Evolution of the Internet QoS and Support for Soft Real-Time Applications

MOHAMED A. EL-GENDY, MEMBER, IEEE, ABHIJIT BOSE, MEMBER, IEEE, AND
KANG G. SHIN, FELLOW, IEEE

Invited Paper

The past few years have witnessed the emergence of many real-time networked applications on the Internet. These types of applications require special support from the underlying network such as reliability, timeliness, and guaranteed delivery, as well as different levels of service quality. Unfortunately, this support is not available within the current "best-effort" Internet architecture. In this paper, we review several mechanisms and frameworks proposed to provide network- and application-level quality of service (QoS) in the next-generation Internet. We first discuss the QoS requirements of many of the above-mentioned real-time applications, and then we categorize them according to the required service levels. We also describe the various building blocks often used in QoS approaches. We briefly present asynchronous transfer mode (ATM) and Internet Protocol precedence. Then, we present and compare two service architectures recently adopted by the Internet Engineering Task Force, called integrated services (IntServ) and differentiated services (DiffServ), for providing per-flow and aggregated-flow service guarantees, respectively. We focus on DiffServ because it is a candidate QoS framework to be used in next-generation Internet along with multiprotocol label switching and traffic engineering. We also examine several operational and research issues that need to be resolved before such frameworks can be put in practice.

Keywords**—***Differentiated services (DiffServ), integrated services (IntServ), quality of service (QoS), real-time applications, scalability.*

I. Internet Quality of Service

Real-time networked applications depend on network parameters such as bandwidth, delay, jitter (interpacket delay variation), and loss for their correct operation. The degree of tolerance or sensitivity to each of these parameters varies widely from one application to another. Critical applications, such as remote surgery and online trading systems, require reliability of such parameters as well as guaranteed delivery. Emerging applications such as home networking, intelligent appliances, factory supply-chain networks, as well as the vast majority of multimedia applications also require different levels of service quality from the underlying network in terms of these parameters.

In traditional circuit-switched networks such as the telephone system, the quality of a connection can be measured and guaranteed in terms of connection setup delays, media quality (e.g., voice or video quality), and trunk availability. On the other hand, packet-switched networks, such as the current Internet, were not designed to provide per-connection service guarantees. As a result, packets of a particular flow can reach their destination via different paths and experience different amounts of delay, jitter, and loss along the way. This is a critical problem on the current Internet as more and more applications, originally developed for circuit-switched networks, are being migrated onto the Internet.

The current Internet architecture is based on the "best-effort" service model, which has worked well for traditional applications such as e-mail, file transfer protocol (FTP), telnet, and hypertext transfer protocol (HTTP). Most of these applications are based on the transmission control protocol (TCP) suite, which provides reliable data delivery service without guaranteeing any delay and jitter bounds. On the other hand, supporting real-time and business-critical applications over a wide-area network requires classification of the application traffic and service-level guarantees from the underlying network for each class of traffic based on a number of factors. A number of proposals have been put forward to enhance the reliability and efficiency of the current Internet. A major objective of these proposals is to provide multiple service classes for either individual flows or flow aggregates with a quality that is suitable for these flows. Once implemented, such service classes will enable

Manuscript received August 21, 2002; revised February 10, 2003. This work was supported in part by Samsung Electronics and in part by the United States Office of Naval Research under Grant N00014-99-1-0465.

The authors are with the Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, MI 48109 USA (e-mail: mgendy@eecs.umich.edu; abose@eecs.umich.edu; kgshin@eecs.umich.edu).

Digital Object Identifier 10.1109/JPROC.2003.814615

the Internet to deliver the growing volume of real-time and mission-critical data to the end users. Quality of service (QoS) is defined in [1] as “the capability to provide resource assurance and service differentiation in a network.”

The purpose of this paper is to identify application requirements for QoS support on the Internet and critically review the proposed approaches to providing such support.

The paper is organized as follows. First, we describe various real-time applications and their QoS requirements, over and above the best-effort service. We emphasize real-time multimedia applications as they are expected to grow significantly in the coming years. A primer on QoS requirements and parameters is given in Section II, and the basic building blocks used to realize these QoS requirements are covered in Section III. Next, we describe the asynchronous transfer mode (ATM) in Section IV as one of the first network infrastructures that support QoS and is closing the gap between circuit-switched and packet-switched networks. In Section V, we cover Internet Protocol (IP) precedence and IP type of service (ToS) field originally placed in IP packets headers and mention why it was not successful. Section VI presents integrated services (IntServ) as one of the initial frameworks for supporting QoS on IP-based networks. Section VII describes differentiated services (DiffServ)—a promising and scalable architecture recently proposed by the Internet Engineering Task Force (IETF) for providing QoS support in the Internet. We also compare DiffServ with IntServ and identify the pros and cons of each. We briefly describe the multiprotocol label switching (MPLS) in Section VIII as a general framework for integrating heterogeneous QoS systems and supporting a new data-forwarding infrastructure, and discuss the interoperation between MPLS and DiffServ as well as the use of MPLS in *traffic engineering*. Section IX presents the operational and research issues for obtaining end-to-end (e2e) and application-level QoS support on the Internet using the DiffServ framework as the candidate framework. Finally, we conclude the paper in Section X.

A. Real-Time Applications and the Need for QoS Support

There have been well-known debates about the need for QoS on the Internet. While managing network performance and utilization by using the various QoS techniques is reasonable, the opponents of QoS frequently suggest adding more bandwidth (i.e., *overprovisioning*) to solve problems related to congestion, packet loss, and delays, as a large amount of bandwidth is becoming available at a cheaper price. However, many real-time applications such as voice over IP (VoIP), video-conferencing, and telemedicine require guarantees on delay, jitter, and packet loss, not just bandwidth. Furthermore, the advent of advanced multimedia applications and the growing use of the Internet in our everyday lives are expected to use up the offered bandwidth very quickly and return to the same (bandwidth-limited) situation again. In the presence of other application traffic (such as telnet, WWW traffic, and bulk file transfers) on the same network, packets from real-time applications can experience varying and unpredictable amounts of delay, jitter, and packet loss, especially in the absence of any prioritization. Therefore, one of the advantages of installing QoS mechanisms in any network is to provide prioritization and protection to chosen traffic streams. It can also protect time-critical packets in case of congestion since almost all network components experience peak usage at certain times. Recently, the debate has given way to a more balanced approach of combining both overprovisioning and installing specific service classes for selected applications. This reduces the complexity of managing a tightly controlled QoS-aware network, which often requires setting up multiple policies and reservation servers, as well as admission control elements at the edges of the network. Another argument in favor of incorporating QoS comes from the disparity of available bandwidths in the core and edges of the Internet. The domains at the edges of the Internet typically have more congestion than the core. Therefore, there is a clear need for prioritization and protection of real-time and mission-critical data packets at the edge routers.

In what follows, we categorize various real-time and multimedia applications in terms of their behavior and resource requirements. Examples of such applications are video-on-demand (VoD), IP telephony, VoIP, Internet radio, multimedia WWW, teleconferencing, interactive games, distance learning, remote medical surgery, High Definition TV (HDTV), and large-scale distributed computing using data grids. The service requirements of these applications differ widely from one another and significantly from the best-effort service of the current Internet. Multimedia applications are usually sensitive to e2e delay and delay variation (jitter), but many of them can tolerate packet losses to some extent. On the other hand, TCP (upon which many Internet applications were based) was designed to give the highest assurance of reliable delivery of data without any consideration to delay and jitter. Other real-time applications such as remote administration, transactions, and factory automation require guarantees on both delay and losses. As an example, VoIP calls require an e2e delay of 150–300 ms for proper comprehension of voice. A longer delay would result in poor and imperceptible voice quality.

Within the current best-effort service model of the Internet, new protocols have been developed and standardized for supporting multimedia applications. Examples of such protocols are real-time protocol (RTP), real-time control protocol (RTCP), and streaming protocol (RTSP), as well as frameworks such as H.323 for video-conferencing applications. Many multimedia applications also employ adaptive techniques for dealing with packet loss or unexpected network conditions. For example, an adaptive play-out buffer can be used for delayed packets or frames. Similarly, forward error correction (FEC) [2] algorithms can be employed to compensate for packet or frame losses. Most of these techniques address the lack of QoS support from the network layer in the current Internet infrastructure. The next-generation Internet should be designed to recognize the service requirements of each application so that a specific “service class” can be assigned to each flow from these applications instead of putting them all in the best-effort service.

B. Categories of QoS-Dependent Applications

This section categorizes current and next-generation Internet applications [3] according to their QoS requirements and behavior.

1) Interactive Versus Noninteractive: Interactive applications are usually human-to-human or human-to-machine applications that involve a sequence of interactions and transfer of information among the endpoints of the application. Some cases of machine-to-machine applications are interactive as in two-way transactions, automated control, monitoring, voice response systems, and others. These types of interactive applications may depend on a number of QoS parameters such as bandwidth, delay, jitter, and loss. As mentioned before, IP-telephone calls need an e2e delay less than 300 ms [4]. Both parties can adapt to delays less than this bound, although noticeable service degradation occurs for delays beyond 150 ms. Jitter is acceptable in voice conversations up to 75 ms, beyond which it is difficult to adapt to the varying quality of the voice conversation. Compensation for jitter involves using jitter buffers but the buffers increase e2e delays and, therefore, should be used with care.

Noninteractive applications do not interact among the endpoints. Examples of such applications are data backup and bulk file transfer. These applications typically have bandwidth requirements to provide reasonable performance.

2) Elastic Versus Inelastic: Elastic applications can work under a variety of network conditions and still perform correctly. This type of application usually does not require any QoS support from the network other than the best-effort service and transport reliability that can be achieved by using the TCP protocol. On the other hand, most real-time applications are inelastic and require QoS guarantees from the underlying network for a certain performance level. Real-time applications are further categorized as "hard" or "soft." Hard real-time applications cannot function at all if their QoS needs are not met by the network at all times, while soft real-time applications, like multimedia, can tolerate some degradation in QoS for a short period of time and operate with lower quality.

3) Tolerant Versus Intolerant: Tolerant applications are not to be confused with elastic ones. While elastic applications do not impose any QoS requirements, tolerant applications impose QoS requirements but with ranges or levels that can allow the application to run even if the optimal QoS levels are not provided. Tolerant applications are usually inelastic with ranges of QoS requirements, but if their QoS bounds are violated, they cannot run correctly. An example of such application is IP-telephony. Again, hard real-time applications are examples of intolerant applications. Any application that specifies fixed values for its QoS requirements and cannot run if these values are not supported is an intolerant application. An example of tolerant application is VoD. The application can tolerate losses and a certain amount of delay and will still be able to run without complete halt. Some video compression techniques (e.g., MPEG) can tolerate losses in frames and use relationships between successive frames to predict the lost ones.

4) Adaptive Versus Nonadaptive: QoS-dependent applications can be further divided into adaptive and nonadaptive applications. Adaptive applications try to maintain the perceived quality at an acceptable level, even under poor network conditions. This can be done by lowering the sending rate or by lowering the resolution of the transmission (e.g., in video transfer) or even by using specialized compression and error-correction techniques. Adaptive applications may use extra buffering to compensate for network transients and allow for graceful degradation in performance. Most audio and video streaming applications on the Internet are adaptive. Nonadaptive applications do not have the ability to cope with network transients—they can still tolerate some QoS degradation, but this directly affects the quality perceived by an end user. Adaptivity and tolerance are considered two different dimensions for real-time applications [5]. In most cases, adaptive applications are tolerant and nonadaptive applications are intolerant, but other combinations may also be found.

5) Real-Time Video/Audio Versus Streaming: Multimedia audio/video applications fall into real-time audio/video transmission such as Internet radio/TV broadcasts, IP-telephony, and video-conferencing (H.323), and nonreal-time streaming such as VoD applications. Real-time video/audio applications have more strict requirements on QoS and usually employ adaptive techniques to cope with network transients. On the other hand, streaming video or audio applications can delay their playback point with the maximum delay of the network to get over delay jitter, but this might cause buffer overflow and loss of data.

