RZ 3395 (# 93445) 02/11/2002
Computer Science 24 pages

Research Report

Differentiated Services in the Internet

Brian Carpenter*, Kathleen Nichols‡

*IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland
brian@hursley.ibm.com

‡Packet Design
2465 Latham Street
Third Floor
Mountain View, CA 94040
USA
nichols@packetdesign.com

Differentiated Services in the Internet

Brian Carpenter* and Kathleen Nichols‡

**IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland*
‡ Packet Design, 2465 Latham Street, Third Floor, Mountain View, CA 94040, USA

Abstract **-** The phrase “Quality of Service” (QoS) has been used frequently, with a variety of different meanings, in recent discussion of the Internet. In this paper we use it in a relatively narrow sense, to describe a set of measureable parameters, such as delay, throughput, and loss rate that can be attached to some identifiable subset of the traffic of Internet Protocol (IP) packets through a given network domain. We expand on this point and emphasize that giving any sort of guarantee about the values of such parameters to a user requires the implementation and deployment of physical mechanisms throughout the network. Configuring these mechanisms in such a way that their effect, when viewed from the edges of the network, composes into the desired QoS for the user can be a complex matter.

A “user” of IP quality of service spans a range of granularities, from a single application program to an entire company. After reviewing the background to this problem, this paper motivates one particular approach to a solution, known as Differentiated Services. It gives a technical overview of this approach, discusses resource allocation and configuration questions, and gives some application examples, before concluding with a look at the future.

1.0 Background: toward an Internet-friendly quality of service

The global communications network that predates the Internet is the telephone system and thus, it is often in the minds of those who seek to architect applications and new components for the Internet, including quality of service. Yet, the telephone system is quite different from the Internet. The telephone system, at least in its original purely electrical form, had a relatively simple congestion problem and an equally simple quality of service requirement. The congestion constraint was that a reserved electrical path must be available between the two terminals; this is a binary state. The quality requirement was that this path must have adequate bandwidth and gain, and low enough noise, to transmit an intelligible voice signal. Although the telephone network has evolved considerably, it remains true that a reserved path and a single grade of quality are the two defining characteristics of a standard telephone call.

The situation has never been this simple in packet switching networks including the Internet. There is no requirement in the Internet for a reserved path from source to destination – in principle, every packet follows its own path as it travels, and shares every link it travels with packets from many other sessions. In fact, paths are determined by routing tables at each hop, and may change due to extraneous circumstances, though in most cases they are constant throughout a session. Individual packets are routed separately and the session as a whole cannot assume an unchanging path. At the same time, packets are of variable length (unlike the case of ATM cells), typically between 20 and 4000 bytes. The arrival of packets at network links is both random and bursty, with considerable experimental evidence that the time distribution of packet arrival follows a self-similar law where the underlying distributions are heavy-tailed, rather than a Poisson distribution [FAIL, SELF].

The practical implication of this is that even when the average rate of traffic is small, arbitrarily long bursts of traffic at full hardware speed are to be expected. Periods of congestion are therefore commonplace in the Internet, resulting in both packet loss and jitter. Furthermore, a given link or router in the core of the Internet may be carrying traffic for thousands or millions of sessions from a wide variety of application types whose network signature and intrinsic QoS requirements vary enormously. The mixture and distribution of application types has evolved, and will likely continue to evolve, over the Internet's history [FAIL, CAIDA]. Unlike the telephone network, measurements show the dominant traffic pattern is typified by short sessions with a handful of packets in each direction. Thus an Internet flow, in general, cannot be characterized as a "call " where substantial setup costs are acceptable due to the length of the call. On the contrary, setup costs need to be minimized or eliminated. In summary, the Internet has no standard QoS requirement and no analytically tractable (e.g., Poisson) traffic model.

Today's Internet comprises a complex interconnection of network domains, some under the same administrative entities, some under different administrative entities. Currently, most of these network domains have more capacity available at their interiors and more limited capacity at their borders, but this is neither universally true nor can this be expected to persist indefinitely. The history of the Internet has encompassed many changes in traffic characteristics and applications, in link bandwidths and utilizations, in the number of connected network domains and their degree of connection, and in the administrative complexity of the the Internet. The Internet has survived this history of change by staying true to the original principles that kept it scalable [Clark88, Saltzer].

These "facts of life" for the Internet necessitate very different approaches to its architecture, including implementing a quality of service, than the lessons of classic telephony would dictate. Thus, approaches rooted in the intuitions and science of telephony networks do not, in general, apply to the Internet.

1.1 Previous Approaches: missing the mark

Since the problem is as old as the concept of packet switching, it is not surprising that there have been earlier attempts to solve it. Four approaches will be mentioned briefly here: 1) IP precedence and type of service, 2) SNA class of service, 3) Internet Stream Protocol (ST), and 4) Integrated Services.

IP precedence and TOS: In the original design of IP [RFC 791] a byte known as the "type of service octet" was reserved in the header of every packet. This was defined to contain two important fields: a three bit "precedence" value, and three "type of service" (TOS) bits. The precedence was intended as a simple priority marker, where priority 0 got the worst treatment and priority 7 got the best. The type of service bits were used for a different approach, with different bits identified as "low delay", "high throughput" and "high reliability".

Unfortunately there was no architectural framework in which to employ the type of service octet, and thus no explanation of how these properties could be implemented across a network. Later work [RFC 1349] expanded these definitions, but still without an architectural framework. In practice, although some software exists that sets these bits, the TOS bits have been of little use and generally ignored. IP precedence has had very limited use. Deployments have been confined to small numbers of domains in implementations that have been ad hoc, experimental, and often anecdotal. A well-known practice is the use of high precedence in some network domains for routing updates, but it is not clear how widely deployed this capability is.

SNA: IBM's Systems Network Architecture (SNA) has long distinguished several classes of service, e.g., to give interactive transactions priority over batch operations [SNA]. These classes of service have been enforced using scheduling on the links by the network control points (NCPs). However, this simple approach works only in the context of a tightly managed network with carefully assigned resources for a very small number of traffic types. More sophistication is needed to handle the raging torrent of diverse traffic on the Internet.

ST: The Internet Stream Protocol, known as ST [RFC 1819], was an attempt to accommodate real time traffic streams in parallel with TCP/IP traffic, by adding a second connection-oriented network protocol in parallel with IP. This has failed to grow outside a limited experimental community, since the intended applications have had very limited use in the Internet, and the implementation is stateful and not well-tuned to the Internet's administrative hierarchy.

Integrated Services (IntServ): This is a major effort within the Internet Engineering Task Force to specify a mechanism for supporting end-to-end sessions across the Internet that require a specific quality of service (such as a given peak capacity and transmission delay). The IntServ model [RFC 1633] requires a module in every IP router along the path that reserves resources for each session, and then ensures that each data packet in transit is checked to see what resources it is entitled to receive. The reservations are requested using a specially designed resource reservation protocol known as RSVP. An important notion in IntServ is admission control, i.e., a process that refuses to admit traffic to the network when sufficient resources are not available. If the RSVP request fails, the session will not start (or will do so in a degraded mode). IntServ also has the notion of traffic conformance at the input to the network; packets will be spaced out in time in such a way as to correspond to the resources reserved by RSVP.

IntServ has attractions, but two obvious disadvantages. One is that it requires major new software or firmware in both the forwarding and control of all routers along the network path concerned. The other is that if it were to be used on major ISP trunk connections, carrying millions of packets per second, the overhead per packet of implementing the necessary checks and resource management is widely believed to be unacceptable. IntServ's fundamental unit is the "flow", similar to the "call" of telephony networks, though, as noted above, the typical "flow" of the Internet looks nothing like a call and is not long enough to justify substantial setup costs. Second, in IntServ, Autonomous System or provider boundaries are essentially invisible. IntServ expects reservation state to cross administrative boundaries without pain or challenge. This is an unrealistic business model and offers the potential for denial of service. In fact, this disregard for the administrative boundaries of networks flaws all four approaches listed above. Third, the forwarding path packet treatments required by IntServ have been highly influenced by theoretical results, rather than what is implementable efficiently at high speeds. For these reasons, it is expected that IntServ and RSVP will initially be limited to campus and small corporate networks [RFC 2208].

IntServ has failed to be adopted widely because its assumption of setting state in all routers along a path is nonscalable and nonworkable administratively. Further, the superposition of large numbers of streams with varying burst characteristics make it very difficult to have a sensible admission control. The failure of Integrated Services is largely due to its taking an approach that left behind the roots of success of IP and adopted the notion that "QoS means connections". The connection-oriented model cannot be used to bring a viable end-to-end QoS to the Internet because it assumes a "flat" model of the Internet; that is, one that is administratively homogeneous. The number of connections required to handle all the traffic of the Internet leads to a state explosion in the core routers.

