Differentiated Services:

A Framework for Internet-Friendly QoS

Kathleen Nichols
kmn@cisco.com

A Hot Interconnects Tutorial

August 20, 1999

Goal of Quality of Service?

Van Jacobson, from a talk on differentiated services and bandwidth brokers given at Bay Networks, November, 1997:

> “[The problem we are solving is to] Give ‘better’ service to some (at the expense of giving worse service to others - QoS fantasies to the contrary, it’s a zero sum game).”

In other words, QoS is managed unfairness.

Then “who gets ‘better’ service?” and “who decides?” are critically important questions for any viable QoS architecture

Disclaimer: This tutorial contains opinions

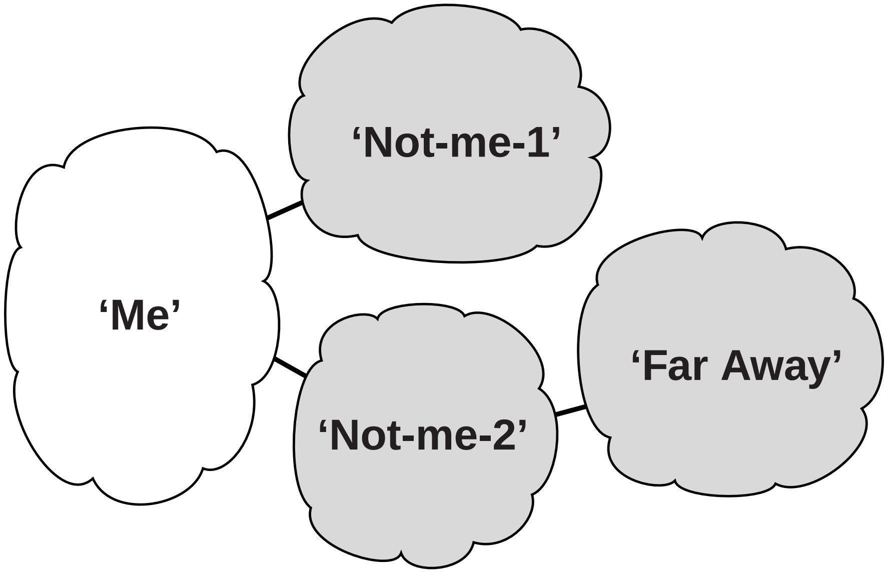
Outline

- Why differentiated services is “Internet friendly”
- The IETF Differentiated Services Working Group’s role
  - Standards-track RFCs
  - Specific PHBs
- Performance issues
- Bandwidth broker allocation framework
- VoIP example
- Related allocation work

Differentiated Services in a Nutshell

- The differentiated services architectural model is an approach to delivering QoS in a scalable, incrementally deployable way that:
  - keeps control of QoS “local”
  - pushes work to the edges and boundaries
  - requires minimal standardization, encourages maximal innovation
- Diffserv’s model is based on an Internet made up of independently administered domains, each of which is connected to at least one other
- The IETF Differentiated Services WG is working on the “minimal standardization” part of this. For information on the IETF Differentiated Services Working Group, see www.ietf.org/html.charters/diffserv-charter.html. The diffserv RFCs are at the bottom of the page.

An Architectural Framework based on Clouds and Boundaries

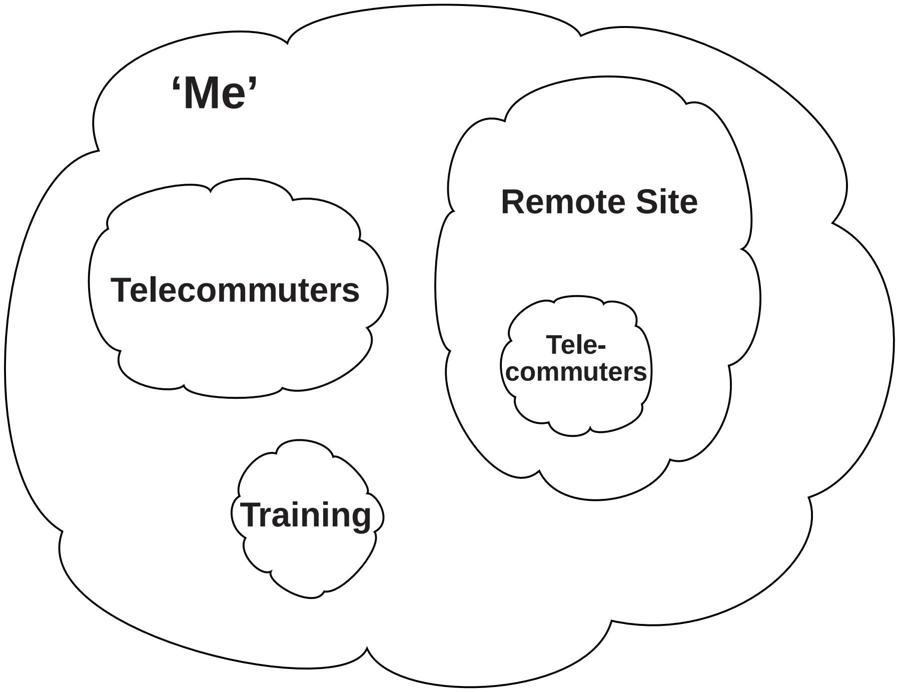


Follows the structure of today's Internet: Clouds are regions of relative homogeneity in terms of administrative control, technology, and/or bandwidth.

