A shorter version of this paper is to appear in the Proc. 8th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), San Jose, California, September 2002

Supporting a Low Delay Best-Effort Class in the Presence of Real-Time Traffic

Samarjit Chakraborty[*] Matthias Gries[†] Lothar Thiele[*]

Abstract

There exists a large volume of literature in the real-time systems area, on integrating *soft* real-time (or best-effort) tasks into a hard real-time environment. However, all the work in this area pertains to the processor scheduling domain. Current practise in packet scheduling is still based on the straight-forward approach of serving best-effort packets only when no real-time packets are present at the scheduler. The problem of providing an improved service to best-effort flows has not been adequately addressed by packet scheduling research.

Motivated by the work in the real-time systems area, in this paper we propose a scheduler to provide a low delay service to best-effort packets without jeopardizing the delay bounds associated with real-time flows. We also show that some of the well known service mechanisms such as the Total Bandwidth Server of Spuri and Buttazzo turn out to be special cases of our scheme. Our experiments with realistic traffic, consisting of a mix of real-time flows such as voice and video and several best-effort flows show an average improvement of upto 45% in the delay experienced by the best-effort flows, without any real-time flows missing their deadlines.

1 Introduction

The problem of integrating jobs with *soft* deadlines into a hard real-time environment has been well studied in the real-time systems area (see [1] and [2] and the references therein). The motivation behind this is that multimedia tasks such as audio and video require some form of quality of service (QoS) guarantees, but their worst case execution times can vary widely. Therefore, treating them as *hard* real-time tasks would result in wasting a lot of processing resources, and underestimating their execution requirement might jeopardize other real-time tasks. Treating them as soft real-time tasks might result in some of their deadlines getting missed, but this usually does not adversely affect their quality and on the other hand improves the overall system utilization. If d is the deadline assigned to such a job and f is its finishing time, then the goal here is to minimize the mean tardiness ($\max\{0, f - d\}$) of all such jobs, without any of the hard real-time jobs missing their deadlines. Alternatively, when such jobs do not have explicit deadlines associated with them, the goal is to minimize their average response time.

All the work on this problem, however, pertains to the processor scheduling domain, and the equivalent problem in the packet scheduling domain has remained largely ignored. In the packet scheduling area, there has been an extensive amount of work on advanced buffer management and scheduling algorithms to provide QoS guarantees to real-time continuous media traffic. But relatively

[*]Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology (ETH) Zürich, Gloriastrasse 35, CH-8092 Zürich, Switzerland, E-mail: {samarjit, thiele}@tik.ee.ethz.ch

[†]Department of Electrical Engineering and Computer Sciences, 231 Cory Hall, University of California, Berkeley, CA 94720-1770, USA, E-mail: gries@eecs.berkeley.edu

little has been done to exploit these algorithms to better support best-effort traffic. In the presence of a mix of real-time and best-effort traffic, the most widely followed scheme blindly gives higher priority to the real-time packets and serves the best-effort packets only if no real-time packets are present at the scheduler.

Motivated by the work done in the real-time systems area, in this paper we attempt to address the above shortcoming by proposing a packet scheduling algorithm to provide a low delay service to best-effort packets without violating any of the delays associated with the real-time flows. Apart from improving the delays experienced by best-effort flows in general, the need for such a *low delay best-effort* service class arises in the context of serving special packets carrying, for instance, network control information such as routing table updates. Another example belonging to this class would be sporadic http requests. Since the number packets belonging to such classes are usually very small, the overhead involved in designating a distinct flow for these packets and explicitly reserving a network bandwidth to serve them is very high. As a result, such packets are treated as best-effort packets but at the same time they have a short *time-to-live*. Our proposed scheduler although continues to treat such packets as best-effort packets, would guarantee that they are delivered as early as possible without jeopardizing the deadlines associated with real-time packets.

The problem, our results and relation to previous work. The problem of integrating soft real-time or best-effort tasks into a hard real-time environment has very different manifestations and concerns in the context of processor scheduling and packet scheduling. Most of the real-time systems literature in the context of processor scheduling assume the hard real-time tasks to be periodic, with a specified period and a worst case execution requirement within each period. In several approaches based on executing the periodic tasks using a Rate Monotonic scheduling algorithm, the best-effort tasks are served using *server mechanisms* such as the Priority Exchange Server [3], the Sporadic Server [4], and the Deferrable Server [5]. All of these are based on the abstract framework of designating a special *server process*, whose *capacity* is equal to the remaining processor bandwidth after serving the periodic hard real-time tasks. This server capacity is used to service the best-effort tasks. Alternatively, slack stealing techniques under Rate Monotonic scheduling have been used in [6, 7, 8]. To better utilize system resources, dynamic priority algorithms such as the Earliest Deadline First (EDF) was used in [9, 10]. Server mechanisms under EDF were proposed in [11], where dynamic versions of the Deferrable and Sporadic Servers called the Deadline Deferrable Server and the Deadline Sporadic Server were presented. The Deadline Sporadic Server was then extended to an algorithm called the Deadline Exchange server. Lastly, five different algorithms with varying performance and complexity, for serving soft aperiodic tasks under EDF were proposed in [12, 13]. One of these algorithms, called the Total Bandwidth Server was extended to handle overload situations in [14], and an optimal algorithm for deadline assignment to soft aperiodic tasks under this scheme was proposed in [1].

All the above schemes are based on computing the remaining processor bandwidth after serving the periodic hard real-time tasks, and using this remaining bandwidth to schedule best-effort tasks. This remaining bandwidth (or utilization factor in the case of server mechanisms) is invariant over time and can therefore be specified as a single number. This is even the case when real-time tasks are not strictly periodic but their jobs are constrained by a minimum interarrival separation, or by other mechanisms different from the minimum interarrival time, such as that in the Rate-Based Execution task model [15, 16].

The setup in the case of packet scheduling is very different. Here packet arrivals from any real-time flow are constrained by *arrival curves*, which are described by general subadditive functions, each specifying the maximum amount of traffic that can arrive within any given time interval [17, 18].

For example, the (σ, ρ)-model [17] describes the worst case traffic from a flow j by a burst parameter σ_j and a long-term rate parameter ρ_j. This can be policed by a *leaky bucket mechanism* [19] and guarantees that within *any* time interval of length t, the maximum amount of traffic from flow j is bounded by $\sigma_j + \rho_j t$.

If with each such real-time flow j, a deadline d_j is associated with the interpretation that any packet from this flow has to be transmitted within d_j time units after its arrival, then it is possible to calculate the link capacity used for serving all such real-time flows. However, the *remaining link capacity* which can be used for serving best-effort flows is not invariant over time (as in the case of the remaining processor bandwidth) and can not be specified as a single number. It turns out to be a function (over time interval lengths) dependent on the arrival curves and deadlines of the real-time flows. None of the known algorithms from the processor scheduling domain extend to this case in a straightforward manner.

Deadline assignment for best-effort packets. Our algorithm is based on EDF scheduling. The real-time flows are specified using their arrival curves and a deadline is associated with each such flow. We assume that there is a single best-effort flow. This might be an aggregated flow combining multiple flows (see Section D for further clarifications on this). The proposed algorithm assigns a deadline to each best-effort packet (on a packet by packet basis), and schedules this packet along with the other real-time packets using EDF. Central to our algorithm is this *deadline assignment*, and we prove that the overall system always remains schedulable and the deadline assignment is optimal (in the sense that with a shorter deadline, some packet might miss its deadline).

The first algorithm we present, although optimal, is not feasible in practise since for any best-effort packet it requires the history of *all* the previous best-effort packets that arrived at the scheduler. However, based on this algorithm we propose several approximations which represent tradeoffs between the amount of computation that is required for each best-effort packet and the delay assigned to it. We show that the well known Total Bandwidth Server due to Spuri and Buttazzo [13] turn out to be one of these approximations. Our results with realistic traffic mixes consisting of audio, video, real-time transactions and best-effort flows like ftp, http and mail show between 25–45% improvements on the average in the delays experienced by the best-effort flows, without any of the real-time packets missing their deadlines.

Organization of the paper. In the next section we formally describe the model for characterizing real-time traffic in terms of arrival curves and deadlines; this forms the basis for all our algorithms. In Section 3 we briefly review the main concept behind the Total Bandwidth Server and other related server mechanisms. Its connection to our algorithm is established in Section 5.1. We also point out the different concerns existing while designing algorithms for processor scheduling, in contrast to packet scheduling, and motivate the design issues behind our algorithm. Section 4 describes our optimal algorithm, following which we present our different approximation schemes in Section 5. Our experimental results are presented in Section 6. Because of space restrictions all proofs in this paper have been deferred to the appendix.

2 Traffic Characterization

We denote by RT the set of real-time flows with traffic arrivals to the packet scheduler, and we have one best-effort flow which might be an aggregate of several best-effort flows. Packet arrivals from the real-time flows are constrained by arrival curves [17, 18] which specify an upper bound on the amount of traffic that can arrive from a flow within any specified time interval. Packet arrivals from the best-effort flow are not constrained in anyway, and are queued up, waiting to be served.

If $a_j(t)$ denotes the traffic that arrives at the scheduler from a real-time flow j at time t, then $A_j[t, t+\tau] = \int_t^{t+\tau} a_j(t)dt$ denotes the traffic arrivals from the flow j in the time interval $[t, t+\tau]$. The maximum traffic arrival from any flow $j \in RT$ to the packet scheduler is bounded by a right-continuous subadditive *traffic constraint function* A_j^* such that for all times $t > 0$ and for all $\tau \geq 0$ we have:

$$A_j[t, t+\tau] \leq A_j^*(\tau)$$

where $A_j^*(t) = 0$ for all $t < 0$ and $A_j^*(t) \geq 0$ for $t \geq 0$.