1.2 A fresh approach

More recently, the IETF has taken another approach, Differentiated Services (Diffserv), currently being finalized in the IETF and implemented by various vendors. Diffserv is based on an architectural framework that recognizes the relevant entity for effecting service guarantees in the Internet is the administrative domain of a single network operator (either an enterprise network, or a single ISP). Thus the model is oriented towards edge-to-edge service across a single domain, with an appropriate service level agreement assumed to be in place at the edges of the domain. In this way, simple but effective QoS can be built from the components during early deployments and Internet-wide QoS can evolve into a more sophisticated structure as rollouts and experience increases. Internet-wide, differentiated service levels will, of necessity, require agreements between adjacent network providers. The technical part of these agreements will consist of the edge-to-edge service level to which a network provider is committed.

Also, the emphasis has been on developing QoS building blocks first rather than the services, recognizing the need for highly scaleable mechanisms with minimum impact on the data path elements of core routers that handle multi-gigabit links. The emphasis is on minimalist mechanisms that are useful and implementable. The Diffserv architecture also recognizes the highly variable traffic profiles encountered in the Internet. Unlike IP Precedence and SNA Class of Service, it aims at "sophisticated simplicity" in which the data path mechanisms are simple to implement, but allow very rich network behaviors to be created. Unlike ST and Integrated Services, it avoids the creation of state information along the path of each individual traffic flow. Diffserv takes the approach that connections that are visible at the network's edge do not necessarily mean connections that are visible in the interior of the network.

In the next section, we explore the motivations for the Diffserv approach with a high level discussion of the general considerations and constraints for quality of service in IP networks. Section 3 describes the Differentiated Services approach technically and covers Differentiated Services in the IETF (Internet Engineering Task Force) to date. Section 4 covers the current working item of Diffserv in the IETF, Per-Domain Behaviors, illustrating its importance in creating and deploying Internet-friendly QoS. Section 5 gives considerations for allocation and considerations for edge-to-edge services to emerge. Section 6 gives some examples using Diffserv. Section 7 is our view of where we go from here.

2.0 Motivating Diffserv's approach

In this section, we examine six questions important to service differentiation in IP networks.

2.1 Why do we need quality of service in IP networks?

IP networks deliver packets with a type of service known as "best effort" with definition "as much as possible as soon as possible." There is no quantification inherent in its definition and each packet has the same expectation of treatment as it transits a network. The objective of network quality of service (QoS) is to quantify the treatment a particular packet can expect as it transits a network. The object of differentiated QoS is to give some packets better treatment and some packets worse treatment; QoS can't create additional bandwidth. Thus a workable QoS architecture should provide a means for specifying performance objectives for different types of packets as well as a means of delivering on those performance objectives.

Service Providers generally publish a Service Level Agreement for their services that includes the measurable quality levels that a customer can expect for the single class (or "best effort") traffic they currently handle. For example [UUNET] lists, for U.S. enterprise service, a round-trip US latency of under 65 milliseconds and a North American packet delivery of at least 99%. Such guarantees are common in today's Internet.

About a decade ago, the advent of voice and video packet traffic motivated initial attempts to bring different levels of service to packet traffic. The premise of these approaches was and is that such traffic is "more important" by some measure and should be treated accordingly. More recently, economic reasons to differentiate packet traffic have emerged; the Internet has become much more widely used and has become mission-critical to many companies. For example, to meet network service objectives, some companies have resorted to a private infrastructure. When viable methods for differentiating traffic in IP networks exist (and can be coupled with security concerns), ISPs can offer services with specific performance targets and may offer an array of services with cost linked to quality. Further, with a useful IP QoS available in the general Internet, more communications can migrate to the Internet, increasing the provider's revenue while decreasing network costs for the purchasing company (ISP customer). The key to a multiservice network is to be able to differentiate traffic both by forwarding treatment and by the rules applied to the traffic.

Enterprise networks have both similarities to and differences from the ISPs. Even where an enterprise has sufficient bandwidth that network administrators feel comfortable with one level of service for all data traffic, packet telephony works best with a different service level, one that requires relatively little bandwidth, but more controlled delay variations and is robust to congestive periods of any length. Individual corporate networks sometimes want to designate different types of traffic for different types of treatments, but the specific policies desired cover a wide range.

Expected major initial applications are 1) to distinguish traffic importance (e.g., "mission-critical" or preferred customer) within a network cloud, 2) to enable good quality voice over IP networks, and 3) to permit service providers to sell services that compete with leased line (circuit replacement).

2.2 What applications should get improved QoS?

The answer fundamentally impacts architecture. How this question is answered is fundamental to a QoS architecture. At present, there are two major ways of looking at QoS. The most obvious one is as a service that an end-user requests, either directly or indirectly, quantifiable at the end-user's machine. In this case, it's usually possible for a user to determine whether the QoS objective is being met by simple measurement. For example, a user initiates a voice-over-IP call and expects the call to be intelligible. From a human point of view, call quality is subjective but objective measures of packet rate, delay, jitter, and loss are required for an intelligible call and must be supplied by the network. Enabling a telephony connection or a video session between two endpoints is the network QoS problem of interest for many people and there have been many attempts to entwine a particular type of QoS with a particular application, particularly voice and video. This unnecessarily limits the usefulness and extensibility of QoS and does not fit the successful model the Internet has followed thus far. To see this, note that a fixed bandwidth, limited delay, low loss QoS isn't just for voice or video. Consider a case where a quantity of data, as from an experiment or a medical record, must be transferred in a certain period of time. Here QoS that delivers fixed bandwidth between two endpoints would work nicely, as long as there is a buffer at the rate mismatch point. Early attempts at this type of QoS simply policed the data traffic to the constant fixed bandwidth rate with disastrous results for the Internet's typical "elastic" traffic [Wittbrodt, RFC2638]. QoS levels should not be intimately bound with applications through architectural design, though the policy of a particular network may express some such entwinement.

The second way of looking at QoS is from the network administrator's point of view. In this case, there are administrative objectives for different types of traffic that may not be quantifiably apparent to an end-user, but can be stated in objective criteria that are observable by the network administrator. For example, "in the presence of congestion, at least 1 Mbps will be available to data traffic from the Customer Service department". Or "regardless of network load, VoIP traffic up to 200 kps will get priority through all network nodes." Despite its usual association with "better service", QoS can be used by network administrators to limit certain kinds of traffic in a network. This view of QoS provides means for network administrators to set network traffic objectives based on more complex and administratively important traffic differentiators than application type. Although compliance to those set targets may not be easily measurable with today's network tools, it is easy to imagine more tools becoming available as the means to do the traffic engineering become available.

Individual applications' intrinsic traffic characteristics and requirements vary enormously. Some sessions may be long-lived telnet logins with very little traffic, but needing rapid interactive response. Some may be FTP sessions with bursts of kilobytes or megabytes at the speed of a disk drive. Some may be audio or video streams, with fixed bit rates and no way to slow down or retransmit. Some may be HTTP transactions, which typically go to the expense of opening a transport session for the sake of transmitting a handful of packets between a browser and server. Currently, the dominant traffic pattern is typified by the latter case: short sessions with a handful of packets in each direction. Thus an Internet flow, in general, cannot be characterized as a "call" where substantial setup costs are acceptable due to the length of the call [CAIDA, Wilder]. On the contrary, setup costs need to be minimized or eliminated; telephony style signalling would be useless for the majority of "sessions" in the Internet.

QoS approaches have been proposed that attempt to leverage the congestion controls of the Transmission Control Protocol (TCP) currently used by nearly all Internet traffic. Unfortunately, not all applications can reasonably make use of TCP with its elastic response to congestion. Further, TCP's congestion controls can be inappropriate for short transactions (despite which it is used for even the shortest HTTP GET command) and for remote procedure calls or simple command/response usage. And it is virtually useless for real time applications such as IP telephony or video streaming. Thus, command/response applications and streaming tend to be built on top of UDP, RTP, or recently SCTP.

In practice, real-time protocols built over UDP work quite badly when the network is congested: packets are lost, words in a phone call are lost, videos freeze, and so on. At exactly the same time, the same user may observe email being successfully transferred and Web pages loading, if slowly, due to TCP's elastic response. It is not necessary for the network's long term utilization to be high for this to occur; the traffic burstiness mentioned above can lead to congestive incidents even when average traffic is modest. Thus we have a mix of traffic with differentiated characteristics, yet every router in the network is treating all packets in the same way.

This situation may lead Internet Service Providers (ISPs) to identify a requirement to treat traffic from (or to) different subscribers according to a specified policy. In the absence of other contractual agreements, all subscribers might expect to receive a "fair share" of the available resources, but this is not guaranteed by standard TCP behavior. A network QoS-based solution might be to enforce some kind of "equal access" under congestion. In another scenario, Company *A* does not want its overnight file transfers to be held up by those for Company *B*. Alternatively, Company *X* may be prepared to pay more for Web hosting than Company *Y*, but certainly expects its web pages to be served up faster as a result. Another possibility is that an ISP wants to offer two grades of service to its domestic subscribers, with two different price levels and two different typical response times or throughput targets.

