Delay-Stable Communications in Simultaneous Multicast Networks

David A. Miller and Ahmed E. Kamal
Dept. of Electrical and Computer Eng., Iowa State University, Ames, IA 50011, U.S.A.
E-mail: {davidam, kamal}@iastate.edu

Abstract—**To properly operate closed industrial control networks, it is required that communication with nearly constant delay bounds be supported. In [24] the authors introduced FlexTDMA in order to provide this support, and with minimal delay-jitter in an asynchronous network, under unicast communication. In this paper we consider providing this support for simultaneous multicasting, and introduce the FlexTDMA++ protocol. Under this protocol, and with periodic on-off traffic that is directed to multiple receivers, frame losses and switch failures are managed in the presence of component clock drifts and bandwidth loads.**

I. 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. For example, robotic controllers communicate through a network to control multiple target robots. 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. In [24] the authors introduced a new strategy, FlexTDMA, which provided near constant delay bounds, with minimal delay jitter in such networks, when one-to-one (or unicast) communication is used. In many systems, the 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 paper 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 [25] using a probabilistic model, and in [18] by exchanging messages for synchronous operation. References [17] [21] [23] [22] support nearly constant delayed traffic in a synchronous network, and require message exchange to maintain a synchronous state. Simultaneous message arrival is supported in [19] with a solution not designed for packet networks, and in [10] for TCP Internet connections by using bandwidth reservation. The authors of [16] achieve SM by attaching a transmit release time-stamp to messages while maintaining a synchronized state. The authors of [15] 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 [24] 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.

The remainder of this paper is organized as follows. In Section II, we review FlexTDMA. In Section III, we introduce the basics of our approach for supporting SM, FlexTDMA++, and review Gang Scheduling and preemption approaches which are used for concurrent scheduling by FlexTDMA++. In Section IV, we characterize the delay bound calculations of SM under FlexTDMA++. In Section V, we introduce the operational details of FlexTDMA++. Section VI includes a performance study of FlexTDMA++, which is followed by a summary of findings and conclusions in Section VII.

II. Background

In this section we describe the key properties of the FlexTDMA protocol. The FlexTDMA frame processing stages, shown in Algorithm 1, are arrival, eligibility, and transmission. When a frame arrives the flow ET is determined. The frame deadline is determined from ET and the flow delay bound computed at the output port. A frame, held until eligibility, is scheduled for transmission on flow$_{01}$ or in a FIFO queue.

Algorithm 1: 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] +
        $X_{\min} \cdot (1 - clockDrift)$)
        frame deadline ← ET[k] + flow(k) delay bound
        enqueue(Eligibility Queue, frame)
    case Frame Eligibility
        frame ← dequeue(Eligibility Queue)
        store frame in FIFO or Flow$_{01}$
    case Frame Transmission Completion
        if flow$_{01}$ queue head scheduled time > now then
            frame ← dequeue(Flow$_{01}$ Queue)
            Baselined[flow k] ← true
            transmit frame
        else if not empty(FIFO Queue) then
            frame ← dequeue(FIFO Queue)
            transmit frame
```

In [24] we defined a *Baselined flow* as: *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 newly baselined frame causes a series of eligibility times in subsequent switches. Once a maximally delayed frame k is received by switch j from switch j-1, the eligibility time ET_j^k will be computed as

$$\max\left(AT_j^k, ET_j^{k-1} + X_{\min} \cdot (1 - clockDrift)\right) \quad (1)$$

where X$_{min}$ is the minimum frame inter-arrival time on the flow and AT_j^k is the arrival time of frame k at switch j. This generates a series of eligibility times separated by the frame period reduced by the clock drift so that the eligibility times will not out pace arriving frames.

A virtual flow called flow$_{01}$ provides high priority transmission opportunities for baselining [24]. Flows are scheduled for baselining by reserving a transmission opportunity in flow$_{01}$ constrained by the allocated transmission period. Flow$_{01}$ scheduled transmissions are queued in ascending order.

