NETWORK CALCULUS

A Theory of Deterministic Queuing Systems for the Internet

JEAN-YVES LE BOUDEC
PATRICK THIRAN

Online Version of the Book Springer Verlag - LNCS 2050

Version January 16, 2002

A Annelies
A Joana, Maëlle, Audraine et Elias
A ma mère
—- *JL*

A mes parents
—- *PT*

Pour éviter les grumeaux
Qui encombrent les réseaux
Il fallait, c'est compliqué,
Maîtriser les seaux percés

Branle-bas dans les campus
On pourra dorénavant
Calculer plus simplement
Grâce à l'algèbre Min-Plus

Foin des obscures astuces
Pour estimer les délais
Et la gigue des paquets
Place à "Network Calculus"

—- *JL*

Summary of Changes

2002 Jan 14, JL Chapter 2: added a better coverage of GR nodes, in particular equivalence with service curve. Fixed bug in Proposition 1.4.1

2002 Jan 16, JL Chapter 6: M. Andrews brought convincing proof that conjecture 6.3.1 is wrong. Redesigned Chapter 6 to account for this. Removed redundancy between Section 2.4 and Chapter 6. Added SETF to Section 2.4

Contents

Introduction

What this Book is About

Network Calculus is a set of recent developments that provide deep insights into flow problems encountered in networking. The foundation of network calculus lies in the mathematical theory of dioids, and in particular, the Min-Plus dioid (also called Min-Plus algebra). With network calculus, we are able to understand some fundamental properties of integrated services networks, window flow control, scheduling and buffer or delay dimensioning.

This book is organized in three parts. Part I (Chapters 1 and 2) is a self contained, first course on network calculus. It can be used at the undergraduate level or as an entry course at the graduate level. The prerequisite is a first undergraduate course on linear algebra and one on calculus. Chapter 1 provides the main set of results for a first course: arrival curves, service curves and the powerful concatenation results are introduced, explained and illustrated. Practical definitions such as leaky bucket and generic cell rate algorithms are cast in their appropriate framework, and their fundamental properties are derived. The physical properties of shapers are derived. Chapter 2 shows how the fundamental results of Chapter 1 are applied to the Internet. We explain, for example, why the Internet integrated services internet can abstract any router by a rate-latency service curve. We also give a theoretical foundation to some bounds used for differentiated services.

Part II contains reference material that is used in various parts of the book. Chapter 3 contains all first level mathematical background. Concepts such as min-plus convolution and sub-additive closure are exposed in a simple way. Part I makes a number of references to Chapter 3, but is still self-contained. The role of Chapter 3 is to serve as a convenient reference for future use. Chapter 4 gives advanced min-plus algebraic results, which concern fixed point equations that are not used in Part I.

Part III contains advanced material; it is appropriate for a graduate course. Chapter 5 shows the application of network calculus to the determination of optimal playback delays in guaranteed service networks; it explains how fundamental bounds for multimedia streaming can be determined. Chapter 6 considers systems with aggregate scheduling. While the bulk of network calculus in this book applies to systems where schedulers are used to separate flows, there are still some interesting results that can be derived for such systems. Chapter 7 goes beyond the service curve definition of Chapter 1 and analyzes adaptive guarantees, as they are used by the Internet differentiated services. Chapter 8 analyzes time varying shapers; it is an extension of the fundamental results in Chapter 1 that considers the effect of changes in system parameters due to adaptive methods. An application is to renegotiable reserved services. Lastly, Chapter 9 tackles systems with losses. The fundamental result is a novel representation of losses in flow systems. This can be used to bound loss or congestion probabilities in complex systems.

Network calculus belongs to what is sometimes called “exotic algebras” or “topical algebras”. This is a set of mathematical results, often with high description complexity, that give insights into man-made systems such as concurrent programs, digital circuits and, of course, communication networks. Petri nets fall into this family as well. For a general discussion of this promising area, see the overview paper [32] and the book [26].

We hope to convince many readers that there is a whole set of largely unexplored, fundamental relations that can be obtained with the methods used in this book. Results such as “shapers keep arrival constraints” or “pay bursts only once”, derived in Chapter 1 have physical interpretations and are of practical importance to network engineers.

All results here are deterministic. Beyond this book, an advanced book on network calculus would explore the many relations between stochastic systems and the deterministic relations derived in this book. The interested reader will certainly enjoy the pioneering work in [26] and [11]. The appendix contains an index of the terms defined in this book.

Network Calculus, a System Theory for Computer Networks

In the rest of this introduction we highlight the analogy between network calculus and what is called “system theory”. You may safely skip it if you are not familiar with system theory.

Network calculus is a theory of *deterministic queuing* systems found in computer networks. It can also be viewed as the *system theory* that applies to computer networks. The main difference with traditional system theory, as the one that was so successfully applied to design electronic circuits, is that here we consider another algebra, where the operations are changed as follows: addition becomes computation of the minimum, multiplication becomes addition.

Before entering the subject of the book itself, let us briefly illustrate some of the analogies and differences between min-plus system theory, as applied in this book to communication networks, and traditional system theory, applied to electronic circuits.

Let us begin with a very simple circuit, such as the RC cell represented in Figure 1. If the input signal is the voltage $x(t) \in \mathbb{R}$, then the output $y(t) \in \mathbb{R}$ of this simple circuit is the convolution of x by the impulse response of this circuit, which is here $h(t) = \exp(-t/RC)/RC$ for $t \geq 0$:

$$y(t) = (h \otimes x)(t) = \int_0^t h(t-s)x(s)ds.$$

Consider now a node of a communication network, which is idealized as a (greedy) shaper. A (greedy) shaper is a device that forces an input flow $x(t)$ to have an output $y(t)$ that conforms to a given set of rates according to a traffic envelope σ (the shaping curve), at the expense of possibly delaying bits in the buffer. Here the input and output 'signals' are cumulative flow, defined as the number of bits seen on the data flow in time interval $[0,t]$. These functions are non-decreasing with time t. Parameter t can be continuous or discrete. We will see in this book that x and y are linked by the relation

$$y(t) = (\sigma \otimes x)(t) = \inf_{s \in \mathbb{R} \text{ such that } 0 \leq s \leq t} \{\sigma(t-s) + x(s)\}.$$

This relation defines the min-plus convolution between σ and x.

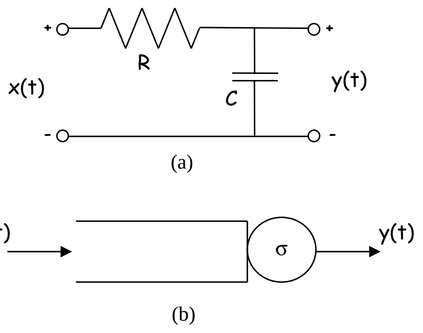

Figure 1: An RC circuit (a) and a greedy shaper (b), which are two elementary linear systems in their respective algebraic structures.

Convolution in traditional system theory is both commutative and associative, and this property allows to easily extend the analysis from small to large scale circuits. For example, the impulse response of the circuit of Figure 2(a) is the convolution of the impulse responses of each of the elementary cells:

$$h(t) = (h_1 \otimes h_2)(t) = \int_0^t h_1(t-s)h_2(s)ds.$$

The same property applies to greedy shapers, as we will see in Chapter 1. The output of the second shaper of Figure 2(b) is indeed equal to $y(t) = (\sigma \otimes x)(t)$, where

$$\sigma(t) = (\sigma_1 \otimes \sigma_2)(t) = \inf_{s\in\mathbb{R} \text{ such that } 0\leq s\leq t} \{\sigma_1(t-s) + \sigma_2(s)\}\,.$$

This will lead us to understand the phenomenon known as "pay burst only once" already mentioned earlier in this introduction.

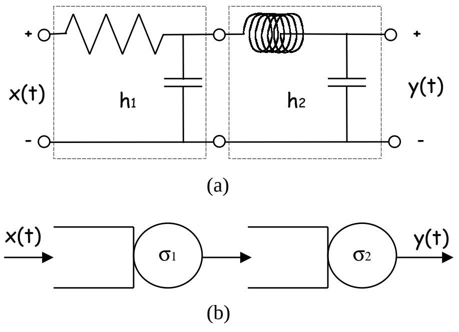

Figure 2: The impulse response of the concatenation of two linear circuit is the convolution of the individual impulse responses (a), the shaping curve of the concatenation of two shapers is the convolution of the individual shaping curves (b).

There are thus clear analogies between "conventional" circuit and system theory, and network calculus. There are however important differences too.

A first one is the response of a linear system to the sum of the inputs. This is a very common situation, in both electronic circuits (take the example of a linear low-pass filter used to clean a signal $x(t)$ from additive noise $n(t)$, as shown in Figure 3(a)), and in computer networks (take the example a link of a buffered node with output link capacity C, where one flow of interest $x(t)$ is multiplexed with other background traffic $n(t)$, as shown in Figure 3(b)).

Since the electronic circuit of Figure 3(a) is a linear system, the response to the sum of two inputs is the sum of the individual responses to each signal. Call $y(t)$ the response of the system to the pure signal $x(t)$, $y_n(t)$ the response to the noise $n(t)$, and $y_{tot}(t)$ the response to the input signal corrupted by noise $x(t) + n(t)$. Then $y_{tot}(t) = y(t) + y_n(t)$. This useful property is indeed exploited to design the optimal linear system that will filter out noise as much as possible.

If traffic is served on the outgoing link as soon as possible in the FIFO order, the node of Figure 3(b) is equivalent to a greedy shaper, with shaping curve $\sigma(t) = Ct$ for $t \geq 0$. It is therefore also a linear system, but this time in min-plus algebra. This means that the response to the minimum of two inputs is the minimum of the responses of the system to each input taken separately. However, this also mean that the response to the sum of two inputs is no longer the sum of the responses of

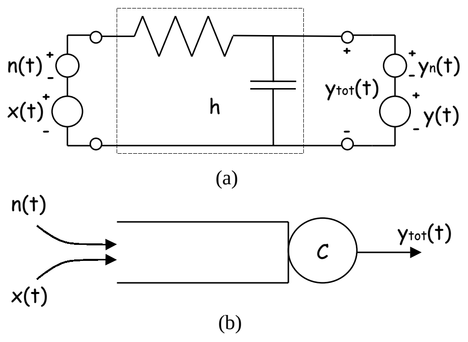

Figure 3: The response $y_{tot}(t)$ of a linear circuit to the sum of two inputs $x + n$ is the sum of the individual responses (a), but the response $y_{tot}(t)$ of a greedy shaper to the aggregate of two input flows $x + n$ is not the sum of the individual responses (b).

the system to each input taken separately, because now $x(t) + n(t)$ is a nonlinear operation between the two inputs $x(t)$ and $n(t)$: it plays the role of a multiplication in conventional system theory. Therefore the linearity property does unfortunately not apply to the aggregate $x(t) + n(t)$. As a result, little is known on the aggregate of multiplexed flows. Chapter 6 will learn us some new results and problems that appear simple but are still open today.

In both electronics and computer networks, nonlinear systems are also frequently encountered. They are however handled quite differently in circuit theory and in network calculus.

Consider an elementary nonlinear circuit, such as the BJT amplifier circuit with only one transistor, shown in Figure 4(a). Electronics engineers will analyze this nonlinear circuit by first computing a static operating point $y^\star$ for the circuit, when the input $x^\star$ is a fixed constant voltage (this is the DC analysis). Next they will linearize the nonlinear element (i.e the transistor) around the operating point, to obtain a so-called small signal model, which a linear model of impulse response $h(t)$ (this is the AC analysis). Now $x_{lin}(t) = x(t) - x^\star$ is a time varying function of time within a small range around $x^\star$, so that $y_{lin}(t) = y(t) - y^\star$ is indeed approximately given by $y_{lin}(t) \approx (h \otimes x_{lin})(t)$. Such a model is shown on Figure 4(b). The difficulty of a thorough nonlinear analysis is thus bypassed by restricting the input signal in a small range around the operating point. This allows to use a linearized model whose accuracy is sufficient to evaluate performance measures of interest, such as the gain of the amplifier.

In network calculus, we do not decompose inputs in a small range time-varying part and another large constant part. We do however replace nonlinear elements by linear systems, but the latter ones are now a lower bound of the nonlinear system. We

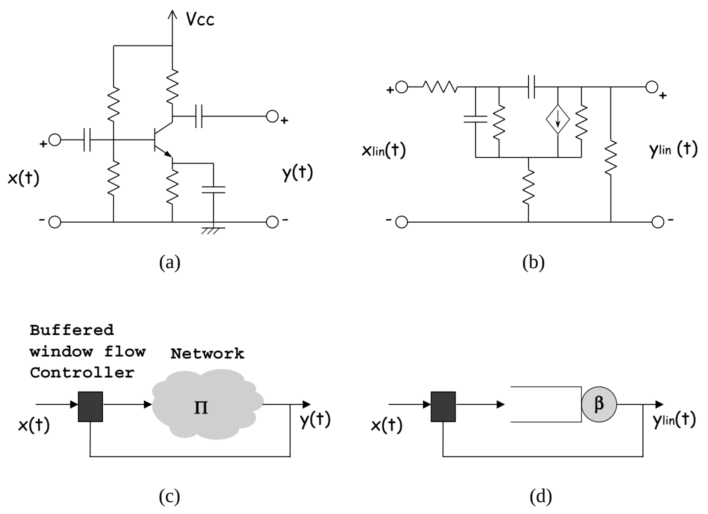

Figure 4: An elementary nonlinear circuit (a) replaced by a (simplified) linear model for small signals (b), and a nonlinear network with window flow control (c) replaced by a (worst-case) linear system (d).

will see such an example with the notion of service curve, in Chapter 1: a nonlinear system $y(t) = \Pi(x)(t)$ is replaced by a linear system $y_{lin}(t) = (\beta \otimes x)(t)$, where β denotes this service curve. This model is such that $y_{lin}(t) \leq y(t)$ for all $t \geq 0$, and all possible inputs $x(t)$. This will also allow us to compute performance measures, such as delays and backlogs in nonlinear systems. An example is the window flow controller illustrated in Figure 4(c), which we will analyze in Chapter 4. A flow x is fed via a window flow controller in a network that realizes some mapping $y = \Pi(x)$. The window flow controller limits the amount of data admitted in the network in such a way that the total amount of data in transit in the network is always less than some positive number (the window size). We do not know the exact mapping Π, we assume that we know one service curve β for this flow, so that we can replace the nonlinear system of Figure 4(c) by the linear system of Figure 4(d), to obtain deterministic bounds on the end-to-end delay or the amount of data in transit.

The reader familiar with traditional circuit and system theory will discover many other analogies and differences between the two system theories, while reading this book. We should insist however that no prerequisite in system theory is needed to discover network calculus as it is exposed in this book.

Acknowledgement

We gratefully acknowledge the pioneering work of Cheng-Shang Chang and René Cruz; our discussions with them have influenced this text. We thank Anna Charny, Silvia Giordano, Olivier Verscheure, Frédéric Worm, Jon Bennett, Kent Benson, Vicente Cholvi, William Courtney, Juan Echaguë, Felix Farkas, Gérard Hébuterne, Milan Vojnović and Zhi-Li Zhang for the fruitful collaboration. The interaction with Rajeev Agrawal, Matthew Andrews, François Baccelli, Guillaume Urvoy and Lothar Thiele is acknowledged with thanks. We are grateful to Holly Cogliati for helping with the preparation of the manuscript.

Part I

A First Course in Network Calculus

Chapter 1

Network Calculus

In this chapter we introduce the basic network calculus concepts of arrival, service curves and shapers. The application given in this chapter concerns primarily networks with reservation services such as ATM or the Internet integrated services (“Intserv”). Applications to other settings are given in the following chapters.

We begin the chapter by defining cumulative functions, which can handle both continuous and discrete time models. We show how their use can give a first insight into playout buffer issues, which will be revisited with more detail in Chapter 5. Then the concepts of Leaky Buckets and Generic Cell Rate algorithms are described in the appropriate framework, of arrival curves. We address in detail the most important arrival curves: piecewise linear functions and stair functions. Using the stair functions, we clarify the relation between spacing and arrival curve.

We introduce the concept of service curve as a common model for a variety of network nodes. We show that all schedulers generally proposed for ATM or the Internet integrated services can be modeled by a family of simple service curves called the rate-latency service curves. Then we discover physical properties of networks, such as “pay bursts only once” or “greedy shapers keep arrival constraints”. We also discover that greedy shapers are min-plus, time invariant systems. Then we introduce the concept of maximum service curve, which can be used to account for constant delays or for maximum rates. We illustrate all along the chapter how the results can be used for practical buffer dimensioning. We give practical guidelines for handling fixed delays such as propagation delays. We also address the distortions due to variability in packet size.

1.1 Models for Data Flows

1.1.1 Cumulative Functions, Discrete Time versus Continuous Time Models

It is convenient to describe data flows by means of the cumulative function $R(t)$, defined as the number of bits seen on the flow in time interval $[0, t]$. By convention, we take $R(0) = 0$, unless otherwise specified. Function R is always wide-sense increasing, that is, it belongs to the space $\mathcal{F}$ defined in Section 3.1.3 on Page 128. We can use a discrete or continuous time model. In real systems, there is always a minimum granularity (bit, word, cell or packet), therefore discrete time with a finite set of values for $R(t)$ could always be assumed. However, it is often computationally simpler to consider continuous time, with a function R that may be continuous or not. If $R(t)$ is a continuous function, we say that we have a *fluid model*. Otherwise, we take the convention that the function is either right or left-continuous (this makes little difference in practice).[1] Figure 1.1.1 illustrates these definitions.

Convention: A flow is described by a wide-sense increasing function $R(t)$; unless otherwise specified, in this book, we consider the following types of models:

- discrete time: $t \in \mathbb{N} = \{0, 1, 2, 3, ...\}$
- fluid model: $t \in \mathbb{R}^+ = [0, +\infty)$ and R is a continuous function
- general, continuous time model: $t \in \mathbb{R}^+$ and R is a left- or right-continuous function

---

If we assume that $R(t)$ has a derivative $\frac{dR}{dt} = r(t)$ such that $R(t) = \int_0^t r(s)ds$ (thus we have a fluid model), then r is called the rate function. Here, however, we will see that it is much simpler to consider cumulative functions such as R rather than rate functions. Contrary to standard algebra, with min-plus algebra we do not need functions to have "nice" properties such as having a derivative.

It is always possible to map a continuous time model $R(t)$ to a discrete time model $S(n), n \in \mathbb{N}$ by choosing a time slot δ and sampling by

$$S(n) = R(n\delta) \tag{1.1}$$

In general, this results in a loss of information. For the reverse mapping, we use the following convention. A continuous time model can be derived from $S(n), n \in \mathbb{N}$ by letting[2]

$$R'(t) = S(\lceil \frac{t}{\delta} \rceil) \tag{1.2}$$

[1]It would be nice to stick to either left- or right-continuous functions. However, depending on the model, there is no best choice: see Section 1.2.1 and Section 1.7

[2]$\lceil x \rceil$ ("ceiling of x") is defined as the smallest integer $\geq x$; for example $\lceil 2.3 \rceil = 3$ and $\lceil 2 \rceil = 2$

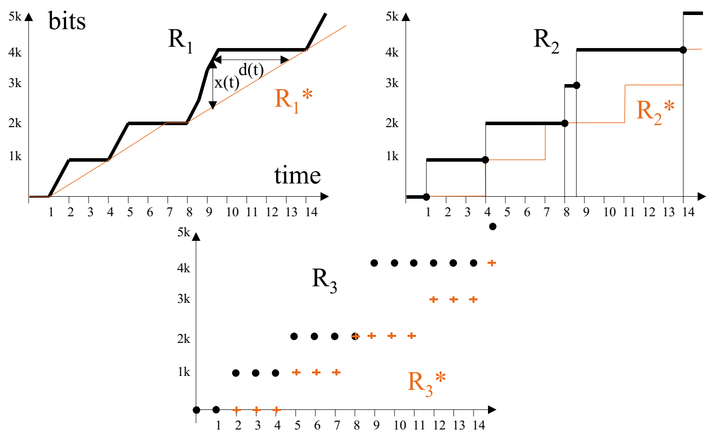

Figure 1.1: Examples of Input and Output functions, illustrating our terminology and convention. R_1 and R_1^* show a continuous function of continuous time (fluid model); we assume that packets arrive bit by bit, for a duration of one time unit per packet arrival. R_2 and R_2^* show continuous time with discontinuities at packet arrival times (times 1, 4, 8, 8.6 and 14); we assume here that packet arrivals are observed only when the packet has been fully received; the dots represent the value at the point of discontinuity; by convention, we assume that the function is left- or right-continuous. R_3 and R_3^* show a discrete time model; the system is observed only at times $0, 1, 2...$

The resulting function R' is always left-continuous, as we already required. Figure 1.1.1 illustrates this mapping with $\delta = 1$, $S = R_3$ and $R' = R_2$.

Thanks to the mapping in Equation (1.1), any result for a continuous time model also applies to discrete time. Unless otherwise stated, all results in this book apply to both continuous and discrete time. Discrete time models are generally used in the context of ATM; in contrast, handling variable size packets is usually done with a continuous time model (not necessarily fluid). Note that handling variable size packets requires some specific mechanisms, described in Section 1.7.

Consider now a system $\mathcal{S}$, which we view as a blackbox; $\mathcal{S}$ receives input data, described by its cumulative function $R(t)$, and delivers the data after a variable delay. Call $R^*(t)$ the *output function*, namely, the cumulative function at the output of system $\mathcal{S}$. System $\mathcal{S}$ might be, for example, a single buffer served at a constant rate, a complex communication node, or even a complete network. Figure 1.1.1 shows input and output functions for a single server queue, where every packet takes exactly 3 time units to be served. With output function R_1^* (fluid model) the assumption is that a packet can be served as soon as a first bit has arrived (cut-through assumption), and that a packet departure can be observed bit by bit, at a constant rate. For example, the first packet arrives between times 1 and 2, and leaves between times 1 and 4. With output function R_2^* the assumption is that a packet is served as soon as it has been fully received and is considered out of the system only when it is fully transmitted (store and forward assumption). Here, the first packet arrives immediately after time 1, and leaves immediately after time 4. With output function R_3^* (discrete time model), the first packet arrives at time 2 and leaves at time 5.

1.1.2 Backlog and Virtual Delay

From the input and output functions, we derive the two following quantities of interest.

Definition 1.1.1 (Backlog and Delay). *For a lossless system:*

- *The* backlog *at time t is $R(t) - R^*(t)$.*
- *The* virtual delay *at time t is*

$$d(t) = \inf\left\{\tau \geq 0 : R(t) \leq R^*(t+\tau)\right\}$$

The backlog is the amount of bits that are held inside the system; if the system is a single buffer, it is the queue length. In contrast, if the system is more complex, then the backlog is the number of bits "in transit", assuming that we can observe input and output simultaneously. The virtual delay at time t is the delay that would be experienced by a bit arriving at time t if all bits received before it are served before it. In Figure 1.1.1, the backlog, called $x(t)$, is shown as the vertical deviation between input and output functions. The virtual delay is the horizontal deviation. If the input and output function are continuous (fluid model), then it is easy to see that $R^*\left(t + d(t)\right) = R(t)$, and that $d(t)$ is the smallest value satisfying this equation.

In Figure 1.1.1, we see that the values of backlog and virtual delay slightly differ for the three models. Thus the delay experienced by the last bit of the first packet is $d(2) = 2$ time units for the first subfigure; in contrast, it is equal to $d(1) = 3$ time units on the second subfigure. This is of course in accordance with the different assumptions made for each of the models. Similarly, the delay for the fourth packet on subfigure 2 is $d(8.6) = 5.4$ time units, which corresponds to 2.4 units of waiting time and 3 units of service time. In contrast, on the third subfigure, it is equal to $d(9) = 6$ units; the difference is the loss of accuracy resulting from discretization.

1.1.3 Example: The Playout Buffer

Cumulative functions are a powerful tool for studying delays and buffers. In order to illustrate this, consider the simple playout buffer problem that we describe now. Consider a packet switched network that carries bits of information from a source with a constant bit rate r (Figure 1.2) as is the case for example, with circuit emulation. We take a fluid model, as illustrated in Figure 1.2. We have a first system $\mathcal{S}$, the network, with input function $R(t) = rt$. The network imposes some variable delay, because of queuing points, therefore the output R^* does not have a constant rate r. What can be done to recreate a constant bit stream ? A standard mechanism

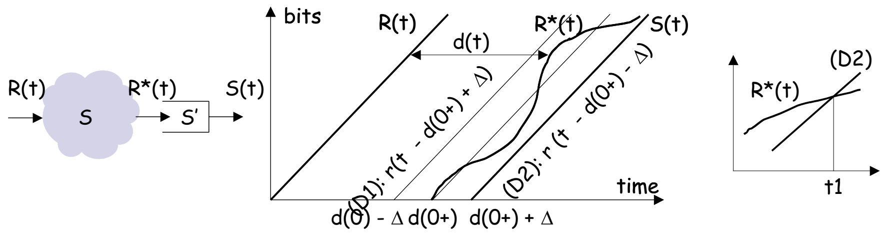

Figure 1.2: A Simple Playout Buffer Example

is to smooth the delay variation in a playout buffer. It operates as follows. When the first bit of data arrives, at time $d_r(0)$, where $d_r(0) = \lim_{t\to 0, t>0} d(t)$ is the limit to the right of function d[3], it is stored in the buffer until a fixed time Δ has elapsed. Then the buffer is served at a constant rate r whenever it is not empty. This gives us a second system $\mathcal{S}'$, with input R^* and output S.

Let us assume that the network delay variation is bounded by Δ. This implies that for every time t, the virtual delay (which is the real delay in that case) satisfies

[3]It is the virtual delay for a hypothetical bit that would arrive just after time 0. Other authors often use the notation $d(0+)$

$$-\Delta \leq d(t) - d_r(0) \leq \Delta$$

Thus, since we have a fluid model, we have

$$r(t - d_r(0) - \Delta) \leq R^*(t) \leq r(t - d_r(0) + \Delta)$$

which is illustrated in the figure by the two lines (D1) and (D2) parallel to $R(t)$. The figure suggests that, for the playout buffer $\mathcal{S}'$ the input function R^* is always above the straight line (D2), which means that the playout buffer never underflows. This suggests in turn that the output function $S(t)$ is given by $S(t) = r(t - d_r(0) - \Delta)$.

Formally, the proof is as follows. We proceed by contradiction. Assume the buffer starves at some time, and let t_1 be the first time at which this happens. Clearly the playout buffer is empty at time t_1, thus $R^*(t_1) = S(t_1)$. There is a time interval $[t_1, t_1 + \epsilon]$ during which the number of bits arriving at the playout buffer is less than $r\epsilon$ (see Figure 1.2. Thus, $d(t_1 + \epsilon) > d_r(0) + \Delta$ which is not possible. Secondly, the backlog in the buffer at time t is equal to $R^*(t) - S(t)$, which is bounded by the vertical deviation between (D1) and (D2), namely, $2r\Delta$.

We have thus shown that the playout buffer is able to remove the delay variation imposed by the network. We summarize this as follows.

Proposition 1.1.1. *Consider a constant bit rate stream of rate r, modified by a network that imposes a variable delay variation and no loss. The resulting flow is put into a playout buffer, which operates by delaying the first bit of the flow by Δ, and reading the flow at rate r. Assume that the delay variation imposed by the network is bounded by Δ, then*

1. *the playout buffer never starves and produces a constant output at rate r;*
2. *a buffer size of $2\Delta r$ is sufficient to avoid overflow.*

We study playout buffers in more details in Chapter 5, using the network calculus concepts further introduced in this chapter.

1.2 Arrival Curves

1.2.1 Definition of an Arrival Curve

Assume that we want to provide guarantees to data flows. This requires some specific support in the network, as explained in Section 1.3; as a counterpart, we need to limit the traffic sent by sources. With integrated services networks (ATM or the integrated services internet), this is done by using the concept of arrival curve, defined below.

Definition 1.2.1 (Arrival Curve). *Given a wide-sense increasing function α defined for $t \geq 0$ (namely, $\alpha \in \mathcal{F}$), we say that a flow R is constrained by α if and only if for all $s \leq t$:*

$$R(t) - R(s) \leq \alpha(t - s)$$

We say that R has α as an arrival curve, or also that R is α-smooth.

Note that the condition is over a set of overlapping intervals, as Figure 1.3 illustrates.

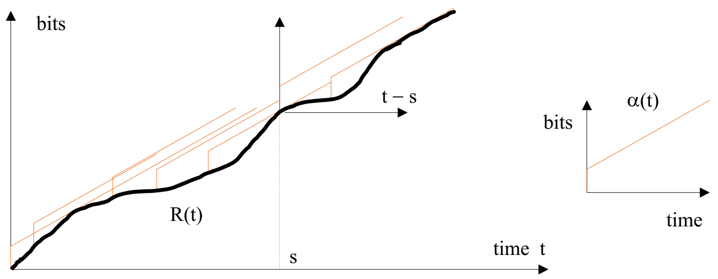

Figure 1.3: Example of Constraint by arrival curve, showing a cumulative function $R(t)$ constrained by the arrival curve $\alpha(t)$.

Affine Arrival Curves: For example, if $\alpha(t) = rt$, then the constraint means that, on any time window of width τ, the number of bits for the flow is limited by $r\tau$. We say in that case that the flow is peak rate limited. This occurs if we know that the flow is arriving on a link whose physical bit rate is limited by r b/s. A flow where the only constraint is a limit on the peak rate is often (improperly) called a "constant bit rate" (CBR) flow, or "deterministic bit rate" (DBR) flow.

Having $\alpha(t) = b$, with b a constant, as an arrival curve means that the maximum number of bits that may ever be sent on the flow is at most b.

More generally, because of their relationship with leaky buckets, we will often use *affine* arrival curves $\gamma_{r,b}$, defined by: $\gamma_{r,b}(t) = rt + b$ for $t > 0$ and 0 otherwise (see Section 3.1.3 for an illustration). Having $\gamma_{r,b}$ as an arrival curve allows a source to send b bits at once, but not more than r b/s over the long run. Parameters b and r are called the burst tolerance (in units of data) and the rate (in units of data per time unit). Figure 1.3 illustrates such a constraint.

Stair Functions as Arrival Curves: In the context of ATM, we also use arrival curves of the form $kv_{T,\tau}$, where $v_{T,\tau}$ is the stair functions defined by $v_{T,\tau}(t) = \lceil \frac{t+\tau}{T} \rceil$ for $t > 0$ and 0 otherwise (see Section 3.1.3 for an illustration). Note that $v_{T,\tau}(t) = v_{T,0}(t + \tau)$, thus $v_{T,\tau}$ results from $v_{T,0}$ by a time shift to the left. Parameter T (the "interval") and τ (the "tolerance") are expressed in time units. In order to understand the use of $v_{T,\tau}$, consider a flow that sends packets of a fixed size, equal to k unit of data (for example, an ATM flow). Assume that the packets are spaced by at least T time units. An example is a constant bit rate voice encoder, which generates packets periodically during talk spurts, and is silent otherwise. Such a flow has $kv_{T,0}$ as an arrival curve.

Assume now that the flow is multiplexed with some others. A simple way to think of this scenario is to assume that the packets are put into a queue, together with other flows. This is typically what occurs at a workstation, in the operating system or at the ATM adapter. The queue imposes a variable delay; assume it can be bounded by some value equal to τ time units. We will see in the rest of this chapter and in Chapter 2 how we can provide such bounds. Call $R(t)$ the input function for the flow at the multiplexer, and $R^*(t)$ the output function. We have $R^*(s) \leq R(s-\tau)$, from which we derive:

$$R^*(t) - R^*(s) \leq R(t) - R(s-\tau) \leq kv_{T,0}(t-s+\tau) = kv_{T,\tau}(t-s)$$

Thus R^* has $kv_{T,\tau}$ as an arrival curve. We have shown that *a periodic flow, with period T, and packets of constant size k, that suffers a variable delay $\leq \tau$, has $kv_{T,\tau}$ as an arrival curve*. The parameter τ is often called the "one-point cell delay variation", as it corresponds to a deviation from a periodic flow that can be observed at one point.

In general, function $v_{T,\tau}$ can be used to express *minimum spacing* between packets, as the following proposition shows.

Proposition 1.2.1 (Spacing as an arrival constraint). *Consider a flow, with cumulative function $R(t)$, that generates packets of constant size equal to k data units, with instantaneous packet arrivals. Assume time is discrete or time is continuous and R is left-continuous. Call t_n the arrival time for the nth packet. The following two properties are equivalent:*

1. *for all m, n, $t_{m+n} - t_m \geq nT - \tau$*
2. *the flow has $kv_{T,\tau}$ as an arrival curve*

The conditions on packet size and packet generation mean that $R(t)$ has the form nk, with $n \in \mathbb{N}$. The spacing condition implies that the time interval between two consecutive packets is $\geq T-\tau$, between a packet and the next but one is $\geq 2T-\tau$, etc.

Proof: Assume that property 1 holds. Consider an arbitrary interval $]s,t]$, and call n the number of packet arrivals in the interval. Say that these packets are numbered $m+1, \ldots, m+n$, so that $s < t_{m+1} \leq \ldots \leq t_{m+n} \leq t$, from which we have

$$t - s > t_{m+n} - t_{m+1}$$

Combining with property 1, we get

$$t - s > (n-1)T - \tau$$

From the definition of $v_{T,\tau}$ it follows that $v_{T,\tau}(t-s) \geq n$. Thus $R(t) - R(s) \leq kv_{T,\tau}(t-s)$, which shows the first part of the proof.

Conversely, assume now that property 2 holds. If time is discrete, we convert the model to continuous time using the mapping in Equation 1.2, thus we can consider that we are in the continuous time case. Consider some arbitrary integers m, n; for all $\epsilon > 0$, we have, under the assumption in the proposition:

$$R(t_{m+n} + \epsilon) - R(t_m) \geq (n+1)k$$

thus, from the definition of $v_{T,\tau}$,

$$t_{m+n} - t_m + \epsilon > nT - \tau$$

This is true for all $\epsilon > 0$, thus $t_{m+n} - t_m \geq nT - \tau$. □

In the rest of this section we clarify the relationship between arrival curve constraints defined by affine and by stair functions. First we need a technical lemma, which amounts to saying that we can always change an arrival curve to be left-continuous.

Lemma 1.2.1 (Reduction to left-continuous arrival curves). *Consider a flow $R(t)$ and a wide sense increasing function $\alpha(t)$, defined for $t \geq 0$. Assume that R is either left-continuous, or right-continuous. Denote with $\alpha_l(t)$ the limit to the left of α at t (this limit exists at every point because α is wide sense increasing); we have $\alpha_l(t) = \sup_{s<t} \alpha(s)$. If α is an arrival curve for R, then so is α_l.*

Proof: Assume first that R is left-continuous. For some $s < t$, let t_n be a sequence of increasing times converging towards t, with $s < t_n \leq t$. We have $R(t_n) - R(s) \leq \alpha(t_n - s) \leq \alpha_l(t - s)$. Now $\lim_{n \to +\infty} R(t_n) = R(t)$ since we assumed that R is left-continuous. Thus $R(t) - R(s) \leq \alpha_l(t - s)$.

If in contrast R is right-continuous, consider a sequence s_n converging towards s from above. We have similarly $R(t) - R(s_n) \leq \alpha(t - s_n) \leq \alpha_l(t - s)$ and $\lim_{n \to +\infty} R(s_n) = R(s)$, thus $R(t) - R(s) \leq \alpha_l(t - s)$ as well. □

Based on this lemma, we can always reduce an arrival curve to be left-continuous[4]. Note that $\gamma_{r,b}$ and $v_{T,\tau}$ are left-continuous. Also remember that, in this book, we use the convention that cumulative functions such as $R(t)$ are left continuous; this is a pure convention, we might as well have chosen to consider only right-continuous cumulative functions. In contrast, an arrival curve can always be assumed to be left-continuous, but not right-continuous.

In some cases, there is equivalence between a constraint defined by $\gamma_{r,b}$ and $v_{T,\tau}$. For example, for an ATM flow (namely, a flow where every packet has a fixed size equal to one unit of data) a constraint $\gamma_{r,b}$ with $r = \frac{1}{T}$ and $b = 1$ is equivalent to sending one packet every T time units, thus is equivalent to a constraint by the arrival curve $v_{T,0}$. In general, we have the following result.

Proposition 1.2.2. *Consider either a left- or right- continuous flow $R(t), t \in \mathbb{R}^+$, or a discrete time flow $R(t), t \in \mathbb{N}$, that generates packets of constant size equal to k data units, with instantaneous packet arrivals. For some T and τ, let $r = \frac{k}{T}$ and $b = k(\frac{\tau}{T} + 1)$. It is equivalent to say that R is constrained by $\gamma_{r,b}$ or by $kv_{T,\tau}$.*

[4]If we consider $\alpha_r(t)$, the limit to the right of α at t, then $\alpha \leq \alpha_r$ thus α_r is always an arrival curve, however it is not better than α.

Proof: Since we can map any discrete time flow to a left-continuous, continuous time flow, it is sufficient to consider a left-continuous flow $R(t), t \in \mathbb{R}^+$. Also, by changing the unit of data to the size of one packet, we can assume without loss of generality that $k = 1$. Note first, that with the parameter mapping in the proposition, we have $v_{T,\tau} \leq \gamma_{r,b}$, which shows that if $v_{T,\tau}$ is an arrival curve for R, then so is $\gamma_{r,b}$.

Conversely, assume now that R has $\gamma_{r,b}$ as an arrival curve. Then for all $s \leq t$, we have $R(t) - R(s) \leq rt + b$, and since $R(t) - R(s) \in \mathbb{N}$, this implies $R(t) - R(s) \leq \lfloor rt + b \rfloor$, Call $\alpha(t)$ the right handside in the above equation and apply Lemma 1.2.1. We have $\alpha_l(t) = \lceil rt + b - 1 \rceil = v_{T,\tau}(t)$. □

Note that the equivalence holds if we can assume that the packet size is constant and equal to the step size in the constraint $kv_{T,\tau}$. In general, the two families of arrival curve do not provide identical constraints. For example, consider an ATM flow, with packets of size 1 data unit, that is constrained by an arrival curve of the form $kv_{T,\tau}$, for some $k > 1$. This flow might result from the superposition of several ATM flows. You can convince yourself that this constraint cannot be mapped to a constraint of the form $\gamma_{r,b}$. We will come back to this example in Section 1.4.1.

