Queueing Delays in Rate Controlled Networks

Anindo Banerjea

The Tenet Group
Computer Science Division
University of California, Berkeley
and
International Computer Science Institute
Berkeley, California

Srinivasan Keshav

AT&T Bell Laboratories
600 Mountain Ave., Murray Hill NJ 07974

ABSTRACT

This paper addresses the problem of finding the worst case end-to-end delay and buffer occupancy bounds in networks of rate-controlled, non-work conserving servers.

The calculations are based on a simple fluid model, but care is taken so that the computed delay and buffer occupancy values are upper bounds on actual values. A simple algorithm is presented to perform these calculations in linear time.

Simulation results compare the computed worst case delays with the actual delays obtained on some simple network topologies. The algorithm is found to predict node delays well for bursty input traffic, but poorly for smooth input traffic. Buffer requirements are predicted well in both cases.

A. Banerjea was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Hitachi, Ltd., Hitachi America, Ltd., the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.

1. Introduction

Recent work has shown that *framed, non-work-conserving* servers can provide end-to-end delay bounds to users who need strict guarantees on network performance [ZhaKes91]. By *framed* we mean that the servers work on the basis of an interval, called the *frametime*, during which they allocate a number of *transmission slots* to each channel being served. Each slot corresponds to the transmission of a fixed size packet or *cell* on the output trunk. By *non-work-conserving* we mean that given enough cells in the queue, the server will send out exactly the allocated number of cells (the *chunksize*) in each frame, even if this will leave the output trunk idle. If there are fewer than chunksize cells in the queue for the channel, the server will send out the available cells, then use the free slots for non-real time traffic.

One example of such servers are those that obey the Hierarchical Round Robin (HRR) service discipline [KKK90]. HRR servers maintain of a hierarchy of round robin frames, so that a choice of frametimes is available during channel establishment. While earlier work considered the behavior of a single HRR server [KKK 90], and of a channel traversing a sequence of identical servers [ZhaKes91], the general case, where the frame-time could be different at each server, and the input traffic could be bursty, was not analyzed. This paper determines the upper bound on the end-to-end queueing delays and buffer requirements of a simplex channel established over any sequence of HRR servers. We present a novel graphical analysis technique and compare the predicted delay and buffer requirements with simulation results. The analysis extends to a series of Stop-and-Go servers [Gol90] as well.

2. Model of the Network

In our model a channel is associated with a static path through the network and has state and reserved resources at each switch (node) along this path. We focus our attention on the performance of one channel in the network and will analyze the behavior of this channel by induction. That is, given the traffic characteristics at the end of a section of the channel consisting of the first *i-1* nodes (denoted by $channel_{i-1}$) and the node characteristics of $node_i$, we will calculate the characteristics of $channel_i$ (Figure 1). We develop our network model by modeling the input traffic, the nodes, and the traffic as it flows through the network.

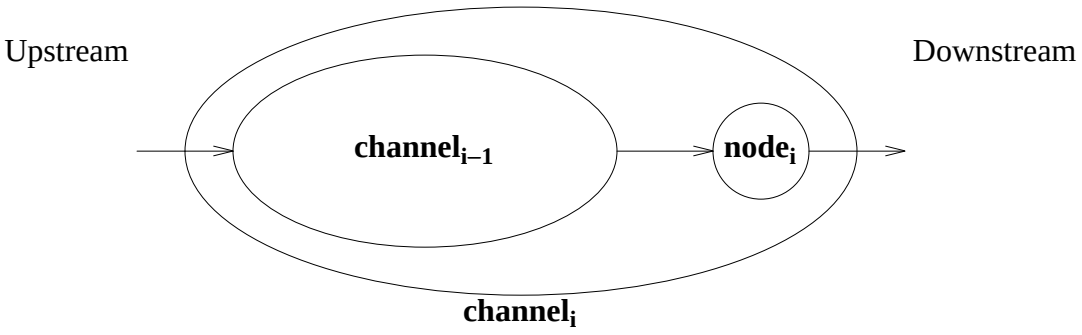

Figure 1. Network Model

2.1. Node Model

For each $node_j$ along the path of the channel, we know the following node characteristics:

a_j *chunksize*, the number of slots allocated to the channel per frame.

