

Using Feedback Control for a Network and CPU Resource Management Application[†]

Ashvin Goel[¶], Molly H. Shor^ψ, Jonathan Walpole[¶], David Steere[¶], Calton Pu[⊗]

Abstract:

Interactive multimedia and sensor-based real-time applications on the Internet must be delivered to the receiver at the sender's transmission rate and have bounded end-to-end delay requirements. We call such flows *real-rate* flows. The resource requirements of such flows can not be met in many environments. Researchers have proposed selectively dropping data in the flow so that the data rate matches available resource capacity (i.e., *QoS adaptive, real-rate* flows). Application-level QoS adaptation is not well integrated with network scheduling mechanisms and resource management, resulting in slow and inaccurate QoS adaptation, poor flow quality, and unpredictable interactions between adaptive mechanisms.

Our proposed resource management service integrates QoS adaptation and network/CpU scheduling mechanisms to offer support for QoS adaptive, real-rate flows. Its goal is to deliver the highest quality flow to the receiver within a bounded delay at the sender's real rate. This service combines two techniques – both based on feedback – *real-rate scheduling* and *QoS adaptation*. Real-rate scheduling uses packet *time-stamps* to schedule flows along the flow path at the sender's real rate. QoS adaptation drops data at a node when the node's resources are limited, in application-specified *priority* order. System modeling issues are discussed and a controller structure is proposed.

1 Overview, Modeling Issues, and System Architecture

The advent of high-speed networks and processors has led to a shift from desktop computing to a “networked-device” computing environment, where desktop computers are inter-networked with real-time devices such as sensors, cameras and displays. These devices stream data that must be delivered and processed in real-time with bounded end-to-end delay. To achieve this goal, network nodes (such as routers) may manage the flow delay by dynamically adjusting the flow rate. These nodes may also automatically adapt flow quality when resources are fully utilized. A feedback-based resource management approach is well suited for performing these functions.

This work proposes a feedback-based resource manager that ensures real-rate delivery of flows within delay bounds without requiring reservations.

Multimedia and sensor-based real-time applications use a pipeline abstraction and stream information in *real time* from sources to sinks via intermediate stages. These real-rate flows consume CPU, network and other resources at each pipeline stage. (Figure 1)

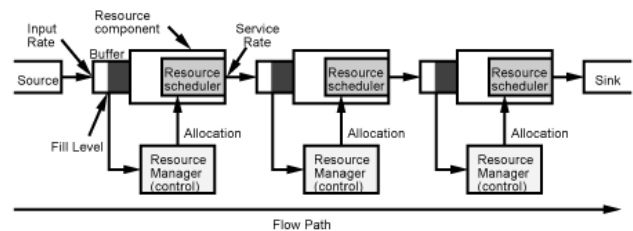


Figure 1. Pipeline configuration

The resources allocated to the flow at each stage of the pipeline are regulated using a feedback mechanism, based on local measurements. Since both length and configuration of pipelines change dynamically, it is desirable to create local resource managers (controllers) along the pipeline that can be cascaded safely. (Figure 2)

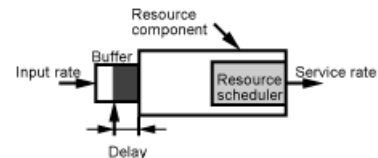


Figure 2. Single stage of pipeline

A means of measuring, locally, how far ahead or behind a flow is from achieving the sender's real rate, is necessary to achieve control. One means of measuring this can be achieved by marking packets with time stamps, with the delivery time offset from first packet in flow. (Figure 3) This information allows the resource manager locally to track the flow's rate and determine how it differs from its rate bound. The resource manager can adjust resources

[†] This work was supported in part by DARPA/ITO under the Information Technology Expeditions, Ubiquitous Computing, Quorum, and PCES programs, and in part by NSF Grant CCR-9988440, NSF Grant ECS-9988435, and by Intel Corporation.

[¶] Dept. of Computer Science and Engineering, Oregon Graduate Institute, Beaverton, Oregon 97921-1000 USA.

^ψ Dept. of Electrical and Computer Engineering, Oregon State University, Corvallis, Oregon 97331-3211 USA.

[⊗] College of Computing, CCB Room 261, Georgia Institute of Technology, Atlanta, Georgia 30332-0280 USA

allocated to each flow based on whether it is running ahead or behind.