There are usually two mechanisms to ensure that the traffic from a flow j entering the scheduler conforms to the traffic constraint function A_j^*. The first is a *traffic policer* placed at the entrance of a network node, which rejects any traffic from a flow j that does not comply to A_j^*. The second mechanism is a *rate controller* which temporarily buffers packets to ensure that the traffic from the flow j conforms to A_j^*. Example constraint functions such as that given by the (σ, ρ)-model [17] can be policed by a leaky bucket mechanism [19].

Each real-time flow j has an associated deadline d_j, and any packet from this flow must be completely transmitted within d_j time units after its arrival. We denote the deadlines assigned to (by our scheduling algorithm) the best-effort packets by δ_k, where δ_k is the (relative) deadline assigned to the k-th best-effort packet. If r_k is the arrival time of this packet then its *absolute deadline* is equal to $r_k + \delta_k$. Throughout this paper, we refer to both absolute deadlines and relative deadlines, by the word *deadline*. It should be clear from the context which one we are referring to. Lastly, we denote the maximum packet size (from packets belonging to any flow) by s_{max}, and the transmission rate of the scheduler is equal to one.

3 Designing Schedulers for a Mix of Real-Time and Best-Effort Tasks

In contrast to static priority based schemes such as those mentioned in Section 1, server mechanisms based on EDF (such as [1, 2]) achieve full processor utilization. In the Total Bandwidth Server [13] algorithm it is assumed that the processor utilization due to periodic hard real-time tasks is U_p, and the remaining capacity $U_s = 1 - U_p$ is assigned to serve best-effort tasks. Whenever a best-effort job arrives at the scheduler, it is assigned a deadline and scheduled using EDF along with the real-time jobs. If the k-th best-effort job arrives at time r_k, then it receives a deadline:

$$d_k = \max\{r_k, d_{k-1}\} + c_k/U_s$$

where c_k is the worst case execution requirement of the job. It can be shown that with this deadline assignment, the whole system remains schedulable at all points in time [13].

Note that here the schedulability of the system heavily relies on an accurate estimation of the worst case execution requirement c_k. Underestimating this might lead to some real-time job missing its deadline. Since estimating worst case execution times of jobs is considerably difficult in the context of processor scheduling, several papers have proposed means for isolating real-time jobs from best-effort jobs served in this manner using a global EDF scheduler. Well known among these is the Constant Bandwidth Server due to Abeni and Buttazzo [2]. However, note that in the context of packet scheduling, this problem does not arise since packet lengths are known upon their arrival at the link scheduler and therefore real-time packets do not run into the risk of missing their deadlines.

3.1 EDF Vs Proportional Share

Our algorithm presented in this paper is based on EDF and is similar in spirit to the Total Bandwidth Server. However, as mentioned in Section 1, the remaining link capacity available for serving best-effort packets is no longer a single number, but a function of the traffic constraint functions A_j^* (described in Section 2) and deadlines associated with the real-time flows. Therefore, the simple deadline calculation as done in the case of the Total Bandwidth Server does not extend to this case.

Several algorithms based on resource reservations, sometimes with dynamic feedback, have been proposed in the context of processor scheduling to support a mix of real-time and best-effort tasks [20, 21, 22, 23, 24]. Indeed, a competing alternative to our proposed scheduler, would be one based on the idea of proportional share resource allocation (such as the generalized processor sharing or any of its packetized versions) [18]. Given the arrival curves A_j^* and the deadlines d_j for each real-time flow, it is possible to deduce the minimum link bandwidth required to serve each such flow such that its deadline is satisfied, and assign the remaining bandwidth to the best-effort flow. However, schedulers based on proportional share are primarily motivated by the need for *fair sharing* of surplus link bandwidth between different flows. On the other hand, our primary goal is to greedily allocate the maximum possible link bandwidth to the best-effort flow to improve its response time, subject to the constraint that the real-time packets do not miss their deadlines. In a deterministic setup, this is the only concern since real-time packets getting served much ahead of their designated deadlines do not improve the system performance.

Our choice of EDF as a scheduling discipline is also motivated by the fact that it is known to have a larger schedulability region compared to generalized processor scheduling [25]. It has also been shown in [25] that the RSVP parameters in an IntServ framework [26] can be mapped to EDF reservations. This guarantees the conformance of our algorithm with IntServ, which happens to be one of the widely accepted service disciplines for guaranteeing real-time constraints in the Internet. Further, it was also shown in [27] that for supporting hybrid real-time multimedia applications, deadline based schemes are more appropriate compared to proportional sharing.

4 Optimal Deadline Assignment for Best-Effort Packets

In this section we present our main algorithm. As mentioned before, it is similar to the Total Bandwidth Server of Spuri and Buttazzo [13] in the sense that best-effort packets are assigned a deadline on a packet-by-packet basis and are scheduled with the real-time packets using an EDF scheduler. However, the mechanism for deadline assignment in our case is considerably more involved and is a generalization of the algorithm for the Total Bandwidth Server.

In this section we make use of the traffic characterization defined in Section 2. Before describing our algorithm we need to define a few additional terms. For this, consider a mix of real-time and best-effort packets being served by the simple scheme in which real-time packets are served non-preemptively using EDF, and a best-effort packet is served only when no real-time packets are present at the scheduler. Then it can be shown using the results from [28] that the set of all real-time flows RT is schedulable if and only if for all $t \geq \min\{d_j \mid j \in RT\}$:

$$t \geq \sum_{k \in RT} A_k^*(t - d_k) + s_{max}$$

Based on this result, we define the *residual link capacity* over any time interval of length t, available to serve the best-effort flow, by a function $R_{RT}(t)$ where

$$R_{RT}(t) = t - (\sum_{k \in RT} A_k^*(t - d_k) + s_{max})$$

Recall from Section 2 that A_k^* is the traffic constraint function and d_k is the delay associated with the real-time flow k. s_{max} is the maximum packet length of packets belonging to any flow. We define $E_{RT}(t)$ to be the *effective residual link capacity* available within any time interval of length t to serve best-effort packets. This is given by

$$E_{RT}(t) = \min_{t' \geq t} R_{RT}(t')$$

Based on these two functions and the specifications of the real-time flows, our scheme for assigning deadlines to the best-effort packets is given by Algorithm 1.

Algorithm 1 Computing Deadlines for the Best-Effort Packets

```
Given a set $RT$ of real-time flows and one aggregated best-effort flow.
The residual link capacity $R_{RT}(t)$ to serve the best-effort flow, over any time interval of length $t$ is
$R_{RT}(t) = t - (\sum_{k \in RT} A_k^*(t - d_k) + s_{max})$.
Effective residual link capacity $E_{RT}(t) = \min_{t' \geq t} R_{RT}(t')$.

**Computing the deadline $\delta_n$ of the $n$-th best-effort packet:**
  The $n$-th best-effort packet arrives at time $r_n$ and has a transmission time of $w_n$
  $\delta_n^n = \min\{t \mid E_{RT}(t) \geq w_n\}$
  **for** $i = 1$ to $n - 1$ **do**
    $\delta_n^{n-i} = \min\{t \mid E_{RT}(t) \geq \sum_{j=n-i}^{n} w_j\} - (r_n - r_{n-i})$
  **end for**
  $\delta_n = \max\{\delta_n^i \mid i = 1, 2, \ldots, n\}$
```

Theorem 1 states that with the deadline assignment given by Algorithm 1, the order in which an EDF scheduler serves them is the same as the order in which they arrive at the scheduler. Therefore, Algorithm 1 does not disrupt the order of the best-effort packets.

Theorem 1 *For any $n \geq 1$, if δ_n and δ_{n+1} are the deadlines assigned to the n-th and the $(n+1)$-th best-effort packets by Algorithm 1 then $r_{n+1} + \delta_{n+1} > r_n + \delta_n$.*

To informally explain Algorithm 1, we introduce a family of functions $\alpha_n(t), n = 1, 2, \ldots$, where $\alpha_n(t)$ denotes the *service demand* within any time interval of length t due to a sequence of best-effort packets $1, 2, \ldots, n$, ending with the n-th packet. It means that within any time interval of length t, a sequence of best-effort packets from the set ordered $\{1, \ldots, n\}$ and ending with packet n would require a total transmission time equal to $\alpha_n(t)$ if they have to meet their assigned deadlines. Algorithm 1 assigns to each best-effort packet n the *shortest possible deadline*, maintaining the constraint that the *curve* $\alpha_n(t)$ always lies *below* the effective residual link capacity *curve* $E_{RT}(t)$. We formally prove this idea below in Theorems 2 and 3. Theorem 2 states that with the deadline assignment due to Algorithm 1, the overall system (consisting of real-time and best-effort packets) still remains schedulable, and Theorem 3 proves the optimality of the deadline assignment.

Based on the above idea of service demand functions $\alpha_n(t)$, if the first best-effort packet arrives at time r_1 and is assigned a (relative) deadline δ_1 (i.e. it has an absolute deadline equal to $r_1 + \delta_1$),

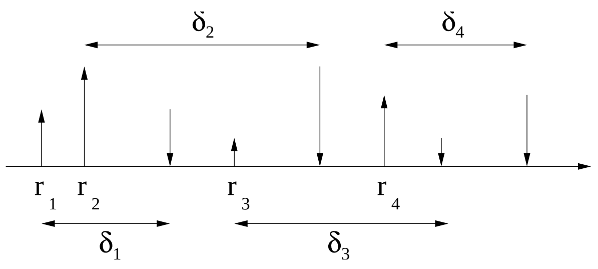

Figure 1: A possible sequence of best-effort packet arrivals and the corresponding packet deadlines.