F_j *frametime*, the duration of the frame.

B_j reserved rate, the maximum rate at which the node can transmit cells on this channel. ($B_j \equiv a_j/F_j$)

Different channels through a given node will have different values for the above parameters. Since we are concentrating on one channel, we will subscript the parameters just by the node indices.

We assume the physical links between adjacent nodes are acting as constant delay pipes with link delay between $node_{i-1}$ and $node_i$ equal to Δ_i.

2.2. Input Traffic Model

We characterize the traffic entering the channel as suggested in [Fer90, FerVer90]:

$x_{\min}$ The minimum cell inter-arrival time.

x_{ave} The minimum average cell inter-arrival time, over **any** time interval of length *I*.

I The averaging interval for calculating x_{ave}.

Recent work has shown that this model accurately describes most types of rate-controlled traffic expected in high-speed networks [Ver91]. We assume that the user of the channel is bound to obey these restrictions on the input traffic, and the delay and buffer values calculated need only hold if these restrictions are not violated.

2.3. Fluid Approximation

Note that each server is rate-controlled and so will send cells from a channel at its allocated rate until the channel's queue is empty. A simple fluid model, where the output from a server is modeled as a continuous flow at the rate B_j, captures this behavior. This is illustrated in Figure 2, where the X axis shows time elapsed, and the Y axis shows the sequence number of cells sent from $node_j$.

Observe that while the actual traffic is a step function, where one chunksize set of cells are transmitted once every frametime, a fluid model is a straight line approximation to it. Since the cells in each frame can be sent out at any time within the frame, we must assume the worst case and correct for it. To get the maximum number of cells which could have been sent out from $node_j$ by time t, we take value given by the straight line approximation and apply the following function to it.

$$\text{fluid_correction}(y) \equiv \left(\left\lfloor \frac{y}{a_j} \right\rfloor + 1 \right) a_j$$

This gives us the value of the step immediately over the point on the straight line that we are looking at. In terms of Figure 2, it raises the points on the straight line to the dotted line above it. By taking floor and adding one we ensure that for the points of discontinuity in the step function, we take the higher value.

3. Graphical Analysis of Worst Case Behavior

We now introduce a novel technique for analyzing the worst case behavior of a series of rate-controlled servers. Consider the graph of the sequence number of a cell on the output trunk of a node *vs.* time (as in Figure 2). We will use the fluid approximation described earlier. We need only consider the output graph for the worst case arrival pattern at the input, which will repeat every interval *I*, so the graphs are periodic with period *I*, and we consider only one period.

The worst case traffic arrival pattern at $node_i$ is the output graph corresponding to $node_{i-1}$. The service at $node_i$ can be represented by a straight line with slope B_i. The output from $node_i$ is then the composition of the input graph and the service graph. The