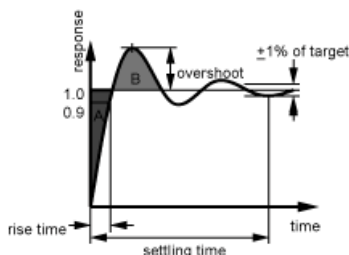


Packets of QoS adaptive, real-rate flows are marked with a logical time-stamp and a priority field. Real-rate scheduler uses time-stamp field to deliver flows at sender's real rate. QoS adaptation mechanism uses priority field to adapt flows by dropping lower priority data when resources are limited.

Figure 3. Packet format for QoS adaptive, real-rate flows

The buffer fill levels can be measured in terms of packet time-stamps. The local resource manager monitors the packet time-stamps and assigns a proportion of the resource (e.g., link bandwidth) over a given period of time to each flow, to adjust a flow's transmission rate (in terms of time-stamps) to match the sender's real rate. The proportion-period (PP) resource scheduler ensures that each flow receives its allocated proportion over that period. The sum of assigned proportions across all flows is the *resource utilization*.

The resources available at each stage are bounded, as are buffer-sizes (both in time and bytes), creating input- and state-variable constraints throughout the system. Large buffers and buffer fill levels result in substantial end-to-end queuing delay, which may violate delay bounds. Consequently, an important design consideration in this system is appropriate sizing of the buffers in the pipeline. Fast rise time and low overshoot of the local feedback control loops help prevent buffers upstream or downstream from a resource manager from overflowing. (Figure 4)



Service rate response of resource manager to step change in input rate of data flow into upstream buffer. Buffer space is required in upstream buffer to hold data in area **A** not processed promptly. Buffer space is required in downstream buffer to hold data in area **B** until resource manager downstream responds.

Figure 4. Effect of rise time and overshoot on buffer fill level variations

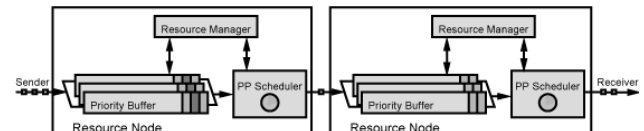
For perfect transmission of data from the sender along the pipeline to the receiver, resource constraints (e.g., bandwidth or CPU processing) may have to be violated, especially in the case of high-bandwidth multimedia flows. Quality of Service (QoS) adaptation approaches adapt the flow rate at the sender, or at an intermediate stage, by selectively dropping data items to ensure that the data rate matches the available resource capacity. Quality gradation can be implemented by assigning priority levels to packets to meet a desired user quality function, then dropping packets with priorities below a current priority-drop threshold. (Figure 5).



Resource manager drops packets with priorities below priority-drop threshold. Mapping from quality to priority-drop threshold should monotonically decrease

Figure 5. A hypothetical mapping of flow quality to prioritized data

Data dropping thins the flow and allows its delay requirements to be met when resources are limited. QoS adaptation adds a second feedback control mechanism to the system at each node. (Figures 6 and 7)

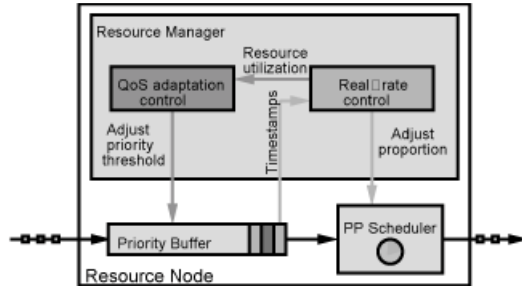


Incoming data is stored in priority buffer before scheduler transmits it. Resource manager implements real-rate and QoS adaptation controllers. The real-rate controller adjusts the scheduler's behavior. The QoS adaptation controller adjusts priority buffer operation.

Figure 6. Architecture of the resource nodes along the flow path

Data dropping based on priority is accomplished using a priority buffer, a bounded circular buffer from which data is dropped in priority order. This scheme limits the physical quantity of data in the buffer (in bytes), the logical time that the data can stay in the buffer (based on time stamps), and the priorities of the packets in the buffer. The logical size limit on the buffer is designed to limit the flow's queuing delay at each resource node. (The logical length of data in the buffer is the difference in time-stamps between the last packet in the buffer and the first packet.) The physical length of the buffer could also

cause data to be dropped. However, a good combination of QoS adaptation controller, real-rate controller, and buffer sizing, will result in data dropped from the buffer based mainly on priority dropping, not random dropping.



