificially high required cumulative end-to-end delay bound of 6014 μs. There were a few delay bounds ranging 5018 to 6014 μs due to baseline contention (this can be seen in the graph as the lower foot of the CDF for FlexTDMA), with the majority at the bounding delay of 6014 μs. This demonstrates the ability of FlexTDMA to support delay bounds higher than the bounding delay values. This delay bound would be supported as long as the bounding delay is less than the required value.

Each policy tested offered very small (sub micro-second) delay-jitter performance, with delay-jitter measured as the reduction in the inter-arrival time of frames. Figure 3.7 shows the average delay-jitter under FlexTDMA over a one second simulated run. The delay-jitter

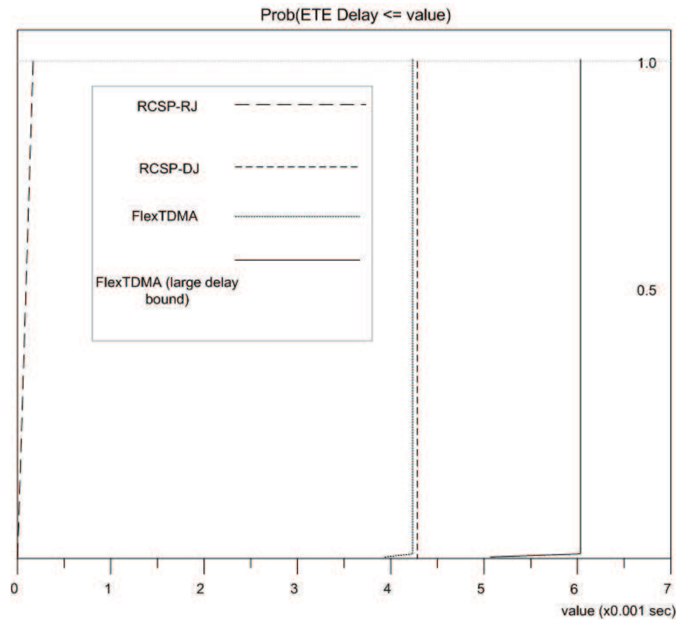

Figure 3.6 A CDF of the end-to-end delay bound experienced by RCSP-RJ (large dashed), RCSP-DJ (medium dashed), FlexTDMA (small dashed), and FlexTDMA with a large required delay bound (solid).

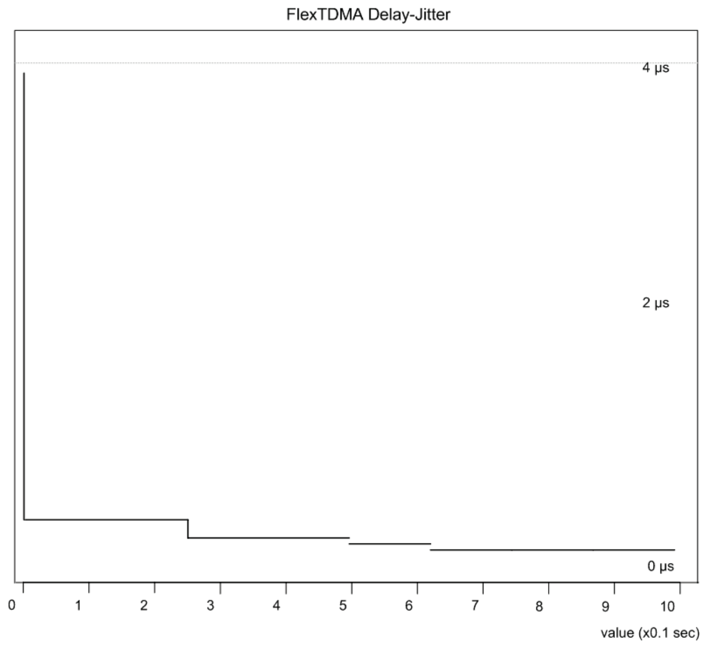

Figure 3.7 A plot of the average delay-jitter under FlexTDMA over the course of a one second simulation.

continually reduces as flows are baselined.

3.7 Chapter Summary

In this chapter we presented a switch frame processing methodology called FlexTDMA, which will enhance the system level performances of a closed network by insuring a constant delay, a low delay bound, and low delay-jitter in a network of asynchronous end systems and networking switches.

These results demonstrate the ability of FlexTDMA to offer nearly equal end-to-end delay and delay-jitter performance of RCSP-DJ in an asynchronous network. The difference in performance between FlexTDMA and RCSP-DJ results from the baseline contention existing in FlexTDMA.

CHAPTER 4. FLexTDMA+ Improvements

In this chapter we consider the FlexTDMA+ enhancements to FlexTDMA to consider real-world conditions of end node periodic on-off transmission, and network conditions of clock drift, frame loss and network bandwidth load. We propose three improvements to FlexTDMA 1) baseline preemption, 2) partial baselining and 3) baseline deadline density control. We evaluate the performance of each improvement combination in the presence of network conditions.

4.1 Introduction

In (63) we introduced FlexTDMA to provide minimal delay-jitter with nearly maximal delays in an asynchronous network. Here asynchronous refers to the lack of clock coordination between network components. FlexTDMA works by periodically transmitting a maximally delayed frame on each flow allowing downstream switches to establish a maximal eligibility time (ET) basis, where ET is the time at which an arriving frame is in conformance with the original traffic envelope of the transmitting node. Each FlexTDMA switch traffic shapes arriving frames using this ET basis. The FlexTDMA protocol shares maximal delay bound transmission opportunities in a process called baselining using a dedicated flow called flow$_{01}$.

Here we expand the consideration of FlexTDMA to include end node behavior (periodic on-off traffic), network conditions and improvements to FlexTDMA. Periodic on-off node transmissions occur when end nodes discontinue the flow of periodic messaging traffic due to reset, maintenance, or entering a different mode. Three network conditions are considered: clock drift, frame loss due to bit errors, and bandwidth load. Clock drift of 10 to 100 ppm (59) is common. When the components experience relative drift the accumulated clock error degrades the ability to deliver data at the maximal delay bound.

The improvements included in FlexTDMA+ are baseline preemption, partial baselining and baseline deadline density control. These improvements are motivated by poor performances, in FlexTDMA, resulting from insufficient coordination of baselining opportunities. Each improvement makes FlexTDMA+ more tolerant of multiple concurrent flow baseline demands. Baseline preemption allows a flow more urgently requiring baselining to preempt a flow scheduled for baselining. Partial baselining, which is used to manage multiple concurrent flow baseline demands, baselines at a time prior to the deadline of the frame. Baseline deadline density control utilizes unused baseline opportunities to prematurely baseline flows which have a baseline deadline in close proximity to other flows. These improvements included in FlexTDMA+ enhance the delay-jitter and data delivery at maximal delay performance in the presence of frame loss in the network, periodic on-off traffic, and clock-drift.

Utility of Research: The improvements to FlexTDMA included in FlexTDMA+ enhance the delay-jitter and data delivery at maximal delay performance in presence of frame loss in the network, periodic on-off, and clock-drift.

Frame loss is a realistic part of any network operation. The FlexTDMA protocol must deal with frame loss and offer an acceptable level of performance considering typical network frame error rates.

Periodic on-off node transmission is a realistic part of any network operation. The network being considered is a closed network collection of nodes and switches. The FlexTDMA protocol is primarily focused on delivery of periodic messaging traffic although aperiodic traffic can be supported in the network but not directly by FlexTDMA (63). During network operation it is common for connected end system nodes to discontinue the flow of periodic messaging traffic. There are several reasons for these interruptions including reset of the transmitting node, maintenance of the system, or the transmitting node entering a different mode of operation. Each of these cause the transmitting node to temporarily pause transmission and resume transmission once the node enters normal operating mode. When the FlexTDMA protocol supports periodic on-off, the transmitting end system nodes may stop and resume transmission while preserving the delay-jitter and maximal delay bound performance of the FlexTDMA protocol.

Clock drift among components of a network is typical. Clock drift in the amount of 10 to 100 ppm (59) is common. When the transmitting and receiving components of the network experience relative drift the accumulated clock error degrades the ability of the network to deliver data at the maximal delay bound. When the FlexTDMA protocol supports clock drift, asynchronous components can be connected each having an independent clock with drift.

Frame loss, periodic on-off and clock drift degrade the performance of FlexTDMA. With the enhancements included in FlexTDMA+, FlexTDMA performance can be offered in a real-world operating environment. These improvements enhances the utility of FlexTDMA in closed networking operation.

Proposed Research: Our research focus was on the FlexTDMA+ improvements to FlexTDMA under different network operational circumstances. This section details these improvements.

We propose the following FlexTDMA+ improvements: baseline deadline density reduction, partial baselining, and baselining preemption. These improvements will allow the delay-jitter performance of FlexTDMA in the presence of frame loss, periodic on-off, and clock drift in the network.

Baselining preemption allows a flow not yet baselined to preempt another baselined flow that is currently scheduled for baselining. This helps reduce the impact of baseline collisions among flows when the currently scheduled baselined flow is simply a baseline renewal, and another flow has not yet been baselined. The impact to the baselined flow is a function of the clock drift rates of the switch components of the network, in that the delivery of data at the maximal delay bound becomes less accurate. The impact to the flow not yet baselined is more significant in that frames will be delivered with minimal delay rather than approximating the maximal delay bound of the flow. Baseline preemption also mitigates the impact of circumstances where a flow becomes non-baselined - such as frame loss or periodic on-off. In real-world operation these cases are typically sparse among the flow set supported, and the flow requiring baselining is given priority over those currently scheduled for baselining so that the non-baselined flow does not compete with flows simply renewing their baselined status under FlexTDMA+. Preempted flows are moved from the flow$_{01}$ queue to the FIFO queue for the port, and retain their baselined status. A preempted flow will attempt a baseline at a future frame arrival.

Partial baselining occurs when a flow needs to be baselined but there is no baselining opportunity on flow$_{01}$ at the deadline time of the flow. In some cases multiple flows require baselining at the same time. This is either due to the phase of periodic traffic on the flows aligning or an event causing multiple flows to enter a non-baselined state such as a transmitting node reset. When this occurs it is desirable to have the interrupted flows attain a baselined state as quickly as possible so that the maximal delay bound of FlexTDMA can be achieved for the flow. When multiple flows require a baseline at the same deadline time it is only possible to baseline one flow at its deadline, and the other flows will need to wait for another baselining opportunity ideally at the flow deadline. Rather than have the other flows deliver frames with a minimal delay bound until baselining occurs, the flow may be partially baselined. That is, the frame on the flow is scheduled for baselining at a time that precedes the deadline by some amount rather than exactly at the flow deadline. This means that the frame is transmitted with a delay that is less than the delay bound of the flow. Downstream switches perceive this as a baselining event and will schedule the flow for baselining (either with the current frame or another). Until the flow is baselined using the flow deadline at the current switch the total delay bound will be less than the maximal delay bound by the amount of time the partial baselining was early relative to the deadline of the flow.

Baseline deadline density control works by computing the baseline deadline density of the flow on which a frame is received relative to the average baseline deadline density of all flows pending baseline. When a frame is received on a flow having a baseline deadline density greater than the average baseline density and there is an unused transmission opportunity available for baselining the flow, the frame is scheduled for baselining. The baseline deadline is updated to the minimum baseline interval for the flow following the baseline time. This reestablishes a new baselining phase for the flow. When this is performed on all flows, those flows having densely scheduled baseline deadlines tend to be dispersed. This makes baselining collisions between flows a one-time occurrence rather than an event that occurs over and over when baseline deadlines align. Baseline deadline density control helps maintain the baseline deadline schedule uniformly distributed so that asynchronous flow renewals are more easily supported as they do not collide with dense portions of the baseline deadline schedule.

The remainder of this chapter is organized as follows. In Section 4.2, we discuss the improvements to the FlexTDMA protocol offered in FlexTDMA+. In Section 4.3, we describe network operational characteristics and the influence on FlexTDMA. In Section 4.4, we review the operational details of the FlexTDMA+ protocol. In Section 4.5, we provide the evaluation approach of the FlexTDMA+ protocol. In Section 4.6, we summarize the findings of the improvements found in the FlexTDMA+ protocol.

4.2 FlexTDMA+

In this section we discuss FlexTDMA+ and show how it improves over FlexTDMA. FlexTDMA+ improvements support operation in real networking environments. Real networks have frame loss and periodic node on-off transmissions.

4.2.1 Baseline Deadline Density Control

The FlexTDMA+ protocol maintains a schedule of baseline deadlines for each active flow.

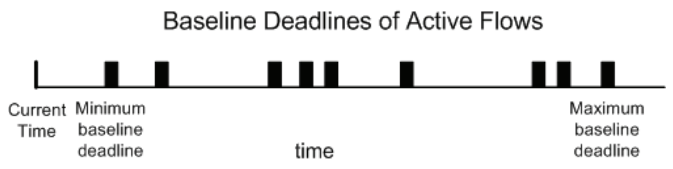
Baseline Deadlines of Active Flows
Current Time
Minimum baseline deadline
time
Maximum baseline deadline

Figure 4.1 Each active flow has a baseline deadline maintained in flow$_{01}$ queue.

Figure 4.1 shows the FlexTDMA switch baseline deadline schedule for all active flows. Each active flow must be re-baselined by the established baseline deadline in order to bound the effect of clock drift to planned limits. When a flow is rebaselined, the baseline deadline is updated in the baselined deadline schedule to the minimum baseline interval for the flow.

Baseline deadline density control attempts to maintain a uniform distribution for the scheduled baseline deadlines of active flows. The result is that flows tend to reach their baseline deadlines at a steady rate rather than bursting collections of flows requiring baselining. Each flow requiring baselining must be scheduled in flow$_{01}$ queue for transmission at the flow deadline. When other flows have recently been scheduled for baselining in flow$_{01}$, the probability of a baselining collision increases. This forces the baseline event to occur at a future frame on the flow resulting in degradation of FlexTDMA performance for constant delay bound and delay-jitter. By maintaining a uniform distribution for the scheduled baseline deadlines of active flows the baselining transmission opportunities are applied to the flows that need it most. Each frame arriving to a baselined flow is considered for baseline deadline density control. When the flow is not yet baselined, density control is not an issue - instead the goal is to reach a baselined state as soon as possible. The baseline density of the flow is computed relative to all flows with a baseline deadline (which is all active flows which have reached a baselined state). When the density of the flow is greater than the average, the flow is eligible for baselining at the flow deadline. When the flow$_{01}$ queue has a baselining opportunity at the flow deadline the flow is scheduled in flow$_{01}$ queue at the deadline, otherwise it is not scheduled for baselining.

The baseline deadline density control function within FlexTDMA+ works by computing the density of baseline deadlines surrounding flow (i) compared to the average baseline deadline density. When a frame arrives on a flow having a baseline density greater than average, the flow is scheduled on flow$_{01}$ if available. When a baseline occurs on this flow the baseline deadline is advanced by the baseline interval. The expectation is that the resulting density is decreased to average.

As shown in Figure 4.2 the baseline density of a flow is computed by determining the distance in time between the preceding and the following baseline deadline. This value is then divided by two as there are two intervals surrounding the baseline deadline of the flow. The average baseline density for all flows pending baseline is determined by computing the distance in time from the first scheduled baseline deadline to the last scheduled baseline deadline and dividing this amount by $n-1$ where there are n scheduled baselines.

In order to lessen the overhead of the baselining deadline density computation the flow$_{01}$ queue can be stored so that the baseline deadline of the previous and subsequent flows are easily determined. This requires a sorted structure based on baseline deadline of each flow, and a mapping from a flow number to an element of the structure so the element relating the current frame can be determined.

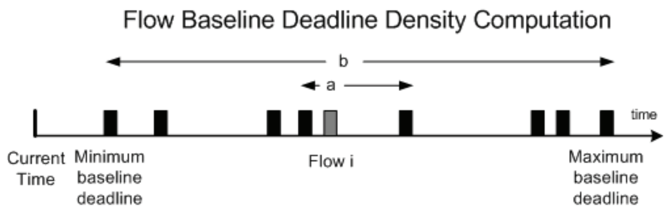
Flow Baseline Deadline Density Computation
b
a
time
Current Time
Minimum baseline deadline
Flow i
Maximum baseline deadline

Figure 4.2 Baseline deadline density for flow i is computed as $(a/2)/(b/(n-1))$.

Figure 4.2 shows the FlexTDMA switch baseline deadline schedule for each active flow. The baselined deadline for flow i is shown. The distance between the baseline deadline preceding and following flow i is a, and the distance from the first baseline deadline to the last is b. Therefore the density assigned to flow i is $(a/2)/(b/(n-1))$ when there are n flows.

4.2.2 Partial Baselining

Partial baselining of a flow occurs when a baseline is scheduled for a flow prior to the deadline time of the flow. This occurs when several flows are contending for baselining having nearly the same deadline and results in approximated baselining for those flows not baselined at their baseline deadline. This allows flows that are not yet baselined to deliver data with a delay that approximates the maximal delay bound, improving the overall performance of the flow set supported by the protocol.

Each arriving frame on a flow is considered for partial baselining when the flow$_{01}$ queue has no baselining opportunity. There are two classes of flows that are candidates for partial baselining. Those flows which are not baselined, and those flows that are.

When a frame is received on a non-baselined flow and flow$_{01}$ queue has no baselining opportunity the flow may be partially baselined. This will establish an eligibility time basis based on this partial baseline in the downstream switch, and will make the end-to-end delay bound approximate the maximal delay bound by the error in partial baselining.

When a frame is received on a baselined flow, and flow$_{01}$ queue has no baselining opportunity the flow may be partially baselined at the next earlier transmission opportunity with one important constraint: the partial baseline must extend the baseline deadline - that is, cause the baseline deadline of the flow to increase. When the error (the amount of time the partial baseline precedes the flow deadline) is large enough the new baseline deadline resulting from the partial baseline will actually be earlier than the current baseline deadline. The purpose in baselining a baselined flow is to update the frame eligibility time in the downstream switch given that relative drift that may occur between baselines. The new baseline deadline will be $now + baseline\,interval - error/maximum\,drift\,rate$. When baseline interval is less than $error/maximum\,drift\,rate$, the new baseline deadline would precede the current. In this case the partial baseline will not be performed.