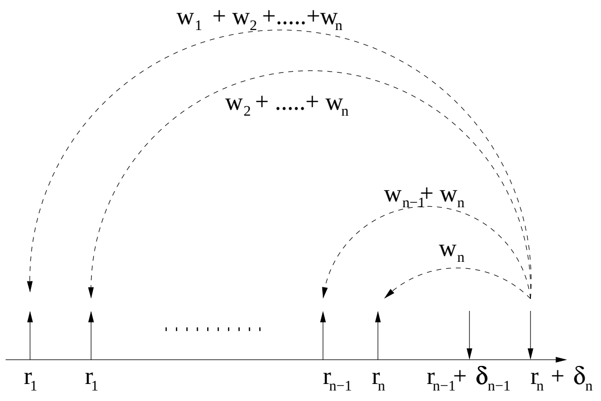

Figure 2: Service demands arising out of the n-th best-effort packet.

then within an interval of length δ_1 the service demand due to this best-effort packet is equal to w_1. Hence $\alpha_1(\delta_1) = w_1$, and $\alpha_1(\delta) = 0$ for all $\delta < \delta_1$. Now if the second best-effort packet arrives at time r_2 and is assigned a deadline δ_2, then within an interval of length δ_2, the service demand is w_2 and within an interval of length $\delta_2 + r_2 - r_1$ the service demand is $w_1 + w_2$. These two constraints are captured in $\alpha_2(t)$, and the deadline δ_2 is chosen so that $\alpha_2(t)$ for all values of $t \geq 0$ lies below $E_{RT}(t)$. This procedure is followed for any subsequent packets (see Figure 1). Therefore, for the n-th packet, the service demands within an interval of length—δ_n is w_n, $\delta_n + r_n - r_{n-1}$ is $w_{n-1} + w_n$, $\ldots$, $\delta_n + r_n - r_1$ is $w_1 + \ldots + w_n$ (see Figure 2). As before, δ_n is chosen by Algorithm 1 such that within any of these intervals the service demands is below the effective residual link capacity available within the interval.

4.1 An Alternative Interpretation

If we list all the above constraints for each packet, we get (see also Figure 1):

$$\begin{aligned}
\alpha_1(\delta_1) &= w_1 \\
\\
\alpha_2(\delta_2) &= w_2 \\
\alpha_2(\delta_2 + r_2 - r_1) &= \alpha_2(\delta_1 + (r_2 + \delta_2) - (r_1 + \delta_1)) = w_1 + w_2 \\
\\
\alpha_3(\delta_3) &= w_3 \\
\alpha_3(\delta_3 + r_3 - r_2) &= \alpha_3(\delta_2 + (r_3 + \delta_3) - (r_2 + \delta_2)) = w_2 + w_3
\end{aligned}$$

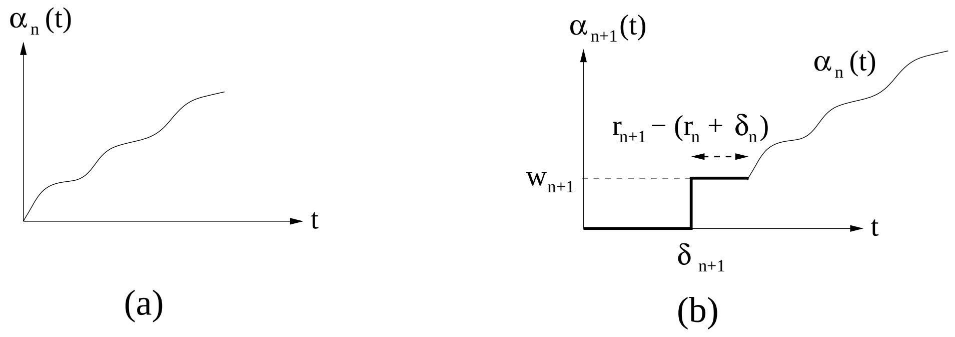

Figure 3: Constructing the service demand curve $\alpha_{n+1}(t)$ from the curve $\alpha_n(t)$. (a) The service demand curve $\alpha_n(t)$. (b) The service demand curve $\alpha_{n+1}(t)$.

$$\alpha_3(\delta_3 + r_3 - r_1) \quad = \quad \alpha_3(\delta_1 + (r_2 + \delta_2) - (r_1 + \delta_1) + (r_3 + \delta_3) - (r_2 + \delta_2)) = w_1 + w_2 + w_3$$

$$\cdots \quad \cdots \quad \cdots$$

It follows from the above list of constraints that, given the curve $\alpha_n(t)$, and $r_{n+1}, \delta_{n+1}, w_{n+1}$, it is possible to construct the curve $\alpha_{n+1}(t)$ by shifting $\alpha_n(t)$ horizontally by $(r_{n+1} + \delta_{n+1}) - (r_n + \delta_n)$, vertically by w_{n+1}, and appending a step function to it as shown in Figure 3. Note that the segment (in the middle) of length $r_{n+1} - (r_n + \delta_n)$ might be positive or negative in length. In the later case, a part of the last segment of $\alpha_n(t)$ gets deleted.

With this observation, Algorithm 1 can be alternatively interpreted as follows. Given the service demand curve $\alpha_n(t)$, δ_{n+1} is the smallest possible value assigned by Algorithm 1 such that the resulting curve $\alpha_{n+1}(t)$ lies below the curve $E_{RT}(t)$. With this deadline assignment the overall system remains schedulable, and with a deadline smaller than δ_{n+1} some packet (either real-time or best-effort) might miss its deadline. This is proved next in Theorems 2 and 3.

Theorem 2 *Given a set RT of schedulable real-time flows, under the deadline assignment for best-effort packets as given by Algorithm 1, the set RT still remains schedulable and all best-effort packets are transmitted by their assigned deadlines.*

Theorem 3 *If any best-effort packet is assigned a deadline smaller than that assigned by Algorithm 1, then some packet (either from a real-time or from the best-effort flow) might miss its deadline.*

It follows from the schedulability guarantee given by Theorem 2, and also the discussion in Section 4.1 that the deadlines assigned to the best-effort packets are dependent on their arrival rate. If too many best-effort packets arrive back-to-back and the effective residual link capacity is not too large to serve them, then the deadlines assigned to them grow larger and larger, but the real-time packets are never jeopardized. An *overload* situation created by best-effort traffic only increases their average response time. Lastly, it is clear that our scheme is never worse compared to the simple scheme of serving best-effort packets only when no real-time packets are present at the scheduler.

Note that the deadline assignment for any best-effort packet requires the entire history of all the previous packets that arrived at the scheduler. As an implementation hint (which is as well obvious to note), we point out that this history can be *reset* whenever the scheduler is idle, i.e. there are no real-time or best-effort packets with already assigned deadlines waiting to be served. The next packet arriving after such a rest can be treated as the *first* best-effort packet. For any realistic traffic flow, such resets of the scheduler will be fairly common.

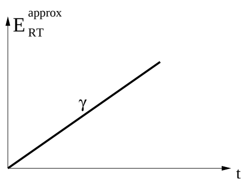

Figure 4: Approximating $E_{RT}(t)$ with a single line passing through the origin.

5 Approximating the Effective Residual Link Capacity

Algorithm 1 in spite of being optimal in terms of the response time experienced by best-effort packets, is infeasible for all practical purposes. This is because it requires maintaining the history of all the best-effort packets that arrived at the scheduler, to assign a deadline to the next packet. The reason for this is that the effective residual link capacity $E_{RT}(t)$ can in general be arbitrarily shaped. Combined with the fact that packet sizes can also be arbitrary, to fit the service demand functions $\alpha_n(t)$ as tightly as possible below $E_{RT}(t)$, the full structure of $\alpha_n(t)$ is required. The complexity of the algorithm can be reduced either by approximating the effective residual link capacity $E_{RT}(t)$ by a simple function, or by approximating the packet size to allow only a fixed number of distinct sizes. In this section we consider the former approach. It should be possible to derive the approximations based on later approach from the ideas that we present in this section. For the deadline assignment for any best-effort packet, the algorithms that we present in Sections 5.1 and 5.2 require the arrival times and deadlines of the previous packet only (in contrast to *all* the previous packets). The algorithm presented in Section 5.3 is more involved and requires the history of a bounded number of packets, and not all. Each of these algorithms represent a tradeoff between the complexity involved in the deadline assignment and the length of the deadline assigned.

5.1 With a Straight Line Passing Through the Origin

In this section we approximate the effective residual link capacity $E_{RT}(t)$ after serving the set of real-time flows RT, by a single straight line of slope γ passing through the origin (see Figure 4). To satisfy the schedulability condition given by Theorem 2, the slope γ is chosen to be the largest possible value such that this line lies below the exact $E_{RT}(t)$ calculated from the traffic constraint functions (or arrival curve) $A_j^*(t)$ of the real-time flows $j \in RT$, and the maximum packet size s_{max}. Therefore, $\gamma = \max\{\gamma' \mid \gamma' t \leq E_{RT}(t)\ \forall t \geq 0\}$. The deadlines of the best-effort packets are now computed with $E_{RT}^{approx}(t) = \gamma t$ as the effective residual link capacity available for transmitting the best-effort flow. We show that under this approximation, the deadline of any best-effort packet can now be optimally calculated by using parameters (such as arrival times and deadlines) belonging to the previous packet only. This is in contrast to our exact algorithm which required the arrival times of *all* the previous best-effort packets.

Consider the $(n+1)$-th best-effort packet which arrives at the time r_{n+1} and has a transmission time equal to w_{n+1}. It follows from Algorithm 1 that for the system to be schedulable the deadline assigned to this packet should satisfy:

$$\delta_{n+1} \geq w_{n+1}/\gamma \qquad (1)$$

Let $\alpha_n(t)$ be the service demand due to the best-effort packets $1, 2, \ldots, n$ and (x, y) be the point on $\alpha_n(t)$ which is closest to $E_{RT}^{approx}(t) = \gamma t$. Then it follows from our discussion in Section 4.1 that the new coordinates of this point in $\alpha_{n+1}(t)$ become:

$$(x + r_{n+1} + \delta_{n+1} - (r_n + \delta_n), y + w_{n+1})$$

For the system to be still schedulable, we require that:

$$(x + r_{n+1} + \delta_{n+1} - (r_n + \delta_n))\gamma \geq y + w_{n+1}$$

Since (x, y) was closest to $E_{RT}^{approx}(t) = \gamma t$, if (x, y) lies below this line then all other points on $\alpha_n(t)$ are also guaranteed to lie below it.