6) Multimedia Versus Large-Scale Data and Computation: Not all QoS-dependent applications involve the transmission of audio and video data. There are many other real-time applications that require timely and reliable delivery of sampled or periodic data. Examples of this type of application are grid computing applications, electronic-trading transactions (e.g., stock exchanges) and remotely controlled instruments. Distributed collaboration systems also require large volumes of data (often in terabytes) to be exchanged and processed among geographically distributed locations and within a certain time interval.

It is expected that with the advance of broadband at home and metro area networks, future real-time applications will require large amounts of network bandwidth and, at the same time, need certain service-level assurances (see the QoS debate in Section I-A). For example, video transmissions require a large amount of bandwidth. Throughput requirements range from 300 to 800 kb/s for video-conferencing applications to 19.2–1500 Mb/s (1.5 Gb/s [6]) for HDTV to transmit high-quality noncompressed video frames integrated with high-quality audio (similar to DVD picture and audio quality or better).

The quality of a video transmission is sensitive to data loss since most video compression techniques seek to reduce redundancies between successive frames. The effect of loss on video quality is also influenced by parameters such as the encoding technique used, loss rate, loss pattern, and transmission packet size. The effects of loss of large packets are often more pronounced than the loss of smaller ones. The end-user video quality also depends on the type of frames that are lost, e.g., for MPEG encoding, loss of I or P-frames is always severer than loss of B-frames. In case of a multimedia stream consisting of both audio and video flows, the synchronization between the two flows is important for the overall quality. Therefore, one has to guarantee quality of both audio and video streams in this case.

A most common standard for measuring voice and video quality is called *mean opinion score* (MOS) (ITU-T Recommendation P.800), which involves a large number of human listeners and takes the statistical average of their opinions about the quality of the media transferred.

II. QoS Requirements

Before delving into the various QoS support frameworks, we briefly discuss the common parameters that have been widely used to describe QoS requirements. We also identify important issues in any QoS framework, such as service commitment and admission control.

A. QoS Parameters

Specification of QoS parameters usually depends on the context of the applications involved. Different applications or end systems may have different interpretations for their QoS requirements, but the following parameters are considered as the basic form of QoS because other forms can always be mapped to them [7].

- **Throughput** is the effective number of data units transported per unit time (e.g., bits/second). This parameter is usually specified as a "bandwidth guarantee." The traffic entitled to this bandwidth guarantee can be modeled either as "constant bit rate," specifying only a fixed transmission rate, or as other models like "leaky buckets," specifying a rate (r) and a burst size (b).[1] The bandwidth guarantee involves allocation of the link capacity as well as processing capacity of the intermediate nodes. A bandwidth bottleneck can jeopardize the bandwidth guarantee for an entire e2e path.
- **Delay** is the time interval between the departure of data from the source to the arrival at the destination. This is usually referred to as e2e delay. The source and the destination can span multiple layers such as application layer, transport layer, network layer, link layer, or even physical layer, and different delays are associated with different layers. The required performance is expressed in terms of the maximum delay bound $D_{\max}$. Measuring *one-way* delay is usually difficult because of clock synchronization problems. Sometimes *round-trip* delay is used, instead, as a representation for the delay bound, but due to typical dissimilarities between forward and return paths on the Internet, this may not give a good indication for the delay parameter.

  Fig. 1 illustrates the major components of delay that a packet experiences in a network, while Fig. 2 shows the typical probability density function for delay [1].

[1]Other models are also available, but are not discussed here.

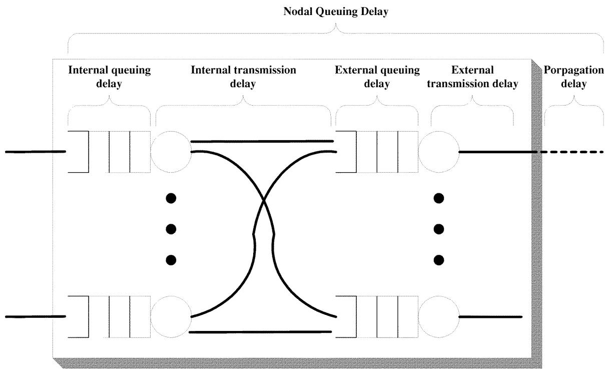

Fig. 1. Components of packet delay.

- **Jitter** is usually referred to as "delay variation" as in the case of ATM. Like delay, the jitter bound is specified by the maximum value $J_{\max}$. There are various definitions for how jitter can be quantified, but we will present only three [7].
  1) Jitter can be calculated as the difference between the interdeparture times I_i and the interarrival times A_i of the ith and the $(i-1)$th data units, i.e., $J_i = I_i - A_i$.
  2) Jitter can be calculated as the difference between the delays of the ith and the $(i+1)$th data units, i.e., $J_i = D_{i+1} - D_i$.
  3) In RTP standards [8], jitter is measured accumulatively through the following smoothing equation:

$$J = J + \frac{(|D(i-1,i)| - J)}{16}.$$

- **Loss** is the percentage of data units that did not make it to the destination in a specific time interval. It is usually represented as a "probability" of loss. Retransmission, however, does not change the value of the loss probability of the network, but it is a method for loss recovery.
- **Reliability** is not to be confused by loss, although sometimes it subsumes loss. By reliability, here we mean the correct delivery of data units to their destination. To show the difference between reliability and loss, a "reliable" protocol can use retransmission to recover from losses and deliver reliable service to the upper layer. Errors, misinsertion (like in ATM), duplicate, and others are examples of lack of reliability. In some systems, mean time to failure (MTTF) [9] and other failure rate measures contribute to reliability.

There are other QoS parameters such as availability and security that can affect application performance, but because of space limitation, we omit their discussion.

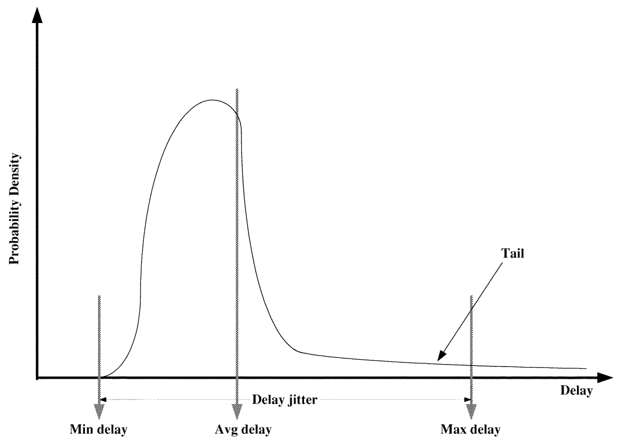

Fig. 2. Packet delay probability density.

B. Service Commitment

In any QoS framework, there should be a specification for the *service commitment* [5]. Two kinds of service commitment are usually offered by network providers—*guaranteed* and *predicted* services. The former level of commitment specifies *a priori* bounds on service parameters, while the latter specifies the service expected from the network given the current status.[2] Another way of characterization is *quantitative* versus *qualitative* services. While the former specifies numbers and "quantities" for the service level, the latter usually specifies relative levels such as *better than* or *low loss*. Both levels of service are found in the QoS frameworks discussed in this paper. Quantitative service levels can provide *deterministic* or *statistical* guarantees such as percentile or average values.

In order to have a valid service commitment, the network provider has to know information about the traffic entitled to this service. This is called *traffic profile*, and it is a description of the input traffic in terms of average rate, burstiness, distribution, packet size, etc.

C. Admission Control

Regardless of the QoS approach used, the network still has finite amounts of resources (in terms of link and node-buffer capacities, and node processing power). The key behind QoS-enabled networks is how to distribute these resources appropriately among their customers to meet their service requirements. Admission control is used as a protection against oversubscription of the available resources. It usually employs comparison between the service requirements and the resources available and then decides to accept or reject the service requests.

Admission control can be done either explicitly or implicitly. Traffic conditioning (policing and shaping) is an example of implicit admission control. Policing is used to limit the amount of traffic input to the network according to a certain profile. Admission control can be further categorized as "predefined" and "measurement-based" [10].

[2]Typically via measurements.

III. Building Blocks for Providing Network-Level QoS

Providing network-level QoS requires extra functionalities from the network devices beyond packet forwarding and routing. These functionalities include classifications, queueing and scheduling, policing and shaping, and buffer management [11]. In this section, we review the common building blocks for these functionalities used in a QoS-enabled network. There can be many combinations of the basic building blocks—a particular choice depends on hardware capabilities, type of applications, and desired QoS guarantees.

A. Scheduling

A number of scheduling algorithms have been proposed for network-layer packet handling [9]. These algorithms seek to allocate link bandwidth among the various classes of traffic in *a priori* or fair manner, and provide statistical, aggregate or per-flow guarantees on network parameters such as delay, jitter, and packet loss. Most scheduling algorithms can be divided into two classes [12]: 1) a *work-conserving* scheduler is not idle when there is a packet to transmit in any of its queues and 2) a *nonwork-conserving* scheduler may choose to remain idle even if there is a packet waiting to be served. The latter may be useful in reducing burstiness of the traffic or in providing a strict guarantee for a particular class of traffic.

- **First in first out (FIFO) scheduling:** FIFO provides the simplest scheduling mechanism—packets are served in the order they are received. The delay and packet-loss properties are directly proportional to the buffer size available at the queue. However, no guarantees can be provided to individual flows, and moreover, the FIFO scheduling works best when all flows behave in the same way. Therefore, FIFO scheduling is not suitable for providing service differentiation and QoS guarantees [9].
- **Priority scheduling:** A static priority scheduler is based on multiple FIFO queues where each queue is assigned a priority parameter. The queues are served in the order of their priority. There is also a preemptive version of priority scheduling in which a packet from a lower-priority queue already in transmission may be delayed or dropped if a high-priority packet arrives at the interface. However, most routers deploy nonpreemptive scheduling, and a small delay is added to any high-priority packet awaiting service while a packet is being transmitted over the outgoing link. Priority scheduling [9] is able to give predictable performance for the case when all flows entering the scheduler (irrespective of the priority queues) have the same packet size and input rate. Priority queueing can be implemented easily since it requires maintenance of only a small number of states per queue. Consider a priority scheduler with k priorities $(p_1 < p_2 < \cdots < p_k)$, uniform packet size of B bits, and a link capacity

of C bits/s. The delay experienced by a packet of priority p_i (at the head of queue i) can be written as

$$D_i = \frac{B}{C} \times \sum_{j=i+1}^{j=k} n_j$$

where n_j is the number of packets in priority-j queue. This means that a packet must wait for all other higher-priority packets to be forwarded. This can cause *starvation* to lower-priority classes.

- **Generalized processor sharing (GPS) and variants:** To get over the starvation problem in priority scheduling, GPS scheduling assigns a logical queue for each flow, and the scheduler serves an infinitesimal amount of data from each queue within a given quantum or finite time interval [13], [14]. GPS is ideal in achieving the max–min fair allocation; however, the scheme is not implementable due to the infinitesimal data requirement. Instead, variations such as round robin (RR) and weighted-round robin (WRR) schemes are often used as simple implementations for GPS. In RR scheduling [15], each queue outputs a packet (instead of an infinitesimal amount of data) in a round-robin fashion within each specified time-frame or cycle. This is a fair scheduling method since even the high-throughput or bursty flows output only one packet per cycle. However, this only works if the packet sizes of all flows are equal. As pointed out in [9], for flows with different packet sizes, the fairness in bandwidth allocation is no longer provided. Flows with larger packet sizes will consume more share of bandwidth than flows with smaller packets. WRR [16] was developed to address this issue. It outputs n packets instead of a single packet from a queue in each cycle where n is a weight of the queue. Therefore, flows with smaller packet sizes can be served more often and get a fair share. However, WRR is not efficient in handling variable-sized packets in a single flow since it works best when the mean packet size of a queue is known *a priori*. The deficit round robin (DRR) scheme [17] addresses this problem by keeping track of queues that are not able to transmit a packet in the previous round because of a large packet size, and by adding the deficit (the remainder from the previous quantum) to the quantum of the next round.
- **Weighted fair queueing (WFQ):** A WFQ scheduler [18] prevents the starvation problem for the lower priority queues in priority scheduling while trying to approximate the GPS scheduling [14]. WFQ calculates the *finish time* for each packet as if it was served by GPS and then use this time stamp to order the service of packets. The calculated finish time is

$$F_i^k = \max\left[F_i^{k-1}, R(t)\right] + \frac{L_i^k}{\phi_i}$$

where F_i^k and L_i^k are the finish time and the length, respectively, of the kth packet in flow $i, R(t)$ is called the round number and it is taken from a bit-by-bit RR scheduler, and ϕ_i is the weight used for flow i. WFQ provides a delay bound for guaranteed flows that follow the leaky bucket model (r, b) [13]. Consider a set of n WFQ queues with weights $(w_1, w_2, \ldots, w_n)$. The queue with weight w_1 will have a fraction equal to $w_1/(\sum_i w_i)$ of the total bandwidth of the outgoing link. If there are no packets waiting, any queue can be allowed to borrow the full link capacity. The maximum delay experienced by a packet in a burst of b_i packets is given by [9]

$$D_{\max} = b_i \bigg/ \left(\frac{R \times w_i}{\sum w_j}\right)$$

where R is the service rate of the flow, and w_i is the weight assigned to flow i. The worst case delay for a K number of WFQ hops is given by [14][3]

$$D_i^{\max} \leq \frac{b_i}{r_i} + \frac{(K-1)L_{\max}}{r_i} + \sum_{j=1}^{K} \frac{L_{\max}}{C_j}$$

where C_j is the output link rate for node j, and $L_{\max}$ is the maximum packet size.