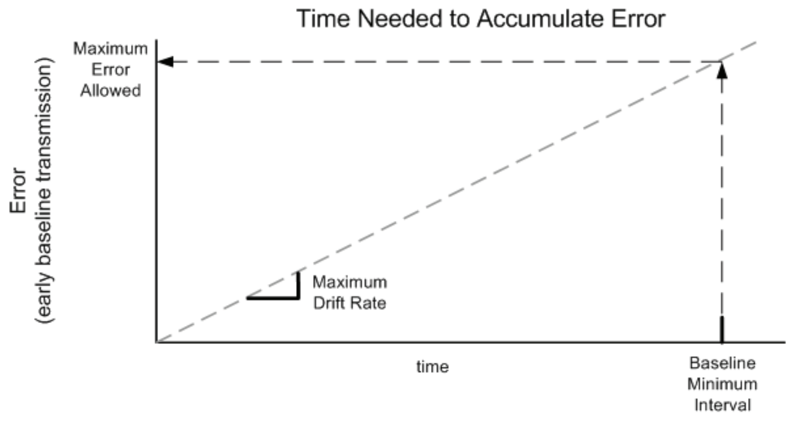

Figure 4.3 Accumulated error $E = drift * t$, which is clock drift applied over time t.

Figure 4.3 shows the relationship between the maximum baselining error allowed on a flow and the minimum baseline interval for the flow. Here *error* refers to the degree to which a flow is baselined prior to the baseline deadline of the flow. The maximum error allowed is equal to the maximum allowed drift rate multiplied by the baseline interval for the flow.

A disadvantage of partial baselining is that it requires another baselining in downstream switches once the baselining event occurs in the current switch since the baselining frame will be transmitted with a larger eligibility time basis than the partial baselining frame. This requires all downstream switches to baseline again.

4.2.3 Baselining Preemption

When a frame is received and flow_{01} has no baselining opportunity at the arriving flow deadline, that flow may preempt the current frame scheduled in the flow_{01} queue when three conditions are met. First, the current scheduled frame in flow_{01} queue must be from a flow that is baselined. No preemption will be allowed when the current scheduled frame is non-baselined as it is important to allow that flow to attain baselined status. Second, either the preempting flow is not baselined or the baseline deadline of the preempting flow is less than the baseline deadline of the preemption candidate. Thus a non-baselined flow is allowed to preempt a baselined flow since it is urgent to baseline the non-baselined flow, and a baselined flow may preempt another baselined flow when the baseline deadline is less, and so on. Third, as shown in Figure 4.4, in order for a frame scheduled in flow_{01} to be a candidate for preemption it must also be the case that the preempted frame would meet its deadline when placed in the FIFO queue. Each frame under FlexTDMA must be transmitted at or before its deadline. To determine this, the laxity of the frame to its deadline is compared with the existing workload in the FIFO queue. Figure 4.4 shows a frame on flow 1 preempting a frame from flow 2 scheduled for baselining using flow_{01}. The deadline time of flow 1 corresponds the deadline time of flow 2. When flow 2 is preempted from flow_{01} to the FIFO queue, the transmission opportunity created in flow_{01} will be used by flow 1 to schedule a baseline. The workload of the FIFO queue is less than the scheduled transmission time of flow 2 in the flow_{01} queue. Therefore, flow 2 can be preempted from flow_{01} and meet its transmission deadline when placed at the tail of the FIFO queue.

When a preemption occurs the frame is removed from the flow_{01} queue, placed at the tail of the FIFO queue, and the preempting frame is placed at the created transmission opportunity location in flow_{01} queue. The preempted frame is placed at the tail of the FIFO queue in order to preserve the maximal delay to service of the FIFO queue for other flows.

4.3 Network Operation

In this section we describe network operational characteristics and the influence on FlexTDMA.

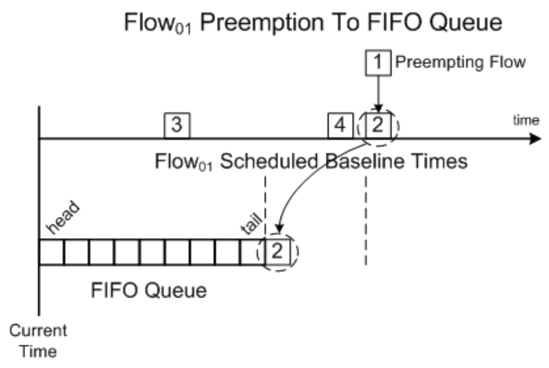
Flow01 Preemption To FIFO Queue
1 Preempting Flow
3
4
2
time
Flow01 Scheduled Baseline Times
head
tail
2
FIFO Queue
Current Time

Figure 4.4 Scheduling Baselines Preempted From Flow$_{01}$ to FIFO Queue.

4.3.1 Bandwidth Loading

The bandwidth loading of all flows managed by FlexTDMA at each switch is important as the FlexTDMA operation must maintain each flow in a baselined state. More flows imply a higher rate of baseline transmission opportunity utilization.

4.3.2 Periodic On-Off

Nodes transmitting periodic traffic occasionally pause and resume transmission of data. This affects the FlexTDMA protocol as each flow must be baselined. The nodes simulated during testing of FlexTDMA+ each pause all periodic transmission of data with some probability at each frame transmission.

When a node resumes transmission of periodic traffic the periodic traffic resumption will generally be asynchronous with the traffic timing preceding the interruption. This means that the flow on which the traffic is supported must again be baselined to the new timing of the transmission of the periodic traffic flow. When a simulated transmitting node is paused, a random period of time is selected for the pause, after which transmissions resume. The resumed periodic transmissions are not synchronized with those terminated on each flow preceding the interruption in transmission as the pause period is random. This mimics the general behavior of a transmitting node that was reset.

Figure 4.5 shows a node periodic transmission pausing for period k and resuming. Period k is not necessarily an integer multiple of the flow periodic transmission period p. Thus the

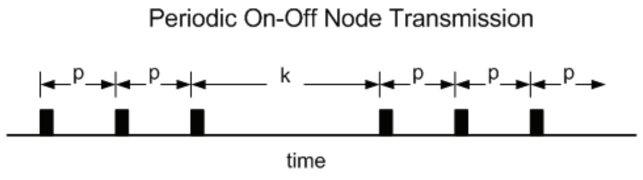

Figure 4.5 Periodic On-Off transmissions pause for a time period and resume.

resumed traffic is not synchronized with the traffic prior to the interruption.

4.3.3 Frame Loss Due to Bit Errors

Networks occasionally drop frames due to bit errors in transmission. This affects the FlexTDMA+ protocol as the flow must be baselined following the interruption in transmission of the lost frame. To simulate frame loss during testing of the FlexTDMA+ protocol each switch will discard a frame at a configured probability to act as a lost frame. This probability is called a frame error rate (FER). Following a frame loss on a flow a baseline is necessary since the FlexTDMA protocol cannot distinguish between a gap due to a frame loss and a pause in transmission.

4.3.4 Node and Switch Clock Drift

In (63) we described the advantages of network operation with asynchronous clock operation. Switch design is simplified, correct network function only relies on the operation of those switches traversed by a flow, and no complex synchronization mechanisms are needed for correct network operation.

As each node and switch in the FlexTDMA network operate asynchronously, each node and switch clock operate independently with no knowledge of the relationship to any other clock. Clock oscillators determine the actual clock rate of each component relative to the ideal rate. Each node and switch has a part-per-million (ppm) clock drift rate relative to the ideal rate. Commercial parts may vary by 10 to 100 ppm (59), running fast or slow.

It is important that the FlexTDMA protocol provide the delay-jitter and stable delay performance when there are variations in clock speed among nodes and switches in the network.

This ability helps make FlexTDMA a practical network traffic scheduling protocol.

4.3.5 Switch Clock Drift

4.3.5.1 General Impact

The FlexTDMA protocol ability to deliver frames at the maximal delay bound for each flow is affected by clock drift. When a switch along the path of a flow has a slow clock the switch will hold the frame longer than needed resulting is a longer delay bound through the switch.

4.3.5.2 Clock compensated duration

During FlexTDMA protocol simulation clock compensated duration values are computed as duration x (1 - ppmFast). The ppmFast is the rate at which the device clock is running in excess of real-time. A positive (negative) value implies the device is actually running faster (slower) than real-time. For example when ppmFast = 0.01 this means that the device clock runs 1% fast relative to real-time.

4.3.6 Clock Drift Operational Effect

There are three specific operational effects of clock drift on the operation of the FlexTDMA protocol. These are flow period for each flow, time to eligibility and flow delay bound.

The FlexTDMA switch is configured with the flow period of each flow and is used to determine an expected inter-arrival time of frames on the flow. The eligibility time of the next frame is set to the clock compensated duration of the flow period added to the current eligibility time. Using a clock compensated duration insures that the next frame will be eligible when the maximum drift rate is experienced. A FlexTDMA switch having clock drift with a fast running clock will determine each arriving frame eligible before a switch with no clock drift.

The FlexTDMA protocol determines the eligibility time of each arriving frame. The clock drift of the current switch will affect the duration until the arriving frame is eligible. This makes a difference in the end-to-end delay bound as the time-to-eligibility is the hold time from early transmission in the previous switch.

The FlexTDMA protocol computes the delay bound of each flow based on the port transmission rate and number of flows. The actual delay bound offered will be affected by the clock drift of the FlexTDMA switch.

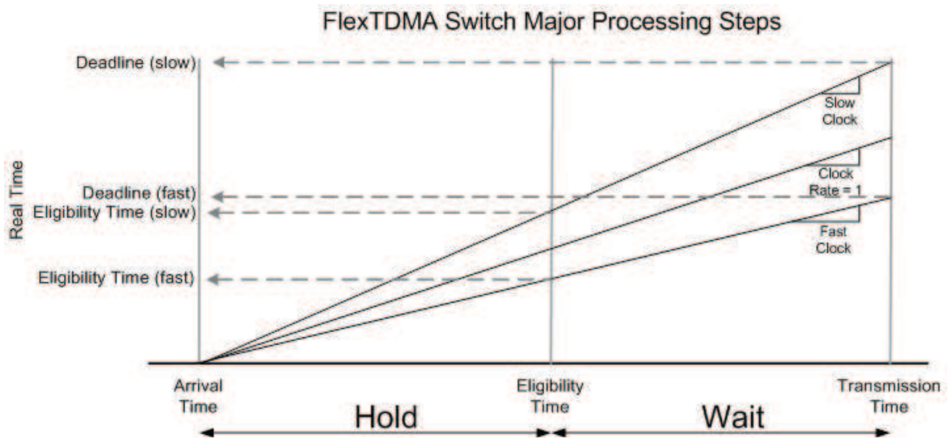

Figure 4.6 Clock drift effects time-to-eligibility (hold time) and time-to-transmission (wait time).

Figure 4.6 shows the effect of a fast or slow switch clock has on the real-time eligibility time and computed deadline for each flow. When a switch operates with a fast (slow) clock the duration of the hold time is decreased (increased) which reduces (increases) the dampening of the delay-jitter of the flow. Similarly, when a switch operates with a fast (slow) clock the duration of the wait time is decreased (increased), which increases (reduces) the delay-jitter of the flow by compressing (expanding) the deadline times realized for each flow.

4.3.7 Switch Delay Bound Computation Under FlexTDMA+

The delay bound for each flow in a FlexTDMA+ switch is computed based on the configured set of flows. Delay bounds are computed at each switch output port using the service rate of the output port, the periods and frame sizes of flows using the port. The periods used to compute delay bounds are clock compensated so that the switch is tolerant of node clock drift. This allows a switch to accept flow traffic from a node that is operating with clock drift of a fast clock. Thus the delay bound computation of each FlexTDMA switch is computed assuming that each node is transmitting the periodic message traffic using a fast clock. This insures that all data will be accepted in the presence of clock drift.

4.4 FlexTDMA+ Switch Operation

In this section we review the operational details of the FlexTDMA+ protocol.

4.4.1 Stages of processing

There are three stages to frame processing in a FlexTDMA+ switch: frame arrival, frame eligibility, and frame selection for transmission.

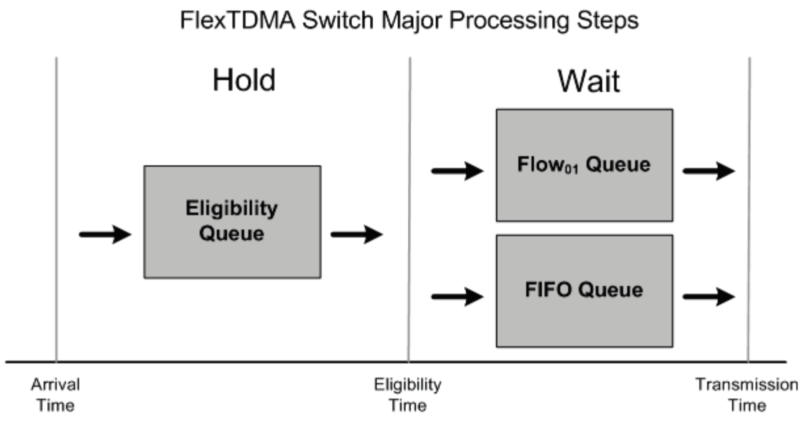

Figure 4.7 FlexTDMA Sswitch Major Processing Phases and Events

Figure 4.7 shows the major processing phases and events of the FlexTDMA switch. Arriving frames are held a *hold* period until the eligibility time of the frame (10), (7). The hold time is that time needed to restore the flow arrival frame pattern to the original regulated transmission from the source node (10), (7). Once eligible, the frame is scheduled for transmission at the output port in either the $flow_{01}$ queue when a baseline transmission opportunity exists, or in the FIFO queue (63). The period of time the frame exists in a queue pending transmission is called the *wait* time. The wait time is the actual delay a frame experiences in the packet scheduler of the switch prior to transmission (10), (7).

4.4.2 Switch Behavior at Frame Arrival

4.4.2.1 Frame Discard at Probability For FER

During simulation testing the switch imposes a switch FER at each arriving frame to simulate frame loss due to bit errors in transmission. The probability of FER is a test parameter.

4.4.2.2 Frame arrival after eligibility time

When the arrival time of a frame is greater than the eligibility time of the flow ($currentTime > nextET$), several parameters are updated. The last frame id of the frame with $AT > ET$ is recorded (parameter AtGtEt). This parameter stores the frame id number of the most recent frame received on each flow with $AT > ET$. The flow is set to non-baselined at each forwarding port.

4.4.2.3 Determine eligibility time of received frame

The eligibility time of the arriving flow is computed as $ET = max(now, nextET)$. The next eligibility time is updated based on the current eligibility time as $nextET = ET + flow\,Period$. The flow period is clock compensated assuming maximal drift and is computed as $(flow\,Period)(1 - max\,PPM\,Drift)$. This allows the switch to have maximally fast clock drift (unknown to the switch) and each frame will be accepted as non-eligible when periodic traffic arrives with inter-arrival times of the flow period.

4.4.2.4 Frame Queued Pending Eligibility

Once the eligibility time of the frame is determined, the frame is stored in an eligibility queue pending eligibility. It is possible that the frame is immediately eligible in which case the frame will be immediately removed and processed. Most frames arrive and are stored pending eligibility for a short time (up to the delay bound of the previous switch) since they were transmitted by the upstream switch with a delay much less than the maximal delay bound.

4.4.3 Switch Behavior at Frame Eligibility

Here we discuss the details of FlexTDMA+ protocol actions at frame eligibility. Once a frame reaches its eligibility time it is processed for each forwarding output port. There are three steps to frame processing at eligibility time. Step 1 - a frame is created for each output port the flow is forwarded. This is a duplicate of the received frame. Step 2 - Flow$_{01}$ availability is determined for each output port at the deadline time for each output port. Step 3 - frames are scheduled in a frame queue at each output port pending transmission.

4.4.3.1 Eligibility Step 1 - Determine Frame Forwarding Deadlines

The delay bound for each forwarded frame is set as the delay bound for the port the frame is forwarded. The delay bounds for each output port may be different as the complement of flows to each output port may differ. The delay bound used is clock compensated and is computed as $(computed\,Delay)(1 - max\ PPM\ Drift)$. Using the clock compensated delay bound insures the actual delay will not exceed the intended delay bound when the switch is operating with maximally slow clock drift.

The deadline of the frame is set to the eligibility time (the current time as the frame is currently eligible) plus the clock compensated delay bound. The frame will complete transmission before the deadline time.

4.4.3.2 Eligibility Step 2 - Determine Flow$_{01}$ Availability and Preemption Potential (if needed)

The next step is to determine the flow$_{01}$ availability and preemption potential when another flow is currently scheduled at the deadline time of the current flow.

Scheduling the flow$_{01}$ queue is limited so that elements are separated in time by the allocated flow$_{01}$ queue period. This ensures a maximal flow$_{01}$ queue bandwidth utilization at the output port. Figure 4.8 shows the minimal interval period p between scheduled baseline times in the flow$_{01}$ queue. This limits the bandwidth utilization of flow$_{01}$ to $(1\,frame)/(period\,p)$. Baseline scheduled times for a flow must be located in flow$_{01}$ in accordance with this constraint, but as

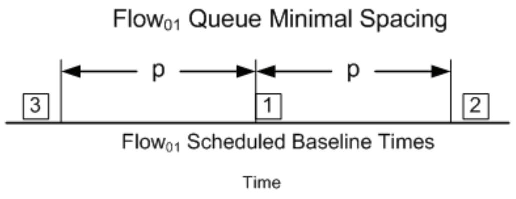

Figure 4.8 Flow$_{01}$ queue is serviced using a minimal frame period to limit bandwidth utilization.

close to the deadline of the flow as possible. Ideally the scheduled time of a frame in flow$_{01}$ is exactly the deadline of the frame. This exactly establishes the eligibility time basis of the flow at the maximal delay bound of the flow.

Flow$_{01}$ queue will be traversed starting at the deadline of the flow and continuing until the stop point. Therefore the interval of time considered in the traversal is $[stop, deadline]$. No traversal is done for times exceeding the deadline as the frame must be transmitted by the deadline of flow.

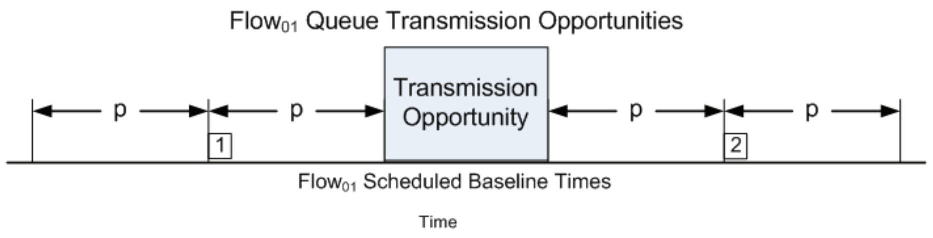
Flow01 Queue Transmission Opportunities
p
p
Transmission Opportunity
p
p
1
2
Flow01 Scheduled Baseline Times
Time

Figure 4.9 Transmission opportunities in flow$_{01}$ queue are restricted to the minimum transmission period.

