Internet Draft
draft-ietf-issll-ds-map-01.txt
Expires August, 2001

J. Wroclawski
MIT LCS
A. Charny
Cisco Systems
February, 2001

# Integrated Service Mappings for Differentiated Services Networks

## Status of this Memo



## Abstract

This document describes mappings of IETF Integrated Services onto IETF differentiated services networks. These mappings allow appropriately engineered and configured differentiated service network clouds to play the role of "network elements" in the Integrated Services framework, and thus to be used as components of an overall end-to-end Integrated Services QoS solution.

## 1. Introduction

The IETF Integrated Services framework [INTSERV] defines mechanisms and interfaces for providing network Quality of Service control useful for applications that require more predictable network service than is





available with the traditional best-effort IP delivery model. Provision of end-to-end QoS control in the Intserv model is based on the concatenation of "network elements" along the data transmission path. When all of the concatenated network elements implement one of the defined Intserv "services" [G,CL], the resulting data transmission path will deliver a known, controlled QoS defined by the particular Intserv service in use.

The IETF Differentiated Services framework [DIFFSERV] defines a number of mechanisms for differentiating different traffic streams within a network and providing different levels of delivery service to those different streams. These mechanisms include differentiated per-hop queuing and forewarding behaviors, as well as behaviors such as traffic classification, metering, policing and shaping that are intended to be used at the edge or boundary of a diffserv cloud. Crucially, the Differentiated Services framework manages traffic forwarding behavior within a diffserv cloud at the aggregate level, rather than the per-application-flow level.

The availability of Differentiated Services per-hop and cloud-edge behaviors, together with additional mechanisms to statically or dynamically limit the absolute level of traffic within a traffic class, allows an IETF Differentiated Services network cloud to act as a network element within the Integrated Services framework. In other words, an appropriately designed, configured and managed Diffserv network cloud can act as one component of an overall end-to-end QoS controlled data path using the Integrated Services framework, and therefore support the delivery of Intserv QoS services.

This document is one of a set that together describe the usage of Differentiated Services networks in this manner. This document describes methods for implementing Intserv using Diffserv network behaviors and mechanisms. Companion documents [RSVPAGGR, DCLASS] define extensions to the RSVP signaling protocol [RSVP] that are useful in this environment.It is recommended that readers be familiar with the overall framework in which these mappings and protocols are expected to be used; this framework is described fully in [ISDSFRAME].

Within this document, Section 2 describes the overall approach and discusses issues that are independent of the class of Intserv service being implemented. Section 3 discusses implementation of the Controlled Load service. Section 4 discusses implementation of a mathematically correct Guaranteed service, and presents information about the

performance and limitations of this implementation. Section 5 discusses implementation of close approximations to the Guaranteed service that may be acceptable in some circumstances and may allow more efficient use of network resources. Section 6 briefly describes the relationship of the mechanisms described here to the Intserv Null Service [NULL].

## 2. Basics

### 2.1. Components

Figure 1 shows the basic use of a Diffserv network cloud as an Intserv





network element. In this figure, Intserv functions within the non-Diffserv regions take place at the level of individual switches, routers, subnets, and similar ojects. In contrast, the entire Diffserv region acts as a _single_ Intserv network element; using components of the Diffserv architecture to implement the behaviors expected of an object in the Intserv environment.

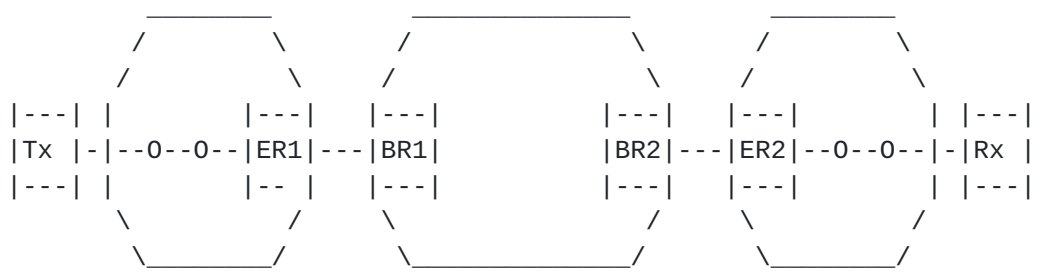


Figure 1: Sample Network Configuration

Figure 1 <ascii art TBA>

The figure shows that required Intserv network element functions are mapped to the Diffserv cloud as follows:

- Traffic scheduling. The Intserv traffic scheduling function is supported by appropriately selected, configured, and provisioned PHB's within the Diffserv network. These PHB's, when concatenated along the path of traffic flow, must provide a scheduling result that adequately approximates the result defined by the Intserv service.