Because WFQ allows a fair share of bandwidth among all the queues, it is one of the most popular scheduling algorithms implemented in commercial routers. It is suitable for traffic with variable-sized packets such as the Internet. One of the disadvantages of WFQ is the need to maintain per-flow queueing information since an appropriate weight for each flow must be generated. Several variations of WFQ have been proposed, such as self-clocked fair queueing (SCFQ) [19] and start-time fair queueing (STFQ) [20], in order to reduce the complexity of per-flow queueing.
- **Class-based queueing (CBQ):** CBQ [21] is a class of link-sharing scheduling algorithms that enables a hierarchical division of bandwidth among various classes of traffic for a particular link in times of congestion as shown in Fig. 3.

These algorithms create a sharing tree for all classes to be supported for a link. Both interior and leaf classes should receive its allocated link-sharing bandwidth over a specified time interval. Moreover, any excess bandwidth in the link should be distributed among the classes according to a sharing policy. A link-sharing structure may mark classes as *exempt*, *bounded*, or *isolated*. An exempt class is allowed to have 100% of the total link bandwidth. However, the scheduler and admissions control schemes ensure that the traffic from this class is within the limits of the link sharing goals. A bounded class is not allowed to borrow any excess bandwidth from any of its parent classes in the sharing tree, whereas an isolated class does not allow classes from a different branch to borrow its unused bandwidth and does not borrow from other classes

[3]This equation has been simplified.

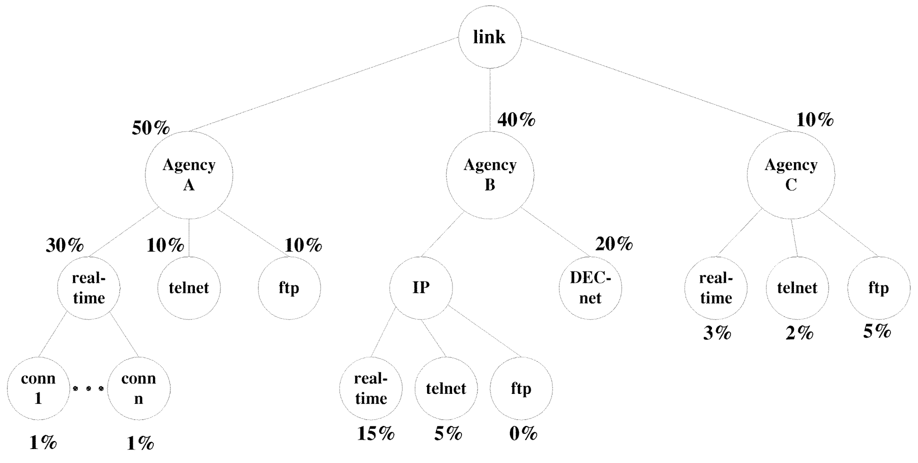

Fig. 3. Hierarchical link sharing structure (courtesy of [21]).

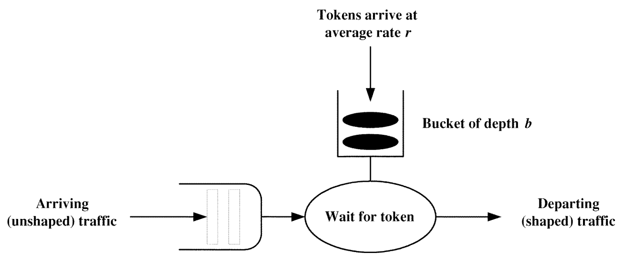
Tokens arrive at average rate *r*
Bucket of depth *b*
Arriving (unshaped) traffic
Wait for token
Departing (shaped) traffic

Fig. 4. Traffic shaping with token bucket filter (TBF).

either. In practice, the link-sharing approach is used in conjunction with priority scheduling.

B. Buffer Management

Random early detection (RED) [22] gateways are often employed to avoid congestion in packet networks by detecting the onset of congestion and by dropping packets arriving at the gateway. The RED algorithm calculates an average queue size using a low-pass filter with an exponential weighted moving average (EWMA). The queue size is compared to predetermined threshold values. When the average queue size is less than the minimum threshold, no packets are dropped. However, when it is larger than the maximum threshold, the algorithm drops each arriving packet. In between these two limits, the algorithm drops each packet with a probability p that is based on the average queue size. Since the average queue size is controlled during transient congestion, RED gateways can provide high throughput and low average delay for high-speed TCP connections, as well as accommodate short bursts (e.g., during TCP slow-start phase). Weighted RED (WRED) [23] is a variant of the RED algorithm, sometimes called multi-RED, that drops packets selectively based on IP precedence, i.e., packets with higher IP precedence bits have a lower probability of being dropped than packets with lower precedence. WRED is designed for the core of a network—the packets entering the network are marked with appropriate IP precedence bits at the edge routers and, hence, have differentiated drop probabilities.

C. Policing

Traffic policing is typically deployed at the edge of a network and/or close to the source. Upon arrival of a packet, a policing algorithm first determines if the packet is in compliance with the service-level agreement negotiated between the source of the traffic and the network. If not, it may drop the packet entirely or decrease its priority based on policy and current priority of the packet. Traffic policing can be based on a single negotiated parameter, such as the peak rate, or a combination of parameters, such as peak rate, burst size, time of day, etc.

The token buckets [24] (and leaky buckets) are the most common mechanisms used for policing traffic at a network node. A token bucket has a bucket of depth b and generates tokens at the rate of r. Each arriving packet consumes a token (or a number of tokens directly proportional to the packet size, depending on implementation) before it can be transmitted into the network. A flow is considered *in-profile* if its average bit rate is less than or equal to r and its burst size is less than or equal to b. At any given time period t, the maximum amount of traffic allowed equals $r \times t + b$. Flows that violate these conditions are considered *out-of-profile* and may be dropped or marked for another class of service at the router. A dual leaky bucket (DLB) is often used for policing average and peak rates of traffic. Besides policing, token buckets can also be used for traffic shaping as shown in Fig. 4.

As mentioned earlier, any QoS framework needs a combination of schedulers, classes (or queues), and policing mechanisms to provide per-flow or per-traffic class guarantees. For example, Fig. 5 shows a realization of several of the above schemes to construct three different per-hop forwarding behaviors (in DiffServ). The schedulers, classes, and filters shown in the figure are part of the traffic control mechanisms built into the standard Linux kernel. Other operating systems such as Windows 2000 (WinSock 2) and AIX also support the above QoS building blocks. Many of the scheduling and policing schemes are increasingly found in commercial networking equipment and routers [23].

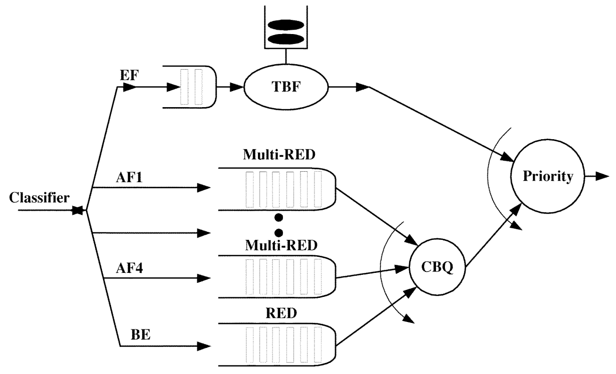

Fig. 5. Implementation of QoS building blocks.

IV. ATM

ATM [2], [25], [26] was one of the first frameworks proposed to explicitly support QoS. The history of the ATM [14] goes back to the plain old telephone service (POTS), where support for voice calls was the only requirement. With the advent of computers and data technologies, support for data or packet networks began to take place in addition to voice calls. Transcending from integrated services digital network (ISDN) and "broadband" or B-ISDN, ATM was proposed to handle integrated services of data and voice traffic. In this section, we briefly present the basic ATM principles that relate to QoS.

The ATM technology was developed to optimize bandwidth utilization while ensuring different QoS levels by using a combination of traffic control and management. Furthermore, ATM can handle diverse access speeds and adapts itself easily to nonnative ATM traffic while consolidating traffic from various protocols over a single network infrastructure. ATM is a rather complex framework for supporting QoS. The complexity of managing ATM traffic—which hampered its deployment—is not inherent to the ATM technology, but it comes from the attempt to resolve the conflicting goals of guaranteeing QoS and maximizing network utilization.

A key point of the ATM operation is the use of fixed-size data units (called cells) to make scheduling, queueing, and buffer management easier than dealing with variable-sized packets and, hence, more predictable behavior can be estimated. Moreover, the establishment of virtual connections (VCs) and virtual paths (VPs) to carry the cells provide robust QoS facilities.

ATM provides several service categories [27] such as constant bit rate (CBR) with dedicated bandwidth, extremely low probability of cell loss, as well as low and predictable delay. This is different from variable bit rate (VBR), which was designed for more bursty traffic like video. Other service categories such as available bit rate (ABR), guaranteed frame rate (GFR), and unspecified bit rate (UBR) are also supported.

QoS in ATM is specified by the cell loss rate (CLR), maximum cell-transfer delay (Max-CTD), peak-to-peak cell delay variation (P2P-CDV), severely errored cell block ratio (SECBR), cell misinsertion rate (CMR), and cell error ratio (CER).

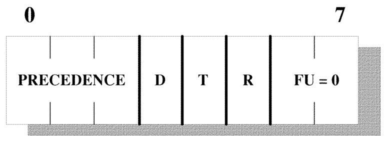

Fig. 6. IP ToS field.

In order to provide predictable QoS, it is necessary to know the traffic parameters. ATM uses the term "traffic descriptors" for this purpose. Traffic descriptors consist of peak cell rate (PCR), sustained cell rate (SCR), maximum burst size (MBS), minimum cell rate (MCR), and maximum frame size (MFS).

Now, ATM is found in the optical backbone of the Internet. This was encouraged through the adoption of the Frame-based ATM over synchronous optical network/synchronous digital hierarchy (Sonet/SDH) transport (FAST) specification as an industry standard for heavily data-oriented ATM networks (such as OC-1, OC-3, and OC-12). In this way, ATM continues to add value to IP networks and enables them to scale while simultaneously enabling other non-IP applications and services to reside on the same core infrastructure.

The main reasons for ATM being confined to the core of the Internet are the difficulty of deploying the ATM interfaces to the end-host applications and the large overhead incurred (e.g., fixed-size cells and virtual circuit establishment).

V. IP Precedence and ToS

IP precedence and the ToS field were first introduced in [28]. These are considered to be the first support of QoS on IP networks. Although ToS has been little used in the past, its use by hosts is now mandated by the requirements for Internet. Many of the router vendors now support ToS and IP precedence as a first aid solution for QoS and value-added services [23]. ISPs are also using IP precedence and ToS to provide QoS to their customers in a simple and easy manner. The notion of precedence was defined broadly as "an independent measure of the importance of this datagram." Not all values of the IP precedence field were assumed to have meaning across boundaries; for instance, "The Network Control precedence designation is intended to be used within a network only. The actual use and control of that designation is up to each network" [28], [29].

