8

Experimental Comparison of Quality of Service Systems

In the previous chapter we investigated with the help of analytical methods the potential benefit of a Quality of Service (QoS) system over a plain Best-effort (BE) system. In the analytical approaches, a single bottleneck was assumed. Also, the QoS systems are modelled in an abstract way (e.g. with strict priority queueing in Section 7.2). To work out the differences between real QoS systems (e.g. Intserv and different Diffserv systems) that use more sophisticated admission control and scheduling algorithms, actual implementations of the systems should be used. We do so in this chapter, using packet-level, event-based simulations. The following *QoS systems* based on the main Internet Engineering task Force (IETF) architectures were implemented and used for these simulations:

- **Integrated Services (Intserv)**
  The Intserv QoS architecture was presented and discussed in Section 6.2.2. Intserv guaranteed service (GS) allows deterministic loss and delay guarantees. In that sense, it is the 'strongest' service we are investigating.
  The Stateless Core (SCORE) architecture with Dynamic Packet State (DPS) (see Section 6.2.3) can be used to offer a scalable GS; it therefore leads to results very similar to those of Intserv and can be evaluated on the basis of the Intserv results in this chapter.
- **'Standard' Differentiated Services (Diffserv)**
  The Diffserv QoS architecture was discussed in Section 6.2.4. We name the Diffserv systems that use the expedited and Assured Forwarding (AF) behaviour from RFC 2597 (see Heinanen *et al*. (1999)) and RFC 2598 (see Jacobson *et al*. (1999)) 'standard' Diffserv.
  For resource management and admission control in the Diffserv systems, we consider three different types of bandwidth brokers (BBs):
  - Centralised Bandwidth Broker
    The centralised BB has full knowledge of the routing by keeping track of the paths that the different flows take through the network. We designed and implemented a very sophisticated centralised BB that can also guarantee the delay bounds for

The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design
Oliver Heckmann © 2006 John Wiley & Sons, Ltd. ISBN: 0-470-01293-5

admitted flows, thus mimicking the Intserv GS behaviour while still maintaining the low Diffserv per-class scheduling complexity. To increase the efficiency of the Diffserv system, we allow relaxing the service guarantees to stochastic guarantees and investigate overbooking of the service classes.

  - Decentralised Bandwidth Broker
    The decentralised BB is a simplified version of the central one. It uses a decentralised admission control algorithm that is based on information *locally* available at the ingress node. Thus, it is easier to implement and maintain than the centralised broker, but it is less efficient. In addition, it cannot give delay bound guarantees along a path.
  - No BB/No Admission Control
    A Diffserv network can also be operated without admission control if it is well dimensioned and relying on mid-term and long-term traffic and network engineering. These methods are discussed in Part IV of this book. In our experiments, a system without BB and other admission control mechanism is therefore included as reference.

- Olympic Differentiated Services
  Contrary to the 'standard' Diffserv systems, Olympic Diffserv systems are based on a very low number of Per Hop Behaviours (PHBs) (in our case three) that are differentiated by strict priority queueing. The three services built on these PHBs are called *gold, silver* and *bronze*, hence the name 'Olympic'[1].
  We use the same BB types that we use for standard Diffserv with adaptation to the Olympic service scheme.
- Overprovisioned Best-Effort
  As the QoS of a system can be expected to be satisfying if it is dimensioned well enough, we use plain BE networks that are overprovisioned with different overprovisioning factors (similar to the previous chapter) as reference. This allows us to determine overprovisioning factors and compare the results with the analytical results of the previous chapter.

As defined in Section 6.2, a *QoS system* consists of the *QoS architecture* that describes the general technical foundation of the QoS system and the *QoS strategy* that determines how an Internet Network Service Provider (INSP) exploits the technical features offered by the chosen architecture. The strategy includes the configuration of the architecture.

In the experiments of this chapter, we show how different QoS systems perform when facing a certain traffic mix and a certain network topology. The *performance* is evaluated by technical criteria like the dropping probability or the throughput and by application-specific utility functions. Utility functions are important because different applications of the traffic mix have different QoS requirements. For TCP-based file transfer applications, the utility largely depends on the overall throughput as they can recover from losses and delay variations (jitter) to a certain extent. For multimedia applications that are – at the timescale of the experiment – not rate adaptive, the loss and the delay will typically be more important. Utility functions are therefore necessary to evaluate the benefit a *user* has if a certain QoS system is used.

We developed and implemented an *experimentation environment* on top of the packet-level network simulator NS2 (see NS2 (2004)). NS2 is commonly used for QoS

[1] Please note that the term 'Olympic' in the context of Diffserv services is in other works sometimes used for a cascade of AF services, see Heinanen *et al.* (1999).

experiments. For an experiment, a certain traffic mix plus a network topology is used as input. The experiment is conducted in several steps, in each step a different QoS system is used and a complete packet-level simulation is performed. All steps use exactly the same traffic, allowing us to directly compare their results.

We consider different *traffic* mixes that consist of different types of traffic, for example, Constant bit-rate (CBR) and Variable bit-rate (VBR) traffic. For our experiments, we considered using traffic sessions or direct individual flows as traffic input. A *session* consists of a number of closely related and interdependent flows. For example, a World Wide Web (WWW) session could represent a series of webpages[2] a user is reading with short variable reading times after each page is downloaded. It can be represented as a series of Transmission Control Protocol (TCP)/Hypertext Transfer Protocol (HTTP) flows, each transferring a potentially different amount of data. For this example, in an experiment that uses traffic session semantic as traffic input, a flow would not start until the previous flow of the same session is finished plus possibly a certain variable 'reading' time. Because the starting times of flows depend on the network condition, it is not possible to generate the traffic flows off-line. If traffic is modelled on the session layer, the application behaviour can be modelled more realistically. The traffic emulator[3] GenSyn (see Heegaard (2000)) is an example for a session-based traffic emulator. It models user behaviour with different state machines for different application types (WWW, File Transfer Protocol (FTP), video streaming, voice over Internet Protocol (VoIP), etc).

Alternatively, the individual *flows* could be specified directly and used as traffic input. They can be generated off-line from session models. However, as the network conditions (loss rates, delays, etc.) are not known in advance, certain aspects of the application/user behaviour will then not be modelled as nicely as when using sessions as input with online flow generation.

For the purpose of our experiments, however, using flows instead of sessions has one crucial advantage in that it allows a direct comparison: If flows are specified and used as input, the amount of load 'offered' to the network remains constant in each step of an experiment – that means for each evaluated QoS system. If sessions would be used where a second flow is only started once the first is finished, a QoS system offering poor throughput performance for the first flow would in fact be 'rewarded' with less traffic as the second flow would start delayed or not at all. This would not only seem unfair, it also makes the direct comparison of technical parameters like loss and throughput impossible because large variations in the network load would occur. The overall evaluation would then only be possible based on 'session' utility functions that evaluate the overall utility of a session. We want to avoid this for the following reasons: Utility functions that evaluate the performance of a single flow can be based directly on the technical parameters like loss and delay of the flow. Few assumptions have to be made for these 'flow' utility functions (see Section 8.2.3.2). For the higher-level 'session' utility functions, however, more assumptions are necessary and therefore more subjectivity would be introduced.

Because of these reasons, we chose to use the session concept for off-line flow generation and use flows as input for the simulations and as a basis for the evaluation; the

[2] Each consisting of a Hypertext Markup Language (HTML) file plus possibly some graphics.

[3] We use the term traffic emulator for software/hardware that generates artificial traffic for a physical network and traffic simulator for software that generates traffic for simulations.

evaluation can thus be based on flow utility functions backed by the technical parameters as 'hard' facts.

We start with describing the technical details of the admission control mechanisms and implementation details for the QoS systems. Then, Section 8.2 sheds light on the experiment set-up. A fairly sophisticated experimentation environment is used to run the experiment with the same traffic flows using different QoS systems; this approach allows us to directly compare the results obtained from the simulations. Section 8.2 also describes the experimentation and evaluation parameters, for example, the chosen topologies, traffic mixes and utility functions. Finally, the different *experiments* and their results are presented as follows.

- In the first set of experiments (Section 8.3), the QoS systems that can give loss and delay-bound guarantees are compared: the Intserv system using per-flow scheduling and the Diffserv systems with the centralised BB and per-class scheduling.
  The experiments shed light on the trade-off between additional data-path complexity and more efficient resource allocations. In addition, it sheds light on the overbooking potential of the Expedited Forwarding (EF) service class when using the central BB for stochastic service guarantees.
- For Diffserv systems, a decentralised admission control decision promises less computational complexity and communication overhead. However, as it has no control of the interior of the network, the risk of service disruptions (packet drops, delay-bound violations) increases. This effect is investigated in Section 8.4.
- In the direct comparison experiments of Section 8.5, the QoS systems that performed best in the previous experiments are pitted against each other directly. Different traffic mixes and topologies are evaluated. These experiments display and quantify the individual strengths and weaknesses of the QoS systems. In addition, we determine the range of overprovisioning factors for the QoS systems and compare them with the analytical results of the previous chapter.

This chapter concludes with a summary and conclusion.

8.1 QoS Systems

First, we describe the implementations of the admission control mechanisms for the QoS systems. While the design space of admission control mechanisms for Intserv is limited by the according Request for Comments (RFCs), there are almost no restrictions for admission control in Diffserv. The central BB we specify below for Diffserv is able to give very strong guarantees on one side and allows for overbooking and efficient network usage on the other side. The admission control algorithms introduced in this section were implemented for the experiments of this chapter and those in Chapter 13 (Section 13.1).

8.1.1 Intserv/RSVP QoS Systems

For our experiments, we use the traditional Intserv/Resource Reservation Protocol (RSVP) QoS architecture as discussed in Section 6.2.2. We use Intserv/RSVP as reference for the 'strongest' service, the GS, as it is a deterministic service with per-flow guarantees; therefore, we focus on GS within the Intserv/RSVP architecture. The controlled load

service is not evaluated, as it does not promise significant advantages over the various Diffserv services.