  In general, the PHB concatenation will only be able to approximate the defined Intserv service over a limited range of operating conditions (level of traffic, allocated resources, and the like). In that case, other elements of the network, such as shapers and policers, must ensure that the traffic conditions seen by the PHB's stay within this range.

- Traffic classification. The Intserv framework requires that each network element (re)classify arriving traffic into flows for further processing. This requirement is based on the architectural assumption that network elements should be independent, and not depend on other network elements for correct operation.

  NOTE: the Intserv framework does not specify the granularity of a flow. Intserv is often associated with per-application or per-session end-to-end flows, but in fact any collection of packets that can be described by an appropriate classifier can be treated as an Intserv traffic flow.

  When Intserv is mapped to Diffserv, packets must be classified into flows, policed, shaped, and marked with the appropriate DSCP before they enter the interior of the diffserv cloud. Strictly speaking, the independence requirement stated above implies that the ingress boundary router of each diffserv cloud must implement a MF classifier to perform the classification function. However, in keeping with the





  diffserv model, it is permissible to push the flow classification function further towards the edge of the network if appropriate agreements are in place. For example, flows may be classified and marked by the upstream edge router if the Diffserv network is prepared to trust this router.

- Policing and shaping. In terms of location in the network, these functions are similar to traffic classification. A strict interpretation of the Intserv framework would require that the ingress boundary router of the diffserv cloud perform these functions. In practice, they may be pushed to an upstream edge router if appropriate agreements are in place.

  Note that moving the shaping function upstream of the diffserv ingress boundary router may result in poorer overall QoS performance. This is because if shaping is performed at the boundary router, a single shaper can be applied to all of the traffic in the service class, whereas if the shaping is performed upstream separate shapers will be applied to the traffic from each upstream node. As discussed further in Section 4, the single shaper may be preferable in some circumstances.

- Admission control. The quantitative Intserv services (Guaranteed and Controlled Load) require that some form of admission control limit the amount of arriving traffic relative to the available resources. Two issues are of interest; the method used by the diffserv cloud to determine whether sufficient resources are available, and the method

used by the overall network to query the diffserv cloud about this availability.

Within the cloud, the admission control *mechanism* is closely related to resource allocation. If some form of static resource allocation (provisioning) is used, the admission control function can be performed by any network component that is aware of this allocation, such as a properly configured boundary router. If resource allocation within the network cloud is dynamic (a dynamic "bandwidth broker" or signaling protocol) then this protocol can also perform the admission control function, by refusing to admit new traffic when it determines that it cannot allocate new resources to match.

The admission control *mechanism* used is independent of the admission control *algorithm* used to determine whether sufficient resources are available to admit a new traffic flow. The algorithm used may range from simple peak-rate allocation to a complex statistical measurement-based approach. The choice of algorithm is dependent on the Intserv service to be supported. Admission control algorithms appropriate for each service are <not yet> discussed in the service specific sections below.

The admission control mechanism used within the diffserv cloud is also independent of the mechanism used by the outside world to request service from the cloud. As an example, end-to-end RSVP might be used together with any form of interior admission control mechanism - static provisioning, a central bandwidth broker, or aggregate RSVP internal signalling.





## 2.2. Per-Cloud versus Per-Path Control

The key to providing absolute, quantitative QoS services within a diffserv network is to ensure that at each hop in the network the resources allocated to the PHB's used for these services are sufficient to handle the arriving traffic. As described above, this can be done through a spectrum of mechanisms ranging from static provisioning to dynamic per-hop signaling within the cloud. Two situations are possible:

- With per-cloud provisioning, sufficient resources are made available in the network so that traffic arriving at an ingress point can flow to *any* egress point without violating the PHB resource allocation requirements. In this case, admission control and traffic management decisions need not be based on destination information.

- With per-path provisioning, resources are made available in the network to ensure that the PHB resource allocation requirements will

not be violated if traffic arriving at an ingress point flows to one (in the unicast case) specific egress point. This requires that admission control and resource allocation mechanisms take into account the egress point of traffic entering the network, but results in more efficient resource utilization.

Two points are important to note:

- Both approaches are valuable, but all functions must adopt the same approach. Particularly, if resource allocation is per-path, traffic shaping and policing, and hence classification must be destination aware as well.

- The per-cloud vs per-path decision is independent of decisions about static vs. dynamic provisioning. It is often assumed that dynamic provisioning is necessarily per-path, while static provisioning is more likely to be per-cloud. In reality, all four options may be useful in differing circumstances.