Figure 4.9 shows a transmission opportunity to schedule a baselining frame. The interval between the scheduled times of flows 1 and 2 is larger than $2p$. The gap generated is the interval $[flow\,1\,time + p, flow\,2\,time - p]$. When a frame is scheduled for baselining in this gap the minimal spacing of scheduled baseline times is respected.

The stop point is established as the maximum of the current time and the deadline minus the duration limit. The current time is a lower limit as the frame must be scheduled for transmission in the future. The transmission opportunity for flow$_{01}$ queue having the maximum scheduled time will be selected for scheduling the current flow.

When there is no transmission opportunity for flow$_{01}$ queue at the deadline of the current flow a determination will be made as to whether the currently scheduled flow in flow$_{01}$ queue can be preempted by the current eligible flow.

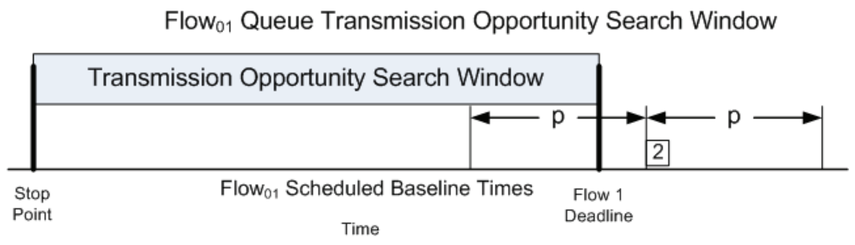
Flow01 Queue Transmission Opportunity Search Window
Transmission Opportunity Search Window
p
p
2
Stop Point
Flow01 Scheduled Baseline Times
Time
Flow 1 Deadline

Figure 4.10 Flow$_{01}$ queue transmission opportunity search window extends from the frame deadline to the stop point.

Figure 4.10 shows the interval of flow$_{01}$ that will be searched for baselining transmission opportunities for the eligible flow. The interval window is $[stop\,point, flow\,deadline]$. The scheduled baseline time must be less than the deadline of the flow or the frame would not be transmitted by the deadline time. Ideally a transmission opportunity will be located in the window close to the deadline. The search will continue until the stop point. When no transmission opportunities are located flow$_{01}$ is unavailable for baselining.

The duration limit is established that limits the extent to which a flow is scheduled in flow$_{01}$ queue prior to its deadline. This is known as partial baselining. The extent of partial baselining allowed depends on the baseline state of the flow (and whether partial baselining is enabled in the current test mode). When the current flow baseline state is either baselined or baseline exceeded the duration limit is set to $driftPpmMax * ((now + BDinterval) - curBD)$. This limits partial baselining to amounts that update the baseline deadline. This follows as the flow is already baselined, and there is no need to establish an approximate eligibility time basis for the flow.

Figure 4.11 shows the relationship between the baseline deadline of the flow and the duration limit for early baselining. When a baseline is preformed the new baseline deadline ($newBD$) is updated to the current time (t) plus the baseline interval (BI) minus the error induced interval. That is $newBD = t + BI - error$. The duration limit is set $(t + BI - BD)/(maximumdrift)$ so that the new baseline deadline will minimally be set to the current baseline deadline. When a baseline has recently occurred the resulting duration limit will be small as any error will

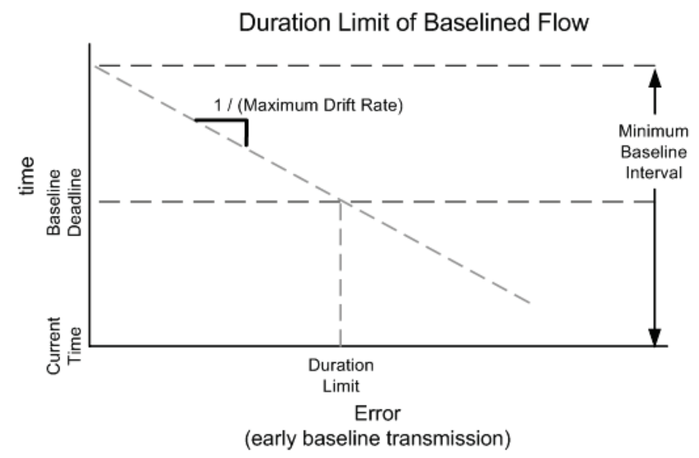
Duration Limit of Baselined Flow
1 / (Maximum Drift Rate)
Minimum Baseline Interval
time
Baseline Deadline
Current Time
Duration Limit
Error
(early baseline transmission)

Figure 4.11 Baseline error duration limit computation.

reduce the updated baseline deadline to be earlier than the current. When the current time is approaching the baseline deadline the resulting duration limit will be larger as an error can be tolerated which will reduced the updated baseline deadline to current baseline deadline.

When the current flow baseline state is non-baselined the duration limit is set so that the time needed to accumulate the error is less than the minimum baseline interval. Otherwise, the flow must be immediately re-baselined since the baseline deadline will be updated past the current time.

When a transmission opportunity is found in flow$_{01}$ the flow is marked as a candidate for baseline scheduling.

4.4.3.3 Eligibility Step 3 FlexTDMA+ Queuing Decision

The actions taken by the FlexTDMA+ protocol at frame eligibility are characterized in Tables 4.1, 4.2 and 4.3. These tables are used to determine which queue an eligible frame will be stored in pending transmission (either the flow$_{01}$ or the FIFO queue).

Table 4.1 shows the queuing decision logic for the 'Not Baselined' state of the FlexTDMA Protocol. Using this table each current eligible frame will be scheduled in either the FIFO queue or the flow$_{01}$ queue. The 'Baseline Deadline Density Above Average' column is NA since the flow is not yet baselined. The emphasis is on achieving a baselined status rather than baseline density control. Table 4.1 row 1 shows that when flow$_{01}$ has a transmission opportunity at the

Table 4.1 FlexTDMA+ Queuing Decision Logic for Non-Baselined State

	Baseline Deadline Density Above Average	**Flow$_{01}$ Availability at Deadline**	**Flow$_{01}$ Availability Within Duration Limit**	**Preemption Eligible**	**Queue**
1	NA	**Yes**	NA	NA	Flow$_{01}$ at deadline
2	NA	No	No	No	FIFO
3	NA	No	No	**Yes**	Preempt Flow$_{01}$ at deadline
4	NA	No	**Yes**	No	Flow$_{01}$ at latest available location
5	NA	No	**Yes**	**Yes**	Preempt Flow$_{01}$ at deadline

flow deadline the frame will be scheduled in flow$_{01}$ at the deadline time. Table 4.1 row 2 shows that when flow$_{01}$ does not have a transmission opportunity at the flow deadline of the frame, no transmission opportunity is found within the duration limit of flow$_{01}$ and the current frame is not eligible for preemption of the scheduled frame in flow$_{01}$, the frame is simply scheduled in the FIFO queue. Table 4.1 row 3 shows that when the current frame is eligible for preemption of the scheduled frame in flow$_{01}$, a preemption will be performed. Table 4.1 row 4 shows that when a transmission opportunity is found within the duration limit of flow$_{01}$, the frame is scheduled in the flow$_{01}$ queue. This is a partial baseline. Table 4.1 row 5 shows that when a transmission opportunity is found within the duration limit of flow$_{01}$ and the current frame is eligible for preemption of the scheduled frame in flow$_{01}$, a preemption will be performed. Here preemption is preferred to partial baselining as the current frame was found to have a higher priority than the scheduled frame. By performing a preemption the higher priority flow is baselined at its deadline.

Table 4.2 shows the queuing decision logic for the ‘Baselined’ state of the FlexTDMA Protocol. Using this table each current eligible frame will be scheduled in either the FIFO queue or the flow$_{01}$ queue. Table 4.2 row 1 shows that when the baseline deadline density is not above average the frame is queued in the FIFO queue. The status of flow$_{01}$ transmission opportunity at the flow deadline, transmission opportunity found within the duration limit of

Table 4.2 FlexTDMA+ Queuing Decision Logic for Baselined State

	Baseline Dead-lineDen-sityAbove Average	**Flow$_{01}$ Availability at Deadline**	**Flow$_{01}$ Availability Within Duration Limit**	**Preemption Eligible**	**Queue**
1	No	NA	NA	NA	FIFO
2	**Yes**	**Yes**	NA	NA	Flow$_{01}$ at deadline
3	**Yes**	No	No	No	FIFO
4	**Yes**	No	No	**Yes**	Preempt Flow$_{01}$ at deadline
5	**Yes**	No	**Yes**	No	Flow$_{01}$ at latest available location
6	**Yes**	No	**Yes**	**Yes**	Flow$_{01}$ at latest available location

flow$_{01}$ and the preemption eligibility status of the scheduled frame in flow$_{01}$ are not considered. This follows as the only motivation to baseline a baselined flow is to reduce the baseline deadline density of the flow. Table 4.2 row 2 shows that when flow$_{01}$ has a transmission opportunity at the flow deadline the frame will be scheduled in flow$_{01}$ at the deadline time. Table 4.2 row 3 shows that when there is no flow$_{01}$ transmission opportunity and the scheduled frame in flow$_{01}$ is not preemption eligibility, the frame is queued in the FIFO queue. Table 4.2 row 4 shows that when the current frame is eligible for preemption of the scheduled frame in flow$_{01}$, a preemption will be performed. Table 4.2 row 5 shows that when a transmission opportunity is found within the duration limit of flow$_{01}$, the frame is scheduled in the flow$_{01}$ queue at the located transmission opportunity. This is a partial baseline. Table 4.2 row 6 shows that when a transmission opportunity is found within the duration limit of flow$_{01}$ and the current frame is eligible for preemption of the scheduled frame in flow$_{01}$, the frame will be scheduled in flow$_{01}$ scheduled at the latest time available. This is a partial baseline. Here a partial baseline is preferred to preemption since the flow is already baselined. There is no need to preempt another flow to simply update the baseline deadline of the current baselined flow.

Table 4.3 shows the queue decision logic for the 'Baselined Exceeded' state of the FlexTDMA Protocol. Using this table each current eligible frame will be scheduled in either the FIFO queue

Table 4.3 FlexTDMA+ Queuing Decision Logic for Baselined Exceeded State

	Baseline Deadline Density Above Average	**Flow$_{01}$ Availability at Deadline**	**Flow$_{01}$ Availability Within Duration Limit**	**Preemption Eligible**	**Queue**
1	NA	**Yes**	NA	NA	Flow$_{01}$ at deadline
2	NA	No	No	NA	FIFO
3	NA	No	**Yes**	NA	Flow$_{01}$ at latest available location.

or the flow$_{01}$ queue. The 'Baseline Deadline Density Above Average' column is NA since the flow baseline deadline has been exceeded. The emphasis is on renewing the baselined status rather than baseline density control. The preemption eligibility status of the flow is column is NA. When the flow baseline has been exceeded, but the flow is baselined, no preemption will occur. Instead the flow will either be baselined or partial baselined. Table 4.3 row 1 shows that when flow$_{01}$ has a transmission opportunity at the flow deadline the frame will be scheduled in flow$_{01}$ at the deadline time. Table 4.3 row 2 shows that when no transmission opportunity is found within the duration limit of flow$_{01}$ the frame is queued in the FIFO queue. Table 4.3 row 3 shows that when a transmission opportunity is found within the duration limit of flow$_{01}$, the frame is scheduled in the flow$_{01}$ queue. This is a partial baseline.

4.5 FlexTDMA+ Evaluation

In this section we provide an evaluation of the FlexTDMA+ protocol.

4.5.1 Phases of Testing

There are two phases of testing.

4.5.1.1 Testing Phase: FlexTDMA+ Periodic On-Off Probability

The testing phase FlexTDMA+ periodic on-off probability focusses on the influence of the probability of periodic on-off on the performance of FlexTDMA.

4.5.1.2 Testing Phase: FlexTDMA+ Flow Loss probability

The testing phase FlexTDMA+ frame loss probability focuses on the influence of the probability of frame loss on the performance of FlexTDMA+.

4.5.2 Protocol Parameters of Testing

There are three protocol parameters evaluated individually and in combination.

4.5.2.1 Baseline Density Control

For each test run the baseline density control is either turned on or off. When baseline density control is turned on a flow may potentially be baselined prior to the flow baseline deadline.

4.5.2.2 Partial Baselining

For each test run partial baselining is either turned on or off. When partial baselining is enabled, a frame may be baselined using a scheduled time that is less that the deadline of the flow.

4.5.2.3 Preemption

For each test run preemption is either enabled or disabled. When preemption is enabled baseline preemption may be utilized.

4.5.3 Test Runs Performed

The FlexTDMA+ protocol was evaluated by configuring the network operational parameters (i.e. probability of frame loss) and the FlexTDMA+ parameters (i.e. enablement of baseline deadline density reduction) and performing a run using the configuration. Data is collected for each key performance criteria during each run.

Table 4.4 shows the parameters configured for each test run along with the values the parameter was configured. There are 384 total combinations of the parameter values shown.

Table 4.4 FlexTDMA+ Parameters Tested

FlexTDMA+ Phase	FlexTDMA+ Phase: Periodic On-Off	FlexTDMA+ Phase: Frame Loss
Periodic On-Off	4 Probabilities	-
Frame Loss	-	4 Probabilities
Preemption	2 (On/Off)	2 (On/Off)
Partial Baselining	2 (On/Off)	2 (On/Off)
Baseline Density Control	2 (On/Off)	2 (On/Off)
Clock Drift Modes	4 (none, increasing, decreasing, mixed)	4 (none, increasing, decreasing, mixed)
Bandwidth Load	3 (20, 50, 90%)	3 (20, 50, 90%)
Total combinations	384	384

A test run was performed for each combination for a total of 384 runs for each of the two phases. There were 4 probability values selected for periodic on-off and frame loss. These were selected to induce low, medium and high levels of flow discontinuations and resumptions. The preemption, partial baselining and baseline deadline density control improvements features were enabled and disabled showing the performance with and without them. The evaluation included four clock drift configurations: 1) no clock drift in nodes or switches, 2) increasing clock drift along the forwarding path of flows in the network, 3) decreasing clock drift along the forwarding path of flows in the network and 4) mixed clock drift in the flow propagation path. Several bandwidth loads were tested to determine the sensitivity of FlexTDMA+ to bandwidth variations.

4.5.4 Key Performance Criteria

The key performance criteria of the FlexTDMA+ protocol are: time-to-baseline, delay-jitter and laxity.

The time-to-baseline criterion is defined as the time needed for an active flow to achieve a baselined state. This is an important criterion as the performances of FlexTDMA+ are not fully provided until a flow has entered a baselined state. Entering a baselined state allows a FlexTDMA switch to nearly reconstruct the traffic envelope of the arriving flow.

The delay-jitter criterion is defined as the extent of compression between any two arriving frames (7). When a flow is baselined the FlexTDMA+ switch is able to nearly reconstruct the traffic envelope of the arriving flow, which minimizes the delay-jitter values achieved for frames delivered to destination nodes.

The laxity criterion is defined as the extent to which data is delivered to a destination node prior to the delay bound of the flow. For example when the flow delay bound is 10 ms and a frame is delivered in 9 ms the laxity is 1 ms. Laxity is used, rather than age of received data, since this allows a common characterization of all flows to all flow destinations. For example when flow 1 and flow 2 have delay bounds to their destinations of 10 ms and 100 ms with actual delivery times of 9 ms and 99 ms both have laxity performance of 1 ms. This allows comparison of the performance offered to both flows. This is useful since FlexTDMA+ will ideally deliver each frame at the delay bound of the flow.

4.5.5 Testing Topology Details

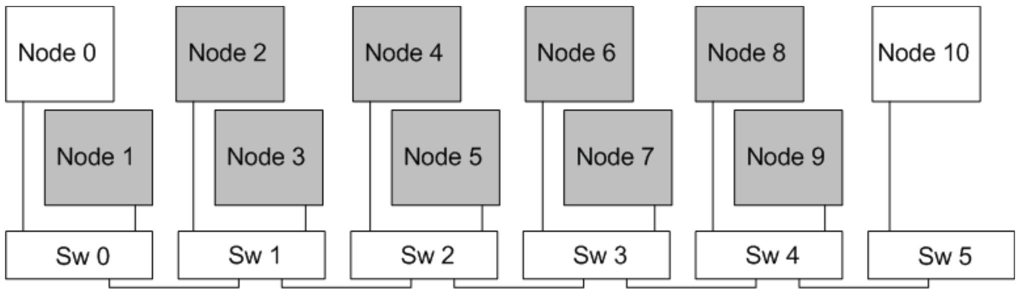

Figure 4.12 FlexTDMA+ Testing Topology

The topology in Figure 4.12 shows the simulated physical topology used to demonstrate the FlexTDMA+ protocol. The topology is configured so that nodes 0 to 9 transmit to node 10. In this way the loading on the switches increases from switch 0 to switch 4. Switch 0 carries traffic from nodes 0 and 1, while switch 4 carries traffic from node 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. The total bandwidth is divided among the flows allocated to each node so that the bandwidth allocation varies. The variation in flow bandwidth allocation causes the periodic transmissions to not maintain a fixed relative offset, but instead to phase so that collisions occur. The delay bounds for the flows of this testing topology range from 227 μs (0.227 ms) to 1133 μs (1.133 ms). The magnitude of these delay bounds is important to keep in mind relative to the performance criteria evaluation that follows.

4.5.6 Results

4.5.6.1 Clock Drift Effect on FlexTDMA+

Table 4.5 shows a summary of the effect of clock drfit on the FlexTDMA+ protocol for each key performance criterion and for each phase of evaluation.