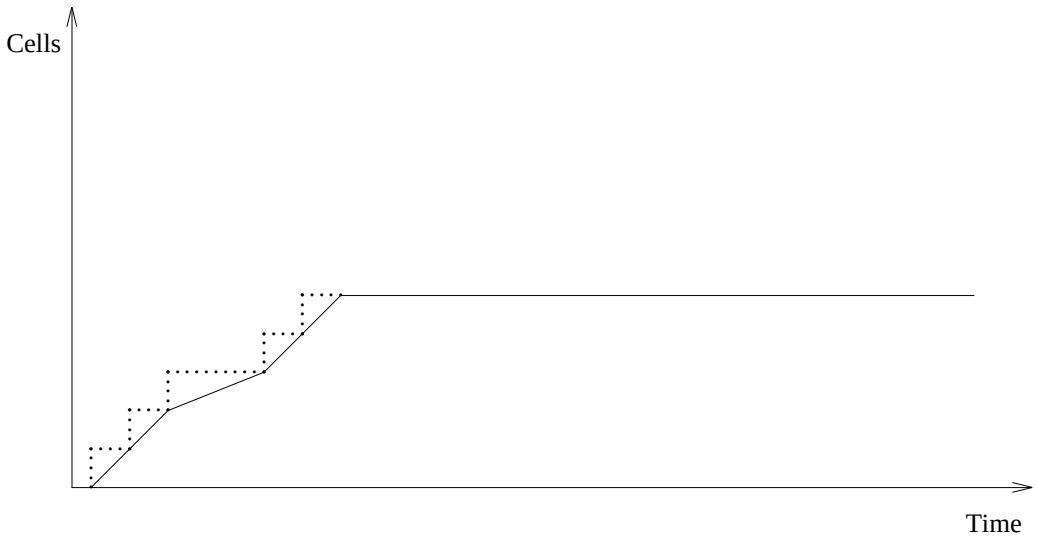

Figure 2. Actual Departure of Traffic vs. Fluid Model

graphical analysis technique can be viewed as a way to combine an input arrival curve and a service curve to produce the output curve for a rate-controlled server. The technique is explained in detail in § 5.

The output graph for $node_i$ encodes information about all the nodes "of interest" up to and including $node_i$. To compute the worst case delay at the next node, we do not need to keep information about all the previous nodes along the path. In the inductive process some nodes cease to be of interest because their effect on the output of the channel is completely masked by a node somewhere downstream of them. For example, if a $node_i$ has a $node_j$ downstream of it, such that $B_j < B_i$, then $node_j$ receives cells from $node_i$ at the higher rate B_i, and transmits them at its own lower rate B_j. So nodes downstream of $node_j$ will not be affected by the output curve of $node_i$ directly.

We use the term *visible* to refer to the set of nodes which cannot be neglected at any given point in the channel. This set changes as we move downstream, since new nodes which come into consideration make some of the earlier ones *invisible*.

We define the *visible envelope* at $node_i$ as the worst case traffic that could come out $channel_i$ (the portion of the channel consisting of the first i nodes) as a function of time. It is represented by the output graph for $node_i$. It can also be thought of as a list of all the nodes *visible* from this point on the channel, together with their reserved rates (B_j) and the period of time they can send data before they must wait from input from some previous node. The visible envelope for $channel_i$ summarizes the information we need to know about the nodes preceding $node_i$. Given this, and the information about $node_{i+1}$, we can calculate the maximum delay and buffer requirements at $node_{i+1}$ as well as the visible envelope for $channel_{i+1}$.

3.1. Example

A simple example of a visible envelope is the description of the input traffic shown in Figure 3. The traffic comes in the maximum possible rate, with inter-cell gaps equal to $x_{\min}$. When I/x_{ave} cells have entered the channel, the input stops because any more cells would violate the average rate restriction on the traffic. (This is often called on-off traffic.)

We can think of this source to consist of one *visible* node, with a reserved rate of $1/x_{\min}$ which is throttled to rate of 0 after $\frac{I * x_{\min}}{x_{ave}}$ time units due to an empty queue.