## 3. Implementation of the Controlled Load Service

### 3.1. Summary of CL Requirements

The essence of the Controlled Load service is that traffic using it experiences the performance expected of an unloaded network. The CL specification [CL] refines this definition.

- Controlled Load traffic is described by a token bucket Tspec. When traffic is conformant to the Tspec, network elements will forward it with queuing delay not greater than that caused by the traffic's own burstiness - that is, the result of the source emitting a burst of size B into a logical network with capacity R. Further, in doing this no packets will be discarded due to queue overflow. Statistically rare deviations from this ideal behavior are permitted. A measure of the "quality" of a CL service is how rare these deviations are.





NOTE: the actual behavior requirements stated in the CL spec are slightly more detailed than what is presented here.

- Network elements must not assume that that arrival of nonconformant traffic for a specific controlled-load flow will be unusual, or indicative of error. In certain circumstances large numbers of packets will fail the conformance test *as a matter of normal operation*. Some aspects of the behavior of a CL network element in the presence of nonconformant traffic are specified.

(These circumstances include elements carrying traffic from adaptive applications that use the CL service to provide a floor on performance

but constantly try to do better, elements acting as the "split points" of a multicast distribution tree or carrying multi-source aggregate flows, such as those generated by RSVP's wildcard or shared-explicit reservation styles supporting a shared reservation).

In the presence of nonconformant packets arriving for one or more controlled-load flows, each network element must ensure locally that the following requirements are met:

1) The network element MUST continue to provide the contracted quality of service to those controlled-load flows not experiencing excess traffic.

2) The network element SHOULD prevent excess controlled-load traffic from unfairly impacting the handling of arriving best-effort traffic.

3) Consistent with points 1 and 2, the network element MUST attempt to forward the excess traffic on a best-effort basis if sufficient resources are available.

These points lead to two observations about a well implemented CL service.

- CL traffic can be sorted into "delay classes" based on burstiness. Highly bursty flows, having a large ratio of Tspec parameters B/R, should expect to experience more queuing delay than their low-burstiness counterparts. Thus, a good CL implementation will sort the offered CL traffic into sub-classes that are expecting roughly equivalent delay, and queue these subclasses independently to achieve this result.

- The CL specification leaves open the precise treatment of nonconformant traffic, giving only the minimum requirements listed above.

  NOTE: The phrase "best effort basis" in the portion of the CL spec quoted above has sometimes been taken to mean "the traffic must be placed in the best effort traffic class and treated identically to BE traffic". This interpretation is incorrect. It is easy to see this at one level, because if nonconformant CL traffic from non-adaptive applications is simply lumped in with adaptive best-effort traffic it will tend to unfairly impact that traffic, in





contravention of point 2). However, the intent of the specification is more general. An appropriate reading is "nonconformant CL traffic should be transmitted, when possible, in the way that is most advantageous to users and applications, subject to the requirements on non-interference with other traffic". This allows

the CL service to be used both to provide a specific QoS for non-adaptive applications and as to provide a "floor" or minimum QoS for adaptive applications.

### 3.2. Implementation of CL using the AF Per-Hop Behavior

The CL service can be supported most effectively using an appropriately designed and configured Assured Forwarding PHB implementation [AF] as the data forwarding element. This approach SHOULD be used whenever possible.

The basics of the AF-based approach are as follows:

- Sort the offered CL traffic into delay classes based on the B/R ratio of the Tspec. The packets of each delay class will be forwarded using a separate instance of the AF PHB.

- For each delay class, construct an aggregate Tspec for the admitted traffic according to the rule for summing Tspecs given in [CL]. This Tspec will be used to police the traffic for conformance at the ingress to the diffserv cloud.

- For each delay class, police arriving packets against the token bucket Tspec derived above. Mark conforming packets with a DSCP indicating the selected AF instance, and highest priority forwarding within that instance. Mark nonconformant packets with a DSCP indicating the selected AF instance, and lowest priority forwarding within that instance.

- At each node within the diffserv network, configure each AF instance appropriately by:

  a) setting the actual queue size (or alternatively the dropping parameters for high priority packets) to limit queuing delay to the delay class's target. (In other words, packets that have been delayed beyond the class target should be dropped).

  b) setting the dropping parameters for low priority packets to drop such packets as soon as any significant non-transient queuing of these packets is detected.

  c) setting the service rate of the AF instance to a bandwidth sufficient to meet the delay and loss behavior requirements of the CL spec when only high-priority packets are present.

