Bounded Delay for Weighted Round Robin with Burst Crediting

Sponsor: Sprint

Kert Mezger
David W. Petr

Technical Report TISL-10230-08

Telecommunications and Information Sciences Laboratory
Department of Electrical Engineering and Computer Sciences
University of Kansas

May 1995

Contents

1 Introduction

In progressing past the simple weighted round robin resource allocation method [1], the new system under analysis adds a burst crediting feature in order to smooth out the performance of extremely bursty sources [2]. The nature of these very bursty sources is such that they send information at rates much higher than their average rate for a short period followed by a lengthy interarrival time. This poses a problem when using the simple weighted round robin allocation method because several slots allocated to a bursty source may be skipped during the lengthy interarrival times which can lead to a reduction in observed bandwidth experienced by the bursty source. Another problem is that the service to the bursty source controlled by the weighted round robin allocation method may be so time infrequent as to contribute to a high burst delay. By introducing a burst crediting scheme, the goal is that bursty sources will be served in a timely manner without allowing the bursty nature to noticeably affect the performance of nonbursty sources.

2 Description of Burst Crediting

The burst crediting is analogous in some respects to a loan. When a burst comes into a queue and fills the queue to a certain maximum threshold point (Th_{max}), the queue is put into an interrupt condition which gives that queue highest priority. Whenever anything comes into that queue, it is immediately served at the beginning of the

next time slot. The only way to bring that queue out of the interrupt condition into a normal weighted round robin mode is for the server to observe the end-of-packet flag in the cell header from the bursty source queue as defined by AAL-5 [3].

In order to constrain the bursty source from taking over all the bandwidth of the server, a limit is placed on the number of cells a bursty source can have served while in the interrupt mode. In essence, the bursty source is allowed to borrow against its future service slots up to some borrowing limit. In order to pay off some of the borrowed service, two payback methods are offered. The first occurs when the bursty source has nothing to send and the weighted round robin allocation method has determined that the bursty source is the next queue to be served. In this situation, the weighted round robin schedule is immediately advanced to the next queue in the schedule as with the simple weighted round robin allocation method. In addition, one credit is paid back for having skipped the allotted servicing time.

The second payback method occurs when there *are* cells in the bursty source queue and the weighted round robin allocation method has determined that the bursty source is the next queue to be served under a non-interrupt mode (e.g. control or acknowledgment information is sent as opposed to a large information burst). To handle this case, a payback parameter (*PB*) has been set up such that under non-interrupt mode the bursty source will be forced to skip some scheduled service slots in order to pay back credits. For example, if *PB* is set to 3, two cells will be served and the third dedicated service slot will be skipped in order to pay back a credit. In

effect, for a short term gain in bandwidth, the queue will experience a reduction in bandwidth by at least a factor of (*PB*-1)/*PB* in order to compensate and equalize the bandwidth back to the guaranteed bandwidth.

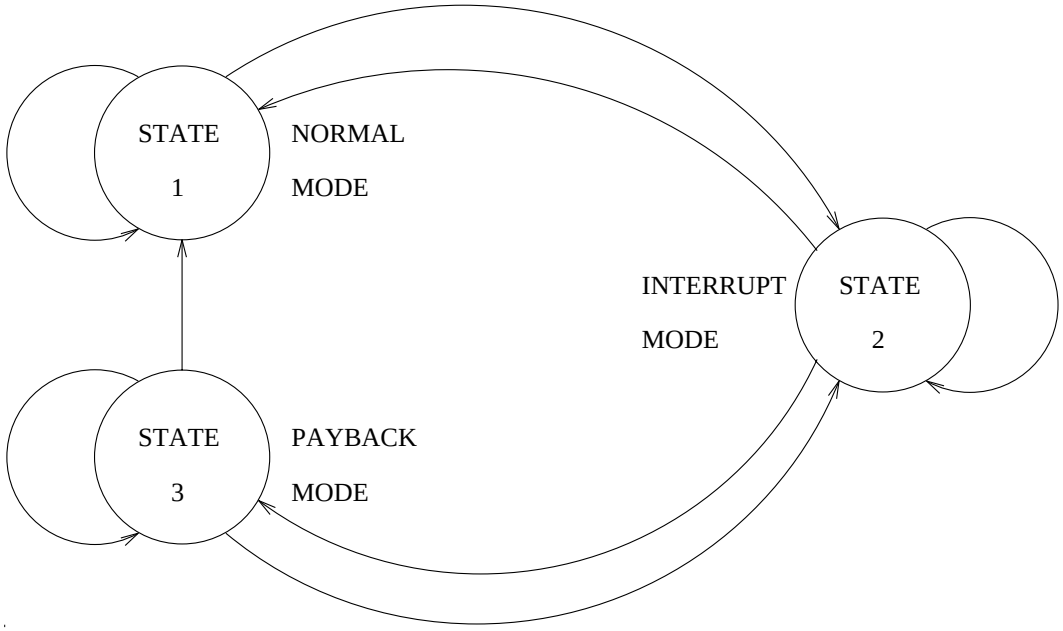

Figure 1: State Diagram of the Burst Crediting Scheme

A state diagram is shown in Figure 1 which consists of three states: the regular WRR mode (state 1), the interrupt mode (state 2), and the payback mode (state 3). The transition from state 1 to state 2 is triggered when the bursty source exceeds the threshold point. The transition from state 2 to state 1 occurs after the AAL-5 end-of-packet indicator is read, that last cell is served, and no credits need to be paid back. The transition from state 3 to state 2 occurs when a credit or credits are paid back and the bursty source has or receives new cells which are part of the current burst. The transition from state 2 to state 3 occurs when all of cells are

served from the bursty source queue with credits owed, an end-of-packet identifier is recognized with credits owed, or the credit limit is reached. The transition from state 3 to state 1 occurs when all of the credits are paid back and the bursty source has recognized the end-of-packet indicator without any further bursts queued. Note that state 1 cannot get to state 3 directly. Credits must be owed to get to state 3. While in state 1, the resource allocation mechanism acts as a simple WRR for all sources. While in state 2, the bursty source gets exclusive use of the server as long as there are cells in the bursty source's server queue. If the bursty source queue is empty, the nonbursty sources share the resources in a WRR fashion. State 3 is the same as state 1 except that the bursty source is in payback mode. In this situation, the bursty source must skip scheduled service based on the PB parameter described above.

In comparing further the throughput to the guaranteed bandwidth, two approaches were taken. The first is a limiting argument which shows that as time goes to infinity then the throughput of the bursty source approaches (from above) the guaranteed bandwidth. The argument assumes that the bursty source is constantly trying to use as much bandwidth as possible. In the following equation, BW rate is the guaranteed bandwidth rate, and CL is the maximum borrowing limit.

$$\lim_{time \to \infty} \text{BW rate * time + CL} = \text{BW rate * time}$$

Since bandwidth not used by the bursty source is available to the nonbursty sources according to their WRR allocation, the throughput of the non-bursty sources will also approach (from below) their guaranteed values. Thus, in the long term, the guaranteed bandwidth is preserved, and the short term effects of the borrowing are compensated.

The second approach is to look at how long the system takes on average to compensate for the borrowed bandwidth. One case will assume that one maximum size burst, equal to the borrowing limit, is sent followed by a long silence time. In this situation, the burst will immediately put the system into interrupt mode until the whole burst is served. If the burst is of size *CL* and the guaranteed bandwidth for the bursty source is *r*, the time to completely pay back all credits will be the following.

$$\text{Payback time} = \frac{CL}{r}$$

This equation is not effected by the payback rate because no new bursts arrive which would need to be served. If the burst arrives at time 0, the nonbursty sources will have recovered their lost bandwidth by the following.

$$\begin{aligned} \text{Recovery time} &= CL + \frac{CL}{r} \\ &= CL * (1 + \frac{1}{r}) \end{aligned}$$

This brings us to case two in which the original maximum size burst is followed by subsequent bursts with an average rate equal to the guaranteed bandwidth, yet the bursts are not large enough to trigger subsequent interrupt modes after the first one. In this situation, the payback rate does take effect since other bursts are waiting to be served after the initial burst. The average compensation time is therefore dependent on the payback parameter, *PB*, as well as the borrowing limit, *CL*, as shown in the following equation.

$$\text{Payback time} \quad = \quad \frac{PB * CL}{r}$$

The higher the payback parameter is the lower the payback time will be. In fact, the case two payback time is a scaled version of the average compensation time of case one based on *PB*. As before, nonbursty sources will have recovered their lost bandwidth by the following.

$$\text{Recovery time} \quad = \quad CL * (1 + \frac{PB}{r})$$

In starting to analyze the burst crediting scheme, the system model must first be explained. The entire system is shown in Figure 2. The non-bursty sources have the same configuration as that described in the simple weighted round robin method shown in Figure 3 where the maximum delay is measured from the output of the shaper to the output of the server.