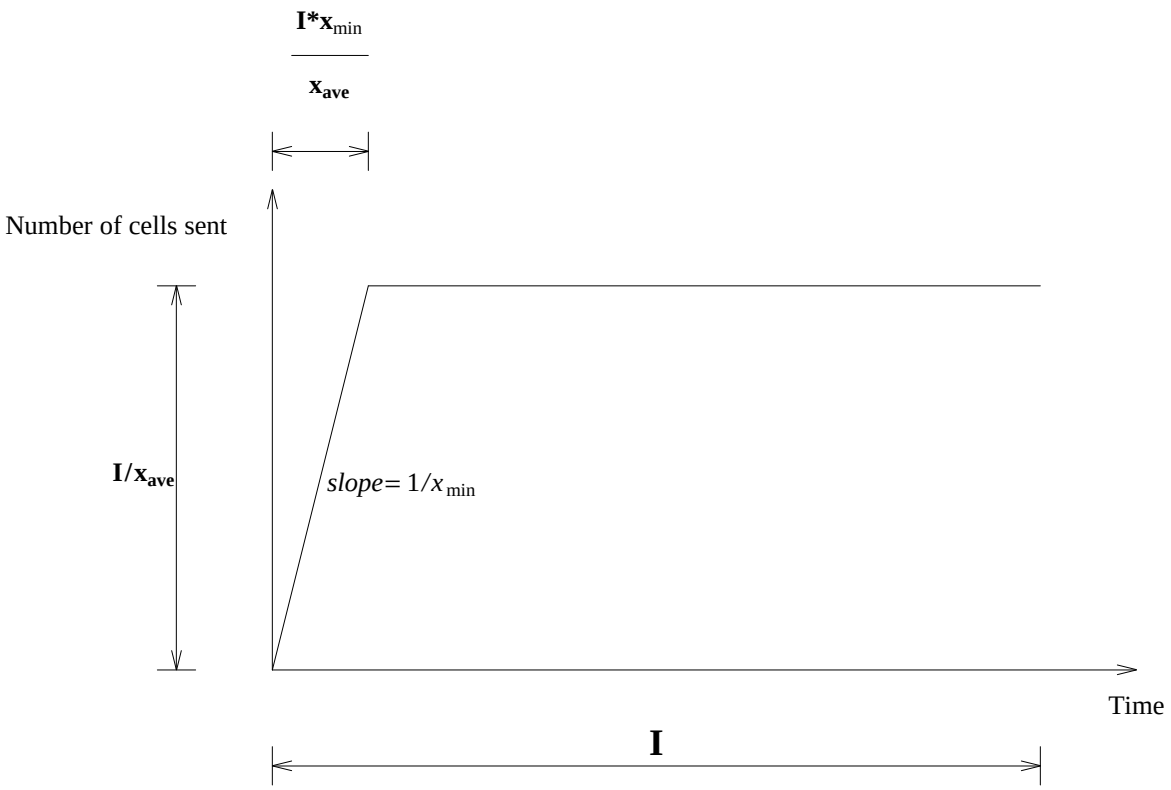

Figure 3. Worst Case Input Pattern

This is the worst case input traffic within the constraints of the input traffic parameters.

The worst case traffic pattern is periodic with period *I*. To see this we only need note that the restriction on x_{ave} is over **any** interval of length *I*. Thus if we have a burst of traffic with inter-cell gaps equal to $x_{\min}$ for a time period $\frac{I * x_{\min}}{x_{ave}}$ then we must have a silent period of length $I - \frac{I * x_{\min}}{x_{ave}}$ immediately following it *and immediately preceding it*. The worst case is to have another burst just before and after the silent period. Thus the sequence of bursts repeat at intervals of length *I*.

3.2. Slippage

After the first cell of the input traffic arrives at an empty queue for the channel at a given $node_i$, a maximum time F_i can elapse before the next frame begins and the channel becomes *active* at $node_i$. By *active* we mean that the service mechanism on $node_i$ is aware of a non-empty queue for the channel, and by the definition of *framed, non-work-conserving* servers $node_i$ must send out the allocated number of cells in each frame from this point in time until the queue becomes empty again. Then the server may put the channel to a non-active status, and the first cell to arrive after that point may observe the phenomenon of *slippage* again.

We use the term *slippage* to refer to the phenomenon and also to this time interval. During this time, cells build up in the input queue at the rate determined by the graph for the input traffic to $node_i$ (which corresponds to the visible envelope of $node_{i-1}$). After this time, the node starts sending out cells at its reserved rate B_i (or a_i cells in every F_i time units) and the visible envelope of $channel_i$ rises above the x-axis. Due to slippage the x-intercept of the visible envelope of $channel_i$ is different from that of the visible envelope of $channel_{i-1}$.

4. Assumptions

The analysis in the following sections makes some assumptions which we shall state and justify in this section. These are:

1. The reserved service rate at each $node_i$ (B_i) is not less than the average input rate ($1/x_{ave}$).
2. The input pattern is the burstiest possible one, given the values of x_{min}, x_{ave} and I.
3. The slippage at each node is the maximum possible, F_i.
4. The effect of cells from previous intervals of time can be ignored in analyzing the current interval of duration I.