The ToS field in an IP datagram header provides an indication for the QoS required for this datagram. It is used in selecting the appropriate service parameters at network elements. The main choice is a three-way tradeoff among low delay, high reliability, and high throughput. As shown in Fig. 6, bits 0–2 are used for precedence. Precedence can take one of eight values as shown in Table 1. Bit 3 is used for delay (D) specification. $D = 0$ indicates "normal delay," and $D = 1$ indicates "low delay." Bit 4 is used for throughput (T). $T = 0$ indicates "normal throughput," and $T = 1$ indicates "high throughput." Bit 5 is used for reliability (R). $R = 0$ indicates "normal reliability," and $R = 1$ indicates "high reliability." Bits 6–7 are reserved for future use (FU).

Table 1
IP Precedence

Bits	Precedence
111	Network Control
110	Internetwork Control
101	Critical
100	Flash Override
011	Flash
010	Immediate
001	Priority
000	Routine

The network can choose service mapping to map these values to appropriate service classes implemented in the network. The mapping technique and the service classes are not part of the IP ToS specifications. It is worth mentioning that in IP version 6 (IPv6), the IP ToS field name has been changed to the *class field*, and another field called *flow label* was added. Flow label was targeted to serve the IntServ, and class field was targeted to serve the DiffServ. Both are considered as QoS support features of IPv6 [30].

The main reason why IP ToS and IP precedence did not work as a viable solution for supporting QoS on the Internet is the absence of strict rules for processing the IP ToS field in IP routers. Therefore, incompatibilities and lack of support hampered their deployment. As will be shown in Section VII, DiffServ provides compatibility with IP precedence through the "class selector" per-hop behavior. This allows fast and easy deployment for DiffServ, especially during early deployment stages.

VI. INTSERV AND RSVP

The main difference between ATM and IP is that the former is connection-oriented and the latter is connectionless. The connectionless property of the IP and, hence, the Internet, was required for scalability and fault-tolerance purposes. IP is usually associated with the concept of "datagrams" and "per-hop" routing and, by nature, does not support virtual circuits or virtual paths. Each packet (datagram) contains full information about its source and destination and, hence, routing across intermediate routers can be done to deliver the packet to its final destination. The IP and, hence, the Internet, were based on the "best-effort" delivery of datagrams. There are no guarantees on datagrams' delivery, so there are no provisioning, traffic conditioning, admission control, or traffic protection. Datagrams can be delayed, jittered, mangled, or even lost (dropped) on their way to destinations.

On the contrary, ATM was designed from the beginning to have virtual circuits and paths, traffic conditioning, admission control traffic classes with different guarantees and, hence, QoS. Although ATM and other frame-based schemes have been available for some time, a similar mechanism is necessary for providing network-level QoS for IP-based networks. Since most of the traffic on the Internet uses either TCP or UDP over IP, IntServ was standardized by the IETF to address this need.

The IntServ framework [31] introduces service classes with different traffic characteristics to match the application QoS requirements. Traffic from these service classes is treated differently at the routers with the aid of classifiers, queues, schedulers, and buffer-management schemes. An application in the IntServ environment uses a resource reservation protocol (RSVP) [32], [33] to signal and reserve the appropriate resources at the intermediate routers along the path from its source to destination(s). Like ATM, IntServ provisions virtual paths for each flow and sets up the required resources on these virtual paths. However, routing is not affected by these virtual paths, and they follow the default routing paths provided by the Internet routing protocols.

A. *IntServ Model*

IntServ was designed to provide QoS to individual flows (or an individual session in case of multicast applications). A session requesting specific QoS guarantees is required to initiate a setup procedure using RSVP. RSVP sets up "soft states" in each router along the path from source to destination specifying the class and resource requirements of the initiating session. The reservations remain valid as long as the session is active, but expire if not refreshed periodically, i.e., soft states are used. Using this service model, if there are N individual sessions passing through a router at any time, the router has to maintain the necessary state information for all N sessions. This per-flow service model achieves the maximum flexibility as it can meet QoS requirements of individual flows.

An important aspect of the IntServ model, as mentioned earlier, is that the reservation follows the entire route (i.e., e2e path) of the data packets. If the route changes, the reservation is *automatically* redone to follow the new route. This is also allowed by using the soft-state feature. The service model is unidirectional, i.e., if a two-way communication session needs QoS, it has to reserve resources in both directions.

B. *Service Classes*

The IntServ introduces two additional service classes in addition to the basic best-effort (BE) service of the Internet: guaranteed service (GS) [34], and controlled load (CL) service [35].

1) Guaranteed Service: GS is used by applications that require strict bounds on the e2e delay. The traffic source is modeled by a token bucket profile (r, b), where r is the sending rate and b represents the burstiness of the traffic. Routers supporting GS must allocate a forwarding rate of R for each session with a request rate of $r < R$. In addition, each router is required to compute two parameters, C and D, where C is the per-node rate-dependent queueing delay, and

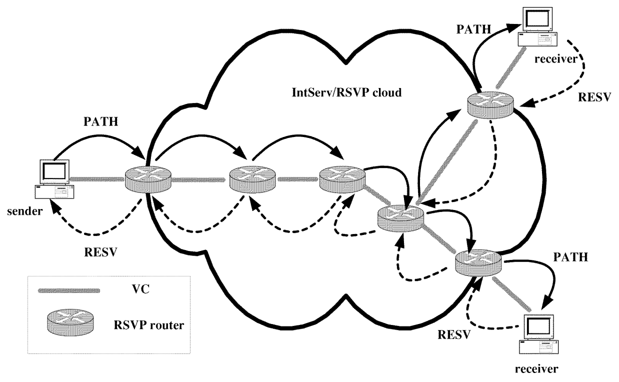

Fig. 7. RSVP reservation.

D is the rate-independent queueing delay. The delay offered by this router is then calculated as $(D+C)/(r)$. Similarly, $C_{\rm tot}$ and $D_{\rm tot}$ represent the "accumulated" rate-dependent and rate-independent queueing delays along the path, respectively. If the session has a peak rate of p, the e2e delay can be bounded [34] by

$$\left[\frac{(b-M)}{R}\times\frac{(p-R)}{(p-r)}\right]+\frac{(M+C_{\rm tot})}{R}+D_{\rm tot}$$

where M is the average packet size for the session.

2) Controlled Load Service: The CL service is qualitative and provides QoS guarantees similar to an unloaded network. CL traffic should experience small queueing delays, low loss, and an overall performance as if the network in not loaded. Although this service is less strict than guaranteed service, it is still better than best-effort, and can be used for today's multimedia applications that are designed to operate well if the network is not loaded but their performance may degrade significantly if the network becomes loaded.

C. Reservation With RSVP

RSVP is a receiver-oriented reservation protocol that is used within an IntServ network to signal QoS requirements of an application session along the path from source to destination. The process of reservation, as described in RFC2210 [33], [36] and illustrated in Fig. 7, starts with the sender sending a PATH message toward the receiver that traverses every router along the path. This message records each of the intermediate routers as part of the forwarding path and calculates the e2e network parameters. Each PATH message carries a traffic specification object called *Tspec* that describes the traffic profile generated by the source. *Tspec* has the following parameters: rate (r), bucket (b), peak rate (p), minimum policed unit (m), and maximum packet size (M).

The PATH message also carries an advertisement specification object (*ADspec*), which is used in computing the accumulated QoS parameters along the e2e path. The contents of the *ADspec* object depends on the service class, i.e., guaranteed or controlled load service. For example, $C_{\rm tot}$ and $D_{\rm tot}$ are parts of an *ADspec* object for guaranteed service. The PATH message carries the reverse path information while it traverses the path from source to destination so that the reverse path can be used to return a RESV confirmation message as explained below.

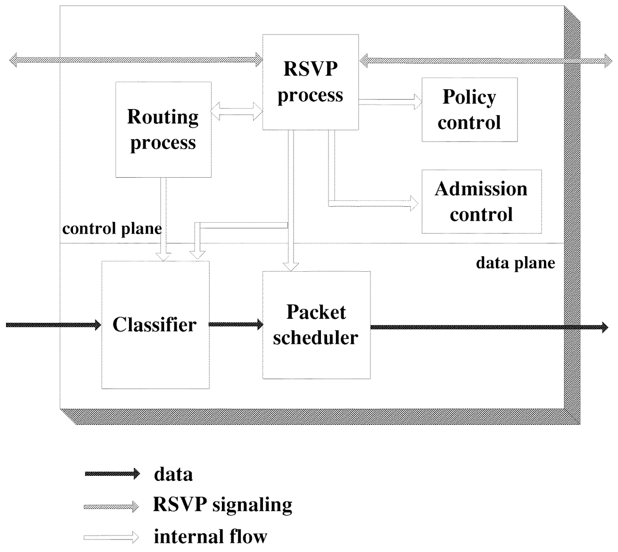

Fig. 8. RSVP router.

When the PATH message reaches the receiver, it replies back with a RESV message to the sender that carries the reservation specification (*Rspec*) contained in a *FlowSpec* object, provided the e2e delay and other parameters are within acceptable limits. This message, if accepted by all routers along the path, sets up the actual reservation and filters on these intermediate routers. This step is known as admission control.

If an error occurs or there are not enough resources to be allocated, then either a PATH*err* or a RESV*err* message is generated by the corresponding router. This message is returned to the sender and any reservation already made on the intermediate routers are canceled along the way. In this case, the session cannot be established with the requested QoS and the specified traffic profile.

Finally, once an application session ends, a PATH*tear* and RESV*tear* messages are sent to remove reservation states on all the routers along the path. Fig. 8 shows a functional block diagram of a typical RSVP-capable router.

Reservation Styles: Filters are set up on routers to classify incoming traffic for different levels of service at each router. Three different types of filters are supported in a RESV message. A *fixed filter* is used for specifying a fixed sender IP address and port number. A *shared filter* can be used for multiple senders. Finally, a *wildcard filter* applies to all senders upstream of the router. As mentioned before, all filters and reservation states kept on routers are "soft," meaning that they will expire if not refreshed periodically. This adds an overhead to the number of messages in an IntServ network.

D. Evaluation of the IntServ Model

The IntServ with RSVP has the following features.

- **Flexibility in meeting QoS needs:** As mentioned in Section VI-A, the resource reservation is done on a per-flow basis and therefore, it can satisfy resource requirements of individual flows.

- **Assured and deterministic QoS:** RSVP messages traverse the same e2e path as application data traffic from source to destination and establish reservation states in each router along this path. This makes the reservation process accurate in terms of providing the required QoS.
- **Adjustments to route changes:** RSVP reservations are soft states and need to be refreshed periodically. The refreshment process detects any route changes during the lifetime of a session and adjusts the reservation path accordingly. This is possible because the RSVP messages follow the same route taken by the data packets.
- **Support of multicast communication:** Since RSVP is a receiver-oriented protocol, it can support multicast sessions as multiple receivers join the multicast and reply with RESV messages to the source. The required resources are then reserved accordingly.

However, to date, the deployment of IntServ has been limited for the following reasons.

- The IntServ model lacks scalability. This is a direct consequence of per-flow resource reservation and traffic handling at the intermediate routers and, therefore, it does not scale in the core routers or in backbone networks.
- Applications have to wait until the reservation using RSVP is complete. This may delay the starting time and may be unacceptable for certain real-time applications that require immediate response to meet strict deadlines. This becomes more problematic for short-lived sessions (e.g., HTTP), where the setup time is much longer than the data-transfer time and becomes more significant.
- Because the resource reservation and the QoS calculations are done *a priori* and as closely as possible to the specified traffic profile given by the applications, any unforeseen and major changes of data traffic because of application upgrade or application change of requirements may have unpredictable effects unless renegotiation and rereservation is done again using the new traffic profiles.
- IntServ is not compatible with the IP security protocol IPsec, because of the multifield classification required at each router in the path to identify individual flows. Since IPsec encrypts the transport-layer headers in a packet, the routers do not have access to these headers for classification. However, this problem has been solved by the introduction of IPv6 [30] flow label as we mentioned in Section V, which identifies the flow by only looking at the network-layer header without the need for the transport-layer header information.
- Being a receiver-based protocol, RSVP is not compatible with client-server applications for which the server is the sender of the data frames and the client is the receiver. However, clients are usually the ones that initiate the connection with the server. This requires RSVP to be applied in the reverse direction, so that the server sends out the PATH message and the client replies with a RESV message.

