Internet Engineering Task Force
INTERNET-DRAFT
draft-clark-diff-svc-alloc-00.txt

D. Clark / J. Wroclawski
MIT LCS

July, 1997
Expires: 12/97

An Approach to Service Allocation in the Internet

Abstract

This note describes the Service Allocation Profile scheme for differential service allocation within the Internet. The scheme is based on a simple packet drop preference mechanism at interior nodes, and highly flexible service profiles at edges and inter-provider boundary points within the net. The service profiles capture a wide variety of user requirements and expectations, and allow different users to receive different levels of service from the network in a scalable and efficient manner.

The note describes the basic form of the mechanism, discusses the range of services that users and providers can obtain by employing it, and gives a more detailed presentation of particular technical, deployment, standardization, and economic issues related to its use.

Status of this Memo

submission deadline for the Munich IETF. The authors apologize in advance for a certain raggedness of presentation..

1. Introduction

This document describes a framework for providing what has been called differentiated service, or allocated capacity service, in the Internet. The goal of the mechanism is to allocate the bandwidth of the Internet to different users in a controlled way during periods of congestion. The mechanism applies equally to traditional applications based on TCP, such as file transfer, data base access or Web servers, and new sorts of applications such as real time audio or video.

The mechanism we describe can provide users with a predictable expectation of what service the Internet will provide to them in times of congestion, and can allow different users to obtain different levels of service from the network. This contrasts with today's Internet, in which each user gets some unpredictable share of the capacity.

Our mechanism provides two additional things that are important to this task. First, it allows users and providers with a wide range of business and administrative models to make capacity allocation decisions. In the public Internet, where commercial providers offer service for payment, the feedback will most often be different prices charged to customers with different requirements. This allows the providers to charge differential prices to users that attach greater value to their Internet access, and thus fund the deployment of additional resources to better serve them. But whether pricing, or some other administrative control is used (as might apply in a corporate or military network), the same mechanism for allocating capacity can be used.

The mechanism also provides useful information to providers about provisioning requirements. With our mechanism in place, service providers can more easily allocate specific levels of 3assured2 capacity to customers, and can easily monitor their networks to detect when the actual service needs of their customers are not being met.

While this document does describe a mechanism, this is a small part of its goal. There are a number of mechanisms that might be proposed, and the issue is not just demonstrating which of them works (most do work in some fashion), but to discuss what the problem to be solved actually is, and therefore which of the possible mechanisms best meets the needs of the Internet. This document is thus as much about what the problem actually is, as it is about a preferred solution.

1.1 Separating results from mechanism

An essential aspect of this scheme is the range of services the user can obtain using the mechanism. The mechanism is obviously not useful if it does not meet a current need. Some initial requirements we see for services are that they must be useful, easy to understand, possible to measure (so the user can tell whether he is getting the service he contracted for), and easy to implement.

At the same time, we should try very hard not to embed a specific set of services into the core of the Internet. As we gain experience in the marketplace, we may discover that our first speculations are wrong about what service the user actually wants. It should be possible to change the model, evolve it, and indeed to try different models at the same time to see which better meets the needs of the user and the market. So this scheme has the two goals: defining and implementing a first set of services, but permitting these services to be changed without modifying the "insides" of the Internet, specifically the routers.

We will return later to the discussion of different sorts of services.

2. Outline of this document

This document is organized as follows:

Section 3 describes the basic mechanism, to give a general idea of how such a service can be implemented.

Section 4 discusses the services which might be desired. It proposes a first set of services that might be implemented, and discusses the range of services that can be built out of this mechanism.

Section 5 describes the location of service profiles in the network.

Section 6 describes details of the mechanism. These include our specific algorithm for the dropper, issues concerning rate control of TCP, and dealing with non-responsive flows.

Section 7 compares this mechanism with some alternatives.

Section 8 discusses deployment issues, incremental deployment, and what portions of the mechanism require standardization.

Section 9 discusses security issues.

3. The basic scheme

The general approach of this mechanism is to define a service profile for each user, and to design a mechanism in the router that favors traffic that is within those service profiles. The core of the idea is very simple -- monitor the traffic of each user as it enters the network, and tag packets as being "in" or "out" of their service profile. Then at each router, if congestion occurs, preferentially drop traffic that is tagged as being "out".

Inside the network, at the routers, there is no separation of traffic from different users into different flows or queues. The packets of all the users are aggregated into one queue, just as they are today. Different users can have very different profiles, which will result in different users having different quantities of "in" packets in the service queue. A router can treat these packets as a single commingled pool. This attribute of the scheme makes it very easy to implement, in contrast to a scheme like current RSVP reservations, in which the packets must be explicitly classified at each node. We have more to say about this issue below.

To implement this scheme, the routers must be augmented to implement our dropping scheme, and a new function must be implemented to tag the traffic according to its service profile. This algorithm can be implemented as part of an existing network component (host, access device or router) or in a new component created for the purpose. Conceptually, we will refer to it as a distinct device called a "profile meter". We use the term "meter" rather than "tagger", because, as we will discuss below, the profile meter can actually take a more general set of actions.

The idea of a service profile can be applied at any point in the network where a customer-provider relationship exists. A profile may describe the needs of a specific user within a campus, the service purchased by a corporate customer from an ISP, or the traffic handling agreement between two international providers. We discuss the location of profiles further in Section 5.

The description above associates the profile with the traffic sender. That is, the sender has a service profile, the traffic is tagged at the source according to that profile, and then dropped if necessary inside the network. In some circumstances, however, it is the receiving user that wishes to control the level of service. The web provides a simple example; a customer using his browser for business research may be much more interested in a predictable level of performance than the casual surfer. The key observation is that "value" in the network does not always flow in the same direction as the data packets.

Thus, for full generality, a "dual" mechanism is required, that can

be either "sender driven" or "receiver driven". Most of this document is written, for simplicity, in terms of a sender scheme, but we briefly describe the receiver version as well, and discuss the circumstances in which it is important. In evaluating alternative mechanisms, it is important to see if both a sender and receiver version can be built.

In later sections, we discuss the specifics of the profiling or tagging mechanism and the treatment of profiled packets within the network. First we turn to the question of the range of services the mechanism ought to support.

4. Range of services

As discussed above, there are two general issues concerning service models. First, we want to start out by implementing a simple set of services, which are useful and easy to understand. At the same time, we should not embed these services into the mechanism, but should build a general mechanism that allows us to change the services as our experience matures.

Our scheme provides this flexibility. To oversimplify, a service is defined by the profile meter, which implements the user's service profile. To change the service, it is necessary "only" to change the profile meter. The routers in the interior of the network implement a single common mechanism which is used by the different meters to provide different services.

Three things must be considered when describing a service:

- what exactly is provided to the customer (an example might be "one megabit per second of bandwidth, continuously available")

- to where this service is provided (examples might be a specific destination, a group of destinations, all nodes on the local provider, or "everywhere")

- with what level of assurance is the service provided (or alternately, what level of performance uncertainty can the user tolerate)

These things are coupled; it is much easier to provide "a guaranteed one megabit per second" to a specific destination than to anywhere in the Internet.