III. Simultaneous Multicasting

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

A. FlexTDMA++ Simultaneous Support Mechanisms

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

1) Coordinated Baselining: FlexTDMA++ switches initiate a baseline event on a flow for each forwarded frame instance so all sub-trees experience a baseline cascade. Each sub-tree root switch schedules a baseline event on the flow.

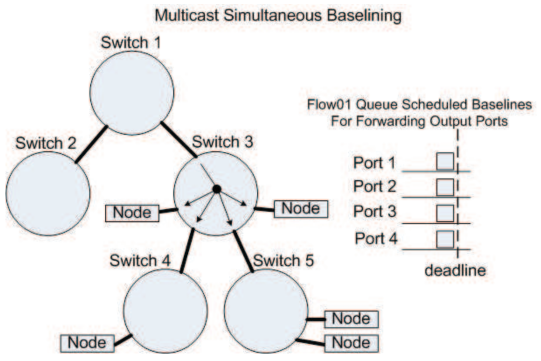
Multicast Simultaneous Baselining
Switch 1
Switch 2
Switch 3
Switch 4
Switch 5
Node
Node
Node
Node
Node
Flow01 Queue Scheduled Baselines
For Forwarding Output Ports
Port 1
Port 2
Port 3
Port 4
deadline

Fig. 1. Simultaneous baselining at all forwarded ports establishes a common eligibility time basis.

Figure 1 shows how concurrent baselining of a flow forwarding set establishes a common ET basis. 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 at each output port. By using the same frame forwarding event the ET basis of each connected switch is updated. The two sub-trees switch 4 and 5 receive the baselined frame and in turn establish a common ET basis.

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. 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 [11], [7], [6].

2) FlexTDMA++ Gang Scheduling: Under Gang Scheduling, tasks of a job are scheduled in the same time slice. This is often used to schedule software tasks of jobs onto a set of available processors [14] [1]. Bin fitting is used to maximize the utilization of processors. For FlexTDMA++, Gang scheduling provides the mechanism to schedule coordinated baselining events. A job is an arriving frame and the task-set is the output ports the flow is transmitted [5]. When an arriving frame on a flow is selected for Gang scheduled baselining, a baseline event will be planned to each forwarding output port.

B. FlexTDMA++ Gang Scheduling and Preemption

Here we review the gang scheduling approaches and preemption strategies.

1) Preemption Classes: FlexTDMA++ allows preemption of a flow scheduled for baselining in flow$_{01}$ when its deadline conflicts with the flow of an arriving frame. There are three classes of preemption under FlexTDMA++ Gang scheduled multicasting. The preemption type 'none' allows no preemption, 'Per Output Port' allows preemption on a per port basis, and 'strict' which requires all ports of the same flow be preempted if any are [8].

2) Gang Scheduling Bin Fitting Strategies: FlexTDMA++ supports six Gang scheduling approaches:

- *First Fit:*
  Under the first fit policy [20] a Gang scheduler allocates the first set of available machines at the first matching opportunity.
- *Concurrent Gang:*
  Under a concurrent gang policy [11] flow preemption priority is given to a flow with less time remaining to its baseline deadline.
- *Lazy Gang:*
  Under a lazy gang policy [7] flow preemption priority is given to a flow having more frame arrivals after the baseline deadline is exceeded.
- *Best Fit:*
  Under a best fit policy [12] [13] a Gang scheduler gives priority to a flow reducing idle output ports.
- *FCFS w/Backfill:*
  Under the FCFS with Backfill policy [13], [4], [9] a Gang Scheduler, with a certain probability does not commit when one or more ports are left idle. No preemptions are allowed. Three probability values are tested. Probabilities of no commit with 1 idle output port are 5%, 10% and 15%, and for 2 idle output ports 25%, 50% and 75%.
- *Bandwidth Weighted Fit For FlexTDMA:*
  We propose the Bandwidth Weighted Fit algorithm which gives preemption priority to a frame on a non-baselined or baseline deadline exceeded flow having lower bandwidth than the currently scheduled flow in flow$_{01}$. Low bandwidth flows are given priority as they have fewer potential baseline event opportunities.