Clouds within Clouds

Clouds are not just for separate Autonomous Systems

Clouds may be used to indicate regions whose resources differ or which are administered by different departments of one larger organization



QoS and Clouds

- Within a cloud, QoS is allocated according to some locally determined set of rules
- Almost all the work is confined to the boundaries of clouds and covered by a set of rules
- Rules might not be symmetric across a boundary
- QoS information exchanged between clouds is confined to boundaries and covered by bilateral agreements where clouds have different owners

Advantages of Model Based on Clouds

- Clouds can map to the independently administered regions of the Internet
- Architecturally agnostic: within a cloud any technology might be used to deliver QoS
- Signaling agnostic and signaling can develop and evolve
- Possibility of multiple paths increases reliability
- By focusing on bringing QoS to the Internet a cloud at a time, get incremental deployability
- QoS can be deployed in only one cloud, doesn’t need to be signaled per connection, and the state in most nodes can be reduced considerably as compared to connection-oriented approaches which tie up resources, require state for every connection and are not incrementally deployable or scalable

Scalability through Aggregation

- Fundamental to the diffserv approach is:
  - there are a relatively small number of ways to handle packets in the forwarding path
  - the number of traffic conversations requiring QoS may be quite large and subject to a wide range of rules which devolve from policy
- Packets are grouped by the forwarding behavior they are to receive within a cloud
- Nodes in the center of a network only have to deal with the small number of traffic aggregates rather than keeping track of every separate traffic conversation that passes through

Aggregation and Conversations

- The per-conversation state is kept at the edges
- Flows or conversations are classified into aggregates and are "conditioned" to meet the rules of that aggregate
- Packets are not marked for the "services" individual conversations may be receiving. Many services may use the same marking. Any viable service must make sense under aggregation
- Don't distinguish between flows, so the treatment the behavior aggregate receives should not result in different performance for different traffic compositions of the behavior aggregate

Two Major Components to Diffserv QoS

One is the forwarding path behavior, the differential treatment an individual packet receives, such mechanisms as:

- queue service disciplines and/or queue management disciplines,
- the packet classification and traffic conditioning (e.g., shaping, policing) that can be used to control aggregates

These behaviors are useful and required in network nodes to deliver differentiated treatment of packets no matter how we construct end-to-end or intra-domain services.

For the most part, these primitives are well-understood and must be fairly simple as they are done per-packet at forwarding speed

Here, diffserv focuses on the general semantics of the behaviors rather than the specific mechanisms used to implement them since these behaviors will evolve less rapidly than the mechanisms.

Two Major Components to Diffserv QoS (cont'd)

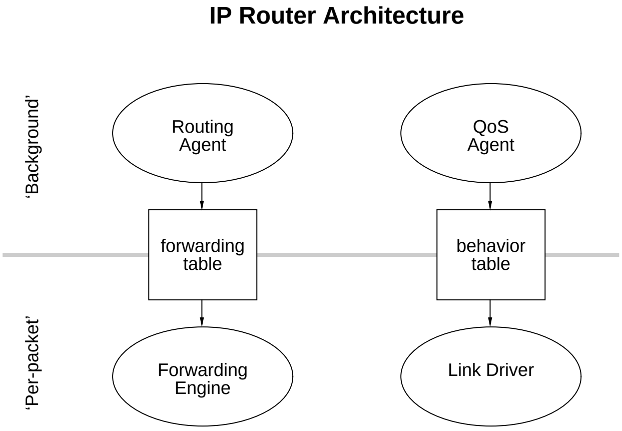
The second is the control plane used to implement a cloud's policy goals and to configure the forwarding path accordingly: Which packets get special treatment? What kinds of rules are to be applied to the use of resources?

This is much less well-understood. Yet, using simple policies and static configurations, it is possible to deploy useful differentiated services in networks.

There are a number of such ways to configure the forwarding path to create services (e.g., RFC 2475, RFC 2598, RFC 2638). Deployment of these gains us experience that will guide more complex policies and allocations.

This approach has an analogy in the original evolution of the Internet in the development of forwarding and routing.

Mirrors the Development of the Internet



Packet forwarding is a simple task, performed on a per-packet basis as quickly as possible. It uses the packet header to find an entry in a routing table that determines the packet's output interface. Has changed little from the start.

Routing sets and maintains the entries in that table and may need to reflect a range of transit and other policies as well as to keep track of route failures. The work of routing is more complex and happens outside the forwarding path. Has continued to evolve.

