To Appear: Proc. IEEE Infocom'95, Boston, April 1995.

A Service with Bounded Degradation in Quality-of-Service Networks *

Jörg Liebeherr *Dongwei Liao*

Department of Computer Science
University of Virginia
Charlottesville, VA 22903

Abstract

Many network applications that require Quality-of-Service (QoS) support, such as transmission of digital voice and video, tolerate a certain level of service degradation. In this study, a novel network service, referred to as service with bounded degradation (BD service), is presented that can take advantage of the delay tolerance of applications. Different from previous proposals for similar services, the BD service can guarantee hard lower bounds of the service degradation. This is achieved by strictly limiting the amount of traffic that is subject to service degradation. To implement a BD service, network traffic is partitioned into components with different delay requirements. Policing of the traffic components is achieved by multi-level leaky buckets. A novel scheduler, referred to as EDD-BD (Earliest Deadline Due - Bounded Degradation) scheduler, performs switching of packets. Properties of the EDD-BD scheduler are presented and analyzed.

1 Introduction

Current computer network technology can provide very high bandwidth up to several hundred megabits per second. The dramatic increase in the capacity of communication links enables the design of packet-switching networks which provide performance guarantees to traffic that is carried by the network. The set of performance guarantees given to a network connection is referred to as *quality-of-service* (QoS). Performance guarantees are specified in terms of network bandwidth, network delay, network delay jitter, and loss probability. We refer to a packet-switching network that can provide quality-of-service to connections as *QoS network*.

In recent years, several researchers have proposed architectures for connection-oriented QoS networks [1, 2, 3]. In all proposals the concept of quality-of-service is quite similar. When requesting a new connection a client submits to the network a specification of its traffic and the requested QoS. The network verifies if the requested QoS can be guaranteed. If the network is not able to support the QoS requirements it rejects the establishment of the connection, otherwise the connection is established. After connection establishment the network must support the admitted QoS until the connection is released.

Since many network clients do not require that service commitments be guaranteed for *all* packets of a connection, a QoS network architecture offers several levels of service commitments. At the highest level, a so-called *deterministic service* [2] guarantees that performance requirements are met for all packets at all times. Since this type of service commitment must support the requested QoS even during possibly rare periods of congestion, the amount of network resources that must be allocated for a deterministic service typically exceeds by far the average resource requirements. At the lowest level of service commitments, the network does not guarantee any QoS requirements at all. This level of service is commonly referred to as *best-effort* service. The network does not reserve any resources for a connection with a best-effort commitment.[1] Also, best-effort traffic can be subject to any form of service degradation, that is, packet delays can be arbitrarily long or packets may be dropped.

Most network clients which require performance guarantees can tolerate a certain level of service degradation. As an example, transmission of digitized voice does not need reliable delivery, but can tolerate a limited number of packet losses. However, the amount of lost voice packets must be within application specific bounds; otherwise, the voice stream will be not intelligible at the receiver [4]. Therefore, for voice applications neither the highest nor the lowest level of

*This work is supported in part by the National Science Foundation under Grant No. NCR-9309224.

[1] Note that the network may, however, reserve certain resources, e.g., link bandwidth, for the class of all best-effort connections.

performance guarantees is ideal. The highest level of performance guarantees does not permit any packet losses at all, and thus, does not allow the network to take advantage of the loss tolerance of voice applications. On the other hand, a best-effort service is not able to support the minimal service requirement for packet voice applications, that is, intelligibility of voice at the receiver.

All architectures for QoS networks proposed in the literature offer a service with commitments that are weaker than a deterministic service, but stronger than a best effort service. The *statistical channel* in the Tenet protocol suite [2] permits network clients to specify probabilistic bounds on the percentage of delayed or lost packets. Probabilistically bounded QoS guarantees, similar to the statistical channel, have been studied in [3, 5]. The architecture proposed in [1] proposes a so-called *predictive service* which adapts QoS guarantees to the load conditions in the network.

A drawback of all existing proposals for QoS service commitments which are neither deterministic, nor best-effort, is that they cannot guarantee hard lower bounds on the degree of service degradation as compared to a deterministic service. Note that any probabilistic service guarantees are by definition not verifiable in a finite time interval. Therefore, a probabilistic QoS guarantee can be satisfied even if the QoS guarantee is violated in a time interval of finite, but otherwise arbitrary, length. On the other hand, a service which adapts the QoS guarantees to the currently prevailing network load may degrade to a *best effort* service without violating any service commitments.

