Part II

Network Architecture

The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design
Oliver Heckmann ISBN: 0-470-01293-5

6

Network Architecture Overview

6.1 Introduction

Intelligent Network Service Providers (INSPs) offer layer 3 packet forwarding services by operating an IP network. The technical infrastructure of the network provides the core packet transport service that an INSP bases its business upon; for the quality and costs of the transport services it thus plays a vital role. Therefore, in this Part II of the book we discuss the network architecture which defines the characteristics of the network infrastructure:

With the term *Network Architecture* we describe the technology used for building the network of an INSP. The properties of a network depend on its network architecture and the configuration of that architecture. We distinguish the four sub-architectures that are depicted in Figure 6.1 as:

- **Quality of Service (QoS) Architecture**
  The QoS architecture describes the technical measures that provide quality of service. The nature of the QoS architecture has strict consequences for the forwarding and signalling architecture. For example, Intserv as QoS architecture makes the use of a QoS signalling protocol such as RSVP as part of the signalling architecture (see following text) very likely and works well with both a plain IP or a Multi-protocol Switching Label (MPLS) data forwarding architecture.
  We discuss QoS architectures in Section 6.2.
- **Data Forwarding Architecture**
  The data forwarding architecture describes the actual technical packet forwarding technology. INSPs can use plain IP packet forwarding where every hop in the path of the packet through the network is an IP router that looks up IP header information in its routing table to decide on how to forward the packet. An alternative data forwarding architecture is label switching packets using MPLS technology.
  Data Forwarding Architectures are discussed in Section 6.3.
- **Signalling Architecture**
  The signalling architecture encompasses the different signalling and control protocols to manage the network. This includes interior and exterior routing protocols, QoS

The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design
Oliver Heckmann ISBN: 0-470-01293-5

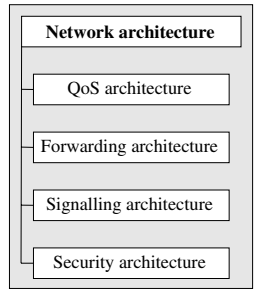

Figure 6.1 Network Architecture

signalling protocols and label distribution protocols. They are discussed in more detail in Section 6.4.

- **Security Architecture**
  The security of an INSP's network depends on many factors, for example, the IP-level security architecture, the quality of its implementation, router operating system security and the physical security of the network. The IP-level security architecture of an INSP provides security at the IP packet level. Security issues encompass data encryption, authentication, confidentiality and network-level protection against denial-of-service attacks. The IP security architecture is discussed in Section 6.5.

In the remainder of this chapter, we give a detailed overview of the different QoS architectures discussed in the community. Then, data forwarding, signalling and security architectures are presented. Towards the end of the chapter, admission control mechanisms are discussed.

In the next two chapters, we analytically and experimentally compare different QoS architectures and admission control mechanisms with respect to several aspects to shed light on the advantages and drawbacks of these architectures and mechanisms.

6.2 Quality of Service Architectures

We use the term *QoS architecture* to describe the general technology upon which actual *QoS systems* are based. The range of technical forwarding services an INSP can offer to his customers depends on his QoS system. The efficiency with which these services are provided also depends on the QoS system. Therefore, the QoS architectures upon which those QoS systems can be based are highly important for the purpose of this book and are discussed in detail next. We will start by defining a *QoS system* and its components (Section 6.2.1) and then we will discuss different QoS architectures for IP networks.

- Integrated Services in Section 6.2.2,
- Stateless Core (SCORE) with Dynamic Packet State (DPS) in Section 6.2.3,
- Differentiated Services in Section 6.2.4,
- Several best-effort based approaches in Section 6.2.5 and
- Finally other more exotic approaches in Section 6.2.6.

We conclude the discussion of these architectures with a summarising classification in Section 6.2.7. Admission control plays an essential role in offering service guarantees and high quality services. In addition, many admission control works are relatively general and independent of specific QoS architectures. For these reasons, we present a separate overview and classification of admission control mechanisms in Section 6.6. In that context, we also present specific implementations of admission control mechanisms, for example, in the form of a Diffserv bandwidth broker which is used for the experiments later in this part.

6.2.1 Components of a Quality of Service System

The following definitions are based on Schmitt (2001); their inter-relation is shown in Figure 6.2.

A *QoS system* consists of the *QoS architecture* that describes the technical part of the QoS system and the *QoS strategy* that determines how an INSP exploits the technical features offered by the chosen architecture. The strategy involves the configuration of the architecture, policy decisions and tariffing. While there are only a low number of QoS architectures under discussion in the community, the number of QoS systems

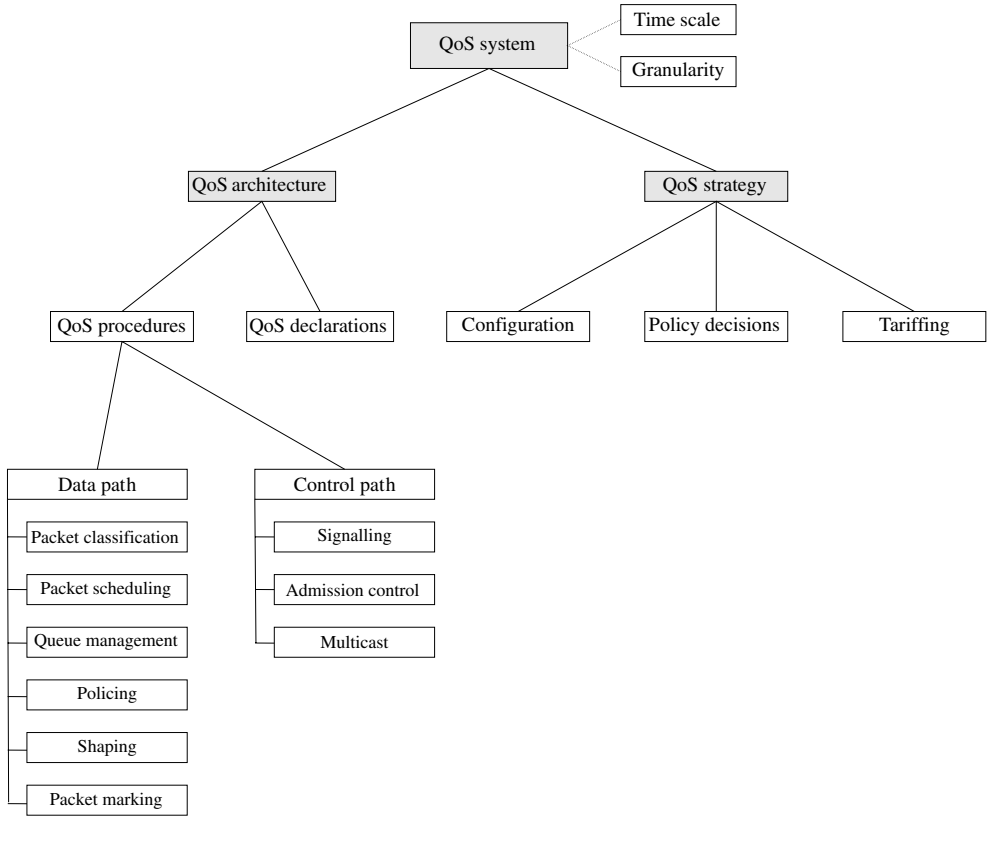

Figure 6.2 QoS System, Based on Schmitt (2001)

that can be built upon these architectures is much larger. This becomes visible for example, in Chapter 8 where more than 20 QoS systems based on three QoS architectures are evaluated.

A QoS architecture can be divided into *QoS declarations* and *procedures*. The QoS declarations form the static part of the architecture and contain properties like service classes, parameters and their specification units. QoS procedures constitute the dynamic part of the QoS architecture and consist of the data and control path mechanisms.

QoS procedures[1] on the control path are signalling, admission control and multicast. Some QoS architectures use a *QoS signalling protocol* as part of the signalling architecture to signal user demands, see Section 6.4.2.

QoS procedures on the data path are packet classification, packet scheduling, queue management, policing, shaping and packet marking. Packet classification is necessary to identify the service class, or flow, the packet belongs to, as that determines the service the packet receives.

If there are several packets competing for a link, then the **scheduling** algorithm decides the order in which these packets are sent. Sending packets on a First-come First-served (FCFS) basis is usually not enough to give delay guarantees or to split the bandwidth in a given proportion among flows or service classes. There are many different families of scheduling algorithms available, for example:

- *Priority* schedulers,
- the *EDF* (Earliest-deadline First) scheduler described in Liu and Layland (1973) and advanced EDF schedulers like *Rate-controlled EDF* as in Zhang and Ferrari (1994),
- Round Robin schedulers like *WRR* (Weighted Round Robin) and *DRR* (Deficit Round Robin, see Shreedhar and Varghese (1996)),
- the PGPS/WFQ family: *PGPS* (Packetised General Processor Sharing) as *WFQ* (Weighted Fair Queueing) (see Demers *et al.* (1989); Parekh (1992)), *SCFQ* (Self-clocked Fair Queueing, see Davin and Heybey (1994)), *FFQ* (Frame-based Fair Queueing, see Stiliadis and Varma (1996)), *SFQ* (Start-time Fair Queueing, see Goyal *et al.* (1997)), *WF2Q* (Worst-case Weighted Fair Queueing, see Bennett and Zhang (1996)),
- virtual-clock schedulers like the original *VC* (Virtual Clock, see Zhang (1990)) and *LFVC* (Leap Forward Virtual Clock, see Suri *et al.* (1997)),
- hierarchical schedulers like *CBQ* (Class Based Queueing, see Floyd and Jacobson (1995)), *HPFQ* (Hierarchical Packet Fair Queueing, see Bennett and Zhang (1997)), *HFSC* (Hierarchical Fair Service Curve, see Stoica *et al.* (1997)),
- and dynamic packet state (DPS) schedulers like *CSFQ* (Core-stateless Fair Queueing, see Stoica *et al.* (1998, 2002)),
- If the parameters of scheduling algorithms are not configured statically but instead adapted automatically based on current measurement information, we speak of *adaptive* variants of scheduling algorithms, see for example, Antila and Luoma (2003, 2004); Christin *et al.* (2002); Liao and Campbell (2001).

Queue management is typically closely connected to scheduling. While schedulers manage the access to an outgoing link's bandwidth, queue management controls the

[1] Contrary to Schmitt (2001) who counts traffic engineering and network design/engineering as (mid-term and long-term) QoS procedures we treat these procedures as part of a separate problem area (see Part IV of this book) because they affect the whole network, not only the QoS system.

buffer space inside a router that is used to store the packets that have not yet been served by the scheduler. If buffer space is running out, packets have to be dropped. Some queue management schemes drop only newly arriving packets (FIFO) while others can also drop already buffered packets, for example, from the head of the queue. Another decision with respect to buffer management is whether the buffer space is split up statically between the different queues.

Besides that, a router can employ *Active Queue Management* (AQM) strategies to actively keep the average queue length small. AQM promises to improve the end-to-end congestion control, to lower queueing delays, more fairness among the flows and buffer reserves for absorbing bursts of packets. This is done by *actively* signalling congestion early. Congestion is signalled by dropping packets or by marking packets if the sender supports *Explicit Congestion Notification (ECN)* as introduced by Ramakrishnan *et al.* (2001).

The classical AQM algorithm is *RED (Random Early Detection)*, see Floyd and Jacobson (1993). It maintains an exponentially weighted moving average of the queue length. When the average queue length exceeds a minimum threshold packets are randomly dropped/marked; if a maximum threshold is exceeded all packets are dropped/marked.

RED has been improved in a number of ways. The sensitivity of RED to parameter settings led to proposals like *Gentle RED* (see Rosolen *et al.* (1999)), *Adaptive RED* (see Floyd *et al.* (2001)), *Stabilised RED* (see Ott *et al.* (1999)) while other works like *Flow Random Early Drop* (see Lin and Morris (1997)), *RED with Preferential Dropping* (see Mahajan *et al.* (2001)) and *CHOKe* (see Pan *et al.* (2000)) aim more at improving the fairness of RED.

Virtual Queue (VQ) approaches like that of Kunniyur and Srikant (2004) maintain a VQ whose capacity is less than the actual link capacity. Packets arriving at the real queue are also accounted for in the VQ. If the VQ overflows, this is taken as congestion indication and packets arriving at the real queue are marked as dropped.

A control theoretic approach to AQM is the *Proportional Integrator (PI)* controller of Hollot *et al.* (2001). It is a based control theory applied to a linearised TCP/AQM model. PI regulates the queue length to a target value (queue reference) using instantaneous samples of the queue length contrary to the moving average of RED that can be influenced largely by past values of the queue length. An improved version of PI is presented by Heying *et al.* (2003).

The *Random Exponential Marking (REM)* AQM scheme of Athuraliya *et al.* (2001) uses a congestion measure labelled 'price'. This 'price' measures the mismatch between packet arrival (demand rate) and departure rates (service rate) and the mismatch between the actual and target queue lengths.

Another approach is called *Blue* by Feng *et al.* (2002); it is based on buffer overflow and link idle events contrary to the average queue length of RED. Several AQM mechanisms are compared and evaluated for example, in Bitorika *et al.* (2004); Le *et al.* (2003).

If a QoS architecture uses reservations or Service Level Agreements (SLA) that specify the amount of traffic a user is entitled to, a mechanism is necessary to control whether an arriving packet is conforming with the agreed traffic specification. The mechanism to detect non-conforming packets is called *policing*. The network can react to non-conforming packets by dropping these packets, by delaying these packets with a *shaper* until they conform or by downgrading the service these packets receive (the latter might require

a *packet marker*). A shaper can also be used at an outgoing link, for example, at an interconnection to make the traffic conformant to a service level agreement with the next interconnection partner or just to smooth out bursts. Packet markers can also be necessary at ingress nodes for example, in Diffserv networks to write the Diffserv CodePoint (DSCP) into the IP header or in routers that use explicit congestion notification (ECN) to mark packets.

We now discuss different QoS architectures and then summarise them by the classification of Section 6.2.7.

6.2.2 The Integrated Services Architecture

6.2.2.1 Overview

The term *Integrated Services Network* was introduced by Scott Shenker. It describes one network for all kinds of applications, especially real-time multimedia traffic like voice, video conferencing and TV like applications. In the early 1990s, the Internet Engineering Task Force (IETF) realised that the Internet's egalitarian best-effort model is not suited for this kind of real-time multimedia traffic if the network is significantly loaded. The IETF's first answer to this problem was the Integrated Services architecture. Later, with the Differentiated Services architecture a second fundamentally different approach was pursued (see following text).

The general Integrated Services (Intserv) architecture is specified in RFC 1633 (see Braden *et al*. (1994)). It builds upon a QoS signalling protocol. The IETF proposed signalling protocol is RSVP. The IETF Intserv specifications can be broken into two parts, the signalling as RSVP part in RFC 2205 (see Braden *et al*. (1997)) and the integrated service specifications in RFCs 2211 and 2212 (see Shenker *et al*. (1997); Wroclawski (1997)); because of this the Intserv architecture is often described as 'RSVP/Intserv'.

Guarantees are given for individual flows, for each flow a path is reserved through the network. A flow is defined as a *distinguishable stream of related datagrams that result from a single user activity and require the same QoS*; it can be seen as a hybrid between the Virtual Circuit model of ATM and the pure datagram model of IP.

The Intserv service model is based on the distinction between real-time and elastic traffic. Elastic traffic is treated as the traditional best-effort traffic. Contrary to Diffserv, no differentiation of the elastic traffic flows is supported. The default service is best-effort; applications using it do not need any modifications.

The real-time traffic is further categorised by whether it is tolerant to loss and whether it is (rate/delay) adaptive. Multicast support was considered vital by the IETF during the development of Intserv and is widely supported by the architecture.