IV. FlexTDMA++ Switch Delay Bound Computation

In this section we review the approach used to compute delay bounds in each FlexTDMA++ switch to support SM.

Per Port Delay A delay bound d_i is computed for each port using configured flows. The delay bound d_i is computed based on schedulability analysis of flows forwarded to the port.

Sub Tree Depth The value $subtree_{k,f}$ is the maximal delay depth of switch k for flow f which is the frame ET at switch k to the maximal delay leaf node. Figure 2 shows the computation of subtree depth and assigned delay bound for a flow in switch 3. The delay bound for output ports 1, 2, 3 and 4 are 1, 2, 3 and 4 ms, respectively. The subTree delay depth

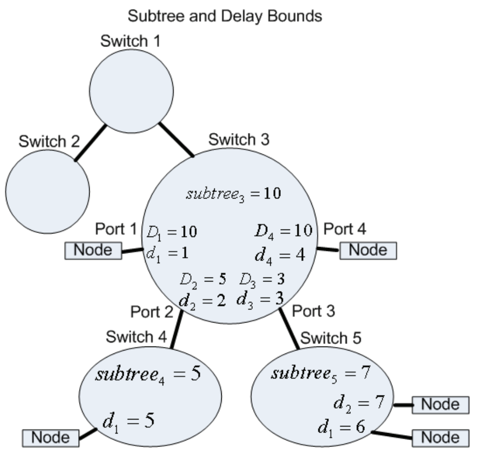

Fig. 2. Delay bounds are computed to insure equal subtree depth per port.

of switch 3 is determined as the maximum difference between ET at switch 3 and AT at the destination node. The subTree depth of switch 3 is computed as the maximum of the delay depth for each output port as subTree$_3$ = 10 = max(1, 7, 10, 4) = max(d$_1$, d$_2$ + subTree$_4$, d$_3$ + subTree$_5$, d$_4$).

Assigned Port Delay Switch k, will establish a modified maximal delay $D_{k,port,f}$ for flow f at output port 'port' in order to insure that the delay-depth of each sub-tree is the same. The delay $D_{k,port,f}$ is 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' (e.g. $switch(3, 2) = 4$), and $subtree_{switch(k,port),f}$ the delay depth of the switch connected to port number 'port' or zero when a leaf node is connected to port number 'port'. The assigned delay bounds shown in Figure 2 are set to the subTree$_3$ - subTree$_{downstream\,switch}$. The assigned delay bounds are computed as follows:

- D$_1$ = 10 = 10 - 0 = subTree$_3$ - 0,
- D$_2$ = 5 = 10 - 5 = subTree$_3$ - subTree$_4$,
- D$_3$ = 3 = 10 - 7 = subTree$_3$ - subTree$_5$,
- D$_4$ = 10 = 10 - 0 = subTree$_3$ - 0.

Notice for all ports of switch 3, D$_i$ + subTree$_{downstream\,switch}$ = 10 ms, so the delay bound from switch 3 ET to destination node AT is 10 ms.

V. FlexTDMA++ Switch Operation

In this section we review the operational details of the FlexTDMA++ protocol. Figure 3 shows the stages of frame processing in a FlexTDMA++ switch: 1) arrival, 2) eligibility, and 3) selection for transmission. Arriving frames are held until frame ET [3], [2]. Once eligible, the frame is scheduled for transmission at the output port in either the flow$_{01}$ or FIFO queue [24].

1) Frame Arrival At frame arrival the frame is stored pending eligibility.

2) Frame Eligibility Once a frame reaches its eligibility time it is processed for each forwarding output port. There are three steps to frame processing.