1.2.2 Leaky Bucket and Generic Cell Rate Algorithm

Arrival curve constraints find their origins in the concept of leaky bucket and generic cell rate algorithms, which we describe now. We show that leaky buckets correspond to affine arrival curves $\gamma_{r,b}$, while the generic cell rate algorithm corresponds to stair functions $v_{T,\tau}$. For flows of fixed size packets, such as ATM cells, the two are thus equivalent.

Definition 1.2.2 (Leaky Bucket Controller). *A Leaky Bucket Controller is a device that analyzes the data on a flow $R(t)$ as follows. There is a pool (bucket) of fluid of size b. The bucket is initially empty. The bucket has a hole and leaks at a rate of r units of fluid per second when it is not empty.*

Data from the flow $R(t)$ has to pour into the bucket an amount of fluid equal to the amount of data. Data that would cause the bucket to overflow is declared non-conformant, otherwise the data is declared conformant.

Figure 1.2.2 illustrates the definition. Fluid in the leaky bucket does not represent data, however, it is counted in the same unit as data.

Data that is not able to pour fluid into the bucket is said to be "non-conformant" data. In ATM systems, non-conformant data is either discarded, tagged with a low priority for loss ("red" cells), or can be put in a buffer (buffered leaky bucket controller). With the Integrated Services Internet, non-conformant data is in principle not marked, but simply passed as best effort traffic (namely, normal IP traffic).

We want now to show that a leaky bucket controller enforces an arrival curve constraint equal to $\gamma_{r,b}$. We need the following lemma.

Lemma 1.2.2. *Consider a buffer served at a constant rate r. Assume that the buffer is empty at time 0. The input is described by the cumulative function $R(t)$. If there*

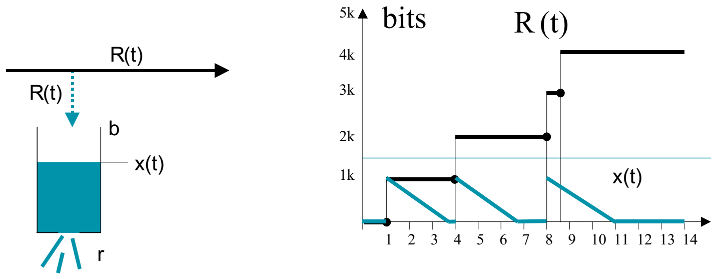

Figure 1.4: A Leaky Bucket Controller. The second part of the figure shows (in grey) the level of the bucket $x(t)$ for a sample input, with $r = 0.4$ kbits per time unit and $b = 1.5$ kbits. The packet arriving at time $t = 8.6$ is not conformant, and no fluid is added to the bucket. If b would be equal to 2 kbits, then all packets would be conformant.

is no overflow during $[0, t]$, *the buffer content at time* t *is given by*

$$x(t) = \sup_{s:s\leq t} \{R(t) - R(s) - r(t-s)\}$$

Proof: The lemma can be obtained as a special case of Corollary 1.5.2 on page 40, however we give here a direct proof. First note that for all s such that $s \leq t$, $(t-s)r$ is an upper bound on the number of bits output in $]s, t]$, therefore:

$$R(t) - R(s) - x(t) + x(s) \leq (t-s)r$$

Thus

$$x(t) \geq R(t) - R(s) + x(s) - (t-s)r \geq R(t) - R(s) - (t-s)r$$

which proves that $x(t) \geq \sup_{s:s\leq t}\{R(t) - R(s) - r(t-s)\}$.
Conversely, call t_0 the latest time at which the buffer was empty before time t:

$$t_0 = \sup\{s : s \leq t, x(s) = 0\}$$

(If $x(t) > 0$ then t_0 is the beginning of the busy period at time t). During $]t_0, t]$, the queue is never empty, therefore it outputs bit at rate r, and thus

$$x(t) = x(t_0) + R(t) - R(t_0) - (t - t_0)r \tag{1.3}$$

We assume that R is left-continuous (otherwise the proof is a little more complex); thus $x(t_0) = 0$ and thus $x(t) \leq \sup_{s:s\leq t}\{R(t) - R(s) - r(t-s)\}$ □

Now the content of a leaky bucket behaves exactly like a buffer served at rate r, and with capacity b. Thus, a flow $R(t)$ is conformant if and only if the bucket content $x(t)$ never exceeds b. From Lemma 1.2.2, this means that

$$\sup_{s:s\le t}\{R(t)-R(s)-r(t-s)\}\le b$$

which is equivalent to

$$R(t)-R(s)\le r(t-s)+b$$

for all $s \le t$. We have thus shown the following.

Proposition 1.2.3. *A leaky bucket controller with leak rate r and bucket size b forces a flow to be constrained by the arrival curve $\gamma_{r,b}$, namely:*

1. *the flow of conformant data has $\gamma_{r,b}$ as an arrival curve;*
2. *if the input already has $\gamma_{r,b}$ as an arrival curve, then all data is conformant.*

We will see in Section 1.4.1 a simple interpretation of the leaky bucket parameters, namely: r is the minimum rate required to serve the flow, and b is the buffer required to serve the flow at a constant rate.

Parallel to the concept of leaky bucket is the Generic Cell Rate Algorithm (GCRA), used with ATM.

Definition 1.2.3 (GCRA (T, τ)). *The Generic Cell Rate Algorithm (GCRA) with parameters (T, τ) is used with fixed size packets, called cells, and defines conformant cells as follows. It takes as input a cell arrival time* `t` *and returns* `result`*. It has an internal (static) variable* `tat` *(theoretical arrival time).*

- *initially,* `tat = 0`
- *when a cell arrives at time* `t`*, then*

```
if (t < tat - tau)
   result = NON-CONFORMANT;
else {
     tat = max (t, tat) + T;
     result = CONFORMANT;
     }
```

Table 1.1 illustrate the definition of GCRA. It illustrates that $\frac{1}{T}$ is the long term rate that can be sustained by the flow (in cells per time unit); while τ is a tolerance that quantifies how early cells may arrive with respect to an ideal spacing of T between cells. We see on the first example that cells may be early by 2 time units (cells arriving at times 18 to 48), however this may not be cumultated, otherwise the rate of $\frac{1}{T}$ would be exceeded (cell arriving at time 57).

In general, we have the following result, which establishes the relationship between GCRA and the stair functions $v_{T,\tau}$.

Proposition 1.2.4. *Consider a flow, with cumulative function $R(t)$, that generates packets of constant size equal to k data units, with instantaneous packet arrivals. Assume time is discrete or time is continuous and R is left-continuous. The following two properties are equivalent:*

arrival time	0	10	18	28	38	48	57
tat before arrival	0	10	20	30	40	50	60
result	c	c	c	c	c	c	non-c

arrival time	0	10	15	25	35
tat before arrival	0	10	20	20	30
result	c	c	non-c	c	c

Table 1.1: Examples for GCRA(10,2). The table gives the cell arrival times, the value of the `tat` internal variable just before the cell arrival, and the result for the cell (c = conformant, non-c = non-conformant).

1. *the flow is conformant to GCRA(T, τ)*
2. *the flow has $(k\ v_{T,\tau})$ as an arrival curve*

Proof: The proof uses max-plus algebra. Assume that property 1 holds. Denote with θ_n the value of `tat` just after the arrival of the nth packet (or cell), and by convention $\theta_0 = 0$. Also call t_n the arrival time of the nth packet. From the definition of the GCRA we have $\theta_n = \max(t_n, \theta_{n-1}) + T$. We write this equation for all $m \leq n$, using the notation $\vee$ for $\max$. The distributivity of addition with respect to $\vee$ gives:

$$\begin{cases} \theta_n = (\theta_{n-1} + T) \vee (t_n + T) \\ \theta_{n-1} + T = (\theta_{n-2} + 2T) \vee (t_{n-1} + 2T) \\ \ldots \\ \theta_1 + (n-1)T = (\theta_0 + nT) \vee (t_1 + nT) \end{cases}$$

Note that $(\theta_0 + nT) \vee (t_1 + nT) = t_1 + nT$ because $\theta_0 = 0$ and $t_1 \geq 0$, thus the last equation can be simplified to $\theta_1 + (n-1)T = t_1 + nT$. Now the iterative substitution of one equation into the previous one, starting from the last one, gives

$$\theta_n = (t_n + T) \vee (t_{n-1} + 2T) \vee \ldots \vee (t_1 + nT) \tag{1.4}$$

Now consider the $(m+n)$th arrival, for some $m, n \in \mathbb{N}$, with $m \geq 1$. By property 1, the packet is conformant, thus

$$t_{m+n} \geq \theta_{m+n-1} - \tau \tag{1.5}$$

Now from Equation (1.4), $\theta_{m+n-1} \geq t_j + (m+n-j)T$ for all $1 \leq j \leq m+n-1$. For $j = m$, we obtain $\theta_{m+n-1} \geq t_m + nT$. Combining this with Equation (1.5), we have $t_{m+n} \geq t_m + nT - \tau$. With proposition 1.2.1, this shows property 2.

Conversely, assume now that property 2 holds. We show by induction on n that the nth packet is conformant. This is always true for $n = 1$. Assume it is true for all $m \leq n$. Then, with the same reasoning as above, Equation (1.4) holds for n. We rewrite it as $\theta_n = \max_{1 \leq j \leq n}\{t_j + (n - j + 1)T\}$. Now from proposition 1.2.1,

$t_{n+1} \geq t_j + (n-j+1)T - \tau$ for all $1 \leq j \leq n$, thus $t_{n+1} \geq \max_{1 \leq j \leq n}\{t_j + (n-j+1)T\} - \tau$. Combining the two, we find that $t_{n+1} \geq \theta_n - \tau$, thus the $(n+1)$th packet is conformant. $\square$

Note the analogy between Equation (1.4) and Lemma 1.2.2. Indeed, from proposition 1.2.2, for packets of constant size, there is equivalence between arrival constraints by affine functions $\gamma_{r,b}$ and by stair functions $v_{T,\tau}$. This shows the following result.

Corollary 1.2.1. *For a flow with packets of constant size, satisfying the GCRA(T, τ) is equivalent to satisfying a leaky bucket controller, with rate r and burst tolerance b given by:*

$$b = (\frac{\tau}{T} + 1)\delta$$

$$r = \frac{\delta}{T}$$

In the formulas, δ is the packet size in units of data.

The corollary can also be shown by a direct equivalence of the GCRA algorithm to a leaky bucket controller.

Take the ATM cell as unit of data. The results above show that for an ATM cell flow, being conformant to GCRA(T, τ) is equivalent to having $v_{T,\tau}$ as an arrival curve. It is also equivalent to having $\gamma_{r,b}$ as an arrival curve, with $r = \frac{1}{T}$ and $b = \frac{\tau}{T} + 1$.

Consider a family of I leaky bucket controllers (or GCRAs), with parameters r_i, b_i, for $1 \leq i \leq I$. If we apply all of them in parallel to the same flow, then the conformant data is data that is conformant for each of the controllers in isolation. The flow of conformant data has as an arrival curve

$$\alpha(t) = \min_{1 \leq i \leq I}(\gamma_{r_i,b_i}(t)) = \min_{1 \leq i \leq I}(r_i t + b_i)$$

It can easily be shown that the family of arrival curves that can be obtained in this way is the set of concave, piecewise linear functions, with a finite number of pieces. We will see in Section 1.5 some examples of functions that do not belong to this family.

Application to ATM and the Internet Leaky buckets and GCRA are used by standard bodies to define conformant flows in Integrated Services Networks. With ATM, a constant bit rate connection (CBR) is defined by one GCRA (or equivalently, one leaky bucket), with parameters (T, τ). T is called the ideal cell interval, and τ is called the Cell Delay Variation Tolerance (CDVT). Still with ATM, a variable bit rate (VBR) connection is defined as one connection with an arrival curve that corresponds to 2 leaky buckets or GCRA controllers. The Integrated services framework of the Internet (Intserv) uses the same family of arrival curves, such as

$$\alpha(t) = \min(M + pt, rt + b) \tag{1.6}$$

where M is interpreted as the maximum packet size, p as the peak rate, b as the burst tolearance, and r as the sustainable rate (Figure 1.5). In Intserv jargon, the 4-uple (p, M, r, b) is also called a T-SPEC (traffic specification).

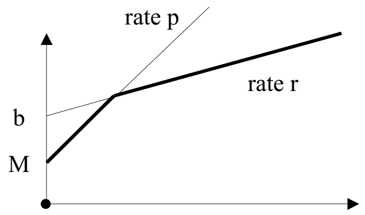

Figure 1.5: Arrival curve for ATM VBR and for Intserv flows

1.2.3 Sub-additivity and Arrival Curves

In this Section we discover the fundamental relationship between min-plus algebra and arrival curves. Let us start with a motivating example.

Consider a flow $R(t) \in \mathbb{N}$ with $t \in \mathbb{N}$; for example the flow is an ATM cell flow, counted in cells. Time is discrete to simplify the discussion. Assume that we know that the flow is constrained by the arrival curve $3v_{10,0}$; for example, the flow is the superposition of 3 CBR connections of peak rate 0.1 cell per time unit each. Assume in addition that we know that the flow arrives at the point of observation over a link with a physical characteristic of 1 cell per time unit. We can conclude that the flow is also constrained by the arrival curve $v_{1,0}$. Thus, obviously, it is constrained by $\alpha_1 = \min(3v_{10,0}, v_{1,0})$. Figure 1.6 shows the function α_1.

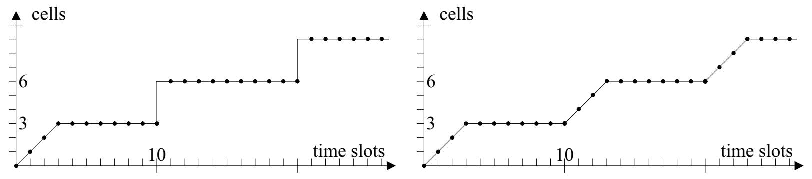

Figure 1.6: The arrival curve $\alpha_1 = \min(3v_{10,0}, v_{1,0})$ on the left, and its sub-additive closure (“good” function) $\bar{\alpha_1}$ on the right. Time is discrete, lines are put for ease of reading.

Now the arrival curve α_1 tells us that $R(10) \leq 3$ and $R(11) \leq 6$. However, since there can arrive at most 1 cell per time unit , we can also conclude that $R(11) \leq R(10) + [R(11) - R(10)] \leq \alpha_1(10) + \alpha_1(1) = 4$. In other words, the sheer knowledge that R is constrained by α_1 allows us to derive a better bound than α_1 itself. This is because α_1 is not a “good” function, in a sense that we define now.

Definition 1.2.4. *Consider a function α in $calF$. We say that α is a “good” function if any one of the following equivalent properties is satisfied*

1. *α is sub-additive and $\alpha(0) = 0$*
2. *$\alpha = \alpha \otimes \alpha$*
3. *$\alpha \oslash \alpha = \alpha$*
4. *$\alpha = \bar{\alpha}$ (sub-additive closure of α).*

The definition uses the concepts of sub-additivity, min-plus convolution, min-plus deconvolution and sub-additive closure, which are defined in Chapter 3. The equivalence between the four items comes from Corollaries 3.1.1 on page 144 and 3.1.13 on page 151. Sub-additivity (item 1) means that $\alpha(s+t) \leq \alpha(s) + \alpha(t)$. If α is not sub-additive, then $\alpha(s) + \alpha(t)$ may be a better bound than $\alpha(s+t)$, as is the case with α_1 in the example above. Item 2, 3 and 4 use the concepts of min-plus convolution, min-plus deconvolution and sub-additive closure, defined in Chapter 3. We know in particular (Theorem 3.1.10) that the sub-additive closure of a function α is the largest “good” function $\bar{\alpha}$ such that $\bar{\alpha} \leq \alpha$. We also know that $\bar{\alpha} \in \mathcal{F}$ if $\alpha \in \mathcal{F}$.

The main result about arrival curves is that *any* arrival curve can be replaced by its sub-additive closure, which is a “good” arrival curve. Figure 1.6 shows $\bar{\alpha_1}$ for our example above.

Theorem 1.2.1 (Reduction of Arrival Curve to a Sub-Additive One). *Saying that a flow is constrained by a wide-sense increasing function α is equivalent to saying that it is constrained by the sub-additive closure $\bar{\alpha}$.*

The proof of the theorem leads us to the heart of the concept of arrival curve, namely, its correspondence with a fundamental, linear relationships in min-plus algebra, which we will now derive.

Lemma 1.2.3. *A flow R is constrained by arrival curve α if and only if $R \leq R \otimes \alpha$*

Proof: Remember that an equation such as $R \leq R \otimes \alpha$ means that for all times t, $R(t) \leq (R \otimes \alpha)(t)$. The min-plus convolution $R \otimes \alpha$ is defined in Chapter 3, page 134; since $R(s)$ and $\alpha(s)$ are defined only for $s \geq 0$, the definition of $R \otimes \alpha$ is: $(R \otimes \alpha)(t) = \inf_{0 \leq s \leq t}(R(s) + \alpha(t-s))$. Thus $R \leq R \otimes \alpha$ is equivalent to $R(t) \leq R(s) + \alpha(t-s)$ for all $0 \leq s \leq t$. □

Lemma 1.2.4. *If α_1 and α_2 are arrival curves for a flow R, then so is $\alpha_1 \otimes \alpha_2$*

Proof: We know from Chapter 3 that $\alpha_1 \otimes \alpha_2$ is wide-sense increasing if α_1 and α_2 are. The rest of the proof follows immediately from Lemma 1.2.3 and the associativity of $\otimes$. $\square$

Proof of Theorem Since α is an arrival curve, so is $\alpha \otimes \alpha$, and by iteration, so is $\alpha^{(n)}$ for all $n \geq 1$. By the definition of δ_0, it is also an arrival curve. Thus so is $\bar{\alpha} = \inf_{n \geq 0} \alpha^{(n)}$.

Conversely, $\alpha \leq \bar{\alpha}$; thus, if $\bar{\alpha}$ is an arrival curve, then so is α. $\square$

Examples We should thus restrict our choice of arrival curves to sub-additive functions. As we can expect, the functions $\gamma_{r,b}$ and $v_{T,\tau}$ introduced in Section 1.2.1 are sub-additive and since their value is 0 for $t = 0$, they are "good" functions, as we now show. Indeed, we know from Chapter 1 that any concave function α such that $\alpha(0) = 0$ is sub-additive. This explains why the functions $\gamma_{r,b}$ are sub-additive.

Functions $v_{T,\tau}$ are not concave, but they still are sub-additive. This is because, from its very definition, the ceiling function is sub-additive, thus

$$v_{T,\tau}(s+t) = \lceil \frac{s+t+\tau}{T} \rceil \leq \lceil \frac{s+\tau}{T} \rceil + \lceil \frac{t}{T} \rceil \leq \lceil \frac{s+\tau}{T} \rceil + \lceil \frac{t+\tau}{T} \rceil = v_{T,\tau}(s) + v_{T,\tau}(t)$$

Let us return to our introductory example with $\alpha_1 = \min(3v_{10,0}, v_{1,0})$. As we discussed, α_1 is not sub-additive. From Theorem 1.2.1, we should thus replace α_1 by its sub-additive closure $\bar{\alpha_1}$, which can be computed by Equation (3.13). The computation is simplified by the following remark, which follows immediately from Theorem 3.1.11:

Lemma 1.2.5. *Let γ_1 and γ_2 be two "good" functions. The sub-additive closure of $\min(\gamma_1, \gamma_2)$ is $\gamma_1 \otimes \gamma_2$.*

We can apply the lemma to $\alpha_1 = 3v_{10,0} \wedge v_{1,0}$, since $v_{T,\tau}$ is a "good" function. Thus $\bar{\alpha_1} = 3v_{10,0} \otimes v_{1,0}$, which the alert reader will enjoy computing. The result is plotted in Figure 1.6.

Finally, let us mention the following equivalence, the proof of which is easy and left to the reader.

Proposition 1.2.5. *For a given wide-sense increasing function α, with $\alpha(0) = 0$, consider a source defined by $R(t) = \alpha(t)$ (greedy source). The source has α as an arrival curve if and only if α is a "good" function.*

VBR arrival curve Now let us examine the family of arrival curves obtained by combinations of leaky buckets or GCRAs (concave piecewise linear functions). We know from Chapter 3 that if γ_1 and γ_2 are concave, with $\gamma_1(0) = \gamma_2(0) = 0$, then $\gamma_1 \otimes \gamma_2 = \gamma_1 \wedge \gamma_2$. Thus any concave piecewise linear function α such that $\alpha(0) = 0$ is a "good" function. In particular, if we define the arrival curve for VBR connections or Intserv flows by

$$\begin{cases} \alpha(t) = \min(pt + M, rt + b) & \text{if } t > 0 \\ \alpha(0) = 0 \end{cases}$$

(see Figure 1.5) then α is a "good" function.

We have seen in Lemma 1.2.1 that an arrival curve α can always be replaced by its limit to the left α_l. We might wonder how this combines with the sub-additive closure, and in particular, whether these two operations commute (in other words, do we have $(\bar{\alpha})_l = \overline{\alpha_l}$?). In general, if α is left-continuous, then we cannot guarantee that $\bar{\alpha}$ is also left-continuous, thus we cannot guarantee that the operations commute. However, it can be shown that $(\bar{\alpha})_l$ is always a "good" function, thus $\overline{(\bar{\alpha})_l} = (\bar{\alpha})_l$. Starting from an arrival curve α we can therefore improve by taking the sub-additive closure first, then the limit to the left. The resulting arrival curve $(\bar{\alpha})_l$ is a "good" function that is also left-continuous (a "very good" function), and the constraint by α is equivalent to the constraint by $(\bar{\alpha})_l$

Lastly, let us mention that it can easily be shown, using an argument of uniform continuity, that if α takes only a finite set of values over any bounded time interval, and if α is left-continuous, then so is $\bar{\alpha}$ and then we do have $(\bar{\alpha})_l = \overline{\alpha_l}$. This assumption is always true in discrete time, and in most cases in practice.

1.2.4 Minimum Arrival Curve

Consider now a given flow $R(t)$, for which we would like to determine a minimal arrival curve. This problem arises, for example, when R is known from measurements. The following theorem says that there is indeed one minimal arrival curve.

Theorem 1.2.2 (Minimum Arrival Curve). *Consider a flow $R(t)_{t\geq 0}$. Then*

- *function $R \oslash R$ is an arrival curve for the flow*
- *for any arrival curve α that constrains the flow, we have: $(R \oslash R) \leq \alpha$*
- *$R \oslash R$ is a "good" function*

Function $R \oslash R$ is called the minimum arrival curve *for flow R.*

The minimum arrival curve uses min-plus deconvolution, defined in Chapter 3. Figure 1.2.4 shows an example of $R \oslash R$ for a measured function R.

Proof: By definition of $\oslash$, we have $(R \oslash R)(t) = \sup_{v\geq 0}\{R(t+v) - R(v)\}$, it follows that $(R \oslash R)$ is an arrival curve.

Now assume that some α is also an arrival curve for R. From Lemma 1.2.3, we have $R \leq R \otimes \alpha)$. From Rule 14 in Theorem 3.1.12 in Chapter 3, it follows that $R \oslash R \leq \alpha$, which shows that $R \oslash R$ is the minimal arrival curve for R. Lastly, $R \oslash R$ is a "good" function from Rule 15 in Theorem 3.1.12. □

Consider a greedy source, with $R(t) = \alpha(t)$, where α is a "good" function. What is the minimum arrival curve ?[5] Lastly, the curious reader might wonder

[5]Answer: from the equivalence in Definition 1.2.4, the minimum arrival curve is α itself.

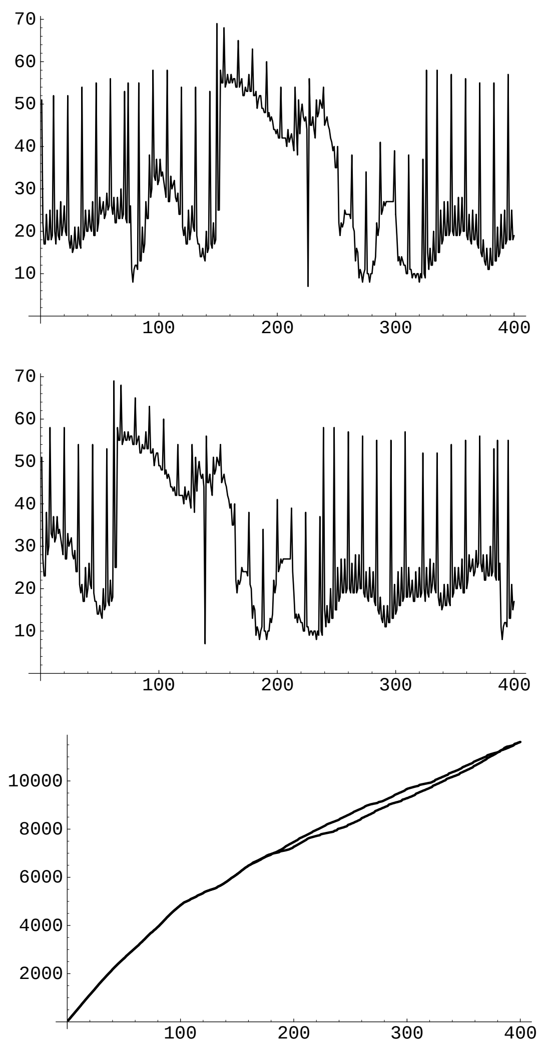

Figure 1.7: Example of minimum arrival curve. Time is discrete, one time unit is 40 ms. The top figures shows, for two similar traces, the number of packet arrivals at every time slot. Every packet is of constant size (416 bytes). The bottom figure shows the minimum arrival curve for the first trace (top curve) and the second trace (bottom curve). The large burst in the first trace comes earlier, therefore its minimum arrival curve is slightly larger.

whether $R \oslash R$ is left-continuous. The answer is as follows. Assume that R is either right or left-continuous. By lemma 1.2.1, the limit to the left $(R \oslash R)_l$ is also an arrival curve, and is bounded from above by $R \oslash R$. Since $R \oslash R$ is the minimum arrival curve, it follows that $(R \oslash R)_l = R \oslash R$, thus $R \oslash R$ is left-continuous (and is thus a "very good" function).

In many cases, one is interested not in the absolute minimum arrival curve as presented here, but in a minimum arrival curve within a family of arrival curves, for example, among all $\gamma_{r,b}$ functions. For a development along this line, see [58].

1.3 Service Curves

1.3.1 Definition of Service Curve

We have seen that one first principle in integrated services networks is to put arrival curve constraints on flows. In order to provide reservations, network nodes in return need to offer some guarantees to flows. This is done by packet schedulers [41]. The details of packet scheduling are abstracted using the concept of service curve, which we introduce and study in this section. Since the concept of service curve is more abstract than that of arrival curve, we introduce it on some examples.

A first, simple example of a scheduler is a Generalized Processor Sharing (GPS) node [60]. We define now a simple view of GPS; more details are given in Chapter 2. A GPS node serves several flows in parallel, and we can consider that every flow is allocated a given rate. The guarantee is that during a period of duration t, for which a flow has some backlog in the node, it receives an amount of service at least equal to rt, where r is its allocated rate. A GPS node is a theoretical concept, which is not really implementable, because it relies on a fluid model, while real networks use packets. We will see in Section 2.1 on page 83 how to account for the difference between a real implementation and GPS. Consider a input flow R, with output R^*, that is served in a GPS node, with allocated rate r. Let us also assume that the node buffer is large enough so that overflow is not possible. We will see in this section how to compute the buffer size required to satisfy this assumption. Lossy systems are the object of Chapter 9. Under these assumptions, for all time t, call t_0 the beginning of the last busy period for the flow up to time t. From the GPS assumption, we have

$$R^*(t) - R^*(t_0) \geq r(t - t_0)$$

Assume as usual that R is left-continuous; at time t_0 the backlog for the flow is 0, which is expressed by $R(t_0) - R^*(t_0) = 0$. Combining this with the previous equation, we obtain:

$$R^*(t) - R(t_0) \geq r(t - t_0)$$

We have thus shown that, for all time t: $R^*(t) \geq \inf_{0 \leq s \leq t}[R(s) + r(t - s)]$, which can be written as

$$R^* \geq R \otimes \gamma_{r,0} \tag{1.7}$$

Note that a limiting case of GPS node is the constant bit rate server with rate r, dedicated to serving a single flow. We will study GPS in more details in Chapter 2.

Consider now a second example. Assume that the only information we have about a network node is that the maximum delay for the bits of a given flow R is bounded by some fixed value T, and that the bits of the flow are served in first in, first out order. We will see in Section 1.5 that this is used with a family of schedulers called "earliest deadline first" (EDF). We can translate the assumption on the delay bound to $d(t) \leq T$ for all t. Now since R^* is always wide-sense increasing, it follows from the definition of $d(t)$ that $R^*(t+T) \geq R(t)$. Conversely, if $R^*(t+T) \geq R(t)$, then $d(t) \leq T$. In other words, our condition that the maximum delay is bounded by T is equivalent to $R^*(t+T) \geq R(t)$ for all t. This in turn can be re-written as

$$R^*(s) \geq R(s-T)$$

for all $s \geq T$. We have introduced in Chapter 3 the "impulse" function δ_T defined by $\delta_T(t) = 0$ if $0 \leq t \leq T$ and $\delta_T(t) = +\infty$ if $t > T$. It has the property that, for any wide-sense increasing function $x(t)$, defined for $t \leq 0$, $(x \otimes \delta_T)(t) = x(t-T)$ if $t \geq T$ and $(x \otimes \delta_T)(t) = x(0)$ otherwise. Our condition on the maximum delay can thus be written as

$$R^* \geq R \otimes \delta_T \tag{1.8}$$

For the two examples above, there is an input-output relationship of the same form (Equations (1.7) and (1.8)). This suggests the definition of service curve, which, as we see in the rest of this section, is indeed able to provide useful results.

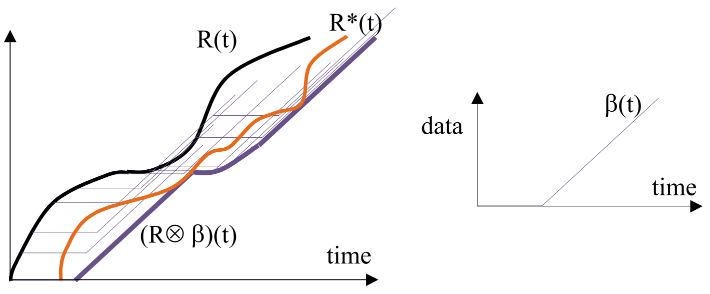

Figure 1.8: Definition of service curve. The output R^* must be above $R \otimes \beta$, which is the lower envelope of all curves $t \mapsto R(t_0) + \beta(t - t_0)$.

Definition 1.3.1 (Service Curve). *Consider a system $\mathcal{S}$ and a flow through $\mathcal{S}$ with input and output function R and R^*. We say that $\mathcal{S}$ offers to the flow a* service curve *β if and only if $\beta \in \mathcal{F}$ and $R^* \geq R \otimes \beta$*

Figure 1.8 illustrates the definition.

The definition means that β is a wide sense increasing function, with $\beta(0) = 0$, and that for all $t \geq 0$,

$$R^*(t) \geq \inf_{s \leq t} (R(s) + \beta(t - s))$$

In practice, we can avoid the use of an infimum if β is continuous. The following proposition is an immediate consequence of Theorem 3.1.8 on Page 139.

Proposition 1.3.1. *If β is continuous, the service curve property means that for all t we can find $t_0 \leq t$ such that*

$$R^*(t) \geq R_l(t_0) + \beta(t - t_0) \tag{1.9}$$

where $R_l(t_0) = \sup_{\{s<t_0\}} R(s)$ is the limit to the left of R at t_0. If R is left-continuous, then $R_l(t_0) = R(t_0)$.

For a constant rate server (and also for any *strict* service curve), the number t_0 in Equation (1.9) is the beginning of the busy period. For other cases, there is not such a simple definition. However, in some cases we can make sure that t_0 increases with t:

Proposition 1.3.2. *If the service curve β is convex, then we can find some wide sense increasing function $\tau(t)$ such that we can choose $t_0 = \tau(t)$ in Equation (1.9).*

Note that since a service curve is assumed to be wide-sense increasing, β, being convex, is necessarily continuous; thus we can apply Proposition 1.3.1.

Proof: We give the proof when R is left-continuous. The proof for the general case is essentially the same but involves some ϵ cutting. Consider some $t_1 < t_2$ and call τ_1 a value of t_0 as in Equation (1.9)) at $t = t_1$. Also consider any $t' \leq \tau_1$. From the definition of τ_1, we have

$$R^*(t') + \beta(t_1 - t') \geq R^*(\tau_1) + \beta(t_1 - \tau_1)$$

and thus

$$R^*(t') + \beta(t_2 - t') \geq R^*(\tau_1) + \beta(t_1 - \tau_1) - \beta(t_1 - t') + \beta(t_2 - t')$$

Now β is convex, thus for any four numbers a, b, c, d such that $a \leq c \leq b$, $a \leq d \leq b$ and $a + b = c + d$, we have

$$\beta(a) + \beta(b) \geq \beta(c) + \beta(d)$$

(the interested reader will be convinced by drawing a small figure). Applying this to $a = t_1 - \tau_1, b = t_2 - t', c = t_1 - t', d = t_2 - \tau_1$ gives

$$R^*(t') + \beta(t_2 - t') \geq R^*(\tau_1) + \beta(t_2 - \tau_1)$$

and the above equation holds for all $t' \leq \tau_1$. Consider now the minimum, for a fixed t_2, of $R^*(t') + \beta(t_2 - t')$ over all $t' \leq t_2$. The above equation shows that the minimum is reached for some $t' \geq \tau_1$. $\square$

We will see in Section 1.4 that the combination of a service curve guarantee with an arrival curve constraint forms the basis for deterministic bounds used in integrated services networks. Before that, we give the fundamental service curve examples that are used in practice.

1.3.2 Classical Service Curve Examples

Guaranteed Delay Node The analysis of the second example in Section 1.3.1 can be rephrased as follows.

Proposition 1.3.3. *For a lossless bit processing system, saying that the delay for any bit is bounded by some fixed T is equivalent to saying that the system offers to the flow a service curve equal to δ_T.*

Non Premptive Priority Node Consider a node that serves two flows, $R_H(t)$ and $R_L(t)$. The first flow has non-preemptive priority over the second one (Figure 1.9). This example explains the general framework used when some traffic classes have priority over some others, such as with the Internet differentiated services [7]. The rate of the server is constant, equal to C. Call $R^*_H(t)$ and $R^*_L(t)$ the outputs for the two flows. Consider first the high priority flow. Fix some time t and call s the

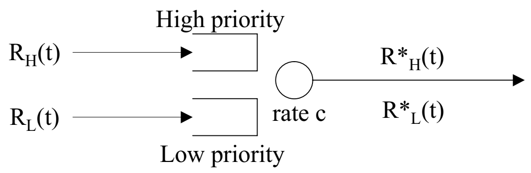

Figure 1.9: Two priority flows (H and L) served with a preemptive head of the line (HOL) service discipline. The high priority flow is constrained by arrival curve α.

beginning of the backlog period for high priority traffic. The service for high priority can be delayed by a low priority packet that arrived shortly before s', but as soon as this packet is served, the server is dedicated to high priority as long as there is some high priority traffic to serve. Over the interval $(s, t]$, the output is $C(t - s)$Thus

$$R^*_H(t) - R^*_H(s) \geq C(t - s) - l^H_{\max}$$

where $l^L_{\max}$ is the maximum size of a low priority packet. Now by definition of s: $R^*_H(s) = R_H(s)$ thus

$$R_H^*(t) \geq R_H(s) + C(t-s) - l_{\max}^L$$

Now we have also

$$R_H^*(t) - R_H(s) = R_H^*(t) - R_H^*(s) \geq 0$$

from which we derive

$$R_H^*(t) \geq R_H(s) + [C(t-s) - l_{\max}^L]^+$$

The function $u \to [Cu - l_{\max}^L]^+$ is called the rate-latency function with rate C and latency $\frac{l_{\max}^L}{C}$ [71] (in this book we note it $\beta_{C,\frac{l_{\max}^L}{C}}$, see also Figure 3.1 on page 130). Thus the high priority traffic receives this function as a service curve.

Now let us examine low priority traffic. In order to assure that it does not starve, we assume in such situations that the high priority flow is constrained by an arrival curve α_H. Consider again some arbitrary time t. Call s' the beginning of the server busy period (note that $s' \leq s$). At time s', the backlogs for both flows are empty, namely, $R_H^*(s') = R_H(s')$ and $R_L^*(s') = R_L(s')$. Over the interval $(s', t]$, the output is $C(t-s')$. Thus

$$R_L^*(t) - R_L^*(s') = C(t-s') - [R_H^*(t) - R_H^*(s')]$$

Now

$$R_H^*(t) - R_H^*(s') = R_H^*(t) - R_H(s') \leq R_H(t) - R_H(s') \leq \alpha_H(t-s')$$

and obviously $R_H^*(t) - R_H^*(s') \geq 0$ thus

$$R_L^*(t) - R_L(s') = R_L^*(t) - R_L^*(s') \geq S(t-s')$$

with $S(u) = (Cu - \alpha_H(u))^+$. Thus, if S is wide-sense increasing, the low-priority flow receives a service curve equal to function S. Assume further that $\alpha_H = \gamma_{r,b}$, namely, the high priority flow is constrained by one single leaky bucket or GCRA. In that case, the service curve $S(t)$ offered to the low-priority flow is equal to the rate-latency function $\beta_{R,T}(t)$, with $R = C - r$ and $T = \frac{b}{C-r}$.

We have thus shown the following.