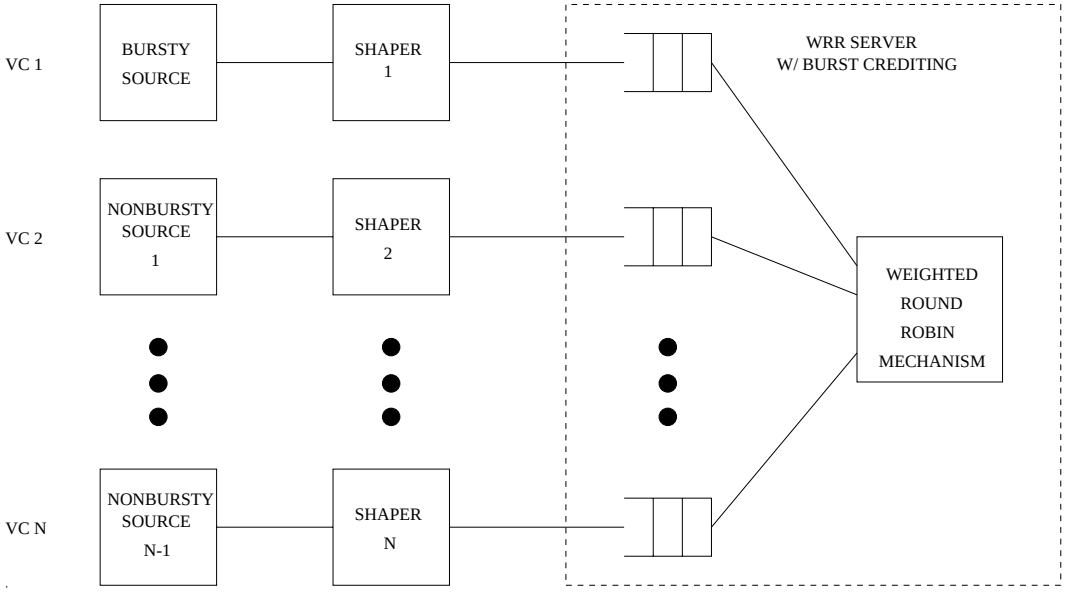

Figure 2: Network System Configuration

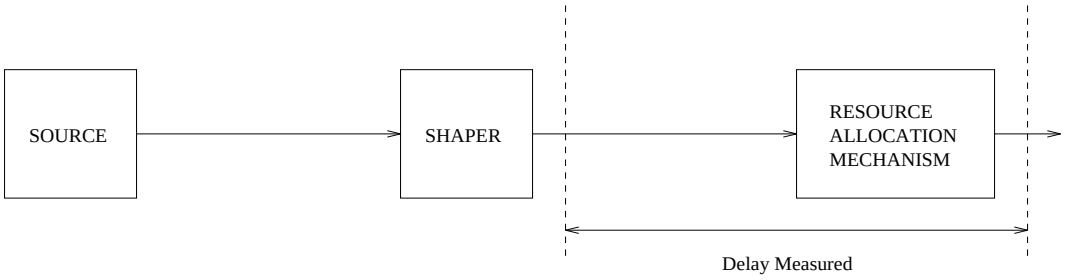

Figure 3: Nonbursty Source System Configuration

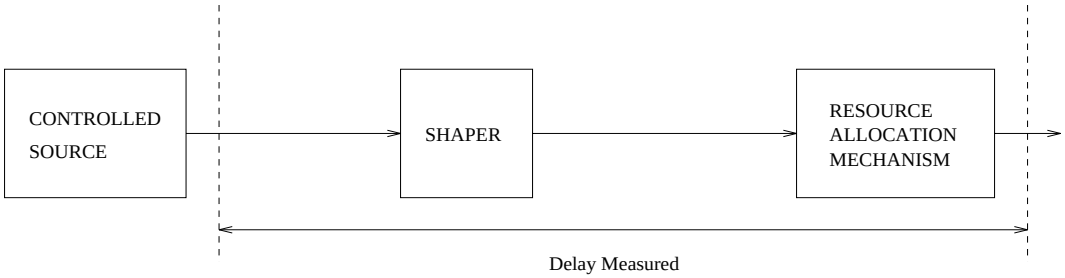

Figure 4: Bursty Source System Configuration

As for the bursty sources, it is configured as in Figure 4. A controlled source is put through a shaper and then finally into the weighted round robin with burst crediting. The routine of the control is discussed below. In contrast to the non-bursty sources, the maximum delay of the bursty sources is measured from the output of the controlled source to the output of the weighted round robin with burst crediting server. This is done because a more effective measure of delay is the maximum delay of the whole burst as it goes through the system. If the measure were to be taken from the output of the shaper to the output of the server, the measure would ignore any shaping delay. The reason it is ignored for the non-bursty sources is the fact that each cell is relatively independent of other cells and that overloading can occur on the user side which could contribute to an infinitely long shaping delay experienced by shaper queue build up. In effect, the shaper is providing source control so that the traffic leaving the shaper conforms to the parameters of the WRR server.

However, due to the extreme burstiness of the bursty source and the presence of the burst crediting mechanism, a simple shaper cannot adequately provide this control for the bursty source. Hence, the control is transferred to the source itself. In effect, the bursty source can look into the server parameters to see how many credits it can borrow based on the number of credits already used and the number of cells already in the shaper and server queues. If the number of credits is not sufficient to trigger the burst crediting interrupt, the source will wait a random amount of time

and check again until the minimum number of credits is reached. These operations of the controlled source are used to insure a minimum and maximum size of the bursts such that every burst is handled in the interrupt mode. The shaper is still included to provide some moderate rate smoothing to the bursts to prevent the bursty sources from completely "hogging" the link during a (relatively long) burst.

3 Results

3.1 Maximum Delay for Bursty Sources

This section describes the maximum delay experienced by bursty sources as well as the conditions under which maximum delay occurs.

3.1.1 Sliding Window Shaper

As with the simple WRR analysis, the first implementation of the system is with a sliding window shaper and a block scheduling window in the weighted round robin. The parameters of the bursty system are as follows.

i = number of cells allowed to be transmitted
from the shaper in a given sliding window

m = number of cells in a sliding window

s = number of slots guaranteed to the customer in

a scheduling window

t = number of total slots in the scheduling window

CL = maximum number of credits the customer is allowed to borrow

= maximum burst size

Th_{max} = number of cells in the bursty source server queue which puts the server into interrupt mode

PB = the frequency of credit payback in non-interrupt mode

For simplification, the following assumptions are made pertaining to the parameters.

$$i = s$$

$$i = Th_{max}$$

This insures that a mini-burst out of the shaper will trigger the interrupt mode in the server. In addition to the above parameters, two more parameters are added.

j = scaling factor of CL to i

k = scaling factor of the shaper rate to the server rate

These parameters yield the following relationships.

$$CL = j * i$$

$$i/m = k * (s/t) = k * r = R$$

We assume that *j* is at least 2 so that some shaping is always done. Also, in order to serve bursts faster than a regular weighted round robin scheme, the shaper rate should be faster than the regular WRR server rate for the bursty sources. Again, *k* should be greater than or equal to 2. It is also assumed that the bursty source has a low guaranteed bandwidth such that *i* is less than 1/*r*. With this constraint on *i*, the shaped cells can cause the system to go into interrupt condition under worse case conditions without having the regular WRR serving part of the burst before the threshold is reached. The above assumptions help in making sure that a burst from a bursty source will trigger the interrupt condition every time.