“Minimal Standardization”: RFC 2474

- A bit-field in the packet header determines the packet’s forwarding treatment. DS field covers the TOS octet in IPv4 and the Traffic Class octet in IPv6; within that uses bits 0-5 as a “codepoint” (DSCP)
- Codepoints should be looked at as an index into a table of packet forwarding treatments at each router.
- This table maps a DSCP to a particular forwarding treatment or “per-hop behavior” (PHB) that is applied to packets with that marking. PHBs are constructed by vendors from, for example, particular queue schedulers
- Behavior for only a few codepoints are to be globally assigned and diffserv-capable equipment must make codepoint to behavior mapping flexible and accessible
- Class Selector Compliant PHBs get DSCPs 000000-111000

Behavior Aggregates, DSCPs, and Classifiers

The DSCP indicates the packet's behavior aggregate within an administrative domain (or cloud)

Packets with the same DSCP get the same treatment in the forwarding path, thus providing aggregation and scalability

A packet's DS field may be marked with a codepoint "anywhere" in the network (but marking is expected at edges and boundaries)

Marking can be based on microflow identification, the packet ingress link, the measured temporal characteristics of a microflow or aggregate, etc.

Flow identification is done using *multifield* (MF) classifiers.

Behavior aggregate identification is done using *behavior aggregate* (BA) or DSCP classifiers. These SHOULD be included in all network nodes in a DS domain.

Use of Traffic Conditioners

Traffic conditioning is used to enforce agreements between domains and to condition traffic to receive a differentiated service within a domain by marking packets with the appropriate codepoint and by monitoring and altering the temporal characteristics of the aggregate where necessary.

- Traffic conditioners may alter the temporal characteristics of a behavior aggregate to conform to some requirements of a particular DS domain.
- Traffic conditioners are entities that perform control functions that can be applied to a “behavior aggregate, application flow, or other operationally useful subset of traffic, e.g., routing updates.”
- Traffic conditioners include metering, policing, shaping, and marking.

The Class Selector Compliant PHB Group

This PHB group is defined in RFC2474 and seems to be widely underappreciated

- Motivation was to preserve some “backward compatibility” with the use of bits 0-2 of the TOS byte (the Precedence Field) while permitting evolution to a more useful PHB group
- RFC 2474 describes minimum requirements on a set of PHBs that are compatible with most of the deployed forwarding treatments selected by the IP Precedence field.
- The DSCPs 0-7 MUST map to PHBs meeting the requirements though the PHBs may impose *additional* requirements.
- Other codepoints MAY map to these same PHBs.

Class Selector PHB Group Requirements

The CSC group must include at least two independently forwarded classes and the one codepoints 6&7 map to must “give packets a preferential forwarding treatment” compared to the PHB selected by DSCP 0 (the “default” behavior DSCP)

Packets with different Class Selector DSCPs may be reordered, though packets with the same DSCP which are not dropped should remain in order

A network node may enforce limits on the amount of the node's resources that can be utilized by each of these PHBs

Note that the CS Requirements can be met with two queues, one of which is mapped to by DSCPs 0-5, the other by DSCP 6&7.

On the other hand, the CS Requirements can be met with a sophisticated CBQ scheduler which may have more than eight queues.

Example Uses of a CSC PHB Group

- Separate traffic by “importance”
- Limit bandwidth given to an aggregate
- Guarantee bandwidth given to an aggregate
- Separate queues for some customers

(These can be discussed in more detail during the tutorial)

Assured Forwarding (AF) PHB Group:RFC 2597

Four independently forwarded AF classes and within each AF class,three levels of drop precedence (two okay)

Drop precedence of a packet determines the relative importance of the packet within the AF class. A congested AF node preferably discards packets with a higher drop precedence value

Packets with the lowest drop precedence value are assumed to be within a “subscribed profile”.

A packet that belongs to AF class i and has drop precedence j within that class is designated as belonging to the AFij aggregate.

An AF-compliant node allocates resources sufficient to (at least) achieve the configured service bandwidth over “both large and small time scales.”

More on the AF PHB Group

The concept of drop precedence within a class is a descendant of the MIT work on RIO (“RED with In and Out”, see diffserv.lcs.mit.edu/Papers/exp-alloc-ddc-wf.pdf), a two-level DP

There are twelve recommended codepoints:

Drop Precedence	Class 1	Class 2	Class 3	Class 4
Low	001010	010010	011010	100010
Medium	001100	010100	0111001	100100
High	001110	010110	011110	100110

It is possible to use some of the CS codepoints since the AF classes could constitute a CS-compliant PHB group

Example Implementations of RFC 2597

The four AF classes can be implemented with four queues serviced by WRR; further, these might be four CSC queues, say DSCPs 1-4

The drop precedence mechanism in each class (queue) can be implemented by RIO where both of the higher DPs map to “outs” and the lower DP maps to “ins”. Cisco has generalized this to “weighted RED” (WRED)

Buffer allocation strategies based on DP might also work

Can’t use separate queues due to the non-reordering constraint

Some Internet-drafts on various “markers” for indicating which DP within a class based on compliance to service profile (Heinanen, Fang)