6.2.2.2 Intserv Control Path

Using RSVP, the applications on the end systems request a specific end-to-end QoS for one session from the network. A *session* in the context of RSVP/Intserv is defined by the triple destination IP address, protocol ID and optionally a destination port. As the destination address can be a multicast address, a session is a data flow from possibly

multiple senders to multiple receivers. The reservation process is described in RFC 1633 and 2205 (see Braden *et al.* (1994, 1997)) and depicted in Figure 6.3:

- A sender application announces itself by sending a PATH message to the destination unicast or multicast address. If multicast is used, each receiver must first join the associated multicast group using a multicast group management protocol like Internet Group Management Protocol (IGMP) (see Cain *et al.* (2002); Fenner (1997)) for IPv4 and Multicast Listener Discovery (MLD) (see Haberman (2003)) for IPv6. This, however, is not part of the QoS negotiation process and RSVP.
  The PATH message
  - contains a traffic specification (*TSpec*).
  - establishes path state in the intermediate routers.
    This path state is used for propagating back reservation requests on the reverse path. Unlike more traditional signalling protocols from telecommunication networks, RSVP does not set up an explicit route for the data transmission; this task is left to the routing protocols (see Section 6.4.1).
  - Optionally, the sender may include an advertisement specification (*AdSpec*) in its PATH message in order to advertise to receivers the characteristics of the communication path. On their way downstream, the advertisements accumulate information about the hop count, minimum propagation latencies, minimal individual link bandwidth along the path, the path MTU, service-specific parameters, and whether all routers along the path support RSVP.
- Each receiver individually determines its QoS requirements. Therefore, the whole process of QoS negotiation is called *receiver-oriented*. The decision is obviously based upon the TSpec and the AdSpec of the PATH message but can be influenced by any knowledge about the locally available resources (e.g. maximum resolution of a video display), application requirements, service prices and so on.
- The receiver then initiates the actual reservation process by responding to the PATH message with a RESV that is routed along the previously set up path back to the sender. The RESV message contains:
  - A flow specification (*FlowSpec*) describing
    - the requested service class,

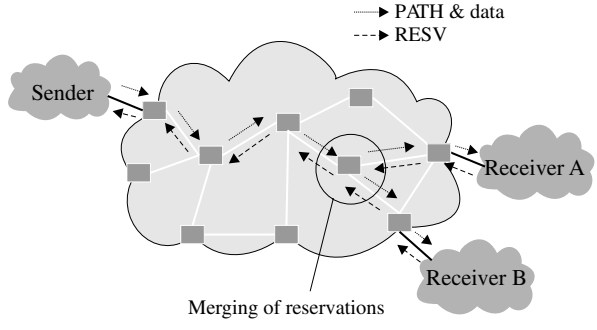

Figure 6.3 Intserv Control Path

    - the specification of desired QoS (*RSpec*), and
    - the description of the data flow (*TSpec*).
  - A filter specification (*FilterSpec*) that identifies the packet subset of the session that has these QoS requirements via the reservation style (see following text).
- Along their way to the sender, the RESV messages have to pass an *admission control* test in each router along the path.
  If admission has to be rejected in one of the intermediate systems, a reservation error is raised and signalled to the receiver with a *RESVERR* message. The receiver can try to initiate another reservation with a less demanding FlowSpec or give up. This QoS negotiation process is called *One-pass with Advertising*, see Shenker and Breslau (1995).
- In the multicast case, a distribution tree is created by *merging* reservations: Multiple receivers indicating a need to receive from the same sender do not install separate reservations. Rather, the largest reservation is granted and the rest are assumed to be using the same resources. Therefore, propagation of a RESV message ends as soon as the reservation encounters an existing distribution tree with sufficient resources.
  Besides having multiple receivers, a multicast group may also have multiple senders. For some applications, for example, video conferencing, where it can be expected that only one person is talking at a time, it is desirable that a resource allocation can be shared among multiple senders of a multicast group. This is supported by the RSVP reservation style. The *reservation style* specifies to which extent intermediate routers may merge the reservation requests from different receivers in the same multicast group. RFC 2205 (see Braden *et al*. (1997)) defines three reservations styles:
  - If the *wildcard filter* is used, all traffic from *all* senders directed to the receiver may be merged.
  - With the *shared explicit filter,* the receiver explicitly identifies the list of senders that share *one* reservation.
  - The *fixed filter* allows for a fixed set of simultaneously transmitting senders; the receiver can specify a set of sources and *for each* of them a certain amount of resources is reserved.
- The state in the intermediate routers is the *soft state*, that is, it times out after a certain period. Therefore, RSVP sends PATH and RESV messages periodically. The PATH refreshments will set up a new path in the case of node and link failures and RESV refreshments can also be used to adapt the resource allocations. Also, the soft state mechanism automatically times out and recovers orphaned reservations.

6.2.2.3 Intserv Data Path

Intserv uses a number of QoS procedures on the data path, see also Figure 6.4:

- Packet Classification
  For each incoming packet, the flow it belongs to and the reservation state associated with it have to be identified from the IP header information at line speeds; multiple fields (destination IP, port, etc.) are used in this classification.

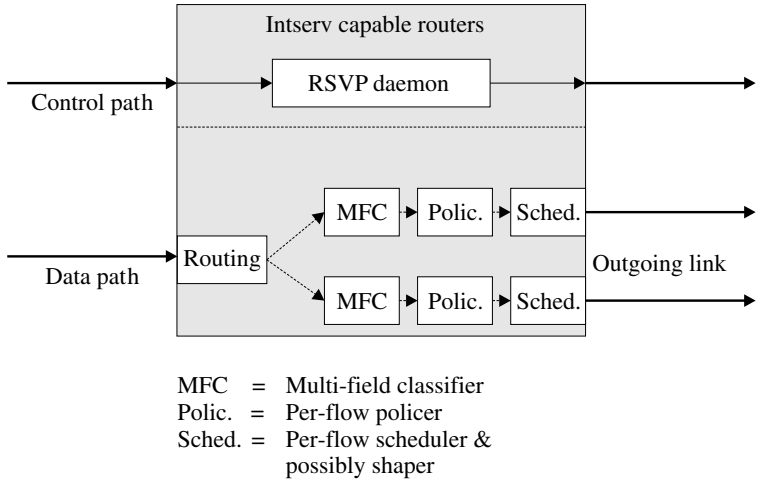

Figure 6.4 Intserv Routers

- Policing
  At least at the edge of the network, policing is necessary to ensure that a host does not violate its promised traffic characteristics.
- Scheduling and Queue Management
  Different queues have to be managed and the packets waiting in the queues have to be scheduled so that the service guarantees are fulfilled. Intserv does not assume one specific scheduler. A wide variety of schedulers can be used, however, the error terms of the scheduling algorithm influence the amount of resources that have to be allocated to provide a certain QoS and thus the efficiency with which a certain service can be provided; see Section 6.2.2.4 for details.
- Shaping
  If multiple senders are used, reshaping is necessary at all points of the multicast distribution tree where traffic from two different sources that share the same reservation merge. Also, at points where the multicast distribution tree from a source branches to multiple distinct paths with differing TSpecs, reshaping is necessary on the outgoing links that have 'lower' TSpecs than the upstream link, see Shenker *et al.* (1997).

6.2.2.4 Intserv Guaranteed Service

Because of the importance of the Intserv guaranteed service (GS) as the 'strongest' service in today's QoS architectures, we now discuss GS in some more detail.

GS offers a deterministic service with zero-loss guarantees and delay bound guarantees: If every router in the flow's path supports guaranteed service (or adequately mimics GS), the flow experiences a delay-bounded service with no queueing loss for all conforming packets. Please note that it does not aim at minimising the jitter.

The Intserv GS is specified in RFC 2212 (see Shenker *et al.* (1997)). Other QoS architectures can be used to provide guaranteed service or at least similar services; see Section 6.2.3 or the central Bandwidth Broker (BB) we developed for the experiments of Chapter 8.

The flow's delay bound d consists of a fixed delay (transmission delay, etc.) d_t and the maximum queueing delay d_q that is a function of the flow's arrival curve and the service function allocated for the flow:

$$d = d_t + d_q \tag{6.1}$$

The mathematical foundation for the GS is the network calculus, see Section 3.2 for an introduction to network calculus.

The arrival curve is given with the TSpec that consists of a token bucket with rate r and buffer b that specifies the flow plus a peak rate p which specifies the maximal rate at which the source may inject bursts into the network and the maximum datagram size M and the minimum policed unit m. The long-term average rate of the flow does not exceed the token-rate r, the maximal burst sent into the network within a short period of length T does not exceed $M + pT$, see Figure 6.5. To assure that a flow conforms to these specifications, policing and reshaping are used.

The service a flow receives at a router is mathematically described by the service curve; it is specified by a service rate R and a latency L; see Figure 6.5. The latency L depends on the scheduling algorithm. The service rate R is specified in the RSpec by the receiver and represents the share of the link's bandwidth the flow is entitled to. Via the service rate R, the receiver can influence the delay bound. If there is a difference between the desired delay bound $\overline{d_q}$ and the bound $d_q = f(R)$ obtained by the chosen service rate R, this difference can be expressed with the slack term S of the RSpec that allows intermediate routers to reduce their resource reservations accordingly. The buffer size B represents the buffer space in the router that the flow may consume.

As the theoretical model behind GS is a fluid model, the rate dependent error term C_l and the rate independent error term D_l of a router (as outgoing link) l are used to express the difference between the fluid model and a real scheduling algorithm operating on packetised data. For WFQ, see Demers *et al.* (1989); Parekh (1992) and other non-preemptive scheduling algorithms, the rate independent error term is given by the delay

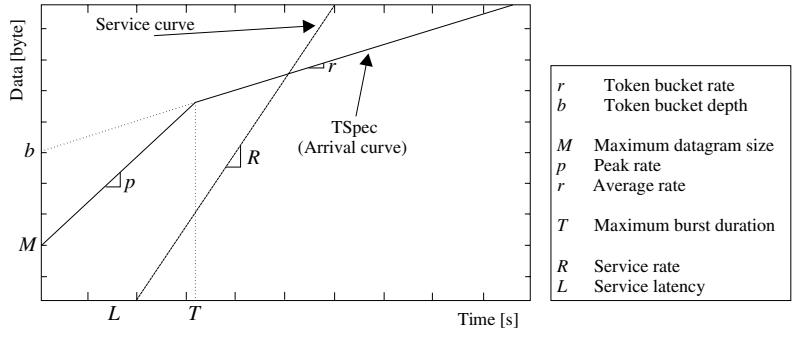

Figure 6.5 Guaranteed Service (Arrival and Service Curve)

caused by a maximum sized packet (size MTU_l) blocking the link with bandwidth bw_l for a conforming packet of the flow that arrives shortly after the maximum sized packet:

$$D_l = \frac{MTU_l}{bw_l} \tag{6.2}$$

The rate dependent error term C_l expresses the backlog of the queueing/scheduling algorithm against a fluid bit-by-bit service, for WFQ it is equal to the flow's maximal packet size M

$$C_l = M \tag{6.3}$$

The scheduler error terms are summed up (e.g. in the AdSpec) to the total error terms $\sum_{\forall l} C_l$ and $\sum_{\forall l} D_l$ for calculating the end-to-end delay. With the maximum burst duration

$$T = \frac{b - M}{p - r} \tag{6.4}$$

the end-to-end delay bound for a given R is given by

$$d_q = \begin{cases} T \cdot \frac{p-R}{R} + \frac{M+\sum_{\forall l} C_l}{R} + \sum_{\forall l} D_l & \text{for } p > R \geq r \\ \frac{M+\sum_{\forall l} C_l}{R} + \sum_{\forall l} D_l & \text{for } R \geq p \geq r \end{cases} \tag{6.5}$$

Besides allocating the rate R, a router l also has to allocate a buffer B_l to ensure no loss. This buffer is

$$B_l = \begin{cases} M + T \cdot (p - R) + \sum_\lambda C_\lambda + R \sum_\lambda D_\lambda & \text{for } p > R \geq r,\ L \leq T \\ M + p\left(\frac{\sum_\lambda C_\lambda}{R} + \sum_\lambda D_\lambda\right) & \text{for } R \geq p \geq r,\ L \leq T \\ b + r\left(\frac{\sum_\lambda C_\lambda}{R} + \sum_\lambda D_\lambda\right) & \text{for } L > T \end{cases} \tag{6.6}$$

with the overall scheduler latency L

$$L = \frac{\sum_\lambda C_\lambda}{R} + \sum_\lambda D_\lambda \tag{6.7}$$

where $\sum_\lambda C_\lambda$ as $\sum_\lambda D_\lambda$ are the error terms summed up from the first hop as the last reshaping point to link l.

If the peak rate p is unknown, it is assumed to be infinite; the arrival curve becomes a token bucket (r, b) and the end-to-end delay bound simplifies to

$$d_q = \frac{b}{R} + \frac{\sum_{\forall l} C_l}{R} + \sum_{\forall l} D_l \tag{6.8}$$

and the buffering required at a router l to

$$B_l = b + \sum_\lambda C_\lambda + \sum_\lambda D_\lambda \cdot R \tag{6.9}$$

6.2.2.5 Intserv Controlled Load Service

The Controlled Load (CL) service is specified in RFC 2211 (see Wroclawski (1997)). It provides flows with approximately the QoS that they would receive using the traditional best-effort service in an unloaded network. Though CL does not provide strict boundaries of QoS parameters like loss and delay, it ensures that a very high percentage of the delivered packets will not experience loss or delay higher than the basic packet error rate as the minimum transit delay along the path. Admission control and policing have to be used to control the amount of CL flows and – obviously – an estimate of the data traffic has to be given (in form of a TSpec).

The CL service allows much more freedom in its implementation than that of the GS. The idea is that the service allows for extremely simple implementations on one side as well as implementations with evolving scheduling and admission control algorithms for a highly efficient use of network resources on the other side.

6.2.2.6 Complexity and Scalability Discussion of Intserv

Two types of per-flow state are needed in Intserv networks:

- Forwarding state to pin the forwarding path of a flow, and
- the FlowSpec/FilterSpec state used by the admission control on the control plane as well as the packet classifier and scheduler of the data plane.

Therefore, each Intserv router has to process per-flow signalling messages, maintain the FlowSpec/FilterSpec tables per flow and perform per-flow packet classification and scheduling. It is obvious that this complexity has its costs, especially in backbone networks with a large number of flows where it can cause scalability issues.

Karsten (2000) and Karsten *et al*. (2001) discuss the scalability of the control path: the complexity of the RSVP daemon and the signalling. These works present and analyse an open-source RSVP implementation (see Karsten (2004)) that with some optimisations like fuzzy timer control can handle 50,000 flows on an off-the-shelf PC with a 450 MHZ Pentium III processor and 128 MB RAM. More importantly, they show that RSVP scales linearly with the number of flows.

There are also some works to reduce the scheduling complexity including proposals that require only constant time complexity, see for example Davin and Heybey (1994); Stephens *et al*. (1999); Wrege and Liebeherr (1997); Zhang and Ferrari (1993), although there is a natural trade-off between the complexity of a scheduler and its flexibility as discussed in Knightly *et al*. (1995).

Also, in packet classification there have been recent and very promising advances, see for example Gupta (2000); Singh *et al*. (2003); Srinivasan and Varghese (1999b) and the works therein[2].

There are proposals to reduce the amount of state via reducing the number of flows by aggregating micro-flows that follow the same path through the network into one macro-flow, see for example Baker *et al*. (2001).

[2] Packet Classification is also necessary for IP routing lookups, see Section 6.3.1.1.

A careful analysis shows that the scalability of an Intserv network is not as critical as often assumed but still has to be taken seriously. Therefore, we next investigate a number of proposals that have a focus on reducing the state complexity compared to the Intserv/RSVP QoS architecture.

6.2.3 Stateless Core Architectures

6.2.3.1 Overview

Because of the scalability concerns with Intserv especially in backbone networks, considerable research went into analysing stateless core (SCORE) QoS architectures. The general idea is to have a network where only edge routers have to perform per-flow management while core routers do not. The IETF QoS architecture Diffserv is an example of the SCORE idea. Besides Diffserv, there are some other proposals, the most famous one is based on Stoica (2000) and called *Dynamic Packet State* (DPS). Because of the importance of Diffserv, we discuss Diffserv in a separate section and focus here on DPS.