4.1 A first service model

As a place to start, a particularly simple service might provide the equivalent of a dedicated link of some specified bandwidth from source to destination. (The virtue of this simple model has been

clearly articulated by Van Jacobson.) This model is easy for the user to understand -- he can take his existing application, connect it across a physical link and see how it performs. If he can make it work in that context, then this service will allow him to run that application over the Internet.

This model has been implemented in a number of network architectures, with different "enhancements". The CBR service of ATM is an example, as is (to some extent) the CIR mechanism of Frame Relay. However, there are some issues and limitations to this very simple model.

One very important limit to a pure virtual link model is that the user may not wish to purchase this virtual link full time. He may need it only some of the time, and in exchange would hope to obtain a lower cost. A provider could meet this desire by offering a more expressive profile; say a committed bandwidth with some duty cycle, e.g. "3 mb/s with a 5% duty cycle measured over 5 minutes". Or, the provider could offer the user a rebate based on observed (non)usage, or allow him to reserve the capacity dynamically on demand.

A second issue is whether the user can exceed the capacity of the virtual link when the network is unloaded. Today, the Internet allows its users to go faster under that circumstance. Continuing to capture that benefit may be important in user acceptance. The CIR of Frame Relay works this way, and it is an important aspect of its market appeal.

An equally important issue is that the user may not wish to set up different distinguished committed bandwidth flows to different destinations, but may prefer to have a more aggregated commitment. There are several drawbacks to making distinct bandwidth commitments between each source and destination. First, this may result in a large number of flow specifications. If the user is interested in 1000 network access points, he must specify one million source-destination pairs. Frame Relay has this specification problem. Second, the sum of the distinct commitments for any source (or destination) cannot exceed the physical capacity of the access link at that point, which may force each individual assurance to be rather small. Finally, the source-destination model implies that the user can determine his destinations in advance, and in some cases that he understands the network topology; two situations which are not universally true.

In fact, several variations of service commitment might make sense to different users; from one source to a specific destination, from a source to a pool of specified destinations (one might configure a Virtual Private Network in this way) and finally from a source to "anywhere", which could mean either all points on the ISP, on a

collection of ISPs, or any reachable node.

The latter sorts of commitments are by their nature more difficult to offer with high assurance. There is no way to know for sure what the service will be to any specific destination, because that depends on what other traffic is leaving the source, and what other traffic is arriving at the destination. Offering commitments to "anywhere within the ISP" implies that the ISP has provisioned its resources adequately to support all in-profile users simultaneously to the same destination. Offering commitments to "anywhere at all" implies that all ISPs in any reachable path from the user have provisioned sufficiently, which is most unlikely.

4.2 Managing bursty traffic

Not all Internet traffic is continuous in its requirement for bandwidth. Indeed, based on measurements on the Internet, much of the traffic is very bursty. It may thus be that a service model based on a fixed capacity "virtual link" does not actually meet user's needs very well. Some other more complex service profile that permits bursty traffic may be more suitable.

It is possible to support bursty traffic by changing the profile meter to implement this new sort of service. The key issue is to insure, in the center of the network, that there is enough capacity to carry this bursty traffic, and thus actually meet the commitments implied by the outstanding profiles. This requires a more sophisticated provisioning strategy than the simple "add 'em up" needed for constant bit-rate virtual links. A body of mathematics that is now maturing provides a way to relate the bursty behavior of a single flow to the resulting implications for the required overall bandwidth when a number of such flows are combined. (see, for example [Kelly97]). This sort of analysis can be employed as a way to predict the capacity that must be provided to support profiles describing bursty traffic. As a practical matter, in the center of the existing Internet, at the backbone routers of the major ISPs, there is such a high degree of traffic aggregation that the bursty nature of individual traffic flows is essentially invisible. So providing bursty service profiles to individual users will not create a substantial provisioning issue in the center of the network, while possibly adding significant value to the service as perceived by the user.

4.3 Degrees of assurance

The next aspect of sorting out services is to consider the degree of assurance that the user will receive that the contracted capacity will actually be there when he attempts to use it.

Statistical bandwidth allocation allows the Internet to support an increased number of users, use bandwidth vastly more efficiently, and deal flexibly with new applications and services. However, it does lead to some uncertainty as to the bandwidth that will be available at any instant. Our approach to allocating traffic is to follow this philosophy to the degree that the user can tolerate the uncertainty. In other words, we believe that a capacity allocation scheme should provide a range of service assurance. At one extreme, the user may demand an absolute service assurance, even in the face of some number of network failures. (Wall Street traders often have two phones on their desk, connected by different building wiring to different phone exchanges, so that they can continue to make money even if a central office goes down or half the building burns.) Less demanding users may wish to purchase a service profile that is "usually" available, but may still fail with low probability. The presumption is that a higher assurance service will cost substantially more to implement.

We have called these statistically provisioned service profiles "expected capacity" profiles. This term was picked to suggest that the profiles do not describe a strict guarantee, but rather an expectation that the user can have about the service he will receive during times of congestion. This sort of service will somewhat resemble the Internet of today in that users today have some expectation of what performance they will receive; the key change is that our mechanism by which different users can have very different expectations.

Statistical assurance is a matter of provisioning. In our scenario, an ISP can track the amount of traffic tagged as "in" crossing various links over time, and provide enough capacity to carry this subset of the traffic, even at times of congestion. This is how the Internet is managed today, but the addition of tags gives the ISP a better handle on how much of the traffic at any instant is "valued" traffic, and how much is discretionary or opportunistic traffic for which a more relaxed attitude can be tolerated.

For traffic that requires a higher level of commitment, more explicit actions must be taken. The specific sources and destinations must be determined, and then the paths between these points must be inspected to determine if there is sufficient capacity. There are two approaches. The static approach involves making a long term commitment to the user, and reserving the network resources to match this commitment. This involves some computation based on the topology map of the network to allocate the needed bandwidth along the primary (and perhaps secondary) routes. The dynamic approach involves using a setup or reservation protocol such as RSVP to request the service. This would explore the network path at the time of the request, and reserve the bandwidth from a pool available for

dynamic services. Information concerning this pool would have to be stored in the routers themselves, to support the operation of RSVP. We have proposed a lightweight version of RSVP, called RSVP with Trusted TOS Tags, or T3, as a way to implement this dynamic service efficiently. Within one ISP, the reservation could be submitted to a central location for acceptance, depending on the design adopted for bandwidth management.

It is important to note that traffic requiring this higher level of assurance can still be aggregated with other similar traffic. It is not necessary to separate out each individual flow to insure that it receives it promised service. It is necessary only to insure that sufficient capacity is available between the specific sources and destinations desiring the service, and that the high-assurance packets can draw on that capacity. This implies that there would be two queues in the router, one for traffic that has received a statistical assurance, and one for this higher or "guaranteed" assurance. Within each queue, "in" and "out" tags would be used to distinguish the subset of the traffic that is to receive the preferred treatment. However, some other discriminator must be used to separate the two classes, and thus sort packets into the two queues. Our specific proposal, which we detail later, is that two values of the TOS field be used, one to mean statistical assurance, and one to mean guaranteed assurance. Statistical assurance would correspond to the service delivered across the Internet today, augmented with "in" and "out" tags.