Real-rate mechanism monitors packet time-stamps and adjusts flow's proportion so that flow is transmitted at the sender's real rate. The QoS adaptation mechanism monitors resource utilization and adjusts priority drop threshold to maintain 100 percent resource utilization.

Figure 7. The resource node architecture

The sum of the logical sizes of the priority buffers at resource nodes along the flow path is the maximum end-to-end queuing delay experienced by the flows. We size the buffers appropriately to keep this value less than the maximum allowable delay bound, and then strive to use control techniques to keep the buffers from overflowing.

A major challenge for control design is that the amount of resource required to process a logical (or physical) unit of data varies dramatically. We found in our earlier work that the amount of CPU required to decode a MPEG I, P, or B frame varied depending on the type of data processed, by a factor of four up and four down. (Figure 8) The link bandwidth to transmit a physical byte of data varies much less from one type of data to the next. We expect, however, that the resources required to process a *logical* unit of data will vary significantly.

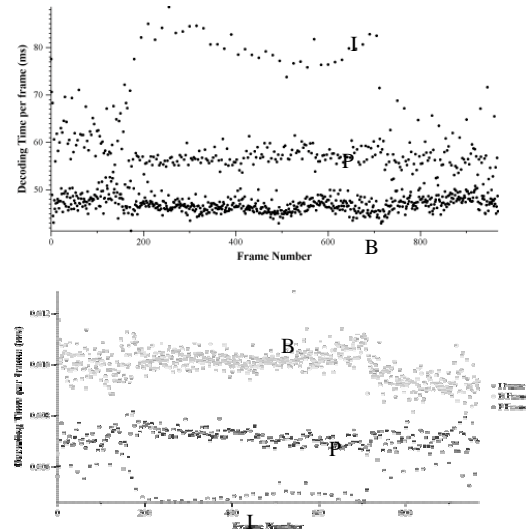
2 Resource Manager Implementation

The resource manager implements real-rate scheduling and QoS adaptation mechanisms by monitoring and adjusting the behavior of PP scheduler and priority buffer. These mechanisms use feedback control techniques. Their behavior should be modeled and analyzed to ensure that the service goals of delivering flows with bounded end-to-end delay at the sender's real rate are met effectively.

2.1 Real-rate control

The real-rate scheduling mechanism aims to transmit data from each resource node at the sender's real rate. To do so, the real-rate mechanism monitors the time-stamps of packets in the priority buffer shown in Figure 7. These time-stamps represent the application logical time. For instance, in a video application, the application logical time progresses at the playback speed of the video. Thus

packets of a frame are marked with the same time-stamp while packets across consecutive frames have a time-stamp interval equal to the inter-frame time. The *real rate* of a flow is defined as the logical time-stamp interval of packets at the head of the priority buffer when monitored one unit time apart. The real-rate mechanism aims to transmit data at a constant unity real rate rather than at a constant bit rate. Thus, packet transmission intervals are matched with packet time-stamp intervals. If this mechanism is used at all nodes along the flow path, data will be delivered to the receiver at the sender's rate.



Decoding time per frame and decoding time per byte were found to vary by factor of at least four up or down depending on the type of data involved.

Figure 8. Decoding time per byte and decoding time per frame for a sample MPEG video

To maintain a target unity real rate (of one unit per unit period), the real-rate mechanism adjusts the flow's transmission rate periodically by modifying the proportion parameter of the PP scheduler (see Figure 7). The real-rate mechanism uses feedback to increase (decrease) the proportion when the monitored real rate is less than (greater than) unity.

2.2 Control Mechanism

The real-rate mechanism uses a discrete-time controller with constant sampling period Δ to maintain a target unity real rate. The rather unusual multiplicative update Equation 2.1 for the controller was motivated in part by the frequent large (multiplicative) changes in the system parameters. (Figure 8)

The controller equation in Equation 2.1 is intended to maintain the unity real rate. The subscript i is the controller's i^{th} sampling instant, the sampling interval is Δ , the proportion p_{i-1} is allocated to the flow between

sampling instants $i-1$ and i , and the proportion p_i is allocated to the flow between sampling instants i and $i+1$.