The basic idea of DPS as described in Stoica (2000) and Stoica and Zhang (1999) is that the ingress (edge) router inserts information into the IP header. This information is used and updated by the core routers to provide deterministic service guarantees like Intserv's GS. The core routers are using a special scheduling mechanism that only depends on the DPS and does not require per-flow state on the data path. In addition, the control path is made stateless in the core as the aggregate reserved rate needed for admission control at one link can be derived from the packet state, too. We now discuss the data path and the control path of DPS before addressing related works and applications.

6.2.3.2 SCORE Data Path

Stoica (2000) and Stoica and Zhang (1999) present the DPS technique and a *Core-jitter-virtual-clock* scheduling algorithm that can approximate Intserv GS without requiring per-flow state on the data path; it is a combination of a delay-jitter rate-controller and a VC scheduler. The algorithm works as follows:

- Each flow is assigned a rate r that is stamped into the packet header and thus does not have to be stored by the core routers.
- In a router, each packet is assigned an eligible time and a deadline upon arrival. A packet is not sent before its eligible time; the scheduler is thus not work-conserving. The next packet to serve is chosen among all eligible packets according to their deadlines (earliest deadline first).
  - One goal of the algorithm is to send packets close to their deadline but not after the deadline, thus incurring the maximal allowed delay and reducing jitter. The extent of time a packet is transmitted before its actual deadline in a node (the local fluctuation) is stamped into the packet header. In the next node the packet is not eligible unless that time has passed, thus the local fluctuation of node n is balanced out at node $n + 1$.

- The eligible time has to be at least as high as the deadline of the previous packet belonging to the same flow. Normally, a scheduler would have to keep per-flow state information to track the deadlines but by introducing a third variable that is stamped into the packet header at the edge (where per-flow state is allowed) and that is updated at each hop, this per-flow state can be eliminated, too. The eligible time at a node n can now be calculated as the sum of the arrival time, the local fluctuation of the previous node and the new slack variable. The slack variable effectively introduces an additional delay for a packet at each hop making sure that a packet is not sent before the deadline of the previous packet. Stoica and Zhang (1999) and Stoica (2000) show that this delay does not increase the overall delay compared to the non-SCORE version of the scheduling algorithm (which does not use the slack variable and instead keeps per-flow state).
- The deadline of a packet is the eligible time plus the time it takes to transmit the packet with rate r assigned to the packet's flow (as encoded in the packet header).

Stoica and Zhang (1999) and Stoica (2000) propose using the IP header to store the DPS but it is also imaginable to add a new header between layers 2 and 3 as in MPLS. These works also show that due to the traffic regulation of the scheduling algorithm, the number of packets in the server at any given time is significantly smaller than the number of flows, which further reduces scheduling complexity. The works further show that DPS can give the same guaranteed service as a network of routers with a WFQ scheduler – the 'typical' Intserv scheduler.

6.2.3.3 SCORE Control Path

The SCORE DPS approach assumes that RSVP messages are only processed by the edge nodes; inside the network a lightweight signalling protocol is used.

Before admitting a new flow, the admission control module at each node has to check whether the aggregated reserved rates and the new rate do not exceed the outgoing link's capacity. Just counting the aggregated reserved rates for each outgoing link without keeping track of the flows is no robust solution because flows can stop sending without notice, for example, because the sender crashed, without the system being able to free the resources again. So, normally the per-flow state would be needed to keep track of the rates of the already accepted flows and to be able to recover from the errors.

However, by introducing a fourth variable that is stamped into the packet header by the ingress node an upper bound of the aggregate reserved rate can be derived; thus, the control path per-flow state can also be eliminated. This variable contains the amount of data that the flow to which the packet belongs to was entitled to send according to its reserved rate since the previous packet. A node can add up these variables of *all* packets traversing a link for a certain period to get an estimate of the aggregate reserved rate on that link. The actual mechanism is more complex because it has to account for jitter, termination and other aspects; it is fully described in Stoica (2000); Stoica and Zhang (1999).

The DPS approach also depends on a route pinning mechanism like MPLS or the label-based one in Stoica (2000).

6.2.3.4 Related Works

The DPS approach can also be used to provide stateless flow protection or relative service differentiation in a network, for details see Stoica (2000); Stoica *et al*. (1998). Kaur and Vin (2003) describe a work-conserving core stateless scheduler for throughput guarantees.

A centralised admission control approach, contrary to the hop-by-hop approach above, is proposed in Zhang *et al*. (2000) for SCORE networks with DPS. It relieves core routers from the admission control functionality. A centralised BB is used instead that keeps track of the QoS reservation states. Besides that, the admission control is generalised with the virtual time reference system towards other types of schedulers and to class-based guarantees in Zhang *et al*. (2000). A methodology to transform any guaranteed rate per-flow scheduling algorithm into a SCORE version is presented in Kaur and Vin (2001).

In a SCORE architecture based on DPS, core routers have to trust the information carried in the packet headers; a single faulty router can disrupt the service in the entire core, therefore these solutions are not very robust. In Stoica *et al*. (2002), an enhancement of the SCORE fair queueing algorithm of Stoica *et al*. (1998) is presented. Core routers no longer blindly trust the incoming packet state (the rate estimates). Instead, they statistically verify and contain flows whose packets are incorrectly labelled.

To summarise, the SCORE approach with dynamic packet state (DPS) presents an interesting approach to provide guaranteed service or other services without having to keep per-flow state in core routers. This comes at the cost of additional fields used in the IP header or a shim header that has to be updated in each hop. The updates require relatively complex[3] operations, and expensive write access at high-speed routers.

6.2.4 The Diffserv Architecture

6.2.4.1 Overview

The Diffserv architecture is specified in RFC 2475 (see Black *et al*. (1998)), the Diffserv field in the IP header in RFC 2474 (see Nichols *et al*. (1998)). Diffserv can be seen as the IETF's response to the concerns about the complexity of Intserv/RSVP. Diffserv takes a more abstract and local view on resource allocation. It is a SCORE approach, the core nodes of a network do not have to keep per-flow state. Per-flow state is kept at edge nodes only where operations like policing and marking are also done exclusively.

On the data path, packets of different flows are aggregated into behaviour aggregates (BA) at the edge nodes. A BA is associated with a certain service class; it is identified by the six bit Diffserv CodePoint DSCP . The DSCP is contained in the Diffserv field[4] of the IPv4 IP header or the traffic class octet of the IPv6 IP header.

The heart of the Diffserv architecture is the Per-hop Behaviour (PHB) that specifies the forwarding behaviour of one router for packets of a DSCP that is locally mapped to that PHB. The edge-to-edge behaviour in a network of one service class – called *Per-domain Behaviour* (PDB) in the Diffserv terminology – results from the concatenation of PHBs. It is assumed that useful services can be constructed from the different PHBs in the standardisation process. The service construction process is mostly left to the INSPs.

[3] Compared to the standard write operations in IP routers: decreasing the hop count and updating the checksum, see also Section 6.3.1.

[4] The Diffserv field was called *type of service* byte before Diffserv was being standardised.

A Diffserv domain is a network over which a consistent set of differentiated service policies are administered in a coordinated fashion – typically, this equals the network of a single INSP. As a flow will typically pass several Diffserv domains for end-to-end QoS, a coordination of those is required. This coordination is done on the control path by the use of SLA and potentially BBs.

Figure 6.6 shows the main functionality of Diffserv edge and core routers. The ingress edge router of a Diffserv domain performs several operations on a packet arriving from outside the Diffserv domain:

- A micro-flow classification[5] is necessary to identify the flow, as flow aggregate to which the packet belongs, and to look up the associated traffic conditioning specification and the traffic profile (see following text).
- For most services, further processing by the traffic conditioning module is necessary. Depending on the service, packets are metered, marked, shaped and/or dropped.
  - A meter measures the traffic stream against the traffic profile and can influence the traffic conditioning actions that follow.

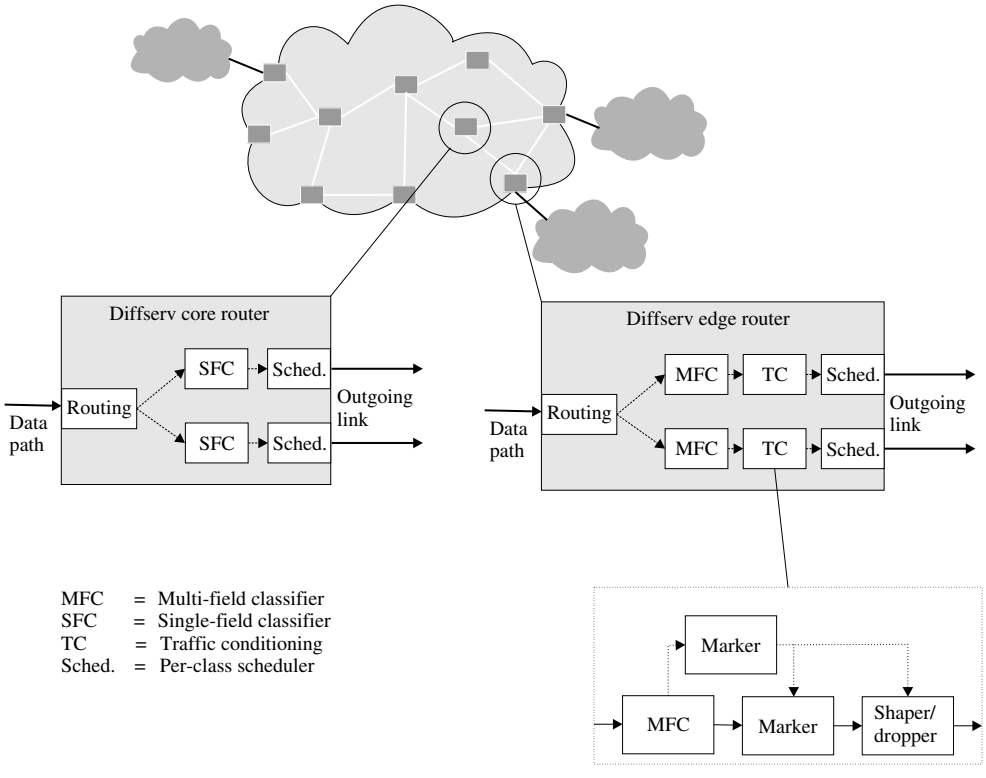

Figure 6.6 Diffserv Edge and Core Routers

[5] This is a multi-field classification based on the value of one or more IP header fields such as source address, destination address, and so on.

- The marker imprints a DSCP on the packet.
- Finally, traffic shaping may be applied to bring the micro-flow into compliance with a traffic profile. Alternatively, a dropper may be used to discard some or all out-of-profile packets.

The exact handling of in-profile and out-of-profile packets is described in the service level specification (SLS) that is part of the service level agreement (SLA), see following text.

- The buffer management and scheduling algorithm in the edge node treats the packet according to its DSCP and the PHB it has locally mapped to that DSCP.

When a packet arrives at a Diffserv core router, the operations are of less complexity:

- Only a single-field classifier is necessary; it reads the DSCP from the Diffserv field and determines the PHB that is locally mapped to that DSCP.
- The buffer management and scheduling algorithm treat the packet according to the PHB.
- Some optional functions like traffic shaping or packet remarking may be used at core routers.

6.2.4.2 Diffserv Services

As mentioned above, edge-to-edge services (Per-domain Behaviours, PDBs) are built by concatenating PHBs. As the number of the PHBs is limited, the queueing and scheduling complexity can be kept low in a Diffserv router. The packet classification in core routers is also relatively simple as the PHB is encoded in the DSCP that is stored in a single field (Diffserv field) in the IP header.

Per-hop Behaviours (PHBs) Besides a default PHB that corresponds to traditional best-effort forwarding, the following PHBs have been specified so far by the IETF:

Class Selector (CS) The CS PHB group is specified in RFC 2474 (see Nichols *et al.* (1998)) and consists of eight classes. CS is mainly intended for backward compatibility with the old IPv4 precedence bits contained in the type of service octet that is now used as the Diffserv field. Contrary to the assured forwarding PHBs (see following text), the CS precedence classes have an ordering with respect to timely forwarding: CS codepoints with a higher relative order have an equal or higher probability of timely forwarding than CS codepoints with a lower relative order.

The CS PHB can be used for relative service differentiation as it is discussed in Dovrolis and Ramanathan (1999). Contrary to an absolute service differentiation scheme where admission control is imposed on users and an admitted user receives absolute performance levels in a relative service differentiation scheme no admission control is necessary; performance guarantees are only given relative to the performance of other classes: a higher class will receive the same or better service than a lower class.

In the proportional differentiation model of Dovrolis and Ramanathan (1999), the INSP assigns a quality differentiation parameter c_i to each class i of his network. While no absolute performance levels are given for short-term performance measures p_i like the loss rate or the queueing delay, the ratio between all classes i and j is controlled by $p_i/p_j = c_j/c_i$. To achieve this ratio, special scheduling and queue management algorithms are necessary, see Dovrolis and Ramanathan (1999).

Expedited Forwarding (EF) The EF PHB was originally specified in RFC 2598 (see Jacobson *et al.* (1999)) and later refined to a more rigorous definition in RFC 3246 (see Davie *et al.* (2002) and for more information also Armitage *et al.* (2002); Charny *et al.* (2002)). It can be used to build a low loss, low delay, low jitter, assured bandwidth service that is called *virtual leased line service* or *premium service* (see Nichols *et al.* (1999)). To provide this service, it is necessary that the aggregate traffic experiences no or at least only very small queues. To achieve this, it is necessary to ensure that the service rate R of EF packets on a given output interface exceeds their aggregate arrival rate A at that interface over long and short time intervals, independent of the amount of other (non-EF) traffic at that interface. It is difficult to define the appropriate timescale at which to measure the service rate R because too small timescales may introduce sampling errors and too large timescales may allow excessive jitter. Also, if there are not enough packets arriving at a queue in a certain interval – externally this might not be obvious – the service rate R cannot (obviously) be obtained. Because of these reasons the formal definition of EF calculates the ideal departure time of an arriving packet by assuming that it is served with rate R either immediately upon arrival or upon the departure time[6] of the previous packet. The deviation of the real departure time of a packet from the ideal departure time is bounded by an error term E. The scheduling requirements of the EF PHB are stricter than the service curve of a rate-latency scheduler[7]. A number of scheduling algorithms satisfy the EF requirements but differ in their error terms, for example:

- a strict non-preemptive priority scheduler where EF has priority over the other classes;
- worst-case fair weighted fair queueing (WF2Q), SFQ and SCFQ;
- DRR.

The EF PHB is intended for low loss services, RFC 3246 (see Davie *et al.* (2002)) leaves it optional to specify a region of operation for an EF node where no losses occur. If this is not used it means that in an RFC conformant operation of an EF node a limited number of EF packets can be dropped due to limited buffers.

For deterministic service guarantees, the worst-case aggregate arrival rate has to be bounded. However, the aggregate arrival rate depends on the topology and the routing through the Diffserv domain. Charny and Le Boudec (2000) derive a delay bound for general topologies that is a function of the maximal link utilisation α and the maximal number of hops h of a flow (as the network diameter); it is named *Charny Bound* after the first author.

For the general assumptions, the delay experienced by a single packet depends not only on the behaviour of the flows sharing at least one queue with the packet, but also on the behaviour of flows in the other parts of the network and potentially on past flows as shown in Charny and Le Boudec (2000).

It is assumed that the incoming flows are characterised by a leaky bucket and on each link l of capacity C_l, the aggregate rates are bounded by αC_l and the aggregate bucket depths by τC_l. The maximum packet size is *MTU*. For link l, a bound on the peak rate of

[6] The real or the ideal departure time of the previous packet, whichever is later.

[7] If a scheduler satisfies the EF requirements it also satisfies the rate-latency curve but not necessarily vice versa.

all incoming flows is given by Γ_l. If no further assumptions about the routing are made, Γ_l is given by the bit rates of all incoming links or if they are unknown $\Gamma_l = \infty$.