In this study, we propose a new network service that enables a specification of degraded quality-of-service commitments for a fixed portion of traffic from a connection. The advantage of our scheme over previous proposals for similar services is that the degree of service degradation can be verified at all times. Since service degradation is bounded to a specified portion of the traffic we refer to our scheme as *service with bounded degradation* (*BD service*). For the implementation of the new service we require network clients to characterize the amount of traffic which can receive a lower grade of service. Also, a network client must explicitly specify the performance bounds for the portion of traffic that can be subject to service degradation.

There are a number of problems that must be solved to make a service with bounded degradation feasible. One problem is illustrated in the following example. Assume a connection wishes to transmit packetized voice over a QoS network. The voice connection may request a delay bound of 100 *ms* for 90% of its traffic, and a delay bound of 200 *ms* for 10% of its traffic. Naively, the network could separate the traffic stream into two components, and provide delay bounds of 100 *ms* and 200 *ms* to the respective components. In this case, it is feasible that packets from the high delay component will catch up with, and possibly pass, packets from the low delay component. As a result, packets may arrive at the receiver in a different order than they were transmitted. To overcome this out-of-sequence problem we propose an extension to traditional packet scheduling mechanisms which guarantees in sequence delivery of packets, yet, maintains different levels of performance commitments to different portions of the traffic.

The remaining paper is structured as follows. We state our assumptions on the network traffic in Section 2. In Section 3 we present a modified packet scheduler which provides different delay bounds to different portions of the traffic from a single connection, yet, maintains the sequence of the packet stream. In Section 4 we investigate properties of the modified packet scheduler. These properties are used in Section 5 where we obtain analytical results on the BD service. These results are applied in an example in Section 6. In Section 7 we conclude our results.

2 Traffic Model

In a QoS architecture, a network client submits with a request for a new connection (a) a set of parameters which characterize the traffic of the connection, and (b) the required QoS guarantees. Once the connection is established, that is, admission control functions have determined that the performance guarantees can be maintained for the traffic as characterized by the network client, so-called *policing functions* at the entrance to the network enforce that all traffic conforms to the specified characterization.

Throughout this study, we assume that traffic from a connection is characterized by two parameters (b, T) where b denotes the maximum number of packets submitted to the network at any point in time (*burst factor*), and $1/T$ denotes the maximum average rate at which packets are generated (*rate factor*). We assume that packets are fixed-sized.

If a network client requests the establishment of a connection with bounded service degradation (*BD connection*) we require that the client characterizes the portion of traffic which may be subject to service degradation. That is, a client characterizes the connection traffic with two sets of parameters (b_p, T_p) and (b_s, T_s). The sets describe the *primary component* and the *sec-*

ondary component of the traffic, respectively. [2] The secondary component specifies the portion of traffic that may be subject to service degradation. We assume that a multi-level leaky bucket [8] is used for policing the two traffic components. In Figure 1 we illustrate the operations of the policing function. Figure 1 shows two token pools, one for primary tokens (*p-tokens*) and one for secondary tokens (*s-tokens*); *p*-tokens and *s*-tokens are added to the respective pool at a rate of $1/T_p$ and $1/T_s$, respectively. If the *p*-token pool contains b_p tokens, no more tokens are added. Likewise, tokens are added to the *s*-token pool only if it contains less than b_s tokens. Packets waiting for transmission are stored in a packet output queue. Before a packet is admitted to the network it has to draw a token from one of the token pools. [3] If both token pools are empty no packet is admitted to the network. If a packet enters the network by drawing a *p*-token, the packet is tagged as a *p*-packet. If the packet enters the network by drawing an *s*-token, the packet is tagged as an *s*-packet. Note, that we do not permit a client to control the tagging of particular packets.

The maximum traffic that can enter the network from the two traffic components in any time interval of length τ can be described by the following two rate-controlling functions $\mathcal{A}_p^*$ and $\mathcal{A}_s^*$:

$$\mathcal{A}_p^*(\tau) = b_p + \left\lfloor \frac{\tau}{T_p} \right\rfloor \quad \mathcal{A}_s^*(\tau) = b_s + \left\lfloor \frac{\tau}{T_s} \right\rfloor \qquad (1)$$

where $\mathcal{A}_s^*(\tau)$ denotes the maximum amount of traffic which may receive degraded services in a time interval of length τ. The total number of packets that may enter the network in any time interval of length τ is then bounded by $\mathcal{A}^*(\tau) = \mathcal{A}_p^*(\tau) + \mathcal{A}_s^*(\tau)$.