Assuming that the system was schedulable with the deadline assignment for the n-th packet, i.e. $x\gamma \geq y$, we have:

$$(r_{n+1} + \delta_{n+1} - (r_n + \delta_n))\gamma \geq w_{n+1}$$

or,

$$\delta_{n+1} \geq w_{n+1}/\gamma + (r_n + \delta_n) - r_{n+1} \tag{2}$$

From inequalities (1) and (2), we get the deadline assignment for the $n + 1$-th best-effort packet to be:

$$\delta_{n+1} = w_{n+1}/\gamma + \max\{0, (r_n + \delta_n) - r_{n+1}\}$$

Therefore, $r_{n+1} + \delta_{n+1} = w_{n+1}/\gamma + \max\{r_{n+1}, r_n + \delta_n\}$ which is exactly the same as the deadline assignment in the case of the Total Bandwidth Server. This follows from the fact, that our straight line with slope γ approximating the effective residual link capacity is equivalent to a *server* with an utilization factor of γ. The Total Bandwidth Server therefore reduces to a special case of our Algorithm 1.

5.2 With a Straight Line Cutting $t = \delta$

Here we approximate the effective residual link capacity $E_{RT}(t)$ again by a single straight line with slope γ, but crossing the t-axis at $t = \delta$ instead of passing through the origin as in our last approximation (see Figure 5). Therefore, it reduces to our last case if $\delta = 0$. The approximate effective residual link capacity is now given by:

$$E_{RT}^{approx}(t) = \begin{cases} 0 & \text{if } t \leq \delta \\ (t - \delta)\gamma & \text{if } t > \delta \end{cases}$$

The values γ and δ are chosen such that $E_{RT}^{approx}(t) \leq E_{RT}$ for all $t \geq 0$. Now consider the $(n+1)$-th best-effort packet that arrives at time r_{n+1} and has a transmission time of w_{n+1}. If δ_{n+1} is the deadline assigned to this packet then we require that:

$$\delta_{n+1} \geq \delta + w_{n+1}/\gamma \tag{3}$$

Let, as before, $\alpha_n(t)$ be the service demand due to the best-effort packets $1, 2, \ldots, n$ and (x, y) be the point on $\alpha_n(t)$ which is closest to E_{RT}^{approx}. For the new coordinate of (x, y) in $\alpha_{n+1}(t)$ to lie below E_{RT}^{approx}, after the deadline assignment δ_{n+1} to the $(n + 1)$-th packet, we require:

$$(x + r_{n+1} + \delta_{n+1} - (r_n + \delta_n) - \delta)\gamma \geq y + w_{n+1}$$

Assuming that the system was schedulable with the deadline assignment of the n-th packet, i.e. $(x - \delta)\gamma \geq y$, we have:

$$(r_{n+1} + \delta_{n+1} - (r_n + \delta_n))\gamma \geq w_{n+1}$$

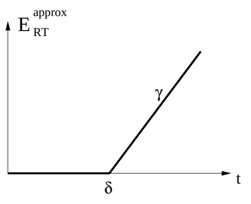

Figure 5: Approximating $E_{RT}(t)$ with a single line cutting $t = \delta$.

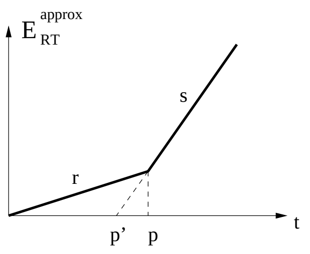

Figure 6: Approximating $E_{RT}(t)$ with a combination of two line segments of slope r and s ($r < s$).

or,

$$\delta_{n+1} \geq w_{n+1}/\gamma + (r_n + \delta_n) - r_{n+1} \tag{4}$$

From inequalities (3) and (4) we have,

$$\delta_{n+1} = w_{n+1}/\gamma + \max\{\delta, (r_n + \delta_n) - r_{n+1}\}$$

5.3 With a Combination of Two Line Segments

To more accurately approximate the effective residual link capacity, in this section we approximate it using a combination of two line segments (instead of one as in the previous cases), one of slope r passing through the origin and the second of slope s ($s > r$). The line segments intersect at a point whose t-coordinate is equal to p (see Figure 6). Hence, it is given by:

$$E_{RT}^{approx}(t) = \begin{cases} rt & \text{if } t < p \\ (t - p')s & \text{if } t \geq p \end{cases}$$

The t-intercept p' of the line with slope s can be calculated to be equal to $p - pr/s$. E_{RT}^{approx} reduces to the case in Section 5.1 if $r = s$, and to the case in Section 5.2 if $r = 0$ and $p = \delta$. As before, r, s and p are chosen such that $E_{RT}^{approx}(t) \leq E_{RT}$ for all $t \geq 0$.

Algorithm 2 gives the deadline assignment under this approximation. In contrast to the previous two algorithms, it requires the history of all the packets which constitute the portion of the service demand curve $\alpha_n(t)$ that lies to the left of $t = p$. This algorithm is based on the idea of maintaining the curve $\alpha_n(t)$ for any n only till the point $t = p$. Beyond $t = p$ only the point which is closest to the curve $E_{RT}^{approx}(t)$ is maintained. With the $(n+1)$-th packet, the deadline δ_{n+1} is chosen such that the part of $\alpha_n(t)$ that lies to the left of $t = p$ still continues to be below $E_{RT}^{approx}(t)$ in $\alpha_{n+1}(t)$, and the point on $\alpha_n(t)$ which is closest to $E_{RT}^{approx}(t)$ and lies to the right of $t = p$ also continues to lie below $E_{RT}^{approx}(t)$. Because of the horizontal shift of $\alpha_n(t)$ due to the $(n+1)$-th packet, some points on $\alpha_n(t)$ which were to the left of $t = p$ would now cross this point and become the new nearest point to $E_{RT}^{approx}(t)$ beyond $t = p$.

Algorithm 2 Computing the approximate deadline δ_{n+1} of the $(n+1)$-th best-effort packet based on approximating $E_{RT}(t)$ by a combination of two line segments

```
Given $S^n_{\leq p}$ = set of all packets $\{i \mid i \leq n$ and $(\delta_n + r_n - r_i) \leq p\}$.
Given $k_{>p}$ where, among the set of packets that DO NOT belong to $S^n_{\leq p}$, $k_{>p}$ is the $k$-th packet $(1 \leq k \leq n)$, such that the point $(\delta_n + r_n - r_k, \sum_{j=k}^{n} w_j)$ on $\alpha_n(t)$ is closest to the approximate effective residual link capacity curve $E^{approx}_{RT}(t)$.
The $(n+1)$-th best-effort packet arrives at time $r_{n+1}$ and has a transmission time of $w_{n+1}$.

    $/*$ Compute the minimum deadline to fit $\alpha_{n+1}(t)$ below $E^{approx}_{RT}(t)$ $*/$
Let $\delta_{min} = \min\{t \mid E^{approx}_{RT}(t) \geq w_{n+1}\}$
**for** all $i \in S^n_{\leq p}$ **do**
  Let $\delta = \min\{t \mid E^{approx}_{RT}(t) \geq \sum_{j=n+1}^{i} w_j\} - (r_{n+1} - r_j)$
  **if** $\delta > \delta_{min}$ **then** $\delta_{min} = \delta$ **end if**
**end for**
$\delta_{n+1} = \delta_{min}$

    $/*$ Find if a new point beyond $t = p$ becomes closer to $E^{approx}_{RT}(t)$ $*/$
**for** all $i \in S^n_{\leq p}$ **do**
  **if** $(r_{n+1} + \delta_{n+1} - r_i) > p$ **then**
    $S^n_{\leq p} = S^n_{\leq p} \backslash \{i\}$
    **if** $(\delta_{n+1} + r_{n+1} - r_i, \sum_{j=i}^{n+1} w_j)$ is closer to the $E^{approx}_{RT}(t)$ curve compared to $(\delta_{n+1} + r_{n+1} - r_{k_{>p}}, \sum_{j=k_{>p}}^{n+1} w_j)$ **then** $k_{>p} = i$ **end if**
  **end if**
**end for**

    $/*$ Update the set of points on $\alpha_{n+1}(t)$ which lies to the left of $t = p$ $*/$
**if** $\delta_{n+1} \leq p$ **then**
  $S^{n+1}_{\leq p} = S^n_{\leq p} \cup \{n+1\}$
**else**
  $S^{n+1}_{\leq p} = S^n_{\leq p}$
**end if**
```

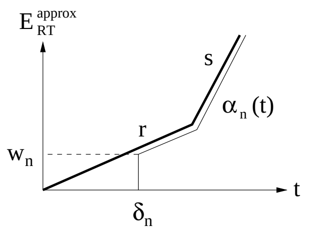

Figure 7: Approximating the service demand curve $\alpha_n(t)$ for all n, by assuming that it coincides with $E_{RT}^{approx}(t)$

5.3.1 Approximating the Deadline

In the last section, the deadline assignment for any best-effort packet required the arrival and the transmission times of multiple previous packets. More precisely, all the packets which constitute the portion of the service demand curve $\alpha_n(t)$ which was to the left of $t = p$ (i.e. $\alpha_n(t)$ for $t < p$). This was in contrast to our previous two approximation schemes where the deadline calculation for any packet required the arrival time and deadline of only one previous packet. While in many cases the number of packets that constitute the portion of the service demand curve lying on the left of $t = p$ can be small (and in any case it is bounded), there might be situations where the number of such packets are very large. In the later case, computing the deadline of an incoming best-effort packet will involve an amount of computation and storage requirement which might be infeasible in practise, as in the case of our exact algorithm in Section 4.