For $\alpha < \min_l \frac{\Gamma_l}{(\Gamma_l - C_l)(h-1)+C_l}$, a bound on the worst-case end-to-end queueing delay for EF traffic is

$$d_q = \frac{h}{1-(h-1)u\alpha}(\Delta + u\tau) \tag{6.10}$$

with

$$u = \max_l \frac{\Gamma_l - C_l}{\Gamma_l - \alpha C_l} \tag{6.11}$$

$$\Delta = \max_l \frac{MTU}{C_l} \tag{6.12}$$

This bound is depicted in Figure 6.7 for different maximal hop counts h assuming a topology with a maximum in-degree of 5, a capacity C_l=155 Mbps, MTU=1500 bytes, EF flows with an average rate of 64 kbps, and a maximum burst size (bucket depth) of 600 bytes.

As can be seen, the delay bound explodes if α approaches $\min_l \frac{\Gamma_l}{(\Gamma_l - C_l)(h-1)+C_l}$; this does not mean that the delay is necessarily unbounded for these cases. It is possible that a better (lower) delay bound can be derived, yet at the time of writing no other delay bound with the same general, assumptions as the Charny bound is known to the author. Besides that, it has been shown in Charny and Le Boudec (2000) that for larger α there exists a large enough network such that the worst-case delay of some packet can exceed any D, even if the maximum hop count never exceeds h.

On the basis of this delay bound, a medium-sized network could only be utilised little more than 10% with EF traffic.

Improvements of the Charny bounds can be obtained if additional mechanisms are added to the Diffserv network; e.g. traffic shaping, see for example Cruz (1998); Fidler (2003); Ossipov and Karlsson (2003).

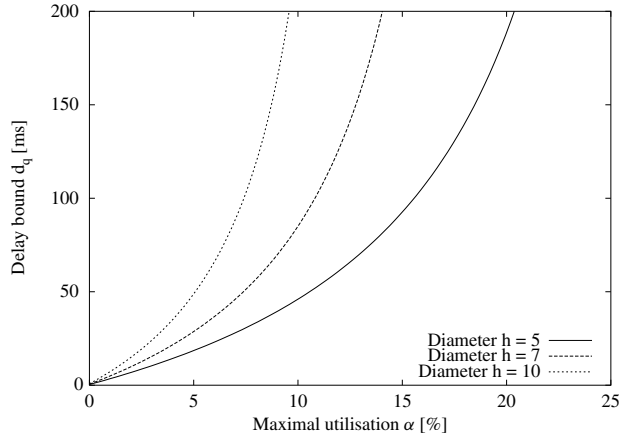

Figure 6.7 Charny Bound

Assured Forwarding (AF) The assured forwarding PHB group is specified in RFC 2597 (see Heinanen *et al.* (1999)). It consists of four independently forwarded AF classes. Within each class, packets are marked with one of three different levels of drop precedences; they are often called *green*, *yellow* and *red*. The drop precedence determines the relative importance of the packet within the AF class. More AF classes or levels of drop precedence may be defined by INSPs for local use. There are no delay or delay variation requirements associated with the forwarding of AF packets. An active queue management algorithm such as RED (Random Early Detection, see Floyd and Jacobson (1993)) is required, but no details about the algorithm are prescribed by the RFC except that flows with different short-term burst shapes but identical longer-term packet rates should have packets discarded with essentially equal probability.

There is no specific order between the AF classes. At each node, a certain amount of forwarding resources (bandwidth and buffer) is assigned to each class. Typical implementations could use admission control to limit the load in the different classes to different levels, for example, by overbooking the classes with different overbooking factors. This, however, is not detailed by the RFC. An overbooking factor is the ratio of the maximal admitted traffic of one class to the resources assigned to that class.

Within a single class, packets are not reordered. A typical implementation could assign each class a different queue and use different RED weights for the different drop precedences within each class.

RFC 2597 (see Heinanen *et al.* (1999)) describes as an example an Olympic service that can be built with the AF PHB group in the following way: The Olympic service consists of the three service classes, bronze, silver and gold that are assigned to the AF classes 1, 2 and 3. Packets in the gold class experience lighter load and thus have greater probability for timely forwarding than packets assigned to the silver class. The same kind of relationship holds true for the silver and the bronze class.

Within each class, all three drop precedences could be used. Packets are marked at the ingress by the traffic conditioner module. Many different marking algorithms could be used, the most common ones designed with the AF PHB in mind :

- RFC 2697 (see Heinanen and Guérin (1999a)) describes with the *Single Rate Three Colour Marker* a way to mark packets according to three traffic parameters: *Committed Information Rate, Committed Burst Size, and Excess Burst Size*. A packet is marked green if it does not exceed the committed burst size, yellow if it does exceed the committed but not the excess burst size, and red otherwise. It is useful for ingress policing of a service, where only the length, not the peak rate, of the burst determines service eligibility.
- The *Two Rate Three Colour Marker* of RFC 2698 (see Heinanen and Guérin (1999b)) describes a way to mark packets based on two rates, the *Peak Information Rate* and the *Committed Information Rate*. A packet is marked red if it exceeds the peak information rate. Otherwise, it is marked either yellow or green depending on whether it exceeds or does not exceed committed information rate. It is useful for ingress policing of a service, where a peak rate needs to be enforced separately from a committed rate.
- Packet marking based on the running average bandwidth of the traffic stream compared to the *Committed Target Rate* and the *Peak Target Rate* is described by RFC 2859 (see Fang *et al.* (2000)), and called *Time Sliding Window Three Colour Marker*. Packets

contributing to the sending rate below or equal to the committed target rate are marked green, those contributing to the rate between committed and peak target rates are marked yellow, the others red. Because of the sliding window rate estimator, burstiness is taken into account and smoothed out to approximate the longer-term measured sending rate of the traffic stream.

Achieving service differentiation with profile-based marking in edge routers is not a straightforward task, especially for a mix of responsive (TCP) and non-responsive (e.g. UDP) flows. Several studies investigate these effects, for example Clark and Fang (1998); Feng *et al.* (1999); Nandy *et al.* (1999); Sahu *et al.* (2000); Stoica and Zhang (1998); Yeom and Reddy (1999).

Per-domain Behaviours (PDBs) Building services and PDBs out of the PHBs is mainly left to the INSPs. There are several Internet drafts describing PDBs but few reached RFC status. The *virtual wire PDB* (see Jacobson *et al.* (2000)) can be constructed with the EF PHB plus appropriate domain ingress policing. As the name says, it is intended to provide a service that behaves like a dedicated circuit by providing an assured peak rate and bounded jitter. The EF PHB efficiency concerns discussed above lead to efficiency concerns about his PDB, too.

The *assured rate PDB* (see Seddigh *et al.* (2000)) is intended to provide a rate assurance but no delay or jitter bounds. It is built with the AF PHBs and suitable policers at the ingress.

A different approach is taken with the *bulk handling PDB* as *lower-effort PDB* of Bless *et al.* (2003); Carpenter and Nichols (2001) that provides a less-than-best-effort service. This service may be 'starved' by other services (including the standard best-effort service) in times of congestion as high load and is intended for low value traffic. The effect that the low value traffic has on other traffic is limited. The CS or AF PHBs can be used to implement the service; policing at the ingress is not required.

Service Level Agreements (SLAs) Between a customer (INSP or end-user) and an INSP operating a Diffserv domain, *Service Level Agreements* (SLAs) are used to specify the service the customer receives. For more information and a general (non-Diffserv) example see Section 9.2.4.

SLAs contain a *Service Level Specification* (SLS). A SLS is a set of parameters and their values that together define the service offered to the traffic by a Diffserv domain as long as it adheres to the *Traffic Conditioning Specification* (TCS) which is an integral part of the SLS. The TCS is a set of parameters and their values which together specify a set of classifier rules and a traffic profile.

SLAs can be dynamic and static, see Figure 6.8. Static SLAs remain in existence and constant on a medium to long timescale; typically, they are set up and maintained manually. Dynamic SLAs change more frequently, and typically, they are negotiated and set up automatically by BBs (in the Diffserv environment), see Nichols *et al.* (1999). With dynamic SLAs, networks can be used more efficiently unless traffic patterns are very stable and constant.

Possible SLS formats for Diffserv Premium service can be found in Bouras *et al.* (2002); Hashmani *et al.* (2001) and the works therein. For SLA trading, we refer to Fankhauser (2000).

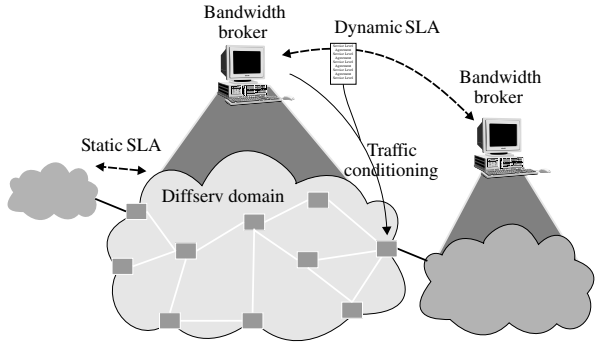

Figure 6.8 Diffserv Service Level Agreements

Preferential Treatment of Acknowledgment Packets TCP flows are bidirectional with data packets being transported on one path and Acknowledgement (ACK) packets being transported back on the opposite path. The TCP throughput suffers when either of the two paths is congested. Because the packets take different paths and enter a Diffserv domain at different ingress nodes, they can receive completely different services. Papagiannaki *et al.* (2001) analyse in which ways ACKs have to be marked so that connections can achieve their performance goal despite congestion on one or both of the paths. In that study, the throughput of AF flows increased by 20% when the ACKs were sent with the highest service class over the throughput achieved when ACKs are marked as best-effort. The recommendations of that study in a nutshell are that ACKs should receive the same class of service as their data packets.

6.2.4.3 Bandwidth Broker

BBs for Diffserv were introduced in Nichols *et al.* (1999). A BB is *a software agent that manages the network resources of a Diffserv domain and makes the admission control decision for it.*

A BB needs inter-domain communication with other BBs to negotiate SLAs and intra-domain communication to allocate resources and configure ingress routers according to the TCS of a new SLA. For intra-domain communication, network management protocols like SNMP can be used. For inter-domain BB communication (and SLA trading), several protocols are under discussion: Border Gateway Protocol (BGP) extensions, RSVP extensions, the Internet Open Trading Protocol (IOTP), the BB transfer protocol, DIAMETER, Common Open Policy Service (COPS), and SNMP.

Schelen *et al.* (1999) describe a BB implementation that obtains a topological database through the Open Shortest Path First (OSPF) routing protocol, obtains link bandwidths through SNMP. The performance of the admission decision speed is evaluated. Other BB implementations are presented in Pop *et al.* (2001); Stattenberger and Braun (2003); Terzis *et al.* (1999a).

The Internet2 QBone group was working on a BB for an EF-based premium service that was later dropped in favour of a lower than best-effort service, largely due to deployment and other problems; see Teitelbaum *et al.* (1999), Teitelbaum and Shalunov (2002), and also Section 6.2.5.3.

In Fidler (2003), a BB is proposed that influences the routing of the Diffserv domain to give deterministic delay bounds.

In Chapter 8, we present and evaluate the design of centralised and decentralised BBs with focus on the admission control and efficiency.

6.2.5 Tuned Best-effort Architectures

In this section, we describe several approaches that we call "tuned" best-effort because they use the best-effort architecture as a foundation and try to improve QoS or offer service differentiation with relatively little changes to it.

6.2.5.1 Overprovisioned Best-effort

The currently dominating approach to improving QoS is adding bandwidth and buffer to a best-effort network. This approach is called the *overprovisioned best-effort* and is based on the fact that packets travelling through a relatively lightly loaded network experience little to no loss and little queueing delay – therefore relatively good QoS. For many applications, this can be enough.

The advantage of this solution is that the basic QoS architecture does not have to be changed. A disadvantage is that in the absence of admission control and service differentiation, an INSP cannot give (absolute or relative) service guarantees. Also, an INSP cannot offer technically different services with that approach. The latter can be necessary, for example, to use price discrimination as a means of increasing profits.

An important question to ask in the context of overprovisioning is *how much overprovisioning is needed?* Except for the work of Breslau and Shenker (1998), this question is not well answered in the scientific literature on an architectural abstraction level. Therefore, we address this question in the next two chapters of this book by comparing an overprovisioned best-effort architecture with other QoS architectures.

6.2.5.2 Price-controlled Best-effort

The Price-controlled Best-effort (PCBE) approach goes back to the works of Frank Kelly and others; see Kelly (2000, 2001a); Kelly *et al*. (1998). PCBE is one possibility to realise congestion pricing, see Henderson *et al*. (2001). The basic idea behind **congestion pricing** is that if congested network resources are priced, there is an incentive for users to back off in the case of congestion and thus reduce the congestion.

Smart Market An *economically* efficient method to implement congestion pricing is described in MacKie-Mason and Varian (1995) and is called *smart market:*

Sending packets in an uncongested network is free of charge, but packets sent in a congested network are charged on a per-packet basis. Thus, the price to send a packet can vary minute-by-minute (or on even much shorter timescales) to reflect the current degree of network congestion.

MacKie-Mason and Varian propose an auction mechanism to realize this smart market:

- The sender puts the amount of money he is willing to pay (willingness-to-pay) for the transmission of the packet in the header of the packet.

- The network then admits all packets with a willingness-to-pay higher than the current cut-off amount, which is determined by the marginal congestion cost imposed by the next additional packet. Rejected packets could be bounced back to the user, or be routed to a slower network. Users then pay the market-clearing price for all transmitted packets, not their bidding price (second price auction).

The outcome of this mechanism is the classic supply-equals-demand level of service of economic theory that maximises social welfare. Social welfare is the total utility of all end-users and providers[8]. MacKie-Mason and Varian (1995) also show that the congestion revenues equal the optimal investment in capacity expansion.

Unfortunately, it is *technically impossible* to hold auctions on a per-packet basis:

- First, the Internet is not and cannot be managed centrally, therefore there cannot be a central auctioneer realising the auctions.
- A packet normally has to cross several boundaries between providers. If each provider is implementing an auction mechanism (e.g. to avoid the point above), there is another problem: If the auctions along the way are held in serial order (one after the other), the end-user might end up winning and paying most auctions along the way but loosing the auction at the last provider. This could be avoided if the auctions are held in parallel order but this again needs a central instance for coordination. Further discussions on these problems and possible solutions can be found in Courcoubetis and Weber (2003); Courcoubetis *et al.* (2001).
- Hard work is undertaken to make networks faster and faster. Collecting the bids, calculating the clearing price, selecting the packets to be forwarded etc. all add additional delays to the treatment of each packet in each router and requires additional buffer memory in the routers. This can be unacceptable for time critical real-time applications and very expensive for high-bandwidth links.
- The charging and accounting effort of this approach is also extremely high.

Despite these strong concerns about the technical feasibility of auction mechanisms, there is a long line of works about auctions for packets, micro-flows or higher aggregates in the scientific community, see for example, Courcoubetis and Weber (2003); Courcoubetis *et al.* (2001) and the works therein.

PCBE The smart-market approach above has one appealing property: Resources are allocated according to the willingness-to-pay of the customers, leading to proportional fairness weighted by the willingness-to-pay. It has been shown in Kelly *et al.* (1998) that the additive increase-multiplicative decrease rate control of TCP also leads to proportional fairness. If the TCP congestion control is modified by introducing a weightage resembling the willingness-to-pay, weighted proportional fairness could be achieved, too. Such a proposal is made and studied in Crowcroft and Oechslin (1998), using a TCP implementation called *MulTCP*; unfortunately, it leads to difficulties in charging, accounting and policing. A quite similar approach without some of these difficulties is the price-controlled best-effort approach based on Kelly (2000, 2001a); Kelly *et al.*

[8] Costs of providers are counted as negative utility.