The time-stamp t_i is on the packet at the *head* of the priority buffer at sampling instant i . Thus, $t_i - t_{i-1}$ is the logical time-stamp interval between packets that are transmitted a real-time interval of Δ apart. The real rate of the flow at sampling instant i is thus $(t_i - t_{i-1})/\Delta$. The error variable x_i is the difference between the current real rate and the target real rate of unity:

$$p_i = \frac{p_{i-1}}{1 + \alpha x_i} \quad \text{where} \quad x_i = \frac{t_i - t_{i-1}}{\Delta} - 1 \quad (2.1)$$

The parameter α in Equation 2.1 satisfies $0 < \alpha < 1$. This real-rate control equation shows that proportion p_i is increased (decreased) by a multiplicative factor relative to previous proportion p_{i-1} when x_i is less than (greater than) zero, or the real rate is less than (greater than) unity. The initial proportion p_0 is chosen to be a small system-defined value greater than zero. Parameter α determines the responsiveness of the controller equation. Section 3 describes how control parameter α can be chosen. Equation 2.1 adjusts flow proportion by comparing the current (short-term) value of real rate to the target unity real rate (similar to a proportion-only controller). This equation is not designed to respond to long term trends in real rate (as a PI controller might). This is not adequate for our application. With this controller, the flow's proportion may return (stabilize) to unity real rate after slowing down temporarily. But the flow would have an increased end-to-end delay after slowing down. That is not acceptable performance.

Equation 2.2 calculates proportion while taking the long term real rate into account. When y_i is less than zero, the flow is being delivered late and the flow's proportion is increased to speed up the flow to "catch up" with the real rate, measured based on the first packet in the flow, rather than just between neighboring packets). If y_i is greater than zero, the flow is being delivered early and the flow's proportion is decreased to slow down the flow to match the real rate. The flow's proportion stabilizes when the short-term and long-term rate mismatch variables x_i and y_i are both zero. The choices of α and β parameters are discussed in Section 3.

$$p_i = \frac{p_{i-1}}{1 + \alpha x_i + \beta y_i} \quad \text{where} \quad (2.2)$$

$$x_i = \frac{t_i - t_{i-1}}{\Delta} - 1 \quad \text{and} \quad y_i = \sum_{j=0}^{i-1} x_j$$

2.3 Control Mechanism during Overload

Equation 2.2 updates a flow's proportion to maintain a target unity real rate. Resource utilization at a resource node is the sum of calculated proportions p_i of all flows. When this calculated resource utilization is greater than available resources, or greater than 100 percent, the resource is overloaded and all flows cannot be assigned their calculated proportions.

The resource manager handles resource overloading with two mechanisms that can operate simultaneously. First, it performs *proportion squishing*, where the real-rate of flows is reduced by assigning proportions that are less than their calculated proportions (similar in effect to saturation in a standard control system). However, this real-rate reduction leads to increasing flow delay as measured by an increase in the priority buffer's logical length. Thus, if overload occurs for an extended period of time, the flow's delay will exceed its delay bound (logical length exceeding logical buffer size). Second, the resource manager uses QoS adaptation to drop lower priority data. Data dropping reduces the resource needs of flows so that a unity real rate can be sustained in the long run and buffers that accumulated data due to proportion squishing can be drained. Section 2.4 describes the QoS adaptation mechanism for dropping lower priority data.

During proportion squishing, the real-rate control assigns proportions to flows that are less than their calculated proportion and sum up to 100 percent. A "fair" method to do this is to assign to flow i the proportion $p_i / \sum p_k$ whenever $\sum p_k > 100\%$. This policy ensures that all flows achieve the same real rate (which is less than unity), but does not allow prioritizing flows. The next section describes a procedure for prioritizing flows using QoS adaptation.

2.4 QoS Adaptation Control

The QoS adaptation mechanism's goal is to transmit the highest quality data within a delay bound at each resource node. The delay bound is the logical size of the priority buffer, which is the maximum amount of queuing delay that can be introduced at the associated resource node (see Section 1). Thus, when the logical sizes are appropriately set at the nodes along the flow path, QoS adaptation aims to deliver flows within their end-to-end delay. To achieve the maximum quality within the delay bound goal, the QoS adaptation mechanism adjusts the quality of the flow

by adjusting the priority buffer's drop threshold variable. Increasing the priority threshold drops lower priority packets and helps in transmitting higher priority packets within the delay bound when resources are limited. Decreasing the priority threshold allows more packets to be transmitted, thus increasing the flow quality.