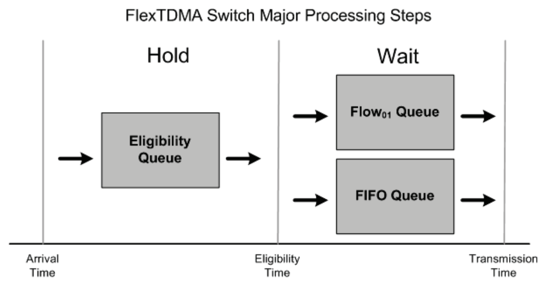

Fig. 3. Major processing phases and events of the FlexTDMA switch.

Eligibility Step 1 - Determine Frame Deadline The frame deadline is set to ET plus the port delay bound.

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 scheduled at the deadline time of the current flow. Figure 4 shows the minimal interval period p between

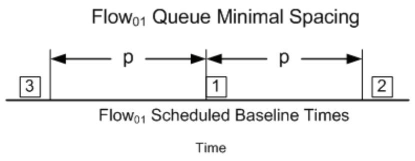

Fig. 4. Flow$_{01}$ queue is serviced using a minimal frame period

scheduled baseline times in the flow$_{01}$ queue. Baseline times are located in flow$_{01}$ in accordance with this constraint, but as close to the deadline as possible. The duration limits the extent to which a flow is scheduled in flow$_{01}$ prior to its deadline. The extent of partial baselining allowed depends on the baseline state of the flow. When the flow is baselined the duration limit is set to $driftPpmMax * ((now + BDinterval) - curBD)$ insuring the extent of early baselining is limited to the maximal drift that may occur until the baseline deadline. When a transmission opportunity is found in flow$_{01}$ the flow is marked as a candidate for baseline scheduling.

Eligibility Step 3 - Apply Gang scheduling Preemption Rules Using the Gang scheduling policies and preemption rules frames are scheduled in flow$_{01}$ or FIFO queue.

3) Frame Transmission A ready frame is selected from flow$_{01}$ if any, otherwise from FIFO queue.

VI. FlexTDMA++ Evaluation

In this section we describe the evaluation of FlexTDMA++. There are three phases of FlexTDMA++ evaluation: Periodic on-off, Frame Loss and Switch Failure. A total of 13 combinations of preemption types and Gang Scheduling policies were tested as all gang scheduling policies support both strict and per port preemption with the exception of first fit and FCFS w/Backfill. FlexTDMA++ was evaluated by configuring the periodic on-off, frame loss or switch failure probability, Gang Scheduling and preemption policy, clock drift, and bandwidth load. The key performance criteria of FlexTDMA++ are: Time-to-baseline, Laxity and SM. The Time-to-baseline criterion is the time needed for an active flow to achieve a baselined state. The laxity criterion is the extent to which data is delivered to a destination node prior to the delay bound. The SM criterion is the maximal difference in delivery times of multicast instances of a source frame. The topology in Figure 5

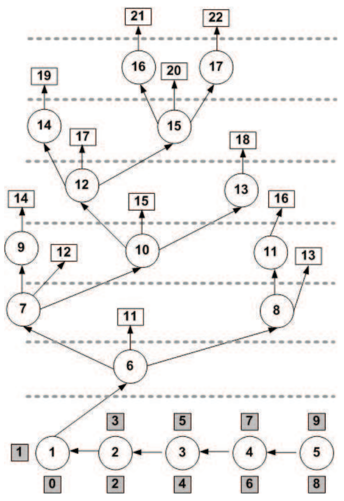

Fig. 5. FlexTDMA++ Testing Topology

shows the simulated physical topology used to demonstrate the use of FlexTDMA++ for SM. The transmissions from end nodes 0 to 9 are forwarded to receiving leaf nodes of the tree. The bandwidth is allocated to the flows of the transmitting nodes in increasing amounts so that the periodic transmissions phase causing collisions. The delay bounds for the flows of this testing topology range from 1,587 to 2,493 μs. This magnitude is important relative to the performance criteria evaluation.