An ISP could avoid the complexity of multiple queues and still provide the high-assurance service by over-provisioning the network to the point where all "in" traffic is essentially never dropped, no matter what usage patterns the users generate. It is an engineering decision of the ISP whether this approach is feasible.

4.4 A service profile for the access path

In some cases, what the user is concerned with is not the end-to-end behavior he achieves, but the profile for utilizing his access path to the network. For example, users today buy a high-speed access path for two different reasons. One is to transfer a continuous flow of traffic, the other to transfer bursts at high speed. The user who has bursty traffic might want on the one hand an assurance that the bursts can go through at some known speed, but on the other hand a lower price than the user who delivers a continuous flow into the Internet. Giving these two sorts of users different service profiles that describe the aggregated traffic across the access link will help discriminate between them, and provide a basis for differential charging.

A service profile of the sort discussed here is a reasonable way to capture this sort of requirement. By tagging the traffic that crosses the access path according to some service profile, the ISP commits to forward that subset of the traffic within its region, and only delivers the rest if the network is underloaded. It is instructive to compare this approach to pricing an access path to the more traditional "usage-based" scheme. In the traditional scheme, the actual usage is metered, and the user is charged a fee that depends on the usage. If the user sends more, he pays more. However, since TCP goes faster if the net is underloaded, it is hard for the user (or the ISP aggregating his traffic) to actually regulate his usage. In contrast, a service profile allows two users with different needs to be distinguished (and presumably charged differently) but each user could be charged a known price based on the profile. If the traffic exceeds the profile, the consequence is not increased fees, but congestion pushback if the network is congested.

4.5 An example of a more sophisticated profile

Our initial service profile modeled a dedicated link of some set capacity. This service profile is easy to understand at one level, but once one runs TCP over this link, it becomes much harder to predict what behavior can actually be achieved. TCP hunts for the correct rate by increasing its window size until a packet is discarded at the bottleneck point, and then cutting its window size by two (in many current implementations). How this behavior interacts with a link of fixed size is a function of buffer size and implementation details in TCP.

A more sophisticated service profile would be one that attempted to provide a specified and predictable throughput to a TCP stream, so long as the TCP was "well-behaved". This would actually make it easier for the user to test the profile, because he could just run a TCP-based application and observe the throughput. This is an example of a "higher-level" profile, because it provides a service which is less closely related to some existing network component and more closely related to the user's actual needs. These profiles are more difficult to define, because they depend on the behavior of both the network and the end-nodes. However, we have experimented with the design of such a profile, and believe that it is possible to implement this sort of service as well. A more detailed description of the profile needed to fix a TCP transfer rate is given in Appendix B.

5. Location of Service Profiles in the Network

In the simple sender-based scheme described so far, the function that checks whether traffic fits within a profile is implemented by

tagging packets as in or out of profile at the edge of the network. The complete story is more complex. A profile describes an expectation of service obtained by a customer from a provider. These relationships exist at many points in the network, ranging from individual users and their campus LANs to the peering relationships between global ISP's. Any such boundary may be an appropriate place for a profile meter.

Further, the packet tagging associated with this service profile will, in the general case, be performed by devices at either side of the boundary. One sort, located on the traffic sourcing side of a network boundary, is a "policy meter". This sort implements some policy by choosing the packets that leave the network (or user's machine) with their in-profile bit set, and thus receive the assured service. Another sort, a "checking meter", sits on the arriving-traffic side of a network boundary, checks the incoming traffic, and marks packets as out of profile (or turns off excess in-profile bits) if the arriving traffic exceeds the assigned profile.

A general model is that the first meter that the traffic encounters should provide the highest degree of discrimination among the flows. A profile meter could be integrated into a host implementation of TCP and IP, where it could serve to regulate the relative use of the network by individual flows. The subsequent meters, looking only at larger aggregates, serve to verify that there is a large enough overall service contract in place at that point to carry all of the traffic tagged as "in" (the valuable traffic) at the interior points. When a traffic meter is placed at the point where a campus or corporate network connects to an ISP, or one ISP connects to another, the traffic being passed across the link is highly aggregated. The ISP, on the arriving- traffic side of the link, will check only to verify the total behavior. On the traffic sourcing side of the link, an additional profile meter can be installed to verify that tags have been applied according to policy of the user.

Additional profile meters installed at intermediate points can provide good feedback on network demand. Consider a specific situation, where traffic is tagged at individual hosts according to policies specific to these hosts, and then passes through a second meter at the point of attachment from the private network to the public Internet. If the number of "in" packets arriving at that point exceeds the aggregate service profile purchased at that point, this means that the user has not purchased enough aggregate capacity to match the needs of his individual policy setting. In the short run, there is no choice but to turn some of these "in" packets to "out", (or to charge an extra fee for carrying unexpected overloads), but in the long run, this provides a basis to negotiate a higher service level with the ISP. So traffic meters actually provide a basis for

monitoring user needs, and moving users to a higher service profile as needed.

5.1 Controlling the scope of profiles

Even in the case where the user wants to obtain a service profile that is not specific to one destination, but rather applies to "all" possible destinations, it is clear that the "all" cannot be literally true. Any service profile that involves an unspecified set of destinations will have to bound the scope of the agreement. For example, a single ISP or a set of co-operating ISPs may agree to provide an assured service profile among all of their end points, but if the traffic passes beyond that point, the profile will cease to apply.

The user might be given further options in the design of his profile. For example, if there are regions of restricted bandwidth within the Internet, some users may wish to pay more in order to have their "in" tags define their service across this part of the net, while others may be willing to have their "in" tags reset if the traffic reaches this point.

This could be implemented by installing a profile meter at that point in the network, with explicit lists of source-destination pairs that are and are not allowed to send "in" traffic beyond this point. The alternative would be some sort of "zone system" for service profiles that is specified in the packets themselves. See [Clark97] for a discussion of a zone system.

6. Details of the Mechanism

This section describes several aspects of our proposed mechanism in more detail.

6.1 Design of the dropper

One of the key parts of this scheme is the algorithm in the router that drops "out" packets preferentially during times of congestion. The behavior of this algorithm must be well understood and agreed on, because the taggers at the edge of the network must take this behavior into account in their design. There can be many taggers, with different goals as to the service profile, the degree of aggregation etc. There is only one dropper, and all the routers have to perform an agreed behavior.

The essence of our dropper is an algorithm which processes all packets in order as received, in a single queue, but preferentially drops "out" packets. There are other designs that could be proposed

for queue management, for example to put the "in" packets in a higher priority queue. There are specific reasons why we prefer drop preference to priority queuing for the allocation of best effort traffic, but we delay that discussion until Section 7.

The primary design goals of the dropper are the following:

- It must allow the taggers to implement a range of service profiles in a useful and understandable way.

- If the router is flooded with "out" packets, it must be able to discard them all without harming the "in" packets. In other words, it must deal well with non-conforming flows that do not adjust their sending rate when they observe packet loss.

- If the router is receiving a number of "well-behaved" TCP flows, which will (as TCP always does) speed up until they encounter a lost packet, it must have enough real buffering available that once it starts to get overloaded with packets, it can discard "out" packets and still receive traffic bursts for a round trip until the affected TCP slows down.