VII. Differentiated Services

The IntServ service model, in spite of its flexibility and per-flow QoS provisioning, has not been successfully deployed in the public Internet. It may still be used in small-scale networks and within a customer network, but it is not suitable for the core of the Internet because of its inability to respond to traffic changes and lack of scalability. In order to address the problems of IntServ, the IETF decided to look for a more scalable alternative and developed the DiffServ architecture [37]. The IETF working group (WG) on DiffServ was formed to develop and standardize the basic building blocks of DiffServ and address deployment issues. The Internet2 Consortium [38] has adopted the DiffServ framework for providing QoS in the QBone network.

The main differences between the DiffServ and its predecessor, IntServ, are as follows [1]. First, the DiffServ is based on resource provisioning rather than resource signaling and reservation as in IntServ. This, in effect, makes it difficult to achieve precise or deterministic guarantees, and we will address this issue in Section IX. Second, the DiffServ handles traffic aggregates[4] instead of microflows. As a result, on a per-flow basis, DiffServ provides *qualitative* instead of *quantitative* QoS guarantees provided by IntServ. Third, the DiffServ standards define forwarding treatments, not e2e services like the guaranteed and the controlled load services in IntServ. Finally, the emphasis in DiffServ is placed on service-level agreements (SLAs) between domains rather than e2e dynamic signaling as in IntServ.

A. Background

The basic idea of providing differentiated services to packets can be traced back to RFC 2638 [39], in which the authors introduced the idea of using two bits in the IP header for marking packets in order to receive differential treatment in network devices. A similar idea based on one bit was presented earlier in August 1997 at the Munich meeting of IETF IntServ WG [40]. The basic approach is simple and scalable. The draft presented two service classes to be added to the prevailing best-effort model: a “premium” service, which has the characteristics of a “virtual wire,” and an “assured” service that follows expected capacity usage profiles.

The architecture presented in the draft uses two bits: a “P-bit” to mark packets for the premium service, and an “A-bit” to mark packets for the assured service. Packets are marked at the leaf routers where multifield (MF) classifiers are employed to differentiate flows based on these two bits. As shown in Fig. 9, packets marked with the premium service are metered against a specific profile using a token bucket meter with a very small bucket size, while the assured service packets are also metered using a token bucket meter with the bucket size equal to the traffic burst size. Out-of-profile packets are not marked and treated as best-effort. Together, these two bits determine the “priority” of handling the packet at the routers. Premium packets

[4]A traffic aggregate is a group of traffic flows handled in a similar way through a part or all of the network.

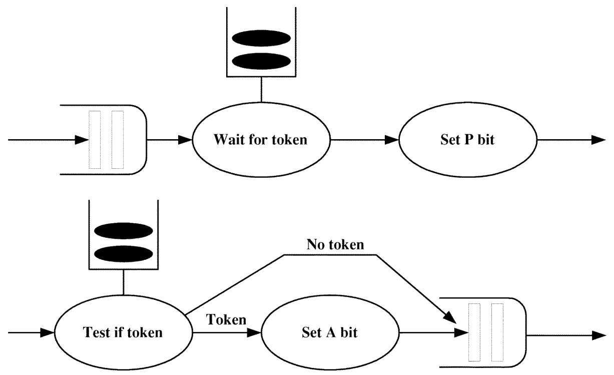

Fig. 9. Two-bit DiffServ marker (courtesy of [39]).

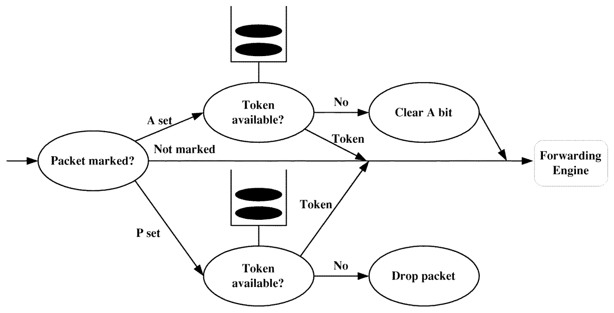

Fig. 10. Two-bit DiffServ edge router (courtesy of [39]).

get higher priority in forwarding than assured packets. On the other hand, assured packets are introduced to a buffer-management scheme such as RED in/out (RIO) to consider the probability of dropping packets between A-bit marked packets and best-effort (i.e., unmarked) packets. The functionality of the edge routers for two-bit marking is shown in Fig. 10.

The two-bit architecture proposed in [39] also includes a section on using bandwidth brokers (BBs) for service allocation and e2e service establishment. We describe BBs in Section VII-B7. These concepts form the basis for the more general DiffServ architecture as described next.

B. IETF DiffServ Architecture

Since its inception in 1998, the DiffServ WG has issued 15 RFCs describing the details of the architecture and its main building blocks. These RFCs, in addition to several other Internet drafts that did not find their way to become RFCs, cover most of the aspects of the architecture while leaving enough room for developers and network researchers to improve them. The following sections describe the main components of the DiffServ architecture and their operations to provide QoS on the Internet.

1) DS Field: The service differentiation requires marking of selected bits in IP packet headers. This is called the "DS field" in the DiffServ context. RFC 2474 [41] defines the DS field coinciding with the ToS byte in the IP header. However, only six bits are used to carry the DS codepoint (DSCP), and the remaining two bits are not currently used.[5]

DSCP is the codepoint used to determine the forwarding behavior that a packet experiences in a typical DiffServ node. This forwarding behavior is called *per-hop behavior* (PHB). DiffServ is based on defining a small number of PHBs implementing the necessary service differentiation at the participating routers and marking the DSCP bits to assign incoming packets to one of these PHBs. Currently, the DiffServ WG has finalized three types of PHBs in addition to the default best-effort service. One of those PHBs, the class selector (CS) PHB, was defined to ensure backward compatibility with the IP precedence bits used in the ToS byte. The other two PHBs are defined as expedited forwarding (EF) and assured forwarding (AF). These two PHBs can be used to implement premium and assured services, respectively, as mentioned in Section VII-A.

2) CS PHB: CS PHB was defined in RFC 2474 [41] to keep backward compatibility with the IP precedence bits in the IP ToS byte. It can be used to create eight different levels of priority with a larger value indicating a higher forwarding priority. CS-compliant PHBs can be realized by a variety of mechanisms, including strict priority queueing, WFQ [14], [18], CBQ [21], WRR [24], and their variants (RPS [42], HPFQA [43], DRR [17]).

3) EF PHB: The EF PHB is designed to implement a service with low delay, jitter, and loss in addition to an assured bandwidth, as specified in RFC 3246 [44]. The idea behind the EF PHB is to make packets marked with an EF DSCP encounter very small queues at the forwarding nodes. This is usually achieved by allocating forwarding resources with a higher rate than the arrival rate of EF packets. EF is used for services that have strict requirements on delay and jitter such as time-critical and multimedia applications. This type of service is usually referred to as *virtual leased line* (VLL).

RFC 3246 gives a formal definition of the EF PHB in terms of ideal and actual departure times of EF packets from an interface I in a DiffServ node configured with rate R

$$d_j \le f_j + E_a \quad \forall j > 0$$

where f_j is defined recursively with the initial condition $f_0 = 0, d_0 = 0$ as

$$f_j = \max(a_j, \min(d_{j-1}, f_{j-1})) + \frac{l_j}{R} \quad \forall j > 0$$

and d_j is the actual time of departure of the jth packet from the interface I, f_j is the "ideal" departure time of the jth packet from the same interface, a_j is the arrival time of the jth packet at the node, and l_j is the length of the jth packet in bits. E_a is an error term in the EF forwarding behavior of the node that may result from nonpreemptive operation of schedulers or any other factor. A similar definition is also given in the RFC to account for multiple input interfaces and

[5]Recently, these two bits are used for explicit congestion notification (ECN).

packet ordering, and it has a corresponding error term E_p instead of E_a. A delay bound for EF packets is given by

$$D = \frac{B}{R} + E_p$$

where the total offered load of EF traffic entering the node from all interfaces and destined for a single outgoing interface is bounded by a token bucket of rate $r \leq R$ and depth B.

The IETF documents suggest use of priority queues to implement the EF PHB as well as other scheduling schemes such as CBQ and WRR. The latter may not result in an efficient implementation due to the nature of RR scheduling. For performance results and validation studies of the EF PHB, we refer the readers to [45]–[49].

4) AF PHB: The AF PHB is used for building services with controlled loss and assured bandwidth. Such services do not have any delay or jitter guarantees. The IETF DiffServ WG defined a PHB group[6] for AF in [50]. The AF PHB group consists of three forwarding behaviors, AFx1, AFx2, and AFx3 in increasing order of drop precedence. This can also be interpreted as AFx1 being the most important and AFx3 being the least important. The notation "x" represents the AF class. The WG recommends the use of four independent AF classes with three drop precedences per class. Packet reordering between AF classes is not allowed. The AF PHB defines two rates: a committed information rate (CIR), which is the minimum bandwidth from the network to be assured, and a peak information rate (PIR) for a rate above the CIR to accommodate bursts.

The AF PHB is usually implemented in terms of buffer-management schemes such as RIO [51] and WRED [23]. It is worth mentioning that the effect of the AF PHB becomes most prominent in case of network congestion when some packets have to be discarded. We refer the readers to [52]–[58] for performance studies of the AF PHB and the effects of factors such as round trip time (RTT) and packet size on the fairness of flows in an aggregate with different traffic characteristics.

5) Per-Domain Behavior (PDB): PHBs are designed to be installed on individual nodes or routers. A group of packets that experience the same forwarding behavior at every node while crossing a domain is called a *behavior aggregate* (BA). The term BA later became synonymous with (PDB)—one of the basic building blocks of creating a DiffServ-enabled network. A PDB [59] is used to define a certain edge-to-edge service that has measurable network parameters as experienced by a set of packets with the same DSCP as they cross a DiffServ domain.[7] Therefore, PDBs are used to construct services between ingress and egress points of a domain. The attributes of PDBs (throughput, drop rate, delay bound, etc.) are advertised as service-level specifications (SLSs) at the edges of the domain. They are usually listed as statistical bounds or percentiles and not as fixed values.

PDBs can be constructed by concatenating a set of PHBs[8] between the edges of a domain. Traffic conditioning (marking/remarking, shaping and policing) on all incoming packets is done so that the PDB can meet the service level for which it was designed. An example of PDBs proposed by the WG so far is the virtual wire (VW) PDB [60] based on the EF PHB, and is suitable for delay-sensitive applications. Other PDBs such as assured rate (AR) [61] and bulk handling (BH) [62] PDBs have also been proposed.

Note that there is no one-to-one relationship between PHBs and PDBs. This means that more than one PDB can be based on the same PHB. On the other hand, the inverse is not currently supported, i.e., a PDB can only be based on one PHB within a single domain. This means that a large number of PDBs can be constructed from a small number of PHBs depending on many factors, such as PHB characteristics, available routes between each ingress–egress pair, and policy. An important consideration for the scalability of any PDB is that its attributes should be independent of the amount of traffic entering the domain or the path taken by this traffic inside the DS domain. Furthermore, the edge-to-edge attributes of a PDB should hold regardless of any splitting or merger of the traffic aggregates inside the domain.

6) DiffServ Framework: With the basic building blocks defined, we now describe how DiffServ can be deployed in practice. Fig. 11 illustrates a typical architecture of a DiffServ network.