A. Results

1) Clock Drift Effect on FlexTDMA++: Table I shows a summary of the effect of clock drift on FlexTDMA++ for each key performance criterion and for each phase of evaluation. Time-To-Baseline had the highest impact from increasing drift. Frame delay bound laxity performances were no more than 4% of the flow delay bound. The SM performance varies between clock drift types. Increasing drift consistently has

TABLE I
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)

the highest impact as it accelerates the number of baselines needed. FlexTDMA++ managed clock drift efficiently for all modes of operation for each key performance criterion, and clock-drift had a minimal but consistent result on performance.

2) Bandwidth Load on FlexTDMA++: The total bandwidth loading is set for each test. Table II shows the average performance relating to bandwidth loads of 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 SM 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 SM 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.

TABLE II
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

TABLE III
COMPARISON OF BEST FIT AND BW WEIGHTED GANG SCHEDULING

Performance Criteria	Parameter	Load Favoring Best Fit	Load Favoring BW Weighting
Time-to-Baseline	Frame Loss	50%-60% 3.9 - 4.1 ms	10%-40% 1.3 - 3.7 ms
	Periodic On-0ff	50%-60% 3.5 - 3.6 ms	10%-40% 1.1 - 2.8 ms
	Switch Failure	50%-60% 3.4 - 3.6 ms	10%-40% 1.0 - 2.6 ms
Laxity	Frame Loss	30%-60% 7 - 22 μs	10%-20% 26 - 28 μs
	Periodic On-0ff	40%-60% 1.4 - 2.0 μs	Same as Best Fit 10%-30% 0.7 - 3.0 μs
	Switch Failure	10%-60% 1.0 - 24 μs	none
SM	Frame Loss	20%-60% 45 - 66 μs	Same as Best Fit 10% 90 μs
	Periodic On-0ff	10%-60% 10 - 52 μs	None
	Switch Failure	10%-60% 30 - 95 μs	None

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 less than the rate needed to maintain the flows in a baselined state. Table III 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 criterion. 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 criterion 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 per port preemption. Under switch failure conditions best fit policy was favored for all bandwidth loads.

The conclusion of this comparison 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 SM performance is critical and bandwidth weighted policy when either time-to-baseline performance or laxity is critical.

4) Effect of Probability of Periodic On-Off, Frame Loss and Switch Failure on FlexTDMA++: Table IV shows the increase achieved by modifying the probability 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 fundamentally 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 SM. Further evaluation uses maximum probability values.

TABLE IV
IMPACT OF PROBABILITY OF FLOW INTERRUPTION

FlexTDMA++ Phase	Periodic On-Off	Frame Loss	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) Improvements to FlexTDMA++: We consider the comparative performances of the gang scheduling algorithms using bandwidth loading 50% and 90%. Table V 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 SM, 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 laxity performance criterion favors the best fit policy using per port preemption. The SM performance criterion favors best fit policy using per

TABLE V
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%	SM	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%	SM	Best Fit (Per Port)	10 - 42 μs
50%	SM	Concurrent Gang (Per Port)	15 - 50 μs
50%	SM	Lazy Gang (Strict)	17 - 56 μs
50%	SM	BW Weighted (Strict)	17 - 62 μs

port preemption.

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 SM criteria. When time-to-baselining performance criterion is most important the strict preemption policy should be used.

VII. SUMMARY

In this paper we introduced an enhancement to the FlexTDMA protocol to support SM. The details needed to support SM were characterized. An evaluation of several approaches to concurrent baseline scheduling and preemption policies supporting SM within FlexTDMA++ was completed. This evaluation demonstrated the ability of FlexTDMA++ to support SM. The evaluation showed the performances of FlexTDMA++ support of SM 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 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 SM 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.

REFERENCES

[1] 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.

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

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

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

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

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

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

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

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

[19] 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.

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

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

[22] 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.

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

[24] D. A. Miller and A. E. Kamal, "FlexTDMA for Delay-Stable Communications in Asynchronous Industrial Control Networks," *Proceedings of the IEEE Local Computer Networks (LCN) Conference,* 2010.

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