(1998). These works describe and model a QoS architecture that leads to the same results as the smart-market approach but that can be implemented technically more easily and in a distributed way. We call this QoS architecture *price-controlled best-effort* (PCBE); it is sometimes also called *ECN charging*, see for example Briscoe *et al.* (2003).

Architecture Technically, the PCBE-architecture is based on a plain best-effort architecture with the ECN mechanism. If congestion is building up in a router, it notifies the end systems by randomly marking packets. Marking is done by setting a single bit[9] in the IP header. The receiver can notify the sender of the fact that it received a marked packet for example, via the TCP ACK. Upon notification, the "normal" reaction of a TCP sender would be to reduce the TCP congestion window in the same way as if the packet was dropped. If PCBE is used, the sender and/or receiver instead have to pay a certain (very small) amount of money for each ECN mark generated as received. Thus, a dynamic price for a stream of packets has to be paid. If that price exceeds the willingness-to-pay of a user, he (or an agent acting on his behalf) will back off by reducing his sending rate. Vice versa, if the willingness-to-pay exceeds the current price, a user will continue to send or even increase his rate. Obviously, this leads to an economically more efficient resource distribution than when all users would be forced to back-off, independent of their willingness-to-pay.

Modelling The PCBE approach has been mathematically modelled in Kelly (2000, 2001a); Kelly *et al.* (1998). The system can be modelled as an optimisation problem and it can be shown that under certain assumptions – for example, increasing concave and differentiable utility functions – it maximises social welfare. Hansen and Naevdal (2000) shows that the resulting system can be treated well with standard economic theory; the work also analyses the resulting loss in social welfare if a (realistic) monopolistic control of network resources is assumed.

Discussion Contrary to the 'typical' QoS architecture (e.g. Intserv and Diffserv) that focus on managing the available resources and thus the 'supply side of QoS', PCBE influences the 'demand side' by giving incentives to impose self-admission control. PCBE follows the end-to-end principle of the Internet which is keeping the network simple by putting the intelligence into the edges of the network.

PCBE maximises social welfare under the assumptions made. However, a competitive INSP is more interested in maximising his medium to long-term profits and not social welfare. This approach might not be ideal under that assumption.

The above described single-bit marking algorithm is a minimalist approach to QoS that does not require many changes in routers – most routers support ECN marking anyway. Significant changes, however, are necessary at the end systems and the INSP's charging and accounting systems. On the end systems, agents (called *dynamic price handlers*) are needed to react to the fluctuating prices on the users' behalf. Such an agent was developed in Briscoe *et al.* (2003). In the same work, the feasibility of a per-marked-packet charging and accounting system was shown.

[9] the ECN congestion experienced bit.

While PCBE is an innovative lightweight approach to QoS, it also has some significant drawbacks:

- Under the more realistic assumptions of monopolistic control of routers, social welfare is no longer maximised, see Hansen and Naevdal (2000). Also, if utility functions are not concave this goal is not met.
- The scheme needs a per-marked-packet accounting system that is potentially expensive.
- The price is dynamic and not known in advance to the user. User trials in Edell and Varaiya (1999) show that end-users will probably not like this situation although other later trials in the context of Briscoe *et al.* (2003) indicate a certain interest in dynamic pricing.
  To avoid dynamic prices for end-users, so-called guaranteed stream providers as brokers have been proposed to offer a kind of insurance service against dynamic prices, see Briscoe *et al.* (2003); Key (1999).
- It is not clear whether the sender or receiver of a marked packet should pay. On one hand, typically the receiver has the benefit of the data transfer. Therefore, it makes sense that he should pay if the transfer causes congestion. But this gives denial-of-service attacks a chance of inflicting direct economic damage on a receiver.
- An INSP is earning money with congested routers. The theory behind PCBE assumes that strong competition between INSPs and a transparent market forces them to upgrade their equipment where necessary and not to cheat on customers. This is not fully convincing, especially as there is hardly a possibility for a customer to control whether a packet was marked correctly.

6.2.5.3 Lower than Best-effort Service

While most QoS architectures and service models aim at introducing additional services that offer a better service than the traditional best-effort service, an interesting approach is to try to do the opposite: Offering a lower than best-effort service.

Carlberg *et al.* (2001) describe an implementation and experiments of this approach realised on a per-flow basis. Contrary to other approaches, such as the above-mentioned Diffserv bulk handling / lower-effort PDBs (see Bless *et al.* (2003); Carpenter and Nichols (2001)) or the QBone Scavenger Service we discuss below, that introduce a service class below the (default) best-effort class, the approach of Carlberg *et al.* (2001) aims at *actively degrading* the QoS of certain flows and to deny them resources even if those resources could *not* be used by other flows. This service is thus intended at punishing certain flows and discouraging certain behaviour: It is suited for punishing non-TCP-friendly flows, to reduce the QoS of certain applications – for example, peer-to-peer applications – or to punish flows suspected to be part of a denial-of-service attack. In Carlberg *et al.* (2001), a modified CBQ scheduling algorithm and some penalty algorithms are used to degrade the quality of flows by increasing their dropping probability. Flows that exceed a certain packet count or service rate are punished. Their experiments show that while it is hard to penalise an individual TCP flow – especially a short flow – and still maintain a minimum throughput, the concept works well for UDP flows and aggregates of TCP flows.

The QBone Scavenger Service (QBSS) follows a different approach. It creates a service class with a lower priority than the best-effort class. Strict priority queueing is not

recommended; instead a small amount of network capacity is allocated to the QBSS class to avoid starvation of TCP flows in the QBSS service class during times when the best-effort service class is significantly loaded. Capacity not used by higher services is fully available to the QBSS service class – contrary to the approach of Carlberg *et al.* (2001). QBSS is thus designed not to punish certain behaviour but for bulk transfers that are currently run voluntarily during periods of low-utilisation (e.g. large nightly transfers of scientific data-sets, network backups, Content Delivery Network (CDN) content pushing). In addition, it is suited to downgrade the performance of non-critical traffic, such as peer-to-peer file sharing traffic at universities.

6.2.5.4 Alternative Best-effort

Alternative Best-effort (ABE), presented in Hurley (2001); Hurley *et al.* (2001), is an enhancement of IP best-effort. The idea is to have two service classes that provide a delay against throughput *trade-off*.

Each IP packet is marked green or blue. The green packets receive a lower delay but via a possibly higher loss probability lower throughput than the blue packets. Hurley *et al.* (2000) describe how the ABE colour can be encoded in the IP packet header.

One important property of the ABE service is that *green packets do not hurt blue packets*; if an application marks some or all of its packets green, the service – that is delay and throughput – received by applications that mark all their packets blue is not degraded. Therefore, if the ABE service model would be introduced in the Internet, unmarked packets would be considered to be blue packets and no harm would be done to applications unaware of the ABE service.

Applications aware of the ABE service would mark their packets blue or green or even mix both colours by marking some blue and others green. Packet sequence is preserved within the blue and within the green queues only, therefore when mixing the colours, packet reordering can be induced.

As green packets receive lower delay but higher loss, they are not strictly 'better' than blue packets and no incentive mechanism like pricing is needed to keep users from sending all their packets with the 'best' service. In addition, no policing mechanism is needed. All this has the additional advantage that the control and data path can be kept almost as simple as a traditional single-class best-effort network. The only additional complexity needed is that the scheduling mechanism has to make sure that green packets do not hurt blue ones. This requirement can be split into two parts:

1. The first part of the requirement is called *local transparency to blue*. It addresses the case of non-TCP-friendly sources.
   Local transparency to blue is defined over a plain best-effort scenario in which a node would treat all packets equally regardless of their colour. Local transparency to blue requires that every blue packet in an ABE node:
   - does not receive a larger delay than in the fictive plain best-effort scenario.
   - is not dropped unless it would also be dropped in the fictive plain best-effort scenario.

   A scheduling algorithm called *duplicate scheduling with deadlines (DSD)* is described in Hurley (2001); Hurley *et al.* (2001). DSD has elements from earliest-deadline-first

(EDF) and first-come-first-service (FCFS) schedulers. The packets are tagged with deadlines like in EDF, but this deadline is used only to determine which of the queues is to be served. Within each queue, FCFS is used.

Duplicates of all incoming packets are sent through a VQ which is served first-come-first-service (FCFS) with the rate of the plain best-effort scenario that is emulated by the VQ. The VQ is used to assign each blue packet a tag that indicates the time at which it would be served in the VQ. This acts as the deadline for the blue packet; a blue packet is always served at the latest moment the deadline permits, subject to work conservation. This ensures the local transparency to blue requirement; a blue packet that arrives when the VQ is full, is dropped.

Green packets are served in the meantime unless they have been in the queue for more than d sec, in the latter case they are dropped. For optimisation purposes a green packet also has to pass an acceptance test upon arrival to be put into the tail of the green queue, otherwise it is dropped. The acceptance test checks whether the queueing delay for the green packet exceeds d sec – imagine d to be in the order of magnitude of 20 ms. The scheduler events are summarised in Table 6.1.

It can happen that the deadlines of both the blue and the green packets at the head of their queues permit them to wait for the other packet to be served first. In that case, the packet to be served first is selected randomly, the probability that the green packet is selected is controlled by an additional parameter called the *green bias* g. The reason for this parameter becomes clear when looking at the second part of the 'green packets do not hurt blue packets' property.

2. If a TCP-friendly and greedy source is sending, its rate depends on the Round-trip Time (RTT) and the loss probability. If that source is sending green packets, it is possible that because the RTT decreases for green packets, that source would receive a higher throughput than when it would send blue packets only. This also carries the risk of hurting blue sources because of the increased rate. This leads to the second part of the requirement: *throughput transparency to blue* – a green flow shall receive a less or equal throughput than if it were blue.

   This requirement is much harder to implement than local transparency to blue because an exact implementation would have to keep track of the exact end-to-end RTTs for every flow. The authors of ABE propose to use a controller in each node that adapts

Table 6.1 Duplicate Scheduling with Deadlines Events

Event	What is served?
Both queues empty	Nothing
Green queue empty, blue queue not empty	Head of blue queue
Head of blue queue cannot wait	Head of blue queue
Blue queue empty, green queue not empty	Head of green queue
Head of blue queue can wait, head of green queue cannot	Head of green queue
Head of both queues can wait	Randomly

the above-mentioned green bias parameter g of the DSD scheduler so that the throughput transparency (estimated via a TCP throughput formula) is ensured. For this, the controller assumes that all flows are greedy and have a total RTT equal to the queueing delay at this node plus a fixed virtual base value (e.g. 20 ms). This tends to underestimate the green RTT but that is no problem as the underestimation is conservative for the blues.

To conclude, the ABE service is a good example for a low overhead tuned best-effort service that can offer some advantages like lower delay without introducing much overhead. One of its drawbacks is that it depends on a special scheduling algorithm and has received little IETF support so far. As blue packets are not harmed, green packets do not receive the same high QoS premium that packets would receive in other QoS systems like Intserv / Diffserv – but that is actually a desired property of ABE because it removes the need of policing and pricing.

6.2.6 Other Architectures

Besides the above discussed QoS architectures, there are a number of other interesting approaches to the question of how and what QoS to provide; we now discuss some of these works.

The above discussed SCORE approaches with dynamic packet state (Section 6.2.3) and Diffserv (Section 6.2.4) can be used to give absolute QoS guarantees without keeping per-flow state on the data or network path. However, they need per-flow state in the edge routers; this can be problematic, too, because also edge routers can be performance bottlenecks. For example, if many interconnection partners are connected via that router (see Part III of this book for interconnections). They typically have important additional tasks like running BGP, counting traffic volumes to transit partners, etc. Also, it is a common policy with providers when core routers are replaced with more modern routers because they can no longer handle the ever-growing traffic, that they are moved 'towards the edge' and used as edge routers. If it was considered problematic to keep per-flow states with them in the old core network, then the same is true in their new role as edge routers.

Approaches like PCBE (Section 6.2.5.2) and ABE (Section 6.2.5.4) do not need per-flow state at the edges but can only give soft relative QoS guarantees. This train of thought leads to the question whether it is possible to give strong absolute QoS guarantees to flows, without the need for per-flow state at the edge *and* core (**stateless edge and core**). Machiraju *et al*. (2002) propose such a solution that can be used to offer a service like the Diffserv premium service (see Section 6.2.4.2):

- The *data plane* is simple: For the reserved (premium) traffic, a single queue is maintained.
- For the *admission control,* a soft-state protocol is used to reserve a peak per-flow bandwidth in the intermediate routers. The reservation is refreshed in a fixed well-known interval $T_{refresh}$. Routers only keep track of the aggregate reserved rate by adding the reservation refresh messages in one interval $T_{refresh}$[10].

[10] The actual mechanism is slightly more complicated to take care of refresh messages that arrive delayed due to jitter, see Machiraju *et al*. (2002) for details.

- On the *control plane,* misbehaving flows could send refresh messages without being admitted. This is countered with router-specific lightweight certificates generated by each router along the path. They are attached to the admission request message if a flow is admitted by the router. Refresh messages are accepted only with a valid certificate. Misbehaving flows could stop sending and resume sending later, without undergoing the admission control test again using their old certificate. To avoid this, routers change the keys for their certificates at regular intervals.
- Another problem arises if flows send more than one refresh message per $T_{refresh}$. This would allow them to send more than what is actually admitted by the admission control. This can be avoided by random sampling. Random packets are chosen and the flows they belong to are monitored for some time to detect extra refreshes.
- Similarly, flows sending more than what is allowed could be detected by monitoring a (limited) number of randomly selected flows. Machiraju *et al.* (2002) present an alternative called *recursive monitoring* that turned out to be superior in simulations. The basic idea is to monitor aggregates of flows. If the aggregate misbehaves, it is recursively split into smaller aggregates until the misbehaving flow is detected.

Another interesting approach called *Paris Metro Pricing* (PMP) is introduced in Odlyzko (1999). PMP is a minimalist relative service differentiation scheme using only the price as differentiation mechanism. The idea is to split the bandwidth among several channels, for example, with (WRR) scheduling. The channels differ only in their price. The QoS of each channel depends only on the load of that channel. Users choose a channel depending on the expected QoS and their willingness-to-pay. PMP relies on self-regulation: If an expensive class is too congested, users can be expected to back off because for them it is not worth the price. The opposite can be expected if the expensive class is not congested while the cheaper class is. An economical analysis of PMP with a single monopolistic provider in Jain *et al.* (2001) indicates that there are profit incentives for monopolistic providers to adapt PMP compared to offering a single channel only.

6.2.7 Classification of Quality of Service Architectures

As a summary, we classify and describe the most relevant QoS architectures in the Tables 6.2 and 6.3 of this chapter with respect to the following points:

- **Shortest Timescale of Control**
  The timescale of a QoS architecture describes the smallest possible timescale that a QoS system based on that architecture can work and react upon to influence the QoS. It ranges
  - from the per-packet timescale which implies manipulation of individual packets for a per-packet QoS
  - over the RTT timescale (round-trip time) that implies a reactiveness of the system with a delay that is in the RTT order of magnitude
  - up to network engineering and capacity expansion timescales; they imply that a reaction is possible only by extending the network capacity.
- **Reactiveness**
  Reactive architectures react to QoS relevant events like congestion while proactive systems actively try to avoid these events before they occur.