- Implement an admission control algorithm that ensures that at each hop in the network the level of conformant traffic offered to each AF instance is equal to or less than that provisioned for in step 4c above (or alternatively dynamically allocates more bandwidth to the relevant AF instance when required).

In addition to these basic actions, two subtleties with the use of AF must be observed.

First the relationship between different AF instances, and between AF and other PHBs, must be more tightly constrained than is required by the the base AF specification.

- Bandwidth should be allocated between AF and BE (and any other relevant PHB's) in such a way that AF cannot simply steal all best-effort bandwidth on demand. A simple WFQ or CBQ scheduler can meet this requirement.

- The bandwidth allocation relationship between different AF instances must be known. Two likely relationships are

  o Bandwidth is allocated to each AF instance independently, as with a WFQ scheduler.

  o Bandwidth is allocated across the AF instances used for CL service on a priority basis, with the AF instance supporting the lowest delay class of CL having the highest priority.

Either of these approaches may be used. However the choice of approach affects the admission control decision, and must be taken into account. In the first case, admission control decisions may be made for each CL delay class independently. In the second case, admission control decisions for high priority classes will affect lower priority classes, which must be taken into account.

The second subtlety is that the implementation of AF must service the AF classes in a timely manner, by ensuring that the bandwidth allocated to an AF instance is made available at a time-scale substantially shorter than the delay target of the class. This requirement is slightly stronger than that stated in the AF specification. In practice, any implementation using a common queuing algorithm is likely to be able to meet this requirement unless other PHB's, such as EF, are served at higher priority. When that is true, the traffic seen by the higher priority PHB will also require limiting and shaping in order to ensure that the CL AF instances receive bandwidth on a timely basis.

The overall result of this procedure is an implementation of the CL service with the following characteristics:

- Conformant CL traffic is carried according to the CL requirements.

- Resources are used efficiently by aggregating traffic with similar requirements, but supporting multiple delay classes for traffic with widely differing requirements.

- Non-CL traffic is carried whenever resources permit, and is not reordered with respect to the CL flow's conformant traffic.

- Nonconformant CL traffic is not able to disrupt traffic of other classes, particular BE.





### 3.2.1 CL/AF Admission Control Approaches

<TBA?>

## 3.3. Implementation of CL using the EF Per-Hop Behavior

It is also possible to implement an approximation of the Controlled Load service using the Diffserv Expedited Forwarding [EF] PHB as the traffic scheduling element. This approach is not preferred, because of two significant limitations. Therefore, this approach SHOULD NOT be used unless the AF-based approach is not available.

- Because there is only one EF scheduling class per node, it is impossible to sort the Controlled Load traffic into queuing delay classes, as described above for the AF implementation. Instead, all CL traffic must be handled as one scheduling class, and sufficient resources must be allocated to the class to cause *all* CL traffic to meet the queuing delay expectations of the most demanding flows.

- Because the EF PHB requires a hard limit on the amount of traffic passing through it, a CL service implemented using EF cannot handle nonconformant (over-Tspec) traffic gracefully, as can be done with AF. Instead, nonconformant traffic must either be discarded at the ingress of the Diffserv cloud or remarked into a different behavior aggregate, and thus potentially reordered in transit. Either of these behaviors is less desirable than the one obtained from the AF-based implementation above.

Notwithstanding these limitations, it may be useful to implement a CL approximation based on the EF PHB when the Diffserv network does not support the AF PHB, or when the implementation of the AF PHB cannot assure the forwarding of traffic in a sufficiently timely manner. In this case:

- All CL traffic is marked with a DSCP corresponding to the EF PHB.

- A single aggregate Tspec for all CL traffic is computed for each network ingress.

- Arriving CL traffic is policed against this Tspec, and nonconformant traffic is either discarded or remarked as BE, at the preference of the network operator.

- At each hop within the network the EF PHB must receive a bandwidth allocation sufficient to meet the requirements given in the EF specification when the arriving CL traffic is at the Tspec level for that point within the network.

- The topology of the network must be designed so that the instantaneous queuing delay caused by fan-in to a node will exceed the CL requirements rarely or never. In practice, this will be a concern only with very high fan-in topologies.





## 4. Implementation of the Guaranteed Service

The Guaranteed service [G] offers a strict mathematical assurance of both throughput and queuing delay, assuming only that the network is functioning correctly. A key concept of the Guaranteed service is that "error terms", referred to as C and D in the specification, are provided by the network element to the customer, allowing the customer to calculate the bandwidth it must request from the network in order to achieve a particular queuing delay target. Thus, the two important tasks in implementing a Guaranteed service network element are providing the traffic scheduling, policing, and shaping functions needed to support a hard bound on performance, and characterizing the network element's error terms so that the customer of the service can accurately characterize the network path and deduce what level of resources must be requested.