A DiffServ-aware network consists of multiple DiffServ domains (DSs) that can be viewed as autonomous systems (ASs). The boundary routers of each domain perform the necessary traffic conditioning at the edges. Every DS domain makes two agreements with each of its neighboring domains: a SLA specifying the services (in terms of SLSs) that this domain will provide, and a traffic conditioning agreement (TCA) that incoming traffic to this domain will be subjected to. Adjacent domains negotiate SLAs among themselves and with customers accessing their network. Each DS domain configures and provisions its internal nodes such that these SLAs can be met. This distribution of configuration responsibilities adds to the flexibility of the DiffServ architecture. It is important to emphasize here that, unlike IntServ, DiffServ employs "resource provisioning" with the help of SLAs and does not use "resource reservation" in setting up different services across domains.

Let us follow the journey of a typical packet in a DiffServ network from the time it leaves the source until it reaches the destination. The packet is first metered against a certain traffic profile negotiated between the customer and the network service provider. The packet is then marked with an appropriate DSCP to meet a certain service level in the ISP network. Example packet marking schemes for the AF PHB can be found in [56], [57], [63]–[65]. At the time of writing this paper, there is no published work on marking schemes

[6]A PHB group is a set of one or more PHBs that can be meaningfully specified only if they are implemented simultaneously.

[7]By DS domain, we mean part of the network under a single administration and compliant with DiffServ standards.

[8]Usually from the same type or group.

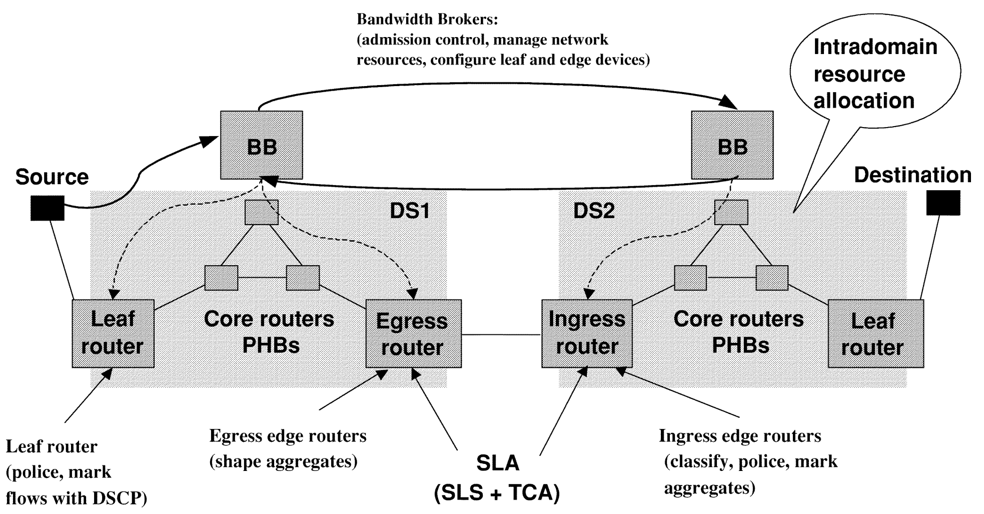
Bandwidth Brokers:
(admission control, manage network resources, configure leaf and edge devices)
Intradomain resource allocation
BB
BB
Source
Destination
DS1
DS2
Leaf router
Core routers PHBs
Egress router
Ingress router
Core routers PHBs
Leaf router
Leaf router (police, mark flows with DSCP)
Egress edge routers (shape aggregates)
SLA (SLS + TCA)
Ingress edge routers (classify, police, mark aggregates)

Fig. 11. DiffServ architecture.

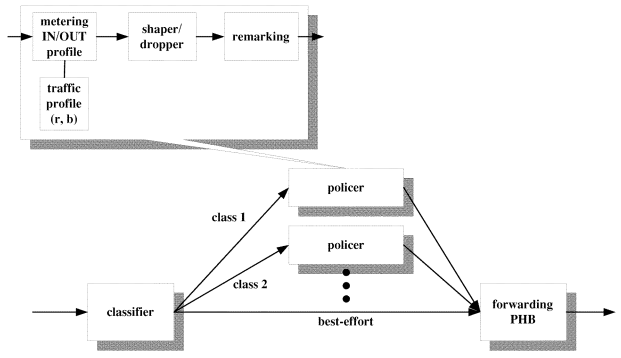

Fig. 12. DiffServ ingress router.

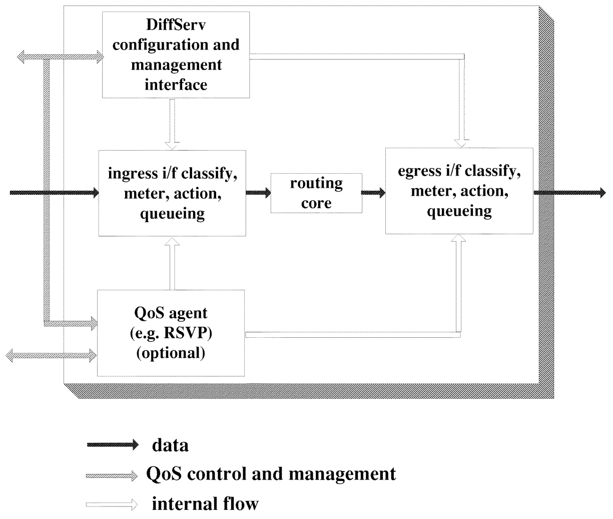
DiffServ configuration and management interface
ingress i/f classify, meter, action, queueing
routing core
egress i/f classify, meter, action, queueing
QoS agent (e.g. RSVP) (optional)
data
QoS control and management
internal flow

Fig. 13. DiffServ edge router main functional blocks.

for EF packets. A simple way of metering EF packets is to meter the traffic against a token bucket profile. Conformant packets are marked as EF and nonconformant packets are either dropped or unmarked. Packet marking can be accomplished either at the host (source) or at the first-hop router (called leaf router) or even at the boundary routers. In fact, a single packet can be remarked successively as it leaves the customer's premises and enters the ISP domain.

Once the packet has been marked, it becomes part of a specific behavior aggregate with all other packets marked with the same DSCP. At the *ingress* router of the DS domain, the packet is subjected to traffic conditioning as shown in Fig. 12. The packet is classified using either MF or BA classifiers. The packet is then metered against the negotiated traffic contract and undergoes a policer/shaper, if necessary. At this point, the packet may also be remarked with a different DSCP to indicate degradation in service level. Fig. 13, taken from [66], shows the main functional blocks of a DiffServ edge router.

The interior or core routers implement the necessary traffic forwarding treatments for different PHBs supported by the DS domain, as mentioned in Sections VII-B3 and VII-B4.

At the exit of a DS domain, the packet may go through another level of traffic conditioning in the *egress* router of the domain. This level of traffic conditioning guarantees that the traffic leaving one DS domain and entering the adjacent domain follows the traffic contract agreed upon between the domains.

From the above discussion, it is clear that the DiffServ architecture pushes the complexity of managing the network to the edges and leaves packet handling and forwarding in the core of the network as simple and fast as possible. This is a major improvement in scalability of DiffServ over other schemes such as IntServ. Although aggregated traffic handling reduces the flexibility of providing QoS guarantees to individual flows, it improves the overall scalability of the architecture.

Some examples of DiffServ have been given in [67]. It also discusses how to set up the main entities in a typical DiffServ network to achieve the required service levels. The IETF DiffServ WG has also worked on standardizing a management information base (MIB) [68] for DiffServ devices and policy information base (PIB) [69] for policy management in accordance with the informal model for DiffServ routers [66]. These models are necessary for setting up policies and managing large DS domains.

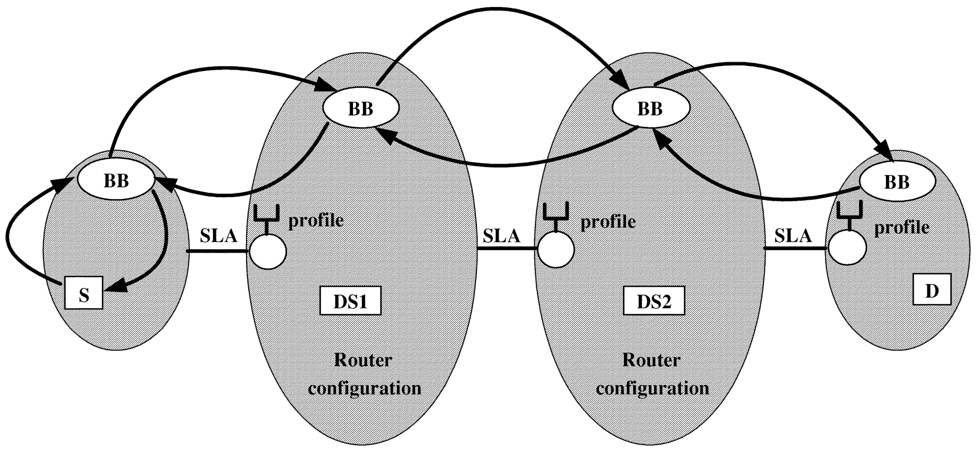

Fig. 14. Bandwidth broker operation for e2e QoS and resource allocation.

7) Bandwidth Broker and Policy Framework: It is clear that a management infrastructure is necessary in order to build e2e services spanning across multiple DS domains. The two-bit DiffServ architecture [39] proposed the use of BBs for this purpose. A BB is an agent that resides either in each DS domain or in-between domains. The BBs communicate with each other in order to establish e2e services and maintain the necessary state information instead of individual routers in the participating domains. If properly configured, an end-host in a customer network can contact its nearest BB agent with a service request. The BB agent, in turn, contacts the adjacent BBs along the e2e data path, negotiating the service required and the corresponding traffic profile. Once the requested service is confirmed by the participating BBs, they configure the classifiers and traffic conditioners at the edge routers so that the traffic from the end-host can be properly mapped to the appropriate service classes.

Fig. 14 shows an example of using BBs in a DiffServ network. We refer the readers to [70]–[72] for several proposed implementations of BBs.

Another proposed method to establish an e2e service is to use a RSVP as a signaling protocol, but with aggregation support [73], tunneling, or with BB as a reservation management entity. A new RSVP object called DCLASS [74] has been defined to carry DSCP on RSVP messages.

BBs are part of what is called "policy framework" being developed by the IETF [75]. The policy is used to regulate the access of network resources and services based on administrative criteria. The policy framework is usually responsible for admission control and resource provisioning through two main entities: a *policy decision point* (PDP) and a *policy enforcement point* (PEP). PDP stores all the policy rules using, for example, *lightweight directory access protocol* (LDAP), and distributes policy decisions to PEP using COPS [76] for their application on network traffic. For policy provisioning, COPS-PR [77] is defined to be used with PIBs as in the case for DiffServ. BB is considered as a PDP in the policy framework and can use RSVP with the POLICY_ DATA object for signaling and admission control with end-users.

C. The QBone Testbed

Recently, an experimental testbed called QBone [38] has been set up as part of Internet2 to study issues related to deployment and implementation of the DiffServ framework. QBone includes participants from universities, network vendors, organizations such as vBNS, Abilene, ESNet, CA*Net2, SURFnet, NREN, and many others. QBone currently offers a service called QBone premium service (QPS), which is based on the EF PHB. Many experiments have been conducted using this service, including multimedia streaming and video-conferencing applications. Based on several studies, the QBone WG has concluded that there are some difficulties in deploying QPS on a wide scale, and further studies will be necessary. A new type of service, called QBone scavenger service, is now being tested on QBone that requires no policing, reservations, and admission control. This type of service provides "nonelevated" forms of QoS [38].

D. Evaluation of the DiffServ Architecture

The DiffServ architecture has the following advantages over IntServ.