Some of these requirements can be partially met by reserving bandwidth for individual subscribers, perhaps in the form of a virtual private network. This would allow all of Company *A*'s intra-company traffic to be isolated from other traffic. But this isolation does nothing in itself to guarantee QoS, does nothing to separate Company *A*'s voice traffic from its file transfer traffic, and does nothing to assist company *A*'s response time for Web hits from the public Internet. A method of differentiating traffic at any and every point in the Internet seems to be called for. Needless to say, it must scale up to the future multi-billion node Internet, be globally deployable, and be manageable by network operators.

An illustrative example. To make this more concrete, we consider a specific simplified example. If all traffic was telephony, a network designer would know that each call required (say) 64 Kbit/s of capacity, and that no more than *N* calls could pass through a given path on the network, where *N=(path capacity in Kbps)/64*.

Each call sends packets adding up to 64 Kbit/s in real time.[1] In such a network, it would be easy to compute the necessary capacity, and to compute the maximum delay any voice packet could experience. "Maximum delay" in this case can be a high probability upper bound rather than an absolute maximum, as long as the degree of confidence matches the voice packet loss that is tolerable.

If all traffic were overnight file transfer, the situation would be different. The total number of transfers, the disk speeds at each end-point, and the size of the files to be transferred would be unknown. No individual transfer would be especially urgent, but lost packets would have to be retransmitted to avoid file corruption.

In this case the file transfers will run on top of TCP, which has two important characteristics: when packets are lost due to congestion, they will be retransmitted in due course; and when the network is congested,

[1] There is of course an optimisation available of not transmitting silent packets, but this does not change the argument.

dropped packets signal each instance of TCP to slow down accordingly, to avoid taking more than a fair share of the total capacity.

Now consider what happens when these two types of applications (telephony and file transfer) are mixed together in the same part of the network. Regardless of congestion and loss, the telephony traffic will preserve its transmission rate at 64*N* Kbit/s. The file transfer traffic will experience congestion and packet loss, and will slow down in an attempt to share the available capacity fairly. Since the telephony traffic cannot slow down, it will obtain an unfair share of the capacity, but will still experience high loss and thus poor voice quality, in part due to jitter or buffer delay variability.

As indicated, this is a simplified example. Not all TCP traffic is bulk file transfer, and not all real time traffic is telephony with a standard bit rate. The real traffic mix is much more complex, with a wide variety of QOS characteristics and requirements. Nevertheless, even the simple example shows that if unconditioned TCP-like traffic (i.e., traffic that slows down in the face of congestion) is mixed in with real time traffic (that keeps going despite congestion), both sides lose. This is a problem that requires solution if the Internet is to become fully viable for real time services.

Summary: the wrong question? In summary, thinking of QoS in a single way or for a single application is the wrong approach. Although voice and video can benefit from and take advantage from QoS, both applications work in the best-effort Internet and whether QoS is required should be the user's decision based on perceived cost-benefit. For example, a university student may be willing to use a standard Internet quality for a voice-over-IP call, taking a quality hit in order to incur no cost. The university president, however, will want a high-quality for voice-over-IP call to major contributors and is likely to have the budget to pay for it. Furthermore, though file transfers are thought of as not needing QoS, it again depends on the objectives of the users. The types of applications and the uses of current applications in the Internet have changed over time. IP QoS should provide a general means to solve problems, not unlike IP itself solving the general problem of moving packets. Thus no assumptions should be made about what type of traffic will require "better" or "worse" treatment, instead the focus is on building a framework where it is possible to deliver different treatment to different types of traffic, where the type of traffic can be determined in a flexible manner.

2.3 Who decides which packets get a specific QoS?

As stated above, QoS in a network provides a means for giving some packets better treatment than others; packet treatment should not tie particular applications to particular levels of QoS. This leads us to the question of "who decides?" which packets get a specific type of QoS. One possibility is to let end-users mark their own packets to decide what gets the best QoS. Although this has an advantage in that one presumes a user knows what is "most important", it is clearly infeasible since the user's desires might not reflect the desires of the organization that pays for the network, and the end-user doesn't have a way of knowing if the current network use pattern can handle additional packets of that QoS level or not. Further, an end-user may not know what kind of QoS is appropriate for, say, a voice call vs. a file transfer. At the very least a method of requesting service and granting or denying it must be in place. Some agent should be in the network to implement the granted service. A variety of approaches may be used to handle the requests for service and then to configure the network agent.

QoS must be allocated to reflect the policies and priorities of the organization that is footing the bill for the network resources that are allocated. Each new request for QoS must be evaluated in light of both policy and current allocation so the the expected QoS levels are achieved. QoS must reflect the local policy. It must be possible to reflect the hierarchy of organizational policies in the way that QoS gets allocated.

The Internet and its component networks are made up of independently administered entities or domains, often referred to as “clouds” and drawn as such in diagrams. Clouds may be different domains or regions within a domain that require different treatment. Thus the structure of IP networks reflects an organizational hierarchy directly related to who has control over the resources of the equipment contained within each cloud. This is discussed further in several later sections.

Network resources, like any other resource of an organization, from office space to paper clips, is allocated according to policy, on a long-term basis, and availability, which may be on a shorter term basis. Although the discussions that go into decisions about resource allocation may be complex, there are usually some simple rules about matching people with space in an organization. These rules are usually recorded somewhere. When a decision is required, as when a new person is hired or a department moves, both the rules and the current availability must be consulted before individuals are assigned to specific spaces. The person doing the space assignment may have nothing to do with the policy process for who gets what kind of space and may not even work for the same administrative suborganization as either the decision-makers or the persons being assigned to spaces, but can simply follow a set of written rules and check for current availability to come up with a decision on a particular request and an assignment of person to space.

The next question is how to provide end-to-end cross-domain QoS with this constraint. It’s not reasonable to expect that one domain can allocate another domains’s resources. How do we make this work? We will look at the Internet and other settlement models and extrapolate.

2.4 What primitive mechanisms does the network require?

Since QoS means packets can be designated for different treatment, the network infrastructure must be capable of distinguishing between packets through means of classification, enqueueing packets separately as a result of the classification, scheduling packet queues to implement the differential treatment, as well as provide means for measuring, monitoring, and conditioning packet streams to meet requirements of different QoS levels. These can all be realized through the implementation of mechanisms in the packet forwarding path.

There are many packet flows that require different QoS treatment, but the number of ways in which a packet can be treated in the forwarding path is limited. Aggregating individual flows according to their common packet forwarding treatment leads to a reduction of state and complexity. The cost for this is that the rules of aggregation must be explicit and must lead to sensible behaviors for both aggregates and the individual flows that compose them. This problem is greatly simplified if the type of QoS offered to individual flows aggregates easily.

The need for QoS to work on high-speed forwarding paths necessitates that all mechanisms in the forwarding path be amenable to high-speed implementation. Further the mechanisms of the forwarding path must be easily composable without causing unintended problems. This argues in favor of both flexibility and simplicity.

2.5 How can end-to-end services be built in an “edge-to-edge” Internet?

We’ve seen that access to QoS must be allocated at the local level, following the lines of administrative control. Since not all connections begin and terminate in the same administrative domain, how should we build end-to-end QoS services? One answer has been a connection- or call- oriented approach. Here a particular path is set up between the endpoints of a data conversation and the resources are reserved *along this path*. If resources are not available along the entire path, the connection is refused. This is the circuit-switched telephony approach and there are those who believe it should be applied to IP networks. We argue that it is the *wrong* approach: it doesn't fit the Internet which is not a "flat" (homogeneous) network in any sense. In particular, it is not administratively homogeneous. Connections tie up resources, require state for every connection, have signaling overhead for every connection and are not incrementally deployable or scalable. This violates every principle that IP networking was built on and which has led to its success. IP QoS must fit the Internet and must scale in the Internet.

If not connections, then what? An Internet-friendly approach takes into account that networks are composed of administratively and technologically diverse clouds. While resource allocation decisions are made within each cloud, what matters outside the cloud is only the behavior across borders with immediate neighbors. Packet traffic is classified into traffic aggregates based on local rules and the traffic aggregate characteristics are checked at the network boundaries. Where we cross a boundary, some rules are needed about what we accept from outside and what we can send outside. Network boundaries get "programmed" with these rules governing what enters or leaves (e.g., what gets sent, what gets received, what gets refused). If each cloud has classified all its QoS requests into a small number of differentially forwarded aggregates, packet traffic crossing the boundaries need only be classified, monitored, and measured on this small number of aggregates, thus only a small amount of state is necessary. This approach moves almost all the work to cloud boundaries and only keeps state pertaining to the flow of packets between any two clouds (bilateral agreements), thus it is much more scalable and amenable to local policy control and to keeping local policy opaque to rest of the outside world.

Complexity can be further reduced and earlier and incremental deployment facilitated by noting that aggregation can eliminate the need for signalling; the intradomain and bilateral nature means that having a uniform signaling protocol is not necessary for a wide range of useful quality of service. The task of allocation is complex and still "under construction", though simple roll-out is possible. The local control and bilateral agreement structure allows us to be agnostic about signaling, which will be critical in rapid and incremental deployment.