Let us assume that a network client requests BD service for a connection by specifying the traffic parameter set (T_p, T_s, b_p, b_s). With the traffic characterization the connection will also specify two delay bounds d_p and d_s ($d_p < d_s$) which denote the maximum tolerable delay at a node for the respective traffic components. Admission control tests in the network must verify that the specified delay bounds can be supported [2, 7]. That is, the network must be able to determine if it can guarantee a maximum delay of d_p for packets from the primary component, and a maximum delay of d_s for packets from the secondary component. Also, the network must verify that the new connection will not result in violations of delay guarantees for previously admitted connections. Note that the admission control tests for a BD connection are performed separately for each component. In the following, we will assume that all connections that are present in the network have passed the admission control tests.

[2] In principle, one could define an arbitrary number of components. Here, we assume that BD connections only specify two components.

[3] One could impose a rule that a packet only attempts to draw an *s*-token if the *p*-token pool is empty. In this study, we assume that the selection is arbitrary.

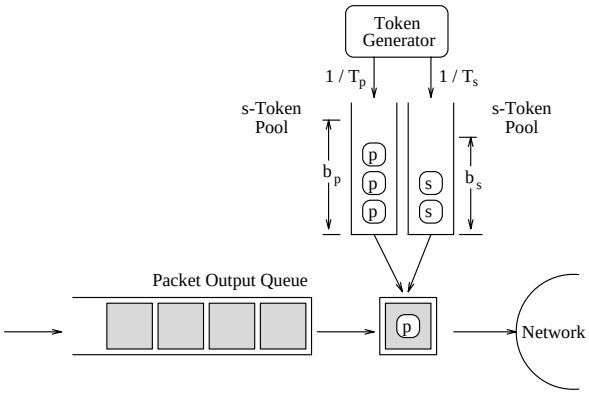


Figure 1: Two-Level Leaky Bucket.

3 Packet Schedulers for a BD Service

Here we show the operation of a packet scheduler in a network node that supports BD connections. We first illustrate that a straightforward implementation of a bounded degradation scheme will result in an out-of-sequence delivery of packets from the same connection. Then we show the design of a packet scheduler that overcomes the misordering problem.

Network traffic from a connection is assumed to traverse a fixed number of nodes on its way from the source to the destination. For each outgoing link each node has a packet scheduler which selects the next packet for transmission. The scheduling discipline of the packet scheduler determines the order in which packets are transmitted. In our study we will assume that the scheduling discipline of packets is *Earliest-Deadline-Due* (EDD), i.e., each packet is assigned a deadline and the scheduler always selects the packet with the lowest deadline for transmission. For the rest of this study we assume that there is only one node on the route from the source to the destination.

Packets arrive to the scheduler as *p*-packets or *s*-packets. Since *p*-packets have a shorter delay bound than *s*-packets, i.e., $d_p < d_s$, the EDD scheduler may cause packets to depart from the node in a different sequence than they arrived in. Consider the example in Figure 2 where we depict the arrival and departure times of three packets from a BD connection at an

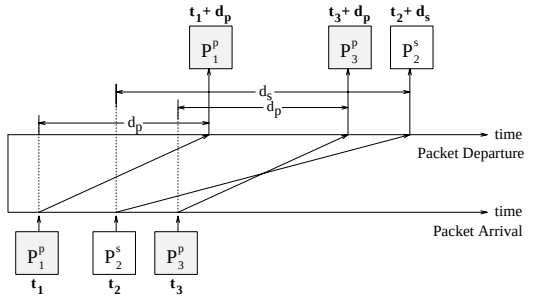


Figure 2: Misordering of Packets in an (ordinary) EDD scheduler.

EDD scheduler. The packets have arrival times t_1, t_2, and t_3. We assume that the first and the third packet, denoted by P_1^p and P_3^p, are tagged as p-packets, and the second packet, P_2^s, is tagged as s-packet. Assuming delay bounds of d_p for p-packets and d_s for s-packets, packet P_1^p obtains a deadline of $t_1 + d_p$, packet P_2^s a deadline of $t_2 + d_s$, and packet P_2^p a deadline of $t_3 + d_p$. If each packet experiences the maximum tolerable delay, all packets will depart at their deadlines. As shown in Figure 2, if $t_3 + d_p < t_2 + d_s$, packet P_3^p will depart from the scheduler before packet P_2^s. As a result, the scheduler has misordered the packet sequence.