6.2 RIO -- RED with In and Out

Our specific dropping scheme is an extension of the Random Early Detection scheme, or RED, that is now being deployed in the Internet. The general behavior of RED is that, as the queue begins to build up, it drops packets with a low but increasing probability, instead of waiting until the queue is full and then dropping all arriving packets. This results in better overall behavior, shorter queues, and lower drop rates.

Our approach is to run two RED algorithms at the same time, one for "in" packets, and one for "out" packets. The "out" RED algorithm starts dropping at a much shorter average queue length, and drops much more aggressively than the "in" algorithm. With proper setting of the parameters, the "out" traffic can be controlled before the queue grows to the point that any "in" traffic is dropped. We call this scheme RIO.

There are some subtle aspects to this scheme. The "in" dropper must look at the number of "in" packets in the queue. The "out" dropper must look at the total queue length, taking into account both "in" and "out". This is because the link can be subjected to a range of overloads, from a mix of "in" and "out" traffic to just "out". In both cases, the router must start dropping "outs" before the "in" traffic is affected, and must continue to achieve the basic function of RED; preserving enough free buffer space to absorb transient loads with a duration too short to be affected by feedback congestion

control.

6.3. Rate control of TCP

A useful, but challenging, problem is to build a traffic meter that causes a TCP to send at a specified maximum rate in periods of congestion. Such a meter works by causing the TCP's bandwidth usage (actually congestion avoidance) algorithm to "hunt" between somewhat over and somewhat under the target rate, by tagging packets such that the RIO algorithm will drop them appropriately when the network is loaded. An important aspect of this is that the meter and RIO work together to avoid *too many* closely spaced packet discards, forcing the TCP into slow-start and causing it to obtain less than the desired bandwidth.

A detailed description of a traffic meter which meets these objectives is given in Appendix B of this note.

6.4. Dealing with non-responsive flows

A well-behaved TCP, or other traffic source that responds similarly to congestion signaled by packet loss, will respond well to the RIO dropper. As more of its packets are marked as "out", one will eventually be dropped. At this point, the source will back off. As a result, most of the time a network of well-behaved TCPs will contain just enough "out" packets to use up any excess capacity not claimed by the "in" packets being sent.

But what if there is a source of packets that does not adjust? This could happen because of a poorly implemented TCP, or from a source of packets, such as a video data application, that does not or cannot adjust.

In this circumstance, if the unresponsive flow1s packets are marked as out of profile, the flood of "out" packets will cause a RIO router to operate in a different way, but well behaved TCPs and similar flows must continue to receive good service. (If the unresponsive flow1s packets are in profile, the network should be able carry them, and there is no issue.)

6.4.1. Robustness against non-responsive flows

In the RIO scheme, once the level of "out" packets exceeds a certain average level, all the incoming "out" packets will be discarded (this is similar to the non-RIO RED behavior). This behavior has the consequence of increasing the router1s queue length. The average queue length will increase by the number of "out" packets that are allowed to sit in the queue before RIO switches over to the phase

where it drops every "out". There must be enough physical room in the buffer so that even when there are this many "out" packets present, there is enough room for the normal instantaneous bursts of "in" packets which would be seen in any event. Thus, a RIO router may require slightly larger queues than a non-RIO router.

In the simulations summarized in Appendix B, the maximum number of "out" packets is approximately 15. (This particular number is not magic -- the point is that it is not 1, nor 100.) So to operate RIO, it will be necessary to increase the minimum physical buffer size by perhaps this amount, or a little more, to allow for swings in the instantaneous numbers of "out" packets as well. But in most circumstances, this is a modest increase in the buffer size.

6.4.2. Filtering out non-responsive flows

Although RIO is reasonably robust against overload from non-responsive flows, it may be useful to consider the alternative strategy of singling out non-conforming flows and selectively dropping them in the congested router. There has been work [FF97] towards enhancing the traditional RED scheme with a mechanism to detect and discriminate against non-conforming flows. Discriminating against these flows requires the installation of a packet classifier or filter that can select these packet flows, so that they can be discarded. This adds complexity and introduces scaling concerns to the scheme. These concerns are somewhat mitigated because only the misbehaving flows, not the majority of flows that behave, need be recognized. Whatever classification scheme that basic RED might use can be used by RIO as well.

The difference between our framework and RED is that the designers of RED are working to design an algorithm that runs locally in each router to detect non-conforming flows, without any concept of a service profile. In that case, the only sort of traffic allocation that can be done is some form of local fairness. However, with the addition of profile tags, the router can look only at the "out" packets, which by definition represent that portion of a flow that is in excess. This might make it easier to detect locally flows that were non- conforming. The alternative approach would be an indication from the traffic meter that the flow is persistently exceeding the service profile in a time of congestion. This indication, a control packet, could either install a classifier in each potential point of congestion, or flow all the way back to the traffic meter nearest the sender, where the traffic can be distinguished and discarded (or otherwise discriminated against). The latter approach has the benefit that the control packet need not follow the detailed hop-by-hop route of the data packet in reverse, which is hard to do in today's Internet with asymmetric routes.

We consider the question of whether such a mechanism provides sufficient benefit over the approach of employing local detection of non-responsive flows at each node to be unresolved at present.

7. Alternatives to the mechanism

Schemes for differential service or capacity allocation differ in a number of respects. Some standardize on the service profiles, and embed them directly in the routers. As discussed above, this scheme has the advantage that the actual service profile is not a part of what is standardized, but is instead realized locally in the traffic meter, which gives this scheme much greater flexibility in changing the profile.

7.1. Drop preference vs. priority

One possible difference is what the router does when it is presented with an overload. Our scheme is based on a specific algorithm for drop preference for packets marked as "out". An alternative would be to put packets marked as "out" in a lower priority queue. Under overload that lower priority queue would be subjected to service starvation, queue overflow and eventually packet drops. Thus a priority scheme might be seen as similar to a drop preference scheme.

They are similar, but not the same. The priority scheme has the consequence that packets in the two queues are reordered by the scheduling discipline that implements the priority behavior. If packets from a single TCP flow are metered such that some are marked as "in" and some as "out", they will in general arrive at the receiver out of order, which will cause performance problems with the TCP. In contrast, the RIO scheme always keeps the packets in order, and just explicitly drops some of the "out" packets if necessary. That makes TCP work much better under slight overload.

The priority scheme is often proposed for the case of a restricted class of service profiles in which all the packets of a single flow are either "in" or "out". These schemes include the concept of a "premium" customer (all its packets are "in"), or a rate-limited flow (packets that exceed the service profile are dropped at the meter, rather than being passed on.) These proposals are valid experiments in what a service profile should be, but they are not the only possibilities. The drop preference scheme has the advantage that it seems to support a wider range of potential service profiles (including the above two), and thus offers an important level of flexibility.

7.2. More bits?

A variation on this scheme is to have more than two levels of control -- more than simple "in" and "out". One reason to have more than two levels is to allow the user to express his range of values more completely. With three or four levels of tagging, the user could express what service profile he would like at different levels of congestion -- none, low, medium and severe. The question is whether this actually brings much real incremental value. In commercial networks, which are usually provisioned in a conservative fashion, it is not clear that there will be enough congestion to discriminate between more than two states. In other circumstances, for example military networks where severe service degradation might occur under adverse circumstances, having several levels of usage preference might be helpful. Asking the user to define these several tiers of service profiles raises one issue, however; it presumes that the user is actually able to determine what his needs are to this degree of precision. It is not actually clear that the user has this level of understanding of how he would trade off usage against cost.