A time diagram of the worst case scenario is shown in Figure 5 for the block scheduling case and in Figure 6 for the distributed scheduling case where *i* = 5, *m* = 25, *s* = 5, *t* = 50, Th_{max} = 5, and *CL* = 10. From these values, both *j* and *k* are equal to 2. The maximum delay occurs when a maximum length burst is sent by the controlled bursty source during a period of congestion. The congestion doesn't allow the burst to be shaped until the end of the shaper window. The maximum burst delay of interest is measured from the time the first cell is emitted from the source until the time the last cell is served. Since cells are served immediately after

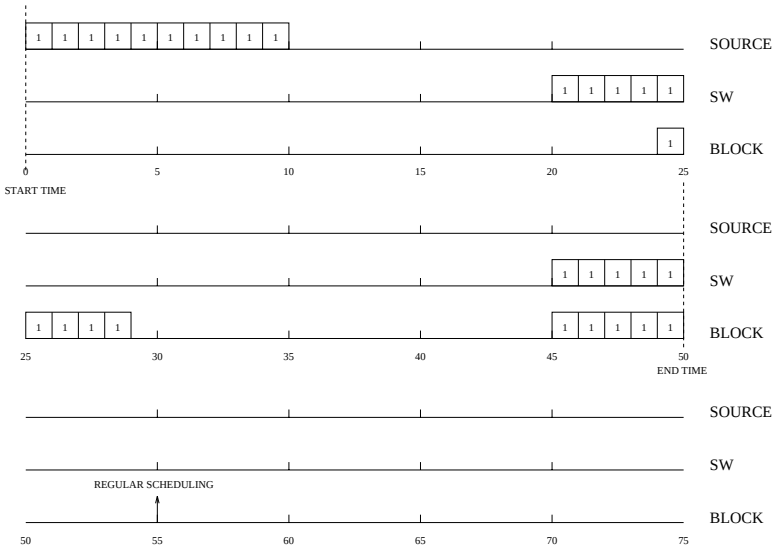

Figure 5: Time reference for Sliding Window/Block Scheduling Case

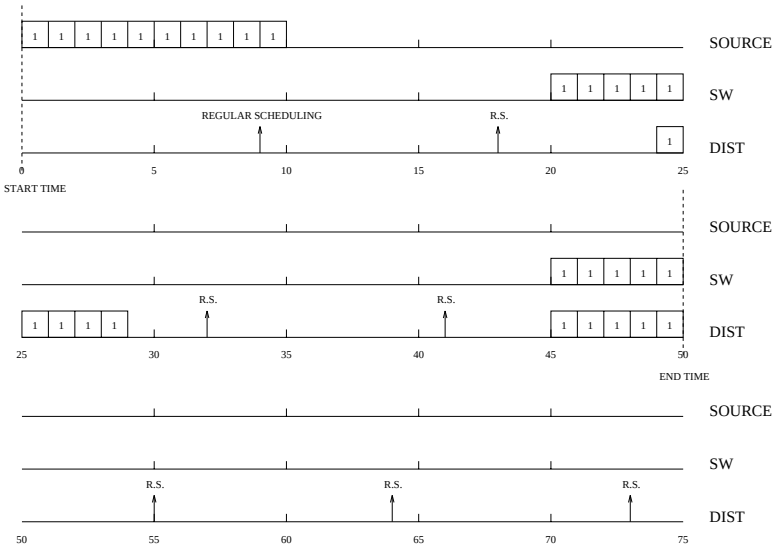

Figure 6: Time reference for Sliding Window/Distributed Scheduling Case

being shaped while in interrupt condition and every burst from the bursty source is eventually served in interrupt condition, it is seen that the shaper will control the amount of maximum delay, and the type of WRR scheduling (Block vs. Distributed) will not affect this delay. In fact, the maximum delay is the amount of time it takes to shape the maximum sized burst. This leads to the following equation.

$$\text{Maximum Delay} = (j*i)/(k*r)$$
$$= CL/R$$

It should be noted that maximum delay and *CL* are measured in terms of cells or cell slots. This leads to the restriction that $R=i/m$ should be set such that maximum delay (CL/R) yields an integer value. If it does not yield an integer value (e.g. $R = 3/17$, $i = 3$, $m = 17$, $j = 7/3$, and $CL = 7$ giving a maximum delay of 119/3), a more complicated form arises under the sliding window shaping case in which a ceiling function must be performed on the variable *j* before entering it into the equation.

$$\text{Maximum Delay} = (\lceil j \rceil *i)/(k*r)$$

Simulations are set up to insure integer values for bounded maximum delay.

To verify this equation, simulations were performed which tested the maximum delay performance over a range of *i* values. These simulations concentrated

primarily on the bursty source delay and assumed that the other nonbursty sources were taking all of their allocated bandwidth. Since the source is controlled, the operation of the source looks into the server and shaper and sees how many credits are available. If this number is greater than or equal to Th_{max}, a burst is sent. Otherwise, a random wait time is incurred and another check is made at the end of that time. The burstiness and load of the source can be controlled by the average of the random wait time.

The results of the simulations are seen in Figures 7 and 8. Figure 7 compares the results of simulations using the sliding window shaper and a block scheduling window under three different load conditions against the theoretical maximum delay. The figure shows that the theoretical equation does in fact bound the delays. Figure 8 compares the results of simulation using the sliding window shaper and a distributed scheduling window under three different load conditions against the theoretical maximum delay. The least bursty simulation refers to the fact that the controlled bursty source has a small average interarrival time. For the given normalized guaranteed load, the average interarrival time should be the following based on a geometrically distributed burst distribution.

$$\text{Average Interarrival Time} = \frac{1}{\text{1-Load}}$$

This measurement is made in terms of cell slots. While the burst pattern is not actually geometrically distributed, the interarrival time is a good estimate for

the given load. For the least bursty simulation, the average interarrival time is half of what the interarrival time should be for the given load. Under this condition, if the control mechanism will not allow the source to send a burst, the source will wait another interarrival time. With the interarrival time being set at half of the normal interarrival time, bursts are often sent as soon as enough credits are available. This allows for a small variance in the interarrival time. The more bursty simulation has an average interarrival time which is equal to the average interarrival time for the given load. The most bursty simulation has an interarrival time which is twice that of the interarrival time for the given load. This leads to a larger variance in the interarrival time, and thus, this leads to a burstier source. Again, the theoretical equation bounds the delays. Note that in both cases, more bursty sources are more likely to encounter the maximum delay.

To show the general trend of the maximum delay equation, Figures 9 and 10 were created. The figures both assume a constant j and k value while the values of r and i vary. Figure 9 shows the maximum delay increasing linearly as i increases. Figure 10 shows the maximum delay decreasing inversely as r (and hence $R=i/m$ increases).

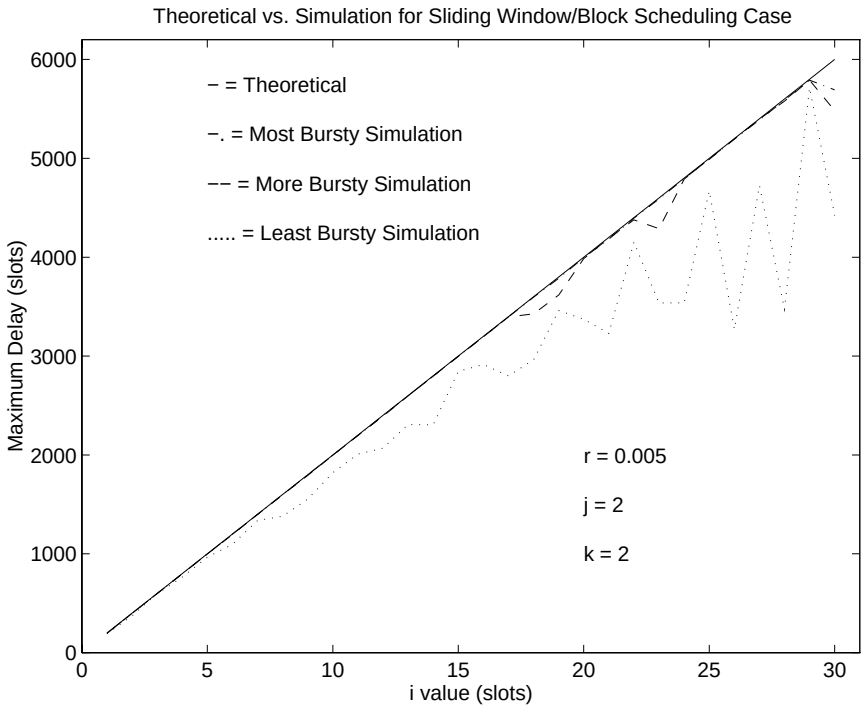

Figure 7: Theoretical vs. Simulation for Sliding Window/Block Scheduling Case

3.1.2 Leaky Bucket Shaper

Again, the same parameters are used here as in the sliding window case, except that *B* and *R* take the place of *i* and *i*/*m*.

$$B = \text{the leaky bucket size}$$

$$R = \text{the rate of token replenishment}$$

Similar to the sliding window case, the following relations are assumed.