The goal of QoS adaptation mechanism can be alternately stated as delivery of highest quality data while ensuring the real-rate mechanism can deliver data at its target unity real rate. At unity real rate, flow delay will be bounded (see Equation 2.2); thus these two goals are compatible. The alternate definition leads to an obvious QoS adaptation mechanism. The real-rate mechanism delivers data at less than its target unity real rate during overload, i.e., when resource utilization (the sum of the calculated proportions p_i across all flows) is greater than 100%. The priority threshold should then be increased to reduce the amount of data transmitted and the resource needs of the flow. Conversely, when resources are available, the priority threshold can be decreased to improve flow quality while increasing resource utilization.

2.5 Control Mechanism

The QoS adaptation mechanism monitors the calculated resource utilization and adjusts the priority threshold variable to keep the resource utilization close to 100%, as shown in Figure 7. The QoS adaptation feedback equation for maintaining 100% target utilization is Equation 2.3:

$$pr_i = pr_{i-1} + P(x_i - x_{i-1}) + I x_i$$

$$\text{where } x_i = \left(\sum_{\text{all flows}} p_j \right) - 1 \quad (2.3)$$

In Equation 2.3, i is the discrete-time controller's sampling instant, pr_{i-1} is the flow's priority threshold between sampling instants $i-1$ and i and pr_i is the flow's priority threshold between sampling instants i and $i+1$. Proportion p_j is the proportion allocated to the j^{th} flow. Error variable x_i is the difference between calculated resource utilization $\sum p_j$ and target 100% resource utilization.

Equation 2.3 is the PI control equation used in classic feedback control systems. Parameters P and I determine damping and responsiveness of the control equation. Larger values of P and I increase control responsiveness by adjusting priority threshold faster in response to changes in utilization, thus potentially avoiding buffer overflow. However, then the flow quality varies more over time. In addition, increasing the parameters beyond a limit causes instability in the assignment of the threshold

values. Selection and tuning of these parameters is discussed in Section 3.

3 Tuning Feedback Controls

The goal of QoS adaptive, real-rate service is to transmit the highest quality data within a delay bound at each resource node. This section describes qualitatively how real-rate and QoS adaptation controls are tuned to achieve these goals. Section 3.1 describes real-rate control tuning, and Section 3.2 describes QoS adaptation control tuning.

3.1 Tuning Real-Rate Control

In real-rate control, the measured variable is the real rate of the flow and the performance goal is unity real rate. In QoS adaptation control, the measured variable is resource utilization and the performance goal is 100% utilization.

The real-rate scheduler aims to transmit flows at unity real rate, i.e., packet transmission intervals matched with packet time-stamp intervals. Our goal of control tuning is to tune the real-rate mechanism so its response to step changes in transmission rate meets various criteria such as limited overshoot and limited additional buffering needs.

A step change in input rate may occur due to an increase in the bandwidth needs of the flow. If the proportion allotted to the flow is kept constant, it will take longer to transmit packets and real rate $(t_i - t_{i-1})/\Delta$ will decrease below unity. Equation 2.2 will eventually increase the flow's proportion so that the real rate reaches its unity target. Choice of α and β parameters (Equation 2.2) affects proportion overshoot and additional delay experienced by the flow after a step input. Larger overshoot will cause faster resource overloading downstream than needed, thus causing QoS adaptation to decrease flow quality. Thus the tuning goal is to reduce overshoot. In addition, the additional delay experienced by the head packet in the priority buffer is y_i in Equation 2.2, shown by shaded area **A** in Figure 4. The tuning goal is to reduce this delay so that the priority buffer can handle changes in the real rate. Otherwise, random dropping will significantly decrease flow quality.

Our experience with the system indicates that we can deal with large step changes in the processing requirements, but that there is a delicate balance between selecting good controller gains for step and good controller gains for impulse changes. We are able to determine experimentally the buffering requirements for various controller gains, for both steps and impulses. Future work includes analysis of conditions for closed-loop stability, robustness, and optimization methods for the design of controllers that provide limited delay and buffering requirements and unity rate tracking in the presence of these large changes.