As it is also possible to provide the same GS service guarantees with a core stateless architecture, the performance of a core stateless architecture like DPS (see Section 6.2.3) can be evaluated on the basis of our results for Intserv/RSVP.

8.1.1.1 Admission Control

The Intserv/RSVP admission control is to a large extent specified in RFCs (e.g. RFC 2212 for GS). In terms of the classification of Section 6.6, it is a hop-by-hop network-based admission control system with deterministic guarantees based on worst-case descriptions of the flow and networking behaviour. The traffic description uses a TSpec; the allocated network resources are buffer and bandwidth. The basic granularity is fine (microflows) although approaches exist for aggregation of flows. Intserv/RSVP has explicit support for multicast. Our implementation is non-preemptive, does not support reservation in advance and no end-time is specified by a flow during the reservation, as these points are also not mentioned in RFC 2212.

The Intserv per-flow admission control is used for GS flows based on the token bucket descriptor (r_f, b_f) of the arrival curve[4]. The Intserv/RSVP reservation process allows the explicit declaration of a queueing delay bound; it influences the amount of resources that have to be allocated for a flow. Our admission control manages two resources for an outgoing link l at a router: the available bandwidth bw_l and the buffer space bf_l. For all our analysed QoS systems, these resources were set to equal values to allow a fair comparison.

For a GS flow f with a queueing delay bound d_f^q, the admission control has to allocate the rate R_f and the buffer space B_f for each link l along the path P (see Section 6.2.2.4):

$$R_f = \max \left\{ \frac{b_f + \sum_{l \in P} C_{fl}}{d_f^q - \sum_{l \in P} D_l}, \; r_f \right\} \tag{8.1}$$

$$B_f = b_f + \sum_{l \in P} C_{fl} + \sum_{l \in P} D_l \cdot R_f \tag{8.2}$$

We do not need to make use of the slack term S of RFC 2212 (see Shenker *et al.* (1997)). C_{fl} and D_l are the scheduling error terms of flow f on link l. Set ϑ_l contains all other currently accepted and active GS flows passing through link l. A flow f is only admitted if R_f and B_f can be allocated for each link l of the path P and do not exceed a given maximal share α_{GS} of that link's bandwidth bw_l and buffer resources bf_l:

$$R_f + \sum_{g \in \vartheta_l} R_g \leq \alpha_{GS} \cdot bw_l \qquad \forall l \in P \tag{8.3}$$

[4] We simplified the TSpec to a token bucket. A small additional efficiency gain can be achieved by using the TSpec as basis for the admission control algorithm, see Section 6.2.2.4.

$$B_f + \sum_{g \in \vartheta_l} B_g \leq \alpha_{GS} \cdot bf_l \qquad \forall l \in P \tag{8.4}$$

As mentioned above, we do not use the Intserv *Controlled Load* service class (see Wroclawski (1997)). BE flows are not admission controlled at all in the Intserv system.

8.1.1.2 Scheduling

Weighted Fair Queueing (WFQ), (see Demers *et al*. (1989)) is used for the scheduling of the flows. WFQ has the following scheduling error terms

$$C_{fl} = \text{ maximum packet size of flow } f \tag{8.5}$$

$$D_l = \frac{MTU}{bw_l} \tag{8.6}$$

In Intserv, per-flow scheduling is used for all GS flows (contrary to the Diffserv per service class scheduling); the WFQ weight w_{fl} assigned to a GS flow f on link l is

$$w_{fl} = R_f / bw_l \tag{8.7}$$

All BE flows share a single queue that is assigned the remaining weight

$$w_{BEl} = 1 - \sum_{g \in \vartheta_l} R_g / bw_l \tag{8.8}$$

8.1.2 Standard Diffserv QoS Systems

We name the Diffserv approach with EF/AF PHB *'standard' Diffserv*. As described in Section 6.2.4, Diffserv is more of a QoS system framework than an exact specification of a certain QoS system, so there cannot be a real 'standard' Diffserv. However, the EF/AF PHBs are up to now the only PHBs in the standardisation process of the IETF and the ones most commonly found in Diffserv related works, which justifies our choice of name.

In this set-up, we proceed according to the RFCs; they prescribe two PHBs:

- Expedited Forwarding (EF) and
- Assured Forwarding (AF).

The EF PHB is intended for traffic with low delay requirements. We refrain from using all three drop precedences from Heinanen *et al*. (1999) to keep the complexity of the experiments low. Further, preliminary experiments showed that their influence on the results of the entire system is negligible for the purpose of our evaluation.

A key issue is whether and what type of admission control is conducted. We evaluate three different types of 'standard' Diffserv QoS systems that differ in their *admission control bandwidth broker*. A BB is *an entity that manages and configures the network devices of a Diffserv domain and keeps state in terms of how loaded the network is and whether a new flow is admissible.* The three different types are as follows.

- **Centralised (global) Bandwidth Broker**
  Please note that the goal of the BB is to show the 'best-you-can-do' approach; this is why it checks and guarantees the delay bounds for individual flows throughout their complete network path.

The global BB checks the entire path throughout the network before admitting a flow. Consequently, it has to keep state about the routes of the network as well as the load throughout the network. It has to find out which routes the new flow will take through the network and check resource availability along each hop.

Additionally, the global BB keeps track of the resource allocations of the individual flows that make up one forwarding class. This allows the BB to check whether the delay bounds of the flows can be guaranteed as we demonstrate below. As our experiments will show, it is possible to reduce the amount of state of this bandwidth broker on the control path without disrupting the service.

In terms of the classification of Section 6.6, the central BB has a centralised network-based admission control system based on worst-case descriptions of the flow and network behaviour that gives deterministic stochastic guarantees.

- **Decentralised (local) Bandwidth Broker**
  We define a local BB as one that operates on each edge node and checks only whether this edge node has the capacity to admit the flows. This is a low complex operation, not much state has to be kept.

  In terms of the classification of Section 6.6, the decentral BB is also network-based but located at the edge; more specifically at the ingress node. It uses a contingent-based algorithm based on worst-case behaviour. It cannot give better than stochastic guarantees.
- **No Bandwidth Broker and no Admission Control**
  The easiest solution is, of course, to refrain from using a bandwidth broker and admission control and rely on a well-dimensioned network.

8.1.2.1 Centralised Bandwidth Broker

Admission Control We assume that the centralised BB has perfect knowledge of the network state at each point in time: It knows all routes through the Diffserv domain and keeps track of the aggregate bandwidth and buffer allocations of each link. It knows which route a newly arriving flow will take through the Diffserv domain. Such a central bandwidth broker is complex to develop and maintain for a large network but represents the 'best-you-can-do' approach in a Diffserv network.

The knowledge of the Diffserv central BB allows it to also check whether it is possible to guarantee delay bounds for EF flows and in this aspect mimic the service guarantees of Intserv guaranteed service.

Because the individual flows that are merged into a single Diffserv class are not protected against each other inside that class, the resource management in the Diffserv network is less efficient than for Intserv. However, this leads to less complexity on the data path, which usually is more important.

Before a new flow f can be admitted, the BB has to check the availability of bandwidth and buffer space along the path P_f of the flow through the network. In addition, the bandwidth broker has to check whether the delay bound of that flow can be guaranteed or not.

Because the flows inside a class are not protected against each other, admitting a new flow to a Diffserv service class C can degrade the quality of the other flows in that class. Therefore, before admitting a new flow f, it has to be checked whether the delay bound

of all flows already admitted to the Diffserv service class C that share at least one hop with the new flow f can still be guaranteed after admitting the new flow. The advantage of our BB approach is that it keeps track of the path a flow takes through the network and that it can thus determine easily which other flows the admittance of the new flow would affect. Without the path information, a worst-case assumption would have to be made about how the flows affect each other, leading to a lower admittance quota.

In order to fulfil these tasks, the admission control of the central BB works in three steps when a new flow with token bucket arrival curve (r_f, b_f) and path P through the Diffserv domain requests admittance to service class C:

1. For service class C, a proportion α_C of the link bandwidth bw_l of each link l is assigned off-line. Service class C is overbooked with an overbooking factor ob_C (see below). Let ϑ_l be the set of all currently active flows passing through link l. The new flow f is only admitted to the network if the *bandwidth* limit on each link along its path is not exceeded:

$$r_f + \sum_{g \in \vartheta_l \wedge g \in C} r_g \leq \alpha_C \cdot ob_C \cdot bw_l \qquad \forall l \in P \tag{8.9}$$

2. Similarly, the availability of *buffer space* bf_l has to be checked. The new flow f is only admitted to the network if the buffer limit on each link along its path is not exceeded:

$$\beta \cdot (b_f + \sum_{g \in \vartheta_l \wedge g \in C} b_g) \leq \alpha_C \cdot ob_C \cdot bf_l \qquad \forall l \in P \tag{8.10}$$

   The problem with the buffer space management is that flows entering the network can become more bursty as they share transmission capacities with other flows. The same holds true for protected Intserv flows (RFC 2212, see Shenker *et al.* (1997)) and is expressed by the error terms in (8.2).
   For the Diffserv central admission control, we have to take into account that – contrary to Intserv – the burstiness of the flows sharing a class mutually influences each other. We introduce the error factor β that captures the increase in burstiness of the flows. For *feed-forward networks*, the burstiness can be calculated exactly (see Le Boudec and Thiran (2001)) but not for arbitrary network topologies. Feed-forward networks are networks in which routes do not create cycles of interdependent packet flows. A typical example for feed-forward networks are access networks; for these networks the central bandwidth broker can thus directly give the same *deterministic* service guarantees that Intserv/RSVP or SCORE architectures with DPS can give.
   For arbitrary non-feed-forward networks, the Charny bound (see Section 6.2.4.2) could be used as a delay bound. However, it does not use the full information that is available to our central BB (e.g. the paths of the microflows through the network) and is therefore not efficient in our context. It leads to very low link utilisations for networks of medium to large diameters, as shown in Figure 6.7.
   Exploiting the knowledge about the routing of microflows for non-feed-forward topologies is generally very complex, see, for example, Charny and Le Boudec (2000);

Starobinski *et al*. (2002) and the works cited therein. One possible approach is to use the turn-prohibition algorithm from Starobinski *et al*. (2002) to change the routing in an arbitrary topology to avoid cycles so that the feed-forward properties hold true for that network and the traffic flows in the network. In that case, deterministic guarantees can be given, see above. A similar approach is used in Fidler (2003). The drawback of that approach is that it influences the routing by extending the length of some paths (causing additional delay), depends on an explicit routing mechanism and limits as well as complicates traffic engineering and load balancing. For the purpose of these experiments, it also would introduce a bias towards this special Diffserv system because the routing would be either optimised specifically for this system in all experiments or different in the Diffserv experiments.