Note that an out-of-sequence delivery of packets may result in violations of QoS guarantees for a BD connection even though individual p-packets and s-packets fulfill the delay requirements. In our example, the receiving host of the packets shown in Figure 2 must wait for the arrival of packet P_2^s before packet P_3^p can be submitted to the receiving application. If this reordering delay is considered, packet P_3^p incurs a delay of $d_s - (t_3 - t_2)$ which may violate the specified delay bound d_p.

We propose a solution to the out-of-sequence problem by designing a packet scheduler which separates packets and scheduling information carried by packets. We refer to the new scheduler as a *bounded degradation* (*BD*) scheduler. A BD scheduler can be implemented for any given scheduling discipline; however, we only consider a BD scheduler that uses the EDD scheduling algorithm (*EDD-BD scheduler*).

A BD scheduler has one FIFO queue for each connection into which arriving packet are inserted. Packets in the FIFO queues do not carry any scheduling information. Each BD scheduler has one so-called *token queue*. For each arriving packet the BD scheduler creates a so-called *token* which is inserted into the token queue. A token that is created during the arrival of a particular packet carries three pieces of information. First, the token contains information on the component membership of the packet. If the token was created for a p-packet then the token is marked as *primary token* (*p-token*), otherwise the token is marked as *secondary token* (*s-token*). Second, the token contains (component dependent) scheduling information. If the scheduling discipline is EDD, the scheduling information of a token consists of the deadline of the arrived packet. Third, the token contains the connection identification of the packet for which the token is created. The token does not need to carry any other information. Scheduling of packets in a BD scheduler is exclusively based on tokens in the token queue. An EDD-BD scheduler always selects the token with the earliest deadline. If a token is selected, the first packet from the FIFO queue that is identified by the selected token is transmitted. The transmitted packet is tagged with the scheduling information of the selected token and the token is destroyed.

Note that for connections which are not BD connections, the BD scheduler will not be different from an ordinary scheduler. For BD connections, however, the BD scheduler may alter the component membership, i.e., a packet that arrived to the BD scheduler as a p-packet may depart as an s-packet and vice versa.

We will give an example to illustrate the operations of a BD scheduler which is based on the Earliest-Deadline-Due algorithm (*EDD-BD scheduler*). The EDD-BD scheduler is shown in the center of Figure 3. We assume that the EDD-BD packet scheduler currently serves only three connections $j = 1, 2, 3$. Here, we ignore the details of connections 2 and 3. For the BD connection 1, we assume that p-packets are assigned a delay bound d_p and s-packets are assigned a delay bound of d_s.

The BD scheduler shown in Figure 3(b) contains three FIFO packet queues, one for each connection, and one *token queue*. In Figure 3 we represent packets by squares, and tokens by ovals. Packets in the FIFO queues of the BD scheduler are labeled by a two-digit number, where the first digit is a connection identifier, and the second digit is a sequence number. The tokens in the token queue are labeled with the deadline (= arrival time + delay bound) of the packet for which the token was created, a connection identifier (in Figure 3(b), we include an identification of the packet that created the token), and the component membership. The component membership of tokens is indicated by the shade of the ovals; a token with a dark shade indicates a p-token, and a token with a light shade indicates an s-token. The tokens are assumed to be ordered according to deadlines, and the token with the earliest deadline is in the first position of the token queue. The following figure summarizes the labeling of

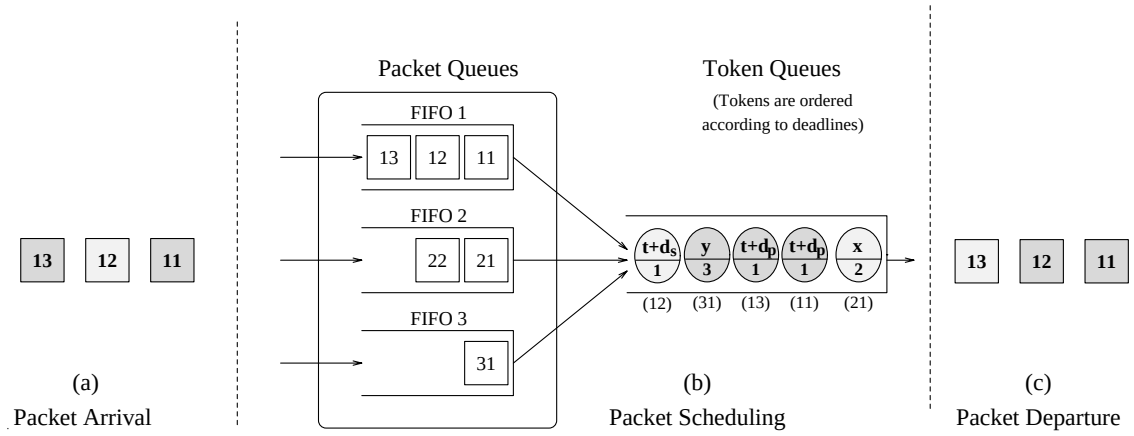
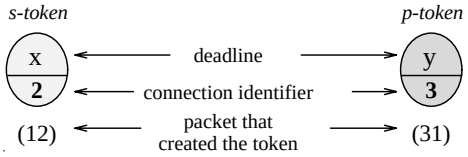