Our strategy for implementing these capabilities within a diffserv cloud revolves around the use of the EF PHB for Guaranteed traffic, together with the shaping and policing functions necessary to obtain a performance bound in this context. The basic traffic policing and shaping requirements for Guaranteed service are discussed more fully in the service specification.

Delay through a Diffserv cloud can be roughly classified into propagation and serialization delay, shaping/reshaping delays at the boundary, and queuing delay inside the cloud. In order to determine the error terms C_dc and D_dc for the Diffserv cloud needed for end-to-end determination of end-to-end delay, each of these delay components need to be evaluated. The difficulty in characterizing C_dc and D_dc is that unlike the Intserv model, where the C and D terms are a local property of the router, in the case of Diffserv cloud these terms depend not only on the topology of the cloud, but also on the internal traffic characteristics of potentially _all_ EF traffic in the cloud.

Hence, the existence of upper bounds on delay through the cloud implies centralized knowledge about the topology of the cloud and traffic

characterization. In turn, dependence of the delay bounds on traffic characterization at any ingress point to the cloud implies the existence of a policy that defines traffic characterization rules, as well as implementation mechanisms at _all_ ingress points in the network that enforce that policy.

These considerations imply that determination of the bound on the delay through the Diffserv cloud should be performed off-line, perhaps as part of a traffic management algorithm, based on the knowledge of the topology, traffic patterns, shaping policies, and other relevant parametersof the cloud. These parameters are discussed in the following sections with respect of each delay component.

Once the delay bounds and determined, the corresponding error terms C_dc and D_dc are configured into the appropriate intserv-capable edge routers, as discussed below.





## 4.1 Propagation and Serialization Delay.

These delay components can be bounded by modeling the Diffserv cloud as a sequence of at most h links, each of which of at most length C. The parameters (h, C) determine the so-called "diameter" of the cloud. The knowledge of this diameter can then be used to obtain upper bounds on the propagation and serialization delay through the cloud.

## 4.2 Shaping delay.

The Diffserv EF PHB assumes that traffic entering the Diffserv region is conditioned at the Diffserv cloud boundary. In the framework of Figure 1, shaping is expected to take place at the ingress edge router ER1, and optionally at the boundary router BR1. Granularity of such shaping is implementation dependent, and can range from microflow shaping to aggregate shaping. The granularity of aggregation can be "all EF traffic between a particular ingress-egress pair", which is frequently referred to as "pipe model", or "all EF traffic originating at a given ingress to all possible destinations", which is frequently referred to as "hose model".

In addition to ingress shaping, the Diffserv model allows re-shaping traffic at the egress point. As for the case of ingress shaping, the egress shaping can be implemented either at BR2 or ER2.

The effect of different choices of the location and granularity of shaping on the delay guarantees that can be provided by a Diffserv cloud will be discussed in section ??. In this section we consider the effect of this choices on the C and D terms advertised by the Interv-capable routers ER1 and ER2. Note that the Intserv capable router downstream from the Diffserv cloud (ER2 in the reference network of Figure 1) is

responsible for exporting the C and D terms of the Diffserv cloud.

### 4.2.1. Shaping at the Edge Routers

If shaping is performed at the ingress edge router ER1, and reshaping, if any, is performed at ER2, but there is no shaping implemented inside the Diffserv cloud, the shaping/reshaping delay is part of the total delay advertised by the edge routers ER1 and ER2, and hence the corresponding C and D terms are exported by the Intserv-capable edge routers. These will be denoted as C_is, D_is, C_es, D_es respectively, where the indices _is and _es denote "ingress shaper" and "egress shaper". The values of these parameters are implementation dependent.

Since the Diffserv cloud itself does not perform any shaping in this case, its C_dc should be set to zero. The determination of the value of D_dc and factors affecting it are discussed in section 4.4 below.

### 4.2.2 Shaping at the boundary routers

In the case where shaping is performed by the boundary routers, shaping and reshaping delay become part of the delay of the Diffserv cloud and hence have to be accounted for in the C_dc and D_dc error terms. Note





that depending on the shaping implementation, the rate-dependent error term may not necessarily be zero, and hence ingress shaping may add a non-zero component to the C_dc value of the Diffserv cloud.

Since the ingress shaping delay depends on the shaping implementation and shaping granularity at the border router, and since different border routers may implement different shaping algorithms, it seems natural to dedicate the responsibility to export the error terms for ingress shaping delay to the ingress edge router(s) attached to the border router.