In this section, apart from approximating the effective residual link capacity $E_{RT}(t)$ using a combination of two line segments, we also approximate the (optimal) deadline that can be assigned using $E_{RT}^{approx}(t)$ as the effective residual link capacity. Since we want to guarantee schedulability, the deadline assigned to any best-effort packet will now be greater than or equal to the deadline assignment in Section 5.3 using $E_{RT}^{approx}(t)$ as the effective residual link capacity. Our approximation is based on the idea of assuming that the service demand curve $\alpha_n(t)$ at all points of time coincides exactly with the approximate effective residual link capacity $E_{RT}^{approx}(t)$. This is shown in Figure 7.

If δ_{n+1} is the deadline assigned to the $n+1$-th best-effort packet, then any point (x, y) on the $\alpha_n(t)$ will be shifted to the point (x', y') on $\alpha_{n+1}(t)$ where (x', y') is given by:

$$(x + r_{n+1} + \delta_{n+1} - (r_n + \delta_n), y + w_{n+1})$$

For the point (δ_{n+1}, w_{n+1}) in $\alpha_{n+1}(t)$ to lie below $E_{RT}^{approx}(t)$, the following must hold:

$$\delta_{n+1} \geq \begin{cases} w_{n+1}/r & \text{if } w_{n+1} < rp \\ (w_{n+1} - pr)/s + p & \text{if } w_{n+1} \geq rp \end{cases} \tag{5}$$

Since all points on $\alpha_n(t)$ get displaced by the same amount in $\alpha_{n+1}(t)$, the deadline δ_{n+1} should be large enough to guarantee that the point (δ_n, w_n) on $\alpha_n(t)$ when shifted to a point in $\alpha_{n+1}(t)$, lies below $E_{RT}^{approx}(t)$. This, along with inequality(5) would guarantee that $\alpha_{n+1}(t)$ lies below $E_{RT}^{approx}(t)$ (it follows from the fact that $E_{RT}^{approx}(t)$ is convex, since $s \geq r$). Such a deadline assignment is given by Algorithm 3.

Following this approach, for the deadline assignment of the $(n+2)$-th packet, $\alpha_{n+1}(t)$ is assumed to exactly coincide with E_{RT}^{approx} and the same steps as before are followed.

Algorithm 3 Approximating the deadline δ_{n+1}

```
The $(n+1)$-th best-effort packet arrives at time $r_{n+1}$ and has a transmission time of $w_{n+1}$.
The effective residual link capacity is given by $E_{RT}^{approx}(t)$.
if $(w_n + w_{n+1}) \geq rp$ then
    if $w_{n+1} < rp$ then
        $\delta_{n+1} = \max\{\frac{w_{n+1}}{r}, \frac{w_n+w_{n+1}}{s} + (r_n - r_{n+1}) + p - \frac{pr}{s}\}$
    else        /* i.e. if $w_{n+1} \geq rp$ */
        $\delta_{n+1} = \frac{w_n+w_{n+1}}{s} + (r_n - r_{n+1}) + p - \frac{pr}{s}$
    end if
else        /* if $(w_n + w_{n+1}) < rp$ */
    $\delta_{n+1} = \frac{w_{n+1}}{r} + \max\{0, (r_n + \delta_n) - r_{n+1}\}$
end if
```

6 Experimental Results

In this section we describe our results from simulations modeling a 10MBit/sec access link with six different traffic classes. Three real-time (RT) flows require about two thirds of the link capacity. The single aggregated best-effort flow is obtained from three additional non real-time flows (NRT) which fairly share the residual link capacity by passing through a Weighted Fair Queuing (WFQ) based scheduler before our deadline assignment for best-effort traffic is applied. A detailed description of this mechanism along with the traffic flows, their corresponding link bandwidth requirements, and the approximations used for the deadline assignment of best-effort traffic can be found in Section D of the appendix.

The results for the maximum and the average delay experienced by packets of the flows are given in the Tables 1 and 2. We have simulated the network node through which the traffic flows are passing, for a period of six minutes, which corresponds to about 5×10^5 processed packets. The first column shows the results for a plain best-effort (BE) scheme where best-effort traffic is transmitted only if the EDF scheduler is idle (i.e. there is no backlog from real-time flows). This column is used as a reference for a comparison with our two approximations. The second column states the results for a deadline assignment for best-effort traffic using our simple approximation of the effective residual link capacity by a shifted straight line (from Section 5.2). The third column shows the delay values corresponding to the more involved approximation using two line segments (from Section 5.3).

Table 1: Maximum delay experienced by flows within 360sec period, given in $msec$.

flow class	*plain BE scheme*	*shifted line approx.*	*two line approx.*
RT transactions	2.11 / 100%	3.36 / 159%	5.73 / 272%
RT video	3.48 / 100%	9.95 / 286%	13.97 / 401%
RT voice	1.31 / 100%	1.31 / 100%	2.54 / 194%
NRT ftp	21.95 / 100%	14.25 / 65%	7.56 / 34%
NRT http	21.29 / 100%	16.14 / 76%	11.14 / 52%
NRT mail	28.82 / 100%	22.84 / 79%	16.69 / 58%

From the simulations it is obvious that a noticeable benefit can be gained even from the application of the very simple linear approximation of the effective residual link capacity that was presented in Section 5.2 to improve the response time for best-effort traffic. If we are able to afford the more involved approximation using two segments, then further improvement in the delays experienced by

Table 2: Average delay experienced by flows within 360sec period, given in $msec$.

flow class	*plain BE scheme*	*shifted line approx.*	*two line approx.*
RT transactions	0.77 / 100%	0.86 / 117%	1.00 / 130%
RT video	1.52 / 100%	1.83 / 120%	1.88 / 124%
RT voice	0.53 / 100%	0.53 / 100%	0.62 / 117%
NRT ftp	2.50 / 100%	1.87 / 75%	1.73 / 69%
NRT http	2.61 / 100%	1.93 / 74%	1.78 / 68%
NRT mail	3.61 / 100%	2.28 / 63%	1.97 / 55%

the best-effort traffic can be achieved. From the results, it is also possible to recognize that the real-time traffic with the most loose deadline (RT video) experiences the largest slow down. Of course, in spite of the injection of the best-effort traffic with deadlines into the EDF scheduler, none of the deadlines associated with the real-time traffic are missed. This is because we only exploit the inherent mobility (in time) of the real-time packets due to the unused link capacity. The representations of the simulation trace in Figures 8 and 9 underpins the positive effect of our algorithm on the responsiveness for the non real-time flows. For readability reasons, only two selected flows are displayed.

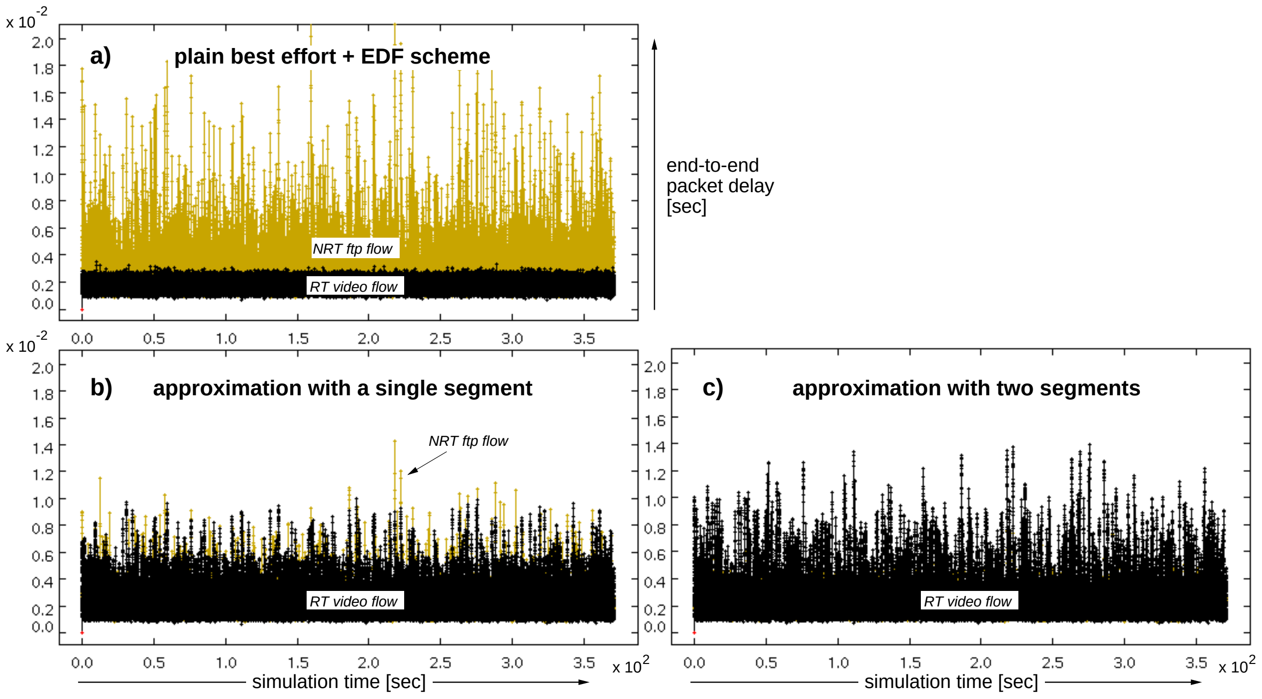

Figure 8: Comparison of the delay experienced by two (out of the six) selected flows using our different approximation schemes for best-effort service. (a) The delays experienced by a RT and a NRT flow when NRT packets are served only when no RT packets are present at the scheduler. (b) The delays experienced when our approximation scheme in Section 5.2 is used. (c) Further improvement in the delay experienced by the NRT flow using our approximation scheme in Section 5.3.