The first assumption is required, since otherwise the defaulting node would quickly overflow its buffers.

As the input pattern becomes more bursty, the queueing delays in the system become larger. The second assumption means that we will do the analysis for the worst case possible, given the input traffic specification. The burstiest possible input pattern is shown in Figure 3. As described before it is periodic with period I. In § 7 we rigorously prove that this pattern is indeed the burstiest possible.

Slippage occurs for the first cell sent by a channel. We assume the largest possible slippage at each node. This allows the most initial build-up at this node, and consequently the highest queue build-ups. It also introduces the highest end-to-end delay. Further, this node will send out cells at its maximum rate B_i for the longest possible time (because it had the largest possible queue build-up). This makes the situation the worst for downstream nodes also. Thus we are correct in saying that assuming the largest possible slippage accounts for the worst case.

The fourth assumption is best justified after presenting the details of graphical analysis in § 5. So, this justification is presented in § 7.

With these assumptions in hand, we now present the computation of the visible envelope for a channel.

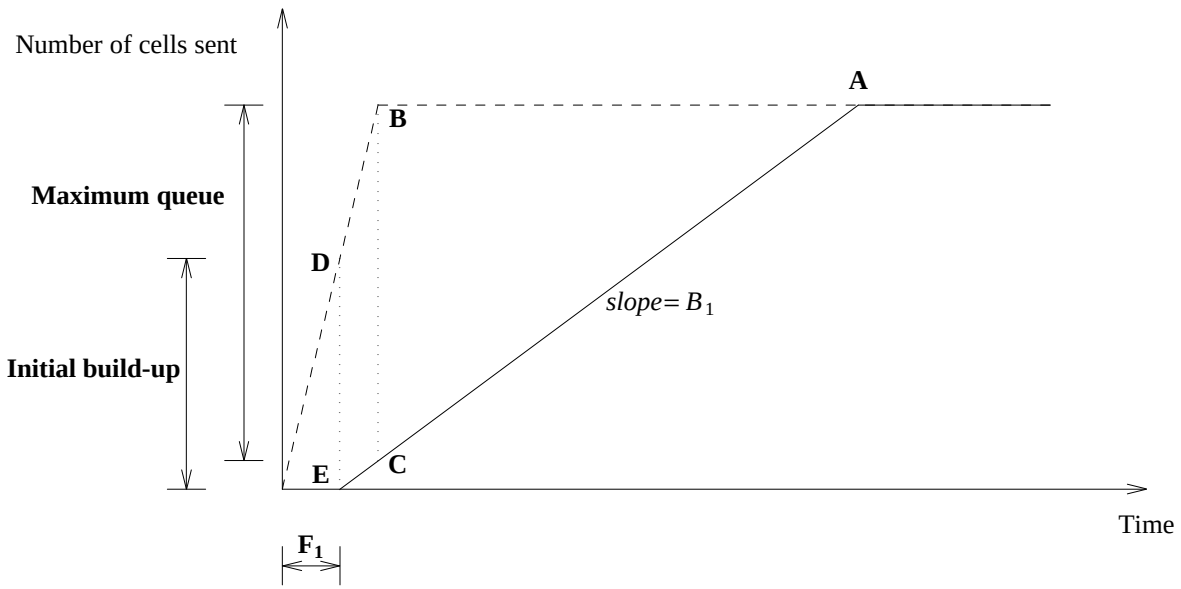

Figure 4. Worst Case Behavior at the Output of the First Node

5. Calculation of the visible envelope

Under the assumptions of the previous section, we shall present the analysis of the queueing delays resulting from the input pattern of Figure 3, as it passes through the nodes in the channel. We first consider the situation at the first node along the path.