Table 6.2 Classification of QoS Architectures Part 1

	Intserv	SCORE with DPS	Diffserv	ABE	PCBE	Over-provisioning
Shortest Timescale of Control	Packet	Packet	Packet	Packet	RTT	Network engineering
Reactiveness	Proactive	Proactive	Proactive	Proactive	Reactive	Reactive
Services	Guaranteed service Controlled load service Best-effort service Open for other services.	Guaranteed service Best-effort service Other services possible.	Extreme variety of possible services.	Two alternative types of best-effort services	Price-controlled best-effort service	Plain best-effort service
Guarantees	Absolute, end-to-end, per-flow, deterministic, loss and delay guarantees	As Intserv	Absolute, per Hop, per-aggregate, deterministic, loss and delay guarantees	Relative guarantees	No guarantees	No guarantees
Data Path Procedures	Edge & Core: Packet classification, scheduling & queue management, policing	Edge: Packet classification, scheduling & queue management, policing, marking Edge & Core: Scheduling & queue management, packet remarking, core-stateless scheduler	Edge: Packet classification, scheduling & queue management, policing, marking Edge & Core: Scheduling & queue management	Edge & Core: Scheduling & queue management Special scheduler necessary	Edge & Core: Queue management	

Table 6.3 Classification of QoS Architectures Part 2

	Intserv	SCORE with DPS	Diffserv	ABE	PCBE	Over-provisioning
Data Path Complexity	Per-flow	Edge: Per-flow Core: Independent of number of flows	Edge: Per-flow Core: Per-class complexity	Low	Minimal	Minimal
QoS Signalling	Required	Required, lightweight signalling protocol in Core	Not required	Not necessary	Not necessary	Not necessary
Admission Control	Required, per-flow and per-hop admission control mechanism tied to the signalling protocol	See Intserv	Not required, Per-SLA admission control by central or decentral bandwidth brokers	None	Endpoint admission control mechanism	None
Control Path Complexity	Per-flow state and processing in every node	See Intserv, reduced complexity in core due to lightweight signalling protocol	Strongly depending on signalling and admission control mechanism, low complexity possible	Minimal	Minimal	Minimal
Implications for the QoS Strategy	Differentiated pricing necessary	Differentiated pricing necessary	Differentiated pricing necessary	No differentiated pricing necessary	Per-packet pricing and accounting	No restrictions

- **Services**
  Different architectures allow different types of services to be offered.
- **Type of Guarantees**
  One of the key questions of a QoS system is what kind of guarantees it can give. If architectures allow different types of guarantees, we focus on the strongest possible guarantees.
  Service guarantees can be absolute or relative, deterministic or statistical. QoS architectures also differ in the granularity of the guarantees, that is, whether guarantees are given per-flow or per-aggregate.
- **Data Path Procedures**
  The QoS architectures differ in the QoS procedures like marking and policing they need on the data path. For some architectures, it is important to split this aspect into two: the procedures applied at edge routers and in the core network. Only the data path procedures needed *in addition* to those available in best-effort routers are listed.
- **Data Path Complexity**
  The data path complexity describes the parameters on which the complexity of the data path procedures mainly depends.
- **QoS Signalling**
  Some architectures depend on the use of a QoS signalling protocol like RSVP.
- **Admission Control**
  Admission control is required in some QoS architectures, see Section 6.6.
- **Control Path Complexity**
  The control path complexity describes on which parameters the complexity of the control path procedures mainly depends.
- **Implications for the QoS Strategy**
  Most QoS architectures have some important implications for the QoS strategy, mainly for pricing and tariff services. If different services are offered and one service is strictly better than another, pricing or similar mechanisms have to be used to keep all users from requesting only for the better service.

6.3 Data Forwarding Architecture

With the term data forwarding architecture, we describe the actual technology used for forwarding packets at a node. In the core of a network, packets can be routed or label switched.

If a packet is *routed*, the router evaluates information from the packet's IP header, mainly the 'time-to-live' and 'destination address' field, and its routing table to decide locally and upon arrival of the packet with which outgoing interface the packet is forwarded to its next hop.

If a packet is *label switched*, it receives a label at the edge of the network and the path that packets with a certain label take is set up beforehand through the core of the network. A label switching router forwards arriving packets solely on the information contained in the label; it does not have to look into the IP or higher-layer headers.

6.3.1 IP Routing

6.3.1.1 Routing Lookup

IP routing occurs at layer 3 of the 5-layer model (Figure 4.1). A router uses a routing protocol (Section 6.4.1) to maintain a routing table that contains the information – which next hop lies on the shortest[11] path to which destination. The routing lookup has to be made upon the arrival of a packet, and on the basis of the result the packet is put into the outgoing queue of the interface connected to the next hop router. It is obvious that the routing lookup is a time critical operation.

Because Classless Inter-domain Routing (CIDR, see Fuller *et al*. (1993); Rekhter and Li (1993)) routing is used in todays IPv4 based Internet, routing tables contain variable-length address prefixes. For a routing lookup, the prefix that matches best with the destination IP address of the packet has to be found. It is the longest match; therefore, the lookup problem is called 'Longest-prefix Matching'.

As an example, a part of a routing table is shown in Table 6.4. The IP prefix is stored in dotted-decimal notation, the number after the slash indicates the length of the prefix in bits. The best match for destination 130.83.198.178 in the routing table would be interface #3, because the first 18 bits match with the destination address. The best match for destination 130.83.64.130 would be interface #2 with 16 matching bits.

Longest-prefix matching algorithms try to optimise the average and worst-case number of memory accesses for routing lookup (and thereby the lookup time) and the memory requirement of the routing table. The routing table size in typical routers has grown exponentially in the last years from 30,000 to 120,000 entries; see Narayan *et al*. (2003). According to Bu *et al*. (2002), the reason for the rapid increase lies mainly in address fragmentation, that is, the fact that an autonomous systems (AS) has several prefixes that cannot be aggregated. Multihoming and load balancing also contribute to this trend and their contribution is growing faster than that of the address fragmentation. The prefixes of a multihomed AS cannot be aggregated by all of its providers and for load balancing, an AS can announce different prefixes via different AS paths. Complementary, Narayan *et al*. (2003) analyse the structure of the routing table and the impact of that structure on routing lookup methods.

The classical solutions for longest-prefix matching algorithms are *trie-based schemes*. A trie is a tree-like structure that exploits the fact that various entries share prefixes of each other and store the shared parts in the same location; see Fredkin (1960). The bits

Table 6.4 Section of a Routing Table (Example)

Destination Address IP Prefix	Next Hop	Output Interface
130/8	145.253.4	#1
130.83/16	145.253.81	#2
130.83.192/18	145.253.183	#3
130.83.192/24	145.253.12	#4
...	...	...

[11] with respect to the used routing distance metric.

of prefixes are used to direct the branching, see Figure 6.9. Nodes that correspond to a routing table entry are marked. Finding the longest prefix in a trie is straightforward; the bits of the destination address are inspected in sequential order, every time a marked node is passed, it is stored as the longest-prefix match found so far, until the end of the trie is reached. Obviously, this type of search can lead to a lot of memory accesses. The binary tree algorithm can be improved in a number of ways:

- *Path compression* as in Gwehenberger (1968) as a Patricia tree as in Morrison (1968), for example, removes internal nodes with only one child and thus the size of the data structure. A skip count has to be stored instead that indicates how many bits have been skipped. The IP lookup implementation for the Backbone Service Provider (BSD) Unix kernel by Sklower (1993) is based on a similar mechanism.
- *Level compression* as another example replaces n complete levels of a binary trie with a single node of degree 2^n, leaving the number of nodes unchanged but shortening the search path.
  Nilsson and Karlsson (1999) present a longest-prefix matching algorithm that combines path and level compression.
- Some router vendors do IP lookups based on compressed *multibit tries*/*tree bitmaps*, see for example, Degermark *et al*. (1997); Srinivasan and Varghese (1999b). These works are based on inspecting multiple bits simultaneously; therefore, a multibit trie is used – a multibit trie node has 2^k children. A comparison of Degermark *et al*. (1997) with Nilsson and Karlsson (1999) can be found in Kencl (1998). Eatherton *et al*. (2002) present a similar work optimised for implementation in hardware.

Many other routing lookup schemes exist. An overview, taxonomy and complexity evaluation is given in Ruiz-Sanchez *et al*. (2001). Gupta (2000); Srinivasan and Varghese (1999a); Waldvogel (2000) also give an overview.

As a more novel approach, Dharmapurikar *et al*. (2003) propose the use of bloom filters for longest-prefix matching.

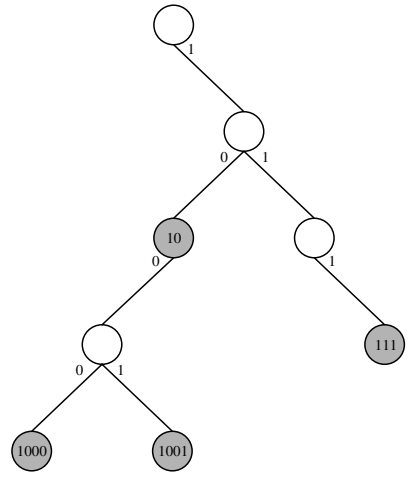

Figure 6.9 Trie Structure

Special hardware solutions like *Ternary Content Addressable Memories* (TCAMs, see McAuley and Francis (1993); Shah and Gupta (2001)) use parallelism to gain lookup speed. They can store the values 0, 1 and X; X is a 'don't care' value. TCAMs can compare a given destination address to all stored prefixes in parallel and return the longest match in a single memory access.

6.3.1.2 Other Routing Tasks

Besides the forwarding decision, a router has to perform some other tasks, see for example, Baker (1995):

- Decrementing the Time-to-Live (TTL) field.
  A router decreases the TTL field of the IP header. If it reaches zero, it is assumed that the packets loops in the network; the packet is discarded and an Internet Control Message Protocol (ICMP) error message is generated.
- Verification and update of the IPv4 header checksum.
  The checksum verification is often omitted for performance reasons because packets hardly ever are corrupted in transit and end systems will recognise the rare cases of corruption anyway. Therefore, IPv6 no longer has an IP header checksum (see Deering and Hinden (1998)).
  If the TTL field of an IPv4 packet is decreased, the checksum has to be updated. An efficient mechanism is described in Mallory and Kullberg (1990); Rijsinghani (1994).
- Fragmentation.
  IPv4 packets that are too large for a subnet are fragmented. However, IP fragmentation rarely occurs on high-speed links because these are designed to handle large enough packets.

6.3.2 Label Switching

As we have shown in the last section, the routing lookup is a time critical operation. It can be replaced with a simple index label lookup if a label switching mechanism like MPLS is used. Multi-protocol Label Switching (MPLS) is the IETF's standardised label switching packet forwarding architecture and has largely replaced prestandard and proprietary solutions like Ipsilon's IP Switching, IBM's Aggregate Route-based IP Switching (ARIS) (Aggregate Route-based IP Switching), Cisco's early Tag Switching and Toshiba's Cell Switch Router technology; see Armitage (2000). With and without explicit traffic engineering, it is growing in popularity for provisioning and managing core networks. Practically every modern router is able to do plain IP packet forwarding and MPLS. The early evolution of MPLS is summarised in Viswanathan *et al*. (1998).

In traditional IP routing, each router analyses the header of the arriving packet and independently chooses the next hop based on the distributed routing algorithm that uses a routing protocol (see Section 6.4.1) and the information of the IP header. Using the MPLS terminology of Rosen *et al*. (2001), an IP routing lookup partitions the IP packets destined to addresses with the same IP address prefix in the routing tables into one forwarding equivalence class (FEC). Each FEC is mapped to the next hop in a routing table; therefore, different packets in the same FEC are treated equally with respect to the

forwarding decision. As the IP packet traverses the network, each hop re-examines the packet and assigns it to a FEC.

Contrary to that, in a network using MPLS as data forwarding architecture, the FEC assignment is done at the MPLS ingress node just once when the packet enters the MPLS domain. An MPLS domain is a contiguous set of nodes using MPLS as the forwarding architecture. The FEC is encoded into a 4 byte label (so-called shim header) that is attached to the packet between the layer-2 and layer-3 headers. Subsequent hops no longer have to examine the IP header of the packet, the label is used as an index into a forwarding table that specifies the next hop as outgoing interface and a new label that replaces the old one (label swapping). Each MPLS node (router/switch) is called a *Label Switching Router* (LSR). The path for one FEC through one or more LSRs is called Label Switched Path (LSP).

MPLS offers some advantages over the conventional IP forwarding architecture:

- MPLS forwarding could be done by switches that do not have to be capable of analysing the IP headers. MPLS forwarding is a simpler operation than IP routing and less expensive to implement for operations at state-of-the-art line speeds.
- The ingress router assigning the label can use any information to assign a label to a packet. Apart from analysing the destination address of the IP header, the transport layer ports could be evaluated, or the DSCP of a packet in a Diffserv domain.
- Additionally, the process of determining the label can become more and more sophisticated without any impact at all on the core routers.
- As information about the ingress router does not travel with an IP packet, traditional IP routing does not allow differentiation between packets from two different ingress routers in the core. With MPLS, this can easily be done if each ingress routers assigns a different label.
- For traffic engineering or policy reasons, it may be desirable to force packets to follow a path different to the standard shortest path as it is determined by the routing protocol algorithm. With MPLS traffic engineering, the path set-up can be controlled centrally and any path through the network can be used.

These advantages make it obvious that a network with MPLS-based forwarding architecture is well-suited for traffic engineering. We discuss and evaluate traffic engineering in Part IV of this book; works related to MPLS in the context of traffic engineering are discussed in Chapter 12.

For one LSP, the direction of the traffic flow is called *downstream*. The assignment of a particular label to an FEC is done by the downstream LSR and has to be signalled opposite to the traffic flow direction of the upstream LSR. The protocols used for signalling the label bindings and setting up an LSP are called *Label Distribution Protocols* (LDPs)[12]. The MPLS architecture of Rosen *et al*. (2001) does not assume one specific protocol; moreover, it does not even assume that there is only a single protocol used. LDPs are part of the signalling architecture of a network and are thus discussed in that context (Section 6.4.3).

[12] Unfortunately, one of the IETF LDPs is called exactly like the general term: *LDP*. It is discussed in the following text.

6.4 Signalling Architecture

This signalling architecture includes the different signalling/control protocols used to manage the network. We distinguish between routing, QoS signalling, and LDPs.

6.4.1 Routing Protocols

Routing protocols can be distinguished as interior and exterior routing protocols.

6.4.1.1 Interior Routing Protocols

Interior routing protocols are used to exchange routing information inside an INSP's network, based on that information the routers are enabled to fill their routing table by calculating the shortest path through an IP network with respect to a certain composite distance metric to a destination. The distance metric can be based on hop count, delay, link bandwidth, utilisation and so on.

Existing routing protocols can be classified as distance-vector or link-state protocols. **Distance-vector** protocols are based on the distributed Bellman-Ford algorithm and exchange their distances to all destinations with their neighbours; each node's calculation of the shortest path depends on the calculation of the other nodes. With *link-state* protocols, a node distributes its connectivity with its direct neighbours to all routers in the network, which can then reconstruct the complete topology and calculate their routing table by constructing the shortest-path tree. Generally, link-state protocols are more stable and converge faster.

The Routing Information Protocol (RIP) (see Hedrick (1988)) was the most widely deployed interior routing protocol for the early Internet. It is a distance-vector protocol. In the mid-1980s Cisco introduced with IGRP a distance-vector protocol that also supports multipath routing and avoids some performance problems of RIP in large heterogeneous networks; see Rutgers (1991). IGRP was widely replaced by its enhanced version EIGRP in the early 1990s; see Cisco (2003). EIGRP is still a distance-vector protocol but uses some features of link-state protocols to overcome some of the disadvantages of distance-vector protocols.

The OSPF (Open Shortest Path First, see Moy (1998)) routing protocol was the IETF's approach to overcome the limitations of RIP. OSPF is a link-state routing protocol using the shortest path first as Dijkstra algorithm (see Dijkstra (1959)) to derive the shortest path to each node. Like IGRP, it supports multipath routing. For scalability reasons, OSPF is a hierarchical protocol and allows a larger network to be split into subnetworks; all nodes of a subnetwork have identical topological databases but limited knowledge of the topology of the other subnetworks.

Another link-state protocol that can be used with TCP/IP networks is OSI's IS-IS routing protocol, see Callon (1990); ISO DP 10589 (1990).

6.4.1.2 Exterior Routing Protocols

For the route advertisement (see also Section 9.2) between two AS as INSP networks, exterior routing protocols like BGP (BGP, see Rekhter and Li (1995)) are used. Contrary

to the interior routing protocols, exterior routing protocols are used between two INSPs to exchange reachability information, enforce policy decisions and hide the details of the internal topology from the interconnection partners. Contrary to the interior routing protocols, routes advertised by BGP consist of AS hops and not individual router hops.