Table 4.5 Clock Drift Effect on FlexTDMA+

FlexTDMA+ Phase	FlexTDMA+ Phase: Periodic On-Off	FlexTDMA+ Phase: Frame Loss
Time To Baseline (range of observed values)	0.977 to 7.05 ms	0.935 to 7.12 ms
Time To Baseline (Variation between drift types)	1 to 11%	3 to 11%
Time To Baseline (Average per bandwidth load)	20% - 1 ms 50% - 1.5 ms 90% - 6 ms	20% - 1 ms 50% - 2.2 ms 90% - 6 ms
Delay Jitter	0.035 to 0.05 μs	0 to 0.371 μs
Laxity	0.169 to 7.21 μs	2.7 to 133 μs

Time-to-baseline had the highest impact from increasing drift. This follows as increasing drift accelerates the number of baselines needed. All time-to-baseline differences resulting from different drift modes are small compared to differences generated from other run parameters. The delay-jitter performance was sub μs. Frame delay bound laxity ranged from 3% to 59% of minimal flow delay bounds, when considering all potential improvements. FlexTDMA+ managed clock drift efficiently for all modes of operation for each key performance criteria, and clock-drift had a minimal but consistent result on performance.

The difference made by clock drift on the key performance criteria is small relative to the other tested parameters. Further evaluation will be limited to increasing drift rate. This reduces the total number of run values to be compared. Total runs per phase assuming a single clock drift configuration mode is $96 = 8$ (improvement combinations of preemption, partial baselining, baseline density control) X 1 (single drift rate) X 3 (bandwidth loads) X 4 (frame loss or periodic on-off).

4.5.6.2 Bandwidth Load on FlexTDMA+

The total bandwidth loading of the network is determined for each test. Table 4.6 shows the average performance relating to the three bandwidth loads of 20%, 50%, and 90%. The delay-jitter performance under FlexTDMA+ was sub μs. The bandwidth load has a consistent effect on the resulting key performance criteria, as time-to-baseline and laxity when baselined were reduced as bandwidth loads were reduced. This is the result of reduced demand for baselining opportunities in flow$_{01}$. As demands lower the protocol is more responsive to re-baselining needs.

Further evaluation will be limited to a single bandwidth load of 90%. Total runs per phase assuming a single bandwidth loading is $32 = 8$ (improvement combinations of preemption, partial baselining, baseline density control) X 1 (single drift rate) X 1 (single bandwidth load) X 4 (frame loss or Periodic On-Off).

Table 4.6 Impact of Bandwidth Load on FlexTDMA+

FlexTDMA+ Phase	FlexTDMA+ Phase: Periodic On-Off	FlexTDMA+ Phase: Frame Loss
Time To Baseline (average)	1.06, 1.81 and 5.70 ms	1.05, 2.05 and 6.02 ms
Delay Jitter	0 to 0.03 μs	0 to 0.37 μs
Laxity (average)	1.6, 2.0 and 7.2 μs	12.6, 22.1, and 28.6 μs

4.5.6.3 Effect of Probability of Periodic On-Off and Frame Loss on FlexTDMA+

Table 4.7 shows the increase achieved by modifying the probability of periodic on-off and frame Loss. The delay-jitter performance under FlexTDMA+ was sub micro-second. The periodic on-off probability and frame loss probability have a consistent effect of increasing both time to baseline and frame delay bound laxity. This follows as a higher periodic on-off probability and frame loss probability means that the density of demanded baselines is increased. Further evaluation will use maximum probability values for both periodic on-off probability and frame loss probability when evaluating FlexTDMA+ protocol improvements.

Total runs per phase assume a single Periodic On-Off probability and Frame Loss probability is 8 = 8 (improvement combinations of preemption, partial baselining, baseline density control) X 1 (single drift rate) X 1 (single bandwidth load) X 1 (maximal frame loss or Periodic On-Off).

Table 4.7 Impact of the Probability of Flow Interruption on FlexTDMA+

FlexTDMA+ Phase	FlexTDMA+ Phase: Periodic On-Off	FlexTDMA+ Phase: Frame Loss
Time To Baseline	9.6%	7.6%
Delay Jitter	0 to 0.244 μs	0 to 0. 371 μs
Laxity	24.1%	28.8%

4.5.6.4 Improvements to FlexTDMA+

Table 4.8 Performance Of FlexTDMA Improvements

Performance Criteria	Improvements	Performance
Time-to-Baseline	No clear difference	5.6 - 7.1 ms
Laxity	Partial Baselining and Baseline Preemption	2 - 10 μs
	Partial Baselining	5 - 10 μs
Delay Jitter	Baseline Density	0.06 - 0.23 μs
	Baseline Density, Partial Baselining and Baseline Preemption	0.08 - 0.23 μs
	Partial Baselining and Baseline Preemption	0.1 - 0.24 μs
	Baseline Density and Partial Baselining	0.12 - 0.22 μs

There were three improvements to FlexTDMA: 1) baseline deadline density control, 2) partial baselining and 3) baseline preemption. We consider the influence these improvements, used individually and in combination, have on the key performance criteria: 1) time-to-baseline, 2) laxity baselined and 3) delay-jitter. Table 4.8 shows the results of testing all combinations of the three protocol improvements to FlexTDMA: partial baselining, baseline preemption and baseline deadline density control. The table includes those combinations of the three protocol improvements that resulted in improved performance compared to no protocol modification. The time-to-baseline performance criteria had no clear performance improvement for any of the protocol improvement combinations. All combinations offered performances in the same range. The time-to-baseline performance was typically 5 to 26 times the flow delay bound depending on the delay bound of the flow. Partial baselining only approximates a baselining frame transmission and requires additional utilization of baselining transmission opportunities once a full baseline state is achieved. A performance improvement from baseline preemption was expected, but the improvement was mitigated by low concurrent baselining demand as most flows do not usually require re-baselining at the same time. Baseline deadline density control had little effect on time-to-baseline as concurrent re-baselining is low. The laxity performance criteria was improved in two combinations. The first combination was partial baselining and baseline preemption and the second was partial baselining alone. The laxity performance was typically 0.04% to 2% of the flow delay bound, when using optimal improvements. Partial baselining allows each flow to approximate the baselining timing prior to being baselined allowing more frames to be delivered nearly at the deadline. Baseline preemption allows flows requiring a baseline status or flows having experienced more clock drift to be baselined first keeping the computed maximal delay bounds more accurate. The delay-jitter performance criterion was improved in four combinations. The first combination was baseline deadline density control, the second was baseline deadline density control, partial baselining and baseline preemption, the third was partial baselining and baseline preemption, the fourth was baseline deadline density control and partial baselining.

The inclusion of baseline deadline density control increased the variance of the laxity performance and therefore was not selected as a recommended improvement. Under this policy flows are scheduled to utilize baselining opportunities only because of the relative baseline deadline relationship. There is enough variability in the frame arrival behavior, given frame loss and periodic on-off transmissions, that the flow baseline density control was not beneficial. When a network has low probabilities of frame loss and periodic on-off transmissions baseline deadline density control also would not be advantageous as there would be relatively lower demands for baselining.

We conclude that the two improvements, partial baselining and baseline preemption, should be defined in FlexTDMA+. This improvement combination showed the same performance for time-to-baseline, the best performance for laxity and sub μs delay-jitter performance.

4.6 Chapter Summary

In this chapter we proposed and evaluated 3 improvements to the FlexTDMA protocol: partial baselining, baseline preemption and baseline deadline density control. In our evaluation we first characterized the ability of FlexTDMA+ to manage clock drift. We determined the effect flow transmission interruption has on the FlexTDMA+ performance. We performed full evaluation of the proposed improvements to FlexTDMA using 90% bandwidth loading. We demonstrated that two improvements, partial baselining and baseline preemption, together offered the most improvement in performance compared to FlexTDMA. Partial baselining improves performances by approximating a baselined state until the flow can actually be baselined. Baseline preemption insures baseline opportunities are granted to those flows in most need.

CHAPTER 5. FlexTDMA++ Simultaneous Multicast

In this chapter we consider the enhanced form of FlexTDMA called FlexTDMA++ which expands FlexTDMA to support simultaneous multicast. We evaluate the simultaneous multicast performance under real-world conditions of switch failures, end node periodic on-off transmission, and network conditions of clock drift, frame loss and network bandwidth load. We evaluate the simultaneous multicast performance in the presence of these network conditions.

5.1 Introduction

In many closed industrial control networks, a central controller may try to control multiple points simultaneously, and once it issues a command, the command should be multicast to all networked nodes. In many applications, for the global task to proceed correctly, it is necessary that the command be received at all receiver nodes nearly at the same time, and with very small delay-jitter. From a system perspective, all nodes receiving the multicast message will react at the same time. At the same time, from the perspective of any single node, fair access to the received data is provided. This is referred to as simultaneous multicasting (SM).

This chapter considers the SM problem in industrial control systems, and introduces the FlexTDMA++ protocol. This strategy will guarantee that data is delivered from the source to any receiver in the multicast session with a nearly constant delay bound, and that the delay-jitter within the flow of frames delivered to the same node is minimized. Moreover, minimum delay variation between multiple nodes, which are receivers within the multicast session, will be achievable. This is done under the assumption of periodic on-off traffic, and will be supported when frames may be lost, when switches may fail, and when components may exhibit clock drifts and sustain different loads.

The literature contains a number of solutions to achieve bounded delay periodic traffic, constant delayed periodic traffic and simultaneous delivery of multicasting data. Bounded delay periodic traffic is supported in (64) using a probabilistic model, and in (56) by exchanging messages for synchronous operation. References (45) (60) (62) (61) support nearly constant delayed traffic in a synchronous network, and require message exchange to maintain a synchronous state. Simultaneous message arrival is supported in (57) with a solution not designed for packet networks, and in (30) for TCP Internet connections by using bandwidth reservation. The authors of (42) achieve SM by attaching a transmit release time-stamp to messages while maintaining a synchronized state. The authors of (40) achieve SM by maintaining a synchronized state between the members of the multicast group. These solutions require a synchronous state, have probabilistic delay bounds, require message exchanges for synchronous operation, are not suitable for sub millisecond message exchanges, or are not suitable for a packet network.

We propose a protocol for simultaneous delivery of multicast data in an asynchronous packet network without the use of clock coordination or message time stamping. In (63) we introduced FlexTDMA to provide minimal delay-jitter with nearly maximal delays in an asynchronous network. FlexTDMA works by periodically transmitting a maximally delayed frame on each flow allowing downstream switches to establish a maximal eligibility time (ET) basis, where ET is the time at which an arriving frame is in conformance with the original traffic envelope of the transmitting node. Each FlexTDMA switch traffic shapes arriving frames using this ET basis, providing maximal constant delays in an asynchronous network. The FlexTDMA protocol shares maximal delay bound transmission opportunities in a process called baselining. We expand the consideration of FlexTDMA to include end node periodic on-off traffic, switch failures and network conditions while supporting SM (FlexTDMA++). Three network conditions are considered: clock drift, frame loss due to bit errors, and bandwidth load. The FlexTDMA++ SM improvement provides data delivery with maximal delay performance for multicast data.

Utility of Research: The simultaneous multicast improvement to FlexTDMA included in FlexTDMA++ provides data delivery with maximal delay performance for simultaneous multicast and is considered in presence of frame loss in the network, periodic on-off, switch failure, and clock-drift.

Switch failure is a case that must be taken into account when considering the operation of FlexTDMA++ simultaneous multicast. In this context switch failure refers to any circumstance when a switch discontinues function for a time period. This may occur during a power outage, a switch reset, switch upgrade of hardware or software, or switch replacement. In any case when a switch in a large closed network discontinues function the remainder of the network should continue to function.

Proposed Research - FlexTDMA++ Simultaneous Multicast Support: The improvements to FlexTDMA included in FlexTDMA++ allow the support of simultaneous multicast. This allows FlexTDMA to deliver periodic message traffic to each individual destination node nearly at a constant delay while also simultaneously delivering each periodic message to all destination nodes. Our research focuses on the ability of FlexTDMA++ to provide simultaneous multicast in the presence of real life network operating conditions. This includes various bandwidth loads, node and switch drift rates, frame loss, periodic on-off and switch failures.

The remainder of this chapter is organized as follows. In Section 5.2 we discuss the improvements to the FlexTDMA protocol offered in FlexTDMA++. In Section 5.3 we describe network operational characteristics and the influence on FlexTDMA. In Section 5.4 we review simultaneous multicast existing implementations and support under FlexTDMA++. In Section 5.5 we review Gang Scheduling approaches used to support concurrent scheduling. In Section 5.6 we characterize the delay bound calculations used to support simultaneous multicast under FlexTDMA++. In Section 5.7 we review the operational details of the FlexTDMA++ protocol. In Section 5.8 we provide the evaluation approach of the FlexTDMA++ protocol. In Section 5.9 we summarize the findings of the improvements found under the FlexTDMA++ protocol.

5.2 Improvements

In this section we review SM existing implementations and then explain how FlexTDMA++ supports it.

5.2.1 Simultaneous Multicast Support in FlexTDMA++

The FlexTDMA protocol is enhanced to provide simultaneous multicast of periodic message traffic. When transmitting nodes introduce a periodic message frame which is multicast to multiple destinations each destination will receive a copy of the original message nearly at the same time.

Mitigation of switch failure and resumption is a critical aspect of simultaneous multicast in a network topology. As switches fail and resume operation the delay-depth of the topological tree supported by simultaneous multicast changes. The FlexTDMA++ support for simultaneous multicast in a topology where switches fail and resume operation was evaluated as part of this research.

Following a switch failure or resumption no flow re-routing was performed in this FlexTDMA++ research. Instead static flow routing was maintained in the presence of switch failures and resumptions. This insured that the evaluation of the FlexTDMA++ constant delay bound and simultaneous multicast properties were not influenced by re-routing issues, but rather the evaluation demonstrated direct results of the properties of FlexTDMA++.

5.2.2 FlexTDMA++ Implementation Overview

To support simultaneous multicast each FlexTDMA++ switch computes the maximal delay depth of the forwarding tree for each message flow. This information is passed up the hierarchical tree and used by predecessor switches to determine maximal delay depth. In this way the FlexTDMA++ switch is able to determine a deadline for each forwarded copy of a message so that it will contribute to the simultaneous message arrival of each forwarded message instance.

5.3 Network Operation

In this section we describe network operational characteristics and the influence they have on FlexTDMA++. Network operational characteristics bandwidth loading, periodic on-off, frame loss due to bit errors, and clock drift were reviewed in Chapter 4.

5.3.1 Switch Failures

Switch failures and resumption are problematic to the FlexTDMA++ protocol for two reasons. First, the simultaneous delay bounds among multicast elements for each forwarded multicast flow change when the switch fails and when it resumes. This requires updates to the eligibility time basis established at baseline time for flow paths not traversing failed switches. Second, when a switch resumes function baseline updates are required to downstream switches as well as potentially sibling switches.

5.4 Simultaneous Multicasting

In this section we introduce the basics of FlexTDMA++, and how it supports SM, including gang scheduling and preemption approaches.

5.4.1 FlexTDMA++ Simultaneous Support Mechanisms

The FlexTDMA protocol baselines each output port independently as baselining opportunities arise. Thus the baselining process is not coordinated between multicast forwarded output ports on each flow.

5.4.1.1 Coordinated Baselining

To enhance FlexTDMA to offer simultaneous multicast function the baselining process of each flow must be coordinated so that a flow is baselined using a common receive frame at all switch output ports for which the flow is forwarded. When a FlexTDMA++ switch initiates a baseline event on a flow for each forwarded frame instance (to each output port) resulting from the same receive frame, all sub-trees will experience a baseline cascade. Each sub-tree

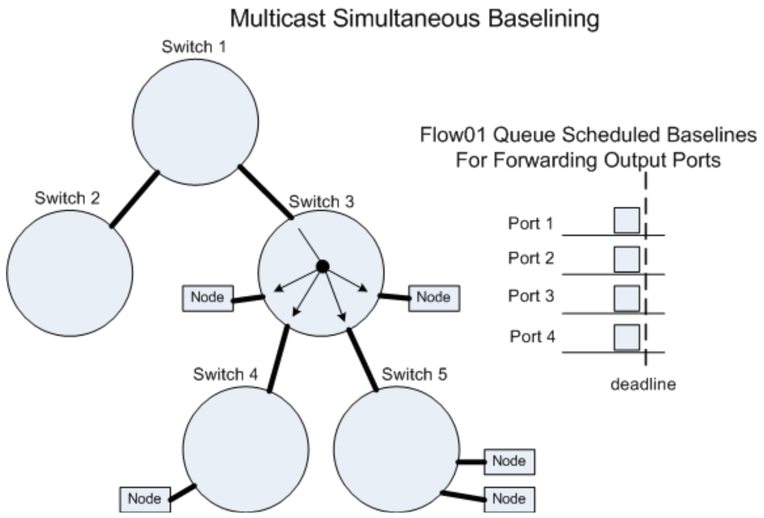
Multicast Simultaneous Baselining
Switch 1
Switch 2
Switch 3
Switch 4
Switch 5
Node
Node
Node
Node
Node
Flow01 Queue Scheduled Baselines
For Forwarding Output Ports
Port 1
Port 2
Port 3
Port 4
deadline

Figure 5.1 Simultaneous baselining at all forwarded ports establishes a common eligibility time basis.

root switch will then schedule a baseline event on the flow at the earliest opportunity. This will maintain the multicast tree with a balanced delay-depth for each sub-tree. Figure 5.1 shows how concurrent baselining of a flow forwarding set in a switch establishes a common eligibility time basis for each member output port. Switch 3 has a flow forwarded to two connected nodes and switches 4 and 5. When this flow is baselined, a baselining event is scheduled using a $flow_{01}$ baselining opportunity at each output port. By using the same frame forwarding event the eligibility time basis of each connected switch is updated. This means that the two sub-trees switch 4 and switch 5 receive the baselined frame resulting from the same received frame in switch 3. Switches 4 and 5 will in turn establish a common eligibility time basis to each forwarding path of the flow.

Coordinated baselining at multiple output ports of a FlexTDMA++ switch requires that a bin-packing type algorithm be applied to baselining opportunities across switch output ports. This fits a general class of problems where job-sets are scheduled on multiple machines. Each arriving job requires a different number of machines to complete service. Much research has been applied to job scheduling on multiple machines. From this research an algorithm called Gang scheduling is defined which requires the task set of each job be scheduled to run in parallel across the machine set (34), (25), (21).

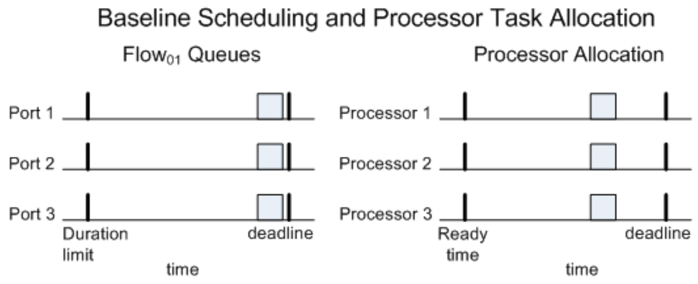

Figure 5.2 Flow baseline gang scheduling is equivalent to processor gang scheduling.