3.2 Tuning QoS Adaptation Control

The QoS adaptation mechanism adapts flow quality by adjusting priority-dropping threshold to keep resource utilization close to 100%. The design and tuning of the PI control in Equation 2.3 depends on the following factors:

Logical Queue Size: A smaller logical queue size will require a more responsive control mechanism to handle variations in the flow's real rate or variations in resource availability.

Change in Resource Needs: The adaptation mechanism must take into account the change in resource needs when the threshold is adjusted. This change directly affects parameters in the system model. If resource utilization increases, the priority threshold may have to be increased by several units if each unit threshold change leads only to a small change in resource need. Conversely, if each change in priority threshold leads to a large change in resource needs, then the granularity, or quantization, of that relationship may lead to behavior typical of systems with friction and other non-smooth nonlinearities. If certain properties of this relationship create fundamental limitations in controller design, the priority marking algorithms can be modified. While we may not be able to use a particular model (based on one application's priority markings) for system-level controller design, it is useful to derive general guidelines for properties of priority threshold marking.

4 Composition of Feedback Controls

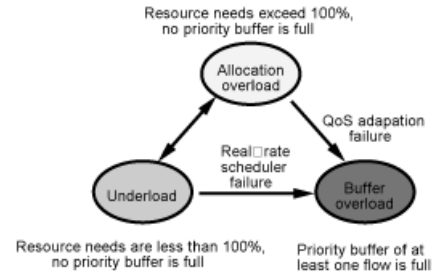
The QoS adaptive, real-rate service is composed of multiple feedback mechanisms that are operating simultaneously. For instance, a separate real-rate control mechanism operates to maintain unity real rate for each flow. In addition, the QoS adaptation mechanism operates across flows to maintain 100% resource utilization.

There are two types of interactions among the feedback mechanisms that implement the QoS adaptive, real-rate service. First, the real-rate control interacts with the QoS adaptation control at each resource node. Second, the controls at each resource node affect the behavior of the rest of the resource nodes along the flow path. These interactions must be analyzed to determine their impact and any lessons for design.

4.1 Real-Rate & QoS Adaptation Control Interaction

The real-rate and QoS adaptation controls operate simultaneously on flows at each resource node and interact with each other. During overload, the real-rate controls for each flow cannot directly assign the calculated proportions. Instead, each flow is assigned a smaller proportion so that all flows achieve the same (less than unity) real rate. This policy during overload increases the delay experienced by all flows. To counter

increased delay, the QoS adaptation policy increases the priority threshold of flows to reduce flow quality and resource utilization. When the resource utilization reaches below 100 percent, flows can be transmitted at unity real rate. However, now the QoS adaptation decreases the priority threshold of flows to increase resource utilization again. This interaction is shown in Figure 9.



In underload, flows are transmitted at unity real rate and QoS adaptation increases flow quality. In allocation overload, flows are transmitted at less than unity real rate and QoS adaptation decreases flow quality. When buffer overload occurs, real-rate or QoS adaptation has failed and data is dropped randomly.

Figure 9. Regions of the resource manager and their definitions

Figure 9 shows that the resource manager can exist in three regions. In *underload* region, resource utilization is less than 100% and the priority buffer of no flow is logically full. Flows in this region are delivered at unity real rate and QoS adaptation increases flow quality by decreasing the priority threshold. In the *allocation overload* region, resources needed are more than 100%. Thus, QoS adaptation decreases flow quality and resource utilization. The *buffer overload* region occurs when the priority buffer of at least one of the flows fills up and the flow's data is dropped randomly. The resource manager reaches this region because either the real-rate or the QoS adaptation controls are not responsive enough to the changing bandwidth needs of flows.

We plan to use different time scales for the two controls. The real-rate control operates in a fast loop and responds to changes in resource needs quickly. The QoS adaptation control operates in a slower loop and updates the system behavior based on observing the performance of the fast loop feedback system. The fast loop control essentially does not depend on the behavior of the slower loop control. This solution allows modeling the two controls in isolation to analyze system behavior.

4.2 Control Composition across Resources

This section describes the feedback interaction among the controls of the resource nodes along the flow path. This interaction occurs because the control response at each node affects the incoming data rate into the priority buffer at the next downstream node. For instance, data is

delivered to the next node slower than unity real rate during the rise time of the response and faster than unity real rate during the overshoot period (see Figure 4). This interaction can affect resource consumption, logical and physical buffer size requirements at downstream nodes. These are each examined in turn.