BGP neighbours exchange full routing information after they establish their BGP connection that uses TCP as transport protocol. When changes to the routing table are detected, the BGP routers exchange only those routes that have changed; they do not send periodic routing updates and advertise only the optimum and not all possible paths to a destination network.

In order to support policy decisions, BGP associates certain properties with the learned routes. They are used to determine the best route when multiple routes exist to a particular destination. These properties are referred to as *BGP attributes*. For example, the *local preference* attribute is used to select the exit point for a specific route if there are multiple exit points from the AS. Related to that, the *multi-exit discriminator attribute* is used as a suggestion to an external AS regarding the preferred route into the AS that is advertising the attribute. The *origin attribute* indicates, for example, whether BGP learned about a particular route was via an exterior routing protocol or whether it was injected into BGP based on information from an interior routing protocol (then the route is local to the originating AS). To simplify administration, the *community attribute* allows group destinations – called *communities* – to which routing decisions (such as acceptance, preference and redistribution) are applied. Finally, the *next hop attribute* is the IP address that is used to reach the advertising router.

6.4.2 Quality of Service Signalling Protocols

Some QoS architectures depend on the use of a QoS signalling protocol. QoS signalling protocols can be receiver or sender oriented, based on which side initiated the process of requesting QoS from the network. The most famous signalling protocol for the Internet is the resource reservation protocol RSVP that is proposed for use with Intserv but can also be used for the label distribution in networks with a MPLS forwarding architecture; see Braden *et al*. (1997) and Awduche *et al*. (2001). The latter functionality is discussed in Section 6.4.3.1.

6.4.2.1 RSVP

The operation of RSVP in conjunction with the Intserv architecture was described in Section 6.2.2.2. The functional specification of RSVP is given in RFC 2205 (see Braden *et al*. (1997)), extensions to the QoS signalling functionality are given in RFC 2961 (Refresh Reduction, see Berger *et al*. (2001)) and RFC 3175 (RSVP Aggregation, see Baker *et al*. (2001)). For a scalability discussion of RSVP see Section 6.2.2.6.

The key functionality of RSVP can be summarised as follows:

- RSVP uses IP datagrams and alternatively UDP encapsulation for the message exchange.
- It supports heterogeneous receivers in large multicast groups by using a *receiver-oriented* reservation style based on the argument that the receivers know best about their QoS requirements.

- Dynamic membership in these large groups is supported with the receiver-oriented approach, too, and by the fact that the data transfer is handled separately from the control by RSVP. Receivers can join and leave the distribution tree installed by RSVP at any time during the data transmission.
- Multiple receivers are supported by the concept of multicast groups. At the same time, RSVP supports multiple senders sharing resources too. For this, the *reservation styles* are used (see Section 6.2.2.2).
- RSVP is independent of and does not interfere with the multicast group management protocol, the data path procedures or the routing protocols of the network.
- For the case of routing changes or network failures, a recovery mechanism is necessary to establish new and release old reservations. Because of the *soft-state* principle of RSVP reservations are frequently refreshed. In the case of a routing change or network failure, new reservations are set up when the refreshing takes place and old reservations are released automatically after some time.

6.4.2.2 Other Protocols

While RSVP is the dominant QoS signalling protocol, there are a number of other QoS signalling protocols for IP networks:

- The Internet Stream Protocol Version 2 (ST-2+) was an experimental IETF QoS signalling protocol. It is specified in RFC 1819 (see Delgrossi and Berger (1995)) and differs from RSVP in many aspects. It is a connection-oriented hard-state protocol. ST-2+ is operating parallel to IP and not compatible with the datagram service of IP. For a comparison of ST-2/ST-2+ and RSVP we refer to Delgrossi *et al*. (1993); Mitzel *et al*. (1994).
- YESSIR (YEt another Sender Session Internet Reservations), see Pang and Schulzrinne (1999, 2000)) is a QoS protocol based on RTP that was developed to avoid complexity and scalability issues that RSVP was believed to have. RTP itself is specified in Schulzrinne *et al*. (1996). YESSIR avoids message processing overhead in the end systems and routers and reduces the bandwidth consumption of the refresh messages. Reservations are triggered by the sender; the protocol is a soft-state protocol.
- The Boomerang protocol of Fehér *et al*. (2002, 1999) aims at reducing part of the RSVP overhead by using a sender-oriented approach. The sender generates a reservation message. Once this reaches the receiver, the reservation is already in place.

6.4.3 Label Distribution Protocols

An MPLS data forwarding architecture implies the use of a label distribution protocol to set up LSPs unless each switch would be configured statically by hand. Within the IETF, two label distribution protocols that also allow the set-up of explicit paths for traffic engineering are under discussion: RSVP-TE and Constraint-based Routing Support For LDP (CR-LDP).

6.4.3.1 RSVP-TE

RSVP-TE stands for RSVP with traffic engineering support. RSVP-TE (described in Braden *et al*. (1997)) is a set of extensions to the basic RSVP protocol (see Section 6.4.2). It is specified in RFC 3209, see Awduche *et al*. (2001).

RSVP messages are exchanged directly via raw IP datagrams. The protocol uses soft-state and refresh reduction allowing it to recover automatically from failure. RFC 3209 (see Awduche *et al*. (2001)) also describes a means of rapid node failure detection via a new HELLO message.

The label distribution method is downstream-on-demand: If an ingress LSR determines that a new LSP has to be set up to a certain egress LSR, a PATH message is sent containing a specified explicit route. That route can be different from the standard hop-by-hop route. The message also contains the traffic parameters for the new route. Each router along the path receiving the message builds up state. The egress router selects a label and answers with a RESV message that is routed back towards the ingress, finishing the set-up of the new LSP. Intermediate routers allocate resources and select a label when the RESV message reaches them. They update the message and forward it to the ingress routers via the interface by which the according PATH message was received.

6.4.3.2 CR-LDP

CR-LDP is a set of extensions to the LDP protocol of Andersson *et al*. (2001) that are specified in RFC 3212 (see Jamoussi *et al*. (2002)). CR-LDP stands for constraint-based routing support for LDP.

CR-LDP uses TCP connections for a reliable message exchange and is a hard-state protocol. It does not need to refresh the set-up of an LSP. With respect to failure recovery, it is not as well placed as RSVP-TE. The loss of the according TCP control connections also results in a failure of all associated LSPs.

The label distribution method is downstream-on-demand as in RSVP-TE. A label request message is sent by the ingress LSR towards the egress. It contains an explicit route. Contrary to RSVP-TE, intermediate routers reserve resources immediately when the label request reaches them. The egress-router responds to the label request with a label mapping message that contains the label and information about the final resource reservation. It is routed back to the ingress nodes.

While RSVP-TE and CR-LDP are quite different as pure protocols, they offer similar functions to the user. For a more detailed comparison of both protocols, we refer to Brittain and Farrel (2000).

6.5 Security Architecture

The *IPsec* security architecture is the security architecture of the IETF[13] for the Internet Protocol (IP); it is specified in Kent and Seo (1998). It offers cryptography-based security services at the IP layer and enables applications like virtual private networks (VPN).

[13] IETF IPSEC working group, http://www.ietf.org/html.charters/ipsec-charter.html

The architecture consists of a set of protocols, mainly the *Authentication Header* (AH, see Kent (1998a)) and *Encapsulating Security Payload* (ESP, see Kent (1998b)) protocols and the *Internet Key Exchange* protocol (IKE, see Kaufman (2004)):

Authentication Header (AH) and ESP can operate in two modes. The transport mode provides protection primarily for upper-layer protocols and the tunnel mode to tunneled IP packets. *AH* provides data integrity and data origin authentication and optional replay protection by embedding an additional header (the AH) that contains an authentication field into the IP datagram. The authentication field contains an integrity check value that is calculated over those IP header fields that do not change in transit, the AH header except for the authentication field and the entire upper-layer protocol data, and protects them against tampering. Replay protection is provided by an additional sequence-numbering mechanism.

The *ESP* protocol additionally offers confidentiality by encapsulating and encrypting the data to be protected. In transport mode, it encrypts and optionally authenticates the IP payload but not the IP header. In transport mode, AH authenticates the IP payload plus selected parts of the IP header. In tunnel mode, the entire IP packet is encapsulated within a new IP packet to ensure that no part of the original packet is changed.

The IPsec encryption mechanisms need a key exchange mechanism. This mechanism can be manual or automated. For automated key exchange, the IKE protocol is proposed in Kaufman (2004).

While security aspects are important for Internet service providers, they are not in the scope of this book and therefore not further discussed here. For more information about security architectures, we refer to Schneier (1995) for a general overview and to Frankel (2001) for more details.

6.6 Admission Control

Admission control is an optional aspect of the control path of the network. Admission control is used to keep the network load within certain bounds on a small timescale. There is a vast amount of general work on admission control and the different proposed admission control systems and schemes vary enormously. Most of them are independent of a specific network architecture. We give an overview and classification of admission control next.

An actual admission control system is characterised by a number of properties, the most important ones are shown in Figure 6.10. It is important to stress that the individual properties influence each other significantly. For example, the type of guarantees a system can support strongly depends on the flow and network behaviour assumptions and the location of the system.

The individual objects for which admission control decisions are made are called *flow* throughout this chapter; this shall neither imply that they are necessarily micro-flows such as individual TCP flows nor that they are unidirectional. They could also be macro-flows consisting of an aggregate of micro-flows, such as the complete traffic of one customer or a group of customers.

We now present a structured overview of different admission control systems.

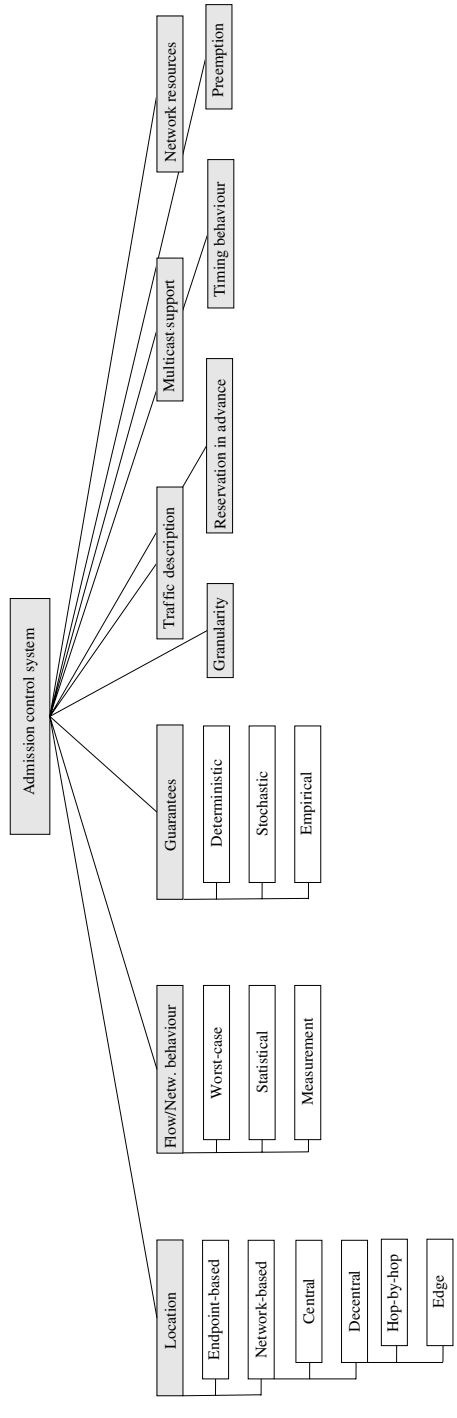

Figure 6.10 Admission Control Systems

6.6.1 Location

One important property of each admission control system is the location where the admission control decisions are made.

6.6.1.1 Endpoint Admission Control

In endpoint admission control schemes, the end-to-end admission control decision is made at the end systems themselves. No admission control instance and, therefore, less 'intelligence' is required in the network itself. As the endpoints have no control and no further information about the traffic of other endpoints, the decision is typically based on probing and measurement information like packet marking.

Pioneering work in the direction of endpoint-based distributed admission control has been done by Gibbens and Kelly (1999); Kelly (2000); Kelly *et al*. (1998). Their analysis shows the basic stability of distributed admission control based on marking at resources even in the case of feedback delays. Building on these results, some works shed light on the influence of delayed system reaction on stability, which presents bounds for the reaction delay, see Johari and Tan (2001); Massoulie (2000). Kelly *et al*. (2000) present a model for an Internet exclusively managed by the end systems and analyses the stability of this system.

As endpoint admission control systems assume no admission control instances in the network that could actually hinder non-admitted flows from sending, a mechanism is needed that forces or gives incentives to the end systems to perform the admission control algorithm and to behave according to its decision. In the works of Kelly (2000); Kelly *et al*. (2000), pricing per ECN mark is used as an incentive mechanism.

A simulative comparison of the basic design options for endpoint admission control is presented in Breslau *et al*. (2000b).

6.6.1.2 Network-based Admission Control

Network-based admission control schemes decide to admit or reject flows to one network. As a flow can pass through several networks, several sequential admission control decisions might be necessary. Network-based admission control schemes can be further distinguished as centralised and decentralised systems:

- In *centralised* systems, the decision is made at a central instance of the network that can have global knowledge of the network's current state. BBs (see Section 6.2.4.3) typically include a centralised admission control system. The bandwidth broker concept goes back to Nichols *et al*. (1999); Schelen (1998). Examples are given for example, in Khalil (2003); Khalil and Braun (2000); Terzis *et al*. (1999b); Zhang *et al*. (2001, 2000). The centralised BB used in the experiments of Chapter 8 is also of this type.
- *Decentralised* systems can be further divided into whether each link/hop or only the network edges are involved into the decision:
  - A typical example for a *hop-by-hop* admission control decision is the Intserv/RSVP admission control mechanism for GS and CL service as described in RFC 2212 (see Shenker *et al*. (1997)) as RFC 2211 (see Wroclawski (1997)). Each router along the

path of the flow through the network checks its resource availability before a new flow is actually admitted to the network, see Section 6.2.2 for more details.

- For *edge-based* admission control, the admission control decision is based on information locally available at the edge node of the network.
  The admission control decision can be made
  - exclusively at the ingress node (*ingress-based*) (see the decentral BB of Chapter 8),
  - exclusively at the egress node (*egress-based*) (e.g. Cetinkaya and Knightly (2000)) or
  - at both nodes (*ingress-egress-based*) (e.g. Bhatnagar and Nath (2003); Bhatnagar and Vickers (2001)).

  Edge-based admission control mechanisms can also be distinguished by the nature of the local information they are using.
  - The information can be *local traffic measurements* that can be constantly updated. Cetinkaya and Knightly (2000) present, for example, an egress-based admission control architecture. It treats the core network as a black box and is based on monitoring the aggregate traffic characteristics of one service class per path at the egress nodes. One-way per-packet delay measurements are used; these are, however, all but trivial to make. On the basis of these measurements, statistical traffic envelopes are derived and used as decision basis for admitting new flows.
    In Karsten and Schmitt (2002), ECN marks are counted and constantly updated at the egress at a per-ingress basis and used as estimation for the congestion level of the network.
  - The information can also be the *status information of the whole network* that is stored in a distributed database at the edge nodes. This allows each edge to base its decisions on the same type of information that is available to a centralised admission control system. However, contrary to the centralised system, for the decentralised one a synchronisation and update mechanism is needed for the distributed database. The distributed database has to be updated on a relatively small timescale or the system will not work efficiently.
    Systems implementing a decentral admission control algorithm based on a distributed database are described in Bhatnagar and Nath (2003); Bhatnagar and Vickers (2001). They use token passing.
    Bhatnagar and Vickers (2001) specify a mechanism to provide bandwidth guarantees that requires only edge routers to implement the admission control scheme. No assumptions about the behaviour of the core routers on the data or control path are made, especially core routers do not have to be able to differentiate between different flows and not even between best-effort and reserved flows. The approach further assumes that the edge nodes have up to date information about the topology of the network and that there is a route-pinning mechanism for the network; this can, for example, be MPLS or IP source routing. RSVP is used as signalling mechanism, but only interacts with the ingress- and egress-router of a network. On the basis of the RSVP message exchange, the route between ingress and egress nodes is pinned.
    The admission control mechanism uses a distributed database. Each ingress router has knowledge of the network's topology and a more or less up-to-date knowledge about the reservation state of the network. For synchronisation, a token passing

mechanism is used. A router is only allowed to change the reservation state of the network if it possesses a token that is passed around the edge routers. A reservation for a new flow does not become effective until the ingress router has fully circulated the updated token once among all edge routers. This prevents several edge routers from over-allocating bandwidth by simultaneously reserving bandwidth on a single link and it gives edge routers the opportunity to reduce the rates of best-effort flows sharing the same links as the new reserved flow. The latter is necessary because core routers cannot differentiate between reserved and best-effort flows.