The EF Per-Hop Behavior (RFC 2598)

This is a forwarding behavior of general use, but its most likely applicability is for “virtual leased lines” (VLLs) and for VoIP

In simple terms, it is a rough equivalent of PQ, can be implemented by PQ with some safety mechanism

More precisely (from RFC 2598):

> “the departure rate of the aggregate’s packets from any diffserv node must equal or exceed a configurable rate.
>
> The EF traffic SHOULD receive this rate independent of the intensity of any other traffic attempting to transit the node.
>
> It SHOULD average at least the configured rate when measured over any time interval equal to or longer than the time it takes to send an output link MTU sized packet at the configured rate. “

Implications of the Definitions

At most 50% of a link can be composed of the EF aggregate. Otherwise, an MTU-sized packet of some other aggregate followed or proceeded by an MTU-sized EF packet will violate the “SHOULDs” quoted on the previous slide.

The worst case displacement (which leads to jitter) of an EF packet is composed of two parts:

- First, the packet might encounter a packet from each of the other microflows which compose the EF aggregate. (If the EF configuration conforms to RFC 2598 then a packet from any microflow can encounter at most one packet from each of the other EF microflows.)
- Second, the packet might have to wait for an MTU-sized packet of some other aggregate at every hop along its path.

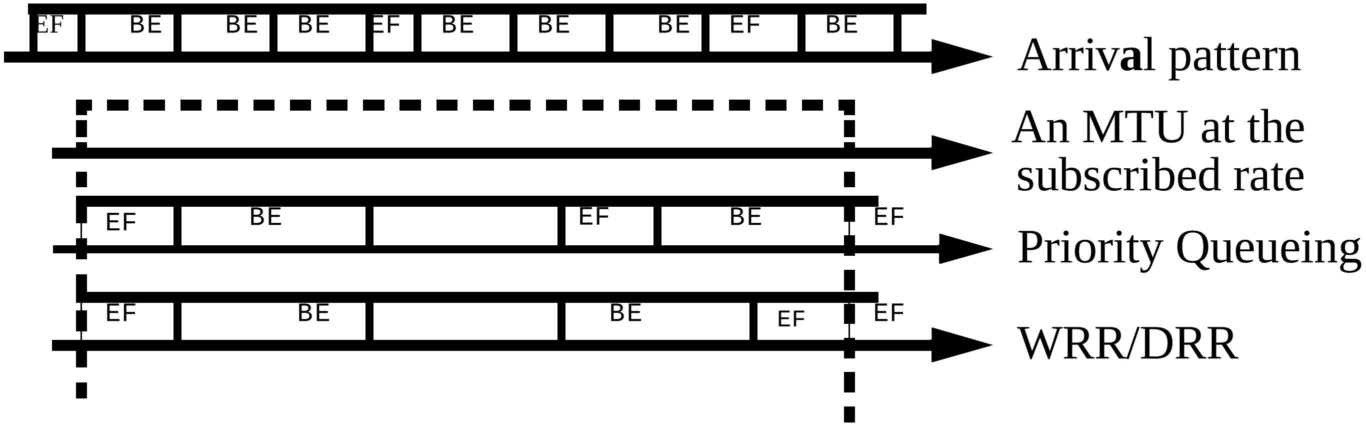
To get this worst case, the next packet of the stream must wait for *no* packets

Example Implementations of EF PHB

Arrival rate is twice the output link bandwidth

EF queue configured to get 25% of the output link bandwidth. BE is configured to get 75% of the output link bandwidth.

(Here packets come in only two sizes, the MTU and one-half the MTU.)



Pending Issues in the Diffserv WG

- A diffserv MIB
- Terminology for a collection of behavior aggregates that have an ordering constraint between them (this only applies to certain AF aggregates at present)
- Correct approach for tunnels
- General purpose PHB identifiers that may be used by a range of protocols
- Just selected two new terms:
  - Ordered Aggregate: a set of Behavior Aggregates which share an ordering constraint (as in different AF DPs in same AF class)
  - PHB Scheduling Class: the set of one or more PHBs applied to the set of Behavior Aggregates forming a given OA

Services in the Diffserv Framework

- That covers the forwarding path, but what about building services?
- Services are built by adding rules to govern behavior aggregates:
  - initial packet marking
  - how particular aggregates are treated at boundaries
  - temporal behavior of aggregates at boundaries
- Classifiers and traffic conditioners at DS boundaries are configured to enforce rules in accordance with the service specification (briefly, let’s punt how the rules get propagated)
- Different user-visible services can share the same aggregate
- Services must be sensible and quantifiable under aggregation

Issues in Evaluating Differentiated Services

There are many Internet Drafts, conference papers, and journal papers that purport to be evaluations of differentiated services