The goal of the turn-prohibition routing is to make the network calculus apply to general topologies. This makes it possible to give relatively efficient deterministic guarantees for general topologies with aggregate scheduling. However, these guarantees are only deterministic within the mathematical models themselves and do not take possible failure reasons outside these models like link failures, router misconfigurations, or packet losses and delays caused by routing changes into account. Because of that, a provider would normally be allowed a limited amount of guarantee violations in a Service Level Agreement (SLA) anyway; even the offered service is a 'deterministic' one.

Additionally, our experiments in Section 8.3 show that the EF service can be overbooked quite massively, especially for realistic topologies. Therefore, it can be assumed that most providers overbook the EF class to a certain extent to efficiently use their network. Then, there is no need to determine the worst-case burstiness exactly, especially if it complicates routing and traffic engineering. In these cases, we make the simplifying feed-forward assumption to determine a base value for the error term β. If delay bound violations or packet drops are observed if the class is not overbooked, we increase β until they disappear. Throughout all experiments in Chapter 8, this was necessary only in very extreme experiment set-ups. β is never set to a value below the feed-forward value. Concluding, we adjust the error introduced by applying the feed-forward formulas to non-feed-forward networks with the error term β. This leads to the admission control being based on a statistically relaxed deterministic model controlled by measurements.

3. The *delay bounds* are only checked for the premium service class based on the EF PHB. A flow is only admitted if its delay bounds can be guaranteed.

   The delay bounds of the new flow f and all already admitted flows of service class C that share at least one hop with flow f have to be checked as follows.

   The maximum queueing delay d_{fl}^{q} for flow f on link l is

$$d_{fl}^{q} = \frac{\sum_{g \in \vartheta_l \wedge g \in C} \beta \cdot b_g}{R_l} + \frac{C_{fl}}{R_l} + D_l \tag{8.11}$$

   where R_l is the link bandwidth $R_l = bw_l$ and C_{fl} and D_l are the scheduling error terms of link l; the rate-dependent term C_{fl} typically also depends on the maximum packet size of flow f.

For each flow, the maximum queueing delays d_{fl}^q along the path have to be added, the propagation delay d_l^p has to be taken into account, and the result has to be compared with the absolute delay bound D_f of that flow:

$$\sum_{l \in P} d_{fl}^q + \sum_{l \in P} d_l^p \leq D_f \tag{8.12}$$

Please note that the 'pay-burst only once' property holds true in networks where flows are protected against each other (e.g. Intserv) but it does not hold true in Diffserv networks. Therefore, the delay bound check in a Diffserv network is much more conservative than in a comparable Intserv network. Our experiments in Section 8.3 demonstrate that.

Implementation Issues From the complexity with respect to control/admission control *information*, the Diffserv central BB is roughly as complex as an Intserv/RSVP implementation. This, however, is not surprising as the Diffserv central BB represents the 'best-you-can-do' approach. The advantage of the Diffserv central BB implementation over an Intserv implementation is that the complexity is located at one point and not distributed amongst the routers. It can thus be handled by a dedicated machine that – contrary to a router – does not have to perform other time-critical tasks as well. Moreover, a provider offering premium services will typically have to use a centralised system for authentication and accounting anyway, which also has to be involved in the admission control process.

The third step of the admission control decision above is the most problematic operation the central BB has to make: For one arriving flow, a possibly large number of other flows have to be analysed with respect to their delay bound. The actual implementation of this mechanism, however, offers a great deal of optimisation potential. For each flow, for example, it could be noted by how much slack Δb_g the flow has until its delay bound is violated. For each newly admitted flow it is sharing a link with, that slack would be reduced accordingly. New flows that would make the slack negative have to be rejected.

In addition, our experiments show that the Diffserv QoS systems can be overbooked significantly before anything goes wrong. Because of that potential, it is not necessary to perform the admission control decision for each flow in real-time and fully exact. It could, for example, be replaced in many cases with simpler heuristics because the risk of wrong decisions is very small.

Scheduling Pseudo-priority queueing is used as first scheduling discipline with Weighted Round Robin (WRR) as the implementing scheduler: The EF packets obtain a higher non-preemptive priority than the AF packets by assigning the EF queue in each hop a very large weight w_{EF}. Because the error terms of WRR depend on the number of service classes, WRR is generally not the most preferable scheduler. In our case, however, this drawback does not weigh very much because the number of service classes is very small for the Diffserv QoS systems (four classes). Moreover, NS2 contains a working and tested Diffserv WRR implementation. Another reason is that WRR is also used in related experiments, for example, those in the original EF PHB RFC (RFC 2598, see Jacobson *et al.* (1999)). The WRR weights are shown in Table 8.1.

Table 8.1 Default Scheduling and Configuration

PHB	EF	AF-1	AF-2	AF-3
WRR Weight w_C	1500	1	1	1
Admission control parameter α_C	0.5	0.25	0.25	n.a.
Overbooking factor ob_C	varies	1.0	2.0	n.a.

We used three AF service classes (AF-1 to AF-3); AF-3 being used as a BE service class upon which no admission control is exerted. The bandwidth, remaining after the pseudo-prioritised EF traffic is served, is by default split up 1:1:1 among the AF service classes.

The parameter α_{EF} for EF traffic of 0.5 was based on prior calibration experiments; because EF traffic is treated with priority, we limited its basic resources to 50% of the available resources to avoid starvation of the other classes. a_C is kept constant throughout the experiments but the resources allocatable to EF traffic are varied through the experiments using the EF overbooking factor ob_{EF}.

The scheduling error term C_{fl} for our WRR implementation is the maximum packet size of the flow f and, with n_l denoting the number of queues of link l ($n_l = 4$), the error term $D_l = (n_l - 1)\,\frac{MTU}{bw_l}$.

Overbooking To differentiate the QoS in the different AF classes, the overbooking factor ob_C is introduced for each class C. The AF-1 class is not overbooked while the AF-2 class is overbooked by 100%. Therefore, more traffic is admitted to the AF-2 class than it can theoretically handle. The quality of AF-2 therefore should be lower than that of AF-1.

The EF class is overbooked with varying overbooking factors.

Random Early Detection Active queue management algorithms like Random Early Detection (RED, see Floyd and Jacobson (1993)) are often used in conjunction with the three different levels of drop precedences of the AF services. In our experiments, we did not activate RED or a similar algorithm (see Section 6.2) for the Diffserv queues, as we do not use active queue management and different levels of drop precedences for the other QoS systems. We do not want to give Diffserv an unfair advantage and we do not want to mix the effect of active queue management with our comparison of QoS systems.

Policing For the EF traffic, we police strictly at the ingress nodes dropping out-of-profile packets. As there are no misbehaving flows in our experiments and as the token buckets in our traffic specification are dimensioned large enough (see Table 8.2), there were no out-of-profile EF packets.

For the AF-1 and AF-2 traffic, out-of-profile packets are put into the same physical queue as the in-profile packets to avoid packet reordering. However, out-of-profile packets are dropped with a higher probability than in-profile packets. Out-of-profile packets are always dropped if the queue is filled by 80% or more while in-profile packets are only dropped when the queue is completely full.

AF-3 packets are not policed.

8.1.2.2 Decentral Bandwidth Broker

The difference to the central BB approach is that the decentral BB does not need complete knowledge of the network. The admission control decision is based only on local knowledge at the ingress node and not on knowledge about traffic in the whole domain. Each ingress link l is assigned a certain contingent of bandwidth Γ_{lC}^{bw} and buffer Γ_{lC}^{bf}. This assignment is done prior to the experiment. The BB admits EF, AF-1 and AF-2 flow only up to this limit. The decentral algorithm does not keep track of the state in the network, therefore the delay bound constraint (8.12) cannot be checked for the flows.

For comparison, we set the contingent proportional to the maximum admissible amount of bandwidth and buffer for that link l in the central BB approach multiplied with a scaling factor $\gamma \leq 1$:

$$\Gamma_{lC}^{bw} = \gamma \cdot \alpha_C \cdot ob_C \cdot bw_l \tag{8.13}$$

$$\Gamma_{lC}^{bf} = \gamma \cdot \alpha_C \cdot ob_C \cdot bf_l \tag{8.14}$$

8.1.2.3 No Admission Control

The performance of a Diffserv network that does not use per-flow admission control is also evaluated. It relies on other methods to (roughly) control the traffic to bandwidth ratio, for example, on long-term service-level agreements or on network-engineering methods. For the purpose of these experiments, all flows are accepted and assigned their initially requested Differentiated Services Codepoint (DSCP). Policing is not used.

8.1.3 Olympic Diffserv

The Olympic service Diffserv approach uses strict priority queueing with three priority classes implemented by a simple non-preemptive priority scheduler. The same BB/admission control approaches (central, decentral, none) as in Section 8.1.2 are used. Admission control is imposed on the gold service in the same way as for the premium service (EF PHB) of the standard Diffserv approach described in Section 8.1.2. Policing for gold service is also the same as in the standard Diffserv approach.

All flows requesting silver or bronze service are admitted to the network without admission control and policing.

