On Average and Worst Case Behaviour in Non-Preemptive Priority Queueing

Jens Schmitt
Darmstadt University of Technology
Multimedia Communications Lab
Jens.Schmitt@KOM.tu-darmstadt.de

Keywords: **QoS, Priority Queuing, Worst Case Analysis, Network Calculus.**

Abstract: **In this paper, we derive worst case bounds on delay and backlog for non-preemptive priority queueing systems using network calculus. There are known results for the average behaviour of non-preemptive priority queueing systems from traditional queueing theory. By use of numerical investigations, we compare our worst case bounds to those average behaviour results in order to give a feel as to how conservative the worst case bounds are.**
A practical application of our results is given for DiffServ-based networks which use simple priority queueing to differentiate several traffic classes by assigning them different delay targets.

1 INTRODUCTION

1.1 Motivation

In providing multiple services in a single network there is two great directions which could be labelled by Quality of Service (QoS) and Class of Service (CoS).

With QoS usually some sort of per-flow queueing is associated which allows to give exact performance guarantees and allows to emulate flow isolation. The latter fact allows to some degree to apply basic queueing theory results for average case behaviour of a flow and there is a well established theory, network calculus [1], for analysing worst case behaviour. The drawback of per-flow traffic control mainly is that it runs into scalability problems in high multiplexing environments and is usually considered infeasible for the backbone of large networks as, e.g., the Internet [2].

By CoS, a class-based differentiation between traffic is meant. The associated per-class queueing achieves high scalability by supporting only a limited number of traffic classes in the network. Within one class flows compete for the available resources in that class. This makes the derivation of performance guarantees more difficult. In particular there is to our knowledge *no results on worst case behaviour*, while there are some results for the average behaviour under Markovian traffic assumptions.

Note that sometimes CoS is considered as subset of QoS, but for the sake of contrasting the per-flow and per-class traffic control approach we use these two terms (as it is often done in the literature).

There are several alternatives for scheduling the different classes in a class-based packet network, the simplest variant is to give non-preemptive priority to packets belonging to more important classes (numerically lower). This is, in the realm of queueing systems, usually called *non-preemptive priority queueing*. Due to its simplicity non-preemptive priority scheduling is usually implemented on today's routers.

There is two great tools or methods to investigate performance in packet networks: queueing theory [3] and network calculus [1]. Queueing theory allows to examine the average case behaviour, whereas network calculus is concerned with worst case behaviour. Seldom, are they used in the same context since their "supporters" usually consider themselves belonging to different "camps".

We use both of them in order to analyse and compare average and worst case behaviour of non-preemptive priority queueing for class-based packet networks. However with respect to average behaviour we draw upon known queueing theory results whereas for the worst case behaviour we derive bounds which have not been published so far, to the best of our knowledge.

1.2 Priority-based Class-of-Service Packet Networks

Non-preemptive priority queueing, which is available in many router products (e.g., in Cisco routers it is available under the label LLQ (Low Latency Queueing) [4]), provides some enhancement to traffic management. A priority queueing mechanism adds the ability to sort packets based on differences in "priority" and insert them into separate internal queues or shuffle their insertion within a single queue. The forwarding algorithm always transmits packets of the highest priority first. If there are no packets of the highest priority level, the next highest priority queue is serviced, and so on.

The priorities in this type of queueing are absolute, i.e., if there is sufficient high priority traffic to saturate a link, all lower priority traffic is locked out. This can be a considerable problem, since some protocols attempt to use the entire available resource. For example, in the absence of competing traffic, a greedy TCP data flow will attempt to maximize its throughput and use all of the available capacity. Thus, a single TCP data flow with a higher priority can lock out all other flows for the duration of the TCP connection. In light of the potential for lower priority traffic to be locked out, great care must be taken when assigning priority levels. The solution to this problem is to enforce bounds on the use of the higher priority classes which means to exert traffic regulation for high priority traffic.

1.3 Contribution

The contributions of this work are in the derivation of results for the worst case behaviour in non-preemptive priority queueing:

- the service curves for each traffic class for non-preemptive priority queueing are derived,
- based on the service curves bounds on delay and buffer requirements for each class are derived.

Besides the new results, we perform a comprehensive comparison between known queueing theoretical results for the average

behaviour in Markovian non-preemptive priority queueing systems with our newly derived worst case results.

1.4 Outline

The rest of the paper is structured as follows:

In Section 2, we recapitulate the known results for average behaviour in non-preemptive priority queueing systems under the assumption of Poisson arrivals and generally distributed service times. In Section 3, the worst case behaviour in non-preemptive priority queueing systems is analysed using network calculus (which is also briefly introduced). After a general derivation of the service curve for each traffic class we derive worst case delay and backlog in each class for the case of token bucket regulated input to the classes. In Section 4, a comprehensive comparison of average and worst case behaviour based on numerical investigations is provided. Section 5 reviews some related work in this area and Section 6 concludes the paper and gives an outlook to possible future work.

2 Priority Queueing: Average Case Analysis

In this section, we mainly recall known results from classical queueing theory. First, we briefly discuss what queueing theory is good for and on which assumption it builds. While this is certainly repetition for most readers it helps to contrast queueing theory results against results from network calculus which we derive in Section 3. Next, we state the available queueing theory results in the domain of non-preemptive priority queueing.