Beware

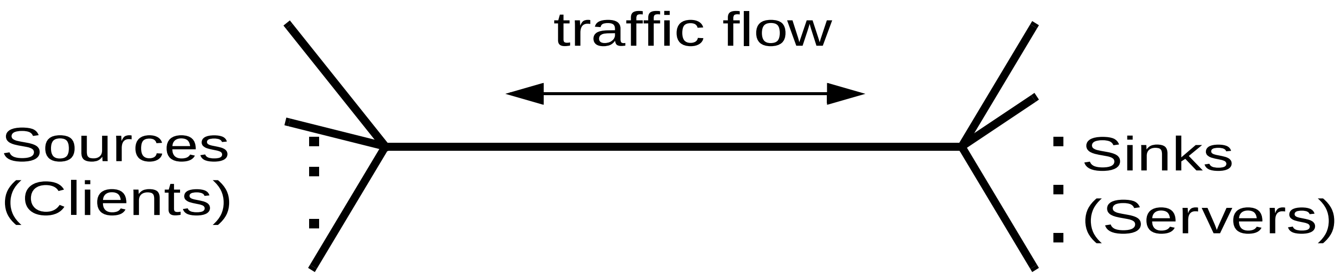
Many of these make sweeping conclusions based on very simple traffic models and topologies.

A large number of studies are based on “dumbell” topologies and use a traffic model consisting *only* of long-lived TCPs with the identical RTTs as their traffic model. It’s not possible to see aggregation issues in such a topology

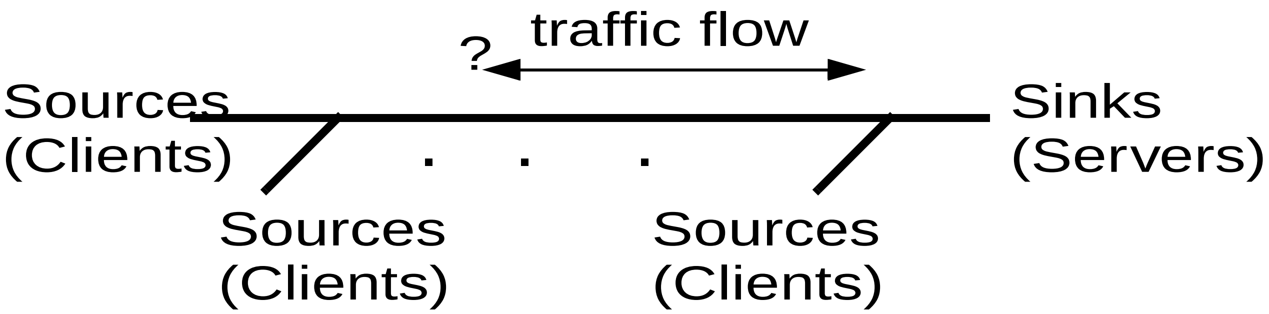
So ask yourself what topology captures the salient features of a realistic network? Were traffic loads sufficient to exercise a range of operating points? Do the traffic loads capture the salient features of real networks? Were the simulations run long enough and for enough variation in random starting conditions?

What kind of network?

Dumbbells and Dancehalls

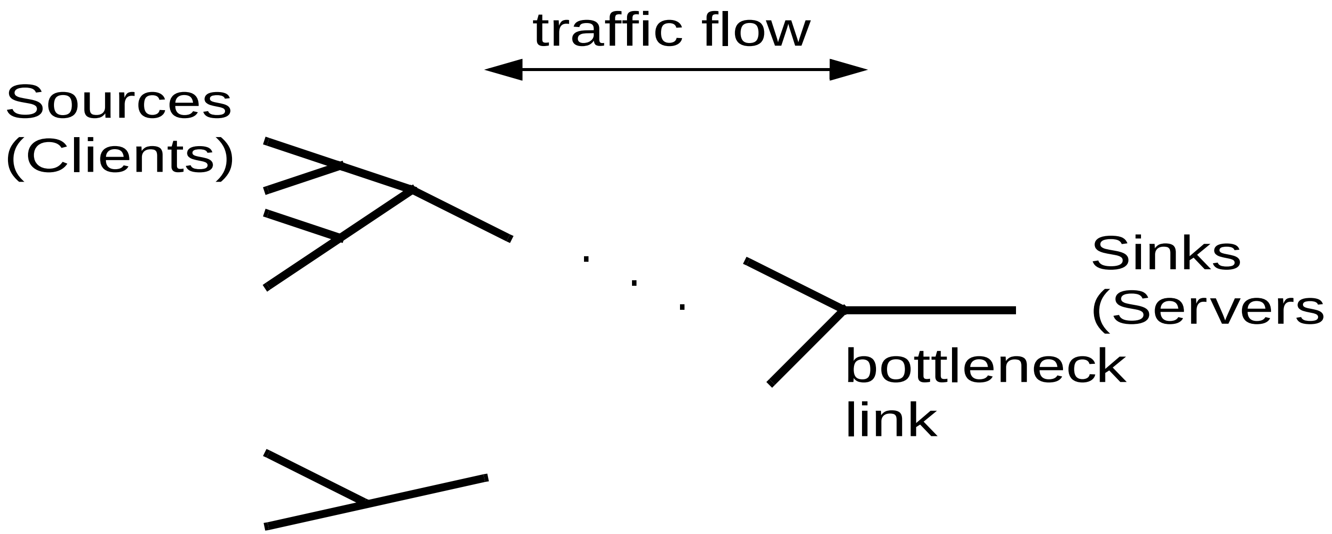


Freeway with On-ramps



More Topologies

Branch-y (Trees)