There is an alternative way to deal with variation in the degree of congestion. Instead of coding the user's desires into each packet, one could imagine a management protocol running in the background that reports to the edges of the network what the current level of congestion is, or whether a special or crisis circumstance exists. Based on information from that protocol, the service profile of the user could be changed. Both approaches may have advantages. An advantage of the first is the lack of need for a management protocol. An advantage of the second is that the management protocol can express a much wider range of policies and reallocation actions.

Another reason to have multiple levels of control is to achieve a smoother transition between the two states of a flow. As discussed above, when controlling TCP, because of the specific congestion schemes used in TCP, it is helpful not to drop a number of packets from one flow at once, because it is likely to trigger a full TCP slow- start, rather then the preferable fast recovery action. Having more bits might enhance this discrimination. However, based on our simulations, if we are going to use more bits from the packet header for control, it might be a better option to move to an Explicit Congestion Notification design for the Internet, which seems to provide a better degree of control overall.

8. Deployment Issues

8.1. Incremental deployment plan.

No scheme like this can be deployed at once in all parts of the Internet. It must be possible to install it incrementally, if it is to succeed at all.

The obvious path is to provide these capabilities first within a single ISP. This implies installing RIO routers within the ISP, and tagging the traffic at the access points to that ISP. This requires a profile meter at each access link into that ISP. The meter could maintain a large amount of user-specific information about desired usage patterns between specific sources and destinations (and this might represent a business opportunity), but more likely would provide only rough categories of traffic classification.

A user of this ISP could then install a profile meter on his end of the access link, which he controls and configures, to provide a finer- grained set of controls over which traffic is to be marked as "in" and "out". Eventually, meters might appear as part of host implementations, which would permit the construction of profiles that took into account the behavior of specific applications, and which would also control the use of network resources within the campus or corporate area.

At the boundary to the region of routers implementing RIO, all traffic must be checked, to make sure that no un-metered traffic sneaks into the network tagged as "in". So the implementation of this scheme requires a consistent engineering of the network configuration within an administrative region (such as an ISP) to make sure that all sources of traffic have been identified, and either metered or "turned out".

If some routers implement RIO, and some do not, but just implement simple RED, the user may fail to receive the committed service profile. But no other major failures will occur. That is, the worst that the user will see is what he sees today. One can achieve substantial incremental improvements by identifying points of actual congestion, and putting RIO routers there first.

8.2. What has to be standardized

In fact, very little of this scheme needs to be standardized in the normal pattern of IETF undertakings. What is required is to agree on the general approach, and set a few specific standards.

8.2.1. Semantics of router behavior

It is necessary to agree on the common semantics that all routers will display for "in" and "out" bits. Our proposal is that routers implement the RIO scheme, as described above. The parameters should be left for operational adjustment.

For the receiver-based scheme, the router has to tag packets rather than drop them. We omit the description of the tagging algorithm,

only noting that it, too, must be agreed to if a receiver-based scheme is to be deployed.

8.2.2. Use of IP precedence field

Currently, the three bit precedence field in the IP header is not widely used. Bit x of this field will be used as the "in/out" bit. This bit will be known as the In Profile Indicator, or IPI. The meaning of the IPI is that a 1 value implies "in". This has the effect that the normal default value of the field, 0, will map to the baseline behavior, which is out of profile service.

8.2.3. Use of IP TOS field

This document proposes to view Type of Service in a slightly different way than has been usual in the past. While previous RFCs have not been explicit (e.g. RFC 1349), the role of the ToS field has been thought of more to control routing than scheduling and dropping within the router. This document explicitly proposes to specify these features. The TOS field can be used for this purpose, but doing so will preclude its use in the same packet to select the service defined in RFC 1349 and RFC 1700: low delay, high throughput, low cost, high reliability and high security.

According to RFC 1349, the TOS field should be viewed as a 4 bit integer value, with certain values reserved for backwards compatibility. We propose that the six defined values of TOS be associated with the statistical service profiles ("expected capacity profiles") defined in this document. That is, the use of the IPI is legal with any of these value of TOS, and the difference among them is routing options.

A new value of TOS (yyyy) shall be used to specify the assured service profile, which has a level of assurance for the service profile that is not statistical in nature. As part of the design of this type of service, the routing will have to be controlled to achieve this goal, so the value yyyy for the TOS will also imply some routing constraints for the ISPs. It is an engineering decision of the service provider how this sort of traffic is routed, so that it follows the routes along which the resources have been reserved.

8.2.4. Additional issues for the sender/receiver based scheme

The combined sender-receiver scheme is capable of expressing a much more complex set of value relationships than the sender-based scheme. However, it implies more complexity and more bits in the header. It does not appear possible to encode all the necessary information for the combined scheme in an IPv4 header. This option is thus proposed

as a consideration for IPv6, if there seems to be sufficient demand.

9. Security considerations

This scheme is concerned with resource allocation, and thus the major security concern is theft of resources. Resources can be stolen by injecting traffic that is marked as "in" but which has not passed through a legitimate profile meter into a RIO-controlled region of the network.

To protect against this, it is necessary to define "regions of shared trust", and engineer and audit all the links that bring traffic into each of these regions to insure that a profile meter has been installed in each such link. Such a region might correspond to a single ISP, the backbone component of a single ISP, a collection of co-operating ISPs and so on. In general, the presence of a profile meter is an indication of a possible boundary where trust is not shared, and the traffic has to be verified.

It is a matter for further research whether algorithms can be designed to detect (locally, at each router) a flow of packets that is not legitimate.

10. Acknowledgments

The simulations reported in this paper were performed by Wenjia Fang. Earlier simulations that proved the concepts of the profile meter and the receiver-based scheme were performed by Pedro Zayas. We acknowledge the valuable discussions with the members of the End-to-End research group.

Appendix A: Description of a receiver-based scheme

The tagging scheme described above implements a model in which the sender, by selecting one or another service profile, determines what service will govern each transfer. However, the sender- controlled model is not the only appropriate model for determining how Internet transmissions should be regulated. For much of the traditional Internet, where information has been made available, often for free, to those users who care enough to retrieve it, it is the value that the receiver places on the transfer, not the sender, that would properly dictate the service allocated to the transfer. In this document, we do not debate the philosophical tradeoff between sender and receiver controlled schemes. Instead, we describe a mechanism that implements receiver control of service, which is similar in approach and meshes with the sender controlled tagging scheme.

One technique that does not work is to have the receiver send some

credentials to the sender, on the basis of which a flag is set in the packet. This runs the risks of great complexity, but more fundamentally does not deal with multicast, where one packet may go to several receivers, each of which attaches a different value to the transfer.

A critical design decision is whether the scheme must react to congestion instantly, or with one round trip delay. If it must react instantly, then each point of congestion must have stored state, installed by the receiver, that will allow that point to determine if the packet is "in" or "out" of profile. This runs the risk of formidable complexity. If, however, we are willing to have the reaction to congestion occur one round trip later, several quite tractable schemes can be proposed, which are similar to the sender controlled scheme in spirit.