2.1 Some Background on Queueing Theory

Queueing theory has been and certainly still is *the* tool to model and analyse systems with shared resources and stochastic behaviour of system user's (see Figure 1 for a basic model of what problems queueing theory is concerned with). Although, there is also much research on transient analysis of queueing systems, queueing theory is mainly a tool for equilibrium investigations and average behaviour. One of the strong results of queueing theory is given by the so-called Pollaczek-Khinchine Formulae for M/G/1 queueing systems, i.e., systems with Poisson arrivals and general service times (although with finite variance).

Theorem 1: M/G/1 – Pollaczek-Khinchine Formulae for Average Waiting Time and Queue Size

Let us assume service times have a general distribution with average $1/\mu$ and variance σ^2 and arrivals follow a Poisson process with parameter λ. Then the average waiting time, i.e., the average time packets spend in the queue, is given by

$$E(W) = \frac{\lambda(\sigma^2 + 1/\mu^2)}{2(1-\rho)} \tag{1}$$

with $\rho = \frac{\lambda}{\mu}$, the utilization.

Under the same conditions, the average number of packets in the queue is given by

$$E(q) = \frac{\rho^2}{2(1-\rho)}(1+\mu^2\sigma^2) \ . \tag{2}$$

Note that this theorem is under the additional assumption of FIFO queueing. There is many proofs of this famous result, three of which can be found in [5].

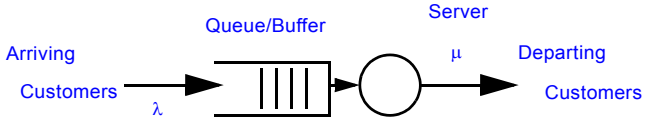

Figure 1: Basic Queueing Theory Model.

2.2 Non-Preemptive Priority Queueing in M/G/1 Systems

Since we assume non-preemptive strict priority queueing instead of FIFO queueing we need an extension on Theorem 1 under this scheduling discipline. In fact this exists and is given by the following theorem.

Theorem 2: M/G/1 – Average Waiting Time under Non-Preemptive Priority Queueing

Let us assume we have n classes each with Poisson arrivals with parameters $\lambda_1, \ldots, \lambda_n$ and general service time distributions with average $1/\mu_1, \ldots, 1/\mu_n$ and variance $\sigma_1^2, \ldots, \sigma_n^2$. The average waiting time for class i, $i = 1, \ldots, n$, is given by:

$$E(W_i) = \frac{E(T_0)}{\left(1 - \sum_{j=1}^{i} \rho_j\right)\left(1 - \sum_{j=1}^{i-1} \rho_j\right)} \tag{3}$$

with $\rho_j = \frac{\lambda_j}{\mu_j}$, and

$$E(T_0) = \lambda E(\tau^2)/2 = \sum_{j=1}^{n} \lambda_j E(\tau_j^2)/2 = \sum_{j=1}^{n} \lambda_j \frac{\sigma_j^2 + (1/\mu_j)^2}{2}$$

with $\rho = \sum_{j=1}^{n} \rho_j$, $\lambda = \sum_{j=1}^{n} \lambda_j$ and τ the service time.

While this theorem is quite powerful it still assumes Poisson arrivals and is restricted to the single node case. While the former has been relaxed to some degree the general case is still intractable as of today, the latter is also extremely difficult to treat since due to the priority queueing arrivals at subsequent nodes become dependent on each other in non-trivial manner. Nevertheless, we use this fundamental result to analyse the average behaviour of priority queueing, i.e., we assume Poisson arrivals and only look at the single-node case. Note that we manly focus on the worst case behaviour and use the average behaviour analysis in order to assess our worst case results on priority queueing.

Note that using Little's formula we can also calculate the average number of packets in class i, $i = 1, \ldots, n$:

$$E(q_i) = \lambda_i E(W_i) = \frac{\lambda_i E(T_0)}{\left(1 - \sum_{j=1}^{i} \rho_j\right)\left(1 - \sum_{j=1}^{i-1} \rho_j\right)} \tag{4}$$

3 Priority Queueing: Worst Case Analysis

In this section, after giving some background information on network calculus and its notation (largely following [1]), we derive basic properties of non-preemptive priority queueing based on network calculus.

3.1 Background on Network Calculus

Network calculus is a tool to analyse flow control problems in networks mainly from a worst case perspective. In particular, it is able to abstract from particular traffic regulation and scheduling schemes and thus allows to arrive at very general results. In our case, it is the framework to derive deteministic guarantees on throughput, delay, and loss-freeness for packet networks that internally operate with non-preemptive priority queueing and which input is constrained by the use of traffic regulation schemes as for example token buckets.

Network calculus could also be interpreted as a system theory for *deterministic* queueing systems. Mathematically it is based on min-plus algebra, a so-called topical algebra. In contrast to queueing theory, network calculus is concerned with worst case instead of average or equilibrium behaviour and therefore does not deal with arrival and departure processes themselves but with bounding processes called arrival and service curves. In the following, we give some basic definitions and notations before we summarize the basic results that network calculus provides.

Definition: Input Function

Call $R(t)$ the input function of an arrival process if it counts the number of bits in interval [0,*t*], in particular $R(0) = 0$. R is wide-sense increasing, i.e., $R(t_1) \le R(t_2)$ if $t_1 \le t_2$.