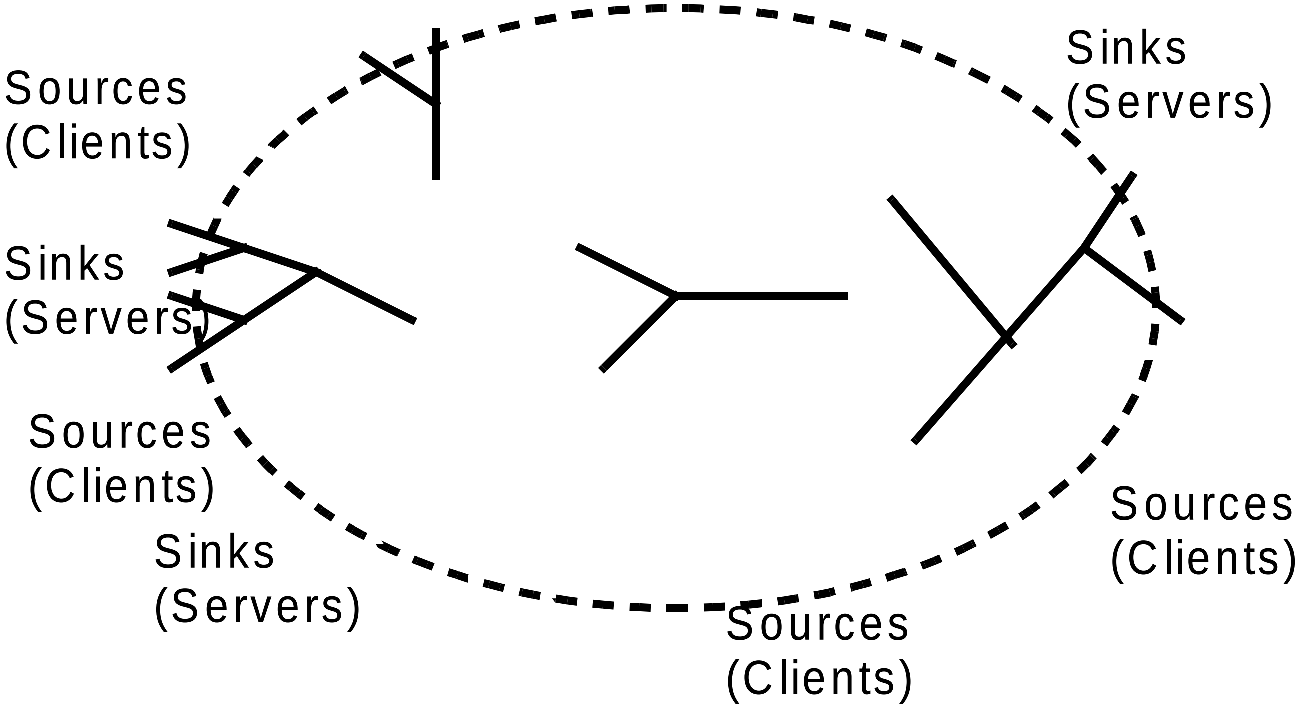
traffic flow

Sources
(Clients)

Sinks
(Servers

bottleneck
link

Branch-y and Mesh-y



This one is both hard to draw and hard to simulate.

This makes it less appealing for graduate students who want to produce papers in a short time period.

Traffic Models

Most credible measurements put the percentage of HTTP traffic in networks at 60-90% of flows.

Measurement studies have found the distribution of file sizes fetched by HTTP transactions are Pareto distributed with most objects fitting into a single packet.

When a user downloads a web page, typically many small transfers take place. Such flows have different characteristics from a long-lived TCP (like FTPing a large file)

Further, many of the simulator models used treat all packets as having the same size or ignore the SYNs and FINs of a TCP connection, both of which obscure web traffic patterns.

Most of the published studies use long-lived TCPs. Some CBR.