Figure 3: Design of a Bounded Degradation Scheduler (*EDD-BD* scheduler).

tokens in Figure 3:



We consider the arrival of three packets from the BD connection 1, labeled '11', '12', and '13', that arrive to the scheduler at time t as shown in Figure 3(a). Packets '11' and '13' are assumed to be p-packets, and packet '12' is assumed to be an s-packet. We assume that prior to the arrival of the packets from connection 1 the scheduler holds one packet from connection 2, labeled as packet '21', and one packet from connection 3, labeled as packet '31'. The tokens that were generated during the arrivals of these packets have deadlines x and y for packets '21' and '31', respectively, with $x < y$.

Upon arrival of the packets from connection 1 at time t (Figure 3(a)), two p-tokens with deadline '$t+d_p$' are generated for p-packets '11' and '13', and one s-token with label '$t+d_s$' is created for the s-packet '12'. The tokens are placed in the correct position of the token queues (We assume that $x < t+d_p < y < t+d_s$). The packets are inserted into the FIFO packet queue for connection 1. Then the state of packet queues and token queues is as shown in Figure 3(b). The EDD-BD scheduler selects tokens in the order of their deadlines. From Figure 3 it can be quickly verified that packets will depart in the following order: '21', '11','12','31', '13'. When the scheduler selects the second token with deadline $t+d_s$, i.e., the token created upon the arrival of packet '13', packet '12' is in the first position of the FIFO queue of connection 1. Thus, packet '12' will be transmitted. Since the selected token was created for a p-packet from connection 1, packet '12' is transmitted as a p-packet. When the token with deadline '$t+d_s$' is selected, packet '13' is transmitted as an s-packet. As shown in Figure 3(c) the packets from connection 1 depart in the same sequence as they arrived. However, the component labeling of the packets has changed.

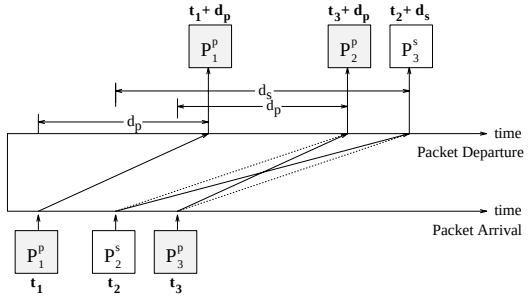


Figure 4: Order of Packets is maintained in an EDD-BD scheduler.

Next we review the example from Figure 2 for an EDD-BD scheduler. Recall from Figure 2 that the (ordinary) EDD scheduler misordered the sequence of packets. In Figure 4 we show the same arrival sequence of packets as in in Figure 2, i.e., packets P_1^p, P_2^s and P_3^p arrive at times t_1, t_2, and t_3. For each packet arrival a token is generated. The tokens are assigned deadlines t_1+d_p, t_2+d_s, and t_3+d_p. As in Figure 2, we assume $t_1+d_p < t_3+d_p < t_2+d_s$. Assuming that each token experiences its maximum tolerable delay, the tokens will be selected at their deadlines. Since the EDD-BD scheduler always selects packets from the same connection in FIFO order, the first packet departs at t_1+d_p, the second packet at t_3+d_p, and the third packet at t_2+d_s. However, since the EDD-BD tags a departing packet with the component identifier of the token that was selected upon the packet's departure, the second packet departs from the scheduler as a p-packet and the third packet departs as an s-packet. Thus, the second and the third packets will depart with different component identifiers than they arrived with.

Although the EDD-BD scheduler maintains the sequence of packets, it may incur extra delays for packets from the primary component. For example, note that in Figure 4 the second packet, which arrived as an s-packet, but departed as a p-packet, stays in the scheduler longer than d_p. Also, the delay of the third packet which departs as an s-packet is lower than d_s but longer than d_p. Therefore, the relabeling of component membership in an EDD-BD scheduler maintains the order of packets, yet, may result in delay bound violations for some packets.