8.1.4 Overprovisioned Best-Effort

Overprovisioned BE networks with different overprovisioning factors are used. The overprovisioning factor OF describes how the bandwidth bw_l of each link l is increased

$$bw_l^{BE} = OF \cdot bw_l \qquad \forall l \tag{8.15}$$

All links of a network are increased by the same overprovisioning factor OF in the experiments.

8.2 Experiment Setup

For the experiments presented in this chapter, we used NS2. Among other things, it contains a load generation module that allows repeating an experiment in different contexts – in our case, with different QoS systems. For one experiment, traffic flows are generated off-line, and the experiment is then repeated with the different QoS systems and evaluated. The results like loss rate, acceptance rate and utility can therefore be compared directly. The NS2 simulator (see NS2 (2004)) is used to conduct the packet-level simulations.

8.2.1 Traffic

8.2.1.1 Traffic Types

For the experiments, we use different traffic mixes consisting of two types of elastic and two types of inelastic sessions. Elastic sessions produce elastic TCP-based traffic flows that react to congestion in the network (indicated by packet loss[5]) by reducing their rate. We use the following two elastic traffic types.

- Short-lived TCP flows ('s_TCP') resemble small file transfers like most WWW traffic. Short-lived TCP flows rarely spend much time in the TCP congestion-avoidance phase.
- Long-lived TCP flows ('l_TCP') resemble larger file transfers, for example, peer-to-peer traffic. Long-lived TCP flows spend much time in the TCP congestion-avoidance phase. Because of the relatively small duration of our individual experiments (a couple of minutes simulation time, at most), the size of the long-lived TCP flows can be kept relatively small, too.

Inelastic sessions consist of inelastic flows that do not adjust their rate to the network condition. We use constant and VBR flows as inelastic flows and assume that these types of flows represent real-time multimedia traffic as follows.

- Constant bit-rate flows ('CBR') resemble Voice-over IP (VoIP), Game and similar real-time traffic. Our CBR traffic is not perfect CBR as that is unlikely to occur in reality. The sending times of the individual packets are randomised. The arrival curve of the randomised CBR traffic can be described by a token bucket with bucket depth $b = 600$ bytes and a rate $r = 78688$ kbps.
- Variable bit-rate flows ('VBR') represent video conferences and similar applications. We generated three different tracefiles (called *L, M, H*) in a loss-less testbed environment without interfering with background traffic using Microsoft NetMeeting Version 3.01[6]. After initialising the connection, 180 seconds of video were recorded. Three different set-ups (*L, M, H*) were considered:
  - *M* is a normal video-conference, that is, a talking head with an average amount of voice traffic.

[5] We do not use Explicit Congestion Notification (ECN) to signal congestion by packet marking as described in Durham *et al*. (2000).

[6] The traces for video-conferences were recorded using Ethereal Version 0.9.12. The bandwidth settings were set to 'Local Area Network'. The image size was set to medium and the quality controller that allows a step-less adjustment for 'Faster video' vs. 'Better Quality' was set fully to 'Better Quality'.

Table 8.2 Trace File Parameters

Parameter	Trace L	Trace M	Trace H
Average rate	83.46 kbps	171.99 kbps	456.54 kbps
Average packet size	929.8 bytes	871.9 bytes	782.2
Average number of packets per second	11.2	24.6	73.0
Token bucket parameter r	128 kbps	320 kbps	640 kbps
Token bucket parameter b	3980 bytes	12520 bytes	14610 bytes

- L is a still picture with hardly any voice.
- H consists of constant talking and an always moving camera.

The trace file statistics are listed in Table 8.2, the parameters of the traffic types are summarised in Table 8.3.

8.2.1.2 Traffic Mix

We use three different traffic mixes *A*, *B,* and *C*. *A* and *B* contain a relatively large amount of inelastic flows with *B* containing twice the amount of inelastic flows than *A*. Mix *C* contains a low amount of inelastic flow and a lower amount of short elastic flows than *A* and *B*. Table 8.4 lists the number of flows that are started on average in each edge node of a topology within a time window of one minute of simulation time. These numbers are scaled with the available bandwidth in our experiments to adjust the point of operation to the type of experiment.

The amount of sent or received packets and the transfer volume for these two traffic mixes depend strongly on the bandwidth and the used QoS system. QoS systems protecting, for example, the inelastic flows obviously increase the transfer volume of these flows at the cost of the TCP throughput. Table 8.5 lists the transfer volume obtained with the two traffic flows in a best-effort network[7]; this bandwidth led to an average dropping probability of 2–4%. For each traffic type, its average percentage of the total amount of received bytes over five simulation runs is depicted. The 95% confidence interval for every value is below +/− 2%.

8.2.1.3 Token Bucket Parameters for Admission Control

For the admission control in Intserv / Diffserv, a token bucket traffic specification is used. The token bucket parameters for the *VBR traces* are listed in Table 8.2. For the VBR traces, the number of possible token bucket parameters is infinite, as there is a trade-off between the rate r and the bucket depth b. We chose the smallest possible parameter combination (r, b) so that all packets of the three minute trace conform to the token bucket. The rate r was chosen as a whole-numbered multiple of 64 kbps and set to the lowest possible value that keeps the buffer size below 15 KByte.

[7] DFN topology with a bandwidth of 30 Mbps.

Table 8.3 Session and Flow Parameters

Parameter	Short TCP	Long TCP	CBR	VBR
Flow distribution within Session	Exponentially distributed inter-arrival time between the start of two flows with $1/\mu = 3.0$.	Exponentially distributed inter-arrival time between the start of two flows with $1/\mu = 30.0$.	Flows have exponentially distributed duration with an expected duration of $1/\mu = 30$. Two successive flows have opposite directions.	One flow per session, played back endlessly from a 180-second trace file with a random starting point.
Flow size	Pareto distributed with $\alpha = 0.63$. Expected size 10 packets each 15 kB .	Pareto distributed with $\alpha = 0.63$. Expected size 100 packets each 150 kB .	The transmission rate is set fixed to 78,688 bps (equals 64,000 bps without UDP and IP header).	The average transmission rate depends on the trace file, see Table 8.2.
Average flow duration	Strongly depends on network conditions.	Strongly depends on network conditions.	30 s.	Duration of the experiment.
Packet size	Constant 1500 bytes	Constant 1500 bytes	Constant 150 bytes	Variable, average 875 bytes
Packet inter-arrival Time (Time between Two Generated Packets)	According to the TCP greedy algorithm, (there is always data available from the application layer if the transport layer is allowed to send a packet).	According to the TCP greedy algorithm, (there is always data available from the application layer if the transport layer is allowed to send a packet).	Constant	Variable, determined by trace (see Table 8.2).

Table 8.4 Traffic Mix–Number of Started Flows in a Time Window of One Minute

Traffic Mix	Short TCP	Long TCP	CBR	VBR
A	100	50	40	5
B	100	50	80	10
C	50	50	10	2

Table 8.5 Traffic Mix–Percentage of Transfer Volume

Traffic Mix	Short TCP	Long TCP	CBR	VBR
A	37.46	40.27	10.02	12.24
B	28.18	31.96	17.89	21.95
C	25.87	63.10	3.61	7.41

The token bucket parameters r for the CBR flows were set to the average rate of the flows that leads to a buffer depth b of four packets with 600 Bytes as the smallest possible value.

For the TCP flows, we set the token bucket parameter r to 100 kbps, which is slightly below the maximal throughput assumed for TCP in the utility function (see below) and the bucket depth b to the default receiver window size of the NS2 TCP implementation of 20 packets of 30 Kbyte each.

8.2.2 Topologies

8.2.2.1 Used Network Topologies

For our experiments, we used the following topologies; they are depicted in Appendix A. Their basic graph properties are presented there, too.

Star The star topology has a single node in the centre where all cross-traffic will occur, see Figure A.3. The analysis of cross-traffic is important for investigating overbooking for Diffserv, see Section 8.3.

Cross The cross topology is depicted in Figure A.3. It is a variation of the star topology where cross-traffic will occur not only in a single node but also in all three central nodes. For the star and the cross topology, the edge nodes that are marked grey in Figure A.3. are the only nodes sending and receiving traffic flows.

DFN For most of the experiments, we use a real-world topology as the basic topology. We chose the DFN GWiN backbone topology as it is a medium sized real-world topology. The DFN GWiN backbone is the backbone network of the German research network that is connecting most universities and research labs in Germany. The topology is depicted in Figure A.1. For the DFN topology, we assumed that every node is a source of traffic flows and can act as traffic sink for any traffic flow.

Table 8.6 Bandwidth Settings and Average Path Length

	Low Bandwidth	High Bandwidth	Av. Path Length
Cross	20 Mbps	–	4
Star	20 Mbps	–	5.07
DFN	10 Mbps	15 Mbps	2.6
Artificial-3	10 Mbps	15 Mbps	4.2

Artificial-3 In addition, an artificially created topology is used, see Figure A.2. The topology generator Tiers (see Tiers (2004)) was used with the parameters listed in Table A.2 to generate that artificial topology. It is roughly 50% bigger than the DFN topology and has different graph properties (as shown in Table A.1).

8.2.2.2 Bandwidth and Buffer Dimensioning

For the experiments, it is important to adapt the link bandwidth to the configuration and purpose of the experiment. For the ease of implementation, we assumed equal link bandwidths in the network. For the same reasons, the buffer space resources are assigned statically to the outgoing links of a router.

The bandwidth setting is shown in Table 8.6. The low bandwidth setting creates a scenario with high load in the network and is used, for example, in the experiments of Section 8.3 where it is very important that the amount of inelastic flows is very high so that the system can be massively overbooked without the acceptance rate reaching 100%.

The high bandwidth setting was set to 1.5 times the low bandwidth value, which creates a congested but not extremely overloaded network (see e.g. Section B.13). Please note that for the BE reference architecture, we further increased the bandwidth by upto a factor of 8 on top of that.

The buffer space bf_l of one link is set proportional to the link bandwidth bw_l, so the given maximum queueing delay $d_l^{q,max}$ for that link l is fix. A maximum queueing delay of 50 ms is used as default value.