It is important to note that in the case of aggregate shaping, the shaping delay may be a function of the combined burst and combined rate of all microflows comprising the shaped aggregate (note that the aggregate may consist of microflows arriving from different ingress points).

To enable an existence of a meaningful upper bound on the shaping delay the shapers at the edge routers must be configured in such a way as to ensure the existence of the bound on the shaping delay at the boundary router. This may be accomplished by emposing a policy such as "token bucket parameters of all flows requiring G support entering the diffserv cloud from any edge router should satisfy the condition (r>=r_min, b<=b_max). Such conditions would enable token bucket characterization

of the aggregate stream, which in combination with the properties of the shaping implementation would enable the computation of an upper bound for a particular microflow.

If the egress boundary router implements reshaping on an aggregate basis, just as in the case if ingress shaping, the egress reshaping delay of a microflow depends on the combined rate and burstiness of the aggregate which is being reshaped. Aggregate burstiness depends, among other things, on the parameters of ingress shapers and on the delay bound of the diffserv cloud incurred by all microflows after the last shaping point.

The C and D terms corresponding to the egress boundary shaping must be configured at the egress edge router, which is responsible for exporting the egress shaping component of the C and D terms of the Diffserv cloud.

In addition, just as in section 4.2.1, the egress edge router is responsible for exporting the D_ds component of the delay inside the diffserv cloud which is not due to the shaping or reshaping delays.

### 4.2.3. Shaping inside the Diffserv cloud

While the Diffserv model does not prevent shaping inside the cloud as well as at the boundaries, this draft will concentrate on the most common case when all internal interfaces of any node in the diffserv cloud implement work-conserving aggregate class-based scheduling only.



## 4.3 Queuing delay

Queuing delay experienced by a given packet is caused by two reasons: contention with other packets in the scheduler and the interruption of service experienced by the scheduler as a whole. A typical example of the latter is the delay in a single processor system when the processor schedules some tasks other than packet scheduler. If a bound on this latter portion of the delay is known for all routers inside the diffserv cloud, then the contribution of this delay component can be bounded by multiplying this bound by the max hop count h.

The component of the queuing delay due to contention with other packets in the link scheduler will be discussed in detail in section 4.4. For the sake of brevity, in the rest of this draft the term queuing delay will be used to refer to just the portion of the queuing delay due to contention with other packets in the scheduler.

## 4.4. Queueing delay bounds in the Diffserv Cloud

The main difficulty in obtaining hard delay bounds for an arbitrary topology cloud arises from the assumption of aggregate scheduling inside the cloud. When a packet of some flow f traverses a sequence of aggregate queues, its worst case delay may depend on the traffic of other flows which do not even share a single queue with the flow. Moreover, the delay of a packet p of flow f at time t may be affected by flows whose last packets have exited the network long before the first packet of flow f entered the network [CHARNY].

The ability to provide hard delay bounds in a Diffserv cloud with aggregate scheduling must rely on cooperation of all devices in the cloud, as well as strict constraints on the traffic entering the cloud.

It has been demonstrated that the knowledge of the following parameters global to the cloud is essential for the ability to provide strict queuing delay guarantees across the Diffserv cloud [CHARNY],[LEBOUDEC]:

- limited number of hops of any flow across the cloud (denoted h)

- low (bounded) ratio of the load of EF traffic to the service rate of the EF queue on any link in the cloud (denoted u)

- minimum rate of the shaped aggregate (denoted r_min)

- maximum token bucket depth of an edge-to-edge aggregate (denoted b_max)

- minimum service rate of the EF queue (denoted S)

- maximum deviation of the amount of service of the EF queue from the ideal fluid service at rate S (denoted E)





Currently, the only known delay bound that holds for an arbitrary topology and arbitrary route distribution is given in [LeBouldec] by

$$D = (E/S + ub\_max/r\_min) \times h/(1-u(h-1))$$

which holds for any utilization $u<1/(h-1)$. This bound holds for the case when the capacity of any single link is substantially smaller than the total capacity of all interfaces of any router. (This bound may be slightly improved if the capacity of a single link is not negligible compared to the total router capacity [LeBouldec]). Unfortunately, this bound explodes when $u=1/(h-1)$.

Some knowledge on either the topology or the routes in the cloud may yield to an improved bound. For example, for a class of network topologies which includes a multistage network it can be shown [CHARNY] that the bound is given by

$$D = (E/S + ub\_max/r\_min)x((1+u)^{h-1})/u$$

While this bound holds for any utilization, due to the exponential term the delay grows very fast with the increase in utilization u.