Definition: Output Function

Call $R^o(t)$ the output function of a system which is offered R as an input function (see Figure 2). $R^o(t)$ counts the number of bits that have left S in interval [0,t], in particular $R^o(0) = 0$. R^o is wide-sense increasing, i.e., $R^o(t_1) \le R^o(t_2)$ if $t_1 \le t_2$.

Definition: Min-Plus Convolution

Let f, g be wide-sense increasing function with $f(0) = g(0) = 0$. The convolution under min-plus algebra for these two functions is defined as $(f \otimes g)(t) = inf_{0 \le s \le t}\{f(t-s) + g(s)\}$.

Under these prerequisites we can now define the major concepts of network calculus, arrival and service curve, by means of the min-plus convolution.

Definition: Arrival Curve

A wide-sense increasing function α with $\alpha(t) = 0$ for t < 0 is called an arrival curve for an input function R if $R \le R \otimes \alpha$. We also say R is α-smooth or R is constrained by α.

Figure 2: Input and Output Function. The system *S* here could be a single buffer, a network node or a complete network.

Definition: Service Curve

Consider a system *S* and a flow through *S* with *R* and R^o. *S* offers a service curve β to the flow if β is wide-sense increasing and $R^o \ge R \otimes \beta$.

Based on the concepts of arrival service curve it is now possible to capture the major worst case properties for data flows: maximum delay and maximum backlog. These are stated in the following theorems.

Theorem 3: Backlog Bound

Assume a flow $R(t)$ constrained by arrival curve α traverses a system *S* that offers a service curve β. The backlog *x(t)* for all *t* satisfies:

$$x(t) \le sup_{s \ge 0}\{\alpha(s) - \beta(s)\} = v(\alpha, \beta). \quad (5)$$

$v(\alpha, \beta)$ is also often called the vertical deviation between α and β.

Theorem 4: Delay Bound

Assume a flow $R(t)$ constrained by arrival curve α traverses a system *S* that offers a service curve β. The virtual delay $d(t)$ for all *t* satisfies:

$$d(t) \le sup_{s \ge 0}\{inf\{\tau \ge 0 \,|\, \alpha(s) \le \beta(s+\tau)\}\} = h(\alpha, \beta) \quad (6)$$

$h(\alpha, \beta)$ is also often called the horizontal deviation between α and β.

A typical example of an arrival curve is given by

$$\gamma_{r,b}(t) = rt + b \quad (7)$$

which results from using the prominent token bucket algorithm as traffic regulation mechanism.

A typical example of a service curve is given by

$$\beta_{R,T}(t) = R(t-T)^+ \quad (8)$$

where the notation $(x)^+$ stands for *x* if $x \ge 0$ and 0 otherwise. This also called a rate-latency service curve and results from the use of many popular schedulers many of which can be generalized in the class of guaranteed rate or latency rate schedulers [6, 7]. An example scheduling algorithm that offers a rate-latency service curve is Packet-by-Packet Generalized Processor Sharing [8].

3.2 Non-Preemptive Priority Queueing under General Arrival Curves

Now we use network calculus to analyse non-preemptive priority queueing for a given number of classes *n* and under the assumption that the input of each class *i* is constrained by α_i for $i = 1, \ldots, n$. To have a constraint on the input function for the last class is often not necessary since with the lowest priority there are often no guarantees associated. However, to demand guarantees for the lowest priority, of course, is the more general assumption therefore we follow this assumption.

Service Curve for Non-Preemptive Priority Queueing

First, we derive the respective service curves for each class under non-preemptive priority queueing. The following theorem states the interesting result that service curves of lower priority classes are dependent on the arrival curves of higher priority classes.

Theorem 5: Let *C* be the overall capacity of the system. Let α_i be the arrival curve for input to class *i*. The service curve β_i^P for class *i* is given by

$$\beta_i^P(t) = \left(Ct - \sum_{j=1}^{i-1} \alpha_j(t) - L_i \right)^+ \quad (9)$$

for $i = 1, \ldots, n$, where $L_i = max_{i+1 \le j \le n} \left\{ l_j^{max} \right\}$

Here l_j^{max} is the maximum size of a packet in class *j*.

Proof:

Let $R_i(t)$, $R_i^o(t)$ be the input and output function for traffic from class *i* for $i = 1, \ldots, n$. Now, let *s* be the start of the last busy period due to traffic from classes 1 to *i* before a fixed time *t*. Then the amount of service given to traffic from class *i* is lower bounded by the server output minus the service given to higher traffic classes and the maximum packet size for lower traffic classes for which a single packet might just have started service before *s*. The server output in interval [*s*,*t*] is given by $C(t-s)$ by definition of a busy period. Thus we have

$$R_i^o(t) - R_i^o(s) \ge C(t-s) - \sum_{j=1}^{i-1} (R_j^o(t) - R_j^o(s)) - L_i \quad (10)$$

Due to s being the start of a busy period for traffic from classes $j = 1, \ldots, i$ we also have $R_j^o(s) = R_j(s)$. Thus

$$\begin{aligned} R_j^o(t) - R_j^o(s) &= R_j^o(t) - R_j(s) \\ &\le R_j(t) - R_j(s) \le \alpha_j(t-s) \end{aligned} \quad (11)$$