Please note that the number of flows (Section 8.2.1.2), the bandwidth and the buffer space is rather low compared to, for example, the actual DFN GWiN network. This is necessary because we are using packet-level simulations. They allow us on one side to obtain realistic flow, delay and dropping behaviour but on the other side are not scalable enough to simulate much larger networks with more flows in reasonable time. This is why packet-level simulations with a halfway realistic amount of traffic flows are practically never found in the literature. The experiments already took much longer than two weeks on 2.2 GHZ Pentium 4 machines with 1 GB RAM.

Individual experiments were repeated with a higher bandwidth setting and a corresponding increase in the number flows; they did not lead to fundamentally different results.

8.2.2.3 Mapping of Sessions to Network Nodes

We distinguish between edge nodes and core nodes in a topology. Only edge nodes are sources and sinks for the traffic sessions respectively flows. For each edge node, a

Table 8.7 Traffic Weight Distribution

Weight	Probability
0.5	25%
1.0	50%
2.0	25%

weight is selected as shown in Table 8.7. The number of sessions instantiated in a node is multiplied by the weight of that node. The probability for a node being selected as the communication partner of a session starting in another node is also proportional to the node's weight.

8.2.3 Utility

8.2.3.1 Delay Bounds

Real-time multimedia applications are typically delay sensitive. We model this by giving each inelastic flow a delay bound. Packets that are exceeding the delay bound are treated the same as dropped packets for the purpose of calculating the utility function (see below).

For the experiments, we varied the delay bound for these flows. In an experiment, each inelastic flow is assigned the same end-to-end queueing delay bound. Because the size of the topology influences the average path length, the influence of the topology has to be accounted for when setting the delay bound. Therefore, the end-to-end queueing delay bound of all flows is set to $\lambda \cdot d^e$ where λ is the average path length of the topology and d^e is the average per hop delay bound specified by the experimenter for a certain experiment. The default value of d^e is 20 ms.

8.2.3.2 Utility Functions

Figure 8.1 shows the utility functions for the different traffic types. With respect to utility, each flow is evaluated individually. Later on in our experiments, we evaluate the average utility of each traffic type; each flow is weighted the same.

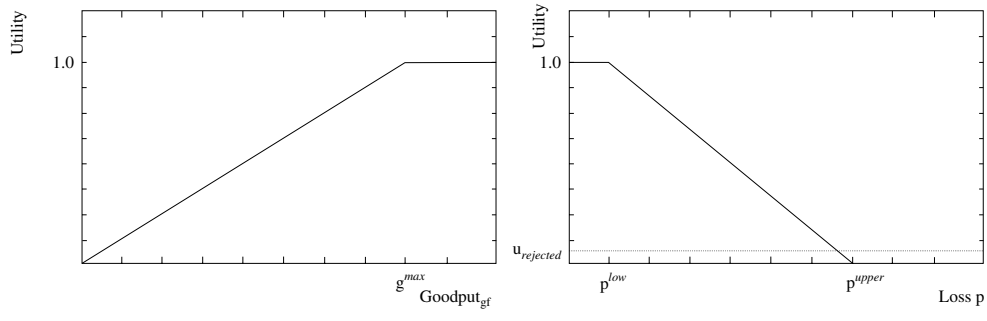

Figure 8.1 Utility Functions

Elastic Flows The elastic flows represent data transfer applications. The utility of data transfer applications mostly depends on the transmission time: the time it takes from the start to the end of the data transfer. The start of the data transfer is the time when the sender sends the first packet to open the TCP connection using the three-phase handshake. As the end of the data transfer, we count the time the receiver has received all bytes of the data transfer. The time until the last packets are acknowledged and the sender disconnects is not counted, because the receiver can use the data before that time.

In the experiments, not all elastic flows finish during the experiment time. Therefore, the transmission time is not known for all flows and approximated through the goodput instead. As the amount of data that a flow is transmitting is given, the transmission time is inversely proportional to the goodput g_f of the flow f. The goodput is defined as:

$$g_f = \frac{\text{number of correctly received packets}}{\text{elapsed time}} \tag{8.16}$$

The goodput can be determined if a data transfer is not complete. For the elastic flows, we use the utility function (a) of Figure 8.1. The utility is a linear function of the goodput g_f upto a certain maximal goodput g^{max}. We assume that once the maximal goodput is reached, the application or the user no longer benefits from a shorter transmission time. We chose a default maximal goodput of 10 pkts/s with 120 kbps, which lies in the same order of magnitude as the transmission rate of the inelastic applications. The utility function is normalised to 1.0.

Inelastic Flows For the inelastic multimedia applications, the loss probability influences the perceived utility. The utility function (b) of Figure 8.1 is used for the inelastic flows. We count packets that are *dropped* (because of congestion) and packets that are not dropped but arrive later than their delay bound (*delayed*) both as *lost* packets. A certain amount of loss can be tolerated (lower threshold p^{low}), then the utility decreases and reaches zero for the upper loss threshold p^{upper}.

As default values, we chose $p^{low} = 1\%$ and $p^{upper} = 10\%$. The utility function is normalised to 1.0.

If an admission control rejects a flow, the information that there are not enough network resources available to transport the flow can be deemed worth a certain amount of utility, especially when compared to a flow that is accepted at first but receives such a high loss probability that its utility is reduced to zero. Therefore, flows that were rejected by the admission control and did not transmit data at utility value $u_{rejected} \geq 0$; the default value for $u_{rejected}$ is 0.05.

8.2.3.3 Assignment of Flows to Services

We assign the delay-sensitive inelastic flows to the 'best' service a QoS system can offer. The elastic applications are assigned to the other services. If there are several alternatives, the short-lived flows are assigned to the higher-quality service. The motivation behind that is that the short-lived flows represent interactive traffic (e.g. web traffic) while the longer flows stand for file sharing (e.g. peer-to-peer traffic) that is supposed to be less time-critical.

Intserv For the Intserv QoS system, all inelastic flows use GS. If they are rejected by the admission control, they are not transmitted. The rate R of the Intserv FlowSpec is set at the assigned rate necessary for guaranteeing the delay bound of the inelastic flow, the slack term S is set to zero.

All elastic flows use the BE service without admission control.

Standard Diffserv All inelastic flows use the premium service using the EF PHB. EF flows rejected by the admission control are not transmitted.

Short-lived TCP flows are assigned the AF-1 class if there are resources available, otherwise they are downgraded to AF-2 and AF-3. As there is no admission control for AF-3, they will always be transmitted.

One-third of the long-lived TCP flows are assigned to AF-2 and downgraded to AF-3 if the resources are not available. The rest is assigned to AF-3 from the beginning.

Olympic Diffserv The inelastic flows use the gold service or – if rejected – do not transmit at all. Short-lived TCP flows are assigned to the standard BE class while the long-lived flows are assigned to the low-priority bulk transfer class. Admission control is imposed only on the gold service.

Overprovisioned Best-Effort In the overprovisioned BE QoS system, all flows use the BE service. There is no admission control that would stop any sources from sending.

8.2.4 Evaluation Metrics

In order to evaluate the QoS systems, the utility and other performance criteria can be measured. Throughout the experiments, the following evaluation metrics are used.

- **Average Utility**
  This is the average utility for each flow of a traffic type.
  All flows of the same type have equal weights irrespective of their actual size.
- **Average Utility of the Accepted Flows Only**
  If admission control is used, only flows that have not been rejected by the admission control are counted for determining this second utility average.
  If no admission control is imposed on a traffic type, this criterion yields the same result as the pure 'Average Utility' criterion above.
  Note: Flows that are downgraded to a lower service class, nevertheless, count as accepted.
- **Acceptance Rate**
  This is the percentage of the flows that were accepted by the admission control.
  If no admission control is imposed on a traffic type, the acceptance rate is automatically 100%.
- **Dropping Probability**
  This is the probability that a packet of a certain traffic type gets dropped because of a full queue before it reaches its destination.
- **Delay Bound Violation Probability**
  This is the number of packets arriving later than their delay bound permits relative to the total number of received packets.

Table 8.8 Abbreviations for the Different Quality of Service Systems

QoS System	Abbrev.	Parameters
Intserv	$IS - \alpha_{GS}$	α_{GS} = Maximum proportion of the link resources available for the guaranteed service class
Standard Diffserv	$sDS - bb - p$	bb = Bandwidth broker type (c = central, d = decentral, n = none) p = Bandwidth broker parameters for the central BB: p = overbooking factor ob for the decentral BB: p = overbooking factor times scaling factor ($ob \cdot \gamma$)
Olympic Diffserv	$oDS - bb - p$	bb = Bandwidth broker type (c = central, d = decentral, n = none) p = Bandwidth broker parameters, same as above
Best-effort	$BE - OF$	OF = Overprovisioning factor

- **Throughput**
  This is the average per-flow throughput of a traffic type in kbps.
- **Traffic Volume**
  This is the amount of volume of correctly received traffic of one traffic type divided by the total received traffic volume of all traffic types.

The graphs depicting the results also always contain the 95% confidence intervals for the different metrics; because of their size, the results are presented in Appendix B. Each experiment was repeated a number of times with new flows but the same bandwidth and topology. The number of repetitions was dynamically increased until the confidence intervals were satisfactory low. The typical number of repetitions is between 5 and 15.

For ease of presentation, abbreviations were assigned to the different QoS systems, see Table 8.8.

8.3 Per-Flow versus Per-Class Scheduling

In the first experiment, we compare the strongest QoS systems of our complete evaluation: The systems using Intserv/RSVP to offer guaranteed service and the Diffserv systems with EF PHB and a central BB. Both systems use per-flow admission control and allocate resources along the path – in the Diffserv case, resources are only allocated within the Diffserv domain. To make sure there are no bottlenecks outside the Diffserv domain, the bandwidth of the links outside the Diffserv domain is set to 10 times the bandwidth of the links inside the domain.

There are two central differences between Intserv and the Diffserv approaches which are as follows.