Proposition 1.3.4. *Consider a constant bit rate server, with rate C, serving two flows, H and L, with non-preemptive priority given to flow H. Then the high priority flow is guaranteed a rate-latency service curve with rate C and latency $\frac{l_{\max}^L}{C}$ where $l_{\max}^L$ is the maximum packet size for the low priority flow.*

If in addition the high priority flow is $\gamma_{r,b}$-smooth, with $r < C$, then the low priority flow is guaranteed a rate-latency service curve with rate $C - r$ and latency $\frac{b}{C-r}$.

This example justifies the importance of the rate-latency service curve. We will also see in Chapter 2 (Theorem 2.1.1 on page 87) that all practical implementations of GPS offer a service curve of the rate-latency type.

Strict service curve An important class of network nodes fits in the following framework.

Definition 1.3.2 (Strict Service Curve). *We say that system $\mathcal{S}$ offers a strict service curve β to a flow if, during any backlogged period of duration u, the output of the flow is at least equal to $\beta(u)$.*

A GPS node is an example of node that offers a strict service curve of the form $\beta(t) = rt$. Using the same busy-period analysis as with the GPS example in the previous section, we can easily prove the following.

Proposition 1.3.5. *If a node offers β as a strict service curve to a flow, then it also offers β as a service curve to the flow.*

The strict service curve property offers a convenient way of visualizing the service curve concept: in that case, $\beta(u)$ is the minimum amount of service guaranteed during a busy period. Note however that the concept of service curve, as defined in Definition 1.3.1 is more general. A greedy shaper (Section 1.5.2) is an example of system that offers its shaping curve as a service curve, without satisfying the strict service curve property. In contrast, we will find later in the book some properties that hold only if a strict service curve applies. The framework for a general discussion of strict service curves is given in Chapter 7.

Variable Capacity Node Consider a network node that offers a variable service capacity to a flow. In some cases, it is possible to model the capacity by a cumulative function $M(t)$, where $M(t)$ is the total service capacity available to the flow between times 0 and t. For example, for an ATM system, think of $M(t)$ as the number of time slots between times 0 and t that are available for sending cells of the flow. Let us also assume that the node buffer is large enough so that overflow is not possible. The following proposition is obvious but important in practice

Proposition 1.3.6. *If the variable capacity satisfies a minimum guarantee of the form*

$$M(t) - M(s) \geq \beta(t - s) \tag{1.10}$$

for some fixed function β and for all $0 \leq s \leq t$, then β is a strict service curve,

Thus β is also a service curve for that particular flow. The concept of variable capacity node is also a convenient way to establish service curve properties. For an application to real time systems (rather than communication networks) see [74].

We will show in Chapter 4 that the output of the variable capacity node is given by

$$R^*(t) = \inf_{0 \leq s \leq t} \{M(t) - M(s) + R(s)\}$$

Lastly, coming back to the priority node, we have:

Proposition 1.3.7. *The service curve properties in Proposition 1.3.4 are strict.*

The proof is left to the reader. It relies on the fact that constant rate server is a shaper.

1.4 Network Calculus Basics

In this section we see the main simple network calculus results. They are all bounds for lossless systems with service guarantees.

1.4.1 Three Bounds

The first theorem says that the backlog is bounded by the vertical deviation between the arrival and service curves:

Theorem 1.4.1 (Backlog Bound). *Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve β. The backlog $R(t) - R^*(t)$ for all t satisfies:*

$$R(t) - R^*(t) \le \sup_{s \ge 0}\{\alpha(s) - \beta(s)\}$$

Proof: The proof is a straightforward application of the definitions of service and arrival curves:

$$R(t) - R^*(t) \le R(t) - \inf_{0 \le s \le t}[R(t-s) + \beta(s)]$$

Thus

$$R(t) - R^*(t) \le \sup_{0 \le s \le t}[R(t) - R(t-s) + \beta(s)] \le \sup_{0 \le s \le t}[\alpha(s) + \beta(t-s)]$$

□

We now use the concept of horizontal deviation, defined in Chapter 3, Equation (3.21). The definition is a little complex, but is supported by the following intuition. Call

$$\delta(s) = \inf\{\tau \ge 0 : \alpha(s) \le \beta(s+\tau)\}$$

From Definition 1.1.1, $\delta(s)$ is the virtual delay for a hypothetical system that would have α as input and β as output, assuming that such a system exists (in other words, assuming that ($\alpha \le \beta$). Then, $h(\alpha, \beta)$ is the supremum of all values of $\delta(s)$. The second theorem gives a bound on delay for the general case.

Theorem 1.4.2 (Delay Bound). *Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve of β. The virtual delay $d(t)$ for all t satisfies: $d(t) \le h(\alpha, \beta)$.*

Proof: Consider some fixed $t \ge 0$; for all $\tau < d(t)$, we have, from the definition of virtual delay, $R(t) > R^*(t+\tau)$. Now the service curve property at time $t+\tau$ implies that there is some s_0 such that

$$R(t) > R(t + \tau - s_0) + \beta(s_0)$$

It follows from this latter equation that $t + \tau - s_0 < t$. Thus

$$\alpha(\tau - s_0) \geq [R(t) - R(t + \tau - s_0)] > \beta(s_0)$$

Thus $\tau \leq \delta(\tau - s_0) \leq h(\alpha, \beta)$. This is true for all $\tau < d(t)$ thus $d(t) \leq h(\alpha, \beta)$. □

Theorem 1.4.3 (Output Flow). *Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve of β. The output flow is constrained by the arrival curve $\alpha^* = \alpha \oslash \beta$.*

The theorem uses min-plus deconvolution, introduced in Chapter 3, which we have already used in Theorem 1.2.2.

Proof: With the same notation as above, consider $R^*(t) - R^*(t - s)$, for $0 \leq t - s \leq t$. Consider the definition of the service curve, applied at time $t - s$. Assume for a second that the $\inf$ in the definition of $R \otimes \beta$ is a $\min$, that is to say, there is some $u \geq 0$ such that $0 \leq t - s - u$ and

$$(R \otimes \beta)(t - s) = R(t - s - u) + \beta(u)$$

Thus

$$R^*(t - s) - R(t - s - u) \geq \beta(u)$$

and thus

$$R^*(t) - R^*(t - s) \leq R^*(t) - \beta(u) - R(t - s - u)$$

Now $R^*(t) \leq R(t)$, therefore

$$R^*(t) - R^*(t - s) \leq R(t) - R(t - s - u) - \beta(u) \leq \alpha(s + u) - \beta(u)$$

and the latter term is bounded by $(\alpha \oslash \beta)(s)$ by definition of the $\oslash$ operator.

Now relax the assumption that the the $\inf$ in the definition of $R \otimes \beta$ is a $\min$. In this case, the proof is essentially the same with a minor complication. For all $\epsilon > 0$ there is some $u \geq 0$ such that $0 \leq t - s - u$ and

$$(R \otimes \beta)(t - s) \geq R(t - s - u) + \beta(u) - \epsilon$$

and the proof continues along the same line, leading to:

$$R^*(t) - R^*(t - s) \leq (\alpha \oslash \beta)(s) + \epsilon$$

This is true for all $\epsilon > 0$, which proves the result. □

A simple Example and Interpretation of Leaky Bucket Consider a flow constrained by one leaky bucket, thus with an arrival curve of the form $\alpha = \gamma_{r,b}$, served in a node with the service curve guarantee $\beta_{R,T}$. The alert reader will enjoy applying the three bounds and finding the results shown in Figure 1.10.

Consider in particular the case $T = 0$, thus a flow constrained by one leaky bucket served at a constant rate R. If $R \geq r$ then the buffer required to serve the flow is b, otherwise, it is infinite. This gives us a common interpretation of the leaky bucket parameters r and b: r is the minimum rate required to serve the flow, and b is the buffer required to serve the flow at any constant rate $\geq r$.

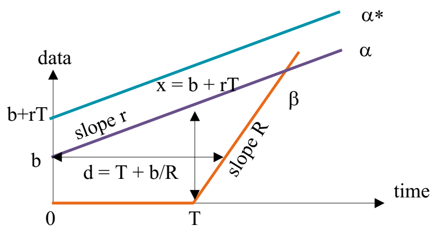

Figure 1.10: Computation of buffer, delay and output bounds for an input flow constrained by one leaky bucket, served in one node offered a rate-latency service curve. If $r \leq R$, then the buffer bound is $x = b + rT$, the delay bound is $d = T + \frac{b}{R}$ and the burstiness of the flow is increased by rT. If $r > R$, the bounds are infinite.

Example: VBR flow with rate-latency service curve Consider a VBR flow, defined by T-SPEC (M, p, r, b). This means that the flow has $\alpha(t) = \min(M + pt, rt + b)$ as an arrival curve (Section 1.2). Assume that the flow is served in one node that guarantees a service curve equal to the rate-latency function $\beta = \beta_{R,T}$. This example is the standard model used in Intserv. Let us apply Theorems 1.4.1 and 1.4.2. Assume that $R \geq r$, that is, the reserved rate is as large as the sustainable rate of the flow.

From the convexity of the region between α and β (Figure 1.4.1), we see that the vertical deviation $v = \sup_{s \geq 0}[\alpha(s) - \beta(s)]$ is reached for at an angular point of either α or β. Thus

$$v = \max[\alpha(T), \alpha(\theta) - \beta(\theta)]$$

with $\theta = \frac{b-M}{p-r}$. Similarly, the horizontal distance is reached an angular point. In the figure, it is either the distance marked as AA' or BB'. Thus, the bound on delay d is given by

$$d = \max\left(\frac{\alpha(\theta)}{R} + T - \theta, \frac{M}{R} + T\right)$$

After some max-plus algebra, we can re-arrange these results as follows.

Proposition 1.4.1 (Intserv model, buffer and delay bounds). *Consider a VBR flow, with TSPEC (M, p, r, b), served in a node that guarantees to the flow a service curve equal to the rate-latency function $\beta = \beta_{R,T}$. The buffer required for the flow is bounded by*

$$v = b + rT + \left(\frac{b - M}{p - r} - T\right)^+ [(p - R)^+ - p + r]$$

The maximum delay for the flow is bounded by

$$d = \frac{M + \frac{b-M}{p-r}(p - R)^+}{R} + T$$

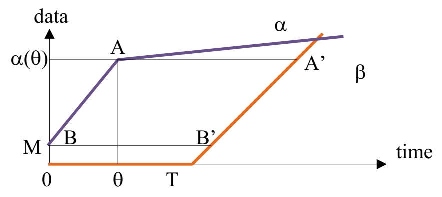

Figure 1.11: Computation of buffer and delay bound for one VBR flow served in one Intserv node.

We can also apply Theorem 1.4.3 and find an arrival curve α^* for the output flow. We have $\alpha^* = \alpha \oslash (\lambda_R \otimes \delta_T) = (\alpha \oslash \lambda_R) \oslash \delta_T$ from the properties of $\oslash$ (Chapter 3). Note that

$$(f \oslash \delta_T)(t) = f(t + T)$$

for all f (shift to the left).

The computation of $\alpha \oslash \lambda_R$ is explained in Theorem 3.1.14 on Page 152: it consists in inverting time, and smoothing. Here, we give however a direct derivation, which is possible since α is concave. Indeed, for a concave α, define t_0 as

$$t_0 = \inf\{t \geq 0 : \alpha'(t) \leq R\}$$

where α' is the left-derivative, and assume that $t_0 < +\infty$. A concave function always has a left-derivative, except maybe at the ends of the interval where it is defined. Then by studying the variations of the function $u \to \alpha(t + u) - Ru$ we find that $(\alpha \oslash \lambda_R)(s) = \alpha(s)$ if $s \geq t_0$, and $(\alpha \oslash \lambda_R)(s) = \alpha(t_0) + (s - t_0)R$ if $s < t_0$.

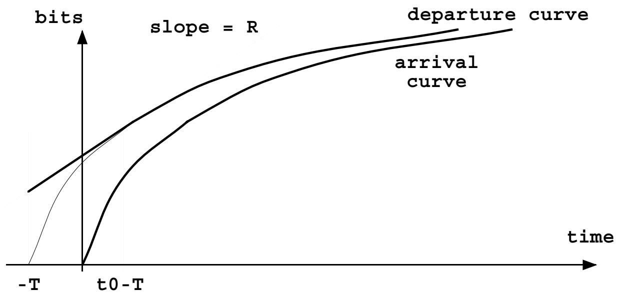

Figure 1.12: Derivation of arrival curve for the output of a flow served in a node with rate-latency service curve $\beta_{R,T}$.

Putting the pieces all together we see that the output function α^* is obtained from α by

- replacing α on $[0, t_0]$ by the linear function with slope R that has the same value as α for $t = t_0$, keeping the same values as α on $[t_0, +\infty[$,
- and shifting by T to the left.

Figure 1.12 illustrates the operation. Note that the two operations can be performed in any order since $\otimes$ is commutative. Check that the operation is equivalent to the construction in Theorem 3.1.14 on Page 152.

If we apply this to a VBR connection, we obtain the following result.

Proposition 1.4.2 (Intserv model, output bound). *With the same assumption as in Proposition 1.4.1, the output flow has an arrival curve α^* given by:*

$$\begin{cases} \text{if } \frac{b-M}{p-r} \leq T \text{ then } \alpha^*(t) = b + r(T+t) \\ \text{else } \alpha^*(t) = \min\left\{ (t+T)(p \wedge R) + M + \frac{b-M}{p-r}(p-R)^+, b + r(T+t) \right\} \end{cases}$$

An ATM Example Consider the example illustrated in Figure 1.13. The aggregate flow has as an arrival curve equal to the stair function $10 v_{25,4}$. The figure illustrates that the required buffer is 18 ATM cells and the maximum delay is 10 time slots. We

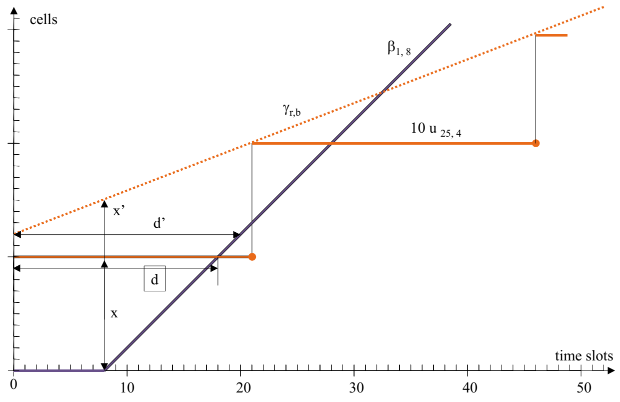

Figure 1.13: Computation of bounds for buffer x and delay d for an ATM example. An ATM node serves 10 ATM connections, each constrained with GCRA$(25, 4)$ (counted in time slots). The node offers to the aggregate flow a service curve $\beta_{R,T}$ with rate $R = 1$ cell per time slot and latency $T = 8$ time slots. The figure shows that approximating the stair function $10 v_{25,4}$ by an affine function $\gamma_{r,b}$ results into an overestimation of the bounds.

know from Corollary 1.2.1 that a GCRA constraint is equivalent to a leaky bucket. Thus, each of the 10 connections is constrained by an affine arrival curve $\gamma_{r,b}$ with $r = \frac{1}{25} = 0.04$ and $b = 1 + \frac{4}{25} = 1.16$. However, if we take as an arrival curve for the aggregate flow the resulting affine function $10\gamma_{r,b}$, then we obtain a buffer bound of 11.6 and a delay bound of 19.6. The affine function overestimates the buffer and delay bounds. Remember that the equivalence between stair function and affine function is only for a flow where the packet size is equal to the value of the step, which is clearly not the case for an aggregate of several ATM connections.

A direct application of Theorem 1.4.3 shows that an arrival curve for the output flow is given by $\alpha_0^*(t) = \alpha(t+T) = v_{25,12}(t)$.

In Chapter 2, we give a slight improvement to the bounds if we know that the service curve is a strict service curve.

1.4.2 Are the Bounds Tight ?

We now examine how good the three bounds are. For the backlog and delay bounds, the answer is simple:

Theorem 1.4.4. *Consider the backlog and delay bounds in Theorems 1.4.1 and 1.4.2. Assume that*

- *α is a "good" function (that is, namely, is wide-sense increasing, sub-additive and $\alpha(0) = 0$)*
- *β is wide-sense increasing and $\beta(0) = 0$*

Then the bounds are tight. More precisely, there is one causal system with input flow $R(t)$ and output flow $R^*(t)$, such that the input is constrained by α, offering to the flow a service curve β, and which achieves both bounds.

A causal system means that $R(t) \leq R^*(t)$. The theorem means that the backlog bound in Theorem 1.4.1 is equal to $\sup_{t\geq 0}[R(t) - R^*(t)]$, and the delay bound in Theorem 1.4.1 is equal to $\sup_{t\geq 0} d(t)$. In the above, $d(t)$ is the virtual delay defined in Definition 1.1.1.

Proof: We build one such system R, R^* by defining $R = \alpha, R^* = \min(\alpha, \beta)$. The system is causal because $R^* \leq \alpha = R$. Now consider some arbitrary time t. If $\alpha(t) < \beta(t)$ then

$$R^*(t) = R(t) = R(t) + \beta(0)$$

Otherwise,

$$R^*(t) = \beta(t) = R(0) + \beta(t)$$

In all cases, for all t there is some $s \leq t$ such that $R^*(t) \geq R(t-s) + \beta(s)$, which shows the service curve property. □

Of course, the bounds are as tight as the arrival and service curves are. We have seen that a source such that $R(t) = \alpha(t)$ is called *greedy*. Thus, the backlog and delay bounds are worst-case bounds that are achieved for greedy sources.

In practice, the output bound is also a worst-case bound, even though the detailed result is somehow less elegant.

Theorem 1.4.5. *Assume that*

1. *α is a "good" function (that is, is wide-sense increasing, sub-additive and $\alpha(0) = 0$)*
2. *α is left-continuous*
3. *β is wide-sense increasing and $\beta(0) = 0$*
4. *$\alpha \overline{\oslash} \alpha$ is not bounded from above.*

Then the output bound in Theorem 1.4.3 is tight. More precisely, there is one causal system with input flow $R(t)$ and output flow $R^*(t)$, such that the input is constrained by α, offering to the flow a service curve β, and α^* (given by Theorem 1.4.3) is the minimum *arrival curve for R^*.*

We know in particular from Section 1.2 that the first three conditions are not restrictive. Let us first discuss the meaning of the last condition. By definition of max-plus deconvolution:

$$(\alpha \overline{\oslash} \alpha)(t) = \inf_{s \geq 0} \{\alpha(t+s) - \alpha(s)\}$$

One interpretation of $\alpha \overline{\oslash} \alpha$ is as follows. Consider a greedy source, with $R(t) = \alpha(t)$; then $(\alpha \overline{\oslash} \alpha)(t)$ is the minimum number of bits arriving over an interval of duration t. Given that the function is wide-sense increasing, the last condition means that $\lim_{t \to +\infty} (\alpha \overline{\oslash} \alpha)(t) = +\infty$. For example, for a VBR source with T-SPEC (p, M, r, b) (Figure 1.5), we have $(\alpha \overline{\oslash} \alpha)(t) = rt$ and the condition is satisfied. The alert reader will easily be convinced that the condition is also true if the arrival curve is a stair function.

The proof of Theorem 1.4.5 is a little technical and is left at the end of this chapter.

We might wonder whether the output bound α^* is a "good" function. The answer is no, since $\alpha^*(0)$ is the backlog bound and is positive in reasonable cases. However, α^* is sub-additive (the proof is easy and left to the reader) thus the modified function $\delta_0 \wedge \alpha^*$ defined as $\alpha^*(t)$ for $t > 0$ and 0 otherwise is a "good" function. If α is left-continuous, $\delta_0 \wedge \alpha^*$ is even a "very good" function since we know from the proof of Theorem 1.4.5 that it is left-continuous.

1.4.3 Concatenation

So far we have considered elementary network parts. We now come to the main result used in the concatenation of network elements.

Theorem 1.4.6 (Concatenation of Nodes). *Assume a flow traverses systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in sequence. Assume that $\mathcal{S}_i$ offers a service curve of β_i, $i = 1, 2$ to the flow. Then the concatenation of the two systems offers a service curve of $\beta_1 \otimes \beta_2$ to the flow.*

Proof: Call R_1 the output of node 1, which is also the input to node 2. The service curve property at node 1 gives

$$R_1 \geq R \otimes \beta_1$$

and at node 2

$$R^* \geq R_1 \otimes \beta_2 \geq (R \otimes \beta_1) \otimes \beta_2 = R \otimes (\beta_1 \otimes \beta_2)$$

$\square$

Examples: Consider two nodes offering each a rate-latency service curve β_{R_i,T_i}, $i = 1, 2$, as is commonly assumed with Intserv. A simple computation gives

$$\beta_{R_1,T_1} \otimes \beta_{R_1,T_1} = \beta_{\min(R_1,R_2),T_1+T_2}$$

Thus concatenating Intserv nodes amounts to adding the latency components and taking the minimum of the rates.

We are now also able to give another interpretation of the rate-latency service curve model. We know that $\beta_{R,T} = (\delta_T \otimes \lambda_R)(t)$; thus we can view a node offering a rate-latency service curve as the concatenation of a guaranteed delay node, with delay T and a constant bit rate or GPS node with rate R.

Pay Bursts Only Once The concatenation theorem allows us to understand a phenomenon known as "Pay Bursts Only Once". Consider the concatenation of two nodes offering each a rate-latency service curve β_{R_i,T_i}, $i = 1, 2$, as is commonly assumed with Intserv. Assume the fresh input is constrained by $\gamma_{r,b}$. Assume that $r < R_1$ and $r < R_2$. We are interested in the delay bound, which we know is a worst case. Let us compare the results obtained as follows.

1. by applying the network service curve;
2. by iterative application of the individual bounds on every node

The delay bound D_0 can be computed by applying Theorem 1.4.2:

$$D_0 = \frac{b}{R} + T_0$$

with $R = \min_i(R_i)$ and $T_0 = \sum_i T_i$ as seen above.

Now apply the second method. A bound on the delay at node 1 is (Theorem 1.4.2):

$$D_1 = \frac{b}{R_1} + T_1$$

The output of the first node is constrained by α^*, given by :

$$\alpha^*(t) = b + r \times (t + T_1)$$

A bound on the delay at the second buffer is:

$$D_2 = \frac{b + rT_1}{R_2} + T_2$$

And thus

$$D_1 + D_2 = \frac{b}{R_1} + \frac{b + rT_1}{R_2} + T_0$$

It is easy to see that $D_0 < D_1 + D_2$. In other words, the bounds obtained by considering the global service curve are better than the bounds obtained by considering every buffer in isolation.

Let us continue the comparison more closely. The delay through one node has the form $\frac{b}{R_1} + T_1$ (for the first node). The element $\frac{b}{R_1}$ is interpreted as the part of the delay due to the burstiness of the input flow, whereas T_1 is due to the delay component of the node. We see that $D_1 + D_2$ contains twice an element of the form $\frac{b}{R_i}$, whereas D_0 contains it only once. We sometimes say that "we pay bursts only once". Another difference between D_0 and $D_1 + D_2$ is the element $\frac{rT_1}{R_2}$: it is due to the increase of burstiness imposed by node 1. We see that this increase of burstiness does not result into an increase of the overall delay.

A corollary of Theorem 1.4.6 is also that the end-to-end delay bound does not depend on the order in which nodes are concatenated.

1.4.4 Improvement of Backlog Bounds

We give two cases where we can slightly improve the backlog bounds.

Theorem 1.4.7. *Assume that a lossless node offers a* strict *service curve β to a flow with arrival curve α. Assume that $\alpha(u_0) \le \beta(u_0)$ for some $u_0 > 0$. Then the duration of the busy period is $\le u_0$. Furthermore, for any time t, the backlog $R(t) - R^*(t)$ satisfies*

$$R(t) - R^*(t) \le \sup_{u:0 \le u < u_0} [R(t) - R(t-u) - \beta(u)] \le \sup_{u:0 \le u < u_0} [\alpha(u) - \beta(u)]$$

The theorem says that, for the computation of a buffer bound, it is sufficient to consider time intervals less than u_0. The idea is that the busy period duration is less than u_0.

Proof: Consider a given time t at which the buffer is not empty, and call s the last time instant before t at which the buffer was empty. Then, from the strict service curve property, we have

$$R^*(t) \ge R^*(s) + \beta(t-s) = x(s) + \beta(t-s)$$

Thus the buffer size $b(t) = R(t) - R^*(t)$ at time t satisfies

$$b(t) \le R(t) - R(s) - \beta(t-s) \le \alpha(t-s) - \beta(t-s)$$

Now if $t - s \geq u_0$, then there is a time $t' = s + u_0$, with $s + 1 \leq t' \leq t$ such that $b(t') = 0$. This contradicts the definition of s. Thus we can assume that $t - s < u_0$. □

Theorem 1.4.8. *Assume that a lossless node offers a service curve β to a flow with sub-additive arrival curve α. Assume that β is* super-additive, *and that $\alpha(u_0) \leq \beta(u_0)$ for some $u_0 > 0$. Then for any time t, the backlog $R(t) - R^*(t)$ satisfies*

$$R(t) - R^*(t) \leq \sup_{u:0 \leq u < u_0} [R(t) - R(t-u) - \beta(u)] \leq \sup_{u:0 \leq u < u_0} [\alpha(u) - \beta(u)]$$

Note that the condition that α is super-additive is not a restriction. In contrast, the condition that β is super-additive is a restriction. It applies in particular to rate-latency service curves. The theorem does not say anything about the duration of the busy period, which is consistent with the fact we do not assume here that the service curve is strict.

Proof: For an arbitrary time t the backlog at time t satisfies

$$b(t) \leq \sup_{u \geq 0} [R(t) - R(t-u) - \beta(u)]$$

For $s \leq t$ define $k = \lceil \frac{t-s}{u_0} \rceil$ and $s' = ku_0 + s$. We have $s \leq s' \leq t$ and

$$t - u_0 < s' \tag{1.11}$$

Now from the super-additivity of β:

$$R(t) - R(s) \leq [R(t) - R(s') - \beta(t - s')] + [R(s') - R(s) - \beta(s' - s)]$$

Note that for the second part we have

$$R(s') - R(s) - \beta(s' - s) \leq k\,[\alpha(u_0) - \beta(u_0)] \leq 0$$

thus

$$R(t) - R(s) \leq [R(t) - R(s') - \beta(t - s')]$$

which shows the theorem. □

1.5 Greedy Shapers

1.5.1 Definitions

We have seen with the definition of the leaky bucket and of the GCRA two examples of devices that enforce a general arrival curve. We call *policer*with curve σ a device that counts the bits arriving on an input flow and decides which bits conform with an arrival curve of σ. We call *shaper*, with shaping curveσ, a bit processing device that forces its output to have σ as an arrival curve. We call *greedy shaper* a shaper that delays the input bits in a buffer, whenever sending a bit would violate the constraint σ, but outputs them as soon as possible.

With ATM and sometimes with Intserv, traffic sent over one connection, or flow, is policed at the network boundary. Policing is performed in order to guarantee that users do not send more than specified by the contract of the connection. Traffic in excess is either discarded, or marked with a low priority for loss in the case of ATM, or passed as best effort traffic in the case of Intserv. In the latter case, with IPv4, there is no marking mechanism, so it is necessary for each router along the path of the flow to perform the policing function again.

Policing devices inside the network are normally buffered, they are thus shapers. Shaping is also often needed because the output of a buffer normally does not conform any more with the traffic contract specified at the input.

1.5.2 Input-Output Characterization of Greedy Shapers

The main result with greedy shapers is the following.

Theorem 1.5.1 (Input-Output Characterization of Greedy Shapers). *Consider a greedy shaper with shaping curve σ. Assume that the shaper buffer is empty at time 0, and that it is is large enough so that there is no data loss. For an input flow R, the output R^* is given by*

$$R^* = R \otimes \bar{\sigma} \tag{1.12}$$

where $\bar{\sigma}$ is the sub-additive closure of σ.

Proof: Remember first that if σ is sub-additive and $\sigma(0) = 0$, then $\bar{\sigma} = \sigma$. In general, we know that we can replace σ by $\bar{\sigma}$ without changing the definition of the shaper. We thus assume without loss of generality that $\bar{\sigma} = \sigma$.

The proof of the theorem is an application of min-plus algebra. First, let us consider a virtual system that would take R as input and have an output S satisfying the constraints:

$$\begin{cases} S \leq R \\ S \leq S \otimes \sigma \end{cases} \tag{1.13}$$

Such a system would behave as a buffer (the first equation says that the output is derived from the input) and its output would satisfy the arrival curve constraint σ. However, such a system is not necessarily a greedy shaper; we could have for example a lazy shaper with $S(t) = 0$ for all $t \geq 0$! For this system to be a greedy shaper, it has to output the bits as soon as possible. Now there is a general result about systems satisfying conditions 1.13.

Lemma 1.5.1 (A min-plus linear system). *Assume that σ is a "good" function (that is, is sub-additive and $\sigma(0) = 0$). Among all functions $S(t)$ satisfying conditions 1.13 for some fixed function R, there is one that is an upper bound for all. It is equal to $R \otimes \sigma$*

Proof of the lemma: The lemma is a special case of a general result in Chapter 4. However, it is also possible to give a very simple proof, as follows.

Define $S^* = R \otimes \sigma$. Since σ is a "good" function, it follows immediately that S^* is a solution to System (1.13). Now, let S' be some other solution. We have $S' \leq R$ and thus

$$S' \leq S_0 \otimes \sigma = S^*$$

Therefore S^* is the maximal solution. □

Note that the lemma proves the existence of a maximal solution to System (1.13). Note also that, in the lemma, function R need not be wide-sense increasing.

Now we can use the lemma by showing that $R^* = S^*$. Function R is wide-sense increasing, thus so is S^*. Obviously, R^* is a solution of System (1.13), thus $R^*(t) \leq S^*(t)$ for all t. Now if there would be some t such that $R^*(t) \neq S^*(t)$, then this would contradict the condition that the greedy shaper attempts to send the bits out as early as possible. □

The following corollary derives immediately.

Corollary 1.5.1 (Service Curve offered by a Greedy Shaper). *Consider a greedy shaper with shaping curve σ. Assume that σ is sub-additive and $\sigma(0) = 0$. This system offers to the flow a service curve equal to σ.*

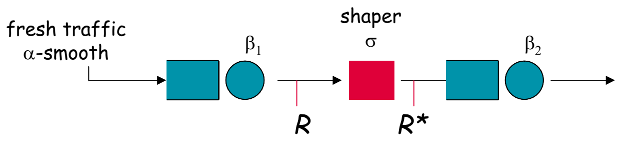

Figure 1.14: Reshaping example.

Example: Buffer Sizing at a Re-shaper Re-shaping is often introduced because the output of a buffer normally does not conform any more with the traffic contract specified at the input. For example, consider a flow with the arrival curve $\sigma(t) = \min(pt + M, rt + b)$ that traverses a sequence of nodes, which offer a service curve $\beta_1 = \beta_{R,T}$. A greedy shaper, with shaping curve σ, is placed after the sequence of nodes (Figure 1.14). The input to the shaper (R in the figure) has an arrival curve α^*, given by Proposition 1.4.2. Corollary 1.5.1 gives a service curve property for the greedy shaper, thus the buffer B required at the greedy shaper is the vertical distance $v(\alpha^*, \sigma)$. After some algebra, we obtain:

$$B = \begin{cases} \text{if } \frac{b-M}{p-r} < T & \text{then } b + Tr \\ \text{if } \frac{b-M}{p-r} \geq T \text{ and } p > R & \text{then } M + \frac{(b-M)(p-R)}{p-r} + TR \\ \text{else} & M + Tp \end{cases} \tag{1.14}$$

Corollary 1.5.2 (Buffer Occupancy at a Greedy Shaper). *Consider a greedy shaper with shaping curve σ. Assume that σ is sub-additive and $\sigma(0) = 0$. Call $R(t)$ the input function. The buffer occupancy $x(t)$ at time t is given by*

$$x(t) = \sup_{0 \leq s \leq t} \{R(t) - R(s) - \sigma(t-s)\}$$

Proof: The backlog is defined by $x(t) = R(t) - R^*(t)$, where R^* is the output. We apply Theorem 1.5.1 and get:

$$x(t) = R(t) - \inf_{0 \leq s \leq t} \{R(s) + \sigma(t-s)\} = R(t) + \sup_{0 \leq s \leq t} \{-R(s) - \sigma(t-s)\}$$

□

Note that Lemma 1.2.2 is a special case of this corollary.

In min-plus algebraic terms, we say that a system is linear and time invariant if its input-output characterization has the form $R^* = R \otimes \beta$ (where β is not necessarily sub-additive). We can thus say from the theorem that greedy shapers are min-plus linear and time invariant systems. There are min-plus linear and time invariant system that are not greedy shapers. For example, a node imposing a *constant* delay T is characterized by the input-output relationship

$$R^* = R \otimes \delta_T$$

Compare to the guaranteed delay node (namely, a node imposing a variable delay bounded by T), for which the input-output relationship is a service curve property :

$$R^* \geq R \otimes \delta_T$$

The rest of this Section illustrates similarly that the input-output characterization of greedy shapers $R^* = R \otimes \sigma$ is much stronger than the service curve property described in Corollary 1.5.1.

1.5.3 Properties of Greedy Shapers

Consider again Figure 1.14. We have seen in the previous section how we can compute the buffer size required at the greedy shaper. Now if greedy shapers are introduced along a path, then some bits may be delayed at the shaper, thus the end-to-end delay might increase. However, this is not true, as the following results state that, from a global viewpoint, “greedy shapers come for free”.

Theorem 1.5.2 (Re-Shaping does not increase delay or buffer requirements). *Assume a flow, constrained by arrival curve α, is input to networks $\mathcal{S}_1$ and $\mathcal{S}_2$ in sequence. Assume a greedy shaper, with curve $\sigma \geq \alpha$ is inserted between $\mathcal{S}_1$ and $\mathcal{S}_2$. Then the backlog and delay bounds given by Theorem 1.4.2 for the system without shaper are also valid for the system with shaper.*

The condition $\sigma \geq \alpha$ means that re-shaping maybe only partial.

Proof: Call β_i the service curve of $\mathcal{S}_i$. The backlog bound in Theorem 1.4.1 is given by

$$v(\alpha, \beta_1 \otimes \sigma \otimes \beta_2) = v(\alpha, \sigma \otimes \beta_1 \otimes \beta_2) \tag{1.15}$$

Now the last expression is the backlog bound obtained if we put the shaper immediately at the entrance of the network. Clearly, this introduces no backlog, which shows that the overall backlog is not influenced by the shaper. The same reasoning applies to the delay bound. □

If you read carefully, you should not agree with the last paragraph. Indeed, there is a subtlety. The bounds in Section 1.4 are tight, but since we are using several bounds together, there is no guarantee that the resulting bound is tight. All we can say at this point is that the bound computed for the system with shaper is the same if we put the shaper in front; we still need to show that the bound for such a system is the same bound as if there would be no shaper. This can be proven in a number of ways. We give here a computational one. The proof relies on Lemma 1.5.2, given below. □

Lemma 1.5.2. *Let α and σ be "good" functions. Assume $\alpha \le \sigma$. Then for any function β, $v(\alpha, \sigma \otimes \beta) = v(\alpha, \beta)$ and $h(\alpha, \sigma \otimes \beta) = h(\alpha, \beta)$.*

Proof: We use the reduction to min-plus deconvolution explained in Section 3.1.11. We have:

$$v(\alpha, \sigma \otimes \beta) = [\alpha \oslash (\sigma \otimes \beta)](0)$$

Now from Theorem 3.1.12 on Page 148: $\alpha \oslash (\sigma \otimes \beta) = (\alpha \oslash \sigma) \oslash \beta$. Also, since $\sigma \ge \alpha$, we have $\alpha \oslash \sigma \le \alpha \oslash \alpha$. Now $\alpha \oslash \alpha = \alpha$ because α is a "good" function, thus

$$\alpha \oslash (\sigma \otimes \beta) = \alpha \oslash \beta \tag{1.16}$$

and finally $v(\alpha, \sigma \otimes \beta) = v(\alpha, \beta)$.

Similarly $h(\alpha, \beta) = \inf\{d \text{ such that } (\alpha \oslash \beta)(-d) \le 0\}$ which, combined with Equation (1.16) proves that $h(\alpha, \sigma \otimes \beta) = h(\alpha, \beta)$. □

Consider again Figure 1.14. Assume that the first network element and the greedy shaper are placed in the same node. Theorem 1.5.2 says that the *total* buffer required for this combined node is the same as if there would be no greedy shaper at the output. Thus, if you can dynamically allocate buffer space from a common pool to the first network element and the greedy shaper, then the greedy shaper costs no memory. However, the greedy shaper does need some buffer space, as given in Equation (1.14). Similarly, the theorem says that there is no penalty for the worst-case delay.

In contrast, placing a greedy shaper has an obvious benefit. The burstiness of the flow admitted in the next network element is reduced, which also reduces the buffer required in that element. To be more concrete, consider the example "Pay Bursts Only Once" in Section 1.4.3. Assume that a re-shaper is introduced at the output of the first node. Then the input to the second node has the same arrival curve as the fresh input, namely, $\gamma_{r,b}$ instead of $\gamma_{r,b+rT_1}$. The buffer required for the flow at node 2 is then $b + rT_2$ instead of $b + r(T_1 + T_2)$.

The following result is another "physical" property of greedy shapers. It says that shaping cannot be undone by shaping.

Theorem 1.5.3 (Shaping Conserves Arrival Constraints). *Assume a flow with arrival curve α is input to a greedy shaper with shaping curve σ. Assume σ is a "good" function. Then the output flow is still constrained by the original arrival curve α.*

Proof:

$$R^* = R \otimes \sigma \leq (R \otimes \alpha) \otimes \sigma$$

since the condition $R \leq R \otimes \alpha$ expresses that α is an arrival curve. Thus

$$R^* \leq R \otimes \sigma \otimes \alpha = R^* \otimes \alpha$$

□

The output of the greedy shaper has thus $\min(\alpha, \sigma)$ as an arrival curve. If α is also a "good" function, we know (Lemma 1.2.5) that the sub-additive closure of $\min(\alpha, \sigma)$ is $\alpha \otimes \sigma$.