$$B = s$$

$$B = Th_{max}$$

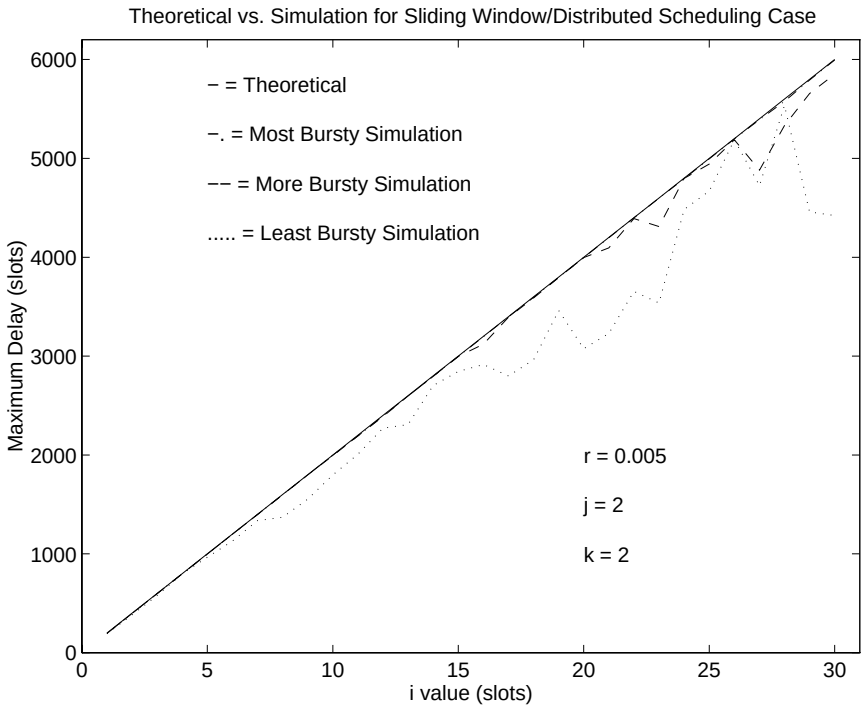

Figure 8: Theoretical vs. Simulation for Sliding Window/Distributed Scheduling Case

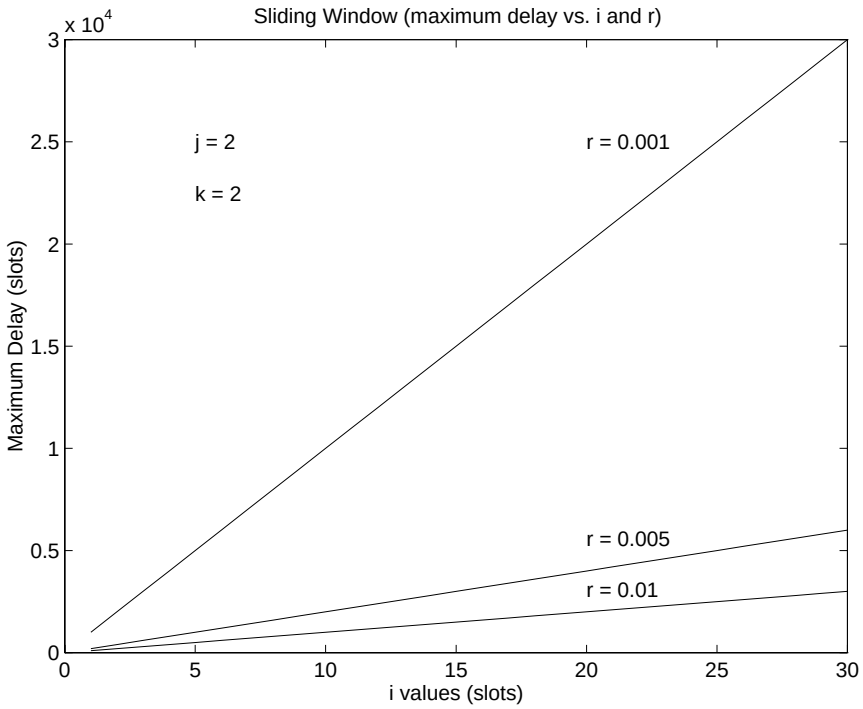

Figure 9: Sliding Window (maximum delay vs. i and r)

$$CL = j * B$$

$$R = k * (s/t) = k * r$$

A time diagram of the worst case scenario is shown in Figure 11 for the block scheduling case and in Figure 12 for the distributed scheduling case where *B* = 5, *R* = 0.2, *s* = 5, *t* = 50, Th_{max} = 5, and *CL* = 10. Again, the maximum delay occurs when a maximum length burst is sent by the controlled bursty source during a period of congestion. Now, the congestion doesn't allow the leaky bucket to fill, therefore, the burst is shaped out at the token replenishment rate. Once in interrupt mode, the token replenishment rate determines the speed at which the maximum

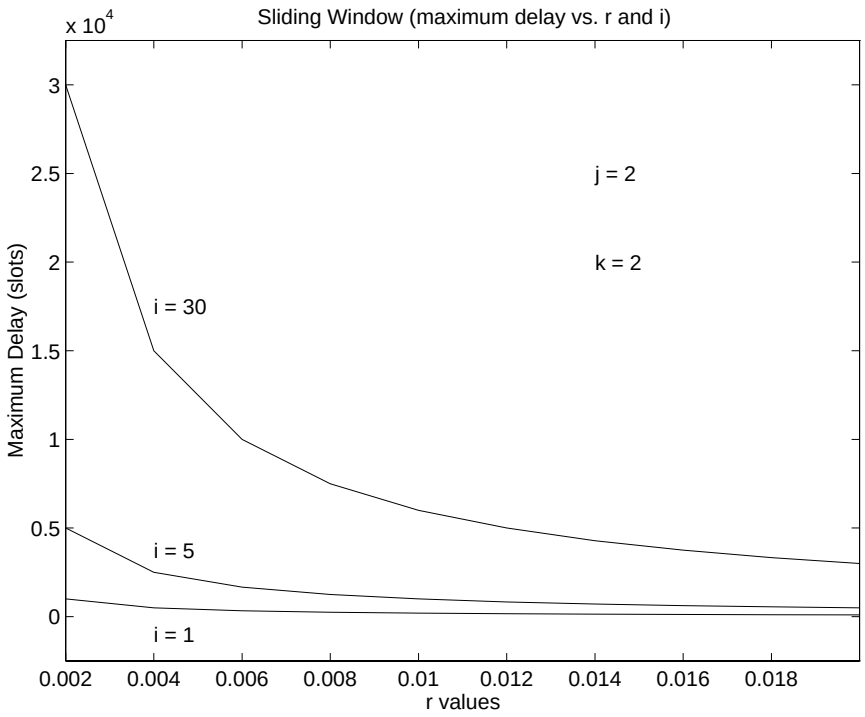

Figure 10: Sliding Window (maximum delay vs. r and i)

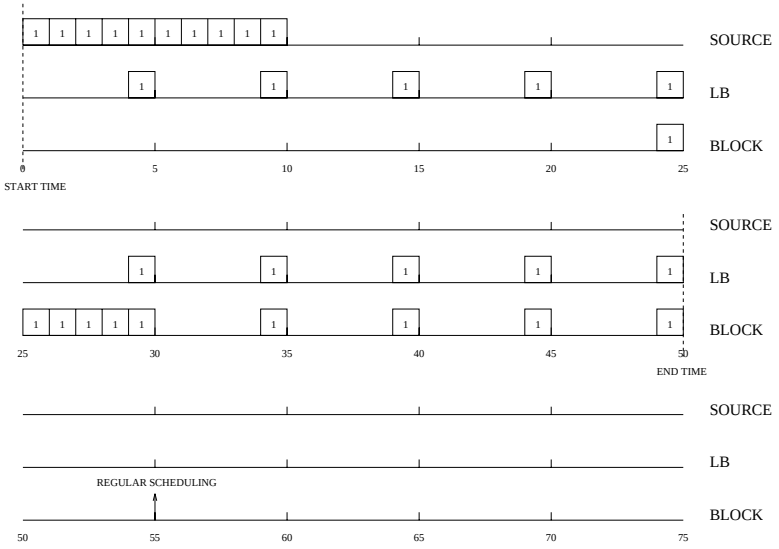

Figure 11: Time reference for Leaky Bucket/Block Scheduling Case

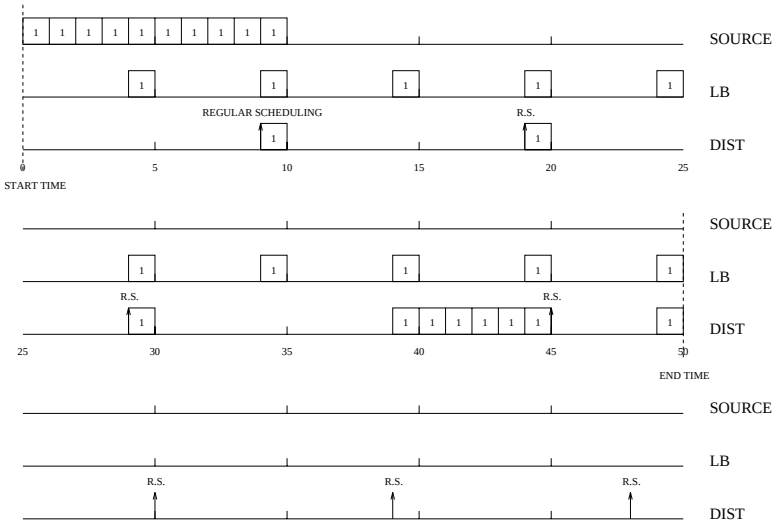

Figure 12: Time reference for Leaky Bucket/Distributed Scheduling Case

length burst is served. The following maximum delay equation comes from this reasoning.