Resource Consumption The real-rate and the QoS adaptation control mechanisms have been designed so that feedback interaction among the resource nodes does not directly affect resource consumption. The input to the real-rate control is the time-stamp of the packet at the *head* of the priority buffer (see Figure 7 and Section 2.2). A change in the incoming rate does not affect the head of the priority queue. Thus, as long as the priority buffer does not become empty, the real-rate control behavior is unchanged. The case when the priority buffer becomes empty is treated as a boundary condition and the real-rate control leaves the proportion unchanged so that when packets appear in the buffer next time, the flow's previous cached proportion is used. This policy reduces the ramp up time for the real-rate control when packets become available, but wastes resources when packets are not available.

The input to the QoS adaptation mechanism is resource utilization, an output of the real-rate control mechanism. Since the real-rate mechanism is unchanged by the feedback interaction, the input to the adaptation mechanism is also unchanged. Thus the QoS adaptation behavior is also not affected. Since neither of the control mechanisms are affected by the interaction, resource consumption, which depends on these mechanisms, is also not affected. This informal analysis depends on the fact that packets are not randomly dropped from the priority buffer, an issue that is discussed next.

Logical Buffer Requirements The control response at each node (say X) affects the buffer requirements at the next downstream node (say Y). Note that further downstream nodes are not affected directly since the real-rate and QoS adaptation control response at the next node Y is unchanged, as described above. To estimate the amount of additional buffer requirements at node Y due to a step input at the previous node X , we examine the graph in Figure 4. During the rise time, real rate of the flow is less than unity (although more data is being transmitted by node X). Thus logical buffer length at node X increases and length at node Y decreases by shaded area **A** shown in Figure 4. During the time when overshoot occurs, the real rate of the flow is greater than unity. Logical buffer length at node X then decreases and length at node Y increases by shaded area **B** shown in Figure 4.

The worst case additional buffer requirements at node Y due to the control behavior at the node X is the shaded area **B** (assuming that node Y 's priority buffer was empty before the step input at node X). Thus, each node Y must provision for buffering based on its response (shown by area **A**) and the response of the previous node X (shown by area **B**). With this amount of buffering, the real-rate scheduler will be sufficiently responsive that data will not be dropped from the priority buffers randomly, i.e., underload to buffer overload transition in Figure 9 will not happen (however, allocation overload to buffer overload transition can still occur). Note that the control response of each node depends on its control parameters and can be different from the response of other nodes.

5. Conclusion

Although advanced formal control system design methods have not been brought to bear on this computer system design problem, a clear understanding of control issues and modeling issues, and some very interesting designs, have been developed. One interesting feature of the designs is the isolation of the dynamic behavior at one node from the dynamics of the next node (with the exception of sizing of buffers to accommodate variations. This makes the control design problem feasible although the length of the pipelines, an aspect of the system structure and order, is not known *a priori*.

Detailed and patient understanding of the system and design considerations, and the extra freedom to design the system as works best, makes it possible to deal with very complex systems in a simple elegant way.

Future work includes analysis of any shortcomings of the multiplicative control law in 3.2, improved understanding of the interactions between the different controllers, and lessons for the design of good priority marking schemes.

References:

- [1] C. Krasic, J. Walpole. QoS scalability for streamed media delivery. Tech. Report CSE-99-011, Oregon Graduate Institute of Science and Technology, Sept. 1999.
- [2] David Steere, Ashvin Goel, Joshua Gruenberg, Dylan McNamee, Calton Pu and Jonathan Walpole. A feedback-driven proportion allocator for real-rate scheduling. *Proc. Third USENIX Symposium Operating Systems Design Implementation*. USENIX, Feb. 1999.
- [3] Bernet, Y., Blake, S., Grossman, D., and Smith, A. An informal management model for diffserv routers. Work in progress (Internet draft draft-ietf-diffserv-model-04.txt, exp. Jan. 2001), July 2000.
- [4] David Steere, Molly H. Shor, Ashvin Goel, Jonathan Walpole, Calton Pu, Control and modeling issues in computer operating systems: resource management for real-rate computer applications, *Proc. 39th IEEE Conf. Decision Contr.*, Sydney, Australia, Dec 2000.

Reference