4 Scheduling Conflicts

In the example in Figure 2 we observed the dilemma that scheduling packets in the order of deadlines results in a misordering of packets, and enforcing in-sequence delivery of packets results in a deadline violation for some packet. We refer to such a situation as a *scheduling conflict*. The EDD-BD resolves scheduling conflicts by exchanging the component identification of packets (see Figure 4). Note that exchanges of component memberships only occur during scheduling conflicts.

For a formal investigation of scheduling conflicts in EDD-BD schedulers we will introduce the following notation. We have informally used some of the notation in the previous section.

- P_i denotes the ith packet from a BD connection. We use P_i^p and P_i^s to denote that the ith packet is a primary (P_i^p) or a secondary packet (P_i^s).
- T_i denotes the token created for packet P_i. T_i^p indicates that the ith token is a p-token. T_i^s indicates that the ith token is an s-token.
- The arrival and departure times of the ith packet are denoted by $A(P_i)$ and $D(P_i)$, respectively; $d(P_i) = D(P_i) - A(P_i)$ denotes the delay of packet P_i.
- The creation time of the ith token is denoted by $A(T_i)$. The time when token T_i is destroyed is denoted by $D(T_i)$; $d(T_i) = D(T_i) - A(T_i)$ denotes the delay of token T_i.

Since a token is created every time a packet arrives we have $A(P_i) = A(T_i)$. Furthermore, since the component identification of a created token is the same as the component membership of the arriving packet we have:

$$A(P_i^p) = A(T_i^p) \qquad \text{or} \qquad A(P_i^s) = A(T_i^s)$$

Note that in the EDD-BD scheduler, the number of packets and the number of tokens are always equal. Since the EDD-BD scheduler handles packets and tokens differently once the packet has arrived, the departure time of the ith packet and the ith token may not be the same, i.e., $D(P_i) \neq D(T_i)$ is feasible. Since the BD scheduler destroys one token for each packet departure and transmits packets from one connection in a FIFO order, we obtain the following relationship between the destruction time of a token and the departure time of a packet.

$$D(T_i) = D(P_j) \iff$$
$$A(P_j) = \min_k \{A(P_k) \mid A(P_k) < D(T_i) \leq D(P_k)\}$$

In other words, the jth packet departs with the ith token if the jth packet arrived earlier than all packets that are present in the scheduler at $D(T_i)$.

With the notation at hand, we can now formally describe scheduling conflicts.

Definition 1 *(a) Two packets P_i and P_j are said to directly conflict with each other if $A(T_i) < A(T_j)$ and $D(T_i) > D(T_j)$, or $A(T_j) < A(T_i)$ and $D(T_j) > D(T_i)$.*

(b) Two packets P_i and P_j ($i < j$) are said to (indirectly) conflict with each other if there exists a sequence of packet pairs (P_i, P_{k_1}), (P_{k_1}, P_{k_2}), ..., (P_{k_m}, P_j), such that packets in each pair directly conflict with each other.

(c) A group of packets is said to form a conflict group if each packet in the group conflicts with all other packets in the group, and no packet in the conflict group conflicts with any packet not in the group.

Note that for any two packets a direct conflict implies an (indirect) conflict. Also note that two packets from the same component cannot directly conflict with each other. The conflict property is transitive, that is, if packet P_i conflicts with P_j, and packet P_j conflicts with P_k, then packet P_i also conflicts with P_k.

5 'Adjusting' Delay Guarantees in the BD Service

In an EDD-BD scheduler, the maximum delays of p-packets in a conflict group can be larger than d_p. In this section we show how to adjust the delay guarantees of p-packets in a BD connection to account for the additional delays in a conflict group.

We use C_k to denote a conflict group with k p-packets. We denote by $m_s^*(k)$ the maximum number of s-packets that can be in any conflict group C_k with k p-packets. We use $d^*(C_k)$ to denote the maximum

delay of the packet in a conflict group C_k that has the smallest delay ("fastest packet") of all packets in the conflict group. We define $B^*(t, C_k)$ to be a function that gives the maximum number of p-packets which can be in conflict groups C_j $(j \leq k)$ in any time interval of length t.

The following results for $m_s^*(k)$, $d^*(C_k)$, and $B^*(t, C_k)$ were proven in [6] for a BD connection that conforms to the traffic model from Section 2 with rate-controlling functions $\mathcal{A}_p^*$ and $\mathcal{A}_s^*$ from equations (1), and delay bounds d_p and d_s.