A receiver controlled scheme can be built using a traffic meter at the receiver, similar to the traffic meter at the sender in the sender tagging scheme. The meter knows what the current usage profile of the receiver is, and thus can check to see whether a stream of received packets is inside of the profile. A (different) new flag in the packet, called Forward Congestion Notification, or FCN, is used to carry information about congestion to the receiver's traffic meter. A packet under this receiver controlled scheme starts out from the sender with the FCN bit off, and when the packet encounters congestion the bit is set on. As the packet reaches the destination, the receiver's traffic meter notes that the bit is on, and checks to see if the packet fits within the profile of the receiver. If it does, the service profile of the receiver is debited, and the bit is turned off in the packet. If the packet cannot fit within the profile of the user, the bit remains on.

When the receiver receives a packet with the FCN on, which means that the receiver's profile does not have sufficient capacity to cover all the packets that encountered congestion, the sender must be instructed to slow down. This can occur in a number of ways. One, for TCP, the receiver could reduce the window size. That is, the receiver as well as the sender could compute a dynamic congestion window. This is complex to design. Second, again for TCP, the ACK packet or a separate control message (such as an ICMP Source Quench) could carry back to the sender some explicit indication to slow down. Third, for TCP, if the traffic meter noted that the receiver seemed to have taken no action in response to the FCN bit, the meter can delete some returning ACKs or an incoming data packet, which will trigger a congestion slowdown in the sender.

The paper by Floyd [Floyd95] contains a detailed discussion of enhancing TCP to include explicit congestion notification, using

either bits in the header or the ICMP Source Quench message with redefined semantics. The range of algorithms explored there for implementing explicit notification are directly applicable to this scheme. In fact, the end node behavior (the source and destination TCP) for her scheme is exactly the same as the scheme here. What is different is the method of notifying the end node of the congestion. In her scheme, random packets are selected to trigger congestion notification. In this scheme, during periods of congestion all packets are marked, but these marks are then removed by the receiver's traffic meter, unless the rate exceeds the installed service profile.

We have simulated the receiver-based scheme, using the ECN mechanism proposed by Floyd to notify the sending TCP to slow down. Because of the very well-behaved characteristics of the ECN scheme, we can regulate TCPs to different sending rates essentially flawlessly.

A key question in the successful implementation of the receiver scheme is defining what constitutes congestion in the router -- under what conditions the router should start setting the FCN bit. Hypothetically, the router should start setting the bit as soon as it detects the onset of queuing in the router. It is important to detect congestion and limit traffic as soon as possible, because it is very undesirable for the queue to build up to the point where packets must be discarded.

Key differences between sender and receiver control

There are a number of interesting asymmetries between the sender and the receiver versions of this tag and profile scheme, asymmetries that arise from the fact that the data packets flow from the sender to the receiver. In the sender scheme, the packet first passes through the meter, where it is tagged, and then through any points of congestion, while in the receiver payment scheme the packet first passes through any points of congestion, where it is tagged, and then through the receiver's meter. The receiver scheme, since it only sets the FCN bit if congestion is actually detected, can convey to the end point dynamic information about current congestion levels. The sender scheme, in contrast, must set the IPI and tag the packet as "in" or "out" without knowing if congestion is actually present. Thus, it would be harder, in the sender scheme, to construct a service that billed the user for actual usage during periods of congestion.

While the receiver scheme seems preferable in that it can naturally implement both static and dynamic payment schemes, the sender scheme has the advantage that since the packet carries in it the explicit assertion of whether it is in or out of profile, when it reaches a

point of congestion, the treatment of the packet is evident. In the receiver scheme, the data packet itself carries no indication of whether it is in or out of profile, so all the point of congestion can do is set the FCN bit, attempt to forward the packet, and trust that the sender will correctly adjust its transmission rate. The receiver scheme is thus much more indirect in its ability to respond to congestion. Of course, the controller at the point of congestion may employ a scheme to discard a packet from the queue, as it does now. However, the receiver scheme gives no guidance as to which packet to delete.

Another difference between the two schemes is that in the sender scheme, the sending application can set the In Profile Indicator in different packets to control which packets are favored during congestion. In the receiver scheme, all packets sent to the receiver pass through and debit the traffic meter before the receiving host gets to see them. Thus, in order for the receiving host to distinguish those packets that should receive preferred service, it would be necessary for it to install some sort of packet filter in the traffic meter. This seems feasible but potentially complex. However, it is again a local matter between the traffic meter and the attached host.

While this scheme works well to control TCP, what about a source that does not adjust when faced with lost packets, or otherwise just floods the congested router? In the receiver-based scheme, there is an increase need for some sort of notification message that can flow backwards through the network from the receiver's traffic meter towards the source of the traffic (or towards the congested routers along the path) so that offending traffic can be distinguished and discriminated against. This sort of mechanism was discussed above in the section on Filtering out Non-Responsive Flows.

Appendix B: Designing traffic meters to control TCP throughput

We have suggested that a useful goal for a traffic meter is to let a well-behaved TCP operate at a specific speed. This is more complex than a service that mimics a link of a specific speed, since a TCP may not be able to fully utilize a physical link because of its behavior dealing with congestion. In order to design a traffic meter that allows a TCP to go at a set speed, the designer must take into account the behavior of TCP. This appendix presents a quick review of the relevant TCP behavior, describes the preliminary design of a traffic meter that directly controls TCP bandwidth, and summarizes some simulation results. Further details of this work can be found in [CF97].

TCP attempts to adjust its performance by varying its window size.

Within the limit imposed by the receive window, the sender increases its window size until a packet is discarded; then reduces its window size and begins again. This process controls the TCP1s effective throughput rate.

There are several different operating regions for TCP. When a number of packets are lost, a TCP reduces its window size to 1, and then (roughly) doubles it each round trip until a threshold is reached. (This threshold is often referred to by the variable name used to implement it in the Berkeley Unix code: ss-thresh.) Once the send window has exceeded ss-thresh, it increases more slowly -- one packet per round trip. When only one (or a small number) of packets are lost, the window size is reduced less drastically; it is cut in half, and ss-thresh is set to the new current window size. It is this latter behavior that is the desired one in order to achieve a reasonable control over the sending rate of the TCP.

When TCP is in this part of its operating range, its window size resembles a saw-tooth, swinging between two values differing by a factor of two. The effect of this saw-tooth window size is to slowly fill up the buffer at the point of congestion until a packet is discarded, then cut the window size by two, which allows the buffer to drain, and may actually cause a period of underutilizing the link. Some thought will suggest that the actual average throughput achieved by the TCP is a function of the buffer size in the router, as well as other parameters. It is difficult to predict.

To design a traffic meter that allows a TCP to achieve a given average rate, it is necessary for the meter to recognize the swings, and fit them into the profile. One approach would be to build a meter that looks at the very long-term average rate, and allows the TCP to send so long as that average is less than the target rate. However, this has the severe drawback that if the TCP undersends for some time because it has no data to send, it builds up a credit in the meter that allows it to exceed the average rate for an excessive time. This sort of overrun can interfere with other TCPs.