1) **Scalability:** The DiffServ architecture addresses the scalability problem of IntServ by providing different service levels to traffic aggregates instead of individual flows and removing the need for per-flow states in each router in the e2e path. This makes it more suitable for the Internet.
2) **No setup time:** There is no need for signaling in DiffServ, and usually services are constructed of SLAs and traffic forwarding and conditioning through network nodes and domains.
3) **On-the-fly admission control:** This is done through traffic policing and remarking at border routers, so there is no need to have an admission control decision in every node in the e2e path.
4) **Compatibility with IPsec:** Since packet handling in DiffServ requires only the IP header, it can be used in conjunction with IPsec unlike IntServ. However, there are some architectural issues about handling the

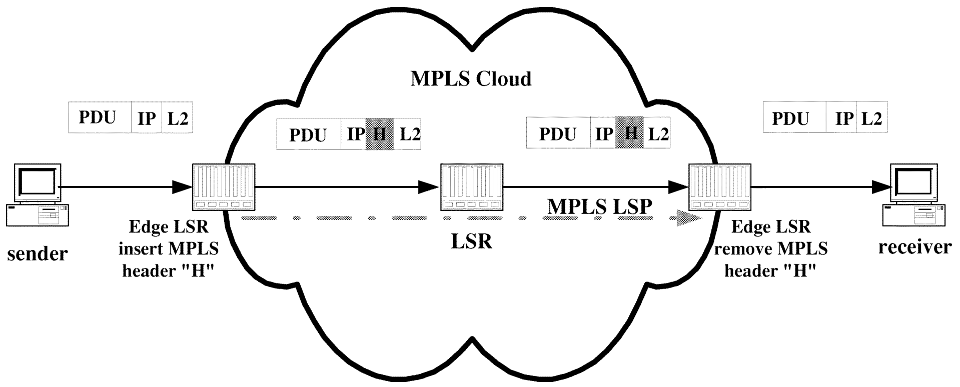

Fig. 15. MPLS network.

"inner" and the "outer" DS fields in IPsec tunnels that can be found in [37] and [41].

Despite these advantages, a number of operational issues need to be resolved before the DiffServ architecture can be deployed in practice.

- The DiffServ standard as proposed does not provide e2e QoS assurances to Internet traffic. It only specifies how individual domains can be configured to provide service differentiation to different traffic classes. In Section IX, we discuss the operational and research issues for supporting e2e QoS on the DiffServ architecture.
- The DiffServ architecture does not have a specific scheme for accurate admission control. Rather, it uses traffic policing and shaping to provide on-the-fly admission control at the edge and boundary routers.
- Mapping and interoperation with other schemes (such as MPLS and ATM) are necessary for DiffServ to be deployed across many domains. Although there are proposals for mapping DiffServ's service classes to IntServ [78], [79] and ATM [80]–[83], more research is needed in order to measure the effectiveness of these proposals.
- In order for applications to take advantage of DiffServ capabilities, there is a need for developing appropriate application programming interfaces (APIs) on end-hosts.

VIII. MPLS and Traffic Engineering

MPLS [84] is an advanced forwarding scheme that works between layer 2 (link layer) and layer 3 (network layer). MPLS is similar to DiffServ, yet more general like ATM and Frame Relay. It is based on tagging each packet with a specific header that determines its path and forwarding at network nodes. MPLS routers, called *label switching routers* (LSRs), use these tags (labels) along with the forwarding tables, to map the packet to a specific path called *label-switched path* (LSP). LSPs run between two LSRs—an ingress LSR and an egress LSR—both located at the edges of the MPLS network. A group of packets that meet the same forwarding behavior over the same path is called *forwarding equivalent class* (FEC). The term "traffic trunk" [85] is defined as an aggregation of flows with the same service class, or FEC, that can be put into a LSP. This is different from the LSP. Configuring LSRs to map specific labels to specific LSPs is done using a label distribution protocol (LDP). Fig. 15 illustrates the basics of MPLS operation.

The ability of MPLS to support "explicit routing" makes it a good candidate to be used in *traffic engineering* [86], [87]. Traffic engineering has recently become an important tool to be used by network service providers for resource control and performance optimization. RSVP extension to support traffic engineering (RSVP-TE) [88], and controlled routing label distribution protocol (CR-LDP) [89] are examples of resource management over MPLS.

DiffServ and MPLS

There have been several proposals for supporting DiffServ over MPLS [90]. MPLS along with constraint-based routing (CBR)[9] can perform DiffServ with scalability and flexibility. The authors of [91] proposed a service architecture based on MPLS and DiffServ. DSCPs can be mapped to different labels in MPLS headers, and LSPs are used to build PDBs inside DS domains. A recent IETF document, RFC 3270 [90] describes a flexible solution for support of DiffServ over MPLS. We refer the readers to RFC 3270 for more details.

IX. Application-Level QoS Using DiffServ

In this section, we discuss application-specific QoS management within the DiffServ framework. The goal is to provide an e2e path between end-systems that can apply specific QoS metrics to meet the performance requirements of individual applications. However, there is a high overhead of state maintenance and management of individual flows. Therefore, the tasks associated with application-specific per-flow management, such as mapping of application requirements to the underlying network parameters, QoS signaling, monitoring and feedback, are typically applied at the edge or access routers only.

The DiffServ architecture requires that data traffic to be classified so that appropriate QoS mechanisms can be deployed to provide necessary service differentiation. This task is complicated by the fact that within a specific class of

[9]A method for traffic engineering.

service and even within a particular class of applications, the requirements vary widely. For example, the requirements of multimedia applications may vary with the encoding/decoding algorithm and the real-time protocol used, packet or frame sizes, and single versus separate channels for each component of a multimedia stream. This raises important questions related to practical implementation of the DiffServ framework, e.g., whether different application flows can be aggregated at the edge to provide a single aggregate across DS domains, or separate virtual PDBs for each class of applications requiring a certain level of service (e.g., separate PDBs for voice, video, and data) will be needed.

The current QoS frameworks such as Tequila [92], MASQ [93], and QuO [94] require that mapping and profiling functions for an application are precomputed by either an end-user/application developer or the service provider. This requires knowledge of network QoS parameters such as bandwidth, delay, jitter, packet loss, and especially, how the e2e performance of the application may be quantitatively affected by these parameters. Moreover, an application may be composed of several components, e.g., teleoperation of a remote microscope in real-time involves transmission of images, instrument control commands, and data. Each component may require a particular class of service. The situation is more complex when the overall performance measure of the application may not be a simple sum of the performance of its components, because the performance metrics of some of the components are correlated.

The mapping of application requirements into appropriate network resource parameters can be accomplished in three steps. In the first step, the application-level QoS parameters are mapped to a set of network-level QoS parameters such as e2e delay, jitter, packet loss, and bandwidth. In the second step, these network-level parameters are mapped to one of the available DiffServ service classes in each of the participating domains. In the final step, appropriate PHBs (i.e., schedulers, policers, and markers) are either chosen or configured inside each domain in order to meet the requirements of the chosen service class. Of these, the mapping from the application-level to the network-level QoS parameters can be performed either at the host itself or at a first-hop gateway. An example of such mapping for telerobotics and distributed digital teleoperation is provided in [95]. The streams from various services are classified into robotics sensory data, still image, slow-motion video, and telephone audio. In [96], the authors evaluate e2e performance of different types of multimedia traffic such as video-conferencing, encoding schemes, and mixtures of video and data traffic over 10/100Base-T Ethernet. In the DiffServ context, more research is needed before all three steps of mapping are well understood so that they can be deployed on a wide scale and on a routine basis.

X. Conclusion

In this paper, we have presented and analyzed a number of proposed standards as well as currently available schemes for providing service-level guarantees to Internet applications. In order to understand application QoS requirements, we have categorized various real-time and multimedia applications according to their QoS requirements and run-time behavior. We also gave a quick overview of QoS parameters and QoS building blocks that are usually used to realize QoS support. We briefly described the ATM network as the first scheme that explicitly supports QoS and, despite its complexity, is still widely deployed. However, newer and cheaper networks are typically based on the IP standard instead of ATM. The IETF has proposed two service models—IntServ and DiffServ—for supporting QoS in an IP-based network. We have described both service models in detail and compared the pros and cons of each. We primarily focused on the DiffServ standard because it is more scalable and suitable for the Internet. It is also being increasingly supported by the network equipment vendors. We mentioned the use of DiffServ and MPLS for QoS support over multiple domains and pointed to the use of traffic engineering in resource and performance optimization in QoS networks. In spite of its advantages, the current DiffServ architecture does not provide e2e QoS support for end-user applications. Although the IETF WG on DiffServ is close to completing the standard, there remain several unresolved issues such as traffic aggregation, admission control, and application-level QoS mapping, which have to be resolved before DiffServ can be deployed in practice.

References

[1] Z. Wang, *Internet QoS: Architecture and Mechanisms for Quality of Service*. San Mateo, CA: Morgan Kaufmann, 2001.

[2] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*. Reading, MA: Addison-Wesley, 2001.

[3] D. Miras. (2002, July) A survey on network QoS needs of advanced Internet applications. Working Document—Internet2 Fellowship on Application QoS Needs. [Online]. Available: http://www.internet2.edu/qos/wg/apps/fellowship/Docs/Internet2AppsQoSNeeds.pdf

[4] D. Collins, *Carrier Grade Voice Over IP*: McGraw Hill, 2001.

[5] D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," presented at the Proc. ACM SIGCOMM'92, Aug. 1992.

[6] "Society of Motion Picture and Television Engineers: Bit-serial digital interface for high-definition television systems," SMPTE-292M, 1998.

[7] H. Knoche and H. de Meer, "QoS parameters: A comparative study," Univ. Hamburg, Hamburg, Germany, Tech. Rep., 1997.

[8] H. Schulzrinne *et al.*, "RTP: A transport protocol for real-time applications," IETF, RFC 1889, Jan. 1996.

[9] S. Jha and M. Hassan, *Engineering Internet QoS*. Norwood, MA: Artech House, 2002.

[10] S. Jamin, P. Danzig, S. Shenker, and L. Zhang, "A measurement-based admission control algorithm for integrated services packet networks," presented at the Proc. ACM SIGCOMM'95, Feb. 1995.

[11] D. Verma, *Supporting Service Level Agreement on IP Networks*. New York: Macmillan, 1999.

[12] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. IEEE*, vol. 83, pp. 1374–1396, Oct. 1995.

[13] A. Parekh and R. Gallagher, "A generalized processor sharing approach to flow control in integrated services networks—The multiple node case," *IEEE/ACM Trans. Networking*, vol. 2, pp. 137–150, Apr. 1994.

[14] S. Keshav, *An Engineering Approach to Computer Networking*. Reading, MA: Addison-Wesley, 1997.

[15] E. Hahne and R. Gallagher, "Round robin scheduling for fair flow control in data communications networks," in *Proc. IEEE Int. Conf. Communications (ICC)*, June 1986.

[16] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1265–1279, Oct. 1991.

[17] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," presented at the Proc. ACM SIGCOMM'95, 1995.

[18] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," presented at the Proc. ACM SIGCOMM'89, Sept. 1989.

[19] J. Rexford, A. Greenberg, and F. Bonomi, "Hardware-efficient fair queueing architectures for high-speed networks," presented at the Proc. ACM SIGCOMM'96, 1996.

[20] P. Goyal, H. Vin, and H. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Trans. Networking*, vol. 5, pp. 690–704, Oct. 1997.

[21] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, vol. 3, pp. 365–386, Aug. 1995.

[22] ——, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, Aug. 1993.

[23] D. Lee, *Enhanced IP Services for Cisco Networks*. Indianapolis, IN: Cisco Press, 1999.

[24] R. Guerin, "Quality of service in IP networks, tutorial," presented at the Proc. 6th IEEE Real-Time Technology and Applications Symp., Washington, DC, May 2000.

[25] N. Giroux and S. Ganti, *Quality of Service in ATM Networks: State-of-the-Art Traffic Management*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[26] ATM forum [Online]. Available: http://www.atmforum.com/

[27] ATM service categories: The benefits to the user [Online]. Available: http://www.atmforum.com/aboutatm/6.html

[28] J. Postel, "Internet protocol," IETF, RFC 791, Sept. 1981.

[29] ——, "Service mappings," IETF, RFC 795, Sept. 1981.

[30] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification," IETF, RFC 2460, Dec. 1998.

[31] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," IETF, RFC 1633, June 1994.

[32] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP)—Version 1 functional specification," IETF, RFC 2205, Sept. 1997.

[33] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource reservation protocol," *IEEE Network*, vol. 7, Sept. 1993.

[34] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," IETF, RFC 2212, Sept. 1997.

[35] J. Wroclawski, "Specification of the controlled-load network element service," IETF, RFC 2211, Sept. 1997.

[36] ——, "The use of RSVP with IETF integrated services," IETF, RFC 2210, Sept. 1997.