Lemma 1 *The maximum number of s-packets in a conflict with k p-packets $(k \geq 1)$ is given by:*

$$m_s^*(k) = k \cdot \left\lceil \frac{d_s - d_p}{T_s} \right\rceil - k + b_s$$

Theorem 1 *The maximum delay of the fastest packet in a conflict group C_k, $d^*(C_k)$, has an upper bound of*

$$d^*(C_k) \leq d(C_k) := \min\{d_s - \frac{m_s^*(k) - b_s}{m_s^*(k) - 1} T_s, \; (1 - \frac{1}{k}) \cdot (d_p + (\lceil \frac{d_s - d_p}{T_s} \rceil - 1) \cdot T_s) + \frac{d_s}{k}\}$$

Theorem 2 *The maximum number of p-packets from a BD connection that can be in conflict groups C_j with $j \leq k$ in any time interval of length t is bounded above by:*

$$B^*(t, C_k) \leq B(t, C_k) := k \cdot \lceil \frac{t}{T_s \cdot m_s^*(k)} \rceil$$

With $d^*(C_k)$ and $B^*(t, C_k)$, we can adjust the traffic and the delay guarantees of the primary and secondary traffic components by selecting a value of $k \geq 1$, and by setting:

$$\begin{aligned} \mathcal{A}_p^{(k)}(t) &:= \mathcal{A}_p(t) - B(t, C_k) \\ \mathcal{A}_s^{(k)}(t) &:= \mathcal{A}_s(t) + B(t, C_k) \\ d_p^{(k)} &:= d^*(C_k) \\ d_s^{(k)} &:= d_s \end{aligned}$$

$\mathcal{A}_p^{(k)}$ and $\mathcal{A}_s^{(k)}$ are the adjusted traffic components and $d_p^{(k)}$ and $d_s^{(k)}$ are the adjusted delay bounds. With Theorems 1 and 2 it is ensured that for each k and for any time interval of length t, the BD scheduler can guarantee a delay bound of $d_p^{(k)}$ for at least $\mathcal{A}_p(t) - B(t, C_k)$ packets. Note that per our assumption, no packet from the BD connection has a delay that exceeds d_s, and therefore, delay bound $d_s^{(k)}$ holds in particular for all packets in $\mathcal{A}_s^{(k)}(t)$.

Since $d(C_k)$ is decreasing in k, and since $B(t, C_k)$ is increasing in k, the selection of k represents the following tradeoff. For a low value of k, the difference of the delay bounds between the primary and secondary traffic components may be small, i.e., $d_p^{(k)} \approx d_s$. However, a small value of k ensures that $B(t, C_k)$, the amount of traffic from the primary component that is 'lost' to the secondary component, is small. On the other hand, if k is selected large, then $d_p^{(k)} \approx d_p$, but the 'remaining' traffic from the primary component as given by $\mathcal{A}_p^{(k)}(t)$ will be small.

6 Numerical Example

In this section, we present an example that illustrates the readjustment of delay guarantees from a BD connection at an EDD-BD scheduler. The parameters of the BD connection are given in Table 1. The primary and secondary traffic components of the BD connection are described by (T_p, T_s, b_p, b_s) and (d_p, d_s).

In Figure 5 we show the values of $d(C_k)$ as obtained from equation (2) for different settings of the parameter T_s. With Table 1 the delay bounds of p-tokens and s-tokens are 100 *ms* and 500 *ms*, respectively. Note that $d(C_k)$ decreases for increasing values of k. That is, the more p-packets a conflict group contains, the lower the maximum delay of the fastest packet in this group. Note that the rate of decrease of $d(C_k)$ approaches zero as k grows large. From Figure 5 we also see that the delay bound of the fastest packet is lower if the amount of secondary packets is decreased (by increasing T_s). For any practical considerations we observe that for $T_s \leq 200$ the value for $d(C_k)$ is close to the delay bound d_s. Therefore, one should select the frequency of s-packets to be $T_s > 200$. Also, the value of k should be chosen such that $k > 8$. Otherwise, the adjusted delay bound $d_p^{(k)}$ will be close to d_s.

Figure 6 depicts the values of function $B(t, C_k)$ when varied over time for different values of k. In Figure 6 we compare the functions $B(t, C_k)$ against the maximum number of p-packets, as given by $\mathcal{A}_p^*(t)$. Note that for small values of k, $B(t, C_k)$ is small compared to the total number of p-packets, $\mathcal{A}_p^*(t)$. Due to the pessimistic estimate of $B(t, C_k)$, we see that for large values of k, $B(t, C_k)$ can take values that exceed those of $\mathcal{A}_p^*(t)$. In these cases, the BD service will not guarantee a delay bound less than d_s for any packet.