The alternative is to build a meter that measures the rate of the sending TCP, and looks for a peak rate (the high point of the saw-tooth). A simple approach is to build a meter that looks for short term sending rates above 1.33 times the target rate R. Once that rate is detected, the meter starts tagging a few packets as "out". When one of these is discarded, the TCP cuts its window size by a factor of two, which will cause some sort of rate reduction, perhaps also to a factor of two. The TCP will thus swing between 1.33 R and .66 R, which averages out to R. One can build a meter that does this, but it is necessary to consider several factors.

The relationship between the window size of a TCP and its sending rate is complex. Once the buffer at the point of congestion starts to fill up, increasing the window size does not increase the sending rate. Each packet added to the window adds one packet in circulation, but adds to the round trip delay by the transmission time of one packet because of the increased waiting time in the buffer. The combination of these two effects is to leave the achieved throughput unchanged. If, on the other hand, the buffer is largely empty, then if the window is cut by 2, the rate will be cut by two.

It is important that the RIO dropper operate in this region, both so that it has enough empty buffer to handle transient congestion, and to improve its ability to control the TCP throughput. With RIO, the average buffer utilization by "out" packets is small, although the instantaneous buffer size can fluctuate due to traffic bursts. As soon as the TCP exceeds its short-term target rate of 1.33 R, some number of "out" packets begin to appear, and if they generate a queue in the router, a packet is dropped probabilistically, which causes the TCP in question to cut its rate by 2.

(Note that in a properly provisioned network, there is enough capacity to carry all the offered "in" packets, and thus "in" packets do not contribute to the RIO buffer load. In a sufficiently underprovisioned network, "in" packet dropping will be triggered, and the TCP congestion control mechanism will limit the packet load as always. Loss of "in" packets indicates to the customer that his provider's provisioning is inadequate to support the customer's profile.)

An important issue in the design of this meter is finding the time period over which to average in order to detect the 1.33 R peak. Average over too long a time, and the average takes into account too much of the saw-tooth, and underestimates the peak rate. Average over too short a period, and the meter begins to detect the short- term bursty nature of the traffic, and detects the 1.33 R peak too soon. Since the round trip of different TCPs can differ by at least one order of magnitude and perhaps two, designing a meter (unless it is integrated into the host implementation and knows the round trip) is difficult. However, reasonable parameters can be set which work over a range of round trip delays, say 10 to 100 ms.

One objection to this approach, in which the meters looks for a short term peak at 1.33 R, is that a creative user could abuse the design by carefully adjusting the window manually so that it achieved a steady-state rate somewhat above R (the long term target average) but below 1.33R. To detect this, the meter has two rate measurements, one of which looks with a short averaging time for a peak of 1.33 R, and

a second one, with a substantially longer period (longer than a saw-tooth) for a flow that exceeds R. If the flow falls short of R, no action is taken, because this might simply be a lack of data to send. But if the TCP exceeds the rate R over a long time, the parameters of the short-term averaging meter are adjusted.

This meter is a sophisticated objective, because it represents a difficult control problem. First, it attempts to set a rate for a sending TCP, rather then just emulating a physical link. Second, it is operating at a low level of traffic aggregation (we have simulated situations with as few as two flows). Third, the meter operates without knowledge of the round-trips of the individual flows. Integrating the meter into the host, so that it can know the measured RTT (which TCP computes anyway) greatly simplifies the design. However, this non-integrated design is more appropriate for an incremental deployment strategy using unmodified hosts.

Avoiding slow-start

As noted above, it is desirable to keep TCP operating in the region where, in response to a lost packet, it cuts its window size in half and sets ss-thresh equal to this new window size. However, if several packets are lost at once, the TCP will execute a different algorithm, called "slow-start", in which it goes idle for some period of time and then sets the window size to 1. It is preferable to avoid this behavior.

One way to avoid this is to avoid dropping several packets in close proximity. There are two halves to achieving this goal.

The first is that the dropper should avoid dropping a block of packets if it has not recently dropped any. That it, it should undergo a gradual transition between the states where it is not dropping any packets, and where it starts to drop. RED, and by extension RIO, has this behavior. Up to some average queue length, RED drops no packets. As the average packet length starts to exceed this length, the probability of loss starts to build, but it is a linear function of how much longer the average is than this minimum. So at first, the rate of drops is very low.

However, if the dropper is overloaded with "out" packets, it will be forced to drop every one that arrives. To deal with this situation, the meter, when it starts tagging packets as "out", also should at first tag the packets infrequently. It should not suddenly enter a mode where it tags a block of packets as "out". However, if the TCP continues to speed up, as it will if the path is uncongested and it can sustain the speed, more and more of the packets will be marked as out, so a gradual transition to tagging in the meter is not

sufficient to avoid all cases of clumped dropping. Both halves of the scheme, the meter and the dropper, must enter the control phase gradually.

In essence, this strategy introduces low-pass filters into both the traffic metering and congestion detection data. These filters are needed to address the two separate cases of the system dropping out packets because the TCP exceeding its profile in an otherwise loaded network, and the system dropping out packets because of new congestion in a network with TCP1s previously operating above profile

Brief simulation results

We have performed some simulations of this traffic meter and the RIO dropper. In this note we show one test case from our simulations. The first column is the target rate, the second column is the actual rate, the third column is the round trip delay.

Target rate	Actual rate	TCP RTT
.1 mb/s	.158 mb/s	20 ms.
1	1.032	20
.1	.193	40
1	1.02	40
.1	.165	60
1	1.01	60
.1	.15	80
1	.95	80
.1	.15	100
1	.93	100

In this simulation, the actual link capacity was exactly the sum of the target rates, so there was no "headroom" for overshoot. As the numbers indicate, we can control the rates of the large flows to within 10% over a range of round trips from 20 to 100 ms, with the longer delay flows having the greater difficulty achieving full speed. The smaller flows, for a number of reasons, are more opportunistic in using any unclaimed capacity, and exceed their target ranges. By adjusting the RIO parameters and the parameters in the meter, different detailed behavior can be produced. We are using this research to fine tune our best understanding of the RIO parameters, as well as the design of advanced meters.

New TCP designs help greatly

Improvements to the dynamic performance of TCP have been proposed for reasons unrelated to this scheme, but rather to more general goals for improved operation. These include SACK TCP, which supports selective acknowledgment when specific packets are lost, and other

TCP tuning changes that deal better with multiple losses. We have simulated our taggers and droppers with these newer TCPs, and the effect is to make the approach work much better. The reason for this is that much of the care in the detailed design is required to avoid triggering slow-start rather than fast recovery, and thus reduce our ability to control the TCP's throughput. The newer TCP designs, which achieve that same goal generally, make our design much more robust.

Another way to improve the operation of this scheme is to use an Explicit Congestion Notification scheme, as has been proposed by Sally Floyd. In this variation of RIO, RIO-ECN, the algorithm does not drop "out" packets at first, but just sends an ECN indication to the destination, where it is returned to the source. The design of Floyd's ECN takes into account the round-trip time, and avoids inadvertent triggering of a slow-start. RIO-ECN, together with a suitable profile meter at the destination, allows us to control TCP sending rates almost without flaw in our simulations.

Appendix C: Economic issues

This is a technical note. However, any discussion of providing different levels of service to different users of a commercial network cannot be complete without acknowledging the presence of economic issues.