That means we can make use of the arrival curve constraint in (10). Note that the bound in (11) is tight because at time *t* input and output function could well be equal and of course traffic could be greedy. Introducing (11) in (10) we obtain

$$R_i^o(t) - R_i^o(s) \ge C(t-s) - \sum_{j=1}^{i-1} \alpha_j(t-s) - L_i \quad (12)$$

Since R_j^o is wide-sense increasing we obtain

$$\begin{aligned} R_i^o(t) &\ge R_i^o(s) + \left(C(t-s) - \sum_{j=1}^{i-1} \alpha_j(t-s) - L_i \right)^+ \\ &= R_i(s) + \beta_i^P(t-s) \\ &\ge inf_{0 \le s \le t} \left\{ R_i(s) + \beta_i^P(t-s) \right\} \\ &= (R_i \otimes \beta_i^P)(t) \end{aligned} \quad (13)$$

Thus, indeed, non-preemptive priority queueing offers β_i^P as a service curve towards traffic from class i.

■

The theorem is the basis for all subsequent findings of the paper. Moreover, it contains a very constructive result:

There is a quantifiable dependency of lower priority classes' service curves on the arrival curves of higher priority classes.

There are several ways to use this in practical networking problems as for example in flow or packet admission control for class-based networks. In particular for flow admission control, it allows to dimension aggregate arrival curves for each class such that certain delay targets for each class are achieved. New flow requests for a class can then be checked by the admission control against whether the sum of arrival curves of already admitted flows and the new flow is still below the aggregate arrival curve which is necessary to achieve the delay target.

In contrast to the standard way of network calculus of being given a certain arrival curve and then calculating the service curve such that given properties like a certain maximum delay are achieved, we kind of reverse the reasoning by calculating the service curve for general arrival curves and then choose the arrival curve such that the service curve of each class leads to certain properties within that class.

While the dependency between arrival and service curve might at first sight give an uneasy feeling about possible circular dependencies due to the fact that service curves operate on arrival curves to calculate quantities like maximum delay or maximum backlog it must be realized that they are not dependent on the actual arrival processes but can be treated as a given.

3.3 Non-Preemptive Priority Queueing with Token Buckets as Arrival Curve

In this section, we now assume a particular arrival curve, the popular token bucket [9]. Under this assumption we can concretise the service curve for general arrival curves and can then derive bounds on maximum backlog and delay per class.

Service Curve

First we apply Theorem 5 to the special case of token buckets as arrival curves for the different classes in order to derive the service curve for non-preemptive priority queueing. Theorem 6 states the result.

Theorem 6: (Service Curve under Token Buckets)

Let $\alpha_j = \gamma_{r_j, b_j}$ be the arrival curve for traffic class *j*, $j = 1, \ldots, n$, i.e., each traffic class is constrained by a token bucket (each with its own parameters). The service curve for class *i* under non-preemptive priority queueing is then given by

$$\beta_i^P = \beta_{R^P, T^P} \text{ with} \quad (14)$$

$$R_i^P = C - \sum_{j=1}^{i-1} r_j \text{ and } T_i^P = \left(\sum_{j=1}^{i-1} b_j + L_i \right) \Big/ \left(C - \sum_{j=1}^{i-1} r_j \right)$$

That means the service curve is of the rate-latency type.

Proof:

The theorem is a consequence of Theorem 5 and the definition of the rate-latency service curve in (8):

$$\begin{aligned} \beta_i^P(t) &= \left(Ct - \sum_{j=1}^{i-1} \alpha_j(t) - L_i \right)^+ = \left(Ct - \sum_{j=1}^{i-1} (r_j t + b_j) - L_i \right)^+ \\ &= \left(C - \sum_{j=1}^{i-1} r_j \right) \left(t - \left(\sum_{j=1}^{i-1} b_j + L_i \right) \Big/ \left(C - \sum_{j=1}^{i-1} r_j \right) \right)^+ \\ &= \beta_{R_i^P, T_i^P} \end{aligned}$$

■

Delay and Backlog

Using the service curve for non-preemptive priority queueing we can now derive the worst case delay bound as well as the maximum backlog bound for each traffic class using the fundamental network calculus results from Theorem 3 and Theorem 4.

The backlog bound is given by the following theorem.

Theorem 7: Per-Class Backlog Bound under Token Buckets

Let $\alpha_j = \gamma_{r_j, b_j}$ be the arrival curve for traffic class j, $j = 1, \ldots, n$, i.e., each traffic class is constrained by a token bucket (each with its own parameters). For stability we further assume that $C \geq \sum_{i=1}^{n} r_i$.