$$\text{Maximum Delay} = (j*B)/(k*r)$$

$$= CL/R$$

Like the sliding window, a more complex form of the equation occurs when the parameters yield a noninteger value for the maximum delay. This form is as follows.

$$\text{Maximum Delay} = \lceil CL/R \rceil$$

This is the exact same burst delay experienced in the sliding window case. In fact, the plots in Figures 13 and 14 show the exact same relationships as those

found in Figures 13 and 14 except for the exchange of the variable *i* for *B*.

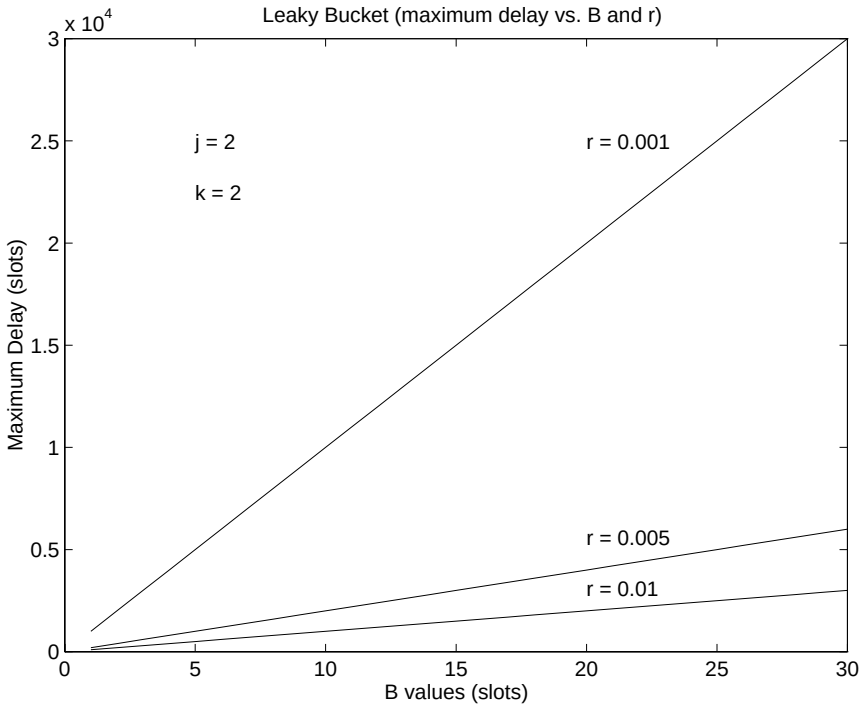

Figure 13: Leaky Bucket (maximum delay vs. B and r)

To verify the above equations, a simulation which is similar in structure to that of the sliding window cases was used (only the shaper structure was changed). The results of the simulations are shown in Figures 15 and 16. As with the sliding window simulations, the maximum delays experienced in the leaky bucket simulations are contained by the theoretical maximum delay bound.

3.2 Maximum Delay for Nonbursty Sources

This section describes the maximum delay experienced by nonbursty sources as well as the conditions under which this maximum delay occurs. In all four cases,

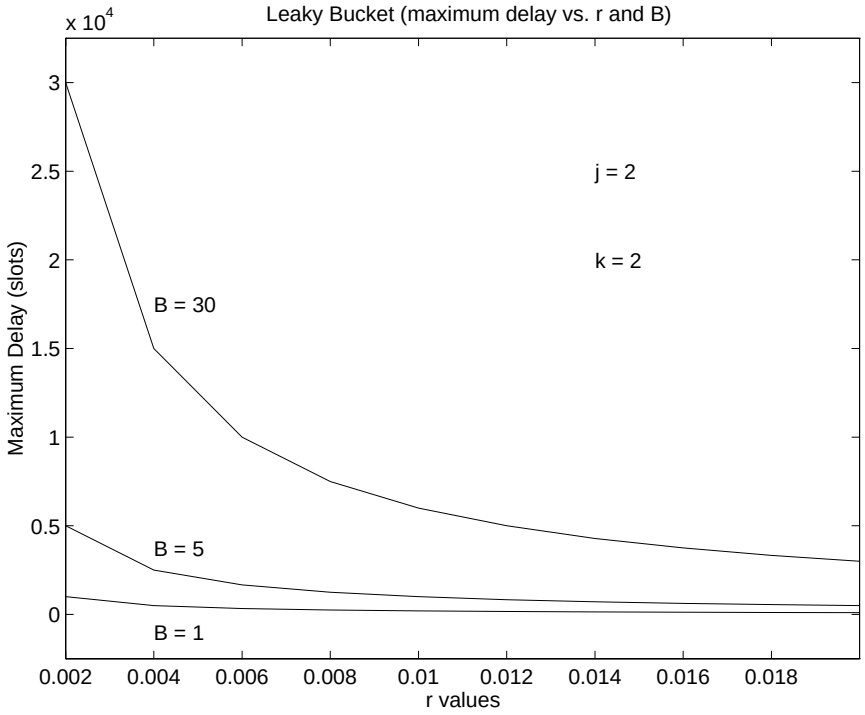

Figure 14: Leaky Bucket (maximum delay vs. r and B)

it is found that because of the *shaper*, nonbursty sources *will* be served during interrupt mode. The bursty source is served mainly in interrupt mode. Therefore, the maximum delay incurred by the nonbursty sources is primarily the theoretical delay bound found for the "classic" weighted round robin case [1]. To this delay is added the brief interruptions by the bursty source as it goes into interrupt mode and is served. The interruption is composed of two components: the interruption due to the maximum sized burst (*CL*) and the interruption due to other bursts being served which arrive during the extended maximum delay of the nonbursty source's delay

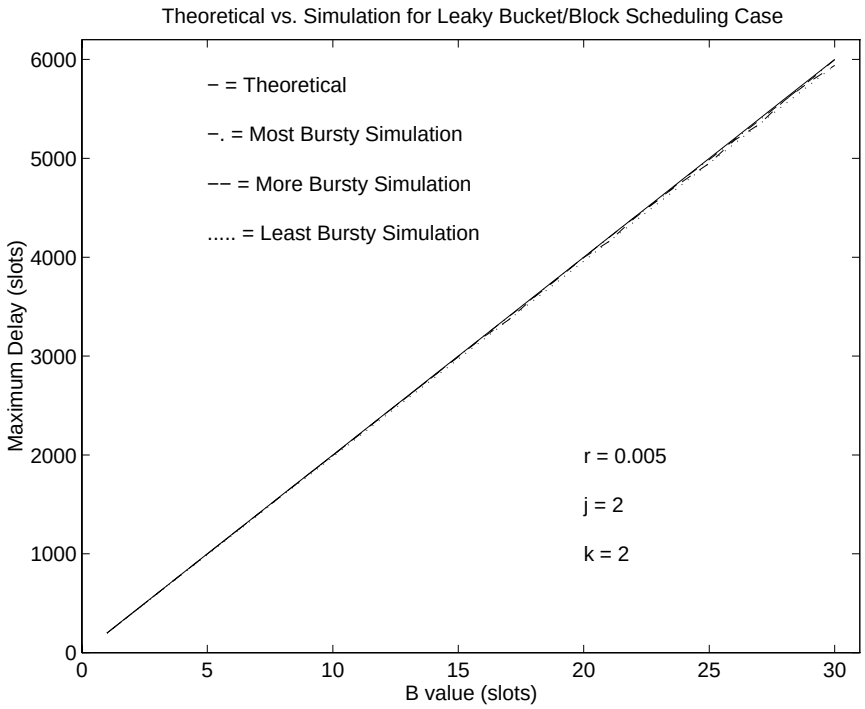

Figure 15: Theoretical vs. Simulation for Leaky Bucket/Block Scheduling Case

bound. The general form is shown below.

$$\text{Maximum Delay} \quad = \quad MD_{NB} + CL + \text{Other bursts}$$

Each subsection will describe the details of how the bursty source affects the maximum delay of the nonbursty source.