Example (ATM Multiplexer): Consider an ATM switch that receives 3 ATM connections, each constrained by GCRA(10, 0) (periodic connections). The switch serves the connection in any work conserving manner and outputs them on a link with rate 1 cell per time slot. What is a good arrival curve for the aggregate output ?

The aggregate input has an arrival curve $\alpha = 3v_{10,0}$. Now the server is a greedy shaper with shaping curve $\sigma = v_{1,0}$, thus it keeps arrival constraints. Thus the output is constrained by $3v_{10,0} \otimes v_{1,0}$, which is a "good" function. We have already met this example in Figure 1.6.

1.6 Maximum Service Curve, Variable and Fixed Delay

1.6.1 Maximum Service Curves

If we modify the sense of the inequation in the definition of service curve in Section 1.3, then we obtain a new concept, called *maximum service curve*, which is useful to (1) account for constant delays and (2) in some cases to establish a relationship between delay and backlog.

Definition 1.6.1 (Maximum Service Curve). *Consider a system $\mathcal{S}$ and a flow through $\mathcal{S}$ with input and output function R and R^*. We say that $\mathcal{S}$ offers to the flow a* maximum service curve *γ if and only if $\gamma \in \mathcal{F}$ and $R^* \leq R \otimes \gamma$*

Note that the definition is equivalent to saying that γ is wide-sense increasing and that

$$R^*(t) \leq R(s) + \gamma(t - s)$$

for all t and all $s \leq t$, or equivalently

$$R^*(t) - R^*(s) \leq B(s) + \gamma(t - s)$$

where $B(s)$ is the backlog at time s. A greedy shaper with shaping curve σ offers σ both as a service curve and a maximum service curve.

In general, the concept of maximum service curve is not as powerful as the concept of service curve. However, as we see below, it can be useful to account for maximum rates and for constant propagation delays. We also see in Chapter 6 that it allows us to find good bounds for aggregate multiplexing.

The following propositions give two special cases of interest. Their proof is easy and left to the reader.

Proposition 1.6.1 (Minimum Delay). *A lossless node offers a maximum service curve equal to δ_T if and only if it imposes a minimum virtual delay equal to T.*

Proposition 1.6.2 (Arrival Constraint on Output). *Assume the output of a lossless node is constrained by some arrival curve σ. Then the node offers σ as a maximum service curve.*

Like minimum service curves, maximum service curves can be concatenated:

Theorem 1.6.1 (Concatenation of Nodes). *Assume a flow traverses systems $\mathcal{S}_1$ and $\mathcal{S}_2$ in sequence. Assume that $\mathcal{S}_i$ offers a maximum service curve of γ_i, $i = 1, 2$ to the flow. Then the concatenation of the two systems offers a service curve of $\gamma_1 \otimes \gamma_2$ to the flow.*

Proof: The proof mimics the proof of Theorem 1.4.6 □

Application: Consider a node with a maximum output rate equal to c and with internal propagation delay equal to T. It follows from Theorem 1.6.1 and the two previous propositions that this node offers to any flow a maximum service curve equal to the rate-latency function $\beta_{c,T}(t) = [c(t - T)]^+$.

Maximum service curves do not allow us to derive as strong results as (ordinary) service curves. However, they can be used to reduce the output bound and, in some cases, to obtain a minimum delay bound. Indeed, we have the following two results.

Theorem 1.6.2 (Output Flow, generalization of Theorem 1.4.3). *Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve β and a maximum service curve γ. The output flow is constrained by the arrival curve $\alpha^* = (\alpha \otimes \gamma) \oslash \beta$.*

Proof: Instead of a computational proof as with Theorem 1.4.3, it is simpler at this stage to use min-plus algebra. Call R and R^* the input and output functions, and consider $R^* \oslash R^*$, the minimum arrival curve for R^*. We have $R^* \leq R \otimes \gamma$ and $R^* \geq R \otimes \beta$, thus

$$R^* \oslash R^* \leq (R \otimes \gamma) \oslash (R \otimes \beta)$$

From Rule 12 in Chapter 3, Theorem 3.1.12, applied to $f = R \otimes \gamma$, $g = R$ and $h = \beta$, we derive

$$R^* \oslash R^* \leq \{(R \otimes \gamma) \oslash R\} \oslash \beta$$

Now from the commutativity of $\otimes$ and from Rule 13 in Theorem 3.1.12:

$$\{(R \otimes \gamma) \oslash R\} = \{(\gamma \otimes R) \oslash R\} \leq \{\gamma \otimes (R \oslash R)\}$$

Thus

$$R^* \oslash R^* \leq \{\gamma \otimes (R \oslash R)\} \oslash \beta \leq (\gamma \otimes \alpha) \oslash \beta$$

□

Theorem 1.6.3 (Minimum Delay Bound). *Assume a flow, constrained by arrival curve α, traverses a system that offers a maximum service curve of γ. Assume that $\gamma(D) = 0$. The virtual delay $d(t)$ satisfies $d(t) \geq D$ for all t.*

Proof: We have $R^*(t) \leq R(t - D) + \gamma(D)$ thus $R^*(t) \leq R(t - D)$ □

Note that the output bound is improved by the knowledge of the maximum service curve since in general we expect $\alpha \otimes \gamma$ to be less than α. In contrast, the minimum delay bound gives some new information only in the cases where there is a latency part in the maximum service curve, which is the case for the first example (Minimum Delay), but not in general for the second example (Arrival Constraint on Output).

Numerical Example: Consider again the example illustrated in Figure 1.13. Let us first apply Theorem 1.4.3 and compute an arrival curve α_0^* for the output. The details are as follows. We have

$$\alpha_0^* = 10v_{25,4} \oslash \beta_{1,8} = 10v_{25,4} \oslash (\lambda_1 \otimes \delta_8)$$

Now from Rule 15 in Chapter 3, we have

$$\alpha_0^* = (10v_{25,4} \oslash \delta_8) \oslash \lambda_1$$

Now $(10v_{25,4} \oslash \delta_8)(t) = 10v_{25,4}(t + 8) = 10v_{25,12}(t)$, and a straightforward application of the definition of $\oslash$ shows that finally $\alpha_0^* = v_{25,12}$.

Assume now that we have more information about the node, and that we can model is as node $\mathcal{S}_1$ defined as the concatenation of two schedulers and a fixed delay element (Figure 1.15). Each scheduler offers to the aggregate flow a service curve β_{R_0,T_0} with rate $R_0 = 1$ cell per time slot and latency $T_0 = 2$ time slots. The delay element is a link with maximum rate equal to 1 cell per time slot, and a fixed propagation and transmission delay equal to 4 time slots. The delay element is thus the combination of a greedy shaper with shaping curve $\lambda_1(t) = t$ and a fixed delay element δ_4. We can verify that the concatenation of the three elements in node 1 offers a service curve equal to $\beta_{1,2} \otimes \lambda_1 \otimes \delta_4 \otimes \beta_{1,2} = \beta_{1,8}$. Now, from the delay element allows us to say that, in addition, the node also offers to the aggregate flow a *maximum service curve* equal to $\beta_{1,4}$. We can apply Theorem 1.6.2 and derive from that the output is constrained by the arrival curve α_1^* given by

$$\alpha_1^* = (\alpha \otimes \beta_{1,4}) \oslash \beta_{1,8}$$

The computation is similar to that of α_0^* and involves the computation of $10v_{25,4} \otimes \lambda_1$, which is similar to the example illustrated in Figure 1.6. Finally, we have:

$$\alpha_1^*(t) = (10v_{25,4} \otimes \lambda_1)(t + 4)$$

Figure 1.15 shows that α_1^* is a better bound than the arrival curve α_0^* that we would obtain if we did not know the maximum service curve property.

Assume next that we change the order of the delay element in node $\mathcal{S}1$ and place it as the last element of the node. Call $\mathcal{S}_2$ the resulting node. Then the conclusion of the previous paragraph remains, since the bounds are insensitive to the order, due to the commutativity of min-plus convolution. Thus the output of system $\mathcal{S}_2$ also has α_1^* as an arrival curve. However, in that case, we can also model the delay element as the combination of a shaper, with shaping curve λ_1 (corresponding to a fixed rate of 1 cell per time slot), followed by a fixed delay element, with constant delay equal to 4 time slots. The input to the shaper has an arrival curve equal to $\alpha \oslash \beta_{1,4}$, where $\alpha = 10v_{25,4}$ is the fresh arrival curve. Thus, from the properties of shapers, the output of the shaper is constrained by

$$\alpha_2^* = (\alpha \oslash \beta_{1,4}) \otimes \lambda_1 = 10v_{25,8} \otimes \lambda_1$$

Since the fixed delay component does not alter the flow, the output of system $\mathcal{S}_2$ has α_2^* as an arrival curve. Figure 1.15 shows that α_2^* is a better bound than α_1^*.

This fact is true in general: whenever a network element can be modeled as a shaper, then this model provides stronger bounds than the maximum service.

1.6.2 Delay from Backlog

In general it is not possible to bound delay from backlog with the framework of service curves, except in one particular but important case.

Theorem 1.6.4. *Assume a lossless node offers to a flow a minimum service curve β and a maximum service curve γ, such that $\beta(t) = \gamma(t - v)$. Let f be the max-plus deconvolution $\gamma \overline{\oslash} \gamma$, that is,*

$$f(t) = \inf_{s \geq 0} [\gamma(s + t) - \gamma(s)]$$

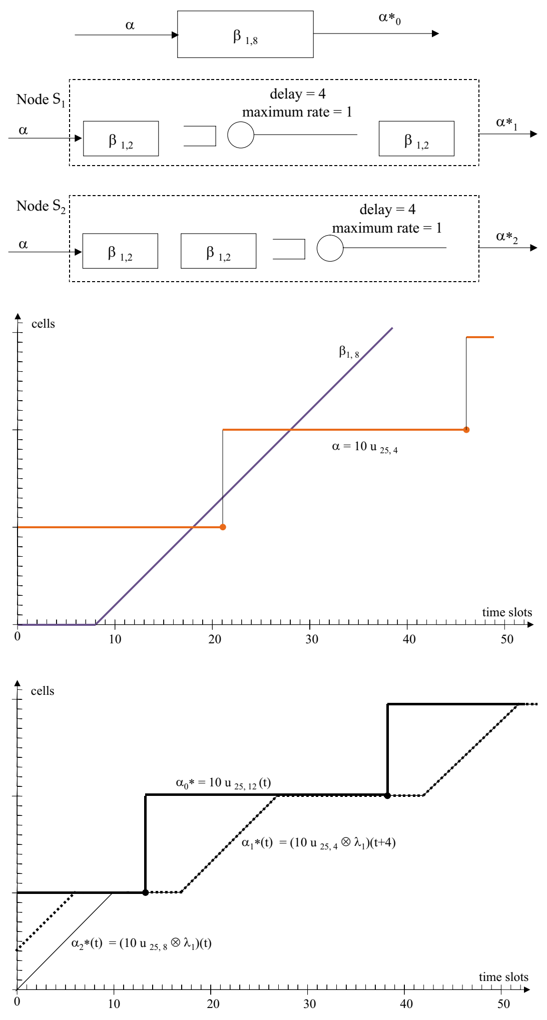

Figure 1.15: Use of maximum service curve to improve output bound. The figure is for the same example as Figure 1.15. Top: nodes $\mathcal{S}_1$ and $\mathcal{S}_2$, two possible implementations of a system offering the overall service curve $\beta_{1,8}$. Middle: arrival curve α and overall service curve $\beta_{1,8}$. Bottom: constraint for the output. α_0^* (top curve, thick, plain line) is obtained with the only knowledge that the service curve is $\beta_{1,8}$. α_1^* (middle curve, thick, dashed line) is obtained assuming the system is $\mathcal{S}_1$. α_2^* (bottom curve, thin, plain line) is obtained assuming the system is $\mathcal{S}_2$.

Then the backlog $B(t)$ and the virtual delay $d(t)$ satisfy

$$f(d(t) - v) \le B(t)$$

If in addition γ is super-additive, then

$$\beta(d(t)) \le B(t)$$

Proof: Fix some $t \ge 0$; we have$d(t) = \inf E_t$ where the set E_t is defined by

$$E_t = \{s \ge 0 : R^*(t+s) \ge R(t)\}$$

Since R^* and R are wide-sense increasing, E_t is an interval. Thus

$$d(t) = \sup\{s \ge 0 : R^*(t+s) < R(t)\}$$

We assume that R and R^* are left-continuous. It follows that

$$R^*(t + d(t)) \le R(t)$$

For some arbitrary ϵ, we can find some s such that

$$R^*(t + d(t)) \ge R(s) + \beta(t - s + d(t)) - \epsilon$$

Now from the maximum service curve property

$$R^*(t) - R(s) \le \gamma(t - s)$$

Combining the three gives

$$B(t) = R(t) - R^*(t) \ge \beta(t-s+d(t)) - \gamma(t-s) - \epsilon = \gamma(t-s+d(t)-v) - \gamma(t-s) - \epsilon$$

and thus

$$B(t) \ge \inf_{u \ge 0}[\gamma(d(t) - v + u) - \gamma(u)] \tag{1.17}$$

From the definition of f, the latter term is $f(d(t) - v)$. Finally, if γ is super-additive, then $\gamma \overline{\oslash} \gamma = \gamma$ □

We can apply the theorem to a practical case:

Corollary 1.6.1. *Assume a lossless node offers to a flow a minimum service curve $\beta = \beta_{r,v}$ and a maximum service curve $\gamma = \beta_{r,v'}$, with $v' \le v$. The backlog $B(t)$ and the virtual delay $d(t)$ satisfy*

$$d(t) \le \frac{B(t)}{r} + v$$

Proof: We apply the theorem and note that γ is super-additive, because it is convex. □

1.6.3 Variable versus Fixed Delay

Some network elements impose fixed delays (propagation and transmission), whereas some other network elements impose variable delays (queueing). In a number of cases, it is important to evaluate separately the total delay and the variable part of the delay. The total delay is important, for example, for determining throughput and response time; the variable part is important for dimensioning playout buffers (see Section 1.1.3 for a simple example, and chapter 5 for a more general discussion). We have seen at the end of end of Section 1.5.2 that a node imposing a constant delay can be modeled as a min-plus linear system. Beyond this, the concept of maximum service curve is a tool for telling apart variable delay from fixed delay, as follows.

Consider a network, made of a series of network elements $1, ..., I$, each element being the combination of a fixed delay d_i and a variable delay. Assume the variable delay component offers a service curve β_i. A fixed delay component offers δ_{d_i} both as a service curve and as a maximum service curve. Define $\beta = \beta_1 \otimes ... \otimes \beta_I$; the network offers as end-to-end service curve $\beta \otimes \delta_{d_1+...+d_I}$, and as end-to-end maximum service curve $\delta_{d_1+...+d_I}$. Assume the input flow is constrained by some arrival curve α; from Theorems 1.4.2 and 1.6.3, the end-to-delay $d(t)$ satisfies

$$d_1 + ... + d_I \leq d(t) \leq h(\alpha, \beta \otimes \delta_{d_1+...+d_I})$$

By simple inspection, $h(\alpha, \beta \otimes \delta_{d_1+...+d_I}) = d_1 + ... + d_I + h(\alpha, \beta)$, thus the end-to-end delay satisfies

$$0 \leq d(t) - [d_1 + ... + d_I] \leq h(\alpha, \beta)$$

In the formula, $d_1 + ... + d_I$ is the fixed part of the delay, and $h(\alpha, \beta)$ is the variable part. Thus, for the computation of the variable part of the delay, we can simply ignore fixed delay components.

Similarly, an arrival curve constraint for the output is

$$\alpha^* = (\alpha \otimes \delta_{d_1+...+d_I}) \oslash (\beta \otimes \delta_{d_1+...+d_I}) = \alpha \oslash \beta$$

thus the fixed delay can be ignored for the computation of the output bound.

For the determination of backlog, the alert reader can easily be convinced that fixed delays cannot be ignored. In summary:

Proposition 1.6.3. 1. *For the computation of backlog and fixed delay bounds, fixed or variable delay are modeled by introducing δ_T functions in the service curves. As a consequence of the commutativity of $\otimes$, such delays can be inserted in any order along a sequence of buffers, without altering the delay bounds.*

2. *For the computation of variable delay bounds, or for an arrival constraint on the output, fixed delays can be ignored.*

1.7 Handling Variable Length Packets

All results in this chapter apply directly to ATM systems, using discrete time models. In contrast, for variable length packets (as is usually the case with IP services), there are additional subtleties, which we now study in detail. The main parts in this section is the definition of a packetizer, and a study of its effect on delay, burstiness and backlog bounds. We also revisit the notion of shaper in a variable length context. For the rest of this section, time is continuous.

Throughout the section, we will consider some wide sense increasing sequences of packet arrival times $T_i \geq 0$. We assume that for all t the set $\{i : T_i \leq t\}$ is finite.

1.7.1 An Example of Irregularity Introduced by Variable Length Packets

The problem comes from the fact that real packet switching systems normally output entire packets, rather than a continuous data flow. Consider the example illustrated in Figure 1.16. It shows the output of a constant bit rate trunk, with rate c, that receives as input a sequence of packets, of different sizes. Call l_i, T_i the size (in bits) and the arrival epoch for the ith packet, $i = 1, 2, ...$. The input function is

$$R(t) = \sum_i l_i 1_{\{T_i \leq t\}} \tag{1.18}$$

In the formula, we used the indicator function $1_{\{expr\}}$which is equal to 1 if *expr* is true, and 0 otherwise.

We assume, as is usual in most systems, that we observe only entire packets delivered by the trunk. This is shown as $R'(t)$ in the figure, which results from the bit-by-bit output R^* by a packetization operation. The bit-by-bit output R^* is well understood; we know from Section 1.5 that $R^* = R \otimes \lambda_r$. However, what is the effect of packetization ? Do the results in Sections 1.4 and 1.5 still hold ?

Certainly, we should expect some modifications. For example, the bit-by-bit output R^* in the figure is the output of a greedy shaper with curve λ_c, thus it has λ_c as an arrival curve, but this is certainly not true for R'. Worse, we know that a greedy shaper keeps arrival constraints, thus if R is σ-smooth for some σ, then so is R^*. However, this is not true for R'. Consider the following example (which is originally from [31]). Assume that $\sigma(t) = l_{\max} + rt$ with $r < c$. Assume that the input flow $R(t)$ sends a first packet of size $l_1 = l_{\max}$ at time $T_1 = 0$, and a second packet of size l_2 at time $T_2 = \frac{l_2}{r}$. Thus the flow R is indeed σ-smooth. The departure time for the first packet is $T'_1 = \frac{l_{\max}}{c}$. Assume that the second packet l_2 is small, specifically, $l_2 < \frac{r}{c} l_{\max}$; then the two packets are sent back-to-back and thus the departure time for the second packet is $T'_2 = T'_1 + \frac{l_2}{c}$. Now the spacing $T'_2 - T'_1$ is less than $\frac{l_2}{r}$, thus the second packet is not conformant, in other words, R' is not σ-smooth. Note that this example is not possible if all packets are the same size.

We will see in this section that this example is quite general: packetizing variable length packets does introduce some additional irregularities. However, we are able

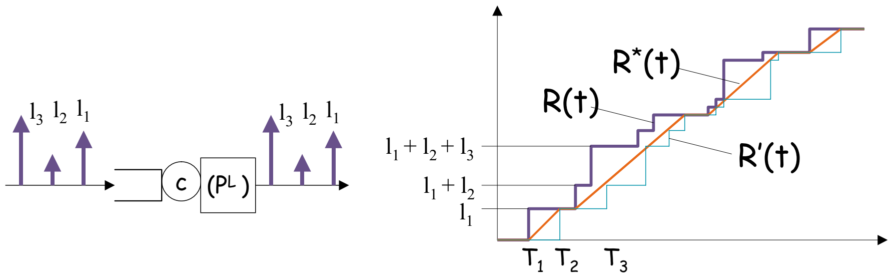

Figure 1.16: A real, variable length packet trunk of constant bit rate, viewed as the concatenation of a greedy shaper and a packetizer. The input is $R(t)$, the output of the greedy shaper is $R^*(t)$, the final output is the output of the packetizer is $R'(t)$.

to quantify them, and we will see that the irregularities are small (but may be larger than the order of a packet length). Most results are extracted from [47]

1.7.2 The Packetizer

We first need a few definitions.

Definition 1.7.1 (cumulative packet lengths). *A sequence L of cumulative packet lengths is a wide sense increasing sequence $(L(0) = 0, L(1), L(2), ...)$ such that*

$$l_{\max} = \sup_n \{L(n+1) - L(n)\}$$

is finite

In this chapter, we interpret $L(n) - L(n-1)$ as the length of the nth packet. We now introduce a new building block, which was introduced in [11].

Definition 1.7.2 (Function P^L [11]). *Consider a sequence of cumulative packet lengths L with $L(0) = 0$. For any real number x, define*

$$P^L(x) = \sup_{n \in \mathbb{N}} \{L(n) 1_{\{L(n) \le x\}}\} \tag{1.19}$$

Figure 1.17 illustrates the definition. Intuitively, $P^L(x)$ is the largest cumulative packet length that is entirely contained in x. Function P^L is right-continuous; if R is right-continuous, then so is $P^L(R(t))$. For example, if all packets have unit length, then $L(n) = n$ and for $x > 0$: $P^L(x) = \lfloor x \rfloor$. An equivalent characterization of P^L is

$$P^L(x) = L(n) \Longleftrightarrow L(n) \le x < L(n+1) \tag{1.20}$$

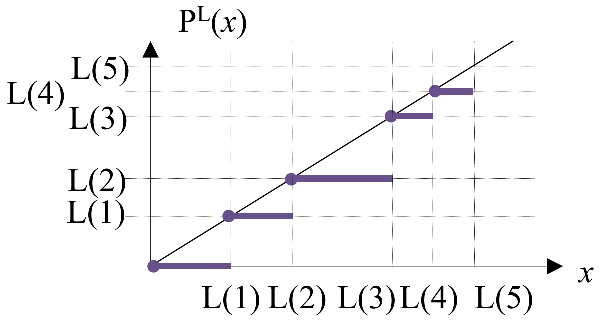

Figure 1.17: Definition of function P^L.

Definition 1.7.3 (Packetizer [11]). *Consider a sequence L of cumulative packet lengths. An L-packetizer is the system that transforms the input $R(t)$ into $P^L(R(t))$.*

For the example in Figure 1.16, we have $R'(t) = P^L(R^*(t))$ and the system can thus be interpreted as the concatenation of a greedy shaper and a packetizer.

The following equation follows immediately:

$$x - l_{\max} < P^L(x) \le x \tag{1.21}$$

Definition 1.7.4. *We say that a flow $R(t)$ is L-packetized if $P^L(R(t)) = R(t)$ for all t.*

The following properties are easily proven and left to the reader.

- (The packetizer is isotone) If $x \le y$ then $P^L(x) \le P^L(y)$ for all $x, y \in \mathbb{R}$.
- (P^L is idempotent) $P^L(P^L(x)) = P^L(x)$ for all $x \in \mathbb{R}$
- (Optimality of Packetizer) We can characterize a packetizer in a similar way as we did for a greedy shaper in Section 1.5. Among all flows $x(t)$ such that

  $$\begin{cases} x \text{ is } L\text{-packetized} \\ x \le R \end{cases} \tag{1.22}$$

  there is one that upper-bounds all, and it is $P^L(R(t))$.

  The proof for this last item mimics that of Lemma 1.5.1; it relies on the property that P^L is idempotent.

We now study the effect of packetizers on the three bounds found in Section 1.4. We first introduce a definition.

Definition 1.7.5 (Per-packet delay). *Consider a system with L- packetized input and output. Call T_i, T'_i the arrival and departure time for the ith packet. Assume there is no packet loss. The per-packet delay is $\sup_i(T'_i - T_i)$*

Our main result in this section is the following theorem, illustrated in Figure 1.18.

Theorem 1.7.1 (Impact of packetizer). *Consider a system (*bit-by-bit system*) with L-packetized input R and bit-by-bit output R^*, which is then L-packetized to produce a final packetized output R'. We call* combined system *the system that maps R into R'. Assume both systems are first-in-first-out and lossless.*

1. *The* per-packet delay *for the combined system is the maximum virtual delay for the bit-by-bit system.*
2. *Call B^* the maximum backlog for the bit-by-bit system and B' the maximum backlog for the combined system. We have*

$$B^* \leq B' \leq B^* + l_{\max}$$

3. *Assume that the bit-by-bit system offers to the flow a maximum service curve γ and a minimum service curve β. The combined system offers to the flow a maximum service curve γ and a minimum service curve β' given by*

$$\beta'(t) = [\beta(t) - l_{\max}]^+$$

4. *If some flow $S(t)$ has $\alpha(t)$ as an arrival curve, then $P^L(S(t))$ has $\alpha(t) + l_{\max} 1_{\{t>0\}}$ as an arrival curve.*

The proof of the theorem is given later in this section. Before, we discuss the implications. Item 1 says that appending a packetizer to a node does not increase

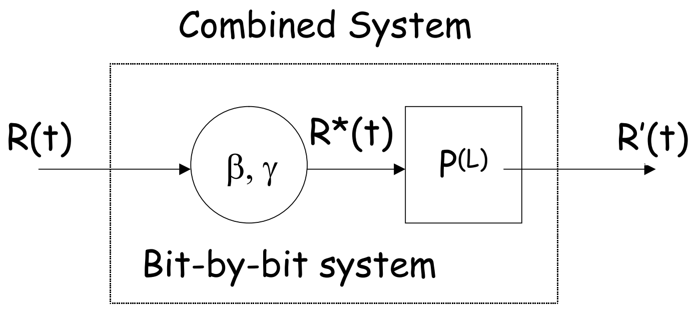

Figure 1.18: The scenario and notation in Theorem 1.7.1.

the packet delay at this node. However, as we see later, packetization does increase the end-to-end delay.

Consider again the example in Section 1.7.1. A simple look at the figure shows that the backlog (or required buffer) is increased by the packetization, as indicated by item 2. Item 4 tells us that the final output R' has $\sigma'(t) = \sigma(t) + l_{\max}1_{t>0}$ as an arrival curve, which is consistent with our observation in Section 1.7.1 that R' is not σ-smooth, even though R^* is. We will see in Section 1.7.4 that there is a stronger result, in relation with the concept of "packetized greedy shaper".

Item 3 is the most important practical result in this section. It shows that packetizing weakens the service curve guarantee by one maximum packet length. For example, if a system offers a rate-latency service curve with rate R, then appending a packetizer to the system has the effect of increasing the latency by $\frac{l_{\max}}{R}$.

Consider also the example in Figure 1.16. The combination of the trunk and the packetizer can be modeled as a system offering

- a minimum service curve $\beta_{c,\frac{l_{\max}}{c}}$
- a maximum service curve λ_c

Proof of Theorem 1.7.1

1. For some t such that $T_i \le t < T_{i+1}$ we have $R(t) = L(i)$ and thus
$$\sup_{t\in[T_i,T_{i+1})} d(t) = d(T_i)$$
now
$$d(T_i) = T'_i - T_i$$
Combining the two shows that
$$\sup_t d(t) = \sup_i (T'_i - T_i)$$

2. The proof is a direct consequence of Equation (1.21).

3. The result on maximum service curve γ follows immediately from Equation (1.21). Consider now the minimum service curve property. Fix some time t. For $T_i \le s < T_{i+1}$ we have $R(s) = R(T_i)$ and β is wide-sense increasing, thus
$$\inf_{T_i\le s<T_{i+1}} (R(s) + \beta(t-s)) = R(T_i) + \beta_r(t - T_i)$$
where $\beta_r(t-T_i) = \inf_{\epsilon>0}\{\beta[(t-T_i)+\epsilon]\}$ is the limit of β to the right. Thus
$$(R\otimes\beta)(t) = \inf_{i \text{ such that } T_i\le t} (R(T_i) + \beta_r(t - T_i))$$
For a fixed t there is only a finite number of i such that $T_i \le t$, thus the $\inf$ in the previous equation is a $\min$ and there is some j such that
$$(R\otimes\beta)(t) = R(T_j) + \beta_r(t - T_j)$$
By hypothesis, $R^*(t) \ge (R\otimes\beta)(t)$, thus

$$R'(t) \geq R^*(t) - l_{\max} \geq R(T_j) + \beta_r(t - T_j) - l_{\max}$$

On the other hand, $R^*(t) \geq R(T_j)$ and R is L-packetized, thus

$$R'(t) \geq R(T_j)$$

Combining the two shows that

$$\begin{array}{rl} R'(t) \geq & \max\left[R(T_j), R(T_j) + \beta_r(t - T_j) - l_{\max}\right] \\ = & R(T_j) + \max\left[\beta_r(t - T_j) - l_{\max}, 0\right] \\ = & R(T_j) + \beta'_r(t - T_j) \end{array}$$

from which we conclude that $R'(t) \geq \inf_{0 \leq s \leq t}\left(R(s) + \beta'(t - s)\right)$

4. The proof is a direct consequence of Equation (1.21).

Example: concatenation of GPS nodes Consider the concatenation of the theoretical GPS node, with guaranteed rate R (see Section 1.3.1 on Page 22) and an L-packetizer. Assume this system receives a flow of variable length packets. This models a theoretical node that would work as a GPS node but is constrained to deliver entire packets. This is not very realistic, and we will see in Chapter 2 more realistic implementations of GPS, but this example is sufficient to explain one important effect of packetizers.

By applying Theorem 1.7.1, we find that this node offers a rate-latency service curve $\beta_{R, \frac{l_{\max}}{R}}$. Now concatenate m such identical nodes, as illustrated in Figure 1.19. The end-to-end service curve is the rate latency-function $\beta_{R,T}$ with

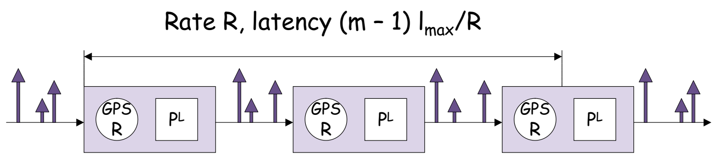

Figure 1.19: The concatenation of several GPS fluid nodes with packetized outputs

$$T = m\frac{l_{\max}}{R}$$

We see on this example that the additional latency introduced by one packetizer is indeed of the order of one packet length; however, this effect is multiplied by the number of hops.

For the computation of the end-to-end delay bound, we need to take into account Theorem 1.7.1, which tells us that we can forget the last packetizer. Thus, a bound on end-to-end delay is obtained by considering that the end-to-end path offers a service curve equal to the latency-function β_{R,T_0} with

$$T_0 = (m-1)\frac{l_{\max}}{R}$$

For example, if the original input flow is constrained by one leaky bucket of rate r and bucket pool of size b, then an end-to-end delay bound is

$$\frac{b + (m-1)l_{\max}}{R} \tag{1.23}$$

The alert reader will easily show that this bound is a worst case bound. This illustrates that we should be careful in interpreting Theorem 1.7.1. It is only at the last hop that the packetizer implies no delay increase. The interpretation is as follows. Packetization delays the first bits in a packet, which delays the processing at downstream nodes. This effect is captured in Equation (1.23). In summary:

Remark 1.7.1. *Packetizers do not increase the maximum delay at the node where they are appended. However, they generally increase the end-to-end delay.*

We will see in Chapter 2 that many practical schedulers can be modeled as the concatenation of a node offering a service curve guarantee and a packetizer, and we will give a practical generalization of Equation (1.23).

1.7.3 A Relation between Greedy Shaper and Packetizer

We have seen previously that appending a packetizer to a greedy shaper weakens the arrival curve property of the output. There is however a case where this is not true. This case is important for the results in Section 1.7.4, but also has practical applications of its own. Figure 1.20 illustrates the theorem.

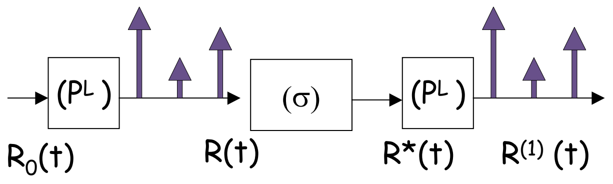

Figure 1.20: Theorem 1.7.2 says that $R^{(1)}$ is σ-smooth.

Theorem 1.7.2. *Consider a sequence L of cumulative packet lengths and call $\mathcal{P}_L$ the L-packetizer. Consider a "good" function σ and assume that*

$$\left\{ \begin{array}{l} \text{There is a sub-additive function } \sigma_0 \text{ and a number } l \geq l_{\max} \text{ such that} \\ \sigma(t) = \sigma_0(t) + l 1_{t>0} \end{array} \right. \tag{1.24}$$

Call $\mathcal{C}_\sigma$ the greedy shaper with shaping curve σ. For any input, the output of the composition[6]$\mathcal{P}_L \circ \mathcal{C}_\sigma \circ \mathcal{P}_L$ *is σ-smooth.*

In practical terms, the theorem is used as follows. Consider an L-packetized flow, pass it through a greedy shaper with shaping curve σ; and packetize the output; then the result is σ-smooth (assuming that σ satisfies condition in Equation (1.24) in the theorem).

Note that in general the output of $\mathcal{C}_\sigma \circ \mathcal{P}_L$ is *not* L-packetized, even if σ satisfies the condition in the theorem (finding a counter-example is simple and is left to the reader for her enjoyment). Similarly, if the input to $\mathcal{P}_L \circ \mathcal{C}_\sigma$ is not L-packetized, then the output is not σ-smooth, in general.

The theorem could also be rephrased by saying that, under condition in Equation (1.24)

$$\mathcal{P}_L \circ \mathcal{C}_\sigma \circ \mathcal{P}_L = \mathcal{C}_\sigma \circ \mathcal{P}_L \circ \mathcal{C}_\sigma \circ \mathcal{P}_L$$

since the two above operators always produce the same output.

Discussion of Condition in Equation (1.24) Condition Equation (1.24) is satisfied in practice if σ is concave and $\sigma_r(0) \geq l_{\max}$, where $\sigma_r(0) = \inf_{t>0} \sigma(t)$ is the limit to the right of σ at 0. This occurs for example if the shaping curve is defined by the conjunction of leaky buckets, all with bucket size at least as large as the maximum packet size.

This also sheds some light on the example in Figure 1.16: the problem occurs because the shaping curve λ_C does not satisfy the condition.

The alert reader will ask herself whether a sufficient condition for Equation (1.24) to hold is that σ is sub-additive and $\sigma_r(0) \geq l_{\max}$. Unfortunately, the answer is no. Consider for example the stair function $\sigma = l_{\max} v_T$. We have $\sigma_r(0) = l_{\max}$ but if we try to rewrite σ into $\sigma(t) = \sigma_0(t) + l 1_{t>0}$ we must have $l = l\max$ and $\sigma_0(t) = 0$ for $t \in (0, T]$; if we impose that σ_0 is sub-additive, the latter implies $\sigma_0 = 0$ which is not compatible with Equation (1.24).[7]

Proof of Theorem 1.7.2: We use the notation in Figure 1.20. We want to show that $R^{(1)}$ is σ-smooth. We have $R^* = R \otimes \sigma$. Consider now some arbitrary s and t with $s < t$. From the definition of min-plus convolution, for all $\epsilon > 0$, there is some $u \leq s$ such that

$$(R \otimes \sigma)(s) \geq R(u) + \sigma(s - u) - \epsilon \tag{1.25}$$

Now consider the set E of $\epsilon > 0$ such that we can find one $u < s$ satisfying the above equation. Two cases are possible: either 0 is an accumulation point for E[8] (case 1) , or not (case 2).

Consider case 1; there is a sequence (ϵ_n, s_n), with $s_n < s$,

[6]We use the notation $\mathcal{P}_L \circ \mathcal{C}_\sigma$ to denote the composition of the two operators, with $\mathcal{C}_\sigma$ applied first; see Section 4.1.3.

[7]The same conclusion unfortunately also holds if we replace sub-additive by "star-shaped" (Section 3.1).

[8]namely, there is a sequence of elements in E which converges to 0

$$\lim_{n\to+\infty} \epsilon_n = 0$$

and

$$(R \otimes \sigma)(s) \geq R(s_n) + \sigma(s - s_n) - \epsilon_n$$

Now since $s_n \leq t$:

$$(R \otimes \sigma)(t) \leq R(s_n) + \sigma(t - s_n)$$

Combining the two:

$$(R \otimes \sigma)(t) - (R \otimes \sigma)(s) \leq \sigma(t - s_n) - \sigma(s - s_n) + \epsilon_n$$

Now $t - s_n > 0$ and $s - s_n > 0$ thus

$$\sigma(t - s_n) - \sigma(s - s_n) = \sigma_0(t - s_n) - \sigma_0(s - s_n)$$

We have assumed that σ_0 is sub-additive. Now $t \geq s$ thus

$$\sigma_0(t - s_n) - \sigma_0(s - s_n) \leq \sigma_0(t - s)$$

we have thus shown that, for all n

$$(R \otimes \sigma)(t) - (R \otimes \sigma)(s) \leq \sigma_0(t - s) + \epsilon_n$$

and thus

$$(R \otimes \sigma)(t) - (R \otimes \sigma)(s) \leq \sigma_0(t - s)$$

Now from Equation (1.21), it follows that

$$R^{(1)}(t) - R^{(1)}(s) \leq \sigma_0(t - s) + l_{\max} \leq \sigma(t - s)$$

which ends the proof for case 1.