The model of building services from the bilateral agreements of "touching" networks reflects the structure of the Internet today, where end-to-end connectivity is built from the bilateral agreements between the owners of the clouds that packets traverse to reach destinations outside their originating cloud. Where clouds are part of a single organization's network, the boundary agreements are expected to be quite simple to implement. Where clouds belong to different organizations, we expect that settlement models will be extended to handle QoS.

Section 3 covers the reasons for a network and aggregate-oriented rather than a connection-oriented model in more depth.

2.6 How is QoS to be quantified?

QoS must give a measurable value-add to the person who “pays” or to a “trusted proxy” of the person who pays; paying can take many forms. A user who has paid for a guaranteed 64 Kbps service can measure packets to determine if this has been delivered. An organization with particular bandwidth targets for particular classes of traffic can monitor the network to ensure that the targets are being met, however, tools for this purpose are still fairly primitive. An organization that contracts with a network service provider for specified levels of QoS must be able to monitor what QoS is actually received or must sufficiently trust the service provider to permit and believe its monitoring results. An Internet-friendly QoS should not exclude any of the possible viable models for who measures QoS and how its delivery is specified. Instead, flexibility is required in addressing the range of needs.

3.0 Diffserv: fitting an architectural framework to the Internet

3.1 Differentiated Services history and a conceptual overview

Two approaches to the application of simple packet marking to signal differentiated router behavior in the core of a network domain were first described by Clark and Jacobson, initially in the IRTF’s End to End Research Group, and later more publicly in [RFC2638] and [Clark95]. This led to a “birds of a feather” session in the April 1997 meeting of the Internet Engineering Task Force (IETF), where major users indicated a clear requirement for such a simple form of service differentiation [FDDIFS]. An IETF working group, naturally named “Diffserv”, was subsequently formed to develop standards for this approach; the authors are the co-chairs of this working group.

A common theme of this work was recognition that IP networks are composed of *clouds*, regions of relative homogeneity in terms of administrative control, technology, bandwidth. The Internet and indeed all non-trivial networks are made up of these clouds and the boundaries between them. By determining where clouds and boundaries lie, we determine where to administer resources and where control is applied to make sure that policy is enforced. Within a cloud it is possible to take advantage of the uniformity within its boundaries by aggregating individual traffic flows (microflows) into a limited number of *traffic aggregates* that each get a distinct forwarding treatment. Within a cloud, all that is important about a packet to determine its forwarding treatment is what aggregate it belongs to, and it is possible to trust a marking put in the packet to indicate its aggregate. Within a cloud, resources are allocated according to a local set of rules. The way a specific policy decision about QoS is implemented is by classifying, monitoring, marking, policing, and other conditioning of packet traffic at the boundaries of the clouds, after which packets receive uniform treatment (according to their traffic aggregate) within the cloud. Almost all the work is confined to the borders of clouds. The rules for allocating resources should not be visible outside of a cloud, only the behavior aggregates.

Recognizing the realities of the Internet, Diffserv utilizes the traffic aggregate as its fundamental unit of traffic, rather than a flow. A 6 bit field in each packet's header identifies its traffic aggregate in the center of the network and thus the forwarding treatment each packet in the aggregate will receive, regardless of which individual microflow it belongs to. This promotes maximal aggregation in the center of the network, where a small number of different forwarding treatments (Per-Hop Behaviors) are applied to all traffic to get the required service quality. At the edge of the network, more state might be kept about packets by matching more packet fields; thus the aggregates of the interior might be made up of packets from several customers, policed and conditioned differently at the edge, but with the same expectation of forwarding treatment once past the edge.

For example, all IP Telephony traffic crossing a single ISP might be treated as a single aggregate; or all the traffic from Customer A. It is assumed that traffic is sorted into aggregates as it enters a given Diffserv domain. Some aggregates may also be subject to admission control; others may not. Then, the routers within the domain will be configured to treat each traffic aggregate differently: thus, in a domain that recognizes N aggregates, the routers will each be capable of N different forwarding behaviors (i.e., combinations of link scheduling and queue management), one appropriate for each aggregate. For example, at individual network nodes, IP Telephony traffic might be scheduled with high priority, but only up to a total bandwidth of 2 Mbit/s; admission control would be used to ensure that interior limit by policing entry to the aggregate at the network's edge.

The Diffserv architectural framework concentrates complexity and state (such as classification, admission control, and SLA enforcement) at the edge of the Diffserv domain, where scaling is less of an issue, and only requires the core of the network to implement simple, highly scalable scheduling behaviors. The basics of Diffserv are available in Proposed Standard [RFC2474] and Informational [RFC2475] documents and have been defined for both IPv4 and IPv6.

3.2 Separation of control and forwarding

The Diffserv model is based on the separation of forwarding path and control plane and on the notion that a small number of forwarding path primitives can be composited to create a wide range of QoS features. In IP forwarding, connectivity is achieved by the interaction of two components; the packet forwarding part and the routing part. From RFC 2474: "Packet forwarding is the relatively simple task that needs to be performed on a per-packet basis as quickly as possible. Forwarding uses the packet header to find an entry in a routing table that determines the packet's output interface. Routing sets the entries in that table and may need to reflect a range of transit and other policies as well as to keep track of route failures. Routing tables are maintained as a background process to the forwarding task. Further, routing is the more complex task and it has continued to evolve over the past 20 years." Similarly, Diffserv made explicit that IP QoS can be separated into the differentiated treatment given to packets in the forwarding path and the task of configuring the parameters of the forwarding path components to allocate QoS according to policy and availablility. This approach makes it possible to rollout simple QoS via static configurations, just as the early Internet provided connectivity using static routes. Figure 1 illustrates this separation in a router.

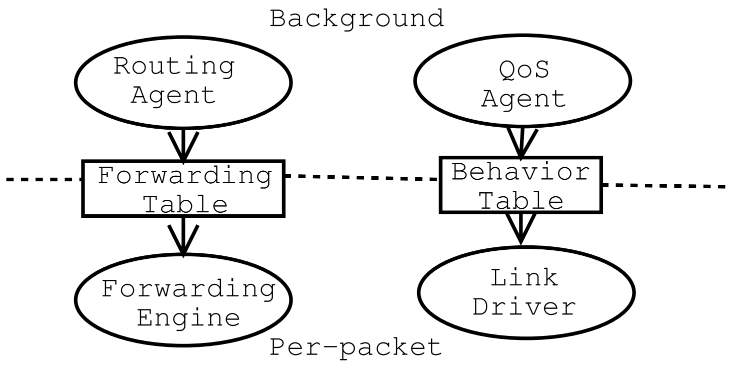

Figure 1: Separation of forwarding path and background control in IP routers

A *service* is a description of the overall treatment of (a subset of) a customer's traffic across a particular administrative domain, across a set of interconnected domains, or end-to-end. Within a domain, service descriptions are covered by administrative policy which is applied to the construction and concatenation of traffic aggregates. Traffic aggregates are described by a particular set of rules, represented by a configuration of classifiers, markers, policers, and shapers, in concert with a specific forwarding treatment configured in a particular way. Each TA so described has a set of quantifiable characteristics across a network cloud which might be used in the creation of a customer-visible service. Multiple services can be supported by a single forwarding treatment used in concert with a range of traffic conditioners and multiple packet "marks" might map to the same forwarding treatment.

Configuration of the rules to deliver the particular characteristics is a control plane function.

3.2 Defining the Forwarding Path Primitives

Basic Requirements. A small number of primitives can be composed to provide all the forwarding path functionality necessary. *Classification* takes apart (input) packet stream. *Policing* enforces that the behavior conforms to the rules governing the packet substream (by dropping, delaying, or remarking). *Marking* propagates information about the aggregate downstream. *Per-hop forwarding behaviors* are generally implemented by packet queues, their management, and the scheduling discipline that moves packets from the queue(s) to the output link. Queuing isolates one traffic stream from another. The configurable queue scheduling discipline constructs an (output) packet stream based on local policy and downstream agreements. Since QoS cannot be delivered without these forwarding path primitives being present in network equipment and their implementation in high-speed routers must follow the development cycle of custom integrated circuits, the Diffserv working group began by defining these, including the designation of a packet's Diffserv “mark”.

Diffserv’s Mark. Every IP packet carries a byte called the Type of Service octet. In all but a few percent of today’s traffic, this byte is set to zero; clearly it is an under-utilized feature of IP. In the new 128-bit version 6 of IP, there is an equivalent byte called the Traffic Class octet. The first task of the Diffserv working group was to respecify this byte, defined identically for the old IPv4 and the new IPv6. This 6-bit field is known as the Differentiated Services field (or **DS field**) and is *marked* with a specific bit-pattern called a DS codepoint (or DSCP) used to indicate how each router should treat the packet. Diffserv packets must have a suitable value in the DSCP field. To emphasize the fact that no session information needs to be stored, this treatment is known as a Per-Hop Behaviour (or **PHB**). To be specific, the octet now looks like this:

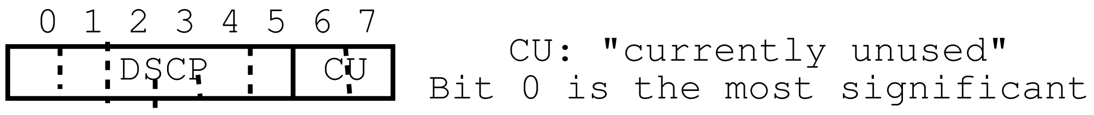

Figure 2: Layout of TOS Octet

The two CU bits have subsequently been allocated for explicit congestion notification.