3.2.1 Sliding Window/Block Schedule

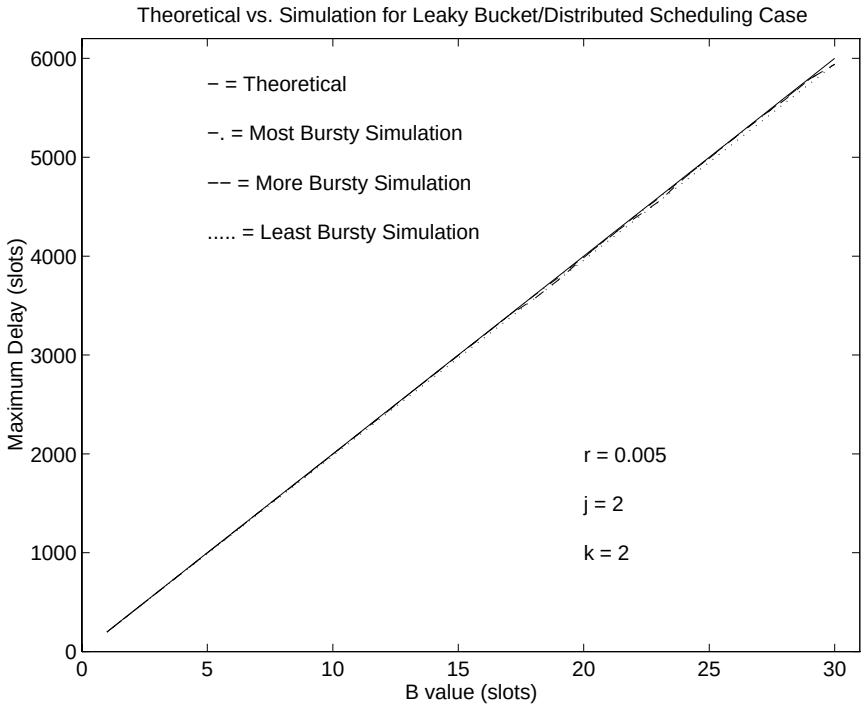

Figure 16: Theoretical vs. Simulation for Leaky Bucket/Distributed Scheduling Case

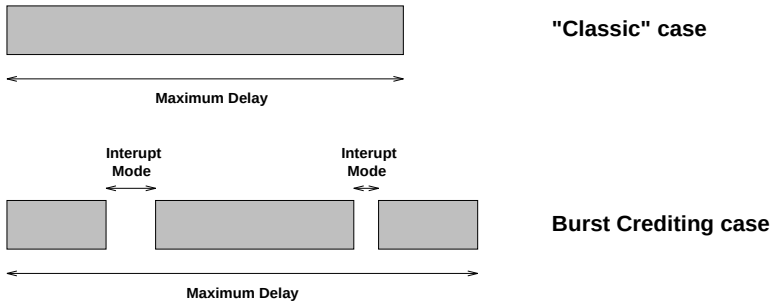

Figure 17: Worst Case Condition for Sliding Window/Block Schedule Case

A conceptual picture of the worst case condition is shown in Figure 17. From this diagram, the maximum delay equations are as follows.

$MD_{SWB/NB}$ = The maximum delay using "classic" weighted round robin under sliding window/block schedule case for nonbursty sources *only*

Maximum Delay = $MD_{SWB/NB}$ + CL + $\lceil MD_{SWB/NB}/(t\text{-}s) \rceil$ *s

In addition to the maximum delay using "classic" weighted round robin, the source experiences delays associated with the bursty source. The first term is a delay associated with the depletion of the burst credits within the maximum delay period. The second term in this equation refers to the number of bursty source scheduling blocks that can occur during the $MD_{SWB/NB}$ time period which can pay back credits and immediately be used to serve new bursts. The result is rounded up and multiplied by the number of slots, *s*, that can be used to pay back credits. This process will cause further interrupt periods which will extend the maximum delay time.

Simulations were performed for the sliding window/block schedule case. The results are in Figure 18. They show that for different nonbursty source loads the corresponding maximum delays are within the maximum delay bound. Also, note that the maximum simulation delays are approaching the maximum theoretical

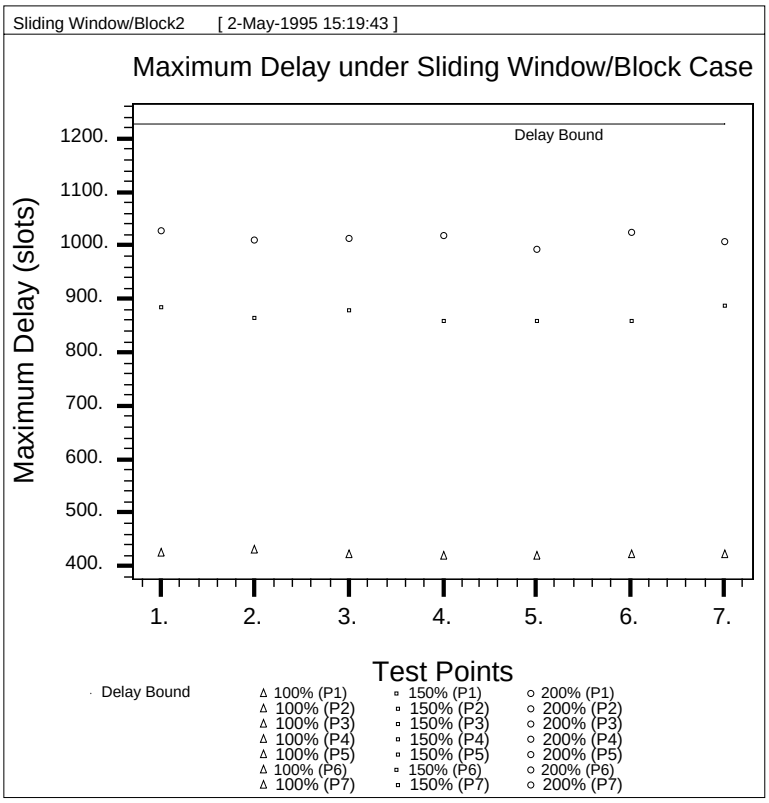

Figure 18: Simulation Results for Nonbursty Sources under Sliding Window/Block Schedule

delay as the nonbursty load increases. The maximum delay for the nonbursty sources without burst crediting for this case is 1188 slots. This shows that the contributions from the second two terms are negligible and that the simulation is not capturing the true bounding of the delays.

3.2.2 Sliding Window/Distributed Case

A conceptual picture of the worst case condition for this case is shown in Figure 19. This differs from the previous case in that the block scheduling can pay back the credits in block segments while the distributed scheduling pays back the credits

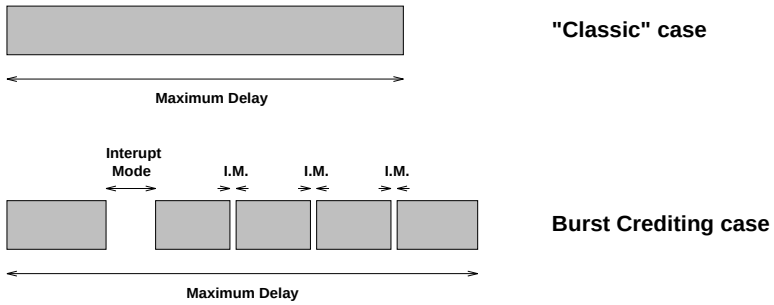

Figure 19: Worst Case Condition for Sliding Window/Distributed Schedule Case

periodically. Thus, the difference is seen in Figures 17 and 19. The maximum delay equations are as follows.

$MD_{SWD/NB}$ = The maximum delay using "classic" weighted round robin under sliding window/distributed schedule case for nonbursty sources *only*

Maximum Delay = $MD_{SWD/NB}$ + CL + $\left\lceil MD_{SWD/NB} * r/(1-r) \right\rceil$

In addition to the maximum delay using "classic" weighted round robin, the source experiences delays associated with the bursty source. The first term is a delay associated with the depletion of the burst credits within the maximum delay period. The second term in this case refers to the number of slots that can be paid back and immediately used for interrupt burst serving. Again, this process will add time to the maximum delay. The result is rounded up to compensate for situations where timing allows an extra interruption.