1. The first difference between the two approaches is that Intserv uses a per-flow scheduler while Diffserv schedules per-class. This also influences the admission control decision as discussed in Section 8.1.1.
2. The second difference between the two approaches is that the Diffserv QoS system also differentiates the non-EF flows into three service classes while the Intserv system treats them all as BE traffic. This difference will only influence the performance of the elastic flows.

The bandwidth for the experiment was set to the lower values of Table 8.6 to allow for massive overbooking of the Diffserv premium and gold service class. The results are depicted in Figures B.1 to B.3 for the DFN topology.

We first focus on the performance of *Intserv*. Configurations with a different parameter α_{gs} are shown. α_{gs} is the maximum proportion of the total link resources available for the GS flows (the inelastic flows). The following things can be noticed.

- The utility of all the accepted *inelastic* guaranteed service flows (CBR and VBR) is 1.0 – the maximum possible value (see Figure B.1) as can be expected from the fact that Intserv offers strict loss and delay guarantees.
- The utility of the *elastic* flows increases slightly (Figure B.1) if the parameter α_{gs} is decreased; in that case more flows requesting GS are rejected, as can be seen in Figure B.2.

We now look at the performance of the *Diffserv* systems (sDS, oDS). The figures show the performance metrics for overbooking factors *ob* from 1 to 8:

- The admission control decision for Diffserv has to be more conservative than that of Intserv because the flows within one service class are not protected against each other. The conservativeness of the decision is visible when comparing the acceptance rate (Figure B.2) of Diffserv with an overbooking factor *ob* of up to 3 (sDS-c-3) with those of Intserv IS-0.9:
  Despite the fact that the bandwidth and buffer *assumed* available for the admission control decision is significantly higher for sDS-c-3 than for IS-0.9, IS-0.9 can still admit slightly more flows to the network than Diffserv because of the flow protection. On the basis of the worst-case assumption in the admission control decision, the Intserv approach has to allocate fewer resources than the Diffserv system to guarantee the same delay bound. To quantify this, Intserv has to allocate only about 44% of the resources that the standard Diffserv system needs in this experiment.
- The same conclusion holds true for the Olympic Diffserv (oDS). The only difference is that the Olympic Diffserv admission control decision can admit slightly more flows than sDS if everything else is the same because of the smaller error terms of the scheduling algorithm (priority versus WRR); see Figure B.2. Intserv has to allocate about 47% of the resources that the Olympic Diffserv system needs in this experiment.

Because of the conservativeness of the sDS and oDS admission control decision, an important question to ask is *how much can the EF-based (premium respectively gold) service class be overbooked?*

- As can be seen from Figure B.3 (also reflected in the utility values of Figure B.1) for the DFN topology, the sDS/oDS systems can be massively overbooked. The first packet drops and delay-bound violations occur at an overbooking factor $ob = 4$ but only on a very small scale (significantly less than 1 per 10^6 packets and therefore hardly noticeable in the figures).
  However, even for an overbooking factor $ob = 8$, the dropping and delay-bound violation probabilities are still very small and for most applications acceptable. For the oDS

Table 8.9 Per-Flow vs. Per-Class Scheduling, Cross and Star Topology, Dropped or Delayed Packets [%], Summary

QoS System	Over-booking	Cross Topology CBR	Cross Topology VBR	Star Topology CBR	Star Topology VBR
sDS-c	1	0	0	0	0
	2	0	0	0	0
	3	0	0	0	0
	4	0.012	0.010	0.194	0.115
	6	4.34	3.58	8.26	7.24
	8	17.16	16.41	20.86	20.85
oDS-c	1	0	0	0	0
	2	0	0	0	0
	3	0	0.0004	0.0003	0
	4	0.020	0.006	0.097	0.052
	6	3.646	2.97	6.80	5.67
	8	16.16	15.23	19.55	19.41

systems, they are lower than for the sDS systems, which is explained by the stricter priority scheduling discipline.

On the basis of the results discussed so far, an overbooking factor $ob = 4$ to 8 can be recommended for the DFN topology when using a central BB for statistical guarantees, or $ob = 3$ if the EF traffic is extremely sensitive to loss respectively delay.

- The Charny bound (see Section 6.7) for this experiment set-up and traffic predicts a maximal utilisation of 7.98%. The centralised BB can raise the utilisation in the experiment to 13.13% without overbooking and with an overbooking factor of 4 to more than 27%.
- The overbooking potential is not as high as demonstrated above for all types of topologies. We repeated the same experiment for the artificial *Cross and Star topologies*. The resulting dropping and delay probabilities are summarised in Table 8.9.

  For the *star topology*, an overbooking factor $ob = 3$ already leads to some dropped respectively delayed packets for Olympic Diffserv. For an overbooking factor $ob = 6$, the loss is already higher than 5.67% for both Diffserv flavours, surely unacceptable for a premium service.

  Comparing the standard Diffserv with the Olympic Diffserv, the latter has a generally lower loss ratio despite the fact that it is accepting more CBR and VBR flows because of the smaller error terms. This can be explained with the strict priority scheduler that empties the EF queues quicker than the WRR-based pseudo-priority scheduler, leading to less loss for small overbooking factors. The additional amount of admitted flows in the oDS systems, however, is noticeable by the fact that for oDS-c-3 the losses are higher than for sDS-c-3, where in fact no loss was observed.

  For the *cross topology,* similar arguments hold true. However, the dropping and delay-bound violation probabilities are generally lower than for the star topology. At the central node of the star topology, all flows cross paths; this creates a lot of cross-traffic within the EF service class on the outgoing links of that node. For the cross topology,

the cross-traffic is distributed among the three central nodes where the paths of the flows cross. For the DFN topology, there are no clear bottlenecks. The cross-traffic creates additional delay and in an overbooked system increases the dropping probability.

Looking back generally at the dropping probabilities shown in Figure B.3, one can notice two phenomena as follows.

- The dropping probabilities for the elastic flows are extremely high (>10%) for all QoS systems. They are higher than one would observe in a physical network with real users. There are several reasons contributing to this extremely high loss rate here, as listed below.
  - First of all, in this experiment the system is extremely loaded and thus bandwidth and buffer are very scarce. This is necessary to analyse overbooking because not all inelastic flows will be accepted even with an overbooking factor $ob = 8$.
    As the number of elastic flows is proportional to the number of inelastic flows, their number is extremely high contributing to the high losses.
  - Second, the elastic flows are treated with lower priority by the Diffserv systems and receive only a small share of the bandwidth than in the fully loaded Intserv systems.
  - The third reason is the experiment set-up itself. In reality, some users would back off in a network loaded as highly as assumed in this experiment. Throughout our experiments, we do not model this type of user behaviour because we want to maximise the comparability of the results for different QoS systems. If a system providing poor QoS would be 'rewarded' with less traffic after some time (by users backing off), this would seem 'unfair' and the comparability would not be warranted.

  These three reasons explain the high dropping probability that would probably not be observed in reality. As ECN is not used in the experiments, the elastic TCP flows rely on packet drops as congestion indication, therefore even for very high bandwidths the dropping probability is significantly above zero; this can be seen, for example, for the BE-8 results in Figures B.15, see also Section 8.5.
- Another phenomenon that can be observed throughout the experiments is that the dropping probability is usually orders of magnitude higher than the delay-bound violation probability.
  This is explained by the relationship between the delay bound and the available buffer space. The star topology has an average path length of 4 hops. In the experiments, we set the delay bound of the flows proportional to the average number of hops that lead to an end-to-end queueing delay bound of 80 ms. The cross topology has an average path length of 5.07 hops, leading to a queueing delay bound of slightly more than 100 ms. For the DFN topology, every node acts as source and sink, which leads to a rather short average path length of 2.6 hops and a queueing delay bound of 52 ms for the flows.
  The available buffer space of an sDS EF queue allows a maximum queueing delay of 25 ms for a conforming packet in a single EF queue before the packet is dropped.
  Comparing these numbers, for a delay-bound violation an EF packet has to traverse several congested queues in a row. When it does so, it automatically has a high dropping probability. For the cross and start topology, the number of congested queues is mostly limited to the central links, so that delay-bound violations are unlikely.

Next, we analyse the effect of changing the delay bound from 20 ms to 10 ms and 40 ms per average number of hops. The effect on the acceptance rate is depicted in Figures B.4 and B.5 as follows.

- Decreasing the delay bound increases the amount of bandwidth and buffer resources allocated to a flow by the Intserv admission control, see (8.1) and (8.2). The resource allocations are largely influenced by the token bucket depth b; this explains why the effect is stronger for the VBR flows.
- For Diffserv, the acceptance rate for sDS and oDS drops by around 20% for the CBR flows and 40% for the VBR flows because the delay-bound check (8.12) more often fails for the lower delay bound. As the delay bound (8.11) depends on the burstiness of the flow, the effect is again stronger for the more bursty VBR flows.
- As the acceptance rate for 10 ms is really low, the dropping probability drops to zero for sDS and oDS even for an overbooking factor $ob = 8$. Delay-bound violations can be observed at $ob = 6$ and 8 but are less than 4 per 10^6 packets.
- If the delay bound is increased, the opposite effects can be observed (Figure B.5).

To conclude, our analysis of per-flow and per-class scheduling showed the following things: Our central Diffserv BB can give Intserv-like deterministic loss and delay guarantees for individual flows despite the fact that these flows are aggregated into classes when routed through the Diffserv domain. However, Intserv-like per-flow scheduling is more efficient than the per-class scheduling.

Because of this and the worst-case decision made by the central Diffserv BB, the Diffserv system can be overbooked. The overbooking factor depends on the topology, especially on the amount of cross traffic. A well-connected topology like the DFN topology can be safely overbooked by a factor of three or four.

8.4 Central versus Decentral Admission Control

The central BB in the Diffserv systems can become a bottleneck itself. If resource allocations are made on a small timescale (e.g. per flow), centrally managing a larger network can quickly become an impossible task. We have already argued that there is some optimisation potential for the BB that could be used. Reservation thresholds could be introduced as described in Schmitt *et al*. (2002).