What we know about Internet traffic is that it will change over time, so we must design for a range of operating points

Results for Drop Preference

Dave Clark of MIT proposed a service based on RIO droppers and edge marking by “time sliding window” markers

RIO droppers use two 93 RED algorithms running on the same queue. For “outs”, dropping starts at a lower average queue size and has a higher probability. The queue average used for the “outs” includes all packets, for the “ins”, only the “in” packets are counted

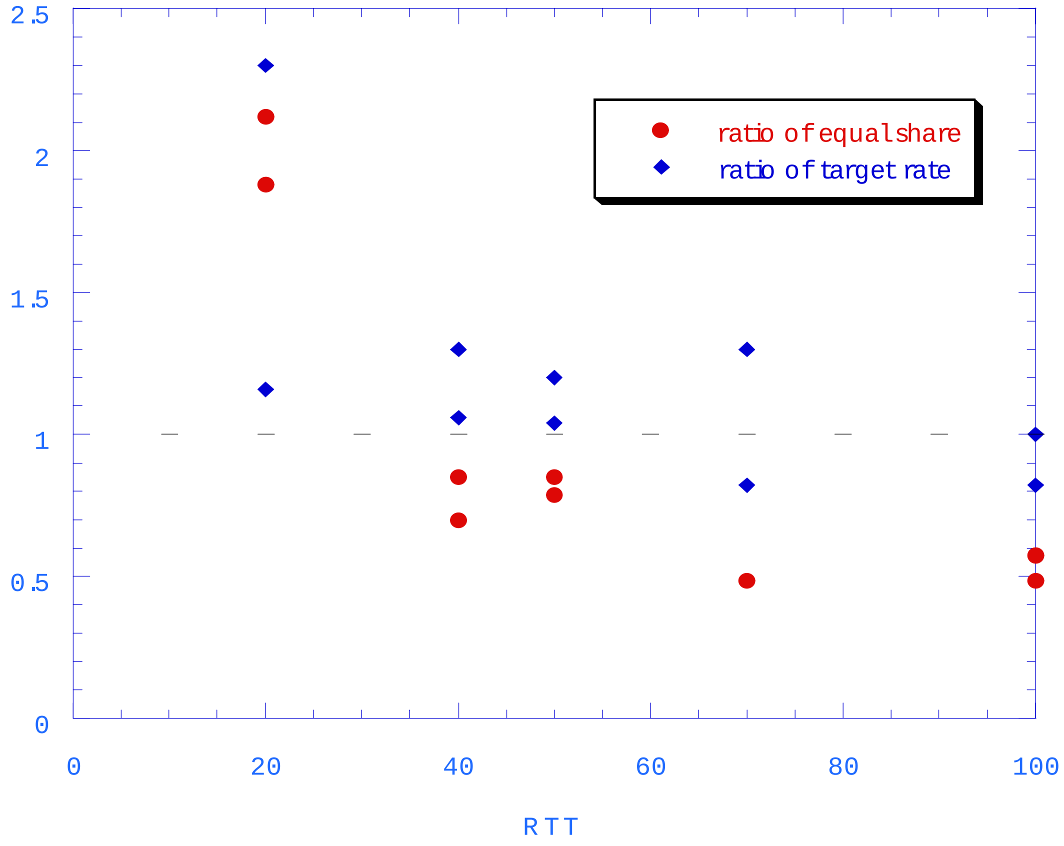
Result were reported in “Explicit Allocation of Best Effort Packet Delivery Services” by Feng and Clark, published in Transactions on Networking (URL on a previous slide)

Table 1of that document shows results for a topology of a single shared 33 Mbps link with 10 sources on one end, 10 sinks on other, 91% of the link was allocated, each flow had a target rate of either 1 Mbps or 5 Mbps and one of 5 RTTs (20-100ms). I plotted the table to get a better picture of the RTT effect.

Achieved-to-Target vs. RTT

Round/red are the results for the all BE flows, divided by 3.3Mbps

Diamond/blue are the RIO results divided by target rate



What do the Results Mean?

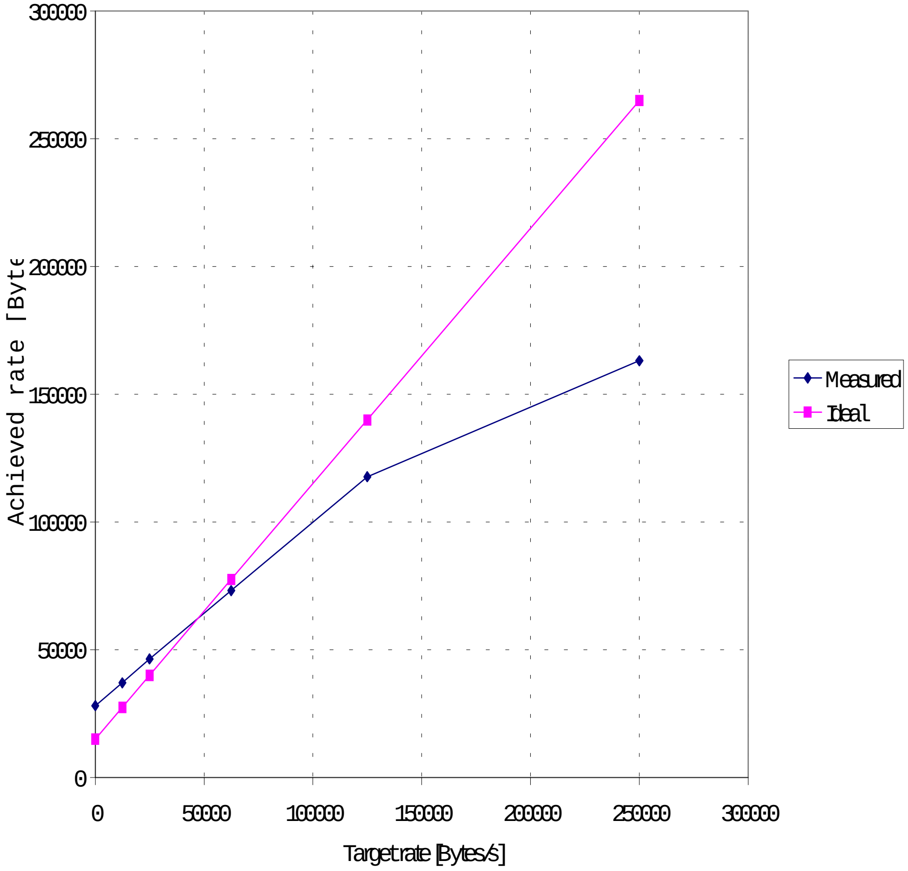
Looked like using “in” marked packets might overcome some RTT effects, but not conclusive, clear, or quantifiable