Figure 5.2 shows a flow scheduled for baselining at three output ports and a set of three tasks scheduled for concurrent execution on three processors. These two cases are similar in that three tasks of a job are scheduled concurrently on a resource for each job instance. The flows must be scheduled in the interval $[duration\,limit, deadline]$ and the task instances must be scheduled in the interval $[ready\,time, deadline]$. Each case presents a bin fitting problem to be solved as jobs arrive for scheduling.

When Gang scheduling is applied to processor scheduling, the tasks within each job are scheduled so that the tasks run concurrently. The advantage of Gang scheduling in processor task allocation is that efficient busy waits are provided since the tasks are running at the same time as if they were allocated on a common virtual processor. Gang scheduling solves the problem of bin-fitting the task sets generated from each job onto the run time schedule of each processor.

5.4.1.2 FlexTDMA++ Gang Scheduling

Under Gang Scheduling, tasks of a job are scheduled to run in the same time slice. This is often used to schedule software tasks of jobs onto a set of available processors (39) (6).

Figure 5.3 shows tasks of task sets generated from a single job scheduled concurrently on multiple processors. Bin fitting is used to maximize the utilization of processors. Notice that there is a gap between the termination of tasks from job 3 until the invocation of the tasks from job 4. This gap is needed so that all tasks of job 4 can be scheduled concurrently. Multiple jobs may be scheduled when the processor sets for the jobs are disjoint. This allows multiple jobs to be scheduled concurrently when disjoint processor sets can be selected to support the tasks.

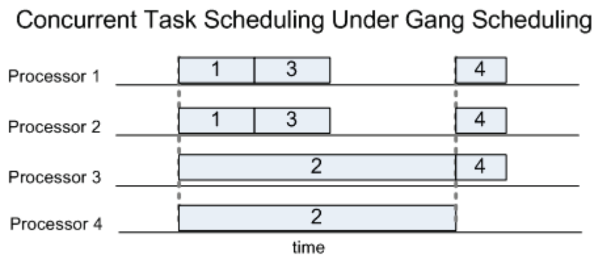

Figure 5.3 Under gang scheduling of tasks, all task instances of a task are scheduled to run concurrently.

Figure 5.3 shows this as tasks 1 and 2 are scheduled concurrently since the required processor sets are disjoint.

In the case of FlexTDMA++, Gang scheduling provides the mechanism to schedule coordinated baselining events. In this context a job is a frame arriving on a flow and the task-set is the set of output ports the associated flow must be transmitted (16). This follows as transmission capacity of each egress port (giga-bits-per-second to support the bits of the frame) is equivalent to processor computation capacity (instructions-bits-per-second to support the computation of a task). When an arriving frame on a flow is selected for Gang scheduled baselining, a baseline event will be planned for each forwarding instance of the frame to each output port. In this way all output ports to which the received flow is forwarded will have a baseline performed in parallel. This insures that all sub-trees from the current switch have a common concept of base time as a baseline event will occur at all output ports. The baseline event fundamentally establishes a relationship between the eligibility time of a frame in the current switch and the planned eligibility time in subsequent switches. Thus, the root-switch of all multicast sub-trees has an established consensus of eligibility time following the Gang scheduled baselining events for a multicast flow. The baselining time for a flow is chosen so that a baseline event can occur on all output ports for which the flow is forwarded.

5.5 FlexTDMA++ Gang Scheduling and Preemption Approaches

Here we review the gang scheduling approaches and preemption strategies.

5.5.1 Preemption Classes

There are three classes of preemption under FlexTDMA++ Gang scheduled multicasting.

5.5.1.1 Preemption : None

The preemption type 'none' allows no preemption (26). Under this preemption class, once a baseline event is scheduled it cannot be preempted by a subsequent frame arrival.

5.5.1.2 Preemption : On a Per Output Port Basis

The preemption type 'Per Output Port' allows preemption to occur on a per output port basis (26). Under this preemption class each individual output port scheduled baseline may be preempted while leaving the remaining output ports scheduled baselines in place.

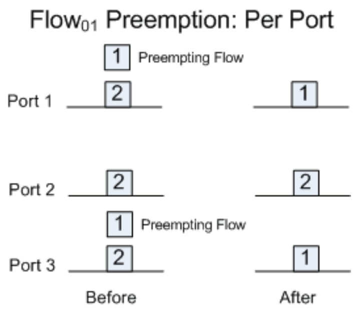

Figure 5.4 Under flow$_{01}$ per port preemption each output port can be individually preempted.

Figure 5.4 shows flow 1 preempting flow 2 which is scheduled at output ports 1, 2 and 3. Flow 1 has preemption precedence over flow 2. Under the *per port* preemption policy a frame may be preempted at an output port without preempting other forwarding instances of the same frame. In this case flow 2 is preempted at output ports 1 and 3. However the flow 2 frame instance scheduled at output port 2 remains scheduled.

5.5.1.3 Preemption : Strict Gang

The preemption type 'Strict' allows preemption but strict Gang (concurrent scheduling) must be maintained (26). Under this preemption class baselines are Gang scheduled at output ports, and can be preempted. However, the Gang scheduled status of each output port set baseline must be maintained. This is an all or nothing approach so that if any one output port must be preempted then all forwarding output ports of the same flow must also be preempted.

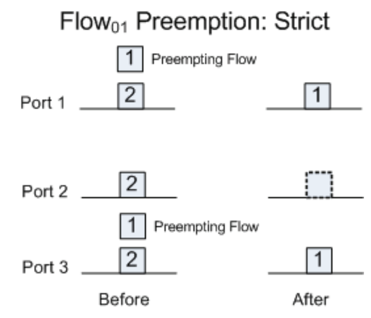

Figure 5.5 Under flow$_{01}$ strict port preemption all forwarded frame instances must be preempted.

Figure 5.5 shows flow 1 preempting flow 2 which is scheduled at output ports 1, 2 and 3. Flow 1 has preemption precedence over flow 2. Under the *strict* preemption policy when a frame is preempted at an output port other forwarding instances of the same frame must also be preempted. In this case flow 2 is preempted at output ports 1 and 3. The flow 2 frame instance scheduled at output port 2 must also be preempted under the strict preemption policy even though there is no direct preemption at the port.

5.5.2 Gang Scheduling Bin Fitting Strategies

For FlexTDMA++ support of simultaneous multicast there will be six scheduling approaches evaluated based on the Gang Scheduling policies for determining when to apply baselining at switch output ports. These approaches are detailed below. In addition the performance of FlexTDMA++ simultaneous multicast performance will be evaluated when no output port baseline coordination is performed. This will serve as a base comparison for the other approaches.

5.5.2.1 First Fit

Under the first fit policy (35) (36) (58) a Gang scheduler allocates the first set of available machines at the first matching opportunity. In the case of FlexTDMA the first time a frame arrives on a flow requiring baselining, where a baseline event is available at the output port set,

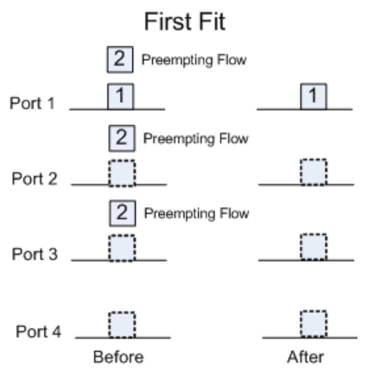

Figure 5.6 Flow 2 will be blocked as Flow 1 was scheduled first.

the baseline event is scheduled. The disadvantage of this approach is that the next arriving flow may make better use of the output ports so that some output port baseline opportunities go unused. The advantage of this approach is that the algorithm is simplified in that no preemption is performed. The FCFS and Greedy are terms used for the same scheduling technique. Figure 5.6 shows the First Fit gang scheduling algorithm. Flow 1 is scheduled at output port 1 and output ports 2, 3, and 4 are idle. The preempting flow 2 arrives and will be transmitted on output ports 1, 2, and 3. Under the First Fit gang scheduling algorithm flow 2 will not be scheduled for baselining in flow$_{01}$ since flow 1 would need to be preempted. Notice that output ports 2 and 3 remain idle.

5.5.2.2 Concurrent Gang

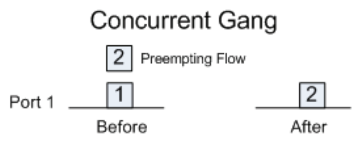

Figure 5.7 Baseline deadline laxity of flow 2 is less than flow 1, thus Pri(2) > Pri(1), so flow 2 preempts flow 1.

Under a concurrent gang policy (34) a Gang scheduler allocates or preempts using a function fA(T) = grade of task. For FlexTDMA++ this grade will be equal to time remaining to the baseline deadline. This approach classifies arriving flows as either Mandatory (baselined and allocated to flow$_{01}$) or Not Mandatory (not baselined and allocated to FIFO). The preemption priority escalates as baseline deadline time approaches. This Gang scheduling policy is tested using preemption types Strict Gang and Per Output Port. Figure 5.7 shows the Concurrent gang scheduling algorithm. Flow 1 is scheduled at output port 1. The preempting flow 2 arrives and has a higher priority than flow 1 as the baseline deadline of flow 2 is closer than flow 1.

5.5.2.3 Lazy Gang

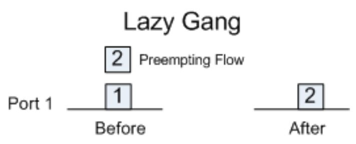

Figure 5.8 Number of frames received on flow 2 when baseline is needed is greater than flow 1, so Pri(2) > Pri(1).

Under a Lazy gang policy (25) a Gang scheduler allocates or preempts using flow priority. Preemption is based on priority, and is established based on the number of frame arrivals with the baseline deadline exceeded. The preemption priority escalates each time a flow pending baseline has an arriving frame that is not selected for baselining. This Gang scheduling policy is tested using preemption types Strict Gang and Per Output Port. Figure 5.8 shows the Lazy gang scheduling algorithm. Flow 1 is scheduled at output port 1. The preempting flow 2 arrives and has a higher priority than flow 1 as the number of frames received on flow 2 with the baseline deadline exceeded is more than the number received on flow 1.

5.5.2.4 Best Fit

Under a Best Fit policy (35) (36) a Gang scheduler gives priority to a flow based on the number of wasted (idle) output ports resulting from scheduling the flow for concurrent baselining. The preemption priority increases when the new job wastes fewer output slot baselining opportunities. This approach mimics the large resource first bin-fitting algorithm. This Gang scheduling policy is tested using preemption types Strict Gang and Per Output Port. Figure 5.9 shows the Best Fit gang scheduling algorithm. Flow 1 is scheduled at output ports 1, 3 and 4. The preempting flow 2 arrives and is forwarded to output ports 1, 2, 3 and 4. In this case

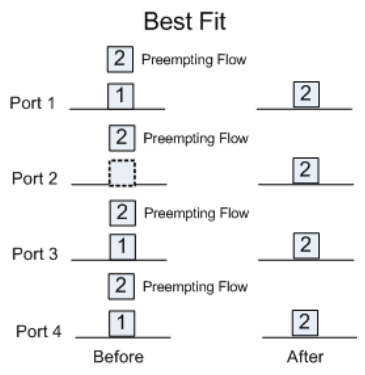

Figure 5.9 When flow 2 preempts flow 1 the number of idle ports is reduced from 1 to 0, so Pri(2) > Pri(1).

the priority of flow 2 is greater than the priority of flow 1 since following preemption the net number of idle ports will be reduced. Thus flow 2 has a better fit than flow 1.

5.5.2.5 FCFS w/Backfill

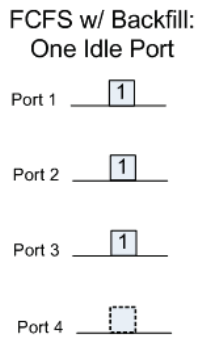

Figure 5.10 At probability ($P > 5\%$) case 1 the frame is not scheduled in flow$_{01}$ for baselining.

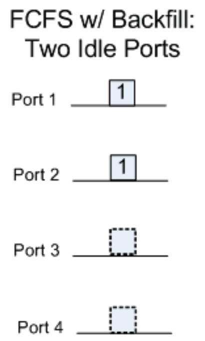

Figure 5.11 At probability ($P > 25\%$) case 1 the frame is not scheduled in flow$_{01}$ for baselining.

Under the FCFS w/Backfill policy (36), (15), (29) a Gang Scheduler, with a certain probability does not commit when one or more ports are left idle (increasing the probability with the number of idle ports). Once a baseline is scheduled preemptions are not allowed. Three cases of probability values are tested. Probability of no commit with 1 idle output port: case 1=5%, case 2=10%, and case 3=15%. Probability of no commit with 2 idle output ports: case 1=25%, case 2=50%, and case 3=75%. Figure 5.10 shows the FCFS with Backfill gang scheduling algorithm. Flow 1 has an arriving flow that is to be scheduled at output ports 1, 2 and 3 leaving output port 4 idle. When a single port is left idle FCFS with Backfill policy will not utilize flow$_{01}$ (instead using the FIFO queue) at a probability of (P>5%). Figure 5.11 shows the FCFS with Backfill gang scheduling algorithm. Flow 1 has an arriving flow that is to be scheduled at output ports 1 and 2 leaving output ports 3 and 4 idle. When two ports are left idle FCFS with Backfill policy will not utilize flow$_{01}$ (instead using the FIFO queue) at a probability of (P>25%).

5.5.2.6 Bandwidth Weighted Fit For FlexTDMA

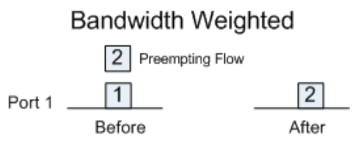

Figure 5.12 Bandwidth allocated to flow 1 is greater than flow 2, so Pri(2) > Pri(1).

We propose the Bandwidth Weighted Fit algorithm here to determine which arriving frame to a FlexTDMA++ switch should be selected for Gang Scheduled multicast concurrent transmission or simultaneous baselining. This Gang Scheduling selection approach is chosen in consideration of properties of FlexTDMA. This approach does not directly relate to a known bin fitting algorithm, but is practical specifically in the context of FlexTDMA++ Gang scheduling of multicast traffic.

Under the Bandwidth Weighted Fit policy a Gang Scheduler will commit a flow, for which the baseline deadline is exceeded, to a multicast baseline event when either the flow$_{01}$ flow at each output port is idle, or will preempt when the allocated bandwidth of the flow currently scheduled on flow$_{01}$ is lower than the flow scheduled for multicast baselining. Under the Bandwidth Weighted Fit policy preemption occurs regardless of the degree to which the output ports are left idle. Instead the goal is to insure that each flow gets fair opportunity to baseline. Thus low bandwidth flows are given priority as they are expected to have fewer potential baseline event opportunities. The Bandwidth Weighted Fit policy will work well as high rate flows have a higher frame density of potential frames to be used for baselining than do lower rate flows. The baseline deadline duration is motivated by the increasing error accumulated from relative clock drift between asynchronous devices under FlexTDMA. The drift induced error accumulates at the same rate for low and high bandwidth flows as the result of passing time. However, high bandwidth flows have more frames to be used for potential baselining events than do low bandwidth flows. When a high rate flow is preempted, the next frame arriving on that flow can be considered for baselining. Thus giving preference to low bandwidth flows in Gang Scheduling baselining helps insure that all flows have a similar opportunity to maintain a baselined flow status with low clock drift induced error. This Gang scheduling policy is tested using preemption types Strict Gang and Per Output Port. Figure 5.12 shows the Bandwidth Weighted gang scheduling algorithm. Flow 1 is scheduled at output port 1. The preempting flow 2 arrives and has a higher priority than flow 1 as the bandwidth allocated to flow 1 is more than the bandwidth of flow 2. Flow 2 preempts flow 1, and flow 1 will be baselined at a future arriving frame.

5.5.3 Gang Scheduling Bin Fitting Preemption Priority Rules

Preemption of an existing flow scheduled in the flow$_{01}$ queue occurs based on the relative priority of the currently scheduled flow and the preempting flow. Table 5.1 describes the rules for relative priority between flows for determining preemption.

Preemption is inhibited for each individual forwarded output port based on the rules of Table 5.1. When using Best Fit Gang scheduling, preemption is inhibited for each forwarded output port for which a collision has occurred, when the total number of idle ports would increase following preemption. When using FCFS with Backfill Gang scheduling, preemption

Table 5.1 Gang Scheduling Relative Priorities For Preemption

Gang Scheduling	**Current Flow Baseline Status**	**Colliding Flow Baseline Status**	**Priority(Current Flow) <= Priority(Colliding Flow)**
First Fit	NA	NA	NA – preemption not allowed
Concurrent Gang	Non-Baselined	Non-Baselined	True since the flows have equal priority since they are non-baselined
Concurrent Gang	Non-Baselined	Baselined	False as the current flow has higher priority as it is not yet baselined
Concurrent Gang	Baselined	Non-Baselined	True as the colliding flow has higher priority as it is not yet baselined
Concurrent Gang	Baselined	Baselined	True when baselined deadline(current flow) >= baselined deadline(colliding flow)
Lazy Gang	NA	NA	True when number of frames received with baselined deadline exceeded for current flow is less than or equal to colliding flow
Best Fit	NA	NA	NA All forwarding ports have higher priority when fewer ports are idle following preemptions
FCFS w Backfill	NA	NA	NA All forwarding ports have higher priority at a probability
Bandwidth Weighted Fit	NA	NA	True when bandwidth allocated to current flow is greater than or equal to colliding flow

is inhibited for each forwarded output port for which a collision has occurred, at a probability given the number of idle ports. Preemption is inhibited for all forwarded output ports when the Gang scheduling preemption type is strict and any forwarded output ports are preemption inhibited. When preemption is inhibited for an output port, the frame forwarded to that port must use the FIFO queue rather than the flow$_{01}$ queue.

5.6 Switch Delay Bound Computation Under FlexTDMA++

Under FlexTDMA++ the delay bound of each flow at each output port must be carefully selected so that the destination to each multicast leaf of a flow is equal. In this section we review the approach used to compute delay bounds in each FlexTDMA++ switch to support simultaneous multicast.

The delay bound required values for FlexTDMA switches forming the multicast tree are chosen so that the path lengths in delay bound are equal for all destination leaf nodes.