Now consider case 2. There is some ϵ_0 such that for $0 < \epsilon < \epsilon_0$, we have to take $u = s$ in Equation (1.25). This implies that

$$(R \otimes \sigma)(s) = R(s)$$

Now R is L-packetized by hypothesis. Thus

$$R^{(1)}(s) = P^L((R \otimes \sigma)(s)) = P^L(R(s)) = R(s) = (R \otimes \sigma)(s)$$

thus

$$\begin{aligned} R^{(1)}(t) - R^{(1)}(s) = & \quad P^L((R \otimes \sigma)(t) - (R \otimes \sigma)(s) \\ \leq & \quad (R \otimes \sigma)(t) - (R \otimes \sigma)(s) \end{aligned}$$

now $R \otimes \sigma$ has σ as an arrival curve thus

$$R^{(1)}(t) - R^{(1)}(s) \leq \sigma(t - s)$$

which ends the proof for case 2. □

Example: Buffered Leaky Bucket Controller based on Virtual Finish Times Theorem 1.7.2 gives us a practical implementation for a packet based shaper. Consider that we want to build a device that ensures that a packet flow satisfies some concave, piecewise linear arrival curve (and is of course L- packetized). We can realize such a device as the concatenation of a buffered leaky bucket controller operating bit-by-bit and a packetizer. We compute the output time for the last bit of a packet (= finish time) under the bit-by-bit leaky bucket controller, and release the entire packet instantly at this finish time. If each bucket pool is at least as large as the maximum packet size then Theorem 1.7.2 tells us that the final output satisfies the leaky bucket constraints.

Counter-example If we consider non-concave arrival curves, then we can find an arrival curve σ that does satisfy $\sigma(t) \geq l_{\max}$ for $t > 0$ but that does not satisfy Equation (1.24). In such a case, the conclusion of Theorem 1.7.2 may not hold in general. Figure 1.21 shows an example where the output $R^{(1)}$ is not σ-smooth, when σ is a stair function.

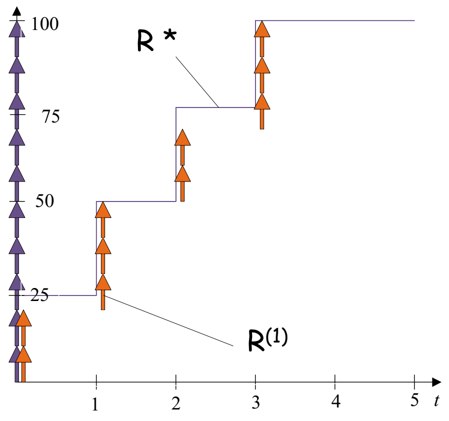

Figure 1.21: A counter example for Theorem 1.7.2. A burst of 10 packets of size equal to 10 data units arrive at time $t = 0$, and $\sigma = 25v_1$. The greedy shaper emits 25 data units at times 0 and 1, which forces the packetizer to create a burst of 3 packets at time 1, and thus $R^{(1)}$ is not σ-smooth.

1.7.4 Packetized Greedy Shaper

We can come back to the questions raised by the example in Figure 1.16 and give a more fundamental look at the issue of packetized shaping. Instead of synthesizing the concatenation of a greedy shaper and a packetizer as we did earlier, we define the following, consistent with Section 1.5.

Definition 1.7.6. *[Packetized Greedy Shaper] Consider an input sequence of packets, represented by the function $R(t)$ as in Equation (1.18). Call L the cumulative packet lengths. We call* packetized shaper, *with shaping curve σ, a system that forces its output to have σ as an arrival curve* and *be L-packetized. We call* packetized greedy shaper *a packetized shaper that delays the input packets in a buffer, whenever sending a packet would violate the constraint σ, but outputs them as soon as possible.*

Example: Buffered Leaky Bucket Controller based on Bucket Replenishment The case $\sigma(t) = \min_{m=1,\ldots,M}(\gamma_{r_m,b_m}(t)$ can be implemented by a controller that observes a set of M fluid buckets, where the mth bucket is of size b_m and leaks at a constant rate r_m. Every bucket receives l_i units of fluid when packet i is released (l_i is the size of packet i). A packet is released as soon as the level of fluid in bucket m allows it, that is, has gone down below $b_m - l_i$, for all m. We say that now we have defined a buffered leaky bucket controller based on "bucket replenishment". It is clear that the output has σ as an arrival curve, is L-packetized and sends the packets as early as possible. Thus it implements the packetized greedy shaper. Note that this implementation differs from the buffered leaky bucket controller based on virtual finish times introduced in Section 1.7.3. In the latter, during a period where, say, bucket m only is full, fragments of a packet are virtually released at rate r_m, bucket m remains full, and the (virtual) fragments are then re-assembled in the packetizer; in the former, if a bucket becomes full, the controller waits until it empties by at least the size of the current packet. Thus we expect that the level of fluid in both systems is not the same, the former being an upper bound. We will see however in Corollary 1.7.1 that both implementations are equivalent.

In this example, if a bucket size is less than the maximum packet size, then it is never possible to output a packet: all packets remain stuck in the packet buffer, and the output is $\overline{R}(t) = 0$. In general, we can say that

Proposition 1.7.1. *If $\sigma_r(0) < l_{\max}$ then the the packetized greedy shaper blocks all packets for ever (namely, $\overline{R}(t) = 0$). Thus in this section, we assume that $\sigma(t) \geq l_{\max}$ for $t > 0$.*

Thus, for practical cases, we have to assume that the arrival curve σ has a discontinuity at the origin at least as large as one maximum packet size.

How does the packetized greedy shaper compare with the concatenation of a greedy shaper with shaping curve σ and a packetizer ? We know from the example in Figure 1.16 that the output has $\sigma'(t) = \sigma(t)+l_{\max}1_{t>0}$ as an arrival curve, but not σ. Now, does the concatenation implement a packetized greedy shaper with shaping curve σ' ? Before giving a general answer, we study a fairly general consequence of Theorem 1.7.2.

Theorem 1.7.3 (Realization of packetized Greedy Shaper). *Consider a sequence L of cumulative packet lengths and a "good" function σ. Assume that σ satisfies the condition in Equation (1.24). Consider only inputs that are L packetized. Then the* *packetized greedy shaper for σ and L can be realized as the concatenation of the greedy shaper with shaping curve σ and the L-packetizer.*

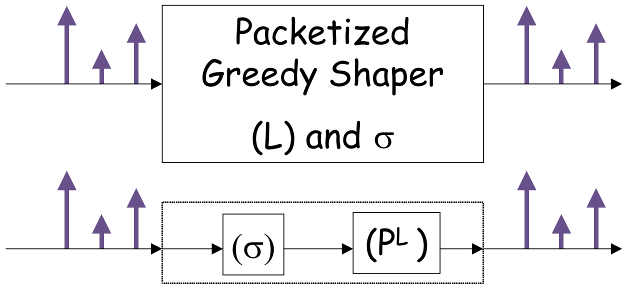

Figure 1.22: The packetized greedy shaper can be realized as a (bit-by-bit fluid shaper followed by a packetizer, assuming Equation (1.24) holds. In practice, this means that we can realize packetized greedy shaping by computing finish times in the virtual fluid system and release packets at their finish times.

Proof: Call $R(t)$ the packetized input; the output of the bit-by-bit greedy shaper followed by a packetizer is $R^{(1)}(t) = P^L(R \otimes \sigma)(t))$. Call $\overline{R}(t)$ the output of the packetized greedy shaper. We have $\overline{R} \leq R$ thus $\overline{R} \otimes \sigma \leq R \otimes \sigma$ and thus

$$P^L(\overline{R} \otimes \sigma) \leq P^L(R \otimes \sigma)$$

But $\overline{R}$ is σ-smooth, thus $\overline{R} \otimes \sigma = \overline{R}$, and is L-packetized, thus $P^L(\overline{R} \otimes \sigma) = \overline{R}$. Thus the former inequality can be rewritten as $\overline{R} \leq R^{(1)}$. Conversely, from Theorem 1.7.2, $R^{(1)}$ is also σ-smooth and L-packetized. The definition of the packetized greedy shaper implies that $\overline{R} \geq R^{(1)}$ (for a formal proof, see Lemma 1.7.1) thus finally $\overline{R} = R^{(1)}$. □

We have seen that the condition in the theorem is satisfied in particular if σ is concave and $\sigma_r(0) \geq l_{\max}$, for example if the shaping curve is defined by the conjunction of leaky buckets, all with bucket size at least as large as the maximum packet size. This shows the following.

Corollary 1.7.1. *For L-packetized inputs, the implementations of buffered leaky bucket controllers based on bucket replenishment and virtual finish times are equivalent.*

If we relax Equation (1.24) then the construction of the packetized greedy shaper is more complex:

Theorem 1.7.4 (I/O characterisation of packetized greedy shapers). *Consider a packetized greedy shaper with shaping curve σ and cumulative packet length L. Assume that σ is a "good" function. The output $\overline{R}(t)$ of the packetized greedy shaper is given by*

$$\overline{R} = \inf\left\{R^{(1)}, R^{(2)}, R^{(3)}, ...\right\} \tag{1.26}$$

with $R^{(1)}(t) = P^L((\sigma \otimes R)(t))$ and $R^{(i)}(t) = P^L((\sigma \otimes R^{(i-1)})(t))$ for $i \geq 2$.

Figure 1.23 illustrates the theorem, and shows the iterative construction of the output on one example. Note that this example is for a shaping function that does not satisfy Equation (1.24). Indeed, otherwise, we know from Theorem 1.7.3 that the iteration stops at the first step, namely, $\overline{R} = R^{(1)}$ in that case. We can also check for example that if $\sigma = \lambda_r$ (thus the condition in Proposition 1.7.1 is satisfied) then the result of Equation (1.26) is 0.

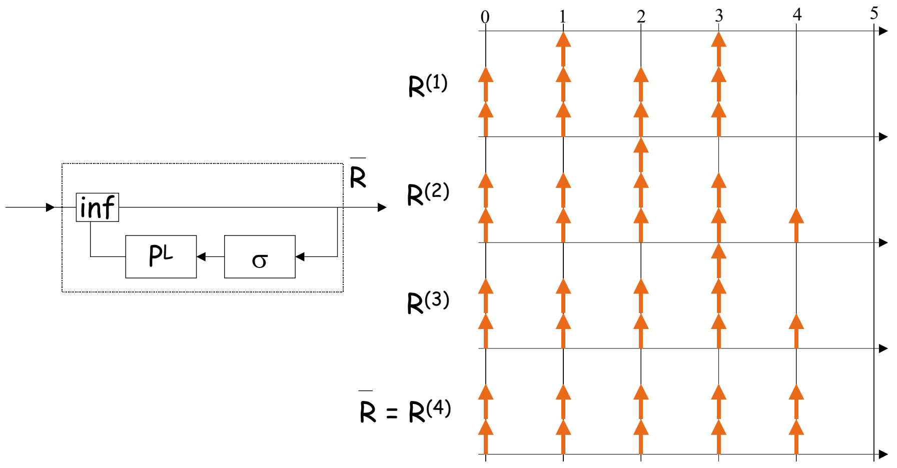

Figure 1.23: Representation of the output of the packetized greedy shaper (left) and example of output (right). The data are the same as with Figure 1.21.

Proof: The proof is a direct application of Lemma 1.7.1 (which itself is an application of the general method in Section 4.3 on Page 175). □

Lemma 1.7.1. *Consider a sequence L of cumulative packet lengths and a "good" function σ. Among all flows $x(t)$ such that*

$$\begin{cases} x \leq R \\ x \text{ is } L\text{-packetized} \\ x \text{ has } \sigma \text{ as an arrival curve} \end{cases} \tag{1.27}$$

there is one flow $\overline{R}(t)$ that upper-bounds all. It is given by Equation (1.26).

Proof: The lemma is a direct application of Theorem 4.3.1, as explained in Section 4.3.2. However, in order to make this chapter self-contained, we give an alternative, direct proof, which is quite short.

If x is a solution, then it is straightforward to show by induction on i that $x(t) \leq R^{(i)}(t)$ and thus $x \leq \overline{R}$. The difficult part is now to show that $\overline{R}$ is indeed a solution. We need to show that the three conditions in Equation (1.27) hold. Firstly, $R^{(1)} \leq R(t)$ and by induction on i, $R^{(i)} \leq R$ for all i; thus $\overline{R} \leq R$.

Secondly, consider some fixed t; $R^{(i)}(t)$ is L-packetized for all $i \geq 1$. Let $L(n_0) := R^{(1)}(t)$. Since $R^{(i)}(t) \leq R^{(1)}(t)$, $R^{(i)}(t)$ is in the set

$$\{L(0), L(1), L(2), ..., L(n_0)\}.$$

This set is finite, thus, $\overline{R}(t)$, which is the infimum of elements in this set, has to be one of the $L(k)$ for $k \leq n_0$. This shows that $\overline{R}(t)$ is L-packetized, and this is true for any time t.

Thirdly, we have, for all i

$$\overline{R}(t) \leq R^{(i+1)}(t) = P^L((\sigma \otimes R^{(i)})(t)) \leq (\sigma \otimes R^{(i)})(t)$$

thus

$$\overline{R} \leq \inf_i (\sigma \otimes R^{(i)})$$

Now convolution by a fixed function is upper-semi-continuous, which means that

$$\inf_i (\sigma \otimes R^{(i)}) = \sigma \otimes \overline{R}$$

This is a general result in Chapter 4 for any min-plus operator. An elementary proof is as follows.

$$\begin{array}{rcl}
\inf_i (\sigma \otimes R^{(i)})(t) = & & \inf_{s \in [0,t], i \in \mathbb{N}} \left[\sigma(s) + R^{(i)}(t-s)\right] \\
& = & \inf_{s \in [0,t]} \left\{\inf_{i \in \mathbb{N}} \left[(\sigma(s) + R^{(i)}(t-s)\right]\right\} \\
& = & \inf_{s \in [0,t]} \left\{\sigma(s) + \inf_{i \in \mathbb{N}} \left[R^{(i)}(t-s)\right]\right\} \\
& = & \inf_{s \in [0,t]} \left[\sigma(s) + \overline{R}(t-s)\right] \\
& = & (\sigma \otimes \overline{R})(t)
\end{array}$$

Thus

$$\overline{R} \leq \sigma \otimes \overline{R},$$

which shows the third condition. Note that $\overline{R}$ is wide-sense increasing. □

Does a packetized greedy shaper keep arrival constraints ? Figure 1.24 shows a counter-example, namely, a variable length packet flow that has lost its initial arrival curve constraint after traversing a packetized greedy shaper.

However, if arrival curves are defined by leaky buckets, we have a positive result.

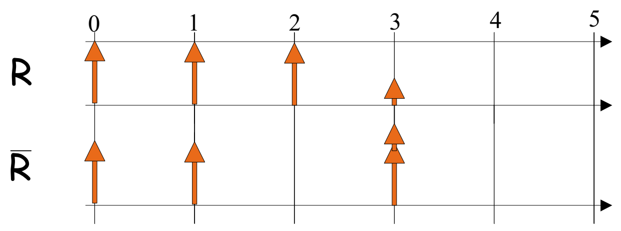

Figure 1.24: The input flow is shown above; it consists of 3 packets of size 10 data units and one of size 5 data units, spaced by one time unit. It is α-smooth with $\alpha = 10v_{1,0}$. The bottom flow is the output of the packetized greedy shaper with $\sigma = 25v_{3,0}$. The output has a burst of 15 data units packets at time 3. It is σ-smooth but *not* α-smooth.

Theorem 1.7.5 (Conservation of concave arrival constraints). *Assume an L-packetized flow with arrival curve α is input to a packetized greedy shaper with cumulative packet length L and shaping curve σ. Assume that α and σ are concave with $\alpha_r(0) \geq l_{\max}$ and $\sigma_r(0) \geq l_{\max}$. Then the output flow is still constrained by the original arrival curve α.*

Proof: Since σ satisfies Equation (1.24), it follows from Theorem 1.7.3 that $\overline{R} = P^L(\sigma \otimes R)$. Now R is α-smooth thus it is not modified by a bit-by-bit greedy shaper with shaping curve α, thus $R = \alpha \otimes R$. Combining the two and using the associativity of $\otimes$ gives $\overline{R} = P^L[(\sigma \otimes \alpha) \otimes R]$. From our hypothesis, $\sigma \otimes \alpha = \min(\sigma, \alpha)$ (see Theorem 3.1.6 on Page 136) and thus $\sigma \otimes \alpha$ satisfies Equation (1.24). Thus, by Theorem 1.7.2, $\overline{R}$ is $\sigma \otimes \alpha$-smooth, and thus α-smooth. $\square$

Series decomposition of shapers

Theorem 1.7.6. *Consider a tandem of M packetized greedy shapers in series; assume that the shaping curve σ^m of the mth shaper is concave with $\sigma^m_r(0) \geq l_{\max}$. For L-packetized inputs, the tandem is equivalent to the packetized greedy shaper with shaping curve $\sigma = \min_m \sigma^m$.*

Proof: We do the proof for $M = 2$ as it extends without difficulty to larger values of M. Call $R(t)$ the packetized input, $R'(t)$ the output of the tandem of shapers, and $\overline{R}(t)$ the output of the packetized greedy shaper with input $R(t)$.

Firstly, by Theorem 1.7.3

$$R' = P^L[\sigma_2 \otimes P^L(\sigma^1 \otimes R)]$$

Now $\sigma^m \geq \sigma$ for all m thus

$$R' \geq P^L[\sigma \otimes P^L(\sigma \otimes R)]$$

Again by Theorem 1.7.3, we have $\overline{R} = P^L(\sigma \otimes R)$. Moreover $\overline{R}$ is L-packetized and σ-smooth, thus $\overline{R} = P^L(\overline{R})$ and $\overline{R} = \sigma \otimes \overline{R}$. Thus finally

$$R' \geq \overline{R} \tag{1.28}$$

Secondly, R' is L-packetized and by Theorem 1.7.5, it is σ-smooth. Thus the tandem is a packetized (possibly non greedy) shaper. Since $\overline{R}(t)$ is the output of the packetized greedy shaper, we must have $R' \leq \overline{R}$. Combining with Equation (1.28) ends the proof. □

It follows that a shaper with shaping curve $\sigma(t) = \min_{m=1,\ldots,M}(r_m t + b_m)$, where $b_m \geq l_{\max}$ for all m, can be implemented by a tandem of M individual leaky buckets, in any order. Furthermore, by Corollary 1.7.1, every individual leaky bucket may independently be based either on virtual finish times or on bucket replenishment.

If the condition in the theorem is not satisfied, then the conclusion may not hold. Indeed, for the example in Figure 1.24, the tandem of packetized greedy shapers with curves α and σ does not have an α-smooth output, therefore it cannot be equivalent to the packetized greedy shaper with curve $\min(\alpha, \sigma)$.

Unfortunately, the other shaper properties seen in Section 1.5 do not generally hold. For shaping curves that satisfy Equation (1.24), and when a packetized greedy shaper is introduced, we need to compute the end-to-end service curve by applying Theorem 1.7.1.

1.8 Lossless Effective Bandwidth and Equivalent Capacity

1.8.1 Effective Bandwidth of a Flow

We can apply the results in this chapter to define a function of a flow called the effective bandwidth. This function characterizes the bit rate required for a given flow. More precisely, consider a flow with cumulative function R; for a fixed, but arbitrary delay D, we define the *effective bandwidth* $e_D(R)$ of the flow as the bit rate required to serve the flow in a work conserving manner, with a virtual delay $\leq D$.

Proposition 1.8.1. *The effective bandwidth of a flow is given by*

$$e_D(R) = \sup_{0 \leq s \leq t} \frac{R(t) - R(s)}{t - s + D} \tag{1.29}$$

For an arrival curve α we define the effective bandwidth $e_D(\alpha)$ as the effective bandwidth of the greedy flow $R = \alpha$. By a simple manipulation of Equation 1.29, the following comes.

Proposition 1.8.2. *The effective bandwidth of a "good" arrival curve is given by*

$$e_D(\alpha) = \sup_{0 \le s} \frac{\alpha(s)}{s+D} \tag{1.30}$$

The alert reader will check that the effective bandwidth of a flow R is also the effective bandwidth of its minimum arrival curve $R \oslash R$. For example, for a flow with T-SPEC (p, M, r, b), the effective bandwidth is the maximum of r and the slopes of lines (QA_0) and (QA_1) in Figure 1.25; it is thus equal to:

$$e_D = \max\left\{\frac{M}{D}, r, p\left(1 - \frac{D - \frac{M}{p}}{\frac{b-M}{p-r} + D}\right)\right\} \tag{1.31}$$

Assume α is sub-additive. We define the sustainable rate m as $m = \liminf_{s \to +\infty} \frac{\alpha(s)}{s}$

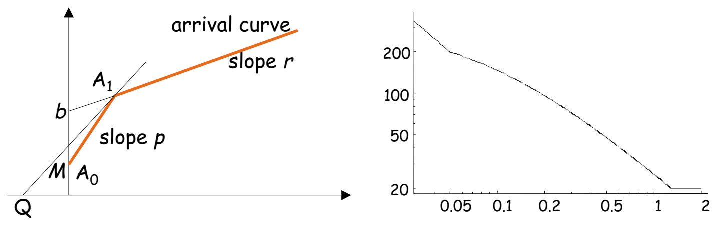

Figure 1.25: Computation of Effective Bandwidth for a VBR flow (left); example for $r = 20$ packets/second, $M = 10$ packets, $p = 200$ packets per second and $b = 26$ packets (right).

and the peak rate by $p = \sup_{s>0} \frac{\alpha(s)}{s}$. Then $m \le e_D(\alpha) \le p$ for all D. Moreover, if α is concave, then $\lim_{D \to +\infty} e_D(\alpha) = m$.If α is differentiable, $e(D)$ is the slope of the tangent to the arrival curve, drawn from the time axis at $t = -D$ (Figure 1.26). It follows also directly from the definition in (1.29) that

$$e_D(\sum_i \alpha_i) \le \sum_i e_D(\alpha_i) \tag{1.32}$$

In other words, the effective bandwidth for an aggregate flow is less than or equal to the sum of effective bandwidths. If the flows have all *identical* arrival curves, then the aggregate effective bandwidth is simply $I \times e_D(\alpha_1)$. It is this latter relation that is the origin of the term "effective bandwidth". The difference $\sum_i e_D(\alpha_i) - e_D(\sum_i \alpha_i)$ is a buffering gain; it tells us how much capacity is saved by sharing a buffer between the flows.

1.8.2 Equivalent Capacity

Similar results hold if we replace delay constraints by the requirement that a fixed buffer size is not exceeded. Indeed, the queue with constant rate C, guarantees a maximum backlog of B (in bits) for a flow R if $C \ge f_B(R)$, with

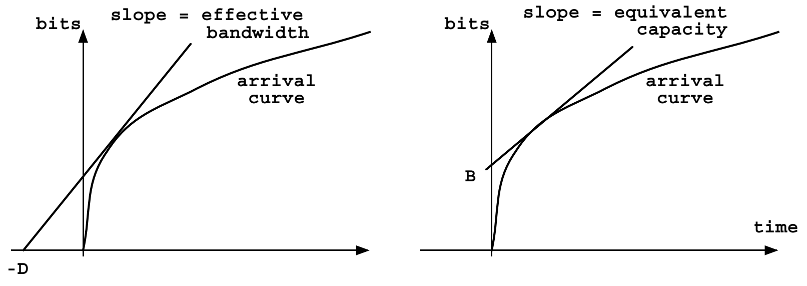

Figure 1.26: Effective Bandwidth for a delay constraint D and Equivalent Capacity for a buffer size B

$$f_B(R) = \sup_{0 \le s < t} \frac{R(t) - R(s) - B}{t - s} \tag{1.33}$$

Similarly, for a “good” function α, we have:

$$f_B(\alpha) = \sup_{s > 0} \frac{\alpha(s) - B}{s} \tag{1.34}$$

We call $f_B(\alpha)$ the *equivalent capacity*, by analogy to [44]. Similar to effective bandwidth, the equivalent capacity of a heterogeneous mix of flows is less than or equal to the sum of equivalent capacities of the flows, provided that the buffers are also added up; in other words, $f_B(\alpha) \le \sum_i f_{B_i}(\alpha_i)$, with $\alpha = \sum_i \alpha_i$ and $B = \sum_i B_i$. Figure 1.26 gives a graphical interpretation.

For example, for a flow with T-SPEC (p, M, r, b), using the same method as above, we find the following equivalent capacity:

$$f_B = \left\{ \begin{array}{l} \text{if } B < M \text{ then } +\infty \\ \text{else } r + \frac{(p-r)(b-B)^+}{b-M} \end{array} \right. \tag{1.35}$$

An immediate computation shows that $f_b(\gamma_{r,b}) = r$. In other words, if we allocate to a flow, constrained by an affine function $\gamma_{r,b}$, a capacity equal to its sustainable rate r, then a buffer equal to its burst tolerance b is sufficient to ensure loss-free operation.

Consider now a mixture of Intserv flows (or VBR connections), with T-SPECs (M_i, p_i, r_i, b_i). If we allocate to this aggregate of flows the sum of their sustainable rates $\sum_i r_i$, then the buffer requirement is the sum of the burst tolerances $\sum_i b_i$, regardless of other parameters such as peak rate. Conversely, Equation 1.35 also illustrates that there is no point allocating more buffer than the burst tolerance: if $B > b$, then the equivalent capacity is still r.

The above has illustrated that it is possible to reduce the required buffer or delay by allocating a rate larger than the sustainable rate. In Section 2.2, we described how this may be done with a protocol such as RSVP.

Note that formulas (1.29) or (1.33), or both, can be used to estimate the capacity required for a flow, based on a measured arrival curve. We can view them as low-pass filters on the flow function R.

1.8.3 Example: Acceptance Region for a FIFO Multiplexer

Consider a node multiplexing n_1 flows of type 1 and n_2 flows of type 2, where every flow is defined by a T-SPEC (p_i, M_i, r_i, b_i). The node has a constant output rate C. We wonder how many flows the node can accept.

If the only condition for flow acceptance is that the delay for all flows is bounded by some value D, then the set of acceptable values of (n_1, n_2) is defined by

$$e_D(n_1\alpha_1 + n_2\alpha_2) \leq C$$

We can use the same convexity arguments as for the derivation of formula (1.31), applied to the function $n_1\alpha_1 + n_2\alpha_2$. Define $\theta_i = \frac{b_i - M}{p_i - r_i}$ and assume $\theta_1 \leq \theta_2$. The result is:

$$e_D(n_1\alpha_1 + n_2\alpha_2) = \max \begin{cases} \frac{n_1M_1+n_2M_2}{D}, \\ \frac{n_1M_1+n_2M_2+(n_1p_1+n_2p_2)\theta_1}{\theta_1+D}, \\ \frac{n_1b_1+n_2M_2+(n_1r_1+n_2p_2)\theta_2}{\theta_2+D}, \\ n_1r_1+n_2r_2 \end{cases}$$

The set of feasible (n_1, n_2) derives directly from the previous equation; it is the convex part shown in Figure 1.27. The alert reader will enjoy performing the computation of the equivalent capacity for the case where the acceptance condition bears on a buffer size B.

i	p_i	M_i	r_i	b_i	θ_i
1	20'000 packets/s	1 packet	500 packets/s	26 packets	1.3 ms
2	5'000 packets/s	1 packet	500 packets/s	251 packets	55.5 ms

Figure 1.27: Acceptance region for a mix of type 1 and type 2 flows. Maximum delay $D = xx$. The parameters for types 1 and 2 are shown in the table, together with the resulting values of θ_i.

Coming back to equation 1.32, we can state in more general terms that the effective bandwidth is a convex function of function α, namely:

$$e_D(a\alpha_1 + (1-a)\alpha_2) \leq ae_D(\alpha_1) + (1-a)e_D(\alpha_2)$$

for all $a \in [0, 1]$. The same is true for the equivalent capacity function.

Consider now a call acceptance criterion based solely on a delay bound, or based on a maximum buffer constraint, or both. Consider further that there are I types of connections, and define the acceptance region $\mathcal{A}$ as the set of values $(n_1, \ldots, n_I)$ that satisfy the call acceptance criterion, where n_i is the number of connections of class i. From the convexity of the effective bandwidth and equivalent capacity functions, it follows that the acceptance region $\mathcal{A}$ is *convex*. In chapter 9 we compare this to acceptance regions for systems with some positive loss probability.

Sustainable Rate Allocation If we are interested only in course results, then we can reconsider the previous solution and take into account only the sustainable rate of the connection mix. The aggregate flow is constrained (among others) by $\alpha(s) = b + rs$, with $b = \sum_i n_i b_i$ and $r = \sum_i n_i r_i$. Theorem 1.4.1 shows that the maximum aggregate buffer occupancy is bounded by b as long as $C \geq r$. In other words, allocating the sustainable rate guarantees a loss-free operation, as long as the total buffer is equal to the burstiness.

In a more general setting, assume an aggregate flow has α as minimum arrival curve, and assume that some parameters r and b are such that

$$\lim_{s \to +\infty} \alpha(s) - rs - b = 0$$

so that the sustainable rate r with burstiness b is a tight bound. It can easily be shown that if we allocate a rate $C = r$, then the maximum buffer occupancy is b.

Consider now multiplexing a number of VBR connections. If no buffer is available, then it is necessary for a loss-free operation to allocate the sum of the peak rates. In contrast, using a buffer of size b makes it possible to allocate only the sustainable rate. This is what we call the *buffering gain*, namely, the gain on the peak rate obtained by adding some buffer. The buffering gain comes at the expense of increased delay, as can easily be seen from Theorem 1.4.2.

1.9 Proof of Theorem 1.4.5

Step 1: Consider a fixed time t_0 and assume, in this step, that there is some time u_0 that achieves the supremum in the definition of $\alpha \oslash \beta$. We construct some input and output functions R and R^* such that R is constrained by α, the system (R, R^*) is causal, and $\alpha^*(t_0) = (R^* \oslash R^*)(t_0)$. R and R^* are given by (Figure 1.28)

$$\begin{cases} R(t) = \alpha(t) \text{ if } t < u_0 + t_0 \\ R(t) = \alpha(u_0 + t_0) \text{ if } t \geq u_0 + t_0 \\ R^*(t) = \inf[\alpha(t), \beta(t)] \text{ if } t < u_0 + t_0 \\ R^*(t) = R(t) \text{ if } t \geq u_0 + t_0 \end{cases}$$

It is easy to see, as in the proof of Theorem 1.4.4 that R and R^* are wide-sense increasing, that $R^* \leq R$ and that β is a service curve for the flow. Now

$$R^*(u_0 + t_0) - R^*(u_0) = \alpha(u_0 + t_0) - R^*(u_0) \geq \alpha(u_0 + t_0) - \beta(u_0) = \alpha^*(t_0)$$

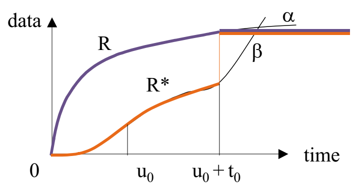

Figure 1.28: Step 1 of the proof of Theorem 1.4.5: a system that attains the output bound at one value t_0.

Step 2: Consider now a sequence of times $t_0, t_1, ..., t_n, ...$ (not necessarily increasing). Assume, in this step, that for all n there is a value u_n that achieves the supremum in the definition of $(\alpha \oslash \beta)(t_n)$. We prove that there are some functions R and R^* such that R is constrained by α, the system (R, R^*) is causal, has β as a service curve, and $\alpha^*(t_n) = (R^* \oslash R^*)(t_n)$ for all $n \geq 0$.

We build R and R^* by induction on a set of increasing intervals $[0, s_0]$, $[0, s_1]$,..., $[0, s_n]$.... The induction property is that the system restricted to time interval $[0, s_n]$ is causal, has α as an arrival curve for the input, has β as a service curve, and satisfies $\alpha^*(t_i) = (R^* \oslash R^*)(t_i)$ for $i \leq n$.

The first interval is defined by $s_0 = u_0 + t_0$; R and R^* are built on $[0, s_0]$ as in step 1 above. Clearly, the induction property is true for $n = 0$. Assume we have built the system on interval $[0, s_n]$. Define now $s_{n+1} = s_n + u_n + t_n + \delta_{n+1}$. We chose δ_{n+1} such that

$$\alpha(s + \delta_{n+1}) - \alpha(s) \geq R(s_n) \text{ for all } s \geq 0 \tag{1.36}$$

This is possible from the last condition in the Theorem. The system is defined on $]s_n, s_{n+1}]$ by (Figure 1.29)

$$\begin{cases} R(t) = R^*(t) = R(s_n) \text{ for } s_n < t \leq s_n + \delta_{n+1} \\ R(t) = R(s_n) + \alpha(t - s_n - \delta_{n+1}) \text{ for } s_n + \delta_{n+1} < t \leq s_{n+1} \\ R^*(t) = R(s_n) + (\alpha \wedge \beta)(t - s_n - \delta_{n+1}) \text{ for } s_n + \delta_{n+1} < t < s_{n+1} \\ R^*(s_{n+1}) = R(s_{n+1}) \end{cases}$$

We show now that the arrival curve constraint is satisfied for the system defined on $[0, s_{n+1}]$. Consider $R(t) - R(v)$ for t and v in $[0, s_{n+1}]$. If both $t \leq s_n$ and $v \leq s_n$, or if both $t > s_n$ and $v > s_n$ then the arrival curve property holds from our construction and the induction property. We can thus assume that $t > s_n$ and $v \leq s_n$. Clearly, we can even assume that $t \geq s_n + \delta_{n+1}$, otherwise the property is trivially true. Let us rewrite $t = s_n + \delta_{n+1} + s$. We have, from our construction:

$$R(t) - R(v) = R(s_n + \delta_{n+1} + s) - R(v) = R(s_n) + \alpha(s) - R(v) \leq R(s_n) + \alpha(s)$$

Now from Equation (1.36), we have:

$$R(s_n) + \alpha(s) \leq \alpha(s + \delta_{n+1}) \leq \alpha(s + \delta_{n+1} + s_n - v) = \alpha(t - v)$$

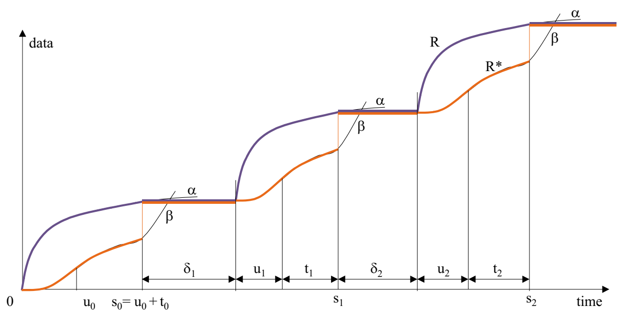

Figure 1.29: Step 2 of the proof of Theorem 1.4.5: a system that attains the output bound for all values t_n, $n \in \mathbb{N}$.

which shows the arrival curve property.

Using the same arguments as in step 1, it is simple to show that the system is causal, has β as a service curve, and that

$$R^*(u_{n+1} + t_{n+1}) - R^*(u_{n+1}) = \alpha^*(t_{n+1})$$

which ends the proof that the induction property is also true for $n + 1$.

Step 3: Consider, as in step 2, a sequence of times $t_0, t_1, ..., t_n, ...$ (not necessarily increasing). We now extend the result in step 2 to the case where the supremum in the definition of $\alpha^* = (\alpha \oslash \beta)(t_n)$ is not necessarily attained. Assume first that $\alpha^*(t_n)$ is finite for all n. For all n and all $m \in \mathbb{N}^*$ there is some $u_{m,n}$ such that

$$\alpha(t_n + u_{m,n}) - \beta(u_{m,n}) \geq \alpha^*(t_n) - \frac{1}{m} \tag{1.37}$$

Now the set of all couples (m, n) is enumerable. Consider some numbering $(M(i), N(i))$, $i \in \mathbb{N}$ for that set. Using the same construction as in step 2, we can build by induction on i a sequence of increasing intervals $[0, s_i]$ and a system (R, R^*) that is causal, has α as an arrival curve for the input, has β as a service curve, and such that

$$R^*(s_i) - R^*(s_i - t_{N(i)}) \geq \alpha^*(t_{N(i)}) - \frac{1}{M(i)}$$

Now consider an arbitrary, but fixed n. By applying the previous equations to all i such that $N(i) = n$, we obtain

$$\begin{aligned} (R^* \oslash R^*)(t_n) \geq & \quad \sup_{i \text{ such that } N(i)=n} \left\{ \alpha^*(t_{N(i)}) - \tfrac{1}{M(i)} \right\} \\ = & \quad \alpha^*(t_n) - \inf_{i \text{ such that } N(i)=n} \left\{ \tfrac{1}{M(i)} \right\} \end{aligned}$$

Now the set of all $\frac{1}{M(i)}$ for i such that $N(i) = n$ is $\mathbb{N}^*$, thus

$$\inf_{i \text{ such that } N(i)=n} \left\{ \frac{1}{M(i)} \right\} = 0$$

and thus $(R^* \oslash R^*)(t_n) = \alpha^*(t_n)$, which ends the proof of step 3 in the case where $\alpha^*(t_n)$ is finite for all n.

A similar reasoning can be used if $\alpha^*(t_n)$ is infinite for some t_n. In that case replace Equation (1.37) by $\alpha(t_n + u_{m,n}) - \beta(u_{m,n}) \geq m$.

Step 4: Now we conclude the proof. If time is discrete, then step 3 proves the theorem. Otherwise we use a density argument. The set of nonnegative rational numbers $\mathbb{Q}^+$ is enumerable; we can thus apply step 3 to the sequence of all elements of $\mathbb{Q}^+$, and obtain system (R, R^*), with

$$(R^* \oslash R^*)(q) = \alpha^*(q) \text{ for all } q \in \mathbb{Q}^+$$

Function R^* is right-continuous, thus, from the discussion at the end of Theorem 1.2.2, it follows that $R^* \oslash R^*$ is left-continuous. We now show that α^* is also left-continuous. For all $t \geq 0$ we have:

$$\sup_{s<t} \alpha^*(s) = \sup_{(s,v) \text{ such that } s<t \text{ and } v\geq 0} \{\alpha(s+v) - \beta(v)\} = \sup_{v\geq 0}\{\sup_{s<t}[\alpha(s+v) - \beta(v)]\}$$

Now

$$\sup_{s<t} \alpha(s+v) = \alpha(t+v)$$

because α is left-continuous. Thus

$$\sup_{s<t} \alpha^*(s) = \sup_{v\geq 0}\{\alpha(t+v) - \beta(v)]\} = \alpha^*(t)$$

which shows that α is left-continuous.