5.1. Analysis of the first node

The worst case situation in the first node can be seen in Figure 4. The worst case input pattern is shown as the input to $node_1$ as a dashed curve. After the first cell arrives at the node, a maximum slippage time F_1 passes before the channel becomes active in $node_1$. During this time the cells build up in the queue at the channel's rate of input to the node. Then the node sends out cells at the rate B_1 until its queue becomes empty at point **A**. From this point onwards the output rate from this queue is limited by input rate to this queue. In this case, no more cells are output until the end of the interval *I*. The solid line shows the visible envelope of $channel_1$. It summarizes the worst case traffic on the link downstream of $node_1$.

The queue length at any time is given by the difference between the input and service curves at that point on the time axis. Of course this is under the assumption of a fluid model, and to get the actual number of cells which could have arrived in that time we need to apply the fluid_correction defined in § 2.3 with $j \equiv 0$*. The buffer requirement is simply the maximum possible queue buildup at this node under worst case assumptions, so we need to find the point on the visible envelope for which the above function is maximized. In this case the maximum buffer required at this node is given by fluid_correction(*BC*).

The graph also tells us the worst case queueing delay at this node. There is a very simple relationship between the queuelength of any point on the visible envelope and the delay seen by a cell which arrived at that point. Let us say the queuelength is *queuesize*. Now, in each frame time F_1, a_1 cells are transmitted. So, the worst case time to empty this queue is simply $\left\lceil \frac{queue_size}{a_1} \right\rceil F_1$. The cells which arrive during slippage also suffer an additional delay waiting for the server to become active. The worst slippage is for the first cell from the channel to arrive at the node. For this cell the additional delay is F_1. In general the additional delay of a cell arriving at time *t* after the arrival of the first cell is given by the time to the end of the slippage interval ($F_1 - t$). Cells arriving after the end of the slippage interval suffer no additional delay.

In this example the worst case delay also corresponds to point **B** and the value is $\left\lceil \frac{\text{fluid_correction}(BC)}{a_2} \right\rceil * F_2$. In general if we calculate the delay and buffer values for each point on the visible envelope and take the maximum of each over all the points on the visible envelope we should have the worst case delay and buffer requirements.

5.2. Subsequent Nodes

Subsequent nodes can also be analyzed in the same way. For example the situation for the second node is as shown in Figure 5. Data arrives at $node_2$ as described by the graph for $channel_1$ (the dashed line in Figure 5). No service is received till point **C**, which is the end of the slippage period. After that service is received at the rate B_2 until the queue becomes empty at **A**. Here onwards the number of cells going out is limited by the number of cells coming into the node. Thus the output curve for $channel_2$ follows the visible envelope of $channel_1$. The maximum queue build-up is fluid_correction(*BC*),

* For the purpose of calculating fluid_correction $a_0 \equiv 1$ (and thus $F_0 \equiv x_{\min}$).