Unfortunately, at the moment no bound is known for a general topology with utilization greater than 1/(h-1). It can be shown [CHARNY], that for utilization values greater than 1/(h-1), for any value of delay D one can always construct a network such that the delay in that network is greater than D. This implies that either no bound exists at all, or if a bound does exist, it must depend on some additional characteristics of the network other than just h and u.

The practical implication of these results is that, barring new results on delay bounds, the amount of traffic requiring end-to-end Guaranteed service across the diffserv cloud should be rather small. Furthermore, it also implies that if substantial amount of other EF traffic is present in the network, in order to ensure strict delay bounds for GS traffic, buffering and scheduling mechanisms must exist that ensure separation of the GS traffic using EF PHB from other traffic using EF PHB.

## 4.5. Relationship to Bandwidth Allocation Techniques and Traffic Conditioning Models

### 4.5.1. Availability of sufficient bandwidth

As discussed in Section 4.4, in order to provide a strict delay bound across the Diffserv cloud the ratio of the EF load to the service rate of the EF queue has to be deterministically bounded on all links in the network. This can be either ensured by signaled admission control (such as using RSVP aggregation techniques [RSVPAGGR] or by a static provisioning mechanism. It should be noted that if provisioning is used, then to ensure deterministic load/service rate ratio on all link the network should be strongly overprovisioned to account for possible inaccuracy of traffic matrix estimates.





In either case deterministic availability of sufficient bandwidth on all links is a necessary condition for the ability to provide deterministic delay guarantees.

### 4.5.2. Effect of Shaping Granularity on Delay Bounds

A related, although different issue for the ability to provide delay deterministic delay guarantees is the granularity of the ingress shaping. The implications of different choices on the resulting delay bounds are discussed in the following subsections.

#### 4.5.2.1 Per-microflow shaping

The known worst case delay bound is linear in the ratio b_max/r_min. In the case of microflow shaping, the minimal rate of the microflow can be quite small, resulting in a large delay bound. There is a substantial advantage therefore in aggregating many small microflows into an aggregate and shaping the aggregate as a whole. While in principle there is a range of choices for aggregation, this document will consider only two: edge-to-edge aggregation and edge-to-everywhere aggregation.

#### 4.5.2.2. Shaping of edge-to-edge aggregates

This type of shaping is natural for explicit bandwidth reservation techiques. In this case r_min and b_max relate to the rate and token bucket depth of the border-to-border aggregates. Since the delay bound is linear in b_max/r_min, aggregating as many microflows sharing the same border-to-border pair as possible results in the increase of r_min, and hence in the decrease of the delay bound. The location of the shaper at the border router is therefore beneficial for reducing the edge-to-edge delay bound.

#### 4.5.2.3. Shaping of edge-to-everywhere aggregates.

This type of shaping is frequently assumed in conjunction with bandwidth provisioning. The effect of this choice on delay bounds depends on exactly how provisioning is done. One possibility for provisioning the network is to estimate edge-to-edge demand matrix for EF traffic and ensure that there is sufficient capacity to accommodate this demand, assuming that the traffic matrix is accurate enough. Another option is to make no assumption on the edge-to-edge EF traffic distribution, but rather admit a certain amount of EF traffic at each ingress edge, regardless of the destination edge, and provision the network in such a way that even if _all_ traffic from _all_ sources happens to pass through a single bottleneck link, the capacity of that link is sufficient to ensure the appropriate load to service rate ratio for the EF traffic.

Depending on which of the two choices for provisioning is chosen, shaping of the edge-to-everywhere aggregate has the opposite effect on the delay bound.

In the case of "edge-to-edge provisioning", the bandwidth of any link may be sufficient to accommodate the _actual_ load of EF traffic while

remaining within the target utilization bound. Hence, it is the minimal rate and the maximum burst size of the _actual_ edge-to-edge aggregates sharing any link that effect the delay bound. However, aggregate edge-to-all shaping may result in individual substreams of the shaped aggregate being shaped to a much higher rate than the expected rate of that substream. When the edge-to-everywhere aggregate splits inside the network into different substreams going to different destinations, each of those substreams may have in the worst case substantially larger burstiness than the token bucket depth of the aggregate edge-to-everywhere stream. This results in substantial increase of the worst case delay over the edge-to-edge shaping model. Moreover, in this case the properties of ingress shapers do not provide sufficient information to bound the worst case delay, since it is the burstiness of the _substreams_ inside the shaped aggregates that is needed, but is unknown.