Another solution to this problem is to decentralise the admission control completely and base the admission control decision purely on local information at the edge. In this section, we evaluate this approach. A contingent-based admission control algorithm is used; each edge node is assigned a contingent of resources (bandwidth and buffer) for each ingress link. As link bandwidths and buffer spaces are equal within the Diffserv domain in the experiments, we made the contingent proportional to the link bandwidth and buffer of the ingress link. This also allows for a better comparison with the central BB approach, where the admission control decision is also based on the link bandwidth and buffer (in the case of all Diffserv domain links on the path).

The efficiency of the decentral algorithm will depend strongly on the correct setting of the contingents. If the contingent assigned to an edge node is too low, then too many flows

will be rejected or degraded. If it is too high, then too many flows are admitted and the QoS suffers. On a medium timescale[8], the INSP can react and reassign the contingents. This, however, does not guarantee a good performance for the future as the traffic patterns can change.

We evaluate the following two situations.

- *Situation A* represents a situation where the contingents match the traffic patterns very well.
  For situation *A,* the traffic sources and sinks are distributed evenly among all edge nodes by assigning each node the node weight 1.0 (see Section 8.2.2.3). Equal contingents are also assigned to all Diffserv ingress links.
- *Situation B* emulates the case in which the traffic prognosis and contingent assignment are not matched with the distribution of the traffic sources and sinks.
  For situation B, we distribute the traffic sources and sinks non-uniformly with the method described in Section 8.2.2.3, while we assign equal contingents to the ingress links. Thus, a mismatch is created.

The results are shown in Figures B.6 to B.11 in the Appendix. We start by analysing *situation A*. Several effects can be noticed as follows.

- The acceptance rate increases massively from the central bandwidth broker/admission control (sDS-c and oDS-c) to the decentral one (sDS-d, oDS-d). The following two reasons can be given for that.
  - First, the delay-bound check is not performed by the decentral but by the central BB. The central BB thus performs a stricter admission control per se.
  - The second reason is that the decentral BB only checks a single ingress link. The central BB checks the complete path through the Diffserv domain that – for the DFN topology – consists of 2.4 links on average. As in situation A, the flows arrive in random order, the used link resources differ from link to link. Thus, when a check for available resources fails, the more likely that more links are part of the check.
- The acceptance rate of the CBR and especially of the VBR flows increases, the higher the parameter $ob \cdot \gamma$ of the decentral BB becomes – that is, the more the decentral BB overbooks.
  This is obvious because overbooking increases the *assumed* amount of available resources for the inelastic flows.
- Evaluating the dropping and delay-bound violation probability for the inelastic flows, one notices that even for a small overbooking factor ($ob \cdot \gamma = 1.5$), loss occurs and the utility drops below 1 because the inelastic flows experience packet drops and delay-bound violations. Please note that for $ob \cdot \gamma = 3$ the losses would be even higher had the acceptance rate not already reached 100%.
  These losses are interesting because our previous experiments of Section 8.3 show that the central BB can be overbooked for the DFN topology by more than a factor of $ob = 3$ until this occurs.

[8] Outside the scope of a single simulation run.

For the decentral BB approach, this obviously no longer holds true. The missing delay-bound check and the fact that only the resources of the ingress links and not the core links are managed leads to a more generous and less controlled admission of flows. The chance of failures increases, thus the network cannot be overbooked so much.

- The loss of the elastic flows drops with an increase of inelastic acceptance rate and so does traffic volume as can be expected. Also, the utility of the short TCP flows is generally higher than that of the long TCP flows because they are preferably assigned to better service classes in sDS.
- Comparing the loss probabilities of sDS and oDS, those of oDS are generally slightly smaller for the higher overbooking factors. This is the same behaviour as observed in Section 8.3 and can be explained by the strict priority scheduler.

Next, we compare the effect of a mismatch of the assigned link contingents of the decentral BB algorithm (*situation B*):

- The acceptance rate effects visible for situation A are also visible for situation B. The general acceptance rate in situation B is similar to situation A, only for high $ob \cdot \gamma$ the acceptance rate is slightly lower. This is explained by the random influence of the scenario generation method.
- The dropping and delay violation probabilities are larger by roughly a factor of 5 in situation B than in situation A. The mismatch of situation B increases the risk of dropped and delayed packets.
  This behaviour shows the additional risk of the decentral bandwidth broker when the contingents are not well matched with the arriving flows at the edge of the network.

To conclude, using the decentral BB, the admission control decision becomes inexact. The risk of losing EF packets increases. The system should no longer be overbooked. If the system is not overbooked (for $ob \cdot \gamma = 1$), we did not observe any packet drops or delay-bound violations.

8.5 Direct Comparison

We next compare the different QoS systems directly. With the previous two experiments, we have already narrowed down the choice of sensible Diffserv configurations. On the basis of the results of these experiments, we evaluate all mentioned QoS systems by comparing their performance for different traffic mixes and different topologies in this experiment. Also, the overprovisioning factors are determined now. First, the DFN topology and later an artificial topology are analysed.

DFN Topology

The results for traffic mix A are depicted in Figures B.12 to B.17; for traffic mix B and C the main results are summarised in Tables B.2 and B.3 in Appendix B. As many of the effects visible in these graphs have already been discussed in the previous sections, we focus on the general performance evaluation here.

The traffic situation analysed in this section is a high-load situation in which the network is significantly congested. But, still, in the analysed situations, the available bandwidth is not too small. The network is more or less well dimensioned (when a QoS system is used). This is reflected itself in the results for sDS-n and oDS-n, where all inelastic flows are admitted to the network (as there is no admission control) and still experience practically no drops or delay-bound violations.

We start by looking at the average *throughput* of the different types of traffic flows, as shown in Figure B.17.

- The throughput of the inelastic flows equals their sending rate very closely for the Intserv and Diffserv systems. This results from their low dropping probability (Figure B.15).
- The average throughput of the elastic flows exceeds that of the inelastic flows, as there are a number of paths through the network that are only lightly loaded, at least for a period of the simulation time. The elastic flows adapt their transmission window to make use of the available bandwidth.
  A throughput of 120 kbps yields a utility of 1.0 for both elastic flow types. As can be seen from the utility results (Figure B.13), a significant number of elastic flows do not reach this throughput because they are on a congested path through the network.

We now analyse the *inelastic flows* admitted to the network. Figure B.12 shows that they achieve maximum utility in all Intserv (IS) and Diffserv (sDS, oDS) configurations. The acceptance rates between these systems, however, differ greatly (Figure B.14) and thus also the overall utility of the admitted *and* rejected inelastic flows (Figure B.13).

- For the given load situation and with respect to the overall utility of the inelastic flows, the sDS and oDS systems without admission control (sDS-n, oDS-n) perform best. The reason is that they admit all inelastic flows to the network and the network just has enough resources to serve them. Also, these systems are the QoS systems that have the lowest implementation complexity, as they require no signalling and admission control and only rely on long-term network engineering.
  It has to be mentioned that using these systems, however, leads to a certain risk. If the number of inelastic flows increases, the inelastic flows experience a service degradation that all the other QoS systems can avoid because they are using admission control (see the results for traffic mix B below and the results of Section 13.1).
- Deterministic service guarantees (with delay-bound guarantees) can only be given by the IS and the non-overbooked sDS-c and oDS-c systems. Looking at Figure B.13, IS-0.9 performs best. It offers the highest utility of the mentioned systems for the inelastic flows and only a slightly lower utility for the elastic flows than IS-0.6.

Next, we evaluate the performance of the *elastic flows*.

- The oDS systems assign higher priority to the short-lived elastic flows than the long-lived ones. This is clearly visible from the dropping probability and utility.
- The sDS systems also give preferential treatment to the short-lived flows. The short-lived flows are assigned preferably to service classes that are not (or not so much)

overbooked. Therefore, for the sDS system, the short-lived flows receive better performance than the long-lived ones. The difference between both flow types is smaller than for the oDS systems.

It must be pointed out that the performance difference for the elastic flows is much more controlled for the sDS system than for the oDS system. A provider has much more adaptation possibilities for sDS by assigning flows to a number of service classes and assigning the weights and overbooking factors to these classes than he has for oDS where strict priority scheduling is used. This advantage of sDS over oDS, however, also depends on whether the provider can explain and sell its customers the more complicated sDS differentiated services.

In addition, it should be stressed that when not using a bandwidth broker and admission control (sDS-n, oDS-n), the service degradation of the elastic flows is not controlled because the higher-priority flows are uncontrolled. This effect is partly visible for sDS-n and oDS-n that offer the lowest utility for the elastic flows.

- The IS systems do not support differentiation of the elastic flows. The long-lived flows receive a slightly higher throughput and fewer packet drops within the same service class as the short-lived ones. The same holds true for the BE systems where also both types of TCP flows are treated equally.
  - The explanation for the higher throughput is that because the flows are long-lived, they are dominated by the congestion-avoidance phase. If a short-lived and a long-lived flow have the same path through the network and that path is only lowly congested for a certain time, both flows will increase their rate. The short-lived flow finishes after transmitting a low number of packets and stops. The long-lived flow continues increasing its rate so that the average rate of the long-lived flows can be expected to be higher.
  - The short-lived flows are dominated by the slow start phase during which they double their congestion window while the long-lived flows are more likely dominated by the congestion-avoidance phase during which they linearly increase the congestion window. Additionally, long-lived flows are typically more aware of the congestion situation along their path than short-lived ones because the latter rarely transmit long enough – as their name implies – to experience loss, go through slow start again and switch to congestion avoidance mode to slowly approach to congestion point of the network path. This explains not only why the dropping probability of the short-lived flows is higher but also why it does only drop insignificantly for the extremely overprovisioned networks BE-4, BE-8; see Figure B.15.

Next, we evaluate how much a BE network has to be overprovisioned to offer the same performance as a network using a QoS system.