The six-bit DS field can contain up to 64 different binary values. Most experts do not believe that 64 different PHBs will ever be needed, but the extra codepoints leave room for innovation and local operational optimizations; thus some of the bit patterns are reserved for local or experimental use. Marking may occur in two places:

- the original source of the traffic, e.g. a web server or IP telephony gateway, marks the traffic. This has the advantage that the classifier may have explicit knowledge of the application in use, and can therefore mark packets in an application-dependent way.

- a router, such as the first router the traffic encounters, or the router at the customer/ISP boundary, classifies and marks the traffic. This has the advantage that no change is needed to servers, but it requires some extra “smarts” in the router. Fortunately many routers have a very similar capability already, for use with IntServ/RSVP.

A third option, combining the advantages and disadvantages of both, is for the server to use the IntServ/RSVP model to communicate with a boundary router, but for the boundary router to use the Diffserv model for wide area communication across the open Internet.

Using the Mark. When a packet enters a router, routing logic selects its output port and the DSCP value is used to steer the packet to a specific queue or treatment at that port. The particular handling depends on the definition of the corresponding PHB. The particular PHB is configured by a network management mechanism setting up the QoS behavior table inside the router.

The Diffserv working group has taken the approach that a few fundamental PHBs should be standardized early. These should derive from some existing experience (primarily from limited deployment and experimentation with use of the IP precedence field to select forwarding behaviors) and might be implemented using a variety of specific mechanisms.The PHBs standardized so far are as follows:

- Default behavior. Here the DSCP value is zero and the service to be expected is exactly today’s default Internet service (i.e. with congestion and loss completely uncontrolled).

- Class selector behaviors. Here seven DSCP values run from 001000 to 111000 and are specified to select up to seven behaviors each of which has a higher probability of timely forwarding than its predecessor. Experts will note that the default behavior plus the class selectors exactly mirror the original eight IP Precedence values.

- Expedited Forwarding (or EF) behavior. The recommended DSCP value is 101110 and the behavior is defined as being such that the departure rate of EF traffic must equal or exceed a configurable rate. EF is intended to allow the creation of real-time services with a configured throughput rate.

- Assured Forwarding (or AF) behaviors. An AF behavior actually consists of three sub-behaviors; for convenience let us call them AF_1, AF_2, and AF_3. When the network is congested, packets marked with the DSCP for AF_1 have the lowest probability of being discarded by any router, and packets marked for AF_3 have the highest such probability. Thus, within the AF class differential drop probabilities are available, but otherwise the class behaves as a single PHB. The standard actually defines four independent AF classes. Quite complex service offerings can be constructed using AF behaviors, and much remains to be understood about them.

The first official IETF Diffserv document [RFC2474] defines the Default behavior and the Class Selectors. Expedited Forwarding is defined in [RFC2598, under revision]. Assured Forwarding is defined in [RFC2597].

Related IETF Work. Other fundamental documents produced by the working group include an informal model of a Diffserv-enabled router [DSMODEL], a Management Information Base for such a router [DSMIB], and a Policy Information Base for such a router [DSPIB]. A method of identifying PHBs in inter-domain protocols where DSCPs are meaningless has been defined [RFC2836, under revision]. The complex considerations for Diffserv traffic inside encrypted or unencrypted tunnels have been documented [RFC2983].

Diffserv has also stimulated work in other IETF working groups or by individual IETF participants. A mechanism for using Integrated Services at the edge of a network, and Diffserv in the core, has been defined [DSINT]. Mappings from Diffserv to the QOS support in MultiProtocol Label Switching have been described [DSMPLS]. Specific packet marking mechanisms for Assured Forwarding have been described [RFC2697, RFC2698, RFC2859]. Work in the Policy Model, Resource Allocation Protocol, and SNMP Configuration working groups is strongly oriented towards Diffserv management, i.e., the control plane. The reader is referred to [IETF] for further details.

Most recently, the Diffserv working group itself has tackled the problem of Per Domain Behaviors (PDBs) which we describe in section 4.

Composing and allocating resources for services are matters not currently on the standards track and would, at least in part, appear to provide areas for competitive differentiation, either through the tools provided by vendors or the expertise in service deployment of specific service providers or consultant organizations. We present some simple examples of composing service across clouds in section 6.

3.3 Control Plane Options

3.3.1 Cloud-based QoS Control

A premise of Diffserv is that it is possible to control resources on a cloud basis rather than per single node or single path. The Diffserv architecture approach to QoS is to “put it where you need it”, perhaps only deploying the QoS forwarding mechanisms at a single bottleneck of a cloud, the limited resource whose access should be shared according to some policy. In practical terms, this means there may be only one router in a cloud that uses the Diffserv “mark” and in that case the cloud’s packets are marked by how they are to be treated at that bottleneck. The criteria for marking (or policing pre-marked) packets can be statically configured or signalled by any means available and expedient. Diffserv is agnostic about signaling and does not build it into the architecture. Control decisions that can be more easily expressed on a per-cloud basis can be deployed earlier and save a lot of unnecessary work, compared with attempting a total system design based on connection-oriented reserved paths. If the cloud-oriented model proves inadequate, the size (number of nodes and links) encompassed by a cloud can be successively shrunken in future deployments, even “shrinking” a cloud to a single network node or path if some deployment requires it. The Diffserv architecture takes the approach that it is better to end up at this more complex point only if it is necessary rather than to attempt to start there

Diffserv's control plane focuses on the behavior of traffic aggregates by ensuring their characteristics on an edge-to-edge basis. The means of ensuring characteristics for a traffic aggregate across a cloud may use a variety of approaches. Connection-oriented approaches can exist within a cloud: ATM, MPLS, RSVP/Intserv, but a fully connection-oriented approach to QoS is not reasonable for future end-to-end services. The Internet requires a framework where the component clouds are in control of their resources and the method used to allocate them. The “edge-to-edge” characteristics of traffic aggregates across a cloud must be quantifiable in fairly simple, measurable terms. These quantifiable characteristics make it possible to build up bilateral agreements and have performance expectations for different traffic aggregates (each associated with a PDB, described in Section 4).

3.3.2 Connecting Network Clouds

This architecture allows end-to-end services to be constructed out of purely bilateral agreements. Each cloud only needs to establish relationships of limited trust with adjacent clouds, unlike schemes that require the setting of flow specifications in routers throughout an end-to-end path. In practical technical terms, this makes it possible to keep state on an administrative domain basis, rather than at every router and makes it possible to confine per flow state to just the boundary routers. Allocation must follow organizational hierarchies; that is, each organization must have complete responsibility for allocation of traffic resources within its domain. Thus, the allocation architecture is made up of agreements across boundaries as to the amount of each traffic aggregate that will be allowed to pass. This is similar to "settlement" models used today. Note that the construction of these agreements is an administrative matter; they must then be translated into router configuration parameters by a QoS policy management system. These actions take place quite distinct from the data path (unlike the case of IntServ configuration using RSVP, which takes place along the data path itself).

Cloud connections will follow an incentive structure based on traffic being policed to a specific set of rules as it enters a domain, with the consequences of violating the rules being such that it is in the upstream or sending cloud's best interests to ensure conformance. These rules must be simple to specify (in a contract, for instance) and to measure. We see these as being of the form "Customer *A* can send up to *R* bps of packets marked with a DSCP *P* across interface *I* that will be given transit across Domain *D* with a per-packet delay not to exceed *T* msec and a loss rate not to exceed *L* between 8 am and 5 pm Monday-Friday. Packets with DSCP *P* which exceed *R* should have an expectation of being dropped." Over time, signalling between clouds may be developed that allows these agreements to be communicated and altered dynamically, but for the present useful QoS can come about from such simple pre-configured rules.