In contrast, if the "worst case" provisioning is assumed, the network is provisioned in such a way that each link can accommodate all the traffic even if all edge-to-everywhere aggregates end up sharing this link. In this case the r_min and b_max of the edge-to-everywhere aggregate should be used without modification in the formula for the delay bound. Intuitively, in this case the actual traffic distribution can only be better than the worst case, in which all the aggregate traffic at a given ingress is destined to the same "worst case egress".

Note that the "worst case" provisioning model targeting a particular utilization bound results in substantially more overprovisioning than the the "point-to-point" provisioning using an estimated traffic matrix, or explicit point-to point bandwidth allocation using signaled admission control.

### 4.6 Concatenation of Diffserv Clouds

In the case where one or more Diffserv clouds are concatenated via an Intserv-capable node, the total delay is simply a concatenation of delays computed for each individual intserv-diffserv-insterv segment along the path. However, obtaining end-to-end delay bound for a concatenation of Diffserv clouds via nodes implementing aggregate scheduling is a more complicated problem which requires further research.

## 5. Implementation of Resource Efficient Close Approximations to the Guaranteed Service

<TBA>

## 6. Relationship to the Null Service

The Intserv "Null Service" [NULL] differs from other defined services by

not expressing any quantitative network performance requirements. Use of the Null Service where an Intserv service class is required allows an application or host requesting QoS control service to express policy related information to the network without making a specific





quantitative QoS request. The assumption is that the network policy management and control elements will use this information to select an appropriate QoS for the requesting entity, and take whatever action is required to provide this QoS.

One possibility is that the network policy mechanisms will determine that a quantitative end-to-end QoS is appropriate for this entity, and that this QoS can be provided using Intserv mechanisms. In this case, the Null service selector can be replaced, at the first hop router or elsewhere along the path, with a different Intserv service class and related parameter information. Once this occurs, the situation with respect to the use of Diffserv networks to provide the desired QoS is identical to that described above for these other services.

A second alternative is that the network policy mechanisms determine that the requesting entity should receive a relative, rather than absolute (quantitative) level of service. In this case, the packets are marked with the appropriate DSCP, but the admission control actions described above are not necessary.

## 7. Security Considerations

<None yet>

## 8. References

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

[CHARNY] Anna Charny, "Delay Bounds in a Network with Aggregate Scehduling", work in progress, ftpeng.cisco.com/ftp/acharny/aggregate_delay_v4.ps

[CL] Wroclawski, J., "Specification of the Controlled-Load Network Element Service", RFC 2211, September 1997

[DCLASS] Bernet, Y., "Format of the RSVP DCLASS Object", RFC 2996, November 2000

[DIFFSERV] Blake, S., Black, D., Carlson, M., Davies, E., Wang Z., Weiss, W., "An Architecture for Differentiated Service", RFC 2475, December 1998.

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

[G] Schenker, S., Partridge, C., Guerin, R., "Specification of Guaranteed Quality of Service", RFC 2212 September 1997

[GENCHAR] Shenker, S., Wroclawski, J., "General Characterization Parameters for Integrated Service Network Elements", RFC 2215, September 1997





[INTSERV] Clark, D. et al. "Integrated Services in the Internet Architecture: an Overview" RFC1633, June 1994

[ISDSFRAME] Bernet, Ford, Yavatkar, Baker, Zhang, Speer, Braden, Davie, Wroclawski, Felstaine, "A Framework for Integrated Services Operation over Diffserv Networks", RFC 2998, November 2000

[LEBOUDEC] Jean-Yves LeBoudec, "A Proven Delay Bound in a Network with Aggregate Scheduling", work in progress, http://ica1www.epfl.ch/PS_files/ds2.ps

[NULL] Bernet, Y., Smith, A., Davie, B., "Specification of the Null Service Type", RFC 2297, November 2000

[RSVP] Braden, R., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource Reservation Protocol (RSVP) - Version 1 Functional Specification", RFC 2205, September 1997

[RSVPAGGR] Baker, F., Iturralde, C., Le Faucheur, F., Davie, B., "Aggregation of RSVP for IPv4 and IPv6 Reservations", Internet Draft draft-ietf-issll-rsvp-aggr-02.txt

[RSVPINTSERV] Wroclawski, J., "The use of RSVP with IETF Integrated Services", RFC 2210, September 1997.

## 9. Authors' addresses

John Wroclawski
MIT Laboratory for Computer Science
545 Technology Sq., Cambridge, MA 02139, USA
EMail: jtw@lcs.mit.edu

Anna Charny
Cisco Systems
300 Apollo Drive, Chelmsford, MA 01824, USA
Email: acharny@cisco.com

## 10. Full Copyright