The maximum backlog per traffic class i is bounded by the vertical deviation between the arrival curve to class i, γ_{r_i, b_i}, and its service curve, β_i^P

$$v(\gamma_{r_i, b_i}, \beta_i^P) = r_i \times \left(\sum_{j=1}^{i-1} b_j + L_i \right) \Big/ \left(C - \sum_{j=1}^{i-1} r_j \right) + b_i \qquad (15)$$

Proof:

Due to the stability condition we have $C - \sum_{j=1}^{i-1} r_j \geq r_i$, i.e., the slope of the service curve is higher than that of the arrival curve for class i. That means the maximum vertical deviation is taken on at the latency of the service curve for class i, because the service curve comes closer and closer to the arrival curve once the service is "started", i.e.,

$$\begin{aligned} v(\gamma_{r_i, b_i}, \beta_i^P) &= sup_{s \geq 0} \left\{ \gamma_{r_i, b_i}(s) - \beta_i^P(s) \right\} \\ &= \gamma_{r_i, b_i}(T_i^P) - \beta_i^P(T_i^P) \\ &= \gamma_{r_i, b_i}(T_i^P) \\ &= r_i \times \left(\sum_{j=1}^{i-1} b_j + L_i \right) \Big/ \left(C - \sum_{j=1}^{i-1} r_j \right) + b_i \end{aligned}$$

■

Next we derive the per-class maximum delay bound under the same assumptions in Theorem 8.

Theorem 8: Per-Class Delay Bound under Token Buckets

Let $\alpha_j = \gamma_{r_j, b_j}$ be the arrival curve for traffic class j, $j = 1, \ldots, n$, i.e., each traffic class is constrained by a token bucket (each with its own parameters). For stability we further assume that $C \geq \sum_{i=1}^{n} r_i$.

The maximum delay per traffic class i is bounded by the horizontal deviation between the arrival curve to class i, γ_{r_i, b_i}, and its service curve, β_i^P

$$h(\gamma_{r_i, b_i}, \beta_i^P) = \left(\sum_{j=1}^{i} b_j + L_i \right) \Big/ \left(C - \sum_{j=1}^{i-1} r_j \right) \qquad (16)$$

Proof:

Following the same arguments as in the proof of Theorem 7, it is clear that the maximum horizontal deviation is taken on at the origin, i.e.,

$$\begin{aligned} h(\gamma_{r_i, b_i}, \beta_i^P) &= sup_{s \geq 0} \left\{ inf \left\{ \tau \geq 0 \,|\, \gamma_{r_i, b_i}(s) \leq \beta_i^P(s + \tau) \right\} \right\} \\ &= inf \left\{ \tau \geq 0 \,|\, \gamma_{r_i, b_i}(0) \leq \beta_i^P(\tau) \right\} \\ &= inf \left\{ \tau \geq 0 \,|\, b_i \leq - \sum_{j=1}^{i-1} b_j - L_i + \left(C - \sum_{j=1}^{i-1} r_j \right) \tau \right\} \\ &= inf \left\{ \tau \geq 0 \,|\, \left(\sum_{j=1}^{i} b_j + L_i \right) \Big/ \left(C - \sum_{j=1}^{i-1} r_j \right) \leq \tau \right\} \\ &= \left(\sum_{j=1}^{i} b_j + L_i \right) \Big/ \left(C - \sum_{j=1}^{i-1} r_j \right) \end{aligned}$$

■

So we can now compute besides the known results for average behaviour also the worst case properties for non-preemptive priority queueing if we assume each traffic class conforms to a token bucket (respectively make it conform to it by either using admission control at ingress to the network or drop packets according to the token bucket).

4 Comparing Average and Worst Case Behaviour of Priority-Based Class-of-Service networks

In this section, we perform some numerical investigations on the formulae derived above, both for the average as well as for the worst case behaviour of non-preemptive priority queueing. When comparing average and worst-case behaviour we need to keep in mind that the assumptions are quite different. For the average case we assume Poisson arrivals whereas for the worst case we have no restricting assumptions on the arrival process for a given traffic class other than that it is bounded by its class-specific token bucket.

At first, we provide a comprehensive numerical example of a non-preemptive priority queueing system with 8 traffic classes,

before we then investigate the influence of different parameters more closely.

For the sake of simplicity we assume in the following investigations that the maximum packet size is the same over all classes. Since traffic for these classes is aggregated traffic of possibly all kinds this is also a realistic assumption. In particular, we set the maximum packet size for all classes to 1500 bytes. Furthermore we assume for the distribution of packet sizes that their average is 420 and their standard deviation is about 521 which corresponds to up-to-date measurements from [10].

4.1 Comprehensive Numerical Example

We assume 8 traffic classes each with 10% load of the overall server capacity which is assumed to be 100 Mbps. For each class' token bucket size we assume it to be 20% of the token bucket rate. This accounts for (infinitely fast) bursts of a volume corresponding to 200 ms of average activity in the class which seemed reasonable to us.

In Table 1 the different delay and queue size values for all classes are given.

Class	Av. Delay	W. C. Delay	Av. Queue Size	W. C. Queue Size
1	0.02	20	0.06	500
2	0.03	45	0.08	556
3	0.04	75	0.10	625
4	0.05	114	0.13	715
5	0.07	167	0.18	834
6	0.11	240	0.27	1001
7	0.18	350	0.45	1251
8	0.36	534	0.91	1668

Table 1: Average and worst case delays (in ms) and queue sizes (in packets).

In addition, in Figure 3 and Figure 4 we have depicted the average and worst case delay for each class.

As can be seen, the worst case behaviour is about 3 orders of magnitude above the average case behaviour for delay and about 4 orders of magnitude above for queue size. While for both, average and worst case behaviour, we can observe good differentiation between the classes we can see a more pronounced and balanced differentiation with respect to worst case delay in particular for high priority classes.