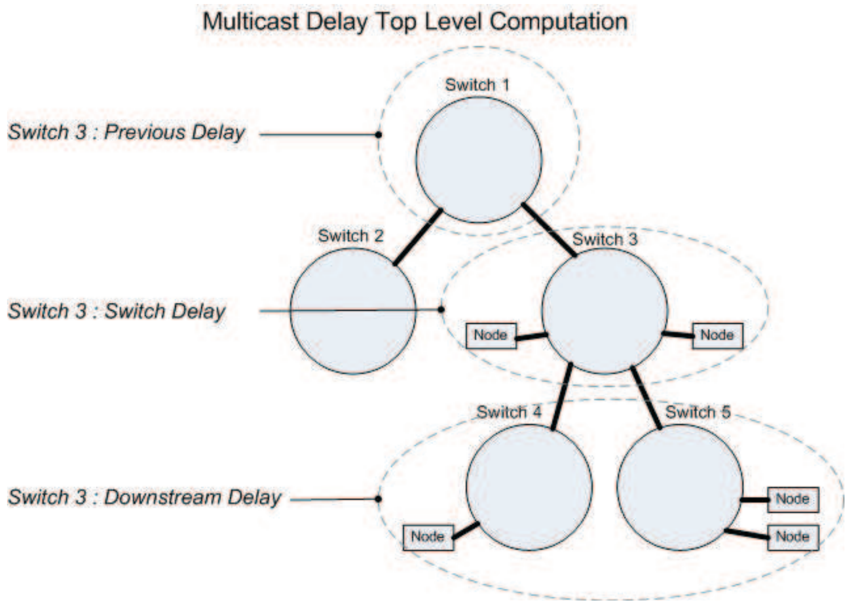
Multicast Delay Top Level Computation
Switch 1
Switch 3 : Previous Delay
Switch 2
Switch 3
Switch 3 : Switch Delay
Node
Node
Switch 4
Switch 5
Switch 3 : Downstream Delay
Node
Node
Node

Figure 5.13 Switch subtree depth computed as previous delay, current switch delay and downstream switch delay.

Figure 5.13 shows the computation breakdown of multicast delay into *previous delay*, *current switch delay* and *downstream delay.* The previous delay is the delay bound from the source transmitting node to the current switch. The current switch delay is the delay time for each forwarded frame at each output port. The downstream delay is the total delay time from transmission from the current switch until the frame arrives to the destination node. The assigned delay at each switch is set so that the total delay to each destination nodes matches. When this is done by each switch in the network the resulting delay depth of each destination node of a multicast set are matched.

5.6.1 Delay Bound at Port

5.6.1.1 Computed Per Port Delay

A delay bound d_i is computed for each output port based on the flows configured to be forwarded to that output port. The computed delay bound $d_i = 0$ when output port i is connected to: 1) no device or 2) a failed switch, otherwise d_i is the computed delay bound based on schedulability analysis of flows forwarded to the output port.

5.6.1.2 Sub Tree Depth

The value $subtree_{k,f}$ is the maximal delay depth of switch k for flow f. Here 'delay-depth' is defined as the frame eligibility time at switch k to the maximal delay of the final leaf node arrival of the frame for any of the output ports of switch k for which flow f is multicast forwarded. The sub-tree delay-depth is formally described as

$$subtree_{k,f} = max\left(d_i + subtree_{switch(k,port=i),f}\right) \tag{5.1}$$

The $subtree_{k,f} = 0$ when output port i is connected to: 1) a leaf node, 2) a failed switch, or 3) no device. Figure 5.14 shows an example of the computation of subtree depth and assigned delay bound for a flow in switch 3. The delay bounds for output port 1, 2, 3 and 4 are 1, 2, 3 and 4 ms, respectively. These delay bounds are computed using schedulability analysis based on the forwarded flow set to the each output port. The subTree delay depth of switch 3 is determined as the maximum delay from the eligibility time of the flow to the arrival time at the destination node. The subTree depth of switch 3 is computed as the maximum of the delay depth for each output port as $\text{subTree}_3 = 10 = \max(1, 7, 10, 4) = \max(\text{d}_1, \text{d}_2 + \text{subTree}_4, \text{d}_3 + \text{subTree}_5, \text{d}_4)$.

5.6.1.3 Assigned Per Port Delay

The assigned per port delay is set to *the subtree depth of the switch - subTree depth of connected switch* at that output port (if any). In this way the delay bound from this switch to each destination node will be the same for all forwarded paths of the flow. When the connected

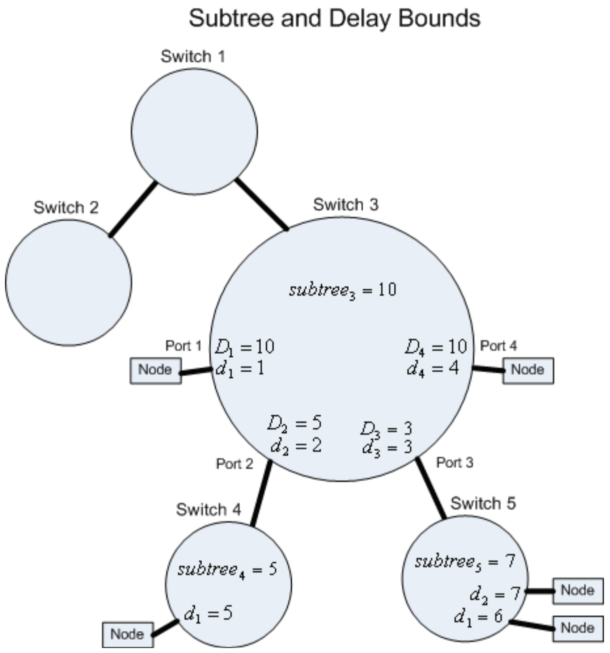

Figure 5.14 Delay bounds are computed to insure equal subtree depth per port.

device is a node, the delay bound to be assigned is simply the subtree depth of the current switch.

When supporting simultaneous multicast a switch will establish a modified maximal delay $D_{k,port,f}$ for output port 'port' in order to insure that the delay-depth of each sub-tree is the same. $D_{k,port,f}$ is the delay bound to be imposed at port number 'port' for flow f. $D_{k,port,f}$ may be larger than d_{port} so that the delay depth for each output port of the switch is the same.

The delay $D_{k,port,f}$ will be computed as: $D_{k,port,f} = subtree_{k,f} - subtree_{switch(k,port),f}$ for output port number 'port' of switch k and flow f, where $switch\,(k, port)$ is the number of the switch connected to switch k port 'port', and $subtree_{switch(k,port),f}$ the delay depth of the switch connected to port number 'port'. Notice the delay value $D_{k,port,f}$ cannot be less than d_{port} since port i is included as one of the ports in the determination of $D_{k,port,f}$ for flow f.

The assigned delay bound for each port is established so that the time from eligibility time to arrival at the destination node is equal. The assigned delay bounds shown in Figure 5.14 are set to the subTree$_3$ - subTree$_{downstream\,switch}$. The assigned delay bounds for switch 3 are computed as follows:

- $D_1 = 10 = 10 - 0 = \text{subTree}_3 - 0$,
- $D_2 = 5 = 10 - 5 = \text{subTree}_3 - \text{subTree}_4$,
- $D_3 = 3 = 10 - 7 = \text{subTree}_3 - \text{subTree}_5$,
- $D_4 = 10 = 10 - 0 = \text{subTree}_3 - 0$.

Notice for all ports of switch 3 $D_i + \text{subTree}_{downstream\,switch} = 10$ ms. Thus the delay bound from eligibility time in switch 3 to arrival at each destination node is 10 ms.

5.6.1.4 Clock Compensated

The clock compensated delay bound, used for determination of frame deadlines at frame eligibility, for each port is set to the assigned delay bound modified by the maximal ppm clock adjustment value of the switch.

5.6.2 Simultaneous Multicast in the Presence of Switch Failure

Switch delay bounds are re-computed with each switch failure and switch restoration. When a switch fails, the network topology may be changed decreasing $D_{k,port,f}$ for some switch output ports. When $D_{k,port,f}$ decreases in switch k for flow f, the VLs traversing output port 'port' must be re-baselined in switch k and in all downstream switches below switch k. Downstream switch re-baselining is needed since the baseline time will be moved back in time since the delay bound is reduced. Without a forced re-bounding event, future baseline frames transmitted will be perceived as arriving before the current deadline, and thus not a re-baselining event.

When a switch is activated, the network topology may be changed increasing $D_{k,port,f}$ for some switch output ports. The resulting delay depth of any sub-tree including this switch may increase.

FlexTDMA++ simultaneous multicast delay bound computation requires re-baselining when the subtree depth is updated. When the simultaneous multicast delay bound is computed for an output port the flow must be re-baselined at that output port in the current switch to reflect the new computed delay bound.

When the simultaneous multicast delay bound for an output port is computed and is less than the existing computed delay bound, a re-baseline command is issued to the sub-tree of that switch output port. When the simultaneous multicast delay bound is increased, a re-baselining cascading event will occur in each switch of the subtree connected to the output port. This follows as the established arrival time of the baselining frame will be greater than the current baseline time.

Baseline updates are required to downstream switches when these switches were disconnected from the tree due to the switch failure. Baseline updates may be required to sibling switches when the switch resumption causes the sub-tree delay depth to increase for other non-failed switches. This occurs when the delay depth of the failed sub-tree is larger than the delay depth of the sibling sub-trees. Figure 5.15 shows an example of the computation of subtree

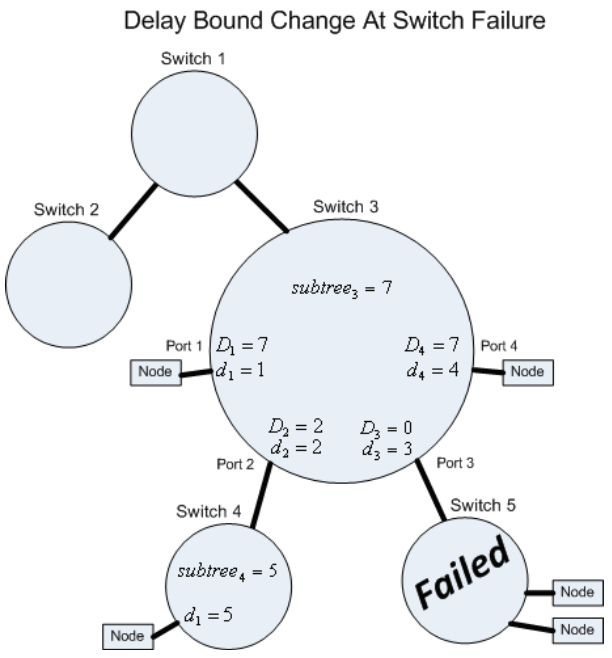

Figure 5.15 Delay bounds are recomputed at switch failure and resumption.

depth and assigned delay bound for a flow in switch 3 after the failure of switch 5. When a switch fails the subtree depth of the switch is by definition zero. The subTree depth of switch 3 is computed as the maximum of the delay depth for each output port as subTree$_3$ = 7 = max(1, 7, zero, 4) = max(d$_1$, d$_2$ + subTree$_4$, zero, d$_4$). The assigned delay bounds for switch

3 are computed as follows:

- $D_1 = 7 = 7 - 0 = subTree_3 - 0$,
- $D_2 = 2 = 7 - 5 = subTree_3 - subTree_4$,
- $D_3 = 0$ by definition,
- $D_4 = 7 = 7 - 0 = subTree_3 - 0$.

5.7 FlexTDMA++ Switch Operation

In this section we review the operational details of the FlexTDMA++ protocol.

5.7.1 Switch Behavior at Frame Arrival

The switch behavior at frame arrival is the same as FlexTDMA+.

5.7.2 Switch Behavior at Frame Eligibility

Once a frame reaches its eligibility time it is processed for each forwarding output port. There are four steps to frame processing. Step 1 - a frame is created for each output port the flow is forwarded. This is a duplicate of the received frame. Step 2 - $Flow_{01}$ availability is determined for each output port at the deadline time for each output port. Step 3 - Gang scheduling preemption rules are applied. Step 4 - frames are scheduled in a frame queue at each output port pending transmission.

5.7.2.1 Eligibility Step 1 - Determine Frame Forwarding Deadlines

The switch eligibility step 1 behavior is the same as FlexTDMA+.

5.7.2.2 Eligibility Step 2 - Determine $Flow_{01}$ Availability and Preemption Potential (if needed)

The switch eligibility step 2 behavior is the same as FlexTDMA+.

5.7.2.3 Eligibility Step 3 - Apply Gang scheduling Preemption Rules

The colliding flow for each forwarding output port, if any, is determined. Using the Gang scheduling preemption priority rules individual forwarding output ports are marked as preemption inhibited when the current flow priority is less than or equal to the colliding flow. When using either Best Fit or FCFS w/ Backfill Gang scheduling all forwarding output ports are marked as preemption inhibited if any are. When operating in Gang scheduling preemption type strict and any forwarding output ports are marked as preemption inhibited, all forwarding output ports are marked as preemption inhibited.

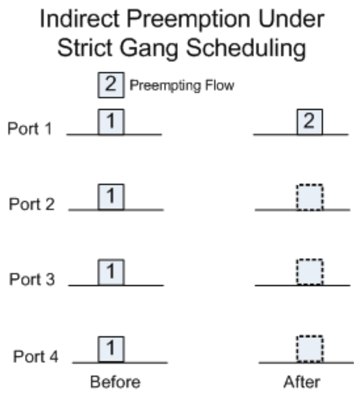

Figure 5.16 Pri(2) > Pri(1) so flow 2 preempts flow 1 at port 1, and due to strict preemption flow 1 is also preempted at ports 2, 3 and 4.

When some forwarding output ports are preemption enabled, while using strict preemption, any indirect colliding flows must also be preempted. Any frames from a preempted flow f scheduled for baselining on a port not forwarded by the current flow must be preempted when that same flow f frame is preempted on another forwarding output port. Figure 5.16 shows the concept of *indirect preemption* under strict Gang Scheduling. Flow 1 scheduled for baselining at output ports 1, 2, 3 and 4. A frame arrives on flow 2 having a higher priority than flow 1, and therefore preempts flow 1. Although flow 2 is not forwarded to output ports 2, 3 and 4 flow 1 must be preempted as the Gang Scheduling preemption policy is strict. These output ports 2, 3 and 4 are indirect to the preemption occurring on port 1.

5.7.2.4 Eligibility Step 4 FlexTDMA++ Queuing Decision Logic

Table 5.2 shows the queuing decision logic for all 3 baselined states of the FlexTDMA++ Protocol. Using this table each current eligible frame will be scheduled in either the FIFO queue or the flow$_{01}$ queue.

Table 5.2 FlexTDMA++ Queuing Decision Logic

	Flow$_{01}$ Availability at Deadline	**Preemption Eligible**	**Queue**
1	**Yes**	NA	Flow$_{01}$ at deadline
2	No	No	FIFO
3	No	**Yes**	Preempt Flow$_{01}$ at deadline

Table 5.2 row 1 shows that when flow$_{01}$ has a transmission opportunity at the flow deadline the frame will be scheduled in flow$_{01}$ at the deadline time. Table 5.2 row 2 shows that when the scheduled frame in flow$_{01}$ is not preemption eligibility, the frame is queued in the FIFO queue. Table 5.2 row 3 shows that when the current frame is eligible for preemption of the scheduled frame in flow$_{01}$, a preemption will be performed. Under the FlexTDMA++ protocol the preemption eligibility depends on the relative baseline state of flows, the relative time to baseline, and the preemption class either 'strict' or 'per port'. This simplifies this state machine by encapsulating much of the logic in the preemption status of the flow.

5.8 FlexTDMA++ Evaluation

In this section we provide the evaluation approach of the FlexTDMA++ protocol.

5.8.1 Phases of Testing

There were three phases of testing.

5.8.1.1 FlexTDMA++: Periodic On-Off Probability

The testing phase FlexTDMA++ Periodic On-Off Probability focusses on the influence of the probability of periodic on-off on the performance of FlexTDMA++.

5.8.1.2 FlexTDMA++: Frame Loss probability

The testing phase FlexTDMA++ Frame Loss probability focusses on the influence of the probability of frame loss on the performance of FlexTDMA++.

5.8.1.3 FlexTDMA++: Switch Failure Profiles

The testing phase FlexTDMA++ Switch Failure Profiles focusses on the influence a switch failure and resumption has on the performance of FlexTDMA++.

5.8.2 Protocol Parameters of Testing

5.8.2.1 Simultaneous Multicast Gang Scheduling Types Tested

Table 5.3 shows the preemption types tested for each Gang Scheduling policy. Both strict gang preemption and per output port gang preemption were tested as appropriate for the Gang scheduling policy. The First Fit Gang scheduling policy does not allow preemption. The FCFS w Backfill Gang scheduling policy applies preemption enablement to all forwarded ports. There are a total of 13 combinations of Gang scheduling policy and preemption type tested. FCFS w Backfill Gang Scheduling policy was tested with three cases of probability values.

Table 5.3 Preemption Types Tested For Each Gang Scheduling Policy

Gang Scheduling	**No Preemption**	**Strict Gang Preemption**	**Per Output Port Preemption**
none	Yes		
First Fit	Yes		
Concurrent Gang		Yes	Yes
Lazy Gang		Yes	Yes
Best Fit		Yes	Yes
FCFS w Backfill	Yes		
Bandwidth Weighted Fit		Yes	Yes

Table 5.4 FlexTDMA++ Parameters Tested

FlexTDMA++ Phase	FlexTDMA++ Phase: Periodic On-Off	FlexTDMA++ Phase: Frame Loss	FlexTDMA++ Phase: Switch Failure
Periodic On-Off	4 Probabilities	-	-
Frame Loss	-	4 Probabilities	-
Switch Failure	-	-	4 Probabilities
Gang Scheduling	13 (Gang and preemption)	13 (Gang and preemption)	13 (Gang and preemption)
Clock Drift Modes	4 (none, increasing, decreasing, mixed)	4 (none, increasing, decreasing, mixed)	4 (none, increasing, decreasing, mixed)
Bandwidth Load	3 (20, 50, 90%)	3 (20, 50, 90%)	3 (20, 50, 90%)
Total combinations	624	624	624

5.8.3 Test Runs Performed

The FlexTDMA++ protocol was evaluated by configuring the network operational parameters (i.e. probability of frame loss) and the FlexTDMA++ parameters (i.e. Gang Scheduling policy) and performing a run using the configuration. Data is collected for each key performance criteria during each run. Table 5.4 shows the parameters configured for each test run along with the values the parameter was configured. There are 624 total combinations of the parameter values shown. A test run was performed for each combination for a total of 624 runs for each of the three test phases. There were 4 probability values selected for periodic on-off, frame loss and switch failure. These were selected to induce low, medium and high levels of flow discontinuations and resumptions. The Gang Scheduling policy and preemption policy was selected for each run. The evaluation included four clock drift configurations: 1) no clock drift in nodes or switches, 2) increasing clock drift along the forwarding path of flows in the network, 3) decreasing clock drift along the forwarding path of flows in the network and 4) mixed clock drift in the flow propagation path. Several bandwidth loads were tested to determine the sensitivity of FlexTDMA++ to bandwidth variations.