Simulations were performed for the sliding window/distributed schedule

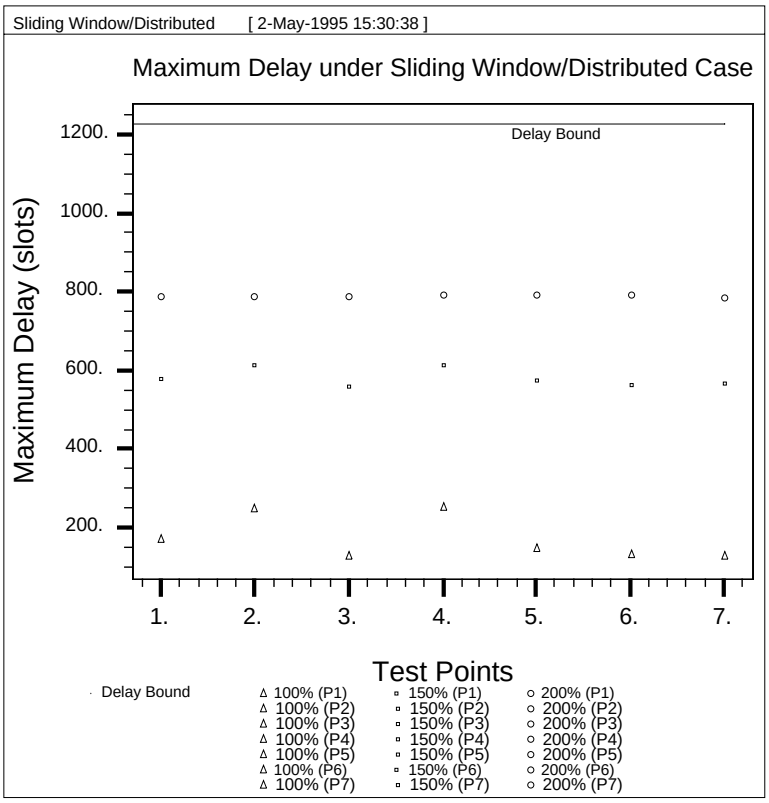

Figure 20: Simulation Results for Nonbursty Sources under Sliding Window/Distributed Schedule

case. The results are in Figure 20. They again show that for different nonbursty source loads the corresponding maximum delays are within the maximum delay bound and that the maximum delays approach the theoretical limit as load increases. The limit is again not realized due to simulation constraints. The maximum delay for the nonbursty sources is 1188 slots in this case. This shows the insignificance of the second two terms in this situation.

3.2.3 Leaky Bucket/Block Schedule

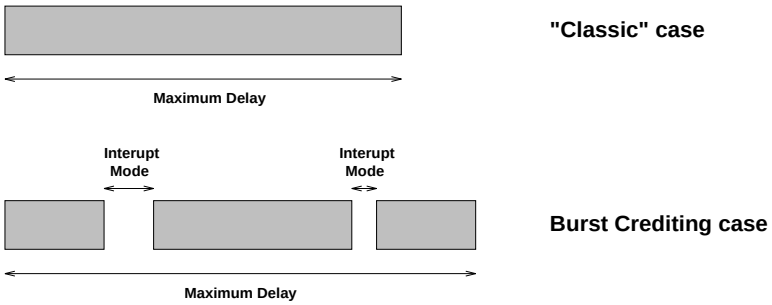

Figure 21: Worst Case Condition for Leaky Bucket/Block Schedule Case

A conceptual picture of the worst case condition is shown in Figure 21. It is the same as Figure 17. The maximum delay equations are as follows.

$MD_{LBB/NB}$ = The maximum delay using "classic" weighted round robin under leaky bucket/block schedule case for nonbursty sources *only*

Maximum Delay = $MD_{LBB/NB}$ + CL + $\left\lceil MD_{LBB/NB}/(t\text{-}s) \right\rceil$ *s

The explanation of the terms is the same as that of the sliding window/block schedule case.

Simulations were performed for the leaky bucket/block schedule case. The results are in Figure 22. They show that for different nonbursty source loads the corresponding maximum delays are within the maximum delay bound. The maximum delay for the nonbursty sources without burst crediting is 2574 slots. This shows that the simulation is not accurately capturing the worst-case situation.

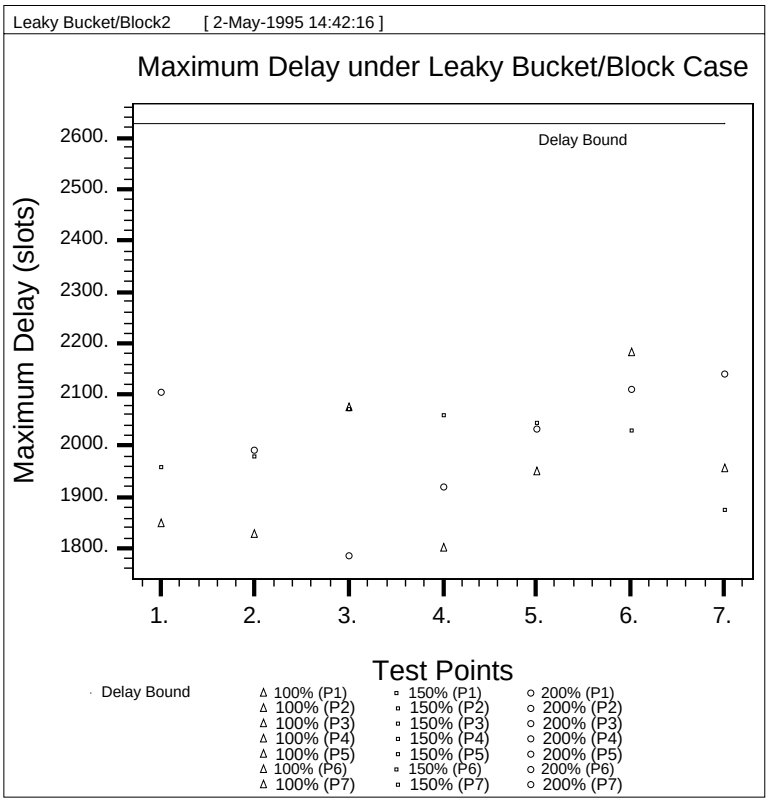

Figure 22: Simulation Results for Nonbursty Sources under Leaky Bucket/Block Schedule

3.2.4 Leaky Bucket/Distributed Schedule

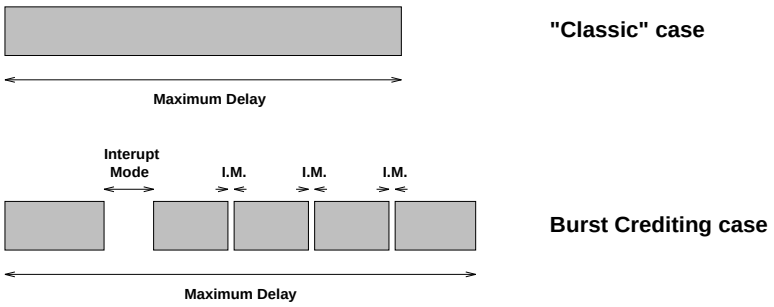

Figure 23: Worst Case Condition for Leaky Bucket/Distributed Schedule Case

A conceptual picture of the worst case condition is shown in Figure 23. It is the same as Figure 19. The maximum delay equations are as follows.

$MD_{LBD/NB}$ = The maximum delay using "classic" weighted round robin under leaky bucket/distributed schedule case for nonbursty sources *only*

Maximum Delay = $MD_{LBD/NB}$ + CL + $\lceil MD_{LBD/NB} * r/(1-r) \rceil$

The explanation of the terms is the same as that of the sliding window/distributed schedule case.

Simulations were performed for the leaky bucket/distributed schedule case. The results are in Figure 24. They show that for different nonbursty source loads the corresponding maximum delays are within the maximum delay bound. The maximum delay for the nonbursty sources without burst crediting is 1392 slots. Again, the last two terms are dominated by the first term.