For the selected example, a selection of $k = 8$ ap-

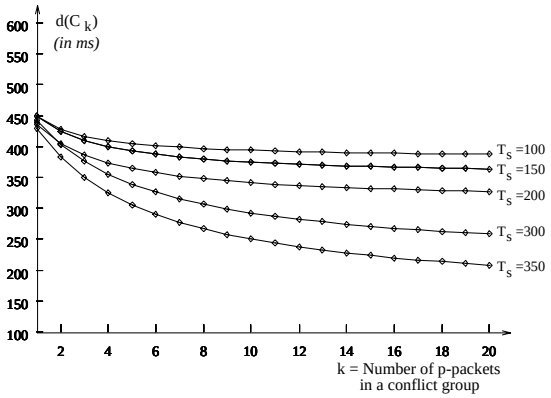


Figure 5: $d(C_k)$ in Conflict Groups C_k $(k \geq 1)$.

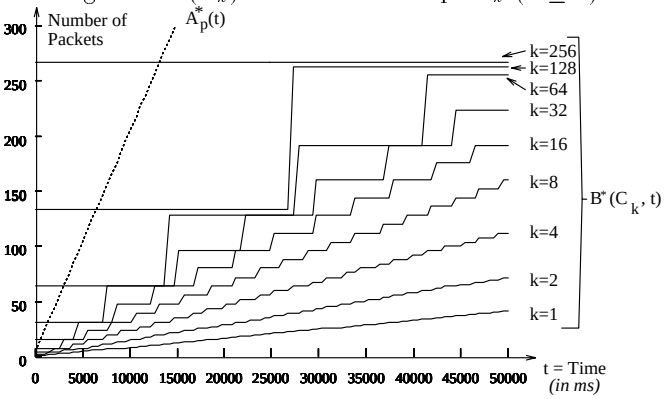


Figure 6: $B(t, C_k)$ in Conflict Groups C_k $(k \geq 1)$.

pears to be a good choice. In this case, the delay bound for primary packets can be given with Figure 5 as $d_p^{(k)} = 275\ ms$. From Figure 6 we see that for $k = 8$ the number of lost p-packets is small compared to the total number of p-packets.

Parameter	T_p	b_p	d_p
Value	50 ms	5	100 ms
Parameter	T_s	b_s	d_s
Value	50-250 ms	5	500 ms

Table 1: Parameters of BD Connection.

7 Conclusions

In this paper, we proposed a network service that can specify different quality-of-service guarantees to different parts of the traffic. The service was referred to as service with bounded degradation (BD service). Traffic from a connection that receives BD service is partitioned into two components, a primary component and a secondary component. Traffic from each component is characterized separately using a leaky bucket scheme. We showed that due to the different delay bounds of packets from the primary and the secondary component, a naive implementation of the BD service may result in an out-of-sequence delivery of packets. We proposed a novel scheduler, referred to as EDD-BD scheduler, which maintains the order of transmitted packets regardless of their membership in a particular component. We showed properties of the EDD-BD scheduler and investigated the effects of the EDD-BD scheduler on the delay of packets from the primary component.

References

[1] D. D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms. In *Proc. Sigcomm '92*, pages 14–26, Baltimore, August 1992.

[2] D. Ferrari and D. C. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, April 1990.

[3] J. M. Hyman, A. A. Lazar, and G. Pacifici. Real-Time Scheduling with Quality of Service Constraints. *IEEE Journal on Selected Areas in Communications*, 9(7):1052–1063, September 1991.

[4] N.S. Jayant and S.W. Christensen. Effects of Packet Losses in Waveform Coded Speech and Improvements Due to Odd-Even Sample-Interpolation Procedure. *IEEE Transactions on Communications*, COM-29:101–109, February 1981.

[5] J. F. Kurose. On Computing Per-Session Performance Bounds in High-Speed Multi-Hop Computer Networks. In *Proc. 1992 ACM Sigmetrics and Performance '92*, pages 128–139, June 1992.

[6] D. Liao. Quality of Service with Bounded Degradation. Master's Project, University of Virginia, Department of Computer Science, December 1993.

[7] J. Liebeherr, D. E. Wrege, and D. Ferrari. Admission Control in Networks with Bounded Delay Services. Technical Report CS-94-30, Department of Computer Science, Universiy of Virginia, July 1994.

[8] J. S. Turner. New Directions in Communications (or Which Way to the Information Age?). *IEEE Communications Magazine*, 25(8):8–15, October 1986.