Note that the maximum delay experienced by some of the best-effort traffic improves by almost 65%, whereas the average delay improves by almost 45% in some cases. In the case of best-effort

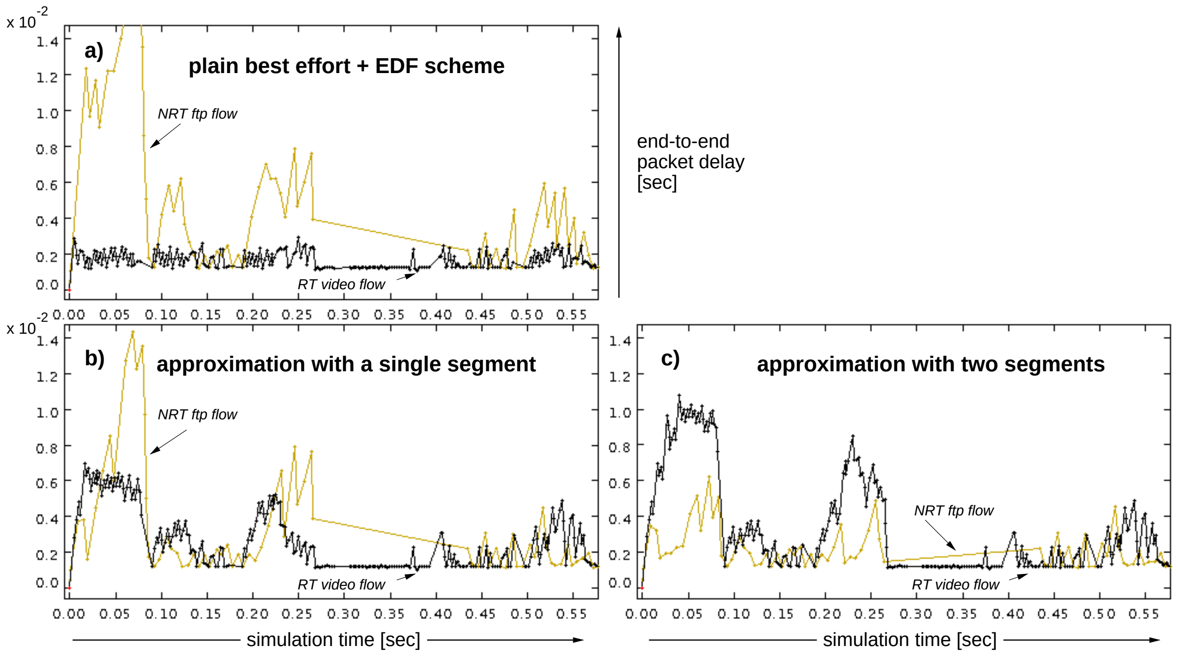

Figure 9: Comparison of the delay experienced by the two selected flows shown in Figure 8 Here an excerpt from the simulation trace given in Figure 8 is shown.

flows like sporadic http requests such benefits can be immediately perceived, proving the effectiveness of our scheduler.

7 Conclusions

In this paper we presented a packet scheduling algorithm to improve the response time of best-effort packets in the presence of real-time traffic flows. Although this algorithm is infeasible to implement in practise, we presented several approximations based on this optimal algorithm, which represent tradeoffs between complexity and performance. All of our algorithms exploit the mobility in time of real-time packets with deadlines relatively far in the future to inject best-effort packets into the scheduler without violating any real-time deadlines. Thus, we reduce the delay experienced by non real-time flows since we do not require the EDF scheduler to be idle before best-effort traffic is eligible for service. Our experiments used a combination of WFQ and EDF schedulers. WFQ was used to fairly distribute the remaining link bandwidth among the different best-effort flows, and EDF was used as an overall scheduling strategy to benefit from its larger schedulability region compared to WFQ.

We are aware of improved EDF-based scheduling disciplines such as service curve-based EDF (SCED [29]) and would like to point here that our deadline assignment for best-effort traffic can also be applied to this case by calculating the effective residual link capacity based on the schedulability test of SCED (eq. (14) in [29]). In the same way it is possible to refine the definition of fair shares for best-effort flows by using the Fair Service Curve approach in [30].

References

[1] G. Buttazzo and F. Sensini, "Optimal deadline assignment for scheduling soft aperiodic tasks in hard real-time environments," *IEEE Transactions on Computers*, vol. 48, no. 10, pp. 1035–1052, October 1999.

[2] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *Proc. 19th IEEE Real-Time Systems Symposium*. 1998, pp. 4–13, IEEE Computer Society Press.

[3] J. Lehoczsky, L. Sha, and J. Strosnider, "Enhanced aperiodic responsiveness in hard real-time environments," in *Proc. Real-Time Systems Symposium*, 1987, pp. 261–270.

[4] B. Sprunt, L. Sha, and J. Lehoczsky, "Aperiodic task scheduling for hard real-time systems," *Real-Time Systems*, vol. 1, pp. 27–60, 1989.

[5] J. Strosnider, J. Lehoczsky, and L. Sha, "The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments," *IEEE Transactions on Computers*, vol. 44, no. 1, pp. 73–91, 1995.

[6] R. Davis, K. Tindell, and A. Burns, "Scheduling slack time in fixed priority preemptive systems," in *Proc. Real-Time Systems Symposium*, 1993, pp. 222–231.

[7] J. Lehoczsky and S. Ramos-Thuel, "An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems," in *Proc. Real-Time Systems Symposium*, 1992, pp. 110–123.

[8] T.-S. Tia, J.-S. Liu, and M. Shankar, "Algorithms and optimality of scheduling soft aperiodic requests in fixed-priority preemptive systems," *Real-Time Systems*, vol. 10, no. 1, pp. 23–43, 1996.

[9] H. Chetto and M. Chetto, "Some results of the earliest deadline scheduling algorithm," *IEEE Transactions on Software Engineering*, vol. 15, no. 10, pp. 1261–1269, 1989.

[10] H. Chetto, M. Silly, and T. Bouchentouf, "Dynamic scheduling of real-time tasks under precedence constraints," *Real-Time Systems*, vol. 2, pp. 181–194, 1990.

[11] T. Ghazalie and T. Baker, "Aperiodic servers in a deadline scheduling environment," *Real-Time Systems*, vol. 9, no. 1, pp. 31–67, 1995.

[12] M. Spuri and G. Buttazzo, "Efficient aperiodic service under earliest deadline scheduling," in *Proc. IEEE Real-Time Systems Symposium*, 1994.

[13] M. Spuri and G. Buttazzo, "Scheduling aperiodic tasks in dynamic priority systems," *Real-Time Systems*, vol. 10, no. 2, pp. 179–210, 1996.

[14] M. Spuri, G. Buttazzo, and F. Sensini, "Robust aperiodic scheduling under dynamic priority systems," in *Proc. 16th IEEE Real-Time Systems Symposium*, 1995, pp. 210–221.

[15] K. Jeffay and D. Bennett, "A rate-based execution abstraction for multimedia computing," in *Proc. 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, 1995, LNCS 1018, pp. 64–75.

[16] K. Jeffay and S. Goddard, “Rate-based resource allocation models for embedded systems,” in *Proc. 1st Workshop on Embedded Software (EMSOFT)*. 2001, LNCS 2211, pp. 204–222, Springer-Verlag.

[17] R. Cruz, “A calculus for network delay, Part I: Network elements in isolation,” *IEEE Transactions on Information Theory*, 1991.

[18] A. Parekh and R. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: The single-node case,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, 1993.

[19] J. Turner, “New directions in communications (or which way to the information age?),” *IEEE Communications Magazine*, vol. 25, no. 8, pp. 8–15, 1986.

[20] M. Jones, D. Rosu, and M.-C. Rosu, “CPU reservations and time constraints: Efficient, predictable scheduling of independent activities,” in *Proc. 16th ACM Symposium on Operating System Principles*, 1997, pp. 198–211.

[21] J. Nieh and M. Lam, “The design, implementation and evaluation of SMART: A scheduler for multimedia applications,” in *Proc. 16th ACM Symposium on Operating System Principles*, 1997, pp. 184–197.

[22] D. Steere, A. Goel, J. Gruenberg, D. McNamee, C. Pu, and J. Walpole, “A feedback-driven proportion allocator for real-rate scheduling,” in *Proc. 3rd USENIX Symposium on Operating Systems Design and Implementation*, 1999, pp. 145–158.

[23] D. de Niz and R. Rajkumar, “Chocolate: A reservation-based real-time java environment on windows/nt,” in *Proc. 6th IEEE Real Time Technology and Applications Symposium*, 2000.

[24] A. Bavier and L. Peterson, “BERT: A scheduler for best effort and real-time tasks,” Tech. Rep. TR-602-99, Department of Computer Science, Princeton University, 2001.

[25] L. Georgiadis, R. Guérin, V. Peris, and K. N. Sivarajan, “Efficient network QoS provisioning based on per node traffic shaping,” *IEEE/ACM Transactions on Networking*, vol. 4, no. 4, pp. 482–501, Aug. 1996.

[26] B. Braden, D. Clark, and S. Shenker, “Integrated Services in the Internet architecture: an overview,” Request for Comments 1633, Internet Engineering Task Force (IETF), June 1994.

[27] L. Abeni, G. Lipari, and G. Buttazzo, “Constant bandwidth vs proportional share resource allocation,” in *Proc. IEEE International Conference on Multimedia Computing and Systems*, 1999, vol. 2, pp. 107–111.

[28] J. Liebeherr, D. Wrege, and D. Ferrari, “Exact admission control for networks with a bounded delay service,” *IEEE/ACM Transactions on Networking*, vol. 4, no. 6, pp. 885–901, 1996.

[29] H. Sariowan, R. Cruz, and G. Polyzos, “SCED: A generalized scheduling policy for guaranteeing Quality-of-Service,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 669–684, Oct. 1999.

[30] T. S. E. Ng, D. Stephens, I. Stoica, and H. Zhang, “Supporting best-effort traffic with Fair Service Curve,” in *GLOBECOM’99*, Dec. 1999, pp. 1799–1807.