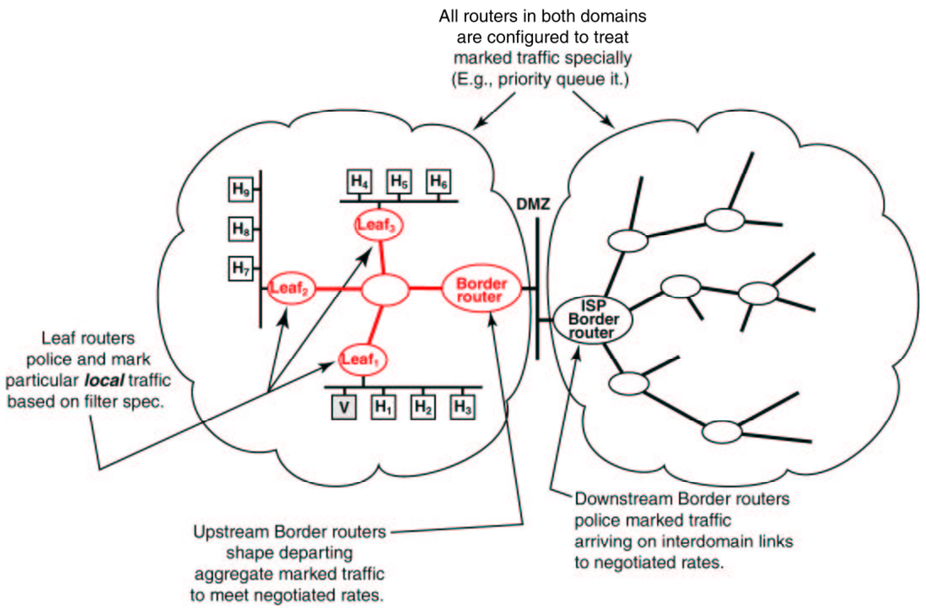
All routers in both domains are configured to treat marked traffic specially (E.g., priority queue it.)
DMZ
H9
H8
H7
H4
H5
H6
Leaf3
Leaf2
Leaf1
Border router
ISP Border router
V
H1
H2
H3
Leaf routers police and mark particular local traffic based on filter spec.
Upstream Border routers shape departing aggregate marked traffic to meet negotiated rates.
Downstream Border routers police marked traffic arriving on interdomain links to negotiated rates.

Figure 3: Connecting Clouds

QoS between clouds is illustrated in Figure 3 where an example of the functionality within an enterprise cloud (on the left) is shown connected to an ISP, on the right. The enterprise's border router shapes to ensure conformance while the ISP's border router polices to verify and enforce conformance.

3.3.3 Cloud-based QoS Example

When an application in a network wants an enhanced level of QoS and the application has already been determined to be a "worthy" application (via one of the methods described in section 5.2), there are a limited number of considerations (see figure 4). First, if source and destination are both in the same network, the decision on granting or denying the request is only dependent on resources within that network. There are a number of options for making that decision, but they only involve **Me**. This means **Me** is free to use any desired methodology and technology for handling requests and allocating QoS. Asking for QoS can be explicit (from signalling to human-in-the-loop) or implicit (packets that match certain fields or enter the network in a certain place). If the application has one endpoint that is not local, the decision on granting or denying the request must consider the resources that are shared between **Me** and the next network cloud on the path to the other endpoint. If the other endpoint is in **Not-me-1**, then this is simply a matter of staying within their shared limits. If the other endpoint is in **Far Away**, then the request must fit within the limits shared by **Me** and **Not-me-2**, while **Not-me-2** must decide if there is sufficient resource to **Far Away**.

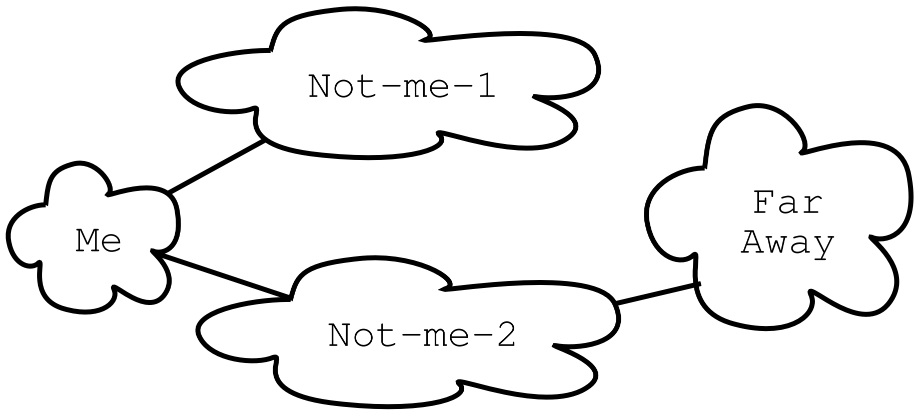

Figure 4: An Internet of clouds

This model is very flexible. Clouds that might appear to be homogeneous by one standard, for example, clouds belonging to Acme Corp., Dullard University, or Big ISP, might themselves be made up of clouds that are homogeneous by some other set of standards. For example, interior clouds could be delineated as the ATM backbone, the East Coast site, the dormitory networks, or services provided to the same customer by multiple ISPs.

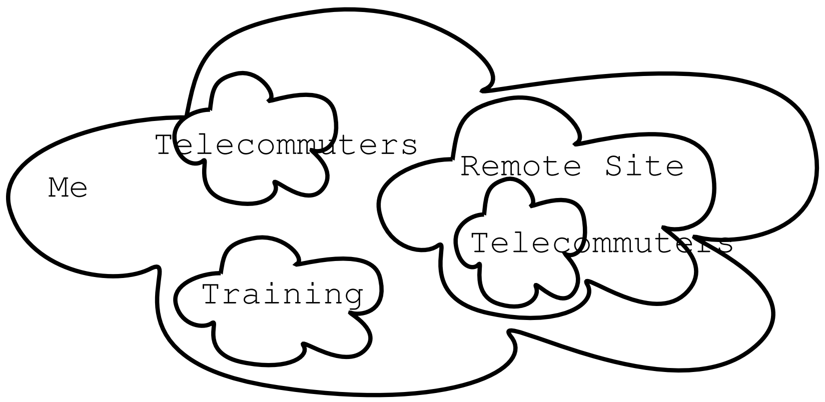

Figure 5: Clouds within clouds

Different sets of rules might apply to different interior clouds. For example, rules might note that aggregate flow into the telecommuter cloud should be bandwidth-limited if the connections there are ISDN or lower rate. Or, rules might note that inside high-bandwidth clouds, VoIP flows are trivial and not worth tracking. Furhter, rules might not be symmetric across a boundary: the out-going traffic from a telecommuter cloud might not be bandwidth-limited, while the in-bound traffic to the telecommuter cloud might be strictly limited.

4.0 PDBs: structuring aggregates to meet expectations

We have discussed how clouds can be connected to create end-to-end services, but where the clouds are independently administered, we expect the evolution of the necessary business agreements and future signalling arrangements will take some time. Early deployments are most likely to be within a single administrative domain. The specification of the transit expectations of specific traffic aggregates across clouds both assists in the deployment of QoS within a cloud and helps enable the composition of end-to-end, cross cloud services to proceed. The IETF has adopted the phrase “per domain behavior”(PDB) to capture the notion of a precise description, including quantitative parameters, of the service provided to a given traffic aggregate between the edges of a given differentiated services network domain. There is a full discussion of PDBs in [RFC3086], on which the following text is based.

A Traffic Aggregate (TA) is formally defined as a collection of packets with a Diffserv codepoint that maps to the same PHB, usually in a Diffserv domain or some subset of a Diffserv domain, i.e. within a single network cloud. Within a cloud, a TA can be uniquely identified by its DSCP. Then a Per-Domain Behavior (PDB) is defined as the expected treatment that a TA (or set of related TAs) will receive from "edge to edge" of a Diffserv domain. A particular PHB (or, if applicable, list of PHBs) and traffic conditioning requirements are associated with each PDB.

Each PDB has measurable, quantifiable, attributes that can be used to describe what happens to its packets as they enter and cross the DS domain. These derive from the characteristics of the traffic aggregate that results from application of classification and traffic conditioning during the entry of packets into the DS domain and the forwarding treatment (PHB) the packets get inside the domain, but can also depend on the entering traffic loads and the domain's topology. PDB attributes may be absolute or statistical and they may be parameterized by network properties. For example, a loss attribute might be expressed as "no more than 0.1% of packets will be dropped when measured over any time period larger than T", a delay attribute might be expressed as "50% of delivered packets will see less than a delay of d milliseconds, 30% will see a delay less than 2d ms, 20% will see a delay of less than 3d ms." A wide range of metrics is possible. In general they will be expressed as bounds or percentiles rather than as absolute values. The configuration of the cloud edge to enforce rules on what goes into a TA and how it should behave temporally is done by the control plane on a very different time scale. Much of this control plane work is still evolving, but examples might include network management tools mentioned in Section 3.3.We present some considerations for the control plane in the next section, without entering details of the control mechanisms as such.

There are currently no archival reference examples of PDBs, but some works in progress [BH, VW, AR].

5.0 Considerations for Allocation and Configuration

5.1 Control Plane Functionality

A particular PDB’s traffic aggregate can be looked at as the forwarding path portion of an edge-to-edge service, while the control plane must be used to configure the QoS tables in each router to produce the correct TA. QoS configuration does not happen at forwarding speeds and may, indeed, take place over very long time scales. Even an extremely dynamic QoS configuration is not expected to cause updates at forwarding rates or even within the lifetime of short microflows. The control plane consists of entities that can produce configuration messages based on information about policy and the state of the network. This information can be detailed or simple and can be obtained in a range of ways. As mentioned in Section 3.3, several control-plane configuration mechanisms are being developed in the IETF. This section discusses principles rather than those mechanisms.