5.8.4 Key Performance Criteria

The key performance criteria of the FlexTDMA++ protocol are: time-to-baseline, laxity and simultaneous multicast.

The simultaneous multicast criterion is defined as maximal difference in delivery times of multicast instances of a source frame to each leaf node. For example when frame 1 is multicast to three destinations having delivery times of 99 ms, 100 ms, and 101 ms the maximal difference in delivery times is 2 ms. When a flow is baselined the FlexTDMA++ switch is able to nearly reconstruct the traffic envelope of the arriving flow, which minimizes the simultaneous multicast maximal difference in delivery time. This approach allows comparison between flows having different delay bounds.

5.8.5 Testing Topology Details

The topology in Figure 5.17 shows the physical topology used to demonstrate the FlexTDMA++ simultaneous multicasting performance. The topology is designed with different depth levels available for each flow destination leaf node. Each flow was configured to be forwarded to a sub-set of the leaf nodes, but all destination leaf nodes of each flow will have the same end-to-end total delay bound. Thus, all multicast instances of each source frame should arrive to their respective destination at the same time.

The transmissions from end nodes 0 to 9 are forwarded to receiving leaf nodes of the tree. This insures that the flows entering the tree structure of switches (switches 6 to 17) are sourced from many transmitting nodes that operate independently and that flow contention exists prior to tree entry.

The bandwidth is divided among the flows allocated to each node so that the bandwidth allocation varies. The variation in bandwidth allocation causes the periodic transmissions to not maintain a fixed relative offset, but instead to phase so that collisions occur.

The maximal difference in arrival time will be tracked for each individual source frame.

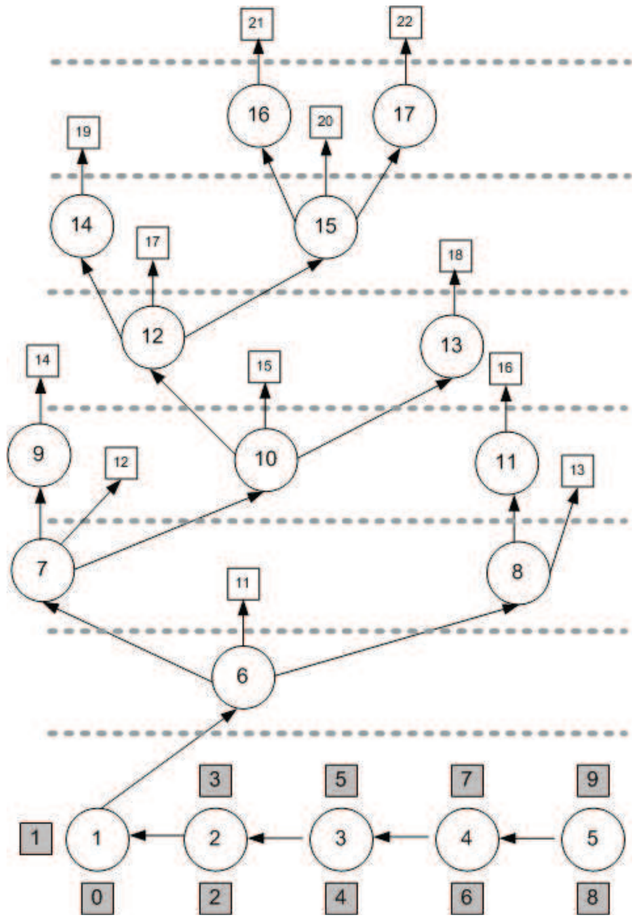

Figure 5.17 FlexTDMA++ Testing Topology

The maximal difference is the difference between the first multicast destination arrival and the final multicast destination arrival.

The delay bounds for the flows of this testing topology range from 1,587 μs (1.587 ms) to 2,493 μs (2.493 ms). The magnitude of these delay bounds is important to keep in mind relative to the performance criteria evaluation that follows.

5.8.6 Results

5.8.6.1 Clock Drift Effect on FlexTDMA++

Table 5.5 shows a summary of the effect of clock drift on the FlexTDMA++ protocol for each key performance criteria and for each phase of evaluation. Time-To-Baseline had the highest impact from increasing drift. An increasing drift accelerates the number of baselines needed.

All Time-To-Baseline differences resulting from different drift modes are small compared to differences generated from other run parameters. Frame delay bound laxity performances were no more than 4% of the flow delay bound. The simultaneous multicast performance varies between clock drift types. Increasing drift consistently has the highest impact as it

Table 5.5 Clock Drift Effect on FlexTDMA++

FlexTDMA++ Phase	FlexTDMA++ Phase: Periodic On-Off	FlexTDMA++ Phase: Frame Loss	FlexTDMA++ Phase: Switch Failure
Time To Baseline	1.15 to 14.2 ms	1.27 to 14.9 ms	1.15 to 13.2 ms
Time To Baseline (Variation between drift types)	0.5 to 13%	0.13 to 12%	0.5 to 16.4%
Laxity	0 to 55.3 μs	4.4 to 36.9 μs	0 to 53.5 μs.
SM (percent difference)	3.6% (0.2 of 6.9 us) to 33% (4.7 of 14.5 us)	1.1% (0.91 of 81.1 us) to 22.8% (11.5 of 57.7 us)	5.5% (2.5 of 48.7 us) to 41.4% (50.3 of 88.4 us)

accelerates the number of baselines needed. FlexTDMA++ managed clock drift efficiently for all modes of operation for each key performance criteria, and clock-drift had a minimal but consistent result on performance. The performance difference made by clock drift on the key performance criteria is small relative to the other tested parameters. Further evaluation will be limited to increasing drift rate. This reduces the total number of run values to be compared. Total runs per phase when using a single clock drift rate is 156 = 13 (Gang scheduling and gang preemption combinations) X 1 (single drift rate value) X 3 (bandwidth loads) X 4 (frame loss or periodic on-off or switch failure).

5.8.6.2 Bandwidth Load on FlexTDMA++

The total bandwidth loading of the network is determined for each test. Table 5.6 shows the average performance relating to the three bandwidth loads as 20%, 50%, and 90%. The average time-to-baseline was consistently influenced by bandwidth load. As the bandwidth load was increased the time needed to achieve a baselined state on a non-baselined flow increased. In phase Periodic on-off the average delay bound laxity and simultaneous multicast increased with increasing bandwidth. This follows as multiple flows are discontinued each time a transmitting node pauses transmission, and all flows from that node must be re-baselined. In phase Frame Loss and phase Switch Failure the average delay bound laxity and simultaneous multicast decreased with increasing bandwidth. More frames are transmitted within the time needed between baseline attempts (minimal baseline interval) as the bandwidth increases. This increases the utilization frame quantity once the flow achieves a baselined state.

Further evaluation will use maximum bandwidth loading of 50% and 90%. Both 50% and 90% bandwidth loads are considered as the best fit and bandwidth weighted gang scheduling policies were not stable at heavy bandwidth loads (70% to 90%). Total runs per phase for each bandwidth load is 156 = 13 (Gang scheduling and gang preemption combinations) X 1 (single drift rate value) X 1 (single bandwidth load) X 4 (frame loss or Periodic On-Off or switch failure).

Table 5.6 Impact of Bandwidth Load on FlexTDMA++

FlexTDMA++ Phase	FlexTDMA++ Phase: Periodic On-Off	FlexTDMA++ Phase: Frame Loss	FlexTDMA++ Phase: Switch Failure
Time To Baseline (average)	1.28, 4.23 and 10.5 ms	1.47, 4.83 and 11.2 ms	1.28, 4.27 and 9.87 ms
Laxity (average)	1.22, 4.05 and 5.93 μs	22.6, 15.7 and 12.4 μs	29.2, 13.5 and 4.44 μs
SM (average)	8.83, 11.0, and 12.8 μs	59.9, 35.3 and 34.0 μs	91.8, 38.2 and 38.4 μs

5.8.6.3 Performance Under Heavy Bandwidth Load on FlexTDMA++

The Gang Scheduling algorithms best fit and bandwidth weighted were tested at loads of 10% to 90%. These policies were not stable at bandwidth loads of 70% to 90% as sufficient numbers of baselining transmission opportunities went unused so that the utilized rate was

Table 5.7 Comparison of Best Fit and BW Weighted Gang Scheduling

Performance Criteria	Parameter	Load Favoring Best Fit	Load Favoring BW Weighting	Preemption
Time-to-Baseline	Frame Loss	50%-60% 3.9 - 4.1 ms	10%-40% 1.3 - 3.7 ms	No difference
	Periodic On-Off	50%-60% 3.5 - 3.6 ms	10%-40% 1.1 - 2.8 ms	No Difference
	Switch Failure	50%-60% 3.4 - 3.6 ms	10%-40% 1.0 - 2.6 ms	No Difference
Laxity	Frame Loss	30%-60% 7 - 22 μs	10%-20% 26 - 28 μs	Per Port better when BW Weighted used
	Periodic On-Off	40%-60% 1.4 - 2.0 μs	Same as Best Fit 10%-30% 0.7 - 3.0 μs	Per Port better when BW Weighted used
	Switch Failure	10%-60% 1.0 - 24 μs	none	No Difference
Simultaneous Multicast	Frame Loss	20%-60% 45 - 66 μs	Same as Best Fit 10% 90 μs	No Difference
	Periodic On-Off	10%-60% 10 - 52 μs	None	No Difference
	Switch Failure	10%-60% 30 - 95 μs	None	No Difference

less than the rate needed to maintain the flows in a baselined state. Table 5.7 shows the bandwidth loads, 10% to 60%, favoring the usage of the best fit or bandwidth weighted gang scheduled baselining policies for each critical performance criteria. In each case the performance is characterized. The time-to-baseline performance criterion favors the best fit policy under bandwidth loads of 50% to 60%, with bandwidth weighted policy favored for loads of 10% to 40%. As the bandwidth loading is reduced the baseline density is reduced. This reduces the importance of best fit, and amplifies the importance of relative bandwidth utilization on each flow. The laxity performance criteria had mixed results depending on the parameter under test. Under frame loss and periodic on-off conditions best fit policy was favored for higher loads with bandwidth weighted policy favored for lower loads using a per port preemption policy. Under switch failure conditions best fit policy was favored for all bandwidth loads. The simultaneous multicast performance criterion favors the best fit policy under all bandwidth loads. The bandwidth weighted policy had nearly the same performance at 10% bandwidth loading.

The conclusion of this comparison of gang scheduling best fit and bandwidth weighted policies is that when the bandwidth load is heavy, 50% to 60%, the best fit policy should be used. When the bandwidth load is 10% to 40% the best fit policy should be used when simultaneous multicast performance is critical and bandwidth weighted policy when either time-to-baseline performance or laxity critical.

5.8.6.4 Effect of Probablity of Periodic On-Off, Frame Loss and Switch Failure on FlexTDMA++

Table 5.8 shows the increase acheived by modifying the probablity of periodic on-off, frame loss and switch failure. Periodic on-off probability, frame loss probability and switch failure testing showed little effect on time-to-baseline. This follows as time-to-baseline is determined by how quickly the flow can be baselined.

The periodic on-off probability, frame loss probability and switch failure probability have a consistent effect of increasing frame delay bound laxity and simultaneous multicast. As these probabilities increase the frequency of interruption of the baselined state of each flow increases forcing the flow to spend proportionally less time in a stable baselined state. Further evaluation uses maximum probability values. Total runs per phase assuming maximum probability values are used is 13 = 13 (Gang scheduling and gang preemption combinations) X 1 (single drift rate value) X 1 (single bandwidth load) X 1 (frame loss or Periodic On-Off or switch failure).

Table 5.8 Impact Of Probability of Flow Interruption on FlexTDMA++ Performance

FlexTDMA++ Phase	FlexTDMA++ Phase: Periodic On-Off	FlexTDMA++ Phase: Frame Loss	FlexTDMA++ Phase: Switch Failure
Time To Baseline	0.4%	0.2%	1.7%
Laxity	30.1%	44.0%	21.3%
SM	41%	50.3%	4.4%

5.8.6.5 Improvements to FlexTDMA++

We consider the comparative performances of the gang scheduling algorithms when the bandwidth loading is 50% and 90%. Table 5.9 shows the favored gang scheduling policies for each performance criteria. All 13 gang scheduling policies and preemption strategy pairs are evaluated. At a 90% bandwidth load the performance results for the three key performance criterion, time-to-baseline, laxity and simultaneous multicast, were similar for all gang scheduling policies. The gang scheduling policies resulting in better performance than no coordinated baselining is listed under a 50% bandwidth load. The time-to-baseline performance criterion favors the best fit policy using strict preemption, the best fit policy using per port preemption, bandwidth weighted using per port preemption, and finally bandwidth weighted using strict preemption. The laxity performance criterion favors the best fit policy using per port preemption, best fit policy using strict preemption, bandwidth weighted using per port preemption, and finally bandwidth weighted using strict preemption. The simultaneous multicast performance criterion favors best fit policy using per port preemption, concurrent gang policy using per port preemption, lazy gang using strict preemption, and finally bandwidth weighted using strict preemption. Other gang scheduling policies did not improve performance when compared to no baselining coordination.

We conclude that the best approach is to increase the allocation to the baselining flow $flow_{01}$ so that the effective load on baseline scheduling is 50%. When this is done the gang scheduling policy best fit using per port preemption will offer the best performance to FlexTDMA++ considering time-to-baseline, laxity and simultaneous multicast criteria. When time-to-baselining

Table 5.9 Gang Scheduling Improvement Performances

Bandwidth Load	Performance Criteria	Gang Scheduling (Preemption)	Performance
90%	Time-to-Baseline	No Clear Difference	10.5 - 14.0 ms
90%	Laxity	No Clear Difference	3 - 25 μs
90%	Simultaneous Multicast	No Clear Difference	20 - 67 μs
50%	Time-to-Baseline	Best Fit (Strict)	3.1 - 3.7 ms
50%	Time-to-Baseline	Best Fit (Per Port)	3.5 - 4.0 ms
50%	Time-to-Baseline	BW Weighted (Per Port)	3.9 - 4.8 ms
50%	Time-to-Baseline	BW Weighted (Strict)	4.0 - 4.8 ms
50%	Laxity	Best Fit (Per Port)	1 - 7 μs
50%	Laxity	Best Fit (Strict)	1 - 10 μs
50%	Laxity	BW Weighted (Per Port)	6 - 18 μs
50%	Laxity	BW Weighted (Strict)	5 - 23 μs
50%	Simultaneous Multicast	Best Fit (Per Port)	10 - 42 μs
50%	Simultaneous Multicast	Concurrent Gang (Per Port)	15 - 50 μs
50%	Simultaneous Multicast	Lazy Gang (Strict)	17 - 56 μs
50%	Simultaneous Multicast	BW Weighted (Strict)	17 - 62 μs

performance criterion is most important the strict preemption policy should be used.

5.9 Chapter Summary

In this chapter we introduced an enhancement to the FlexTDMA protocol to support simultaneous multicast. The details needed to support simultaneous multicast were characterized. An evaluation of several approaches to concurent baseline scheduling and preemption policies supporting simultaneous multicast within FlexTDMA++ was completed. This evaluation demonstated the ability of FlexTDMA++ to support simultaneous multicast.

The evaluation showed the performances of FlexTDMA++ support of simultaneous multicast for the key performance criteria. Comparing the performances achieved to the delay bounds on the flows being supported relates the performances to the frame transmission time rather than the bit-per-second line rate. The time-to-baseline performance was typically 2 times the flow delay bound when 50% bandwidth loaded and 4 times when 90% loaded. This indicates the time needed to wait for a baselined state is a small multiple of the delay bound on the flow. The laxity performance was typically about 2% of the flow delay bound, indicating the delay bounds were nearly maximal. The simultaneous multicast performance was typically 10% of the flow delay bound.

We demonstrated the ability of FlexTDMA++ to manage clock drift. We determined the effect flow transmission interruption has on the FlexTDMA++ performance. We performed full evaluation of all gang scheduling policies and preemption policies at 50% and 90% bandwidth loading. We demonstrated that the best gang scheduling policy under 50% loading is best fit using per port preemption.

CHAPTER 6. Summary and Future Work

6.1 Summay

In this dissertation we addressed the problem of offering constant maximal end-to-end delays, nearly at the delay bound of each flow, in an asynchronous switching network. The focus was on closed industrial networking supporting periodic traffic on allocated flows.

A major contribution of this dissertation is the introduction of the scheduling algorithm we call FlexTDMA. FlexTDMA allows delivery of frame data on each allocated flow nearly at the flow maximal delay bound with minimal delay-jitter.

Another main contribution of this dissertation is the definition and evaluation of the FlexTDMA+ protocol. We evaluated FlexTDMA+ under real-world conditions of end node periodic on-off transmission, and network conditions of clock drift, frame loss and network bandwidth load. We evaluated the relative benefit offered by three improvements to FlexTDMA 1) baseline preemption, 2) partial baselining and 3) baseline deadline density control.

A final contribution of this work is the definition and evaluation of the enhancement to FlexTDMA supporting simultaneous multicast called FlexTDMA++. We evaluated the simultaneous multicast performance under real-world conditions of switch failures, end node periodic on-off transmission, and network conditions of clock drift, frame loss and network bandwidth load.

6.2 Furture Work

We plan to extend the contributions of this dissertation in a number of directions. These are detailed below.

6.2.1 Clock Drift Estimation

The current definition of FlexTDMA could be enhanced by incorporating relative clock drift estimation and compensation. Virtual arrival time of frames is used when computing arrival time relative to eligibility time. This would allow the system to be more stable in the presence of significant clock drift. This would mean that baseling deadlines could be extended, reducing the demand density for baselining, and lesson the impact of clock drift between baselines.