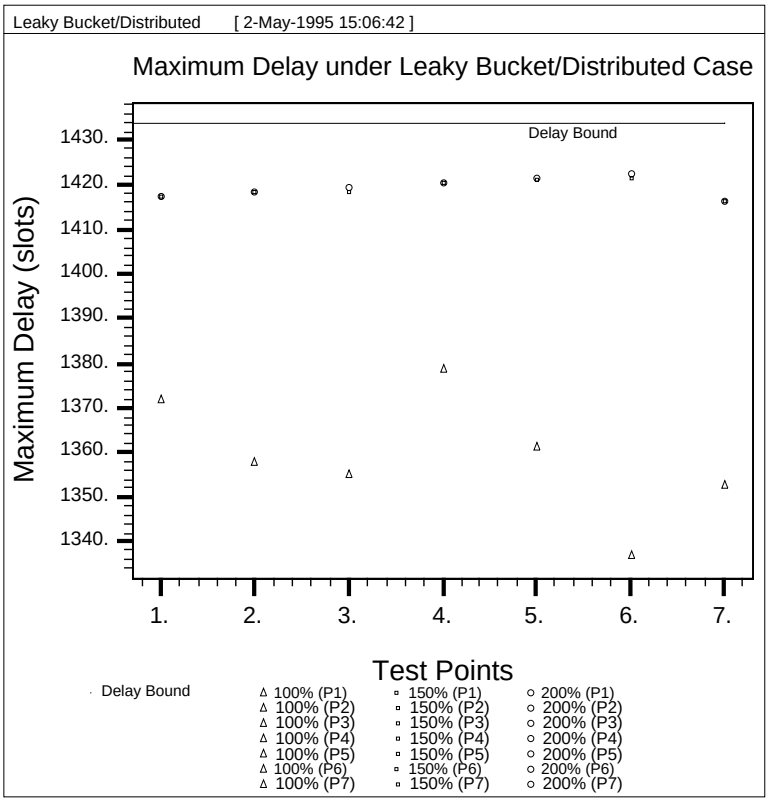

Figure 24: Simulation Results for Nonbursty Sources under Leaky Bucket/Distributed Schedule

A comparison plot of all four different cases is shown in Figure 25. This graph shows that simulations from the leaky bucket/distributed schedule case are the closest to the theoretical delay bound while the leaky bucket/block schedule case seems to level out somewhat before approaching the delay bound. This indicates that the simulations are not necessarily catching the worst case situation in all simulation cases. Since the bounded delay of the simple weighted round robin assumed a large burst caused the maximum delay and the simulation is using nonbursty sources, only a large load would simulate the same effect. Also, the bounded delay assumed that all of the other sources were using their full bandwidth. This is not the case at the beginning of the simulations when the maximum delay values are being captured. The problem with the current simulation structure is that the source is limited by a small output line rate from the source to the shaper. This rate is just larger than 200% of the load. Notice that the leaky bucket/block schedule case reaches a saturation point. We can assume that the other three cases will approach a saturation point which is close to the delay bound at higher loads. All four cases are increasing towards the theoretical boundary as the nonbursty source load increases.

Figures 26 and 27 show the delay penalty incurred by the nonbursty sources when burst crediting is introduced. The maximum delay with burst crediting is normalized to the maximum delay without burst crediting and is compared to a varying credit limit. The figures show that as the credit limit increases, the maximum delay with burst crediting increases for the nonbursty sources. The figure also shows

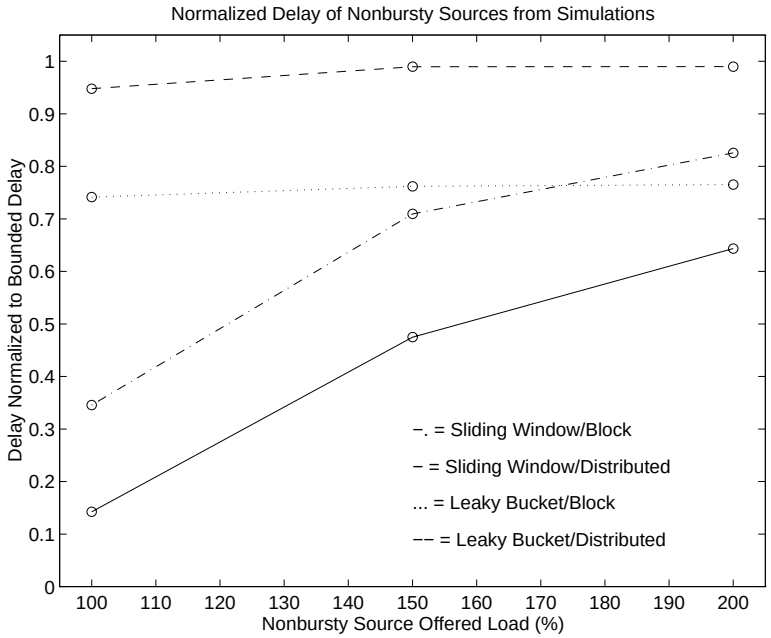

Figure 25: Normalized Delays of Nonbursty Sources from Simulations

that the third term in the equations is insignificant as compared to the first term and is not dependent on *CL*. The offset between the data sets in Figures 26 and 27 results from different maximum delay without burst crediting values.

4 Conclusions

The above work shows a realistic theoretical delay bound using the burst crediting scheme. This initial analysis on a single bursty source system provides sufficient background for further analysis and performance evaluation work in th area of multiple bursty source systems.

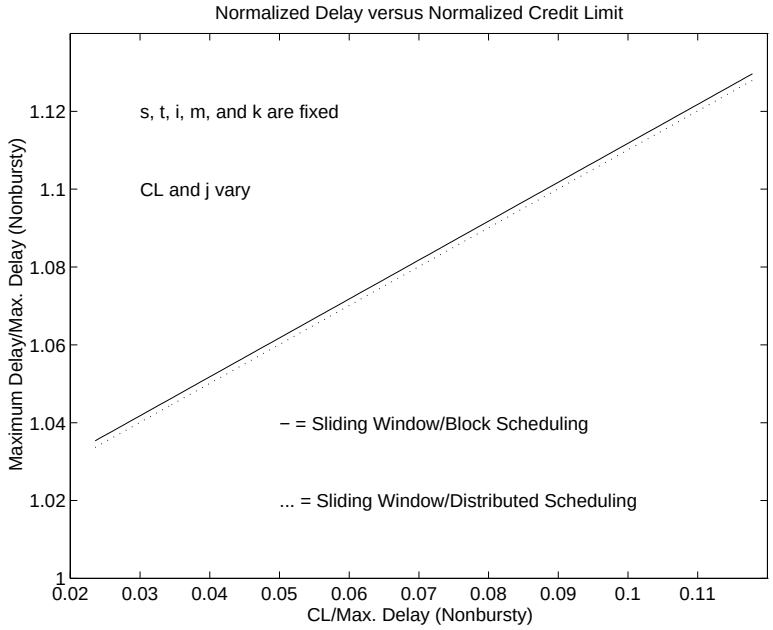

Figure 26: Normalized Delay versus Normalized Credit Limit (Sliding Window)

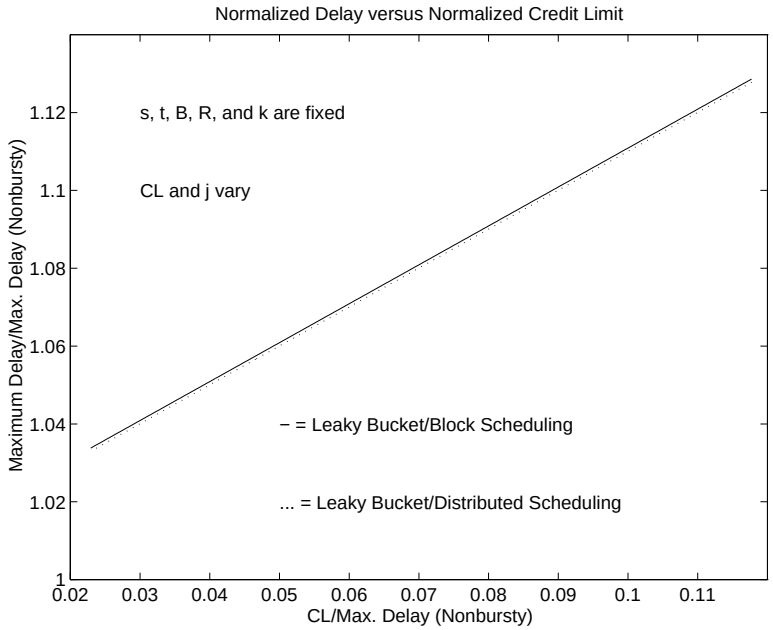

Figure 27: Normalized Delays versus Normalized Credit Limit (Leaky Bucket)

References

[1] Kert Mezger, David W. Petr. *Bounded Delay for Weighted Round Robin*. TISL Technical Report TISL-10230-07.

[2] Kert Mezger, David W. Petr. *The Burst Crediting Concept*. TISL Technical Report TISL-10230-06.

[3] ATM Forum. ATM User-Network Interface Specification, Version 3.0, June 1993.