- The results for BE-1 show that without a QoS architecture and without overprovisioning, it is mainly the inelastic flows that suffer (see Figure B.13). The reason is not so much the dropping probability as the delay-bound violations (see Figures B.16 and B.15). The performance increases when the BE system is overprovisioned. The dropping probability is always significantly higher than zero because TCP is using packet drops as congestion indication and increases its rate until it experiences packet loss.

- Evaluating the performance of the inelastic flows, massive overprovisioning by a factor of 4 to 8 is necessary to compete with the best QoS systems. Only for an overprovisioning factor of 6 in our experiments, exactly the same utility is reached. For that overprovisioning factor, however, the elastic flows show a much better performance in the overprovisioned BE system than in any of the other (non-overprovisioned) QoS systems. The performance is probably acceptable for most inelastic applications with an overprovisioning factor of 4. This result is consistent with the analytical results of the previous chapter.
- The performance of the elastic flows is generally better for the BE systems than for the QoS systems because the latter systems degrade the service of the elastic flows for protecting the inelastic ones. The utility of the elastic flows in the BE-1 system is worse than in most QoS systems; for an overprovisioning factor of 1.5, however, the utility is already better than for most QoS systems (this depends on how the long- and short-lived flows are weighted).
  The reader should not get the idea from these observations that TCP performance is generally bad in the QoS systems. It is just that first, in our experiments, we assigned purely non-TCP flows to the premium service classes (GS, EF, Gold) and second, the load in the premium classes is very high.

The impact of the traffic mix on the performance can be seen by comparing the results for *traffic mix B and C* (summarised in Table B.2 and B.3) with the results for traffic mix A as discussed above.

- In traffic mix B, the number of inelastic flows is doubled compared to traffic mix A while the amount of elastic flows remains equal. The utility of all types of accepted flows remains roughly the same for the systems with admission control (IS, sDS-c, oDS-c) while it drops significantly for the Diffserv system without BB (sDS-n, oDS-n) and the BE systems. This effect is caused by the increased amount of traffic against which the systems without admission offer no protection.
- The acceptance rate of the admission-controlled systems decreases as can be expected by the increased number of flows upon which admission control is exerted. However, it does not halve, as one might expect from the fact that the number of inelastic flows doubles. This effect is explained by the fact that the different flows differ in their starting times, duration, their target nodes and (for VBR) in their size. With an increasing number of offered flows, it is more likely that the admission control can fit in a flow on a path where a certain amount of resources is left. Therefore, more flows fit into the same network when more flows are offered and the acceptance rate does not drop fully by 50%.
- For traffic mix C, the number of inelastic flows is drastically reduced (see Table 8.4). The number of short TCP transfers is halved. As can be expected, the acceptance rate of the inelastic flows increases because less flows are competing for the resources. The performance of the elastic flows generally improves, which can be attributed to the fact that there are less short-lived elastic flows. Short-lived flows are less reactive to congestion than the long-lived flows because of their short lifetime; their reduced number therefore leads to significantly less congestion and better performance of the elastic flows.

- The recommended overprovisioning factor for the BE architectures remains relatively independent of the traffic mix, which is again consistent with the analytical results of the previous chapter.

The resulting overprovisioning factor of four for the different traffic mixes already indicates that the Intserv and Diffserv QoS systems can offer a significant advantage over BE systems. That advantage, however, comes at the cost of increased complexity. After the BE systems, the least complex QoS system is the Olympic Diffserv system without BB (oDS-n). It uses simple priority queueing that is implemented in practically all modern router operating systems; no BB or admission control is required. The only requirement is that packets are marked at the ingress nodes. If the marking is based purely on packet header information[9], not even SLAs have to be negotiated and managed. This QoS system can – despite its simplicity – offer a *tremendous advantage* over a BE system with the same bandwidth and therefore similar costs in times of a high utilisation; this can be seen very clearly in Figure B.14 and Table B.2. The utility[10] of the inelastic (multimedia) flows and the short-lived TCP flows (representing e.g. web traffic) is much higher for oDS-n than for BE-1. This comes at the cost of a reduced performance of the long-lived TCP flows in the experiments that were assumed to be representing, for example, P2P traffic. Considering the fact that the willingness-to-pay of a customer for the high-quality transmission of a multimedia flow is probably significantly higher than that of a P2P flow, the simple oDS-n QoS system can be expected to improve the profitability of a network massively. It has to be kept in mind, however, that it still shares one disadvantage with the BE systems – the lack of admission control – and thus depends on traffic and network engineering measures. This is further elaborated in Part IV of this book, especially Section 13.1.

Artificial Topology

We next analyse the influence of the topology. The DFN topology has a relatively low diameter and a low node-degree (see Table A.1 in Appendix A). The average path length through the topology is 2.6. For comparison, we chose a topology that is quite different, the Artificial-3 topology of Appendix A. First of all, it is created artificially with the topology generator Tiers (see Appendix A for details). The average path length is 4.2; a flow needs significantly more hops through the topology as for the DFN (or a similar) topology. The average node-degree is much higher and much more unevenly distributed, which is reflected itself in the much higher standard deviation of the node-degree (see Table A.1). By visual comparison, the DFN and Artificial-3 topology are also quite different (see Figure A.1 and A.2).The artificial topology has more distinctive star-shaped connections than the DFN topology. As our experiments for the Star and Cross topology in Section 8.3 showed, the cross-traffic occurring at these nodes is more challenging for the admission control and reduces the overbooking potential. This is also reflected in the results for the Artificial-3 topology. They are depicted in Figures B.19 to B.16.

[9] The application a packet belongs to can be guessed by looking at the protocol number and port numbers of the TCP/IP respectively UDP/IP headers.

[10] As there is no admission control, the overall utility and the utility of the accepted flows only is equivalent.

- First of all, one notices the increased dropping probability (Figure B.21 compared to B.15) for the elastic flows. There is more congestion in this network. One explanation for this follows: The absolute number of flows in our experiments is proportional to the number of edge nodes, which are almost 50% higher for the artificial topology than for the DFN topology. At the same time, the average path length increases by more than 60%. The traffic is distributed among an increased number of links. The number of links increases by 85%. So, the average number of flows passing through a single link increases by roughly 30%. This leads to increased congestion, visible for the elastic flows in all architectures.
- The QoS systems with a non-overbooked exact admission control (IS, sDS-c, oDS-c) lead to zero dropped or delayed packets for the inelastic flows because of their admission control and resource management. The overbooked central BB systems (sDS-c-3, oDS-c-3) lead to very few packet drops (1 per 10^5 packets). Compared to the DFN topology, the dropping probability is increased, which can be explained by the cross-traffic effects that were also observed for the Star and Cross topologies (see Section 8.3).
- The decentral BB (sDS-d, oDS-d) and the Diffserv systems without BB (sDS-n, oDS-n) fail and lead to extreme losses for elastic and inelastic flows. The similar performance of these systems is explained with Figure B.20; the acceptance rate of all these systems is very similar and close to 100%, which explains the small differences.
  Obviously, the decentral BB is not well suited for this topology. From the structure of the artificial topology, it can be expected that the load of the individual nodes and links varies a lot. Links connecting two star-shaped subnetworks can be expected to be loaded much higher than the average link. As the decentral BB bases its decision purely on the situation of the ingress links, it admits too many flows to the network. This becomes visible when comparing the acceptance rates of sDS-c-3 and sDS-d; sDS-c-3 checks every link along the path. For the DFN topology, the sDS-d acceptance rates are only 5–10% higher than the sDS-c-3 ones (Figure B.14), whereas for the artificial topology they are around 35% higher (Figure B.20), clearly an indication that most bottlenecks are not at the ingress link.
  The performance of the decentral BB improves significantly if the threshold of the decentral BB is reduced to 0.5 instead of 1 or 3. Still, this result stresses the advantages of the central BB approach.
- The delay-bound violation probability for the BE systems shows a different behaviour for the artificial topology compared to the DFN topology. While it continuously drops for the DFN topology (Figure B.16), it first increases with an increased overprovisioning factor and only later decreases for the artificial topology (Figure B.22). We already argued above why the artificial topology is more congested. Therefore, it shows the same behaviour as the less-congested DFN topology once the bandwidth was increased enough to compensate for the additional congestion. The seemingly illogical increase of the delay-bound violations when the bandwidth is increased by the overprovisioning factor is explained as follows: The overprovisioning factor is applied to the bandwidth *and* the buffer space of the routers. As the buffers increase, the possible queueing delay and thus the chance of a delay-bound violation increases, too. Only if the additional increase in bandwidth is enough to empty the queues and reduce the congestion, the queueing delay and with it the delay-bound violation probability are reduced.

- For the artificial topology, an overprovisioning factor of two is necessary for the elastic flows to show the same average utility as for the sDS-c/oDS-c. For the inelastic delay-sensitive flows, the situation is dramatically different. Even an overprovisioning factor of 8 is not enough to offer the inelastic flows the same utility in the BE systems as in the best QoS systems.

8.6 Summary and Conclusions

In this chapter, QoS systems based on the Intserv, Diffserv and a plain BE architecture were evaluated in a series of experiments. Also, a central BB for Diffserv was developed. It can give Intserv guaranteed service-like guarantees for individual flows without needing per-flow complexity in the core network. Our experiments demonstrate that while it can offer the same QoS as Intserv, it is not efficient without overbooking. Generally, our experiments show that the different Diffserv systems with central BBs can be overbooked significantly. The exact amount of overbooking depends on the topology.

If a decentral admission control is used instead of a central one, significant control over the network is lost, resulting, for example, in the loss of overbooking potential.

The overprovisioning factors determined in our experiments are similar to those found in our analytical study of Section 7.2. The experiments also demonstrate that the BE systems perform generally by for flows with the delay requirements – even with high overprovisioning factors – because of their inability to differentiate between the different service classes.

The performance of the lightweight Diffserv QoS systems without any admission control was generally very good, especially as their implementation and administration costs can be expected to be relatively low. Because of the absence of admission control, however, they – and the BE systems – rely on in-time capacity expansion and other traffic and network engineering measures. This is being further investigated in Part IV of this book, especially in Chapter 13.