The PHBs in the interior of a network cloud are configured to meet operational goals and are not expected to be changed frequently. Decisions as to how much of a particular traffic aggregate can be admitted depend on the resources available in a particular configuration of the cloud's interior. These resources will not be reconfigured on the same time scales on which admission control decisions are made or that provisioning rules could change.

The goal of differentiated services is *controlled* sharing of some organization's Internet bandwidth. The control can be done independently by individuals, i.e., users set bit(s) in their packets to distinguish their most important traffic, or it can be done by agents that have some knowledge of the organization's priorities and policies and allocate bandwidth with respect to those policies. Independent labeling by individuals is simple to implement but unlikely to be sufficient since it's unreasonable to expect all individuals to know all their organization's priorities and current network use and always mark their traffic accordingly. Thus, the architecture requires agents to allocate and control the sharing. One approach designates these agents as bandwidth brokers (BB) [RFC2638], that can be configured with organizational policies, keep track of the current allocation of marked traffic, and interpret new requests to mark traffic in light of the policies and current allocation. Any control plane approach must perform these same tasks so we can discuss the functionality of an allocator architecture without restricting the ways in which a particular allocator might be implemented. In particular, all control plane approaches must contain an allocation component. [RFC2638] discusses some subtleties of the tasks of such agents and how they might be hierarchical within an organization and peer across borders. These are primarily areas for research rather than standardization.

Allocators have two responsibilities. Their primary one is to parcel out a cloud's traffic allocations and set up its edge or border routers (the allocator function). In this capacity, the allocator may respond to (authenticated) requests and to information about the current state of the interior of the cloud. The other is to manage the messages, if any, that are sent across boundaries to adjacent regions. The first task is one that can be deployed in increasing sophisticated approaches as expertise in the deployment of Diffserv QoS grows. The second task is likely going to be longer in coming, but it can be important to bear it in mind architecturally.

Static vs. Dynamic. Most organizations have fixed portions of their budgets, including data communications, that are determined on an annual or quarterly basis while some additional monies might be attached to specific projects for discretionary costs that arise in the shorter term. In turn, Service Providers must do their planning on annual and quarterly bases and thus cannot be expected to provide differentiated services purely "on call". Provisioning sets up static membership and limits on traffic aggregates while call set-up allocates a portion of a traffic aggregate for a single flow's duration. Static levels can be provisioned with time-of-day specifications, but cannot be changed in response to a dynamic message. Both kinds of bandwidth allocation are expected to be important. The purchasers of special network services can generally be expected to work on longer-term budget cycles where these services will be accounted for similarly to many information services today. In addition, there needs to be a dynamic allocation capability to respond to particular events, such as a demonstration, a network broadcast by a company's CEO, or a particular network test. "Dynamic" should cover the range from a telephoned or e-mailed request to a signaled model. Strict static allocations are expected to dominate in initial deployment of QoS.

Preconfiguring of QoS is sometimes construed as "paying for bits you don't use". A more appropriate mindset is paying for the level of service that one expects to have available at any time, not unlike paying for a telephone line. Telephone customers pay an additional flat fee to have the privilege of calling a wide local area or pay by the call. Although a customer might pay on a "per call" basis for every call made anywhere, it generally turns out not to be the most economical option for most customers. It's possible similar pricing structures might arise in the internet.

Allocation refers to the process of making traffic commitments anywhere along this continuum from strictly preallocated to dynamic call set-up and an allocation architecture should be capable of encompassing this entire spectrum in any mix.

5.2 General Components

Figure 6 shows the components of the admission control and allocation. Note that there is a trusted relationship between the network elements within a network cloud, but not with entities outside the boundary. Whenever a cloud boundary is crossed, it is important to ensure that trust is not violated or to pass trust in a controlled way.

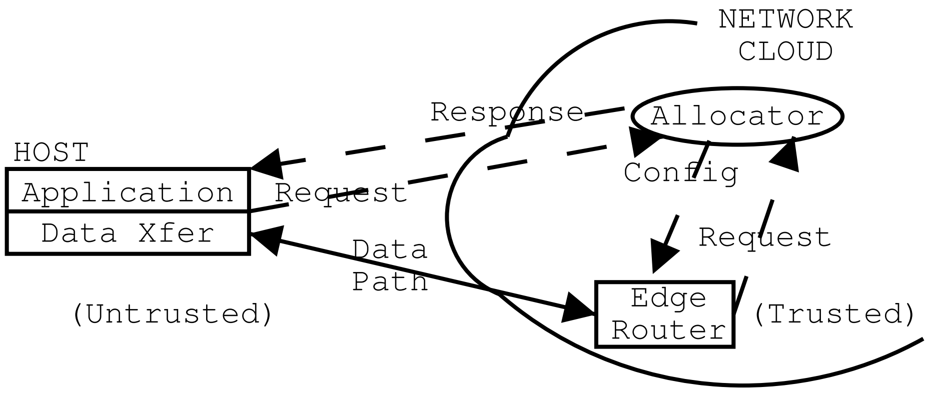

Figure 6: Admission Control

The figure attempts to capture the generality of admission control. For any specific implementation of admission control, a set of questions must be answered: What's the function of the component? How does the necessary information get to the component? In particular, a natural question is how do the border and edge routers know what to mark, police, and shape? Those answers customize a solution, but the general components remain.

Host. A host can be divided into an application and a data transfer component. A host asks for permission to send and sends (and receives) packets for a particular TA. The permission asking and granting might be dynamic or preconfigured or implicit. When an application "aims" a request at the network cloud, its physical path includes the edge router. The host always sends its data to the edge router, but it may perform some additional conditioning functions on the data (like marking and shaping).

Network Cloud. The network has the responsibility for allocating the cloud. It determines if the resources are available (to that particular user) and grants permission, configures the boundaries of the cloud to relect the rules that apply to the service request, and makes it possible/easy for the requestor to get the right allocator. the network should handle failovers. In handling requests, needs to be able to determine the level of authority of the requestor. Two key components are the allocator and, primarily as an enforcement mechanism, the edge router.

1) Allocator: Receives requests and generates a response and/or configuration. As discussed earlier, an allocator can be a single entity or a collection of entities. For simplicity, we will treat it as a single entity in this discussion. An allocator has responsibility for parceling out its cloud's totals of each behavior aggregate by configuring the edge routers. An allocator is responsible for configuring the cloud edge in response to a wide range of possible inputs, including: requests, network status, policy database, configured set ups. This allocator might exist as part of a bandwidth broker. Interior configuration should be a non-job of the allocator although it might take messages from interior that cause reconfiguration of the edge.

The allocator generates responses to requests, plus configuration actions. There are a number of ways edge router configuration can be done, some of which do not require that the allocator know the specific address/location of the edge router. For example, the allocator might send a "cookie" to a host (sender or receiver), thus passing the trust across the boundary in a limited use cookie. The allocator might multicast the configuration information to all the boundary routers. If the allocator knows the specific router to be configured for Diffserv, the information can be unicast (e.g., using SNMP, COPS-PR). Finally, the allocator might do nothing in the case of a preconfigured allocation, except to keep track of the resources allocated.

2) Edge Router: The edge router must be capable of being configured to classify and police flows into behavior aggregates. Its range of capabilities can vary somewhat. In general, it will need to know how to forward a request message to the device which is doing allocation. An edge router might be configured in a number of ways. It can terminate the request or it can pass it on to the Allocator. Termination might just result in the edge router configuring itself to handle the request (perhaps if it meets certain criteria) or the edge router may be preconfigured to handle certain requests. Either of these cases could result in a response message to the host.

In some scenarios, a **server** host may participate in these mechanisms in much the same way as an edge router, or it may cooperate very closely with an edge router, to ensure that traffic aggregates receive appropriate QoS to the very end of their network path to or from the server.

Request. Requestors include hosts, applications, users, system/network admins, trusted signals. Requests need to include the requestor's identifying information as well as information that identifies the flow, microflow, or behavior aggregate for which the request is intended. The requestor need not be either the source or destination of the request. A request is expected to contain such information as rate and burst of the target packet flow and the time period when the request is to be serviced. Requests can come from many sources since Diffserv remains agnostic about what signaling method, if any, is used.