[37] S. Blake, D. Black, M. Carlson, E. Davis, Z. Wang, and W. Weiss, "An architecture for differentiated services," IETF, RFC 2475, Dec. 1998.

[38] Internet2 QBone [Online]. Available: http://qbone.internet2.edu/

[39] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," IETF, RFC 2638, July 1999.

[40] D. Clark and J. Wroclawski, "An approach to service allocation in the Internet," IETF, Internet-draft (work in progress) draft-clark-diff-svc-alloc-00.txt, July 1997.

[41] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers," IETF, RFC 2474, Dec. 1998.

[42] D. Stiliadis and A. Varma, "Rate-proportional servers: A design methodology for fair queueing algorithms," *IEEE/ACM Trans. Networking*, vol. 6, pp. 164–174, Apr. 1998.

[43] J. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," presented at the Proc. ACM SIGCOMM'96, Aug. 1996.

[44] B. Davie, A. Charny, J. Bennet, K. Benson, J. Bouded, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An expedited forwarding PHB," IETF, RFC 3246, Mar. 2002.

[45] T. Ferrari and P. Chimento, "A measurement-based analysis of expedited forwarding PHB mechanisms," in *Proc. IWQoS 2000, IEEE 00EXL00*, Pittsburgh, PA, June 2000, pp. 127–137.

[46] W. Ashmawi, R. Guerin, S. Wolf, and M. Pinson, "On the impact of policing and rate guarantees in diff-serv networks: A video streaming application perspective," in *Proc. ACM SIGCOMM'01*, Aug. 2001, pp. 83–95.

[47] J. K. M. A. Tyagi and H. de Meer, "VoIP support on differentiated services using expedited forwarding," *Proc. 19th IEEE Internet Performance, Computing and Communications Conf.*, pp. 574–580, Feb. 2000.

[48] M. Albrecht, M. Koster, P. Martini, and M. Frank, "End-to-end QoS management for delay-sensitive scalable multimedia streams over diffserv," presented at the Proc. 25th Annu. Conf. Local Computer Networks, Tampa, FL, Nov. 2000.

[49] R. Guerin and V. Pla, "Aggregation and conformance in differentiated service networks: A case study," presented at the Proc. ITC Specialist Seminar on IP Traffic Modeling, Measurement, and Management, Sept. 2000.

[50] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," IETF, RFC 2597, June 1999.

[51] D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," *IEEE/ACM Trans. Networking*, vol. 6, pp. 362–373, Aug. 1998.

[52] N. Seddigh, B. Nandy, and P. Pieda, "Bandwidth assurance issues for TCP flows in a differentiated services network," presented at the Proc. IEEE Globecom'99, Mar. 1999.

[53] ——, "Study of TCP and UDP interaction for the AF PHB," IETF, Internet-draft (work in progress) draft-nsbnpp-diffserv-tcpudpaf-01.txt, Aug. 1999.

[54] M. Goyal, A. Durresi, R. Jain, and C. Liu, "Performance analysis of assured forwarding," IETF, Internet-draft (work in progress) draft-goyal-diffserv-afstdy-00.txt, Feb. 2000.

[55] K. N. J. Ibanez, "Preliminary simulation evaluation of an assured service," IETF, Internet-draft (work in progress) draft-ibanez-diffserv-assured-eval-00.txt, Aug. 1998.

[56] M. El-Gendy and K. Shin, "Equation-based packet marking for assured forwarding services," presented at the Proc. IEEE INFOCOM'02, New York, June 2002.

[57] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Adaptive packet marking for maintaining end-to-end throughput in a differentiated-services Internet," *IEEE/ACM Trans. Networking*, vol. 7, pp. 685–697, Oct. 1999.

[58] H. Wang and K. Shin, "Layer-4 service differentiation and resource isolation," presented at the Proc. IEEE RTAS'02, San Jose, CA, Sept. 2002.

[59] K. Nichols and B. Carpenter, "Definition of differentiated services per-domain behaviors and rules for their specifications," IETF, RFC 3086, Apr. 2001.

[60] V. Jacobson, K. Nichols, and K. Poduri, "The "virtual wire" per-domain behavior," IETF, Internet-draft (work in progress) draft-ietf-diffserv-pdb-vw-00.txt, July 2000.

[61] N. Seddigh, B. Nandy, and J. Heinanen, "An assured rate per-domain behavior for differentiated services," IETF, Internet-draft (work in progress) draft-ietf-diffserv-pdb-ar-01.txt, July 2001.

[62] B. Carpenter and K. Nichols, "A bulk handling per-domain behavior for differentiated services," IETF, Internet-draft (work in progress) draft-ietf-diffserv-pdb-bh-02.txt, Jan. 2001.

[63] J. Heinanen and R. Guerin, "A single rate three color marker," IETF, RFC 2697, Sept. 1999.

[64] ——, "A two rate three color marker," IETF, RFC 2698, Sept. 1999.

[65] W. Fang, N. Seddigh, and B. Nandy, "A time sliding window three color marker (TSWTCM)," IETF, Internet-draft (work in progress) draft-fang-diffserv-tc-tswtcm-01.txt, Mar. 2000.

[66] Y. Bernet, S. Blake, D. Grossman, and A. Smith, "An informal management model for diffserv routers," IETF, RFC 3290, May 2002.

[67] Y. Bernet *et al.*, "A framework for differentiated services," IETF, Internet-draft (work in progress) draft-ietf-diffserv-framework-02.txt, Feb. 1999.

[68] F. Baker, K. Chan, and A. Smith, "Management information base for the differentiated services architecture," IETF, RFC 3289, May 2002.

[69] M. Fine, K. McCloghrie, J. Seligson, K. Chan, S. Hahn, C. Bell, A. Smith, and F. Reichmeyer, "Differentiated services quality of service policy information base," IETF, Internet-draft (work in progress) draft-ietf-diffserv-pib-09.txt, June 2002.

[70] Qbone bandwidth broker, QBone [Online]. Available: http://qbone.internet2.edu/bb/

[71] University of Kansas bandwidth broker implementation, UKANS [Online]. Available: http://www.ittc.ku.edu/ kdrao/BB/

[72] I. Khalil and T. Braun, "Implementation of a bandwidth broker for dynamic end-to-end resource reservation in outsourced virtual private networks," in *Proc. 25th Annu. IEEE LCN*, Nov. 2000, pp. 9–10.

[73] F. Baker, C. Iturralde, F. L. Faucheur, and B. Davie, "Aggregation of RSVP for IPv4 and IPv6 reservations," IETF, RFC 3175, Sept. 2001.
[74] Y. Bernet, "Format of the RSVP DCLASS object," IETF, RFC 2996, Nov. 2000.
[75] Resource allocation protocol working group [Online]. Available: http://www.ietf.org/html.charters/rap-charter.html
[76] D. Durham *et al.*, "The COPS (common open policy service)," IETF, RFC 2748, Jan. 2000.
[77] K. Chan *et al.*, "COPS usage for policy provisioning (COPS-PR)," IETF, RFC 3084, Mar. 2001.
[78] Y. Bernet *et al.*, "A framework for integrated services operation over diffserv networks," IETF, RFC 2998, Nov. 2000.
[79] J. Wroclawski and A. Charny, "Integrated service mappings for differentiated services networks," IETF, Internet-draft (work in progress) draft-ietf-issll-ds-map-01.txt, Feb. 2001.
[80] S. Manjanatha and R. Bartos, "Integrating differentiated services with ATM," presented at the Proc. 1st Int. Conf. Colmar, France, July 2001.
[81] M. Loukola and J. Skytta, "Differentiated services over ATM," presented at the Proc. RTAS'99 Workshop, 1999.
[82] T. Do *et al.*, "ELISA: European linkage between Internet integrated and differentiated services over ATM," presented at the Proc. RTAS'99 Workshop, 1999.
[83] F. Borgonvo, A. Capone, L. Fratta, and C. Petrioli, "VBR bandwidth-guaranteed services over diffserv networks," presented at the Proc. RTAS'99 Workshop, 1999.
[84] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," IETF, RFC 3031, Jan. 2001.
[85] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS," IETF, RFC 2702, Sept. 1999.
[86] J. Boyle *et al.*, "Applicability statement for traffic engineering with MPLS," IETF, RFC 3346, Aug. 2002.
[87] D. Awduche *et al.*, "Requirements for traffic engineering over MPLS," IETF, RFC 2702, Jan. 2001.
[88] ——, "RSVP-TE: Extensions to RSVP for LSP tunnels," IETF, RFC 3209, Dec. 2001.
[89] J. Ash *et al.*, "Applicability statement for CR-LDP," IETF, RFC 3213, Jan. 2002.
[90] F. L. Faucheur *et al.*, "Multi-protocol label switching (MPLS) support of differentiated services," IETF, RFC 3270, May 2002.
[91] X. Xiao and L. M. Ni, "Internet QoS: A big picture," *IEEE Network*, Mar. 1999.
[92] P. Flegkas, P. Trimintzios, G. Pavlou, I. Andrikopoulos, and C. Cavalcanti, "Policy-based extensible hierarchical network management," presented at the Proc. Workshop Policies for Distributed Systems and Networks (Policy 2001), Bristol, U.K., Jan. 2001.
[93] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli, "An active middleware to control QoS level of multimedia services," presented at the Proc. 8th IEEE Workshop Future Trends of Distributed Computing Systems, 2001.
[94] C. Rodrigues, J. P. Loyall, and R. E. Schantz, "Quality objects (QuO): Adaptive management and control middleware for end-to-end QoS," presented at the Proc. OMG Workshop Real-Time and Embedded Distributed Object Computing, July 2000.
[95] K. Nahrstedt and J. M. Smith, "An application-driven approach to networked multimedia systems," presented at the Proc. 18th IEEE Annu. Conf. Local Area Computer Networks, 1993.
[96] W. Ashmawi, R. Guerin, S. Wolf, and M. Pinson, "On the impact of policing and rate guarantees in diffserv networks: A video streaming application perspective," presented at the Proc. ACM SIGCOMM'01, 2001.

Mohamed A. El-Gendy (Member, IEEE) received the B.Sc. degree in electrical engineering and the M.Sc. in computer engineering from Cairo University, Cairo, Egypt, in 1995 and 1998, respectively. He is pursuing the Ph.D. degree from the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor.

He has been a Research Assistant in the Department of Electrical Engineering and Computer Science, The University of Michigan, since August 1999. He is a member of the Real Time Computing Laboratory, working on realizing QoS for real-time applications over the Internet and specifically through the DiffServ framework. His research interests are computer networks, network QoS, real-time systems, and distributed systems.

Abhijit Bose (Member, IEEE) is pursuing the Ph.D. degree in computer science and engineering at The University of Michigan, Ann Arbor.

He is a Research Scientist with the Center for Advanced Computing at The University of Michigan, Ann Arbor. His research interests are in application-level QoS, statistical characterization of networks, and grid computing. He is on the technical leadership team of MGRID, an institutional Grid comprising several centers and colleges in Michigan, currently under development. He is also one of the collaborators in the DZero/SAM Grid project of the Fermi National Laboratory, which is developing algorithms and software for connecting thousands of CPUs around the world for distributed processing of high-energy physics data.

Kang G. Shin (Fellow, IEEE) is the Kevin and Nancy O'Connor Professor of Computer Science and Founding Director of the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. His current research focuses on QoS-sensitive networking and computing as well as on embedded real-time OS, middleware and applications, all with emphasis on timeliness and dependability. He has supervised the completion of 42 Ph.D. theses, and authored/coauthored over 500 technical papers and numerous book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. He coauthored (jointly with C. M. Krishna) the textbook *Real-Time Systems* (New York: McGraw Hill, 1997).

Dr. Shin received the Outstanding IEEE Transactions on Automatic Control Paper Award in 1987, the Research Excellence Award in 1989, the Outstanding Achievement Award in 1999, the Service Excellence Award in 2000, and the Distinguished Faculty Achievement Award in 2001 from The University of Michigan. He coauthored papers with his students that received the Best Student Paper Awards from the 1996 IEEE Real-Time Technology and Application Symposium, and the 2000 UNSENIX Technical Conference. In 2002, he also received a Distinguished Alumni Award from the College of Engineering, Seoul National University, Seoul, Korea.