[31] S. Shenker and J. Wroclawski, "General characterization parameters for integrated service network elements," Request for Comments 1633, Internet Engineering Task Force (IETF), Sept. 1997.

Appendix

A Proof of Theorem 1

It follows from Algorithm 1 that for all $i = 0, 1, \ldots, n-1$, δ_n^{n-i} is the minimum possible value for which

$$E_{RT}(\delta_n^{n-i} + r_n - r_{n-i}) \geq \sum_{j=n-i}^{n} w_j$$

Similarly, for all $i = 0, 1, \ldots, n$, δ_{n+1}^{n+1-i} is the minimum possible value for which

$$E_{RT}(\delta_{n+1}^{n+1-i} + r_{n+1} - r_{n+1-i}) \geq \sum_{j=n+1-i}^{n+1} w_j$$

Now, for any k ($0 \leq k \leq n-1$), consider the following two values δ_n^{n-k} and δ_{n+1}^{n-k}, defined respectively by the following two inequalities:

$$\begin{aligned} \text{minimum}\, \delta_n^{n-k} \text{ such that } E_{RT}(\delta_n^{n-k} + r_n - r_{n-k}) &\geq \sum_{j=n-k}^{n} w_j \\ \text{minimum}\, \delta_{n+1}^{n+1-(k+1)} \text{ such that } E_{RT}(\delta_{n+1}^{(n+1)-(k+1)} + r_{n+1} - r_{n+1-(k+1)}) &\geq \sum_{j=n+1-(k+1)}^{n+1} w_j \end{aligned}$$

Since $E_{RT}(t)$ is non-decreasing, it follows from the above that $\delta_n^{n-k} + r_n < \delta_{n+1}^{n-k} + r_{n+1}$ for $k = 0, 1, \ldots, n-1$. Hence, $\max\{\delta_n^{n-k} + r_n \mid k = 0, 1, \ldots, n-1\}$

$$\begin{aligned} &< \max\{\delta_{n+1}^{n-k} + r_{n+1} \mid k = 0, 1, \ldots, n-1\} \\ &\leq \max\{\delta_{n+1}^{n-k} + r_{n+1} \mid k = 0, 1, \ldots, n\} \end{aligned}$$

Therefore, $\delta_n + r_n < \delta_{n+1} + r_{n+1}$.

B Proof of Theorem 2

Consider a packet from a real-time flow $k \in RT$ that arrives at the scheduler at time t and is completely transmitted at time $t + \delta$. Our proof is based on bounding the workload that has to be transmitted by the scheluder before this packet is completely transmitted. For any real-time flow $j \in RT$, we denote by $A_j^{\leq x}[t, t+\tau]$ the traffic arrival from flow j during the time interval $[t, t+\tau]$ with deadlines less than or equal to x. We denote by $W^{\leq x}(y)$ the workload in the scheduler at time y due to packets with deadlines less than or equal to time x, i.e. $W^{\leq x}(y)$ is equal to the total transmission time that is required to deliver the packets waiting at the scheduler at time y, and having deadlines less than or equal to x. Lastly, let us denote by $W^{k,t}(t+\tau)$ ($0 \leq \tau \leq \delta$) the workload in the scheduler at the time $t+\tau$, that must be served before the packet that arrived at time t from the real-time flow k can be completely transmitted (note that $W^{k,t}(t,\tau)$ includes the transmission time of this packet). Now let $t - \hat{\tau}$ ($\hat{\tau} \geq 0$) be the last time before time t when the scheduler does not contain any traffic with a deadline less than or equal to the deadline of this packet, i.e. $t + d_k$. In other words, in the time interval $[t - t\hat{a}u, t+\delta)$, the scheduler always contains some packet with deadline less than or equal to $t + d_k$.

$W^{k,t}(t+\tau)$ is then composed of the following terms:

- The remaining transmission time of the packet that was in transmission at time $t - \hat{\tau}$. Let us denote this by $R(t-\hat{\tau})$. By our assumption of $t - \hat{\tau}$, the deadline of this packet is greater than $t + d_k$.

- Transmission times of packets arriving in the time interval $[t - \hat{\tau}, t+\tau]$, from real-time flows with deadlines less than or equal to $t + d_k$. This is equal to $\sum_{j \in RT} A_j^{\leq t+d_k}[t - \hat{\tau}, t+\tau]$, and includes the packet from the real-time flow k that arrived at time t.

- Transmission times of best-effort packets that arrive in the time interval $[t - \hat{\tau}, t+\tau]$ and have deadlines less than or equal to $t + d_k$. Let us denote this by $BE^{\leq t+d_k}[t - \hat{\tau}, t+\tau]$.

- The length of the time interval $[t - \hat{\tau}, t+\tau]$, i.e. $\hat{\tau} + \tau$.

Since packet transmissions can not be preempted, during the time interval $[t-\hat{\tau}, t-\hat{\tau}+R(t-\hat{\tau}))$, the packet with deadline greater than $t + d_k$ is transmitted, and during the time interval $[t - \hat{\tau} + R(t - \hat{\tau}), t - \hat{\tau}]$ all the packets transmitted have a deadline less than or equal to $t + d_k$. Hence, $W^{k,t}(t+\tau)$ ($0 \leq \tau \leq \delta$) is given as follows:

$$W^{k,t}(t+\tau) = \sum_{j \in RT} A_j^{\leq t+d_k}[t-\hat{\tau}, t+\tau] + BE^{\leq t+d_k}[t-\hat{\tau}, t+\tau] + R(t-\hat{\tau}) - (\hat{\tau}+\tau)$$

When $\tau = d_k$, the above equality becomes:

$$W^{k,t}(t+d_k) = \sum_{j \in RT} A_j^{\leq t+d_k}[t-\hat{\tau}, t+d_k] + BE^{\leq t+d_k}[t-\hat{\tau}, t+d_k] + R(t-\hat{\tau}) - (\hat{\tau}+d_k) \quad (6)$$

Since any packet from a real-time flow j, that arrives after the time $t + d_k - d_j$ has a deadline greater than $t + d_k$, Equation (6) can be written as:

$$W^{k,t}(t+d_k) = \sum_{j \in RT} A_j[t-\hat{\tau}, t+d_k-d_j] + BE^{\leq t+d_k}[t-\hat{\tau}, t+d_k] + R(t-\hat{\tau}) - (\hat{\tau}+d_k)$$

Now we would like to bound the workload $W^{k,t}(t+d_k)$. For this, we first compute an upper bound on $BE^{\leq t+d_k}[t-\hat{\tau}, t+d_k]$. Let the first best-effort packet that has an arrival time of greater than or equal to $t - \hat{\tau}$ and a deadline less than or equal to $t + d_k$, be the m-th best-effort packet (starting from the very first best-effort packet that was received by the scheduler), i.e. $\tau_m \geq t - \hat{\tau}$

and $r_m + \delta_m \leq t + d_k$. Similarly, let the last such best-effort packet be the n-th ($n \geq m$) packet, i.e. $r_n \geq t - \hat{\tau}$ and $r_n + \delta_n \leq t + d_k$. Then it follows from Algorithm 1 that:

$$E_{RT}(\delta_n + r_n - r_m) \geq w_m + w_{m+1} + \ldots + w_n$$

Clearly,

$$BE^{\geq t+d_k}[t - \hat{\tau}, t + d_k] = w_m + w_{m+1} + \ldots + w_n$$

Hence,

$$BE^{\geq t+d_k}[t - \hat{\tau}, t + d_k] \leq E_{RT}(\delta_n + r_n - r_m)$$

Therefore we have:

$$W^{k,t}(t + d_k) \leq \sum_{j \in RT} A_j[t - \hat{\tau}, t + d_k - d_j] + E_{RT}(\delta_n + r_n - r_m) + R(t - \hat{\tau}) - (\hat{\tau} + d_k)$$

Since $R(t - \hat{\tau}) \leq s_{max}$ and $E_{RT}(\delta_n + r_n - r_m) \leq E_{RT}(\hat{\tau} + d_k)$ (because $\delta_n + r_n - r_m \leq \hat{\tau} + d_k$), we have:

$$\begin{aligned}
W^{k,t}(t + d_k) &\leq \sum_{j \in RT} A^*_j(\hat{\tau} + d_k - d_j) + E_{RT}(\hat{\tau} + d_k) + s_{\max} - (\hat{\tau} + d_k) \\
&= \sum_{j \in RT} A^*_j(\hat{\tau} + d_k - d_j) + \min_{t' \geq (\hat{\tau} + d_k)} R_{RT}(t') + s_{max} - (\hat{\tau} + d_k) \\
&\leq \sum_{j \in RT} A^*_j(\hat{\tau} + d_k - d_j) + R_{RT}(\hat{\tau} + d_k) + s_{max} - (\hat{\tau} + d_k) \\
&= 0
\end{aligned}$$

Hence, there exists a $\tau \leq t + d_k$ such that $W^{k,t}(t + \tau) = 0$, implying that the packet from the real-time flow k, that arrived at time t, is completely transmitted on or before time $t + d_k$, thereby meeting its deadline. Using very similar arguments it is possible to show that all best-effort packets also meet their assigned deadlines.

C Proof of Theorem 3

Let the deadline assignment for the n-th best-effort packet be less than the deadline δ_n calculated by Algorithm 1. Let this be δ'_n ($\delta'_n < \delta_n$). This implies that there exists some i, such that the sum of the transmission times of the best-effort packets $n, n-1, \ldots, i$ (i.e. $w_n + w_{n-1} + \ldots + w_i$) that have to be transmitted within an interval of length $\delta'_n + (r_n - r_i)$ exceeds $E_{RT}(\delta'_n + (r_n - r_i))$. Since for all time intervals of length $t \geq 0$, $E_{RT}(t) = \min_{t' \geq t} R_{RT}(t')$, the above implies that there exists some $j \leq i$, such that the sum of the transmission times of the best-effort packets $n, n-1, n-2, \ldots, j$ (i.e. $w_n + w_{n-1} + \ldots + w_j$) that have to be served within a time interval of length $\delta'_n + (r_n - r_j)$, so that all deadlines are met, exceeds $R_{RT}(\delta'_n + (r_n - r_j))$. Starting from the time r_j, the scheduler has to transmit the best-effort packets $j, j+1, \ldots, n$ within the next $\delta'_n + (r_n - r_j)$ time units, if all the assigned deadlines have to be met.