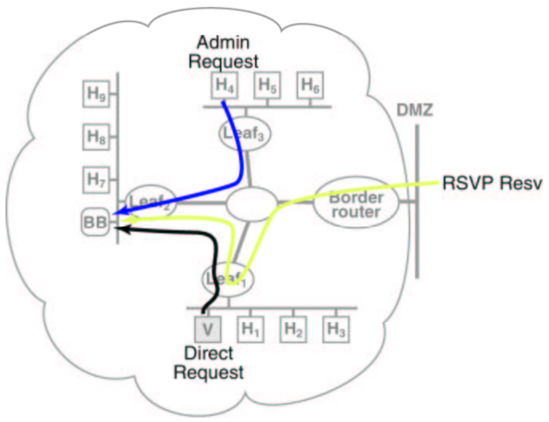

Figure 7: Request from many sources

Considerations in building services. In a simple Diffserv model, an allocator is configured by a network administrator with information about what microflows should be admitted to the TAs of which PDBs and what sort of traffic conditioning, if any, should be applied to these. The allocator pushes this information out to the appropriate edge router(s). This could be done through SNMP, COPS, or proprietary interfaces; the choice of which is unimportant here. This microflow state is only kept at the edge/border routers where packets aremarked with DSCPs and traffic conditioners are employed. In a more complex allocation model, the packet forwarding path doesn't change, just the way the allocator gets its information. In one approach, edge/border routers query the allocator when unmatched packets arrive with a DSCP that is not the default value, but this might have problems with denial-of-service attacks. Network management/alert signals should be generated for: drops by DSCP, alerts for exceeding rate allocations. The Diffserv MIB [DSMIB] will be the primary tool for this.

6 Examples:Voice over IP

We began with a discussion of how different the Internet is from the global telephony network. It seems appropriate to end with a discussion of how the Internet might subsume some of telephony's functionality. Refering to figure 4 may be useful.

6.1 Simple campus telephony

This example is based on the premise that inside a high-bandwidth cloud VoIP flows are trivial and not worth tracking. For such a network, it must be possible to configure the network so that there is a Delay Bound (DB) PHB [DB] at each network node sufficient to handle all calls. In this case, the procedure becomes: when a call is initiated, check if its destination is within the cloud, e.g. **me**. If so, just admit the call. If not, refuse the call (or do "something else"). It may be useful to set up a classifer on the edge router to ensure no "spoofing" by non-telephony traffic, but since the bandwidth for a call is so small, there's little incentive for spoofing and perhaps no real cost to the network. The bandwidth does need to be policed, either explicitly or implicitly.

6.2 Voice between (Administratively Same) Clouds

This is similar to the previous example, but here connections are only tracked in the areas of limited resources, the boundaries between clouds. Consider two bandwidth-rich clouds connected by a low-bandwidth link.The low-bandwidth link can configure a DB PHB allocation that gives a sufficiently low probability of "busy" (*bw_available*) within the bandwidth of the link. The bandwidth can be determined using traditional Erlang models. When a call is initiated, check its destination. If it's in the other cloud, check *(bw_available - call_bw)* >=*0*? If not, refuse the call. If yes, *bw_available -= call_bw* and proceed.

6.3 Voice across Clouds

Each cloud tracks connections only in the areas of limited resources, e.g., the link to the "next cloud". Assume that link is configured for a DB PHB sufficient to handle all outside calls most of the time (*bw_available*). As above, Erlang models can be used. When a call is initiated, check its destination. If the destination is **not-me**, check *(bw_available - call_bw)* >=*0*? If not, refuse call. If yes, signal/message "next cloud" and wait for reply. If reply is positive, *bw_available -= call_bw* and proceed.

If the hosts are signaling, e.g., using RSVP or a variant, the control messages can be intercepted at the edge device and sent to the cloud's allocator. The allocator configures the edge router and, when necessary, the border router, or sends messages from the border router. Several methods have been proposed for doing this, it is not important to the forwarding path which is chosen.

7.0 Going Forward

At this writing, production use of Diffserv is limited, but more products with some level of Diffserv support are entering the market. Vendors have been quick to incorporate basic Diffserv features into their products, which now gives us a rich venue for experimentation, now well under way, and for progressive operational deployment as experience accumulates. Since Diffserv can be incrementally deployed, with no requirement for synchronized deployment, we can expect to see it slowly diffuse throughout the network as routers, servers, and management systems progressively acquire Diffserv functionality. The success of Diffserv has been to use minimal standardization of the forwarding path primitives that are needed for a wide range of QoS purposes. This approach was well-placed in the history of the Internet that builds on running code, incremental deployment, separation of the forwarding and control plane, and approaches that are no more complex than needed. Diffserv's DSCP and PHBs build on past experience with the former IP Precedence bits and simple differential queuing. This is the time for deploying alternate approaches to solve the real QoS problems. The feedback from this "reality check" should provide the impetus for future standardization approaches.

Acknowledgements

K. Nichols would like to note the influence of V. Jacobson, through several years of collaboration on Diffserv, on many of the words and ideas that she contributed to this paper. Some text has been adapted from [DIVERS] and [RFC3086]. We are grateful to the many active participants in the IETF Differentiated Services working group and to successive IETF Transport Area Directors Allyn Romanow, Scott Bradner and Allison Mankin.

References

Note: all RFCs may be found on-line either at http://www.rfc-editor.org/rfcsearch.html or at http://www.ietf.org/rfc

[RFC2474] RFC 2474, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", K.Nichols, S. Blake, F. Baker, D. Black December 1998

[RFC2475] RFC 2475, "An Architecture for Differentiated Service", S. Blake, F. Baker, D. Black, December 1998.

[RFC2597] RFC 2597, "Assured Forwarding PHB Group", F. Baker, J. Heinanen, W. Weiss, J.Wroclawski, June 1999

[RFC2598] RFC 2598, "An Expedited Forwarding PHB," V. Jacobson, K. Nichols, K. Poduri, June 1999.

[RFC2638] RFC 2638, "A Two-bit Differentiated Services Architecture for the Internet", K.Nichols, V. Jacobson, L. Zhang, July 1999

[RFC3086] "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification", K. Nichols and B. Carpenter, April, 2001

[CAIDA] S. McCreary and k claffy, "Trends in Wide Area IP Traffic Patterns", May, 2000, http://www.caida.org/outreach/papers/

[Clark95] D. Clark, "Adding Service Discrimination to the Internet", Proceedings of the 23rd Annual Telecommunications Policy Research Conference (TPRC), Solomons, MD, October 1995.

[Clark88] The Design Philosophy of the DARPA Internet Protocols, D.D.Clark, Proc SIGCOMM 88, ACM CCR Vol 18, Number 4, August 1988, pages 106-114 (reprinted in ACM CCR Vol 25, Number 1, January 1995, pages 102-111).

[Saltzer] End-To-End Arguments in System Design, J.H. Saltzer, D.P.Reed, D.D.Clark, ACM TOCS, Vol 2, Number 4, November 1984, pp 277-288.

[DSMODEL] "An Informal Management Model for Diffserv Routers",Y. Bernet, S. Blake, D. Grossman, A. Smith, work in progress, 2001

[DSMIB] "Management Information Base for the Differentiated Services Architecture", F. Baker, K. Chan, A. Smith, work in progress, 2001

[DSPIB] “Differentiated Services Quality of Service Policy Information Base”, M. Fine, K. McCloghrie, J. Seligson, K. Chan, S. Hahn, C. Bell, A. Smith, F. Reichmeyer, work in progress, 2001

[DSMPLS] “MPLS Support of Differentiated Services”, F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, J. Heinanen, work in progress, 2001

[BH] “A Bulk Handling Per-Domain Behavior for Differentiated Services”, B. Carpenter and K. Nichols, work in progress, 2001, available at http://www.packetdesign.com/publications.html.

[VW] “A Virtual Wire Per-Domain Behavior”, V. Jacobson, K. Nichols, and K. Poduri, work in progress, 2000, available at http://www.packetdesign.com/publications.html.

[AR] “An Assured Rate Per-Domain Behaviour for Differentiated Services”, N. Seddigh, B. Nandy, J. Heinanen, work in progress, 2001

[RFC2998] RFC 2998, “A Framework for Integrated Services Operation over Diffserv Networks”, Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, E. Felstaine, November 2000.

[Wittbrodt] “CAR Talk: Configuration Considerations for Cisco's Committed Access Rate”, C. Wittbrodt, NANOG, November 1998, http://www.nanog.org/mtg-9811/ppt/witt/index.htm

[Wilder] “Wide-area Internet traffic patterns and characteristics”, K. Thompson, G. Miller, and R. Wilder, IEEE Network, Volume: 11 Issue: 6 , Nov.-Dec. 1997, pages 10-23.

[FDDIFS] Minutes on-line at http://www.ietf.org/proceedings/97apr/97apr-final/xrtft122.htm

[IETF] List of IETF working groups on-line at http://www.ietf.org/html.charters/wg-dir.html

[FAIL] V. Paxson and S. Floyd, “Wide Area Traffic: The Failure of Poisson Modeling”, IEEE Transactions on Networking, June1995, pp. 226-244

[SELF] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, “On the Self-Similar Nature of Ethernet Traffic,” Proceedings of ACM SIGCOMM '93, pp. 183-193.

[DIVERS] B.E.Carpenter & D.D.Kandlur, “Diversifying Internet Delivery”, IEEE Spectrum, November 1999, 57-61

[UUNET] http://www.uu.net/us/support/sla/servicessupported/uuhost_enterprise.xml, December 11, 2001.

[DB] G. Armitage, A. Casati, J. Crowcroft, J. Halpern, B. Kumar, J. Schnizleindraft, “A Delay Bound alternative revision of RFC2598”, work in progress, 2001.