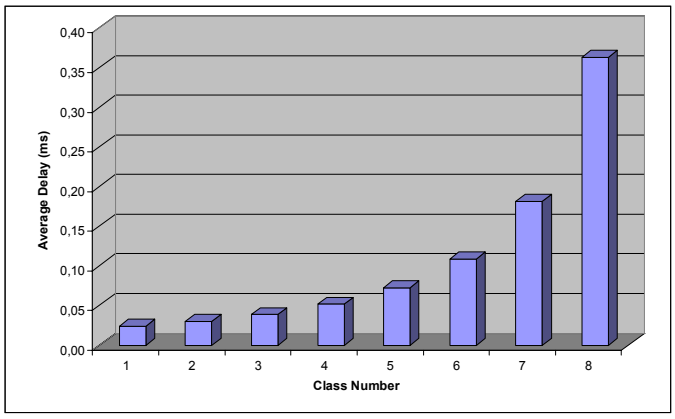

Figure 3: Average Delay for Different Classes.

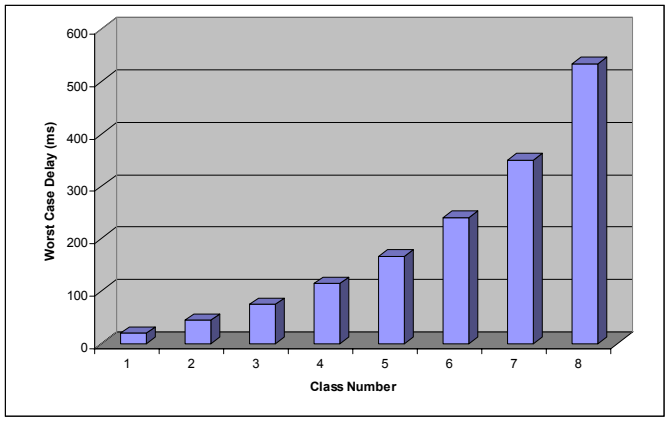

Figure 4: Worst Case Delay for Different Classes.

In the following we examine the influence of different parameters like the ratio of high priority traffic classes to low priority ones, the burstiness of high priority traffic, the server capacity, and the total load. For these it is for presentational purposes more convenient to look only at 2 classes instead of 8 since the basic effects can be more easily identified by contrasting high priority against low priority traffic. Furthermore, we focus on delay when comparing average and worst case behaviour in the following.

4.2 Ratio of High Priority Traffic

In this experiment, the ratio between high and low priority traffic is varied while everything else is kept fixed, in particular the total load in the system is kept at 80%.

Figure 5 shows the average delay for an increasing load from high priority traffic while Figure 6 shows the worst case delays.

While again being 3 orders of magnitude apart from each other we can observe a very similar behaviour: for increasing high priority traffic load the low priority traffic is punished harder once a certain load of high priority traffic is exceeded. This certainly calls for keeping the load from high priority traffic low since other traffic is otherwise suffering considerably.

4.3 Burstiness

Again we assume two classes of traffic, yet now we keep the ratio between high and low priority traffic fixed (at 20:60%) and vary the burst size for the high priority class. Of course, this does not influence the formulae for average behaviour, so we only examine the effect on the worst case delay.

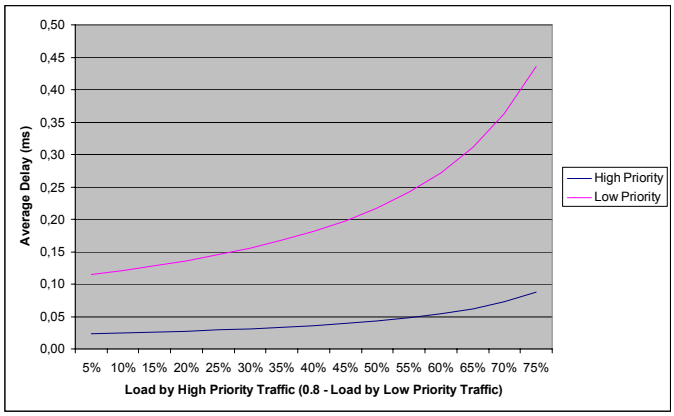

Figure 5: Average Delay for Different Ratios of High and Low Priority Traffic.

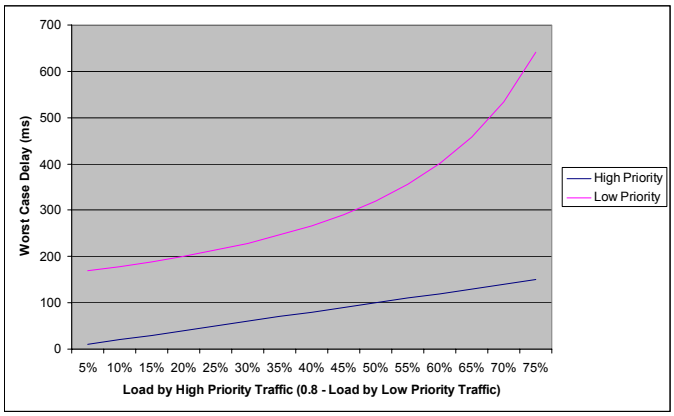

Figure 6: Worst Case Delay for Different Ratios of High and Low Priority Traffic.