Bhatnagar and Nath (2003) adapt the mechanism of Bhatnagar and Vickers (2001) to support GS in core-stateless networks (see Section 6.2.3) and improve the efficiency in several ways, for example, by marking potentially congested links and only requires a full token circulation before admitting a new flow when marked links are involved.

The drawback of these approaches is that compared to a central mechanism, they introduce additional delay (token circulation time) that can become long for large networks before admitting a new flow. In addition, they require each edge router to have the computational resources for managing and updating the database and add additional complexity to protect against lost tokens etc. Therefore, in most cases a specialised centralised system would seem the better choice.

- Finally, the local information used as a basis for an edge-based admission control decision can be a *contingent* as *resource budget* that is assigned off-line to the edge node. While measurement information as the distributed database is updated in rather small intervals, the contingents are updated only on much larger timescales and typically by a central instance based on the past performance of the system.

We call the latter mechanism *contingent-based admission control*. Among other things it is investigated in Chapter 8. Contingent-based admission control systems can be further distinguished by the contingent assignment.

- The contingent is assigned to the edge *node;* all flows entering the network through this node share this contingent.
- The contingent can also be assigned to each ingress and respective egress *link* of the edge node. Only flows with the same first and respective last hop through the network share a contingent.

  The problem with the first two contingent assignments is that they are very inefficient for deterministic guarantees as all – also pathological – traffic patterns through the network have to be taken into account when assigning the contingents. This results in very low contingents for deterministic services. A famous example is the Charny bound, see Charny and Le Boudec (2000) and Figure 6.7.
- The contingents can also be assigned to *tunnels* or MPLS label switched paths through the network if information about the traffic patterns is available. More state has to be kept in this case and it might be necessary to update the contingents more regularly than for the first two cases for this mechanism to be efficient. However, it promises higher possible contingents for deterministic guarantees as the information about the traffic patterns can be exploited in the contingent assignment process.
- A further alternative of contingent-based admission control schemes is assigning each ingress as egress node contingents for all links of the complete network.

This approach decentrally controls flows from the edge but can take the whole path through the network into account. The admission control test for a new flow predicts the path of that flow through the network – not a trivial task. Along the links of that path, the node checks for every link whether there are still contingents that were assigned to it available for that link. This approach necessarily loses efficiency compared to a centralised admission control, as an edge node cannot use the contingents assigned to another edge node for the same link. For this approach to be efficient, the contingent assignment process is of great importance. This approach is discussed in more detail in Menth (2004).

A comparative study of different contingent-based admission control schemes (and other admission control schemes) and a discussion of contingent assignment algorithms are presented in Menth (2004); Menth *et al*. (2003).

6.6.2 Flow and Network Behaviour

For the admission control test, certain assumptions about the flow and network behaviour have to be made.

6.6.2.1 Worst-case Assumptions

If worst-case behaviour of the flow and network elements is assumed, a conservative admission control test is performed. For decisions based on worst-case assumptions, the traditional *network calculus* offers a suitable mathematical framework, see Section 3.2.

6.6.2.2 Statistically Relaxed Assumptions

The admission control test can also be performed with statistically relaxed assumptions. They promise a higher resource usage at the cost of an increased but controlled risk of wrong decisions that will manifest themselves in violations of the (loss and delay) guarantees.

Among other methods, the stochastic network calculus and the queueing theory offer mathematical foundations for statistically relaxed flow and network behaviour assumptions; see Section 3.1 and 3.2.

6.6.2.3 Measurements

Admission control systems that use the two above-mentioned assumptions (worst-case and statistically relaxed) are based on mathematical models for predicting flow and network behaviour and maintain state information about the currently active flows. Contrary to that, predictions about the flow and network behaviour can also be based on measurements. In this case, we speak of *measurement-based admission control*. There are vast amounts of works on that topic; the works can be divided into admission control schemes with *active* and *passive* measurements. Active measurements actively probe the network by sending special probe packets while the passive measurements passively monitor the performance of normal data packets.

In addition, measurement-based admission control schemes can be classified by whether they measure the properties of individual flows (as e.g. in Karlsson (1998); Más and Karlsson (2001); Más *et al*. (2003)) or that of traffic aggregates (as e.g. in Jamin *et al*. (1997a); Qiu and Knightly (2001)).

The probe-based admission control scheme developed in Karlsson (1998); Más and Karlsson (2001); Más *et al*. (2003) for a loss-predictive unicast service and in Más *et al*. (2002) for multicast service is a typical example for an active measurement-based admission control scheme. A controlled load (see Section 6.2.2.5) type of service is offered. Before a new flow is accepted, a loss measurement is done by actively sending constant bit-rate probe packets at the maximum rate of the new flow for a sufficient time from the network ingress to the egress node. At the egress, the loss can be measured and reported back. The core network differentiates data packets from already admitted active flows and the probe packets so that probes do not disturb the active flows. A 0.5 to 2 s probing interval is recommended. In Más and Karlsson (2001) a simulative study of this scheme is contained; Más *et al*. (2003) use queueing theory to analytically evaluate the scheme for a single link and a single probing process. The comparison study of several endpoint and measurement-based admission control schemes also reports probing durations in the order of several seconds in Breslau *et al*. (2000b), whereas recent simulative work in Kelly (2001b) argues for much lower values for the initial probing phase.

Instead of using the measured packet loss as basis for the decision, measured delay such as delay variations are used in Bianchi *et al*. (2000, 2002). Kelly *et al*. (2000), Kelly (2001b) propose using ECN marks for a distributed measurement-based admission control system. Karsten and Schmitt (2002) also use ECN marks as congestion indication. So-called load control gateways running at a backbone network edge use the amount of measured ECN marks that data packets experience to estimate the current congestion level of the network.

The works of Gibbens and Kelly (1997); Gibbens *et al*. (1995) explicitly maximise the expected profit of an admission control that is defined by the reward of utilisation minus the penalty of packet-losses by calculating acceptance bounds for a specific set of flow types.

Jamin *et al*. (1997a) present an algorithm that uses the measured queueing delay of individual packets and the utilisation of the different service classes as inputs to derive an aggregate token bucket descriptor for each class. This measured token bucket is typically much smaller than the sum of the individual worst-case token buckets that describe the individual flows. Before admitting a new flow, the available bandwidth and the delay bounds are checked on the basis of these measured aggregate token bucket descriptors.

In Benameur *et al*. (2002), a measurement-based admission control mechanism is evaluated that explicitly considers elastic (TCP) flows besides real-time multimedia flows. In most other admission control schemes, the special characteristics of elastic flows are ignored or they are assumed to be in a low-priority best-effort upon which no admission control is applied.

The rate a new elastic flow would acquire is estimated either with a TCP phantom connection (an emulated TCP connection over the considered path) or by measuring the loss rate and applying the TCP formula. A new flow (elastic or not) is accepted only if it does not reduce the throughput of ongoing elastic flows below a certain threshold.

6.6.3 Guarantees

Three types of guarantees can be given to newly admitted flows such that their transmission or QoS requirements will be fulfilled by the network. The guarantees very strongly depend on the flow and network behaviour assumptions.

6.6.3.1 Deterministic Guarantees

Deterministic guarantees are based on worst-case assumptions. The Intserv GS of RFC 2212 (see Shenker *et al*. (1997)) is the typical example for a service with deterministic loss and delay-bound guarantees, see Section 6.2.2.4. Other works with deterministic service guarantees are, for example, Choi *et al*. (2000); Elwalid *et al*. (1995b); Knightly *et al*. (1995); Rajagopal *et al*. (1998). For example, Choi *et al*. (2000) offer delay guarantees by an offline worst-case delay calculation for QoS systems like Diffserv with aggregate scheduling.

6.6.3.2 Statistical Guarantees

Statistical guarantees are based on the statistically relaxed assumptions of flow and networking behaviour. There is a broad set of admission control algorithms for stochastic service guarantees; most of them fit into one of the following five classes according to Knightly and Shroff (1999):

- Average and Peak Rate Combinatorics
  Lee *et al*. (1996) use the peak rate and long-term average rate to predict the loss probability assuming a bufferless multiplexer. The loss rate is used as basis for the admission control decision. Ferrari and Verma (1990) use the delay-bound violation probability as basis for the decision and use the peak rates and worst-case average rates of the flows as inputs.
- Additive Effective Bandwidths
  The effective bandwidth is the bandwidth bw_f that has to be provided for a flow f to fulfill its service guarantees. It is a function of the flow's required loss probability and stochastic properties like the peak- and average rate or the mean burst duration. Overviews of the effective bandwidth concept can be found in Bodamer and Charzinski (2000); Gibbens and Teh (1999); Kelly (1996). There are different ways of computing the effective bandwidth, see for example, Courcoubetis and Weber (1995); Elwalid and Mitra (1993); Guérin *et al*. (1991); Kesidis *et al*. (1993). A simple admission control decision based on effective bandwidths makes sure that the added effective bandwidths bw_f do not exceed the link's capacity C: $\sum_f bw_f \leq C$.
- Refined Effective Bandwidths
  The above additive effective bandwidth approach has two shortcomings. First, the result is not applicable to traffic sources that show long-range dependency. Second, the economies of scale such as the multiplexing gain from adding a large number of sources are not exploited by adding the effective bandwidths, resulting in an inefficient admission control mechanism (see Knightly and Shroff (1999)). More advanced effective bandwidth approaches are not additive and incorporate the interdependences of the

traffic flows on each other when calculating the effective bandwidth, see for example, Courcoubetis *et al.* (1998); Duffield and O'Connell (1995); Kelly (1996).

- Loss Curve Engineering
  A loss curve models the loss probability as a function of the buffer size. It can be used as the basis for an admission control scheme. Assuming additive effective bandwidths (see preceding text), the loss curve is an exponential function of the buffer size. For the reasons mentioned above, the additive effective bandwidths and therefore the exponential loss curves are inefficient. Various techniques have been proposed, which seek to engineer the shape of the loss curve to better reflect empirical relationships, see for example, Baiocchi *et al.* (1991); Choudhury *et al.* (1996); Elwalid *et al.* (1995a); Shroff and Schwartz (1998).
- Maximum Variance Approaches
  Maximum variance approaches are based on estimating the loss probability via the tail probability of an infinite queue based on a Gaussian aggregate arrival process. The Gaussian characterisation of the traffic allows for different correlation structures as any function can be a valid autocovariance function, hence it can capture the temporal correlation of the traffic. Some maximum variation-based admission control schemes are Choe and Shroff (1998); Kim and Shroff (2001); Knightly (1997).
  Knightly and Shroff (1999) evaluate typical admission control schemes from these five categories in a set of experiments using Motion Pictures Expert Group (MPEG) video traces and Markov modulated on-off traffic sources. Among other things, they show that the assumption of bufferless network elements significantly reduces the admission control efficiency and network utilisation and that the accuracy of an admission control algorithm for one type of traffic does not assure accuracy for another type of traffic.

6.6.3.3 Empirical Guarantees

If a measurement-based admission control scheme is used, only *empirical guarantees* based on the past networking behaviour can be given. In Jamin *et al.* (1997b) and Breslau *et al.* (2000a), an extensive comparison of measurement-based admission control schemes finally results in the conclusion that all schemes perform fairly similar with respect to the utilisation they yield.

6.6.4 Other Properties

Most admission control systems need an explicit *traffic description* for new flows. At least for deterministic guarantees, usually a policer or shaper is used to force flows to comply with their traffic description. A wide variety of traffic descriptors can be imagined, for example,:

- Peak rate
- Average rate and maximum burst size, for example, a token bucket or if extended by peak rate and maximum packet size a TSpec, see Shenker *et al.* (1997)
- Effective bandwidth
- General arrival curve for network calculus
- Elastic flows could be characterised by their transfer volume alone, see for example, Benameur *et al.* (2002)

- Peak rate and long-term average rate as for example, in Lee *et al.* (1996)
- Peak rate and short-term average rate as for example, in Ferrari and Verma (1990).

Another characteristic is whether *multicast* flows are supported as for example, in Más *et al.* (2002); Shenker *et al.* (1997).

The above-mentioned criteria are in most cases sufficient to roughly classify the vast amount of works on admission control. However, real admission control systems can also be distinguished by a number of further criteria, for example, by whether they are *pre-emptive*.

- *Non-preemptive* admission control systems do not interrupt flows once they have been admitted while
- *pre-emptive* systems can interrupt an admitted flow in order to free resources for another flow, see for example, Yavatkar *et al.* (2000).

Access to different *network resources* can be managed by the admission control system.

- Link *bandwidth* is practically always used as the central resource.
- Additionally, some systems also check the availability of *buffer space*, for example, Shenker *et al.* (1997).

The *granularity* of the system describes which type of flows form the decision objects of the system, ranging from

- individual micro-flows (specified by the source and sink IP address, port and the protocol number) over
- sessions that can consist of multiple flows, senders and/or receivers (e.g. Intserv/RSVP)
- up to large aggregated macro-flows identified by other means.

The *timing behaviour* of the system describes whether the flows also specify their (expected) duration and whether this information is used for the admission control test. This is especially important if the system also supports *reservation in advance* (see e.g. Karsten *et al.* (1999)). Reservation in advance allows customers to request resources long before the actual transmission is started.

After this overview and classification of admission control mechanisms, it is also important to stress that besides testing the availability of resources before admitting a new flow – which the above-mentioned works do in a wide variety of different ways – it is also important for an INSP to apply certain *policies* to the admission control decision. With *policy,* we describe all kinds of non-technical rules that are applied besides technical rules to a certain decision. For the admission control decision, the technical rules are the ones that check the resource availability (see preceding text) while non-technical rules – policies – in that context can, for example, check the identity of the user, his contract and his solvency. On the basis of the policies a flow might be rejected despite resources being available. We do not further investigate the support of policies here but instead refer to Durham *et al.* (2000); Herzog (2000); Yavatkar *et al.* (2000).

6.7 Summary and Conclusions

In this chapter, network architectures were discussed. A network architecture consists of the QoS architecture, the data forwarding architecture, the signalling and the security architecture. The most commonly used QoS architecture is the plain best-effort architecture although Diffserv is becoming more and more popular as QoS architecture with the increased importance of QoS-sensitive applications like for example, VoIP (see also Chapter 5). We also discussed alternative approaches to QoS architectures that are not supported by the IETF but use interesting and innovative concepts.

With respect to the data forwarding architecture, label switching as provided by MPLS routers is an alternative to the standard approach of plain IP routing and is gaining importance. The signalling architecture encompasses the routing protocols, the QoS signalling protocols (if used), and the LDPs (if used). The security architecture adds cryptography-based security services at the IP layer. At the end of this chapter, we discussed the broad spectrum of admission control mechanisms. Admission control can be used to control the network load on a small timescale by not admitting certain traffic flows or customers to the network or at least to certain (high quality) traffic classes.

In the following two chapters of Part II of this book, different QoS systems are evaluated. Chapter 7 does so on an abstract level using two analytical approaches while Chapter 8 uses implementations of systems based on the IETF QoS architectures in an experimental study.