Back to the main argument of step 4, consider some arbitrary $t \geq 0$. The set $\mathbb{Q}^+$ is dense in the set of nonnegative real numbers, thus there is a sequence of rational numbers $q_n \in \mathbb{Q}^+$, with $n \in \mathbb{N}$, such that $q_n \leq t$ and $\lim_{n\to+\infty} q_n = t$. From the left-continuity of $R^* \oslash R^*$ and α^* we have:

$$(R^* \oslash R^*)(t) = \lim_{n\to+\infty} (R^* \oslash R^*)(q_n) = \lim_{n\to+\infty} \alpha^*(q_n) = \alpha^*(t)$$

$\square$

1.10 Bibliographic Notes

Network calculus as has been applied to dimensioning ATM switches in [57]. A practical algorithm for the determination of the minimum arrival curve for ATM system is described in [58]. It uses the burstiness function of a flow, defined in [54] as follows. For any r, $B(r)$ is the minimum b such that the flow is $\gamma_{r,b}$-smooth, and is thus the required buffer if the flow is served at a constant rate r. Note that $B(r)$ is the Legendre transform of the minimum arrival curve σ of the flow, namely, $B(r) = \sup_{t\geq 0}(\sigma(t) - rt)$ [58] gives a fast algorithm for computing $B(r)$. Interestingly, the concept is applied also to the distribution of symbols in a text.

In [74], the concepts of arrival and service curve are used to analyze real time processing systems. It is shown that the service curve for a variable capacity node must be super-additive, and conversely, any super-additive function is a service curve for a variable capacity node. Compare to greedy shapers, which have a sub-additive service curve. This shows that, except for constant bit rate trunks, a greedy shaper cannot be modeled as a variable capacity node, and conversely.

In [9], the authors consider a crossbar switch, and call $r_{i,j}$ the rate assigned to the traffic from input port i to output port j. Assume that $\sum_i r_{i,j} \leq 1$ for all j and $\sum_j r_{i,j} \leq 1$ for all i. Using properties of doubly-stochastic matrices (such as $(r_{i,j})$ is), they give a simple scheduling algorithm that guarantees that the flow from port i to port j is allocated a variable capacity C satisfying $C_{i,j}(t) - C_{i,j}(s) \geq r_{i,j}(t-s) - s_{i,j}$ for some $s_{i,j}$ defined by the algorithm. Thus, the node offers a service curve equal to the rate-latency function $\beta_{r_{i,j},s_{i,j}}$.

A dual approach to account for variable length packets is introduced in [11]. It consists in replacing the definition of arrival curve (or σ-smoothness) by the concept of g-regularity. Consider a flow of variable length packets, with cumulative packet length L and call T_i the arrival epoch for the ith packet. The flow is said to be g-regular if $T(j) - T(i) \geq g(L(j) - L(i))$ for all packet numbers $i \leq j$. A theory is then developed with concepts similar to the greedy shaper. The theory uses max-plus convolution instead of min-plus convolution. The (b, r) regulator originally introduced by Cruz [19] is a shaper in this theory, whose output is g-regular, with $g(x) = \frac{(x-b)^+}{r}$. This theory does not exactly correspond to the usual concept of leaky bucket controllers. More specifically, there is not an exact correspondence between the set of flows that are g-regular on one hand, and that are σ-smooth on the other. We explain why with an example. Consider the set of flows that are g-regular, with $g(x) = \frac{x}{r}$. The minimum arrival curve we can put on this set of flows is $\sigma(t) = rt + l_{\max}$ [11]. But conversely, if a flow is σ-smooth, we cannot guarantee that it is g-regular. Indeed, the following sequence of packets is a flow that is σ-smooth but not g-regular: the flow has a short packet (length $l_1 < l_{\max}$) at time $T_1 = 0$, followed by a packet of maximum size $l_{\max}$ at time $T_2 = \frac{l_1}{r}$. In fact, if a flow is σ-smooth, then it is g'-regular, with $g'(x) = \frac{(x-l_{\max})^+}{r}$.

The strict service curve in Definition 1.3.2 is called "strong" service curve in [43].

1.11 Exercises

Exercise 1.1. *Compute the maximum buffer size X for a system that is initially empty, and where the input function is $R(t) = \int_0^t r(s)ds$, for the following cases.*

1. *if $r(t) = a$ (constant)*

2. *one on-off connection with peak rate 1 Mb/s, on period 1 sec, off period τ seconds, and trunk bit rate $c = 0.5$ Mb/s.*

3. *if $r(t) = c + c \sin \omega t$, with trunk bit rate $c > 0$.*

Exercise 1.2. *You have a fixed buffer of size X, that receives a data input $r(t)$. Determine the output rate c that is required to avoid buffer overflow given that the buffer is initially empty.*

Exercise 1.3. 1. *For a flow with constant bit rate c, give some possible arrival curves.*

2. *Consider a flow with an arrival curve given by: $\alpha(t) = B$, where B is constant. What does this mean for the flow ?*

Exercise 1.4. *We say that a flow is (P, B) constrained if it has $\gamma_{P,B}$ as an arrival curve.*

A trunk system has a buffer size of B and a trunk bitrate of P. Fill in the dots: (1) there is no loss if the input is $(., .)$ constrained (2) the output is $(., .)$ constrained.

2. *A (P, B) constrained flow is fed into an infinite buffer served at a rate of c. What is the maximum delay ?*

Exercise 1.5 (On-Off flows). 1. *Assume a data flow is periodical, with period T, and satisfies the following: $r(t) = p$ for $0 \leq t < T_0$, and $r(t) = 0$ for $T_0 \leq t < T$.*

   (a) *Draw $R(t) = \int_0^t r(s)ds$*
   (b) *Find an arrival curve for the flow. Find the minimum arrival curve for the flow.*
   (c) *Find the minimum (r, b) such that the flow is (r, b) constrained.*

2. *A traffic flow uses a link with bitrate P (bits/s). Data is sent as packets of variable length. The flow is controlled by a leaky bucket (r, b). What is the maximum packet size ? What is the minimum time interval between packets of maximum size ?*

   Application: P = 2 Mb/s, r = 0.2 Mb/s; what is the required burst tolerance b if the packet length is 2 Kbytes ? What is then the minimum spacing between packets ?

Exercise 1.6. *Consider the following alternative definition of the GCRA:*

Definition 1.11.1. *The GCRA (T, τ) is a controller that takes as input a cell arrival time* `t` *and returns* `result`*. It has internal (static) variables* `X` *(bucket level) and* `LCT` *(last conformance time).*

- *initially,* `X = 0` *and* `LCT = 0`

- *when a cell arrives at time* `t`, *then*

```
if (X - t + LCT > tau)
   result = NON-CONFORMANT;
else {
     X = max (X - t + LCT, 0) + T;
     LCT = t;
     result = CONFORMANT;
     }
```

Show that the two definitions of GCRA are equivalent.

Exercise 1.7. 1. *For the following flows and a GCRA(10, 2), give the conformant and non-conformant cells. Times are in cell slots at the link rate. Draw the leaky bucket behaviour assuming instantaneous cell arrivals.*

   (a) 0, 10, 18, 28, 38
   (b) 0, 10, 15, 25, 35
   (c) 0, 10, 18, 26, 36
   (d) 0, 10, 11, 18, 28

2. *What is the maximum number of cells that can flow back to back with GCRA(T, CDVT) (maximum "clump" size) ?*

Exercise 1.8. 1. *For the following flows and a GCRA(100, 500), give the conformant and non-conformant cells. Times are in cell slots at the link rate.*

   (a) 0, 100, 110, 12, 130, 140, 150, 160, 170, 180, 1000, 1010
   (b) 0, 100, 130, 160, 190, 220, 250, 280, 310, 1000, 1030
   (c) 0, 10, 20, 300, 310, 320, 600, 610, 620, 800, 810, 820, 1000, 1010, 1020, 1200, 1210, 1220, 1400, 1410, 1420, 1600, 1610, 1620

2. *Assume that a cell flow has a minimum spacing of γ time units between cell emission times (γ is the minimum time between the beginnings of two cell transmissions). What is the maximum burst size for GCRA(T, τ) ? What is the minimum time between bursts of maximum size ?*

3. *Assume that a cell flow has a minimum spacing between cells of γ time units, and a minimum spacing between bursts of T_I. What is the maximum burst size ?*

Exercise 1.9. *For a CBR connection, here are some values from an ATM operator:*

peak cell rate (cells/s)	100	1000	10000	100000
CDVT (microseconds)	2900	1200	400	135

1. *What are the (P, B) parameters in b/s and bits for each case ? How does T compare to τ ?*

2. *If a connection requires a peak cell rate of 1000 cells per second and a cell delay variation of 1400 microseconds, what can be done ?*

3. *Assume the operator allocates the peak rate to every connection at one buffer. What is the amount of buffer required to assure absence of loss ? Numerical Application for each of the following cases, where a number N of identical connections with peak cell rate P is multiplexed.*

case	1	2	3	4
nb of connnections	3000	300	30	3
peak cell rate (c/s)	100	1000	10000	100000

Exercise 1.10. *The two questions in this problem are independent.*

1. *An ATM source is constrained by GCRA($T = 30$ slots, $\tau = 60$ slots), where time is counted in slots. One slot is the time it takes to transmit one cell on the link. The source sends cells according to the following algorithm.*

   - *In a first phase, cells are sent at times $t(1) = 0$, $t(2) = 15$, $t(3) = 30, \ldots, t(n) = 15(n-1)$ as long as all cells are conformant. In other words, the number n is the largest integer such that all cells sent at times $t(i) = 15(i-1)$, $i \le n$ are conformant. The sending of cell n at time $t(n)$ ends the first phase.*
   - *Then the source enters the second phase. The subsequent cell $n+1$ is sent at the earliest time after $t(n)$ at which a conformant cell can be sent, and the same is repeated for ever. In other words, call $t(k)$ the sending time for cell k, with $k > n$; we have then: $t(k)$ is the earliest time after $t(k-1)$ at which a conformant cell can be sent.*

   How many cells were sent by the source in time interval $[0, 151]$?

2. *A network node can be modeled as a single buffer with a constant output rate c (in cells per second). It receives I ATM connections labeled $1, \ldots, I$. Each ATM connection has a peak cell rate p_i (in cells per second) and a cell delay variation tolerance τ_i (in seconds) for $1 \le i \le I$. The total input rate into the buffer is at least as large as $\sum_{i=1}^{I} p_i$ (which is equivalent to saying that it is unlimited). What is the buffer size (in cells) required for a loss-free operation ?*

Exercise 1.11. *In this problem, time is counted in slots. One slot is the duration to transmit one ATM cell on the link.*

1. *An ATM source S_1 is constrained by GCRA($T = 50$ slots, $\tau = 500$ slots), The source sends cells according to the following algorithm.*

- *In a first phase, cells are sent at times $t(1) = 0$, $t(2) = 10$, $t(3) = 20, \ldots, t(n) = 10(n-1)$ as long as all cells are conformant. In other words, the number n is the largest integer such that all cells sent at times $t(i) = 10(i-1)$, $i \leq n$ are conformant. The sending of cell n at time $t(n)$ ends the first phase.*
- *Then the source enters the second phase. The subsequent cell $n+1$ is sent at the earliest time after $t(n)$ at which a conformant cell can be sent, and the same is repeated for ever. In other words, call $t(k)$ the sending time for cell k, with $k > n$; we have then: $t(k)$ is the earliest time after $t(k-1)$ at which a conformant cell can be sent.*

How many cells were sent by the source in time interval $[0, 401]$?

2. *An ATM source S_2 is constrained by* both *GCRA($T = 10$ slots, $\tau = 2$ slots) and GCRA($T = 50$ slots, $\tau = 500$ slots). The source starts at time 0, and has an infinite supply of cells to send. The source sends its cells as soon as it is permitted by the combination of the GCRAs. We call $t(n)$ the time at which the source sends the nth cell, with $t(1) = 0$. What is the value of $t(15)$?*

Exercise 1.12. *Consider a flow $R(t)$ receiving a minimum service curve guarantee β. Assume that*

- *β is concave and wide-sense increasing*
- *the $\inf$ in $R \otimes \beta$ is a $\min$*

For all t, call $\tau(t)$ a number such that

$$(R \otimes \beta)(t) = R(\tau(t)) + \beta(t - \tau(t))$$

Show that it is possible to choose τ such that if $t_1 \leq t_2$ then $\tau(t_1) \leq \tau(t_2)$.

Exercise 1.13.
1. *Find the maximum backlog and maximum delay for an ATM CBR connection with peak rate P and cell delay variation τ, assuming the service curve is $c(t) = r(t - T_0)^+$*
2. *Find the maximum backlog and maximum delay for an ATM VBR connection with peak rate P, cell delay variation τ, sustainable cell rate M and burst tolerance τ_B (in seconds), assuming the service curve is $c(t) = r(t - T_0)^+$*

Exercise 1.14. *Show the following statements:*

1. *Consider a (P, B) constrained flow, served at a rate $c \geq P$. The output is also (P, B) constrained.*
2. *Assume $a()$ has a bounded right-handside derivative. Then the output for a flow constrained by $a()$, served in a buffer at a constant rate $c \geq \sup_{t \geq 0} a'(t)$, is also constrained by $a()$.*

Exercise 1.15. *1. Find the the arrival curve constraining the output for an ATM CBR connection with peak rate P and cell delay variation τ, assuming the service curve is $c(t) = r(t - T_0)^+$*

2. Find the arrival curve constraining the output for an ATM VBR connection with peak rate P, cell delay variation τ, sustainable cell rate M and burst tolerance τ_B (in seconds), assuming the service curve is $c(t) = r(t - T_0)^+$

Exercise 1.16. *Consider the figure "Derivation of arrival curve for the output of a flow served in a node with rate-latency service curve $\beta_{R,T}$". What can be said if t_0 in the Figure is infinite, namely, if $a'(t) > r$ for all t ?*

Exercise 1.17. *Consider a series of guaranteed service nodes with service curves $c_i(t) = r_i(t - T_i)^+$. What is the maximum delay through this system for a flow constrained by (m, b) ?*

Exercise 1.18. *A flow with T-SPEC (p, M, r, b) traverses nodes 1 and 2. Node i offers a service curve $c_i(t) = R_i(t - T_i)^+$. What buffer size is required for the flow at node 2 ?*

Exercise 1.19. *A flow with T-SPEC (p, M, r, b) traverses nodes 1 and 2. Node i offers a service curve $c_i(t) = R_i(t - T_i)^+$. A shaper is placed between nodes 1 and 2. The shaper forces the flow to the arrival curve $z(t) = \min(R_2 t, bt + m)$.*

1. *What buffer size is required for the flow at the shaper ?*
2. *What buffer size is required at node 2 ? What value do you find if $T_1 = T_2$?*
3. *Compare the sum of the preceding buffer sizes to the size that would be required if no re-shaping is performed.*
4. *Give an arrival curve for the output of node 2.*

Exercise 1.20. *Prove the formula giving of paragraph "Buffer Sizing at a Re-shaper"*

Exercise 1.21. *Is Theorem "Input-Output Characterization of Greedy Shapers" a stronger result than Corollary "Service Curve offered by a Greedy Shaper" ?*

Exercise 1.22. *1. Explain what is meant by "we pay bursts only once".*

2. Give a summary in at most 15 lines of the main properties of shapers

3. Define the following concepts by using the $\otimes$ operator: Service Curve, Arrival Curve, Shaper

4. What is a greedy source ?

Exercise 1.23. *1. Show that for a constant bit rate trunk with rate c, the backlog at time t is given by*

$$W(t) = \sup_{s \le t} \{R(t) - R^*(s) - c(t - s)\}$$

2. *What does the formula become if we assume only that, instead a constant bit rate trunk, the node is a scheduler offering β as a service curve ?*

Exercise 1.24. *Is it true that offering a service curve β implies that, during any busy period of length t, the amount of service received rate is at least $\beta(t)$?*

Exercise 1.25. *A flow $S(t)$ is constrained by an arrival curve α. The flow is fed into a shaper, with shaping curve σ. We assume that*

$$\alpha(s) = \min(m + ps, b + rs)$$

and

$$\sigma(s) = \min(Ps, B + Rs)$$

We assume that $p > r$, $m \leq b$ and $P \geq R$.

The shaper has a fixed buffer size equal to $X \geq m$. We require that the buffer never overflows.

1. *Assume that $B = +\infty$. Find the smallest of P which guarantees that there is no buffer overflow. Let P_0 be this value.*

2. *We do not assume that $B = +\infty$ any more, but we assume that P is set to the value P_0 computed in the previous question. Find the value (B_0, R_0) of (B, R) which guarantees that there is no buffer overflow and minimizes the cost function $c(B, R) = aB + R$, where a is a positive constant.*

   What is the maximum virtual delay if $(P, B, R) = (P_0, B_0, R_0)$?

Exercise 1.26. *We consider a buffer of size X cells, served at a constant rate of c cells per second. We put N identical connections into the buffer; each of the N connections is constrained both by GCRA(T_1, τ_1) and GCRA(T_2, τ_2). What is the maximum value of N which is possible if we want to guarantee that there is no cell loss at all ?*

Give the numerical application for $T_1 = 0.5$ ms, $\tau_1 = 4.5$ ms, $T_2 = 5$ ms, $\tau_2 = 495$ ms, $c = 10^6$ cells/second, $X = 10^4$ cells

Exercise 1.27. *We consider a flow defined by its function $R(t)$, with $R(t) =$ the number of bits observed since time $t = 0$.*

1. *The flow is fed into a buffer, served at a rate r. Call $q(t)$ the buffer content at time t. We do the same assumptions as in the lecture, namely, the buffer is large enough, and is initially empty. What is the expression of $q(t)$ assuming we know $R(t)$?*

   We assume now that, unlike what we saw in the lecture, the initial buffer content (at time $t = 0$) is not 0, *but some value $q_0 \geq 0$. What is now the expression for $q(t)$?*

2. *The flow is put into a leaky bucket policer, with rate r and bucket size b. This is a policer, not a shaper, so nonconformant bits are discarded. We assume that the bucket is large enough, and is initially empty. What is the condition on R which ensures that no bit is discarded by the policer (in other words, that the flow is conformant) ?*

   We assume now that, unlike what we saw in the lecture, the initial bucket *content (at time $t = 0$) is* not 0, *but some value $b_0 \geq 0$. What is now the condition on R which ensures that no bit is discarded by the policer (in other words, that the flow is conformant) ?*

Exercise 1.28. *Consider a variable capacity network node, with capacity curve $M(t)$. Show that there is one maximum function $S^*(t)$ such that for all $0 \leq s \leq t$, we have*

$$M(t) - M(s) \geq S^*(t-s)$$

Show that S^* is super-additive.

Conversely, if a function β is super-additive, show that there is a variable capacity network node, with capacity curve $M(t)$, such that for all $0 \leq s \leq t$, we have $M(t) - M(s) \geq S^*(t-s)$.

Show that, with a notable exception, a shaper cannot be modeled as a variable capacity node.

Exercise 1.29. 1. *Consider a packetized greedy shaper with shaping curve $\sigma(t) = rt$ for $t \geq 0$. Assume that $L(k) = kM$ where M is fixed. Assume that the input is given by $R(t) = 10M$ for $t > 0$ and $R(0) = 0$. Compute the sequence $R^(i)(t)$ used in the representation of the output of the packetized greedy shaper, for $i = 1, 2, 3, ...$.*

2. *Same question if $sigma(t) = (rt + 2M)1_{\{}t > 0\}$.*

Exercise 1.30. *Consider a source given by the function*

$$\begin{cases} R(t) = B \text{ for } t > 0 \\ R(t) = 0 \text{ for } t \leq 0 \end{cases}$$

Thus the flow consists of an instantaneous burst of B bits.

1. *What is the minimum arrival curve for the flow ?*

2. *Assume that the flow is served in one node that offers a minimum service curve of the rate latency type, with rate r and latency Δ. What is the maximum delay for the last bit of the flow ?*

3. *We assume now that the flow goes through a series of two nodes, $\mathcal{N}_1$ and $\mathcal{N}_2$, where $\mathcal{N}_i$ offers to the flow a minimum service curve of the rate latency type, with rate r_i and latency Δ_i, for $i = 1, 2$. What is the the maximum delay for the last bit of the flow through the series of two nodes ?*

4. *With the same assumption as in the previous item, call $R_1(t)$ the function describing the flow at the output of node $\mathcal{N}_1$ (thus at the input of node $\mathcal{N}_2$). What is the worst case minimum arrival curve for R_1 ?*

5. *We assume that we insert between $\mathcal{N}_1$ and $\mathcal{N}_2$ a "reformatter" $\mathcal{S}$. The input to $\mathcal{S}$ is $R_1(t)$. We call $R_1'(t)$ the output of $\mathcal{S}$. Thus $R_1'(t)$ is now the input to $\mathcal{N}_2$. The function of the "reformatter"$\mathcal{S}$ is to delay the flow R_1 in order to output a flow R_1' that is a delayed version of R. In other words, we must have $R_1'(t) = R(t-d)$ for some d. We assume that the reformatter $\mathcal{S}$ is optimal in the sense that it chooses the smallest possible d. In the worst case, what is this optimal value of d ?*

6. *With the same assumptions as in the previous item, what is the worst case end-to-end delay through the series of nodes $\mathcal{N}_1, \mathcal{S}, \mathcal{N}_2$? Is the reformatter transparent ?*

Exercise 1.31. *Let σ be a good function. Consider the concatenation of a bit-by-bit greedy shaper, with curve σ, and an L-packetizer. Assume that $\sigma(0^+) = 0$. Consider only inputs that are L-packetized*

1. *Is this system a packetized shaper for σ ?*

2. *Is it a packetized shaper for $\sigma + l_{\max}$?*

3. *Is it a packetized greedy shaper for $\sigma + l_{\max}$?*

Exercise 1.32. *Assume that σ is a good function and $\sigma = \sigma_0 + l u_0$ where u_0 is the step function with a step at $t = 0$. Can we conclude that σ_0 is sub-additive ?*

Exercise 1.33. *Is the operator (P^L) upper-semi-continuous ?*

Exercise 1.34.

1. *Consider the concatenation of an L-packetizer and a network element with minimum service curve β and maximum service curve γ. Can we say that the combined system offer a minimum service curve $(\beta(t) - l_{\max})^+$ and a maximum service curve γ, as in the case where the concatenation would be in the reverse order ? .*

2. *Consider the concatenation of a GPS node offering a guarantee λ_{r_1}, an L-packetizer, and a second GPS node offering a guarantee λ_{r_2}. Show that the combined system offers a rate-latency service curve with rate $R = \min(r_1, r_2)$ and latency$E = \frac{l_{\max}}{\max(r_1,r_2)}$.*

Exercise 1.35. *Consider a node that offers to a flow $R(t)$ a rate-latency service curve $\beta = S_{R,L}$. Assume that $R(t)$ is L-packetized, with packet arrival times called $T_1, T_2, ...$ (and is left-continuous, as usual)*

Show that $(R \otimes \beta)(t) = \min_{T_i \in [0,t]}[R(T_i) + \beta(t - T_i)]$ (and thus, the $\inf$ is attained).

Exercise 1.36. *1. Assume K connections, each with peak rate p, sustainable rate m and burst tolerance b, are offered to a trunk with constant service rate P and FIFO buffer of capacity X. Find the conditions on K for the system to be loss-free.*

2. *If $Km = P$, what is the condition on X for K connections to be accepted ?*

3. *What is the maximum number of connection if p = 2 Mb/s, m = 0.2 Mb/s, $X = 10$MBytes, $b = 1$Mbyte and P = 0.1, 1, 2 or 10 Mb/s ?*

4. *For a fixed buffer size X, draw the acceptance region when K and P are the variables.*

Exercise 1.37. *Show the formulas giving the expressions for $f_B(R)$ and $f_B(\alpha)$.*

Exercise 1.38. *1. What is the effective bandwith for a connection with p = 2 Mb/s, m = 0.2 Mb/s, b = 100 Kbytes when D = 1msec, 10 msec, 100 msec, 1s ?*

2. *Plot the effective bandwidth e as a function of the delay constraint in the general case of a connection with parameters p, m, b.*

Exercise 1.39. *1. Compute the effective bandwidth for a mix of VBR connections $1, \ldots, I$.*

2. *Show how the homogeneous case can be derived from your formula*

3. *Assume K connections, each with peak rate p, sustainable rate m and burst tolerance b, are offered to a trunk with constant service rate P and FIFO buffer of capacity X. Find the conditions on K for the system to be loss-free.*

4. *Assume that there are two classes of connections, with K_i connections in class i, $i = 1, 2$, offered to a trunk with constant service rate P and FIFO buffer of infinite capacity X. The connections are accepted as long as their queuing delay does not exceed some value D. Draw the acceptance region, that is, the set of (K_1, K_2) that are accepted by CAC2. Is the acceptance region convex ? Is the complementary of the acceptance region in the positive orthant convex ? Does this generalize to more than two classes ?*

Chapter 2

Application of Network Calculus to the Internet

In this chapter we apply the concepts of Chapter 1 and explain the theoretical underpinnings of integrated and differentiated services. Integrated services define how reservations can be made for flows. We explain in detail how this framework was deeply influenced by GPS. In particular, we will see that it assumes that every router can be modeled as a node offering a minimum service curve that is a rate-latency function. We explain how this is used in a protocol such as RSVP. We also analyze the more efficient framework based on service curve scheduling. This allows us to address in a simple way the complex issue of schedulability.

We explain the concept of Guaranteed Rate node, which corresponds to a service curve element, but with some differences, because it uses a max-plus approach instead of min-plus. We analyze the relation between the two approaches.

Differentiated services differ radically, in that reservations are made per class of service, rather than per flow. We show how the bounding results in Chapter 1 can be applied to find delay and backlog bounds. We also introduce the “damper”, which is a way of enforcing a maximum service curve, and show how it can radically reduce the delay bounds.

2.1 GPS and Guaranteed Rate Nodes

In this section we describe GPS and its derivatives; they form the basis on which the Internet guaranteed model was defined.

2.1.1 Packet Scheduling

A guaranteed service network offers delay and throughput guarantees to flows, provided that the flows satisfy some arrival curve constraints (Section 2.2). This requires that network nodes implement some form of packet scheduling, also called service discipline. Packet scheduling is defined as the function that decides, at every buffer inside a network node, the service order for different packets.

A simple form of packet scheduling is FIFO: packets are served in the order of arrival. The delay bound, and the required buffer, depend on the minimum arrival curve of the aggregate flow (Section 1.8 on page 64). If one flow sends a large amount of traffic, then the delay increases for all flows, and packet loss may occur. Thus FIFO scheduling requires that arrival curve constraints on all flows be strictly enforced at all points in the network. Also, with FIFO scheduling, the delay bound is the same for all flows. We study FIFO scheduling in more detail in Section 6.

An alternative [23, 41] is to use per flow queuing, in order to (1) provide isolation to flows and (2) offer different guarantees. We consider first the ideal form of per flow queuing called “Generalized Processor Sharing” (GPS) [60], which was already mentioned in Chapter 1.

2.1.2 GPS and a Practical Implementation (PGPS)

A GPS node serves several flows in parallel, and has a total output rate equal to c b/s. A flow i is allocated a given weight, say ϕ_i. Call $R_i(t), R_i^*(t)$ the input and output functions for flow i. The guarantee is that at any time t, the service rate offered to flow i is 0 is flow i has no backlog (namely, if $R_i(t) = R_i^*(t)$), and otherwise is equal to $\frac{\phi_i}{\sum_{j \in B(t)} \phi_j} c$, where $B(t)$ is the set of backlogged flows at time t. Thus

$$R_i^*(t) = \int_0^t \frac{\phi_i}{\sum_{j \in B(s)} \phi_j} 1_{\{i \in B(s)\}} ds$$

In the formula, we used the indicator function $1_{\{expr\}}$, which is equal to 1 if *expr* is true, and 0 otherwise.

It follows immediately that the GPS node offers to flow i a service curve equal to $\lambda_{r_i c}$, with $r_i = \frac{\phi_i C}{\sum_j \phi_j}$. It is shown in [61] that a better service curve can be obtained for every flow if we know some arrival curve properties for all flows; however the simple property is sufficient to understand the integrated service model.

GPS satisfies the requirement of isolating flows and providing differentiated guarantees. We can compute the delay bound and buffer requirements for every flow if we know its arrival curve, using the results of Chapter 1. However, a GPS node is a theoretical concept, which is not really implementable, because it relies on a fluid model, and assumes that packets are infinitely divisible. How can we make a practical implementation of GPS ? One simple solution would be to use the virtual finish times as we did for the buffered leaky bucket controller in Section 1.7.3: for every packet we would compute its finish time θ under GPS, then at time θ present the packet to a multiplexer that serves packets at a rate c. Figure 2.1 (left) shows the finish times on an example. It also illustrates the main drawback that this method would have: at times 3 and 5, the multiplexer would be idle, whereas at time 6 it would have a burst of 5 packets to serve. In particular, such a scheduler would not be work conserving.

This is what motivated researchers to find other practical implementations of GPS. We study here one such implementation of GPS, called packet by packet generalized processor sharing (PGPS) [60]. Other implementations of GPS are discussed in Section 2.1.3.

PGPS emulates GPS as follows. There is one FIFO queue per flow. The scheduler handles packets one at a time, until it is fully transmitted, at the system rate c. For every packet, we compute the finish time that it would have under GPS (we call this the “GPS-finish-time”). Then, whenever a packet is finished transmitting, the next packet selected for transmission is the one with the earliest GPS-finish-time, among all packets present. Figure 2.1 shows one example. We see that, unlike the simple solution discussed earlier, PGPS is work conserving, but does so at the expense of maybe scheduling a packet *before* its finish time under GPS.

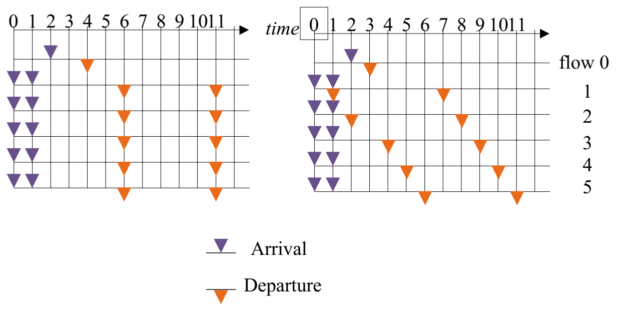

Figure 2.1: Scheduling with GPS (left) and PGPS (right). Flow 0 has weight 0.5, flows 1 to 5 have weight 0.1. All packets have the same transmission time equal to 1 time unit.

We can quantify the difference between PGPS and GPS in the following proposition. In Section 2.1.3, we will see how to derive a service curve property.

Proposition 2.1.1 ([60]). *The finish time for PGPS is at most the finish time of GPS plus* $\frac{L}{c}$*, where* c *is the total rate and* L *is the maximum packet size.*

Proof: Call $D(n)$ the finish time of the nth packet for the aggregate input flow under PGPS, in the order of departure, and $\theta(n)$ under GPS. Call n_0 the number of the packet that started the busy period in which packet n departs. Note that PGPS and GPS have the same busy periods, since if we observe only the aggregate flows, there is no difference between PGPS and GPS.

There may be some packets that depart before packet n in PGPS, but that nonetheless have a later departure time under GPS. Call $m_0 \geq n_0$ the largest packet number for which this occurs, if any; otherwise let $m_0 = n_0 - 1$. In this proposition, we call $l(m)$ the length in bits of packet m. Under PGPS, packet m_0 started service at $D(m_0) - \frac{l(m_0)}{c}$, which must be earlier than the arrival times of packets $m = m_0 + 1, ..., n$. Indeed, otherwise, by definition of PGPS, the PGPS scheduler would have scheduled packets $m = m_0 + 1, ..., n$ before packet m_0. Now let us observe the GPS system. Packets $m = m_0 + 1, ..., n$ depart no later than packet n, by definition of m_0; they have arrived after $D(m_0) - \frac{l(m_0)}{c}$. By expressing the amount of service in the interval $[D(m_0) - \frac{l(m_0)}{c}, \theta(n)]$ we find thus

$$\sum_{m=m_0+1}^{n} l(m) \leq c \left(\theta(n) - D(m_0) + \frac{l(m_0)}{c} \right)$$

Now since packets $m_0, ..., n$ are in the same busy period, we have

$$D(n) = D(m_0) + \frac{\sum_{m=m_0+1}^{n} l(m)}{c}$$

By combining the two equations above we find $D(n) \leq \theta(n) + \frac{l(m_0)}{c}$, which shows the proposition in the case where $m_0 \leq n_0$.

If $m_0 = n_0 - 1$, then all packets $n_0, ..., n$ depart before packet n under GPS and thus the same reasoning shows that

$$\sum_{m=n_0}^{n} l(m) \leq c\left(\theta(n) - t_0\right)$$

where t_0 is the beginning of the busy period, and that

$$D(n) = t_0 + \frac{\sum_{m=n_0}^{n} l(m)}{c}$$

Thus $D(n) \leq \theta(n)$ in that case. □

2.1.3 Guaranteed Rate (GR) Nodes and the Max-Plus Approach

The service curve concept defined earlier can be approached from the dual point of view, which consists in studying the packet arrival and departure times instead of the functions $R(t)$ (which count the bits arrived up to time t). This latter approach leads to max-plus algebra (which has the same properties as min-plus), is often more appropriate to account for details due to variable packet sizes, but works well only when the service curves are of the rate-latency type. It also useful when nodes cannot be assumed to be FIFO per flow, as may be the case with DiffServ (Section 2.4).

GR also allows to show that many schedulers have the rate-latency service curve property. Indeed, a large number of practical implementations of GPS, other than PGSP, have been proposed in the literature; let us mention: virtual clock scheduling [45], packet by packet generalized processor sharing [60] and self-clocked fair queuing [37](see also [28]). For a thorough discussion of practical implementations of GPS, see [77, 28]). These implementations differ in their implementation complexity and in the bounds that can be obtained. It is shown in [29] that all of these implementations fit in the following framework, called “Guaranteed Rate”, which we define in now. We will also analyze how it relates to the min-plus approach.

Definition 2.1.1 (GR Node[29]). *Consider a node that serves a flow. Packets are numbered in order of arrival. Let $a_n \geq 0, d_n \geq 0$ be the arrival and departure times. We say that a node is the a* guaranteed rate *(GR) node for this flow, with rate r and delay e, if it guarantees that $d_n \leq f_n + e$, where f_n is defined by Equation (2.1).*

$$\begin{cases} f_0 = 0 \\ f_n = \max\{a_n, f_{n-1}\} + \frac{l_n}{r} \quad \text{for all } n \geq 1 \end{cases} \tag{2.1}$$

The variables f_n (“Guaranteed Rate Clocks”) can be interpreted as the departures times from a FIFO constant rate server, with rate r. The parameter e expresses how much the node deviates from it. Note however that *a GR node need not be FIFO*. A GR node is also called “Rate-Latency server”.

Theorem 2.1.1 (Max-Plus Representation of GR). *Consider a system where packets are numbered $1, 2, ...$ in order of arrival. Call a_n, d_n the arrival and departure times for packet n, and l_n the size of packet n. Define by convention $d_0 = 0$. The system is a GR node with rate r and latency e if and only if for all n there is some $k \in \{1, ..., n\}$ such that*

$$d_n \leq e + a_k + \frac{l_k + ... + l_n}{r} \tag{2.2}$$

Proof: The recursion Equation (2.1) can be solved iteratively, using the same max-plus method as in the proof of Proposition 1.2.4. Define

$$A_j^n = a_j + \frac{l_j + ... + l_n}{r} \text{ for } 1 \leq j \leq n$$

Then we obtain

$$f_n = \max(A_n^n, A_{n-1}^n, ..., A_1^n)$$

The rest follows immediately. □

Equation (2.2) is the dual of the service curve definition (Equation (1.9) on Page 87), with $\beta(t) = r(t-e)^+$. We now elucidate this relationship.

Theorem 2.1.2 (Equivalence with service curve). *Consider a node with L-packetized input.*

1. *If the node guarantees a minimum service curve equal to the rate-latency function $\beta = \beta_{r,v}$, and if it is FIFO, then it is a GR node with rate r and latency v.*

2. *Conversely, a GR node with rate r and latency e is the concatenation of a service curve element, with service curve equal to the rate-latency function $\beta_{r,v}$, and an L-packetizer. If the GR node is FIFO, then so is the service curve element.*

The proof is long and is given at the end of this section.

By applying Theorem 1.7.1, we obtain

Corollary 2.1.1. *A GR node offers a minimum service curve* $\beta_{r,v+\frac{l_{\max}}{r}}$

The service curve can be used to obtain backlog bounds.

Theorem 2.1.3 (Delay Bound). *For an α-smooth flow served in a (possibly non FIFO) GR node with rate r and latency e, the delay for any packet is bounded by*

$$\sup_{t>0}[\frac{\alpha(t)}{r} - t] + e \tag{2.3}$$

Proof: By Theorem 2.1.1, for any fixed n, we can find a $1 \leq j \leq n$ such that

$$f_n = a_j + \frac{l_j + ... + l_n}{r}$$

The delay for packet n is

$$d_n - a_n \leq f_n + e - a_n$$

Define $t = a_n - a_j$. By hypothesis

$$l_j + ... + l_n \leq \alpha(t+)$$

where $\alpha(t+)$ is the limit to the right of α at t. Thus

$$d_n - a_n \leq -t + \frac{\alpha(t+)}{r} + e \leq \sup_{t\geq 0}[\frac{\alpha(t+)}{r} - t] + e$$

Now $\sup_{t>0}[\frac{\alpha(t)}{r} - t] = \sup_{t\geq 0}[\frac{\alpha(t+)}{r} - t]$. □

Comment: Note that Equation (2.3) is the horizontal deviation between the arrival curve α and the rate-latency service curve with rate r and latency e. Thus, for FIFO GR nodes, Theorem 2.1.3 follows from Theorem 2.1.1 and the fact that the packetizer can be ignored for delay computations. The information in Theorem 2.1.3 is that it also holds for non-FIFO nodes.

Concatenation of GR nodes For GR nodes that are FIFO per flow, the concatenation result obtained with the service curve approach applies. Specifically, the concatenation of M GR nodes (that are FIFO per flow) with rates r_m and latencies e_m is GR with rate $r = \min_m r_m$ and latency $e = \sum_m e_m + (M-1)\frac{L_{\max}}{r}$, where $L_{\max}$ is the maximum packet size for the flow. The term $(M-1)\frac{L_{\max}}{r}$ is due to packetizers.