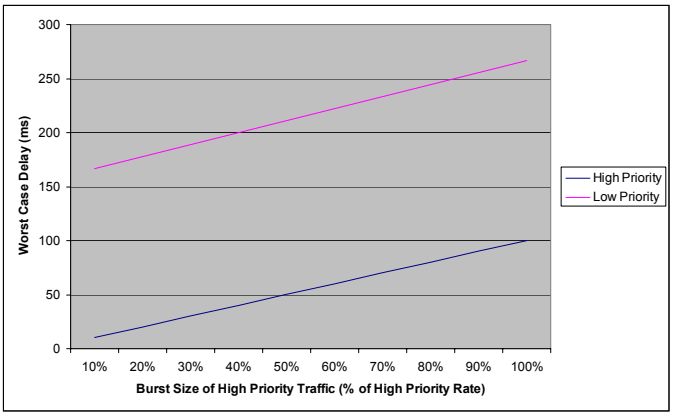

Figure 7: Worst Case Delay for Different Burst Sizes of High Priority Traffic.

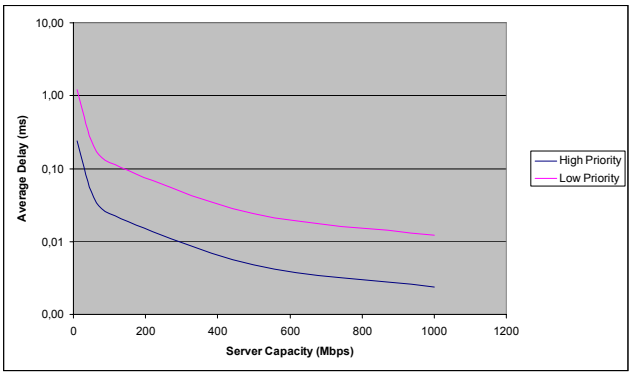

Figure 8: Average Delay for Different Server Capacities.

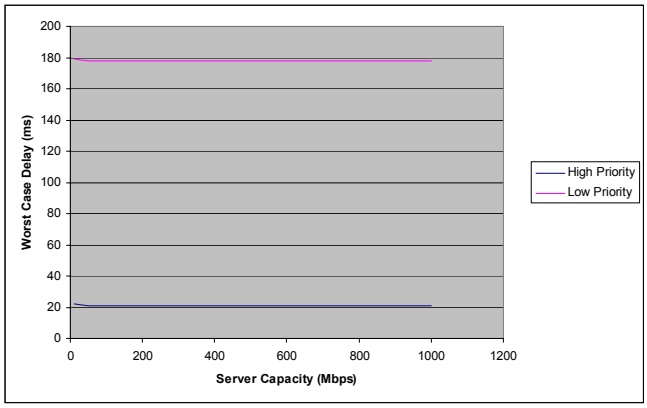

Figure 9: Worst Case Delay for Different Server Capacities.

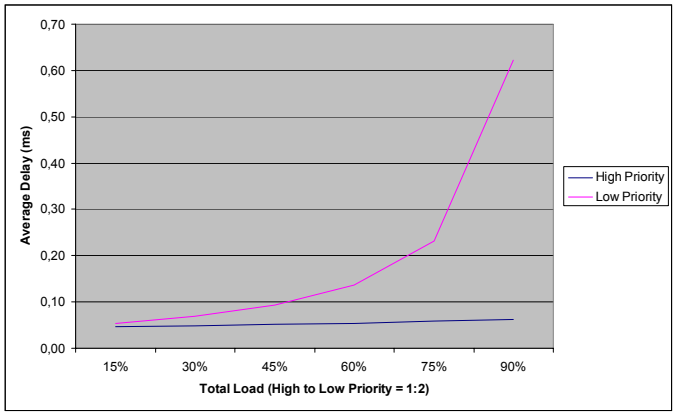

Figure 10: Average Delay for Different Total Load.

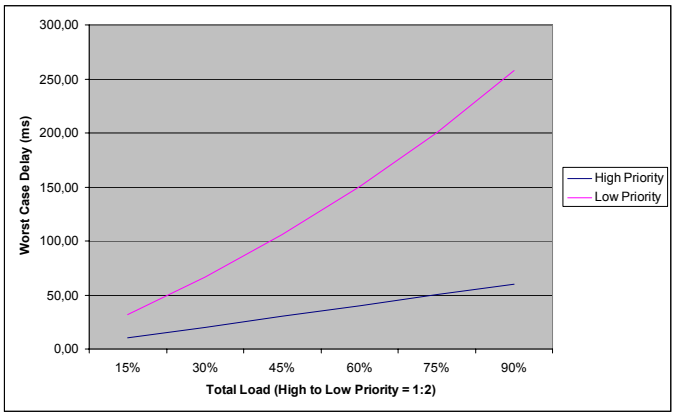

Figure 11: Worst Case Delay for Different Total Load.

Figure 7 shows the results. We observe an almost linear dependency on the burst size of the high priority traffic class and both classes exhibit the same slope.

4.4 Server Capacity

Next, we want to investigate the influence of the server capacity. Figure 8 shows the average delay for varying server capacities, while Figure 9 shows the worst case delay.

As can be seen, while increasing the server capacity has a positive influence on average delays it has no effect on the worst case delays which remain almost constant. This means that the difference between worst case and average delays very much depends on the server capacity and is growing with increasing server capacity. This phenomenon should lead to possibly large deviations between delays, i.e., large jitter in high capacity systems – a reason to have more stringent control over the worst case delay for traffic that is sensitive to jitter.