Assume that the scheduler is empty before the time r_j, and at time r_j^- a maximum sized packet (of size equal to s_{max}) from either a real-time flow l with $d_l > \delta'_n + (r_n - r_j)$, or from a best-effort flow with the deadline of the packet greater than $\delta'_n + (r_n - r_j)$ arrives. Starting from time r_j, packets from the real-time flows k with $d_k \leq \delta'_n + (r_n - r_j)$ arrive according to their maximum arrival rate A_k^*. Additionally, packets from the best effort flow arrive at times $r_j, r_{j+1}, \ldots, r_n$. They are assigned deadlines $\delta_j, \delta_{j+1}, \ldots, \delta_{n-1}$ according to Algorithm 1, with the exception that the n-th packet is assigned the deadline $\delta'_n < \delta_n$ as discussed above.

Since the scheduler is non-preemptive, the packet of size s_{max} that arrived at time r_j^- has to be transmitted first, before any other packet. Therefore, the total transmission time of the packets that have to be transmitted within a time interval of length $\delta'_n + (r_n - r_j)$ starting from the time r_j, if all deadlines have to be met is equal to:

$$s_{max} + \sum_{k \in RT} A_k^{\leq r_j + \delta'_n + (r_n - r_j)}[r_j, r_j + \delta'_n + (r_n - r_j)] + w_n + w_{n-1} + \ldots + w_j$$

Hence, $W^{\leq \delta'_n + (r_n - r_j)}(r_j + \delta'_n + (r_n - r_j))$

$$\begin{aligned} &> s_{max} + \sum_{k \in RT} A_k^*(\delta'_n + (r_n - r_j) - d_k) + R_{RT}(\delta'_n + (r_n - r_j)) - (\delta'_n + (r_n - r_j)) \\ &= 0 \end{aligned}$$

Hence, at the time instant $r_j + \delta'_n + (r_n - r_j)$, the scheduler contains traffic with deadline less than or equal to $r_j + \delta'_n + (r_n - r_j)$, implying a deadline violation for some packet.

D Network traffic characteristics

D.1 Arrival curves and scheduling parameters

The traffic generators used for our simulations greedily generate packets as soon as they comply to a given TSpec-constrained [31] arrival curve as sketched in Figure 10. The parameters for our set of flows are given in Table 3. We distinguish three real-time and three non real-time flows. The real-time transactions class resembles traffic with a low bandwidth but hard deadline requirement to communicate with bandwidth brokers, bank applications, etc. Contrary to that, the video class models a high-bandwidth video with frame sizes considerably larger than the maximum packet length of 1536 Byte. A particular burstiness is caused by varying frame lengths due to predictive inter- or intraframe coding. Finally, the real-time voice class aggregates a couple of constant-bit-rate voice sources. The three non real-time classes can also be distinguished by varying burstiness and bandwidth requirements. For instance, the NRT mail class forms the counterpart of the RT transactions class in the set of non real-time flows since the mail class also generates moderately average rates.

Table 3 also specifies the deadlines associated with real-time flows. Our main EDF scheduler is hierarchically combined with a WFQ scheduler into which the different best-effort flows are first fed (see Figure 11). The WFQ scheduler as a result creates an ordering of the packets belonging to the different best-effort flows and results in an aggregated best-effort flow. Our deadline assignment for best-effort traffic is applied to packets belonging to this aggregated flow, after the WFQ scheduler has determined the ordering. These packets are then injected into the EDF part of the scheduler where they are scheduled along with the other real-time packets. The effective residual link capacity is therefore shared in a fair manner by scheduling the non real-time flows with the WFQ-based scheduler. The corresponding weights used by the WFQ scheduler are also stated in Table 3. The minimum packet length is bounded by 40 byte. The link capacity is set to 10 MBit/sec to model a next-generation access link from a home or an office to a service provider. Based on the parameters in Table 3 we can now derive the effective residual link capacity as shown in Figure 12. In addition, the two different approximations of the effective residual capacity used for our simulations are also sketched in this figure.

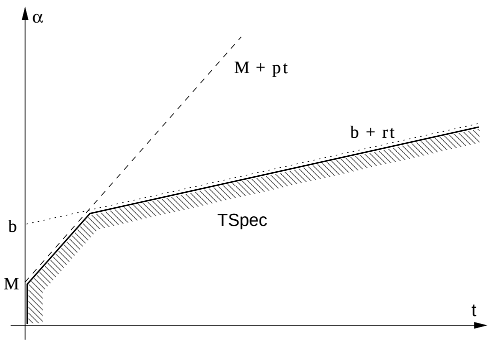

Figure 10: Arrival curve $\alpha(t)$ of a TSpec-constrained flow.

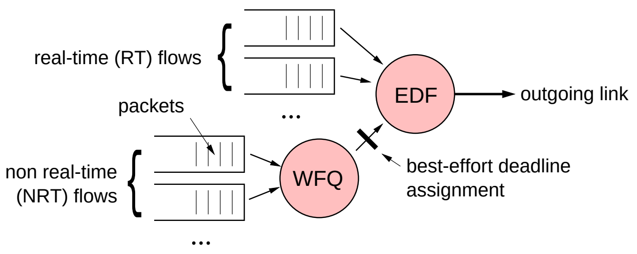

Figure 11: Hierarchical configuration of RT and NRT schedulers.

Table 3: TSpec description of traffic flows, see parameters in Fig. 10.

flow class	*b* [byte], *r* [byte/sec]	*M* [byte], *p* [byte/sec]	*deadline* [msec]
RT transactions	45000, 50000	700, 150000	20
RT video	15000, 600000	1536, 800000	30
RT voice	300, 150000	100, 250000	5

flow class	*b* [byte], *r* [byte/sec]	*M* [byte], *p* [byte/sec]	*WFQ weight*
NRT ftp	30720, 150000	1536, 250000	0.5
NRT http	50000, 100000	1536, 150000	0.2
NRT mail	12288, 46080	1536, 100000	0.1

D.2 Traffic generators

Since our sources generate traffic greedily according to TSpec descriptions, some more parameters are required so that we do not quickly end up in the steady section of the TSpec curve but see some burstiness at the output. We therefore consider periods when a generator is idle and not producing any packets and when the generator is enabled. We call the corresponding intervals *burst length* and *burst spacing periods* respectively. The parameters used for our simulations are given in Table 4. $N(avg, dev)$ stands for a normal distribution with mean avg and standard deviation dev and $U(l, r)$ denotes a uniform distribution in the interval $[l, r)$. Packet lengths are rounded to the next integer. Packet lengths below 40 Byte are rounded up to 40 Byte and lengths above 1536 Byte are round off to 1536 Byte. The fact that we use sources with a mean packet length above the maximum packet length leads to sequences of packets with maximum length followed by just a single or a couple of packets of

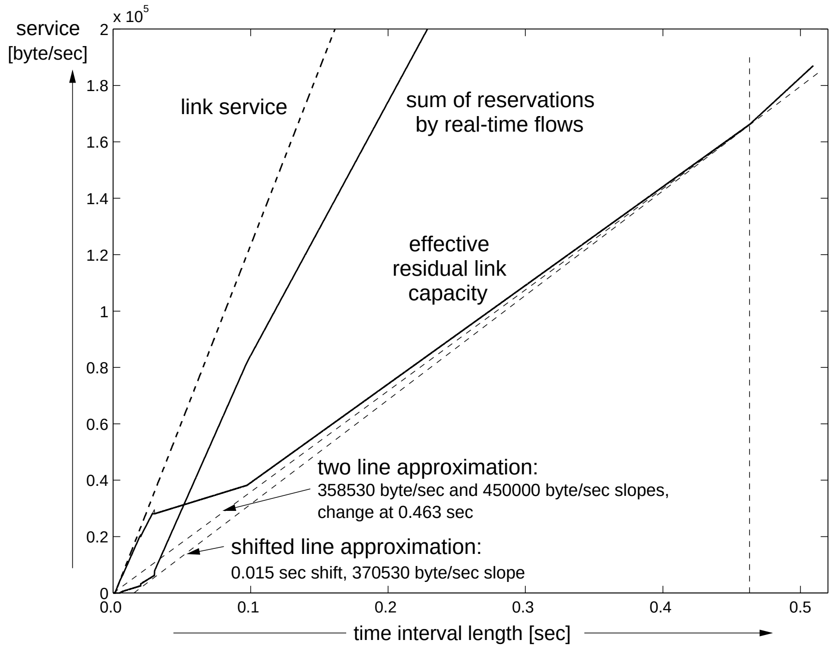

Figure 12: Deadline assignment approximations for best-effort traffic.

smaller length. This behavior imitates the transmission of files larger than the maximum packet length which are therefore distributed over several packets. Moreover, we allow all non real-time flows to excessively use the maximum packet length to increase the negative effect on real-time flows due to non-preemptive link scheduling.

Table 4: Traffic generator parameters.

flow class	*packet length* [byte]	*burst length* [ms]	*burst spacing* [ms]
RT transactions	$N(300, 50)$	$U(5, 35)$	$U(50, 800)$
RT video	$N(1700, 200)$	$U(50, 100)$	$U(10, 20)$
RT voice	100	$U(2, 4)$	$U(6, 10)$
NRT ftp	$N(1700, 200)$	$U(10, 700)$	$U(100, 300)$
NRT http	$N(1700, 200)$	$U(50, 400)$	$U(50, 200)$
NRT mail	$N(1700, 200)$	$U(5, 50)$	$U(100, 300)$