6.2.2 Stable Delays for Less Than Maximal Delay Bounds

The FlexTDMA protocol could be modified to insure stable delay bounds for lower than maximal delay bounds. Currently the FlexTDMA protocol insures maximal delay bounds. Real world traffic is usually much less than maximally allocated bandwidth. Furthermore, actual frame collisions are less than maximal. The constant delay bound might be lowered to more than the recently realized actual delays experienced. This change would allow FlexTDMA to offer more real world delays while offering stable delays.

6.2.3 Alternate Transmission Opportunity Approaches

In the existing definition of FlexTDMA transmission opportunities are allocated as 1 frame per period. Transmission opportunity availability at eligibility time must be determined as a baselining frame is transmitted at the deadline. Therefore a commitment must be made and a transmission opportunity reserved at the eligibility time of the frame. Choosing to use a frame for baselining at transmit time is not possible, as the frame will almost always be transmitted prior to its deadline. Schedulability of the flow set must be determined. This was shown in this paper. Any alternate approach will need to resolve this issue. An alternate approach to provisioning baselining transmission opportunities is the use of a token bucket based service curve. This might lesson the time to baselining as bursts of unbaselined flows can be better managed.

6.2.4 Discard Nonbaselined Frames

A possible modification to FlexTDMA is to discard frames that can not be transmitted in a baselined state. Rather than using a single Flow01, two would be used: 1) Flow01 Highest Priority, and 2) Flow01 Lowest Priority. When a frame is scheduled as a baselined frame, it is inserted into the Flow01 Highest Priority if available, otherwise in the Flow01 Lowest Priority. Frames scheduled in Flow01 Lowest Priority are serviced at their scheduled time if no other traffic is pending, otherwise discarded. Thus insertion into Flow01 Highest Priority represents a strong commitment to baselining, while Flow01 Lowest Priority represents a weak commitment to baselining. This modification to FlexTDMA has the potential for faster baselining performance as some portion of those frames scheduled for baselining in Flow01 Lowest Priority will be baselined. Additionally, only baselined frames are transmitted, so receivers have assurance of timing of all flows.

6.2.5 Dynamic Routing Applied To FlexTDMA Multicast

The FlexTDMA simultaneous multicast support should be adapted to a dynamically routed network. Much research exists on approaches to optimal multicast routing. The FlexTDMA simultaneous multicast protocol has been validated to offer good performance in a static multicast network under real world conditions. Future research should apply FlexTDMA simultaneous multicast to networks supporting dynamically routed multicast trees.

6.2.6 Compare FlexTDMA to Synchronized Networks Using RCSP-DJ

The FlexTDMA protocol should be compared to the use of RCSP-DJ using existing synchronization methods as the network is subjected to frame loss, transmitting node resets, clock drift and bandwidth loads. When using synchronization mechanisms a synchronized state must be obtained and maintained before the RCSP-DJ performances are provided.

APPENDIX A. Acronyms

Table A.1 Acronyms

Acronym	Meaning
AT	Arrival Time
BD	Baseline Deadline
BI	Baseline Interval
BPR	Burst Performance Ratio
BW	Bandwidth Weighted
CDF	Cumulative Distribution Function
Delay-EDD	Delay-Earliest Due Date
EDF	Earliest Deadline First
ET	Eligibility Time
FCFS	First Come First Serve
FER	Frame Error Rate
FIFO	First In First Out
FSC	Fair Service Curve
GPS	General Processor Sharing
OPNET	Optimized Network Engineering Tools
PBS	Preferred Burst Size
PPM	Parts Per Million
RCSP	Rate Constrained Static Priority
RCSP-DJ	Rate Constrained Static Priority Delay-Jitter
RCSP-RJ	Rate Constrained Static Priority Rate-Jitter
RM	Rate Monotonic
SM	Simultaneous Multicasting
SP	Static Priority
TCP	Transmission Control Protocol
TDM	Time Division Multiplex
WF2Q	Worst-Case Weighted Fair Queuing
WFQ	Weighted Fair Queuing

APPENDIX B. Symbols and Variables

Table B.1 Symbols and Variables

Term	Description
$A_k^*(t)$	Traffic envelope for flow k.
AT_j^k	Arrival time of frame k at switch j.
BD_k	Baseline deadline for flow k.
BI_k	Baseline interval for flow k.
d_j	Delay bound of eligible frames in switch j.
ET_j^k	Eligibility time of frame k at switch j.
$F_{i,j}$	Fairness index of flows i and j.
H_j^k	Hold time of frame k at switch j.
fr_{time}	Transmission time of the frame.
l	Line rate of a communication channel.
ppm	Part-Per-Million.
P_{ij}	Frame size of flow j on priority level i.
$\pi_{j-1,j}$	Media transmission time from switch j-1 to switch j.
$s_k^{\max}$	Maximum sized frame on flow k.
W_j^k	Actual packet scheduling delay of frame k at switch j.
w_k	Weight applied to flow k.
X_{min}	Minimum frame inter-arrival time on the flow.

BIBLIOGRAPHY

[1] J. Lehoczky, L. Sha and Y. Ding, “The rate monotonic scheduling algorithm: exact characterization and average case behavior,” *IEEE Real-Time Systems Symposium,* pp. 166-171, doi:10.1109/REAL, 1989.

[2] D. V. Hui, H. Zhang and D. Ferrari, “Delay Jitter Control for Real-Time Communication in a Packet Switching Network,” *Proceedings of TriComm,* 1991.

[3] H. Zhang and S. Keshav, “Comparison of Rate-Based Service Disciplines,” *ACM, In Proceedings of ACM SIGCOMM,* pp. 113-121, 1991.

[4] R. L. Cruz, “A Calculus for Network Delay, Part I - Network Elements in Isolation,” *IEEE Transactions on Information Theory,* vol. 37, no. 1, January 1991.

[5] R. L. Cruz, “A Calculus for Network Delay, Part II: Network Analysis,” *IEEE TRANSACTIONS ON INFORMATION THEORY,* vol. 37, no. 1, January 1991.

[6] D. G. Feitelson and L. Rudolph, “Gang Scheduling Performance Benefits for Fine-Grained Synchronization,” *Journal of Parallel and Distributed Computing,* vol. 16, no. 4, pp. 306-318, 1992.

[7] H. Zhang and D. Ferrari, “Rate-Controlled Static-Priority Queueing,” *In Proc. IEEE Infocom,* pp. 227-236, 1993.

[8] A. Parekh and R. Gallager, “A generalized processor sharing approach to flow control-The single node case,” *IEEE/ACM Trans. Networking,* vol. I, no. 3, pp. 344-357, 1993.

[9] J. Liebeherr, D. E. Wrege and D. Ferrari, "Exact Admission Control for Networks with Bounded Delay Services," *IEEE/ACM Transactions on Networking,* vol. 4, pp. 885-901, 1994.

[10] H. Zhang and D. Ferrari, "Rate-Controlled Service Disciplines," *Journal of High Speed Networking,* 1994.

[11] J. Liebeherr and D. E. Wrege, "Design and Analysis of a High-Performance Packet Multiplexer for Multiservice Networks with Delay Guarantees," *Technical Report CS-94-30, University of Virginia,* 1994.

[12] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control-The multiple node case," *IEEE/ACM Trans. Networking,* vol. 2, no. 2, pp. 137-150, 1994.

[13] K. G. Shin and Y.-C. Chang, "A Reservation-Based Algorithm for Scheduling Both Periodic and Aperiodic Real-Time Tasks," *IEEE TRANSACTIONS ON COMPUTERS,* vol. 44, no. 12, December 1995.

[14] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proceedings of the IEEE,* vol. 83, pp. 1374-1396, Oct. 1995.

[15] D. Lifka, "The ANL/IBM SP Scheduling System," *In Proceedings of JSSPP,* pp. 295-303, 1995.

[16] D. G. Feitelson and L. Rudolph, "Parallel Job Scheduling: Issues and Approaches," *in Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (eds.),* pp. 1-18, Springer-Verlag, 1995.

[17] J. Liebeherr, D. E. Wrege and D. Ferrari, "Exact admission control for networks with a bounded delay service," *IEEE/ACM Transactions on Networking (TON),* vol. 4, Issue 6, pp. 885-901, 1996.

[18] J. C. R. Bennett and H. Zhang, "Why WFQ Is Not Good Enough For Integrated Services Networks," *Proceedings of NOSSDAV,* 1996.

[19] L. Georgiadis, R. Guerin, V. Peris and K. N. Sivarajan, "Efficient Network QoS Provisioning Based on Per Node Traffic Shaping," *IEEE/ACM Transactions on Networking,* vol. 4, no. 4, pp. 482-501, August 1996.

[20] H. Zhang, "Providing End-to-End Performance Guarantees Using Non-Work-Conserving Disciplines," *Computer Communications: Special Issue on System Support for Multimedia Computing,* vol. 18, 1996.

[21] A. C. Dusseau, R. H. Arpaci and D. E. Culler, "Effective Distributed Scheduling of Parallel Workloads," *Proc. ACM SIGMETRICS,* pp. 25-36, 1996.

[22] J. Rexford, F. Bonomi, A. Greenberg and A. Wong, "Scalable Architectures for Integrated Traffic Shaping and Link Scheduling in High-Speed ATM Switches," *IEEE Journal on Selected Areas in Communications,* vol. 15, pp. 938-950, 1997.

[23] D. Stiliadis and A. Varma, "A General Methodology for Designing Efficient Traffic Scheduling and Shaping Algorithms," *In Proceedings of IEEE INFOCOM,* 1997.

[24] R. L. Cruz, "SCED+: Efficient Management of Quality of Service Guarantees," *Technical Report 97-13, Center for Wireless Communications, UCSD,* July 1997.

[25] F. Wang, "Scheduling in Multiprogrammed Parallel Systems," *Research Report RC 19790 (87657), IBM T.J.Watson Research Center,* 1997.

[26] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik and P. Wong, "Theory and Practice in Parallel Job Scheduling," *JSSPP,* pp. 1-34, 1997.

[27] B. Lee, H. Lim, H. Kang and C. Kim, "A Study on Efficient Delay Distribution Methods for Real-time VBR Traffic Transfer," *Institute of Advanced Engineering, Korea,* 1998.

[28] T.-C. Hou and C.-C. Chen, "Jitter-EDD Implementation for Transporting MPEG-2 Video Streams on ATM Networks," *13th International Conference on Information Networking (ICOIN'98),* p. 155, 1998.

[29] D. G. Feitelson and A. M. Weil, “Utilization and Predictability in Scheduling the IBM SP2 with Backfilling,” *In 12th Intl. Parallel Processing Symp. (IPPS),* pp. 542–546, 1998.

[30] J. Pulido and K. Lin, “SM: Real-Time Multicast Protocols for Simultaneous Message Delivery,” *in Proc. RTCSA,* pp. 66-66, 1998.

[31] A. Fernandez and J. Carlos, “On the Isolation of Several Work-Conserving Scheduling Policies,” *Computer Communications and Networks, 1999. Proceedings. Eight International Conference on,* vol., no., pp. 188-192, 1999.

[32] T.S. Eugene Ng, D. C. Stephens, I. Stoica and H. Zhang, “Supporting Best-Effort Traffic with Fair Service Curve,” *Global Telecommunications Conference, 1999. GLOBECOM ’99,* vol. 3, no., pp. 1799-1807, 1999.

[33] J. Liebeherr and E. Yilmaz, “Workconserving vs. Non-workconserving Packet Scheduling: An Issue Revisited,” *Proc. IEEE/IFIP IWQoS 99,* pp. 248-256, 1999.

[34] F. Silva and I. D. Scherson, “Towards Flexibility and Scalability in Parallel Job Scheduling,” *11th IASTED International Conference on Parallel and Distributed Computing and Systems, Boston, USA,* 1999.

[35] F. Silva and I. D. Scherson, “Improvements in Parallel Job Scheduling Using Gang Service,” *International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN ’99),* p. 268, 1999.

[36] W. Leinberger, G. Karypis and V. Kumar, “Multicapacity Bin Packing Algorithms with Applications to Job Scheduling Under Multiple Constraints,” *in Proc. of the Intl. Conf. on Parallel Processing. IEEE,* pp. 404-412, 1999.

[37] W. Chonggang, L. Keping, G. Xiangyang and C. Shiduan, “Effective fairness queuing algorithms,” *Proceedings. IEEE International Conference on Networks (ICON 2000),* pp. 294-301, Sept. 2000.

[38] K. Qutubudin and D. Pease, "A real-time heterogeneous distributed computing environment for multi-robot system," *Object-Oriented Real-Time Distributed Computing, IEEE International Symposium on, Third IEEE International Symposium,* p. 376, 2000.

[39] Y. Zhang, A. Sivasubramaniam, H. Franke and J. E. Moreira, "Improving Parallel Job Scheduling by Combining Gang Scheduling and Backfilling Techniques," *Parallel and Distributed Processing Symposium, International, 14th International Parallel and Distributed Processing Symposium (IPDPS'00),* p. 133, 2000.

[40] A. Benslimane, "A Multimedia Synchronization Protocol for Multicast Group," *in Proc. EUROMICRO,* pp. 1456-1456, 2000.

[41] J. C. R. Bennett, K. Benson, A. Charny, W. F. Courtney and J.-Y. LeBoudec, "Delay Jitter Bounds and Packet Scale Rate Guarantee for Expedited Forwarding," *In INFOCOM,* pp. 1502-1509, 2001.

[42] J. U. Klcking, C. Maihfer and K. Rothermel, "A Smart Card Based Solution to Minimize Inter-receiver Delay Jitter," *Proceedings of the Tenth International Conference on Computer Communications and Networks (IEEE ICCCN 2001),* 2001.

[43] D. Bouzid, Z. Mammeri and P. Lorenz, "Analysis and experimentation of service disciplines to guarantee real-time communications," *5th IEEE International Conference on High Speed Networks and Multimedia Communications,* pp. 243-247, July 2002.

[44] J. C. R. Bennett, K. Benson, A. Charny, W. F. Courtney and J.-Y. Le Boudec, "Delay Jitter Bounds and Packet Scale Rate Guarantee," *IEEE/ACM TRANSACTIONS ON NETWORKING,* vol. 10, no. 4, August 2002.

[45] J. Ferreira, P. Pedreiras, L. Almeida and J. A. Fonseca, "The FTT-CAN protocol for flexibility in safety-critical systems," *Micro, IEEE,* vol. 22, no. 4, pp. 46-55, Jul/Aug 2002.

[46] M. Fidler, "Extending the Network Calculus Pay Bursts Only Once Principle to Aggregate Scheduling," *Springer: In Proceedings of QoS-IP 2003, LNCS,* 2003.

[47] M. Feldmann and D. Biskup, "Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches," *Computers and Industrial Engineering,* vol. 44, pp. 307-323, 2003.

[48] Y. Jiang and Q. Yao, "Impact of FIFO Aggregation on Delay Performance of a Differentiated Service Network," *Springer LNCS 2662,* 2003.

[49] M. Fidler, "On the Impacts of Traffic Shaping on End-to-End Delay Bounds in Aggregate Scheduling Networks," *Springer, LNCS 2811, Proceedings of Cost 263 QoFIS,* pp. 1-10, 2003.

[50] J. Jasperneite and P. Neumann, "How to guarantee realtime behavior using ethernet," *In 11th IFAC Symposium on Information Control Problems in Manufacturing (INCOM2004), Salvador-Bahia, Brazil,* April 2004.

[51] J. Loeser, T. Dresden and H. Haertig, "Using Switched Ethernet for Hard Real-Time Communication," *International Conference on Parallel Computing in Electrical Engineering (PARELEC 2004),* pp. 349-353, 2004.

[52] E. Bini and G. C. Buttazzo, "Schedulability Analysis of Periodic Fixed Priority Systems," *IEEE Transactions on Computers,* vol. 53, no. 11, pp. 1462-1473, 2004.

[53] O. Petrovic, C. Kitts, R. Rasay and M. MacKinnon, "NETROL: An Internet-Based Control Architecture for Robotic Teleoperation," *Conf Advanced Intelligent Mechatronics, Monterey, CA,* July 2005.

[54] R. J. Bril and P. J. L. Cuijpers, "Analysis of hierarchical fixed-priority pre-emptive scheduling revisited," *TU/e, CS-Report 06-36,* December 2006.

[55] S. Gopalakrishnan, M. Caccamo and L. Sha, "Switch scheduling and network design for real-time systems," *In Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium,* 2006.

[56] D. M. Cuong, M. K. Kim and H. C. Lee, "Supporting Hard Real-time Communication of Periodic Messages over Switched Ethernet," *Strategic Technology, The 1st International Forum on,* vol., no., pp. 419-422, Oct. 2006.

[57] R. Yuen and N. F. Xavier, "Simultaneous delivery of wireless LAN and cellular radio signals over optical fiber," *GCC Conference (GCC), 2006 IEEE,* vol., no., pp. 1-6, March 2006.

[58] J. Hurink and J. J. Paulus, "Special Cases of Online Parallel Job Scheduling," CTW University of Twente, pp. 82-85, 2008.

[59] H. Zhou, C. Nicholls, T. Kunz and H. Schwartz, "Frequency Accuracy and Stability Dependencies of Crystal Oscillators," *Carleton University, Systems and Computer Engineering, Technical Report SCE-08-12,* November 2008.

[60] R. Santos, R. Marau, A. Oliveira, P. Pedreiras and L. Almeida, "Designing a costumized Ethernet switch for safe hard real-time communication," *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on,* vol., no., pp. 169-177, May 2008.

[61] M. Jakovljevic, "Synchronous/asynchronous Ethernet networking for mixed criticality systems," *Digital Avionics Systems Conference, 2009. DASC '09. IEEE/AIAA 28th,* vol., no., pp. 1.E.3-1-1.E.3-10, Oct. 2009.

[62] R. Santos, R. Marau, A. Vieira, P. Pedreiras, A. Oliveira and L. Almeida, "A synthesizable ethernet switch with enhanced real-time features," *Industrial Electronics, 2009. IECON 09. 35th Annual Conference of IEEE,* vol., no., pp. 2817-2824, Nov. 2009.

[63] D. A. Miller and A. E. Kamal, "FlexTDMA for Delay-Stable Communications In an Asynchronous Network," *Proceedings of the IEEE Local Computer Networks (LCN) Conference,* 2010.

[64] G. Y. Keung, B. Li and Q. Zhang, "Message Delivery Capacity in Delay-Constrained Mobile Sensor Networks: Bounds and Realization," *Wireless Communications, IEEE Transactions on,* vol. 10, no. 5, pp. 1552-1559, May 2011.