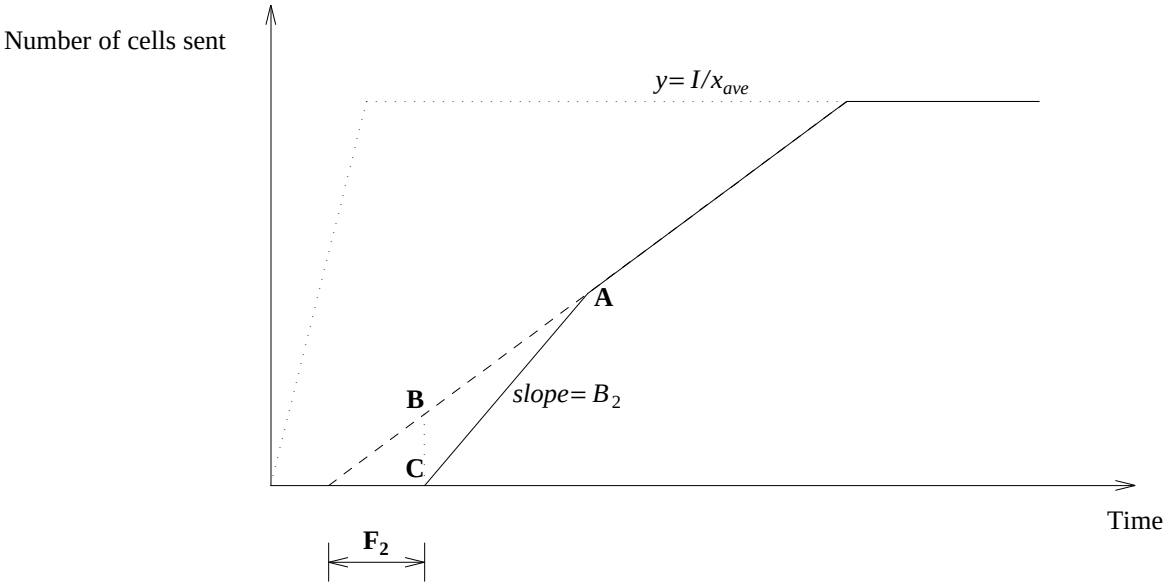

Figure 5. Worst Case Behavior of Second Node

and

the maximum delay is the greater of $\left\lceil \frac{\text{fluid_correction}(\overline{BC})}{a_2} \right\rceil * F_2$ (corresponding to point B) or $\left\lceil \frac{\text{fluid_correction}(0)}{a_2} \right\rceil * F_2 + F_2$ (corresponding to the first cell to arrive). In general, we must compute the delays for all the points separately, and choose the maximum.

Notice that the dotted part of the graph is irrelevant as far as the analysis of $node_2$ goes. The worst case behavior of $channel_1$ (the channel up to and including $node_1$) is summarized by the visible envelope of Figure 4, shown in Figure 5 as a dashed line. The behavior of $channel_2$ is summarized by the visible envelope of Figure 5. This is all that is needed for subsequent nodes, the rest of the information can be discarded.

Notice also that the visible envelope consists, in effect, of a list of all the visible nodes (each segment of the envelope with a constant slope corresponds to a visible node) together with information about the node bandwidth (the slope of the segment) and the length of time the node can transmit at its reserved rate before it must slow down for some previous node.

One point to bear in mind is that the delay over the physical link must also be accounted for while adding up the end to end delays. However, it does not enter the calculation for queue build-ups and queueing delays, if we assume links are behaving as constant delay pipes. The time $node_i$ receives the first cell is Δ_i after the time that $node_{i-1}$ actually sends the cell out, where Δ_i is the link delay over the link from $node_{i-1}$ to $node_i$. However, since the second (and all subsequent) cells also get delayed by the same amount, it does not affect the queue build-ups. In terms of the graphical analysis, the action at the second node occurs shifted to the right on the time axis by Δ_i, but since all events at $node_i$ are shifted identically, we can ignore this effect in calculating the queue build-ups and the queueing delays. We must, of course, add the physical delay in when we are adding up the end to end delay.

6. Formal Algorithm

The visible envelope for $channel_i$ is represented as a list of the points on the envelope ($<x,y>$ coordinate pairs), sorted by y coordinates. The initial envelope describing the worst case input traffic contains three points: (0,0), ($\frac{I\,x_{\min}}{x_{ave}}, \frac{I}{x_{ave}}$), and ($\infty, \frac{I}{x_{ave}}$). To simplify calculations and reduce the number of points we need to carry in the visible envelope, we normalize the envelope on starting the calculations for each node so that the x-intercept of the output envelope is always at the origin. The calculation of the visible envelope for $node_i$ consists of the steps described in the pseudocode of Figure 6.

7. Graphical Justification for Assumptions

Two of the assumptions made in § 4 can be better justified using graphical ideas developed in the preceding discussion. These are:

- The input traffic is the burstiest possible.
- The effect of cells from previous intervals can be ignored.

We first justify our choice of input pattern. In terms of the graphical analysis, we can think of all other input patterns as derivable from the pattern of Figure 3 by shifting the pattern along the x axis, or lowering the envelope of the input pattern by reducing the slope of the input.

Shifting the pattern does not change the behavior of the system because we can shift the origin of the analysis. The restriction on the input traffic, that the average cell interarrival time over *any* interval I must be no less