A bound on the end-to-end delay through such a concatenation is thus

$$D = \sum_{m=1}^{M} v_m + l_{\max} \sum_{m=1}^{M-1} \frac{1}{r_m} + \frac{\sigma}{\min_m r_m} \tag{2.4}$$

which is the formula in [29]. It is a generalization of Equation (1.23) on Page 55.

For GR nodes that are not FIFO per flow, the concatenation result is no longer true; see [48] for some partial results.

Proof of Theorem 2.1.2 Part 1: Consider a service curve element $\mathcal{S}$. Assume to simplify the demonstration that the input and output functions R and R^* are right-continuous. Consider the virtual system $\mathcal{S}^0$ made of a bit-by-bit greedy shaper with shaping curve λ_r, followed by a constant bit-by-bit delay element. The bit-by-bit greedy shaper is a constant bit rate server, with rate r. Thus the last bit of packet n departs from it exactly at time f_n, defined by Equation (2.1), thus the last bit of packet n leaves $\mathcal{S}^0$ at $d_n^0 = f_n + e$. The output function of $\mathcal{S}^0$ is $R^0 = R \otimes \beta_{r,e}$. By hypothesis, $R^* \geq R^0$, and by the FIFO assumption, this shows that the delay in $\mathcal{S}$ is upper bounded by the delay in $\mathcal{S}'$. Thus $d_n \leq f_n + e$.

Part 2: Consider the virtual system $\mathcal{S}$ whose output $S(t)$ is defined by

$$\begin{array}{c} \text{if } d_{i-1} < t \leq d_i \\ \text{then } S(t) = \min\{R(t), \max[L(i-1), L(i) - r(d_i - t)]\} \end{array} \tag{2.5}$$

See Figure 2.2 for an illustration. It follows immediately that $R'(t) = P^L(S(t))$.

Also consider the virtual system $\mathcal{S}^0$ whose output is

$$S^0(t) = (\beta_{r,v} \otimes R)(t)$$

$\mathcal{S}^0$ is the constant rate server, delayed by v. Our goal is now to show that $S \geq S^0$.

Call d_i^0 the departure time of the last bit of packet i in $\mathcal{S}_0$ (see Figure 2.2 for an example with $i = 2$). Let $u = d_i^0 - d_i$. The definition of GR node means that $u \geq 0$. Now since $\mathcal{S}_0$ is a shifted constant rate server, we have:

$$\text{if } d_i^0 - \frac{l_i}{r} < s < d_i^0 \text{ then } S^0(s) = L(i) - r(d_i^0 - s)$$

Also $d_{i-1}^0 \leq d_i^0 - \frac{l_i}{r}$ thus $S^0(d_i^0 - \frac{l_i}{r}) = L(i-1)$ and

$$\text{if } s \leq d_i^0 - \frac{l_i}{r} \text{ then } S^0(s) \leq L(i-1)$$

It follows that

$$\text{if } d_{i-1} + u < s < d_i^0 \text{ then } S^0(s) \leq \max[L(i-1), L(i) - r(d_i^0 - s)] \tag{2.6}$$

Consider now some $t \in (d_{i-1}, d_i]$ and let $s = t + u$. If $S(t) = R(t)$, since $R \geq S^0$, we then obviously have $S(t) \geq S^0(t)$. Else, from Equation (2.1), $S(t) = \max[L(i-1), L(i) - r(d_i - t)]$. We have $d_i^0 - s = d_i - t$ and thus, combining with Equation (2.6), we derive that $S^0(s) \leq S(t)$. Now $s \geq t$, thus finally $S^0(t) \leq S(t)$. One can also readily see that $\mathcal{S}$ is FIFO if $d_{i-1} \leq d_i$ for all i. □

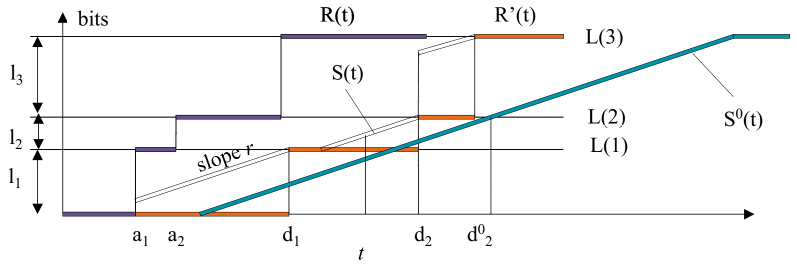

Figure 2.2: Arrival and departure functions for GR node. The virtual system output is $S(t)$.

2.2 The Integrated Services Model of the IETF

2.2.1 The Guaranteed Service

The Internet supports different reservation principles. Two services are defined: the “guaranteed” service, and the “ controlled load” service. They differ in that the former provides real guarantees, while the latter provides only approximate guarantees. We outline the differences in the rest of this section. In both cases, the principle is based on “admission control”, which operates as follows.

- In order to receive the guaranteed or controlled load service, a flow must first perform a reservation during a flow setup phase.
- A flow must confirm to an arrival curve of the form $\alpha(t) = \min(M+pt, rt+b)$, which is called the T-SPEC (see Section 1.2.2 on page16). The T-SPEC is declared during the reservation phase.
- All routers along the path accept or reject the reservation. With the guaranteed service, routers accept the reservation only if they are able to provide a service curve guarantee and enough buffer for loss-free operation. The service curve is expressed during the reservation phase, as explained below.

  For the controlled load service, there is no strict definition of what accepting a reservation means. Most likely, it means that the router has an estimation module that says that, with good probability, the reservation can be accepted and little loss will occur; there is no service curve or delay guarantee.

In the rest of this chapter we focus on the guaranteed service. Provision of the controlled load service relies on models with loss, which are discussed in Chapter 9.

2.2.2 The Integrated Services Model for Internet Routers

The reservation phase assumes that all routers can export their characteristics using a very simple model. The model is based on the view that an integrated services router implements a practical approximation of GPS, such as PGPS, or more generally, a GR node. We have shown in Section 2.1.3 that the service curve offered to a flow by a router implementing GR is a rate-latency function, with rate R and latency T connected by the relationship

$$T = \frac{C}{R} + D \tag{2.7}$$

with $C =$ the maximum packet size for the flow and $D = \frac{L}{c}$, where L is the maximum packet size in the router across all flows, and c the total rate of the scheduler. This is the model defined for an Internet node [71].

Fact 2.2.1. *The Integrated Services model for a router is that the service curve offered to a flow is always a rate-latency function, with parameters related by a relation of the form (2.7).*

The values of C and D depend on the specific implementation of a router, see Corollary 2.1.1 in the case of GR nodes. Note that a router does not necessarily implement a scheduling method that approximates GPS. In fact, we discuss in Section 2.3 a family of schedulers that has many advantages above GPS. If a router implements a method that largely differs from GPS, then we must find a service curve that lower-bounds the best service curve guarantee offered by the router. In some cases, this may mean loosing important information about the router. For example, it is *not* possible to implement a network offering constant delay to flows by means of a system like SCED+, discussed in Section 2.4.3, with the Integrated Services router model.

2.2.3 Reservation Setup with RSVP

Consider a flow defined by TSPEC (M, p, r, b), that traverses nodes $1, \ldots, N$. Usually, nodes 1 and N are end-systems while nodes n for $1 < n < N$ are routers. The Integrated Services model assumes that node n on the path of the flow offers a rate latency service curve β_{R_n,T_n}, and further assumes that T_n has the form

$$T_n = \frac{C_n}{R} + D_n$$

where C_n and D_n are constants that depend on the characteristics of node n.

The reservation is actually put in place by means of a flow setup procedure such as the resource reservation protocol (RSVP). At the end of the procedure, node n on the path has allocated to the flow a value $R_n \geq r$. This is equivalent to allocating a service curve β_{R_n,T_n}. From Theorem 1.4.6 on page 34, the end-to-end service curve offered to the flow is the rate-latency function with rate R and latency T given by

$$\left\{ \begin{array}{l} R = \min_{n=1\ldots N} R_n \\ T = \sum_{n=1}^{N} \left(\frac{C_n}{R_n} + D_n \right) \end{array} \right.$$

Let $C_{\text{tot}} = \sum_{n=1}^{N} C_n$ and $D_{\text{tot}} = \sum_{n=1}^{N} D_n$. We can re-write the last equation as

$$T = \frac{C_{\text{tot}}}{R} + D_{\text{tot}} - \sum_{n=1}^{N} S_n \tag{2.8}$$

with

$$S_n = C_n \left(\frac{1}{R} - \frac{1}{R_n} \right) \tag{2.9}$$

The term S_n is called the “local slack” term at node n.

From Proposition 1.4.1 we deduce immediately:

Proposition 2.2.1. *If $R \geq r$, the bound on the end-to-end delay, under the conditions described above is*

$$\frac{b-M}{R} \left(\frac{p-R}{p-r} \right)^{+} + \frac{M + C_{tot}}{R} + D_{tot} - \sum_{n=1}^{N} S_n \tag{2.10}$$

We can now describe the reservation setup with RSVP. Some details of flow setup with RSVP are illustrated on Figure 2.3. It shows that two RSVP flows are involved: an advertisement (PATH) flow and a reservation (RESV) flow. We describe first the point-to-point case.

- A PATH message is sent by the source; it contains the T-SPEC of the flow (source T-SPEC), which is not modified in transit, and another field, the AD-SPEC, which is accumulated along the path. At a destination, the ADSPEC field contains, among others, the values of $C_{\text{tot}}, D_{\text{tot}}$ used in Equation 2.10. PATH messages do not cause any reservation to be made.

- RESV messages are sent by the destination and cause the actual reservations to be made. They follow the reverse path marked by PATH messages. The RESV message contains a value, R', (as part of the so-called R-SPEC), which is a lower bound on the rate parameters R_n that routers along the path will have to reserve. The value of R' is determined by the destination based on the end-to-end delay objective d_{obj}, following the procedure described below. It is normally not changed by the intermediate nodes.

Define function f by

$$f(R') := \frac{b-M}{R'} \left(\frac{p-R'}{p-r} \right)^{+} + \frac{M + C_{tot}}{R'} + D_{tot}$$

In other words, f is the function that defines the end-to-end delay bound, assuming all nodes along the path would reserve $R_n = R'$. The destination computes R' as the smallest value $\geq r$ for which $f(R') \leq d_{\text{obj}}$. Such a value exists only if $D_{tot} < d_{\text{obj}}$.

In the figure, the destination requires a delay variation objective of 600 ms, which imposes a minimum value of R' =622 kb/s. The value of R' is sent to the next upstream node in the R-SPEC field of the PATH message. The intermediate

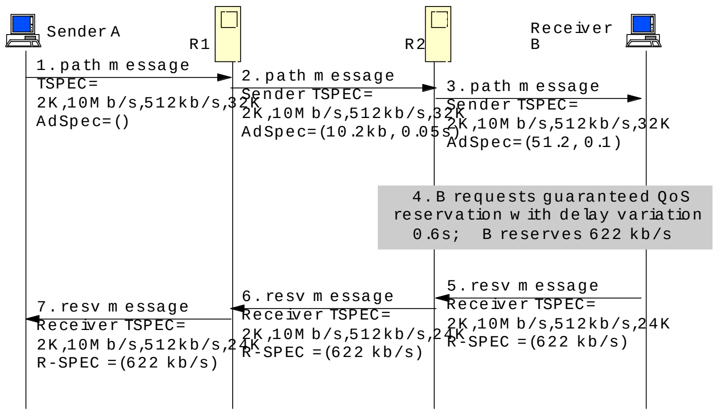

Figure 2.3: Setup of Reservations, showing the PATH and RESV flows

nodes do not know the complete values C_{tot} and D_{tot}, nor do they know the total delay variation objective. Consider the simple case where all intermediate nodes are true PGPS schedulers. Node n simply checks whether it is able to reserve $R_n = R'$ to the flow; this involves verifying that the sum of reserved rates is less than the scheduler total rate, and that there is enough buffer available (see below). If so, it passes the RESV message upstream, up to the destination if all intermediate nodes accept the reservation. If the reservation is rejected, then the node discards it and normally informs the source. In this simple case, all nodes should set their rate to $R_n = R'$ thus $R = R'$, and Equation (2.10) guarantees that the end-to-end delay bound is guaranteed.

In practice, there is a small additional element (use of the slack term), due to the fact that the designers of RSVP also wanted to support other schedulers. It works as follows.

There is another term in the R-SPEC, called the *slack* term. Its use is illustrated on Figure 2.4. In the figure, we see that the end-to-end delay variation requirement, set by the destination, is 1000 ms. In that case, the destination reserves the minimum rate, namely, 512 kb/s. Even so, the delay variation objective D_{obj} is larger than the bound D_{max} given by Formula (2.10). The difference $D_{obj} - D_{max}$ is written in the slack term S and passed to the upstream node in the RESV message. The upstream node is not able to compute Formula (2.10) because it does not have the value of the end-to-end parameters. However, it can use the slack term to increase its internal delay objective, on top of what it had advertised. For example, a guaranteed rate node may increase its value of v (Theorem 2.1.1) and thus reduce the internal resources required to perform the reservation. The figure shows that R1 reduces the slack term by 100 ms. This is equivalent to increasing the D_{tot} parameter by $100ms$, but without modifying the advertised D_{tot}.

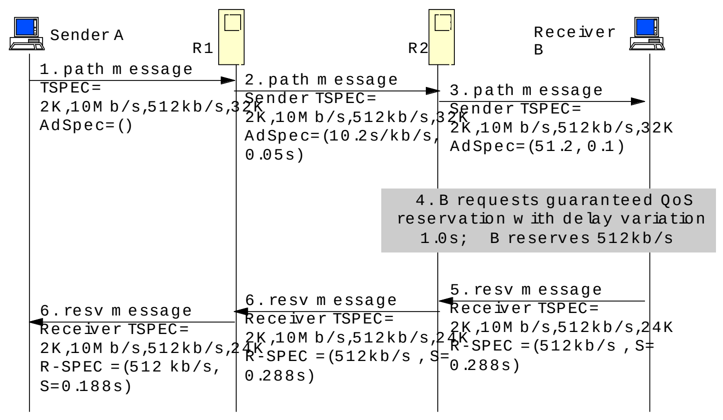
Sender A
R1
R2
Receiver B
1. path message TSPEC= 2K,10Mb/s,512kb/s,32K AdSpec=()
2. path message Sender TSPEC= 2K,10Mb/s,512kb/s,32K AdSpec=(10.2s/kb/s, 0.05s)
3. path message Sender TSPEC= 2K,10Mb/s,512kb/s,32K AdSpec=(51.2, 0.1)
4. B requests guaranteed QoS reservation with delay variation 1.0s; B reserves 512kb/s
5. resv message Receiver TSPEC= 2K,10Mb/s,512kb/s,24K R-SPEC =(512kb/s , S= 0.288s)
6. resv message Receiver TSPEC= 2K,10Mb/s,512kb/s,24K R-SPEC =(512kb/s , S= 0.288s)
6. resv message Receiver TSPEC= 2K,10Mb/s,512kb/s,24K R-SPEC =(512 kb/s, S=0.188s)

Figure 2.4: Use of the slack term

The delays considered here are the total (fixed plus variable) delays. RSVP also contains a field used for advertising the fixed delay part, which can be used to compute the end-to-end fixed delay. The variable part of the delay (called delay jitter) is then obtained by subtraction.

2.2.4 A Flow Setup Algorithm

There are many different ways for nodes to decide which parameter they should allocate. We present here one possible algorithm. A destination computes the worst case delay variation, obtained if all nodes reserve the sustainable rate r. If the resulting delay variation is acceptable, then the destination sets $R = r$ and the resulting slack may be used by intermediate nodes to add a local delay on top of their advertised delay variation defined by C and D. Otherwise, the destination sets R to the minimum value R_{min} that supports the end-to-end delay variation objective and sets the slack to 0. As a result, all nodes along the path have to reserve R_{min}. As in the previous cases, nodes may allocate a rate larger than the value of R they pass upstream, as a means to reduce their buffer requirement.

Definition 2.2.1 (A Flow Setup Algorithm).

- *At a destination system I, compute*

$$D_{max} = f_T(r) + \frac{C_{tot}}{r} + D_{tot}$$

If $D_{obj} > D_{max}$ then assign to the flow a rate $R_I = r$ and an additional delay variation $d_I \le D_{obj} - D_{max}$; set $S_I = D_{obj} - D_{max} - d_I$ and send reservation request R_I, S_I to station $I - 1$.

Else ($D_{obj} \le D_{max}$) find the minimum R_{min} such that $f_T(R_{min}) + \frac{C_{tot}}{R_{min}} \le D_{obj} - D_{tot}$, if it exists. Send reservation request $R_I = R_{min}, S_I = 0$ to *station $I-1$. If R_{min} does not exist, reject the reservation or increase the delay variation objective D_{obj}.*

- *At an intermediate system i: receive from $i+1$ a reservation request R_{i+1}, S_{i+1}.*

  If $S_i = 0$, then perform reservation for rate R_{i+1} and if successful, send reservation request $R_i = R_{i+1}, S_i = 0$ to station $i-1$.

  Else ($S_i > 0$), perform a reservation for rate R_{i+1} with some additional delay variation $d_i \leq S_{i+1}$. if successful, send reservation request $R_i = R_{i+1}, S_i = S_{i+1} - d_i$ to station $i-1$.

The algorithm ensures a constant reservation rate. It is easy to check that the end to end delay variation is bounded by D_{obj}.

2.2.5 Multicast Flows

Consider now a multicast situation. A source S sends to a number of destinations, along a multicast tree. PATH messages are forwarded along the tree, they are duplicated at splitting points; at the same points, RESV messages are merged. Consider such a point, call it node i, and assume it receives reservation requests for the same T-SPEC but with respective parameters R'_{in}, S'_{in} and R''_{in}, S''_{in}. The node performs reservations internally, using the semantics of algorithm 3. Then it has to merge the reservation requests it will send to node $i-1$. Merging uses the following rules:

R-SPEC Merging Rules The merged reservation R, S is given by

$$R = \max(R', R'')$$

$$S = \min(S', S'')$$

Let us consider now a tree where algorithm 3 is applied. We want to show that the end-to-end delay bounds at all destinations are respected.

The rate along the path from a destination to a source cannot decrease with this algorithm. Thus the minimum rate along the tree towards the destination is the rate set at the destination, which proves the result.

A few more features of RSVP are:

- states in nodes need to be refreshed; if they are not refreshed, the reservation is released (“soft states”).
- routing is not coordinated with the reservation of the flow

We have so far looked only at the delay constraints. Buffer requirements can be computed using the values in Proposition 1.4.1.

2.2.6 Flow Setup with ATM

With ATM, there are the following differences:

- The path is determined at the flow setup time only. Different connections may follow different routes depending on their requirements, and once setup, a connection always uses the same path.
- With standard ATM signaling, connection setup is initiated at the source and is confirmed by the destination and all intermediate systems.

2.3 Schedulability

So far, we have considered one flow in isolation and assumed that a node is able to offer some scheduling, or service curve guarantee. In this section we address the global problem of resource allocation.

When a node performs a reservation, it is necessary to check whether local resources are sufficient. In general, the method for this consists in breaking the node down into a network of building blocks such as schedulers, shapers, and delay elements. There are mainly two resources to account for: bit rate (called "bandwidth") and buffer. The main difficulty is the allocation of bit rate. Following [33], we will see in this section that allocating a rate amounts to allocating a service curve. It is also equivalent to the concept of schedulability.

Consider the simple case of a PGPS scheduler, with outgoing rate C. If we want to allocate rate r_i to flow i, for every i, then we can allocate to flow i the GPS weight $\phi_i = \frac{r_i}{C}$. Assume that

$$\sum_i r_i \leq C \tag{2.11}$$

Then we know from Proposition 2.1.1 and Corollary 2.1.1 that every flow i is guaranteed the rate-latency service curve with rate r_i and latency $\frac{L}{C}$. In other words, the schedulability condition for PGPS is simply Equation (2.11). However, we will see now that a schedulability conditions are not always as simple. Note also that the end-to-end delay depends not only on the service curve allocated to the flow, but also on its arrival curve constraints.

Many schedulers have been proposed, and some of them do not fit in the GR framework. The most general framework in the context of guaranteed service is given by SCED (Service Curve Earliest Deadline first) [33],which we describe now. We give the theory for constant size packets and slotted time; some aspects of the general theory for variable length packets are known [11], some others remain to be done. We assume without loss of generality that every packet is of size 1 data unit.

2.3.1 EDF Schedulers

As the name indicates, SCED is based on the concept of Earliest Deadline First (EDF) scheduler. An EDF scheduler assigns a deadline D_i^n to the nth packet of flow

i, according to some method. We assume that deadlines are wide-sense increasing within a flow. At every time slot, the scheduler picks at one of the packets with the smallest deadline among all packets present. There is a wide variety of methods for computing deadlines. The "delay based" schedulers [52] set $D_i^n = A^n + d_i$ where A^n is the arrival time for the nth packet for flow i, and d_i is the delay budget allocated to flow i. If d_i is independent of i, then we have a FIFO scheduler. We will see that those are special cases of SCED, which we view as a very general method for computing deadlines.

An EDF scheduler is work conserving, that is, it cannot be idle if there is at least one packet present in the system. A consequence of this is that packets from different flows are not necessarily served in the order of their deadlines. Consider for example a delay based scheduler, and assume that flow 1 has a lrage delay budget d_1, while flow 2 has a small delay budget d_2. It may be that a packet of flow 1 arriving at t_1 is served before a packet of flow 2 arriving at t_2, even though the deadline of packet 1, $t_1 + d_1$ is larger than the deadline of packet 2.

We will now derive a general schedulability criterion for EDF schedulers. Call $R_i(t)$, $t \in \mathbb{N}$, the arrival function for flow i. Call $Z_i(t)$ the number of packets of flow i that have deadlines $\leq t$. For example, for a delay based scheduler, $Z_i(t) = R_i(t - d_i)$. The following is a modified version of [11].

Proposition 2.3.1. *Consider an EDF scheduler with I flows and outgoing rate C. A necessary condition for all packets to be served within their deadlines is*

$$\text{for all } s \leq t: \sum_{i=1}^{I} Z_i(t) - R_i(s) \leq C(t - s) \tag{2.12}$$

A sufficient condition is

$$\text{for all } s \leq t: \sum_{i=1}^{I} [Z_i(t) - R_i(s)]^+ \leq C(t - s) \tag{2.13}$$

Proof: We first prove the necessary condition. Call R'_i the output for flow i. Since the scheduler is work conserving, we have $\sum_{i=1}^{I} R'_i = \lambda_C \otimes (\sum_{i=1}^{I} R_i)$. Now $R'_i \geq Z_i$ by hypothesis. Thus

$$\sum_{i=1}^{I} Z_i(t) \leq \inf_{s \in [0,t]} C(t - s) + \sum_{i=1}^{I} R_i(s)$$

which is equivalent to Equation (2.12)

Now we prove the sufficient condition, by contradiction. Assume that at some t a packet with deadline t is not yet served. In time slot t, the packet served has a deadline $\leq t$, otherwise our packet would have been chosen instead. Define s_0 such that the time interval $[s_0 + 1, t]$ is the maximum time interval ending at t that is within a busy period and for which all packets served have deadlines $\leq t$.

Now call $\mathcal{S}$ the set of flows that have a packet with deadline $\leq t$ present in the system at some point in the interval $[s_0+1, t]$. We show that if

$$\text{if } i \in \mathcal{S} \text{ then } R'_i(s_0) = R_i(s_0) \tag{2.14}$$

that is, flow i is not backlogged at the end of time slot s_0. Indeed, if s_0+1 is the beginning of the busy period, then the property is true for any flow. Otherwise, we proceed by contradiction. Assume that $i \in \mathcal{S}$ and that i would have some backlog at the end of time slot s_0. At time s_0 some packet with deadline $> t$ was served; thus the deadline of all packets remaining in the queue at the end of time slot s_0 must have a deadline $> t$. Since deadlines are assumed wide-sense increasing within a flow, all deadlines of flow i packets that are in the queue at time s_0, or will arrive later, have deadline $> t$, which contradicts that $i \in \mathcal{S}$.

Further, it follows from the last argument that if $i \in \mathcal{S}$, then all packets served before or at t must have a deadline $\leq t$. Thus

$$\text{if } i \in \mathcal{S} \text{ then } R'_i(t) \leq Z_i(t)$$

Now since there is at least one packet with deadline $\leq t$ not served at t, the previous inequality is strict for at least one i in $\mathcal{S}$. Thus

$$\sum_{i\in\mathcal{S}} R'_i(t) < \sum_{i\in\mathcal{S}} Z_i(t) \tag{2.15}$$

Observe that all packets served in $[s_0+1, t]$ must be from flows in $\mathcal{S}$. Thus

$$\sum_{i=1}^{I}(R'_i(t) - R'_i(s_0)) = \sum_{i\in\mathcal{S}}(R'_i(t) - R'_i(s_0))$$

Combining with Equation (2.14) and Equation (2.15) gives

$$\sum_{i=1}^{I}(R'_i(t) - R'_i(s_0)) < \sum_{i\in\mathcal{S}}(Z_i(t) - R_i(s_0))$$

Now $[s_0+1, t]$ is entirely in a busy period thus $\sum_{i=1}^{I}(R'_i(t) - R'_i(s_0)) = C(t-s_0)$; thus

$$C(t-s_0) < \sum_{i\in\mathcal{S}}(Z_i(t)-R_i(s_0)) = \sum_{i\in\mathcal{S}}(Z_i(t)-R_i(s_0))^+ \leq \sum_{i=1}^{I}(Z_i(t)-R_i(s_0))^+$$

which contradicts Equation (2.13). □

A consequence of the proposition that if a set of flows is schedulable for some deadline allocation algorithm, then it is also schedulable for any other deadline allocation method that produces later or equal deadlines. Other consequences, of immediate practical importance, are drawn in the next section.

2.3.2 SCED Schedulers [69]

Given, for all i, a function β_i, SCED defines a deadline allocation algorithm that guarantees, under some conditions, that flow i does have β_i as a minimum service curve[1]. Roughly speaking, SCED sets $Z_i(t)$, the number of packets with deadline up to t, to $(R_i \otimes \beta_i)(t)$.

Definition 2.3.1 (SCED). *Call A_i^n the arrival time for packet n of flow i. Define functions R_i^n by:*

$$R_i^n(t) = \inf_{s \in [0, A_i^n]} [R_i(s) + \beta_i(t-s)]$$

With SCED, the deadline for packet n of flow i is defined by

$$D_i^n = (R_i^n)^{-1}(n) = \min\{t \in \mathbb{N} : R_i^n(t) \geq n\}$$

Function β_i is called the "target service curve" for flow i.

Function R_i^n is similar to the min-plus convolution $R_i \otimes \beta_i$, but the minimum is computed over all times up to A_i^n. This allows to compute a packet deadline as soon as the packet arrives; thus SCED can be implemented in real time. The deadline is obtained by applying the pseudo-inverse of R_i^n, as illustrated on Figure 2.5. If $\beta_i = \delta_{d_i}$, then it is easy to see that $D_i^n = A_i^n + d_i$, namely, SCED is the delay based scheduler in that case. The following proposition is the main property of SCED. It

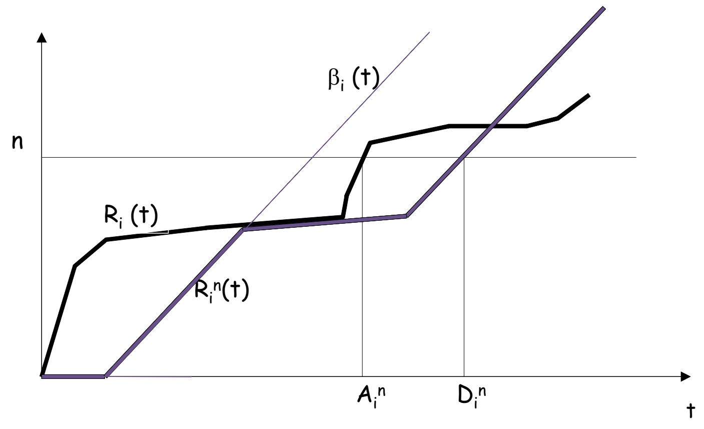

Figure 2.5: Definition of SCED. Packet n of flow i arrives at time A_i^n. Its deadline is D_i^n.

shows that SCED implements a deadline allocation method based on service curves.

Proposition 2.3.2. *For the SCED scheduler, the number of packets with deadline $\leq t$ is given by $Z_i(t) = \lfloor (R_i \otimes \beta_i)(t) \rfloor$*

[1]We use the original work in [69], which is called there "SCED-B". For simplicity, we call it SCED.

Proof: We drop index i in this demonstration. First, we show that $Z(t) \geq \lfloor (R \otimes \beta)(t) \rfloor$. Let $n = \lfloor (R \otimes \beta)(t) \rfloor$. Since $R \otimes \beta \leq R$ and R takes integer values, we must have $R(t) \geq n$ and thus $A^n \leq t$. Now $R^n(t) \geq (R \otimes \beta)(t)$ thus

$$R^n(t) \geq (R \otimes \beta)(t) \geq n$$

By definition of SCED, D^n this implies that $D^n \leq t$ which is equivalent to $Z(t) \geq n$.

Conversely, for some fixed but arbitrary t, let now $n = Z(t)$. Packet n has a deadline $\leq t$, which implies that $A^n \leq t$ and for all $s \in [0, A^n]$:

$$R(s) + \beta(t - s) \geq n \tag{2.16}$$

Now for $s \in [A^n, t]$ we have $R(s) \geq n$ thus $R(s) + \beta(t - s) \geq n$. Thus Equation (2.16) is true for all $s \in [0, t]$, which means that $(R \otimes \beta)(t) \geq n$. □

Theorem 2.3.1 (Schedulability of SCED, ATM). *Consider a SCED scheduler with I flows, total outgoing rate C, and target service curve β_i for flow i.*

1. *If*
$$\sum_{i=1}^{I} \beta_i(t) \leq Ct \text{ for all } t \geq 0 \tag{2.17}$$
then every packet is served before or at its deadline and every flow i receives $\lfloor \beta_i \rfloor$ as a service curve.

2. *Assume that in addition we know that every flow i is constrained by an arrival curve α_i. If*
$$\sum_{i=1}^{I} (\alpha_i \otimes \beta_i)(t) \leq Ct \text{ for all } t \geq 0 \tag{2.18}$$
then the same conclusion holds

Proof:

1. Proposition 2.3.2 implies that $Z_i(t) \leq R_i(s) + \beta_i(t - s)$ for $0 \leq s \leq t$. Thus $Z_i(t) - R_i(s) \leq \beta_i(t - s)$. Now $0 \leq \beta_i(t - s)$ thus
$$[Z_i(t) - R_i(s)]^+ = \max[Z_i(t) - R_i(s), 0] \leq \beta_i(t - s)$$
By hypothesis, $\sum_{i=1}^{I} \beta_i(t - s) \leq C(t - s)$ thus by application of Proposition 2.3.1, we know that every packet is served before or at its deadline. Thus $R'_i \geq Z_i$ and from Proposition 2.3.2:
$$R'_i \geq Z_i = \lfloor \beta_i \otimes R_i \rfloor$$
Now R_i takes only integer values thus $\lfloor \beta_i \otimes R_i \rfloor = \lfloor \beta_i \rfloor \otimes R_i$.

2. By hypothesis, $R_i = \alpha_i \otimes R_i$ thus $Z_i = \lfloor \alpha_i \otimes \beta_i \otimes R_i \rfloor$ and we can apply the same argument, with $\alpha_i \otimes \beta_i$ instead of β_i. □

Schedulability of delay based schedulers A delay based scheduler assigns a delay objective d_i to all packets of flow i. A direct application of Theorem 2.3.1 gives the following schedulability condition.

Theorem 2.3.2 ([52]). *Consider a delay based scheduler that serves I flows, with delay d_i assigned to flow i. All packets have the same size and time is slotted. Assume flow i is α_i-smooth, where α_i is sub-additive. Call C the total outgoing bit rate. Any mix of flows satisfying these assumptions is schedulable if*

$$\sum_i \alpha_i(t-d_i) \leq Ct$$

If $\alpha_i(t) \in \mathbb{N}$ then the condition is necessary.

Proof: A delay based scheduler is a special case of SCED, with target service curve $\beta_i = \delta_{d_i}$. This shows that the condition in the theorem is sufficient. Conversely, consider the greedy flows given by $R_i(t) = \alpha_i(t)$. This is possible because α_i is assumed to be sub-additive. Flow R_i must be schedulable, thus the output R'_i satisfies $R'_i(t) \geq \alpha_i(i-d_i)$. Now $\sum_i R'_i(t) \leq ct$, which proves that the condition must hold. □

It is shown in [52] that a delay based scheduler has the largest schedulability region among all schedulers, given arrival curves and delay budgets for every flow. Note however that in a network setting, we are interested in the end-to-end delay bound, and we know (Section 1.4.3) that it is generally less than the sum of per hop bounds.

The schedulability of delay based schedulers requires that an arrival curve is known and enforced at every node in the network. Because arrival curves are modified by network nodes, this motivates the principle of Rate Controlled Service Disciplines (RCSDs) [40, 78, 28], which implement in every node a packet shaper followed by a delay based scheduler. The packet shaper guarantees that an arrival curve is known for every flow. Note that such a combination is not work conserving.

Because of the ”pay bursts only once” phenomenon, RCSD might provide end-to-end delay bounds that are worse than guaranteed rate nodes. However, it is possible to avoid this by aggressively reshaping flows in every node, which, from Theorem 2.3.2, allows us to set smaller deadlines. If the arrival curves constraints on all flows are defined by a single leaky bucket, then it is shown in [63, 62] that one should reshape a flow to its sustained rate at every node in order to achieve the same end-to-end delay bounds as GR nodes would.

Schedulability of GR nodes Consider the family of GR nodes, applied to the ATM case. We cannot give a general schedulability condition, since the fact that a scheduler is of the GR type does not tell us exactly how the scheduler operates. However, we show that for any rate r and delay v we can implement a GR node with SCED.

Theorem 2.3.3 (GR node as SCED, ATM case). *Consider the SCED scheduler with I flows and outgoing rate C. Let the target service curve for flow i be equal to the rate-latency service curve with rate r_i and latency v_i. If*

$$\sum_{i=1}^{I} r_i \leq C$$

then the scheduler is a GR node for each flow i, with rate r_i and delay v_i.

Proof: From Proposition 2.3.2:

$$Z_i(t) = \lfloor (R_i \otimes \lambda_{r_i})(t - v_i) \rfloor$$

thus Z_i is the output of the constant rate server, with rate r_i, delayed by v_i. Now from Theorem 2.3.1 the condition in the theorem guarantees that $R'_i \geq Z_i$, thus the delay for any packet of flow i is bounded by the delay of the constant rate server with rate r_i, plus v_i. □

Note the fundamental difference between rate based and delay based schedulers. For the former, schedulability is a condition on the sum of the rates; it is independent of the input traffic. In contrast, for delay based schedulers, schedulability imposes a condition on the arrival curves. Note however that in order to obtain a delay bound, we need some arrival curves, even with delay based schedulers.

Better than Delay Based scheduler A scheduler need not be either rate based or delay based. Rate based schedulers suffer from coupling between delay objective and rate allocation: if we want a low delay, we may be forced to allocate a large rate, which because of Theorem 2.3.3 will reduce the number of flows than can be scheduled. Delay based schedulers avoid this drawback, but they require that flows be reshaped at every hop. Now, with clever use of SCED, it is possible to obtain the benefits of delay based schedulers without paying the price of implementing shapers.

Assume that for every flow i we know an arrival curve α_i and we wish to obtain an end-to-end delay bound d_i. Then the smallest network service curve that should be allocated to the flow is $\alpha_i \otimes \delta_{d_i}$ (the proof is easy and left to the reader). Thus a good thing to do is to build a scheduler by allocating to flow i the target service curve $\alpha_i \otimes \delta_{d_i}$. The schedulability condition is the same as with a delay based scheduler, however, there is a significant difference: the service curve is guaranteed even if some flows are not conforming to their arrival curves. More precisely, if some flows do not conform to the arrival curve constraint, then the service curve is still guaranteed, but the delay bound is not.

This observation can be exploited to allocate service curves in a more flexible way than what is done in Section 2.2 [18]. Assume flow i uses the sequence of nodes $m = 1, ..., M$. Every node receives a part d_i^m of the delay budget d_i, with $\sum_{m=1}^{M} d_i^m \leq d_i$. Then it is sufficient that every node implements SCED with a target service curve $\beta_i^m = \delta_{d_i^m} \otimes \alpha_i$ for flow i. The schedulability condition at node m is

$$\sum_{j \in E_m} \alpha_j(t - d_j^m) \leq C_m t$$

where E_m is the set of flows scheduled at node m and C_m is the outgoing rate of node m. If it is satisfied, then flow i receives $\alpha_i \otimes \delta_{d_i}$ as end-to-end service curve and therefore has a delay bounded by d_i. The schedulability condition is the same as if we had implemented at node m the combination of a delay based scheduler with delay budget d_i^m, and a reshaper with shaping curve α_i; but we do not have to implement a reshaper. In particular, the delay bound for flow i at node m is larger than d_i^m; we find again the fact that the end-to-end delay bound is less than the sum of individual bounds.

In [69], it is explained how to allocate a service curves β_i^m to every network element m on the path of the flow, such that $\beta_i^1 \otimes \beta_i^2 \otimes ... = \alpha_i \otimes \delta_i$, in order to obtain a large schedulability set. This generalizes and improves the schedulability region of RCSD.

Extension to variable length packets We can extend the previous results to variable length packets; we follow the ideas in [11]. The first step is to consider a fictitious preemptive EDF scheduler (system I), that allocates a deadline to every bit. We define $Z_i^I(t)$ as before, as the number of bits whose deadline is $\leq t$. A preemptive EDF scheduler serves the bits present in the system in order of their deadlines. It is preemptive (and fictitious) in that packets are not delivered entirely, but, in contrast, are likely to be interleaved. The results in the previous sections apply with no change to this system.