4.5 Total Load

In this experiment we change the total load offered to the system while keeping the ratio between high and low priority traffic fixed (at 1:2), i.e., there is always twice the amount of low priority traffic in the system.

Figure 10 and Figure 11 show the results for average and worst case delays.

For both, average and worst case behaviour, the curves show a higher dependency of low priority traffic on the total load than for high priority traffic. Furthermore, we can observe the typical steep increase for average delay when load approaches capacity whereas for worst case delays the system response is almost linear.

5 Related Work

In this section, we review some related work. Here, we focus on directly related work whereas all the fundamental research results we built upon are referenced at the locations where we made use of them.

Closest to our work is a measurement based analysis of average delays in priority queueing by Ferrari et al. [11]. They build up a small testbed consisting of Cisco routers and employ their priority queueing scheme. Their experimental results are absolutely consistent with the average queueing behaviour results which are predicted by an M/G/1 queueing system (recapitulated in Section 2). Of course, due to the experimental behaviour they cannot provably make any statements about worst case delays for priority queueing.

There is a large body of queueing theoretical work concerned with priority queueing which derives more advanced results than those presented in Section 2. However, despite very thorough treatment especially with respect to flexibilizing the Poisson arrival assumption a "quantum leap" has not been achieved. A good overview can be found in [5]. In particular, it is discussed that networks of priority queueing systems do not have a product form solution which is due to the fact that the priority scheduling introduces complex dependencies between packet arrival processes. Furthermore, there is some work on variants of basic priority queueing as for example priority queueing with priority jumps, where after a certain waiting time packets are elevated towards the next higher priority (see for example [12]). None of these works takes regulated traffic into account and thus is not able to investigate worst case bounds.

6 Conclusions & Outlook

We analytically investigated the behaviour of non-preemptive priority queueing systems. These have (re)gained importance in packet networks due to schemes like DiffServ that allow for simple priority queueing as cheap implementation for offering performance guarantees in so-called class of service networks.

Based on network calculus, we derived so far unknown bounds on delay and backlog per traffic class in non-preemptive priority queueing systems and contrasted them against known average case results from classical queueing theory.

Practical implications from this work, apart from the fundamental insights from the comparison of average and worst case behaviour, are in network performance control. As briefly discussed the results can be applied for admission control purposes to achieve certain delay targets in each traffic class. Furthermore, by appropriately sizing token buckets it might be possible to emulate some more sophisticated and therefore more costly scheduling disciplines by using simple and available priority queueing in routers.

For future work we have planned to do some practical investigations of priority queueing in order to verify our analytical results, in particular we have already implemented priority queueing for FreeBSD and with the aid of the ALTQ traffic control framework [13] it should be relatively simple to perform some lab experiments. As we have seen that the worst case bounds are several orders of magnitude higher than what one would expect from an average behaviour analysis we see a potential for better arrival curves than simple token buckets which would allow to derive more stringent bounds on backlog and delay.

References

[1] J.-Y. Le Boudec and P. Thiran. *Network Calculus - A Theory of Deterministic Queueing Systems for the Internet*. Springer, Lecture Notes in Computer Science, LNCS 2050, 2001.

[2] A. Mankin, F. Baker, R. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang. Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement - Some Guidelines on Deployment. Informational RFC 2208, September 1997.

[3] L. Kleinrock. *Queueing Systems – Theory*. Wiley-Interscience, New York, vol.1, 1975.

[4] Cisco Systems: Configuring Priority Queueing, 2000. Available at http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/qos_%cqcprt2qcdpq.pdf

[5] T. G. Robertazzi. *Computer Networks and Systems*. Springer, 3rd Edition, 2000.

[6] P. Goyal, S. S. Lam, and H. Vin. Determining End-to-End Delay Bounds in Heterogeneous Networks. In *Network and Operating System Support for Digital Audio and Video, 5th International Workshop, NOSSDAV'95, Durham, New Hampshire, USA*. Springer LNCS 1018, April 1995.

[7] D. Stiliadis and A. Varma. Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms. *IEEE/ACM Transactions on Networking*, 6(5):611–624, October 1998.

[8] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[9] J. S. Turner. New Directions in Communications. *IEEE Communications Magazine*, 24(10):8–15, October 1986.

[10] Cooperative Association for Internet Data Analysis (CAIDA). Packet Size Distributions, 2000. Available at http://www.caida.org/analysis/AIX/plen_hist/.

[11] T. Ferrari, G. Pau, and C. Raffaelli. Measurement based analysis of delay in priority queuing. In *Proceedings of IEEE Global Telecommunications Conference 2001 (GLOBECOM'01)*, pages 1834–1840. IEEE, November 2001.

[12] Y. Lim and J. Kobza. Analysis of a Delay-dependent Priority Discipline in a Multi-Class Traffic Packet Switching Node. In *Proceedings of the 7th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'88)*, pages 889–898. IEEE, April 1988.

[13] K. Cho. A Framework for Alternate Queueing: Towards Traffic Management by PC-UNIX Based Routers. In *Proceedings of USENIX 1998 Annual Technical Conference, New Orleans, USA*, June 1998.