The scheme presented here has been conceived in the context of the public commercial Internet, where services are offered for money. It also works in the context of private, corporate or military networks, where other more administrative allocations of high-quality service may be used. But it must work in the context of commercial service. It is therefore crucial that it take into consideration the varying business models of Internet service customers and providers, and that it be consistent with some relevant economic principles.

We discuss these matters briefly below. Note that we are not suggesting that any specific business model, pricing strategy, or service offering be universally adopted. In fact, we believe that a strength of this framework is that it cleanly separates technical mechanism from economic decisions at different points within the network.

Congestion pricing

The first economic principle is that there is only a marginal cost to carrying a packet when the network is congested. When the network is congested, the cost of carrying a packet from user A is the increased delay seen by user B. The traffic of user B, of course, caused delay

for A. But if A somehow were given higher priority, so that B saw most of the delay, A would be receiving better service, and B paying a higher price, in terms of increased delay and (presumably) dissatisfaction. According to economic principles, A should receive better service only if he is willing to pay enough to exceed the "cost" to B of his increased delay. This can be achieved in the marketplace by suitable setting of prices. In principle, on can determine the pricing for access dynamically by allowing A and B to bid for service, although this has many practical problems. For an example of such a proposal, see [MMV95].

When the network is underloaded, however, the packets from A and from B do not interfere with each other. The marginal or incremental cost to the service provider of carrying the packets is zero. In a circumstance where prices follow intrinsic costs, the usage-based component of the charge to the user should be zero. This approach is called "congestion pricing".

The scheme described here is consistent with the framework of congestion pricing. What the user subscribes to, in this scheme, is an expectation of what service he will receive during times of congestion, when the congestion price is non-zero. When the net is underloaded, this scheme permits the user to go faster, since both "in" and "out" packets are forwarded without discrimination in that case.

Pricing need not (and often does not) follow abstract economic principles. An ISP might choose to prevent users from going faster in times of light load, to assess some price for doing so, or whatever. But the scheme is capable of implementing a price/service structure that matches an economically rational model, and we would argue that any scheme should have that characteristic.

This line of reasoning has some practical implications for the design of service profiles. If a provider sells a profile that meters usage over some very long period (so many "in" packets per month, for example) then there will be a powerful incentive for the user not to expend these packets unless congestion is actually encountered. This consequence imposes an extra burden on the user (it is not trivial to detect congestion) and will yield no benefit to either the user or the provider. If there is no cost to sending traffic when the network is underloaded, then there is no cost to having some of those packets carry "in" tags. In fact, there is a secondary benefit, in that it allows providers to track demand for such traffic during all periods, not just during overload. But profiles could be defined that would motivate the user to conserve "in" tags for times of congestion, and these seem misguided.

Getting incentives right

The second economic principle is that pricing can be used as an incentive to shape user behavior toward goals that benefit the overall system, as well as the user. The "incentive compatibility" problem is to structure the service and pricing in such a way that beneficial aggregate behavior results.

Service profiles represent an obvious example of these issues. If a profile can be shaped that closely matches the user's intrinsic need, then he will purchase that profile and use it for those needs. But if the only profile he can get provides him unused capacity, he will be tempted to consume that capacity in some constructive way, since he has been required to purchase it to get what he wants. He may be tempted to resell this capacity, or use it to carry lower value traffic, and so on. These uses represent distortions of the system.

In general, resale of capacity, or arbitrage, results when pricing is distorted, and does not follow cost. It is difficult to design a technical mechanism that can prevent arbitrage, because the mechanism does not control pricing, but the mechanism should not of necessity create situations where arbitrage is a consequence. Generally speaking, this means that price should follow cost, and that profiles should be flexible enough to match the intrinsic needs of a range of users. This scheme attempts to capture this objective by allowing the traffic meters to implement a range of service profiles, rather than standardizing on a fixed set.

Inter-provider payments

One of the places where a traffic meter can be installed is at the boundary between two ISPs. In this circumstance, the purpose is to meter how much traffic of value, i.e. "in" packets, are flowing in each direction. This sort of information can provide the basis for differential compensation between the two providers.

In a pure sender-based scheme, where the revenues are being collected from the sender, the sender of a packet marked as "in" should presumably pay the first ISP, who should in turn pay the second ISP, and so on until the packet reaches its final destination. In the middle of the network, the ISPs would presumably negotiate some long term contract to carry the "in" packets of each other, but if asymmetric flows result, or there is a higher cost to carry the packets onward in one or the other direction, this could constitute a valid basis for differential payment.

As is discussed in [Clark97], the most general model requires both sender and receiver based payments, so that payments can be extracted

from all participants in a transfer in proportion to the value that each brings to the transfer. In this case, the direction of packet flow does not determine the direction of value, and thus the direction of compensating payment. See the referenced paper for a full development of the details of a mixed sender-receiver scheme. It is interesting to note that the sender and receiver-based schemes are to some extent associated with different business models.

The basic sender-based scheme considered in much of this note makes sense in many business contexts. For example, a user with multiple sites, who wants to connect those sites with known service, can equally well express all of these requirements in terms of behavior at the sender, since the senders are all known in advance.

In contrast to this "closed" system, consider the "open" system of a node attached to the public Internet, who wants to purchase some known service profile for interaction with other sites on the Internet. If the primary traffic to that site is incoming (for example, browsing the Web), then it is the receiver of the traffic, not the sender, who associates the value with the transfer. In this case the receiver-based scheme, or a zone scheme, may best meet the needs of the concerned parties.

References

[Clark97] D. Clark, "Combining Sender anbd Receiver Payment Schemes in the Internet"; Proceedings of the Telecommunications Policy Research Conference, Solomon, MD, 1996

[CF97] D. Clark and W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service", (soon) to be available as http://ana.lcs.mit.edu/papers/exp-alloc.ps

[Floyd93] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Trans. on Networking, August 1993

[Floyd95] S. Floyd, "TCP and Explicit Congestion Notification", Computer Communication Review, v 24:5, October, 1995

[FF97] S. Floyd and K. Fall, "Router Mechanisms to Support End-to-End Congestion Control", available at http://www-nrg.ee.lbl.gov/nrg-papers.html

[Kalevi97] K. Kilkki, "Simple Integrated Media Access" Internet Draft, June 1997, <draft-kalevi-simple-media-acccess-01.txt>

[Kelly97] F. Kelly, "Charging and Accounting for Bursty Connections" in "Internet Economics", L. McKnight and J. Bailey, eds., MIT Press,

1997

[MMV95] "Pricing the Internet" in "Public Access to the Internet", B. Kahin and J. Keller, eds., Prentice Hall, 1995. Available as http://www.spp.umich.edu/ipps/papers/info-nets/Pricing_Internet/Pricing_the_Internet.ps.Z

Authors' Addresses:

David D. Clark
MIT Laboratory for Computer Science
545 Technology Sq.
Cambridge, MA 02139
jtw@lcs.mit.edu
617-253-6003
617-253-2673 (FAX)

John Wroclawski
MIT Laboratory for Computer Science
545 Technology Sq.
Cambridge, MA 02139
jtw@lcs.mit.edu
617-253-7885
617-253-2673 (FAX)