The second step is to modify system I by allocating to every bit a deadline equal to the deadline of the last bit in the packet. Call it system II. We have $Z_i^{II}(t) = P^{L_i}(Z_i^I(t))$ where P^{L_i} is the cumulative packet length (Section 1.7) for flow i. From the remarks following Proposition 2.3.1, it follows that if system I is schedulable, then so is system II. System II is made of a preemptive EDF scheduler followed by a packetizer.

The third step consists in defining "packet-EDF" scheduler (system III); this is derived from system II in the same way as PGSP is from GPS. More precisely, the packet EDF scheduler picks the next packet to serve among packets present in the system with minimum deadline. Then, when a packet is being served, it is not interrupted. We also say that system III is the non-preemptive EDF scheduler. Then the departure time of any packet in system III is bounded by its departure time in system II plus $\frac{l_{\max}}{C}$ where $l_{\max}$ is the maximum packet size across all flows and C is the total outgoing rate. The proof is similar to Proposition 2.1.1 and is left to the reader (it can also be found in [11]).

We can apply the three steps above to a SCED scheduler with variable size packets, called "Packet-SCED".

Definition 2.3.2 (Packet SCED). *A PSCED schedulers is a non-premptive EDF schedulers, where deadlines are allocated as follows. Call A_i^n the arrival time for* *packet n of flow i. Define functions R_i^n by:*

$$R_i^n(t) = \inf_{s \in [0, A_i^n]} [R_i(s) + \beta_i(t-s)]$$

With PSCED, the deadline for packet n of flow i is defined by

$$D_i^n = (R_i^n)^{-1}(L_i(n)) = \min\{t \in \mathbb{N} : R_i^n(t) \geq (L_i(n))\}$$

where L_i is the cumulative packet length for flow i. Function β_i is called the "target service curve" for flow i.

The following proposition follows from the discussion above.

Proposition 2.3.3. *[11] Consider a PSCED scheduler with I flows, total outgoing rate C, and target service curve β_i for flow i. Call $l^i_{\max}$ the maximum packet size for flow i and let $l_{\max} = \max_i l^i_{\max}$.*

1. *If*

$$\sum_{i=1}^{I} \beta_i(t) \leq Ct \text{ for all } t \geq 0 \tag{2.19}$$

then every packet is served before or at its deadline plus $\frac{l_{\max}}{C}$. A bound on packet delay is $h(\alpha_i, \beta_i) + \frac{l_{\max}}{C}$. Moreover, every flow i receives $[\beta_i(t - l^i_{\max}) - \frac{l_{\max}}{C}]^+$ as a service curve.

2. *Assume that, in addition, we know that every flow i is constrained by an arrival curve α_i. If*

$$\sum_{i=1}^{I} (\alpha_i \otimes \beta_i)(t) \leq Ct \text{ for all } t \geq 0 \tag{2.20}$$

then the same conclusion holds.

Note that the first part of the conclusion means that the maximum packet delay can be computed by assuming that flow i would receive β_i (not $\beta_i(t - l^i_{\max})$) as a service curve, and adding $\frac{\max}{C}$.

Proof: It follows from the three steps above that the PSCED scheduler can be broken down into a preemptive EDF scheduler, followed by a packetizer, followed by a delay element. The rest follows from the properties of packetizers and Theorem 2.3.1.

2.3.3 Buffer Requirements

As we mentioned at the beginning of this section, buffer requirements have to be computed in order to accept a reservation. The condition is simply $\sum_i X_i \leq X$ where X_i is the buffer required by flow i at this network element, and X is the total buffer allocated to the class of service. The computation of X_i is based on Theorem 1.4.1; it requires computing an arrival curve of every flow as it reaches the node. This is done using Theorem 1.4.2 and the flow setup algorithm, such as in Definition 2.2.1.

It is often advantageous to reshape flows at every node. Indeed, in the absence of reshaping, burstiness is increased linearly in the number of hops. But we know that reshaping to an initial constraint does not modify the end-to-end delay bound and does not increase the buffer requirement at the node where it is implemented. If reshaping is implemented per flow, then the burstiness remains the same at every node.

2.4 Application to Differentiated Services

2.4.1 Differentiated Services

In addition to the reservation based services we have studied in Section 2.2, the Internet also proposes differentiated services [7]. The major goal of differentiated services is to provide some form of better service while avoiding per flow state information as is required by integrated services. The idea to achieve this is based on the following principles.

- Traffic classes are defined; inside a network, all traffic belonging to the same class is treated as one single aggregate flow.

- At the network edge, individual flows (called "micro-flows") are assumed to conform to some arrival curve, as with integrated services.

If the aggregate flows receive appropriate service curves in the network, and if the total traffic on every aggregate flow is not too large, then we should expect some bounds on delay and loss. The condition on microflows is key to ensuring that the total aggregate traffic remains within some arrival curve constraints. A major difficulty however, as we will see, is to derive bounds for individual flows from characteristics of an aggregate.

Differentiated services is a framework that includes a number of different services. The main two services defined today are expedited forwarding (EF)[21, 5] and assured forwarding (AF)[36]. The goal of EF is to provide to an aggregate some hard delay guarantees, and no loss. The goal of AF is to separate traffic between a small number of classes (4); inside each class, three levels of drop priorities are defined. One of the AF classes could be used to provide a low delay service with no loss, similar to EF.

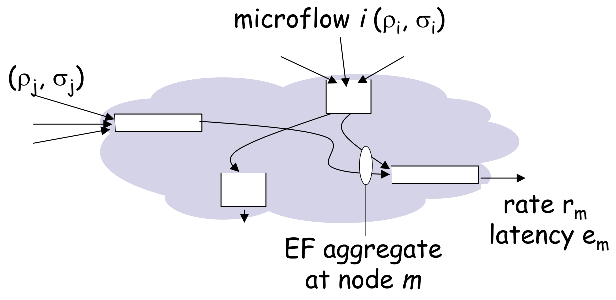

Figure 2.6: Network Model for EF. Microflows are individually shaped and each conform to some arrival curve. At all nodes, microflows R_1 to R_3 are handled as one aggregate flow, with a guaranteed rate (GR) guarantee. Upon leaving a node, the different microflows take different paths and become part of other aggregates at other nodes.

In this chapter, we focus on the fundamental issue of how aggregate scheduling impacts delay and throughput guarantees. In the rest of this section, we use the network model shown on Figure 2.6. Our problem is to find bounds for end-to-end delay jitter on one hand, for backlog at all nodes on the other hand, under the assumptions mentioned above. Delay jitter is is the difference between maximum and minimum delay; its value determines the size of playout buffers (Section 1.1.3).

2.4.2 An Explicit Delay Bound for EF

We consider EF, the low delay traffic class, as mentioned in Section 2.4.1, and find a closed form expression for the worst case delay, which is valid in any topology, in a lossless network. This bound is based on a general time stopping method explained in detail in Chapter 6. It was obtained in [13] and [39].

Assumption and Notation (See Figure 2.6)

- Microflow i is constrained by the arrival curve $\rho_i t + \sigma_i$ at the network access. Inside the network, EF microflows are *not* shaped.
- Node m acts as a Guaranteed Rate node for the entire EF aggregate, with rate r_m and latency e_m. This is true in particular if the aggregate is served as one flow in a FIFO service curve element, with a rate-latency service curve; but it also holds quite generally, even if nodes are non-FIFO (Section 2.1.3). In Chapter 6, we explain that the generic node model used in the context of EF is packet scale rate guarantee, which satisfies this assumption.

  Let e be an upper bound on e_m for all m.

- h is a bound on the number of hops used by any flow. This is typically 10 or less, and is much less than the total number of nodes in the network.
- Utilization factors: Define $\nu_m = \frac{1}{r_m} \sum_{i \ni m} \rho_i$, where the notation $i \ni m$ means that node m is on the path of microflow i. Let ν be an upper bound on all v_m.
- Scaled burstiness factors: Define $\tau_m = \frac{1}{r_m} \sum_{i \ni m} \sigma_i$. Let τ be an upper bound on all τ_m.
- $L_{\max}$ is an upper bound on the size (in bits) of any EF packet.

Theorem 2.4.1 (Closed form bound for delay and backlog [13]). *If $\nu < \frac{1}{h-1}$ then a bound on end-to-end delay variation for EF is hD_1 with*

$$D_1 = \frac{e + \tau}{1 - (h-1)\nu}$$

At node m, the buffer required for serving low delay traffic without loss is bounded by $B_{req} = r_m D_1 + L_{\max}$.

Proof: (Part 1:) Assume that a finite bound exists and call D the least upper bound. The data that feeds node m has undergone a variable delay in the range $[0, (h-1)D]$, thus an arrival curve for the EF aggregate at node m is $\nu r_m(t + (h-1)D) + r_m \tau$. By application of Equation (2.3), the delay seen by any packet is bounded by $e + \tau + (h-1)D\nu$; thus $D \leq e + \tau + (h-1)D\nu$. If the utilization factor ν is less than $\frac{1}{h-1}$, it follows that $D \leq D_1$.

(Part 2:) We prove that a finite bound exists, using the time-stopping method. For any time $t > 0$, consider the virtual system made of the original network, where all sources are stopped at time t. This network satisfies the assumptions of part 1, since there is only a finite number of bits for the entire lifetime of the network. Call $D'(t)$ the worst case delay across all nodes for the virtual network indexed by t. From the above derivation we see that $D'(t) \leq D_1$ for all t. Letting t tend to $+\infty$ shows that the worst case delay at any node remains bounded by D_1.

(Part 3:) By Corollary 2.1.1, the backlog is bounded by the vertical deviation between the arrival curve $\nu r_m(t + (h-1)D) + r_m \tau$ and the service curve $[r_m(t - e_m) - L_{\max}]^+$, which after some algebra gives B_{req} □

The theorem can be slightly improved by avoiding to take maxima for ν_m; this gives the following result (the proof is left to the reader):

Corollary 2.4.1. *If $\nu < \frac{1}{h-1}$ then a bound on end-to-end delay variation for EF is hD'_1 with*

$$D'_1 = \min_m \left\{ \frac{e_m + \tau_m}{1 - (h-1)\nu_m} \right\}$$

Improved Bound When Peak Rate is Known: A slightly improved bound can be obtained if, in addition, we have some information about the total incoming bit rate at every node. We add the following assumptions to the previous list.

- Let C_m denote a bound on the peak rate of all incoming low delay traffic traffic at node m. If we have no information about this peak rate, then $C_m = +\infty$. For a router with large internal speed and buffering only at the output, C_m is the sum of the bit rates of all incoming links (the delay bound is better for a smaller C_m).
- Fan-in: Let I_m be the number of incident links at node m. Let F be an upper bound on $\frac{I_m L_{\max}}{r_m}$. F is the maximum time to transmit a number of EF packets that simultaneously appear on multiple inputs.
- Redefine $\tau_m := \max\{\frac{I_m L \max}{r_m}, \frac{1}{r_m}\sum_{i \ni m} \sigma_i\}$. Let τ be an upper bound on all τ_m.
- Let $u_m = \frac{[C_m - r_m]^+}{C_m - \nu_m r_m}$. Note that $0 \leq u_m \leq 1$, u_m increases with C_m, and if $C_m = +\infty$, then $u_m = 1$. Call $u = \max_m u_m$. The parameter $u \in [0, 1]$ encapsulates how much we gain by knowing the maximum incoming rates C_m (u is small for small values of C_m).

Theorem 2.4.2 (Improved Delay Bound When Peak Rate is Known [13, 39]). *Let $\nu^* = \min_m\{\frac{C_m}{(h-1)(C_m - r_m)^+ + r_m}\}$. If $\nu < \nu^*$, a bound on end-to-end delay variation for EF is hD_2 with*

$$D_2 = \frac{e + u\tau + (1-u)F}{1 - (h-1)u\nu}$$

Proof: The proof is similar to the proof of Theorem 2.4.1. Call D the least bound, assuming it exists.

An arrival curve for the flow of EF packets arriving at node m on some incident link l is $C_m^l t + L_{\max}$, where C_m^l is the peak rate of the link (this follows from item 4 in Theorem 1.7.1). Thus an arrival curve for the incoming flow of EF packets at node m is $C_m t + I_m L_{\max}$. The incoming flow is thus constrained by the T-SPEC (M, p, r, b) (see Page 16) with $M = I_m L_{\max}$, $p = C_m$, $r = r_m \nu_m$, $b = r_m \tau_m + (h-1) D r_m \nu_m$. By Proposition 1.4.1, it follows that

$$D \leq \frac{I_m L_{\max}(1 - u_m)}{r_m} + (\tau_m + (h-1)D\nu_m)u_m$$

The condition $\nu < \nu^*$ implies that $1 - (h-1)\nu_m u_m > 0$, thus

$$D \leq \frac{e_m + \tau_m u_m + \frac{I_m L_{\max}(1-u_m)}{r_m}}{1 - (h-1)\nu_m u_m}$$

The above right-hand-side is an increasing function of u_m, due to $\tau_m \geq \frac{I_m L\max}{r_m}$. Thus we have a bound by replacing u_m by u:

$$D \leq \frac{e_m + \tau_m u + \frac{I_m L_{\max}(1-u)}{r_m}}{1-(h-1)\nu_m u} \leq D_2$$

The rest of the proof follows along lines similar to the proof of Theorem 2.4.1. $\square$

It is also possible to derive an improved backlog bound, using Proposition 1.4.1. As with Theorem 2.4.2, we also have the following variant.

Corollary 2.4.2. *If $\nu < \nu^*$, a bound on end-to-end delay variation for EF is hD'_2 with*

$$D'_2 = \min_m \left\{ \frac{e_m + \tau_m u_m + \frac{I_m L_{\max}(1-u_m)}{r_m}}{1-(h-1)\nu_m u_m} \right\}$$

Discussion: If we have no information about the peak incoming rate C_l, then we set $C_l = +\infty$ and Theorem 2.4.2 gives the same bound as Theorem 2.4.2. For finite values of C_m, the delay bound is smaller, as illustrated by Figure 2.7.

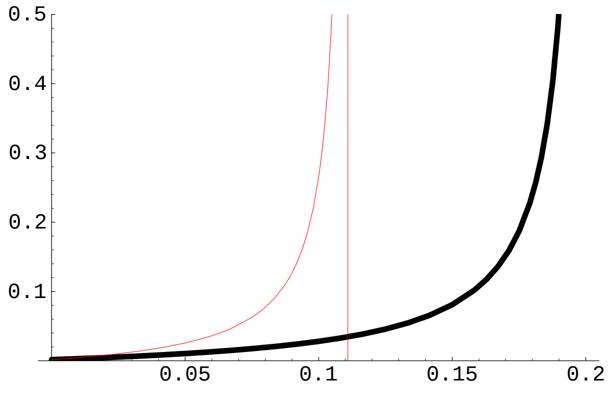

Figure 2.7: The bound D (in seconds) in Theorem 2.4.1 versus the utilization factor ν for $h = 10$, $e = 2\frac{1500B}{r_m}$, $L_{\max} = 1000$ b, $\sigma_i = 100$B and $\rho_i = 32$kb/s for all flows, $r_m = 149.760$Mb/s, and $C_m = +\infty$ (thin line) or $C_m = 2r_m$ (thick line).

The bound is valid only for small utilization factors; it explodes at $\nu > \frac{1}{h-1}$, which does not mean that the worst case delay does grow to infinity [38]. In some cases the network may be unbounded; in some other cases (such as the unidirectional ring, there is always a finite bound for all $\nu < 1$. This issue is discussed in Chapter 6, where we we find better bounds, at the expense of more restrictions on the routes and the rates. Such restrictions do not fit with the differentiated services framework.

Note also that, for feed-forward networks, we know that there are finite bounds for $\nu < 1$. However we show now that the condition $\nu < \frac{1}{h-1}$ is the best that can be obtained, in some sense.

Proposition 2.4.1. *[4, 13] With the assumptions of Theorem 2.4.1, if $\nu < \frac{1}{h-1}$, then for any $D' > 0$, there is a network in which the worst case delay is at least D'.*

In other words, the worst case queuing delay can be made arbitrarily large; thus if we want to go beyond Theorem 2.4.1, any bound for differentiated services must depend on the network topology or size, not only on the utilization factor and the number of hops.

Proof: We build a family of networks, out of which, for any D', we can exhibit an example where the queuing delay is at least D'.

The thinking behind the construction is as follows. All flows are low priority flows. We create a hierarchical network, where at the first level of the hierarchy we choose one "flow" for which its first packet happens to encounter just *one* packet of every other flow whose route it intersects, while its next packet does not encounter any queue at all. This causes the first two packets of the chosen flow to come back-to-back after several hops. We then construct the second level of the hierarchy by taking a new flow and making sure that its first packet encounters *two* back-to-back packets of each flow whose routes it intersects, where the two back-to-back packet bursts of all these flows come from the output of a sufficient number of networks constructed as described at the first level of the hierarchy. Repeating this process recursively sufficient number of times, for any chosen delay value D we can create deep enough hierarchy so that the queuing delay of the first packet of some flow encounters a queuing delay more than D (because it encounters a large enough back-to-back burst of packets of every other flow constructed in the previous iteration), while the second packet does not suffer any queuing delay at all. We now describe in detail how to construct such a hierarchical network (which is really a family of networks) such that utilization factor of any link does not exceed a given factor ν, and no flow traverses more than h hops.

Now let us describe the networks in detail. We consider a family of networks with a single traffic class and constant rate links, all with same bit rate C. The network is assumed to be made of infinitely fast switches, with one output buffer per link. Assume that sources are all leaky bucket constrained, but are served in an aggregate manner, first in first out. Leaky bucket constraints are implemented at the network entry; after that point, all flows are aggregated. Without loss of generality, we also assume that propagation delays can be set to 0; this is because we focus only on queuing delays. As a simplification, in this network, we also assume that all packets have a unit size. We show that for any fixed, but arbitrary delay budget D, we can build a network of that family where the worst case queueing delay is larger than D, while each flow traverses at most a specified number of hops.

A network in our family is called $\mathcal{N}(h, \nu, J)$ and has three parameters: h (maximum hop count for any flow), ν (utilization factor) and J (recursion depth). We focus on the cases where $h \geq 3$ and $\frac{1}{h-1} < \nu < 1$, which implies that we can always find some integer k such that

$$\nu > \frac{1}{h-1} \frac{kh+1}{kh-1} \tag{2.21}$$

Network $\mathcal{N}(h, \nu, J)$ is illustrated in Figures 2.8 and 2.9; it is a collection of identical building blocks, arranged in a tree structure of depth J. Every building block has one internal source of traffic (called "transit traffic"), $kh(h-1)$ inputs (called the "building block inputs"), $kh(h-1)$ data sinks, $h-1$ internal nodes, and one output. Each of the $h-1$ internal nodes receives traffic from kh building block inputs plus it receives transit traffic from the previous internal node, with the exception of the first one which is fed by the internal source. After traversing one internal node, traffic from the building block inputs dies in a data sink. In contrast, transit traffic is fed to the next internal node, except for the last one which feeds the building block output (Figure 2.8). Figure 2.9 illustrates that our network has the structure of a complete

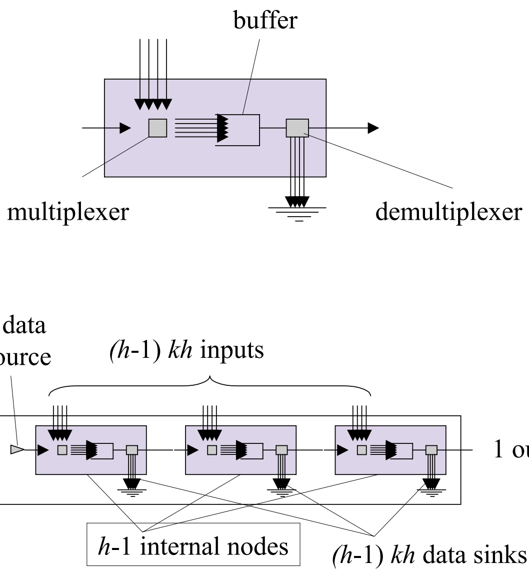

Figure 2.8: The internal node (top) and the building block (bottom) used in our network example.

tree, with depth J. The building blocks are organized in levels $j = 1, ..., J$. Each of the inputs of a level j building block ($j \geq 2$) is fed by the output of one level $j-1$ building block. The inputs of level 1 building blocks are data sources. The output of one $j-1$ building block feeds exactly one level j building block input. At level J, there is exactly one building block, thus at level $J-1$ there are $kh(h-1)$ building blocks, and at level 1 there are $(kh(h-1))^{J-1}$ building blocks. All data sources

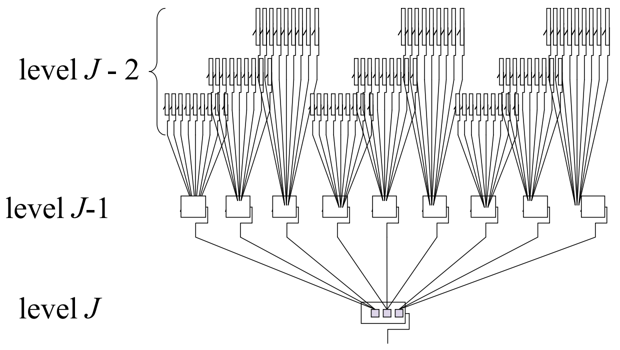

Figure 2.9: The network made of building blocks from Figure 2.8

have the same rate $r = \frac{\nu C}{kh+1}$ and burst tolerance $b = 1$ packet. In the rest of this section we take as a time unit the transmission time for one packet, so that $C = 1$. Thus any source may transmit one packet every $\theta = \frac{kh+1}{\nu}$ time units. Note that a source may refrain from sending packets, which is actually what causes the large delay jitter. The utilization factor on every link is ν, and every flow uses 1 or h hops.

Now consider the following scenario. Consider some arbitrary level 1 building block. At time t_0, assume that a packet fully arrives at each of the building block inputs of level 1, and at time $t_0 + 1$, let a packet fully arrive from each data source inside every level 1 building block (this is the first transit packet). The first transit packet is delayed by $hk - 1$ time units in the first internal node. Just one time unit before this packet leaves the first queue, let one packet fully arrive at each input of the second internal node. Our first transit packet will be delayed again by $hk - 1$ time units. If we repeat the scenario along all internal nodes inside the building block, we see that the first transit packet is delayed by $(h-1)(hk-1)$ time units. Now from Equation (2.21), $\theta < (h-1)(hk-1)$, so it is possible for the data source to send a second transit packet at time $(h-1)(hk-1)$. Let all sources mentioned so far be idle, except for the emissions already described. The second transit packet will catch up to the first one, so the output of any level 1 building block is a burst of two back-to-back packets. We can choose t_0 arbitrarily, so we have a mechanism for generating bursts of 2 packets.

Now we can iterate the scenario and use the same construction at level 2. The level-2 data source sends exactly three packets, spaced by θ. Since the internal node receives hk bursts of two packets originating from level 1, a judicious choice of the level 1 starting time lets the first level 2 transit packet find a queue of $2hk-1$ packets in the first internal node. With the same construction as in level 1, we end up with a total queuing delay of $(h-1)(2hk-1) > 2(h-1)(hk-1) > 2\theta$ for that packet.

Now this delay is more than 2θ, and the first three level-2 transit packets are delayed by the same set of non-transit packets; as a result, the second and third level-2 transit packets will eventually catch up to the first one and the output of a level 2 block is a burst of three packets. This procedure easily generalizes to all levels up to J. In particular, the first transit packet at level J has an end-to-end delay of at least $J\theta$. Since all sources become idle after some time, we can easily create a last level J transit packet that finds an empty network and thus a zero queuing delay.

Thus there are two packets in network $\mathcal{N}(h, \nu, J)$, with one packet having a delay larger than $J\theta$, and the other packet has zero delay. This establishes that a bound on queuing delay, and thus on delay variation in network $\mathcal{N}(h, \nu, J)$ has to be at least as large as $J\theta$. □

2.4.3 Bounds for Aggregate Scheduling with Dampers

At the expense of some protocol complexity, the previous bounds can be improved without losing the feature of aggregate scheduling. It is even possible to avoid bound explosions at all, using the concepts of *damper*. Consider an EDF scheduler (for example a SCED scheduler) and assume that every packet sent on the outgoing link carries a field with the difference d between its deadline and its actual emission time, if it is positive, and 0 otherwise. A damper is a regulator in the next downstream node that picks for the packet an eligibility time that lies in the interval $[a+d-\Delta, a+d]$, where Δ is a constant of the damper, and a is the arrival time of the packet in the node where the damper resides. We call Δ the "damping tolerance". The packet is then withheld until its eligibility time [76, 18], see Figure 2.10. In addition, we assume that the damper operates in a FIFO manner; this means that the sequence of eligibility times for consecutive packets is wide-sense increasing.

Unlike the scheduler, the damper does not exist in isolation. It is associated with the next scheduler on the path of a packet. Its effect is to forbid scheduling the packet before the eligibility time chosen for the packet. Consider Figure 2.10. Scheduler m works as follows. When it has an opportunity to send a packet, say at time t, it picks a packet with the earliest deadline, among all packets that are present in node N, and whose eligibility date is $\geq t$. The timing information d shown in the figure is carried in a packet header, either as a link layer header information, or as an IP hop by hop header extension. At the end of a path, we assume that there is no damper at the destination node.

The following proposition is obvious, but important, and is given without proof.

Proposition 2.4.2. *Consider the combination $\mathcal{S}$ of a scheduler and its associated damper. If all packets are served by the scheduler before or at their deadlines, then $\mathcal{S}$ provides a bound on delay variation equal to Δ.*

It is possible to let $\Delta = 0$, in which case the delay is constant for all packets. A bound on the end-to-end delay variation is then the delay bound at the last scheduler using the combination of a scheduler and a damper (this is called "jitter EDD" in [76]). In practice, we consider $\Delta > 0$ for two reasons. Firstly, it is impractical to assume that we can write the field d with absolute accuracy. Secondly, having some

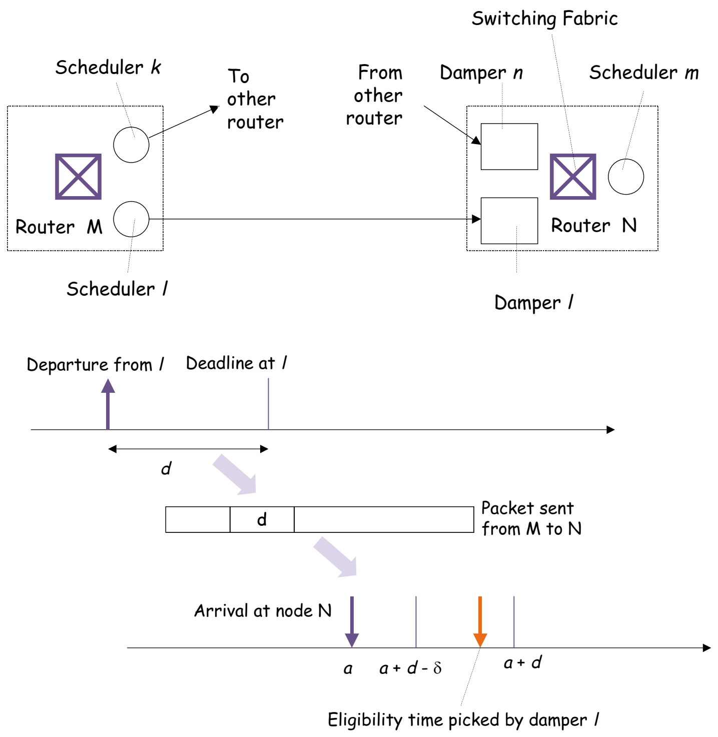

Figure 2.10: Dampers in a differentiated services context. The model shown here assumes that routers are made of infinitely fast switching fabrics and output schedulers. There is one logical damper for each upstream scheduler. The damper decides when an arriving packet becomes visible in the node.

slack in the delay variation objective provides better performance to low priority traffic [18].

There is no complicated feasibility condition for a damper, as there is for schedulers. The operation of a damper is always possible, as long as there is enough buffer.

Proposition 2.4.3 (Buffer requirement for a damper). *If all packets are served by the scheduler before or at their deadlines, then the buffer requirement at the associated damper is bounded by the buffer requirement at the scheduler.*

Proof: Call $R_(t)$ the total input to the scheduler, and $R'(t)$ the amount of data with deadline $\leq t$. Call $R^*(t)$ the input to the damper, we have $R^*(t) \leq R(t)$. Packets do not stay in the damper longer than until their deadline in the scheduler, thus the output $R_1(t)$ of the damper satisfies $R_1(t) \geq R'(t)$. The buffer requirement at the scheduler at time t is $R(t) - R'(t)$; at the damper it is $R^*(t) - R_1(t) \geq R(t) - R'(t)$. □

Theorem 2.4.3 (Delay and backlog bounds with dampers). *Take the same assumptions as in Theorem 2.4.1, we assume that every scheduler m that is not an exit point is associated with a damper in the next downstream node, with damping tolerance Δ_m. Let Δ be a bound on all Δ_m.*

If $\nu \leq 1$, then a bound on the end-to-end delay jitter for low delay traffic is

$$D = e + (h-1)\Delta(1+\nu) + \tau\nu$$

A bound on the queuing delay at any scheduler is

$$D_0 = e + \nu[\tau + (h-1)\Delta]$$

The buffer required at scheduler m, for serving low delay traffic without loss is bounded by

$$B_{req} = r_m D_0$$

A bound on the buffer required at damper m is the same as the buffer required at scheduler m.

Proof: The variable part of the delay between the input of a scheduler and the input of the next one is bounded by Δ. Now let us examine the last scheduler, say m, on the path of a packet. The delay between a source for a flow $i \ni m$ and scheduler m is a constant plus a variable part bounded by $(h-1)\Delta$. Thus an arrival curve for the aggregate low-delay traffic arriving at scheduler m is

$$\alpha_2(t) = \nu r_m(t + \tau + (h-1)\Delta)$$

By applying Theorem 1.4.2, a delay bound at scheduler m is given by

$$D_2 = E + u\nu[\tau + (h-1)\Delta]$$

A bound on end-to-end delay variation is $(h-1)\Delta + D_2$, which is the required formula.

The derivation of the backlog bound is similar to that in Theorem 2.4.1. $\square$

The benefit of dampers is obvious: there is no explosion to the bound, it is finite (and small if Δ is small) for any utilization factor up to 1 (see Figure 2.11). Furthermore, the bound is dominated by $h\Delta$, across the whole range of utilization factors up to 1. A key factor in obtaining little delay variation is to have a small damping tolerance δ.

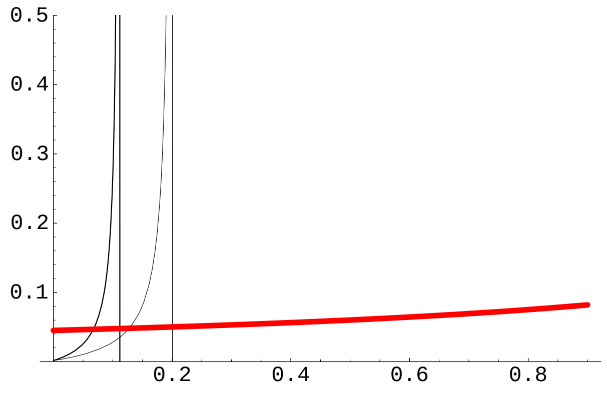

Figure 2.11: The bound D (in seconds) in Theorem 2.4.3 the same parameters as Figure 2.7, for a damping tolerance $\Delta = 5$ ms per damper, and $C_m = +\infty$ (thick line). The figure also shows the two curves of Figure 2.7, for comparison. The bound is very close to $h\Delta = 0.05$s, for all utilization factors up to 1.

There is a relation between a damper and a maximum service curve. Consider the combination of a scheduler with minimum service curve β and its associate damper with damping tolerance Δ. Call p the fixed delay on the link between the two. It follows immediately that the combination offers the maximum service curve $\beta \otimes \delta_{p-\Delta}$ and the minimum service curve $\beta \otimes \delta_p$. Thus a damper may be viewed as a way to implement maximum service curve guarantees. This is explored in detail in [18].

2.4.4 Static Earliest Time First (SETF)

A simpler alternative to the of dampers is proposed by Z.-L. Zhang et al under the name of Static Earliest Time First (SETF) [80].

Assumptions We take the same assumptions as with Theorem 2.4.1, with the following differences.

- At network access, packets are stamped with their time of arrival. At any node, they are served within the EF aggregate at one node in order of time stamps. Thus we assume that nodes offer a GR guarantee to the EF aggregate, as defined by Equation (2.1) or Equation (2.2), but where packets are numbered in order of time stamps (i.e. their order at the network access, not at this node).

Theorem 2.4.4. *If the time stamps have infinite precision, for all $\nu < 1$, the end-to-end delay variation for the EF aggregate is bounded by*

$$D = (e+\tau)\frac{1-(1-\nu)^h}{\nu(1-\nu)^{h-1}}$$

Proof: The proof is similar to the proof of Theorem 2.4.1. Call D_k the least bound, assuming it exists, on the end-to-end delay after k hops, $k \leq h$. Consider a tagged packet, with label n, and call d_k its delay in k hops. Consider the node m that is the hth hop for this packet. Apply Equation (2.2): there is some label $k \leq n$ such that

$$d_n \leq e + a_k + \frac{l_k + ... + l_n}{r} \tag{2.22}$$

where a_j and d_j are the arrival and departure times at node m of the packet labeled j, and l_j its length in bits. Now packets k to n must have arrived at the network access before $a_n - d_k$ and after $a_m - D_{Hh-1}$. Thus

$$l_k + ... + l_n \leq \alpha(a_n - a_m - d_k + D_{h-1})$$

where α is an arrival curve at network access for the traffic that will flow through node m. We have $\alpha(t) \leq r_m(\nu t + \tau)$. By Equation (2.3), the delay $d_n - a_n$ for our tagged packet is bounded by

$$e + \sup_{t \geq 0}\left[\frac{\alpha(t - d_k + D_{h-1})}{r_m} - t\right] = e + \tau + \nu(D_{h-1} - d_k)$$

thus

$$d_{k+1} \leq d_k + e + \tau + \nu(D_{h-1} - d_k)$$

The above inequation can be solved iteratively for d_k as a function of D_{h-1}; then take $k = h-1$ and assume the tagged packet is one that achieves the worst case k-hop delay, thus $D_{h-1} = d_{h-1}$ which gives an inequality for D_{h-1}; last, take $k = h$ and obtain the end-to-end delay bound as desired. □

Comments: The bound is finite for all values of the utilization factor $\nu < 1$, unlike the end-to-end bound in Theorem 2.4.1. Note that for small values of ν, the two bounds are equivalent.

We have assumed here infinite precision about the arrival time stamped in every packet. In practice, the timestamp is written with some finite precision; in that case, Zhang [80] finds a bound which lies between Theorem 2.4.1 and Theorem 2.4.4 (at the limit, with null precision, the bound is exactly Theorem 2.4.4).

2.5 Bibliographic Notes

The delay bound for EF in Theorem 2.4.2 was originally found in [13], but neglecting the $L_{\max}$ term; a formula that accounts for $L_{\max}$ was found in [39].

Bounds that account for statistical multiplexing can be found in [55].

2.6 Exercises

Exercise 2.1. *Consider a guaranteed rate scheduler, with rate R and delay v, that receives a packet flow with cumulative packet length L. The (packetized) scheduler output is fed into a constant bit rate trunk with rate $c > R$ and propagation delay T.*

1. *Find a minimum service curve for the complete system.*
2. *Assume the flow of packets is (r, b)-constrained, with $b > l_{\max}$. Find a bound on the end-to-end delay and delay variation.*

Exercise 2.2. *Assume all nodes in a network are of the GR type. A flow with T-SPEC $\alpha(t) = \min(rt + b, M + pt)$ has performed a reservation with rate R across a sequence of H nodes. Assume no reshaping is done. What is the buffer requirement at the hth node along the path, for $h = 1, ...H$?*

Exercise 2.3. *Assume all nodes in a network are made of a shaper followed by a GR scheduler. A flow with T-SPEC $\alpha(t) = \min(rt + b, M + pt)$ has performed a reservation with rate R across a sequence of H nodes. Assume that the shaper at every node uses the shaping curve $\sigma = \gamma_{r,b}$. What is the buffer requirement at the hth node along the path, for $h = 1, ...H$?*

Exercise 2.4. *Assume all nodes in a network are made of a shaper followed by a FIFO multiplexer. Assume that flow I has T-SPEC, $\alpha_i(t) = \min(r_i t + b_i, M + p_i t)$, that the shaper at every node uses the shaping curve $\sigma_i = \gamma_{r_i,b_i}$ for flow i. Find the schedulability conditions for every node.*

Exercise 2.5. *A network consists of two nodes in tandem. There are n_1 flows of type 1 and n_2 flows of type 2. Flows of type i have arrival curve $\alpha_i(t) = r_i t + b_i$, $i = 1, 2$. All flows go through nodes 1 then 2. Every node is made of a shaper followed by an EDF scheduler. At both nodes, the shaping curve for flows of type i is some σ_i and the delay budget for flows of type i is d_i. Every flow of type i should have a end-to-end delay bounded by D_i. Our problem is to find good values of d_1 and d_2.*

1. *We assume that $\sigma_i = \alpha_i$. What are the conditions on d_1 and d_2 for the end-to-end delay bounds to be satisfied ? What is the set of (n_1, n_2) that are schedulable ?*
2. *Same question if we set $\sigma_i = \lambda_{r_i}$*

Exercise 2.6. *Consider the scheduler in Theorem 2.3.3. Find an efficient algorithm for computing the deadline of every packet.*

Exercise 2.7. *Consider a SCED scheduler with target service curve for flow i given by*

$$\beta_i = \gamma_{r_i,b_i} \otimes \delta_{d_i}$$

Find an efficient algorithm for computing the deadline of every packet.

Hint: use an interpretation as a leaky bucket.

Exercise 2.8. *Consider the delay bound in Theorem 2.4.1. Take the same assumptions but assume also that the network is feedforward. Which better bound can you give ?*