What would happen if there was more BE traffic mixed in (of various types) and more interesting topologies?

Juan-Antonio Ibanez simulated a similar topology, 50 sources and sinks through a single link, using only long-lived TCPs. He found sensitivities to RIO parameters and metering (how the decision to mark “in” is made). An edited form of his report was published as an Internet Draft in July of 1998 and is available at http://ec.eurecom.fr/~ibanez/internet_draft.html.

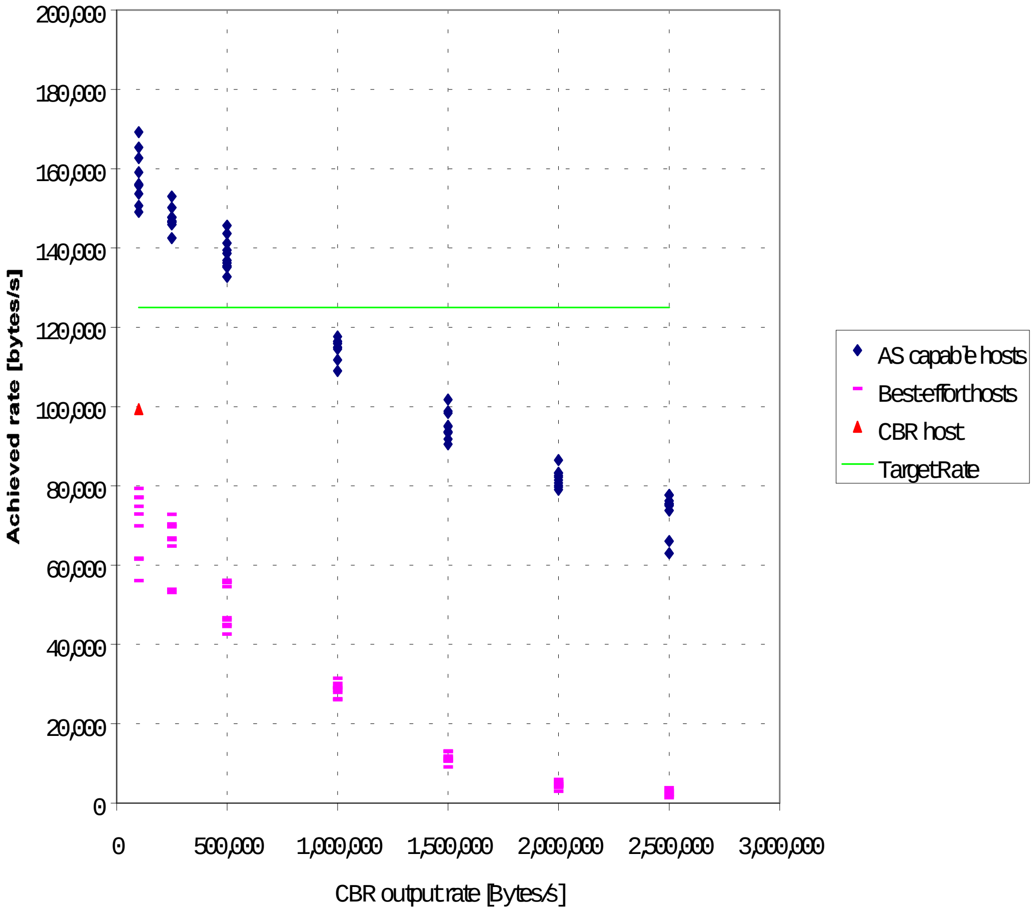
Here are some of those results.

Performance against Target Rates



Mix of 20 RIO, 20 BE FTPs where 76% of shared link bandwidth was allocated to “ins”. Five target rates were allocated to sets of four hosts each. The rates were: 0.1, 0.2, 0.5, 1.0, and 5.0 Mbps.

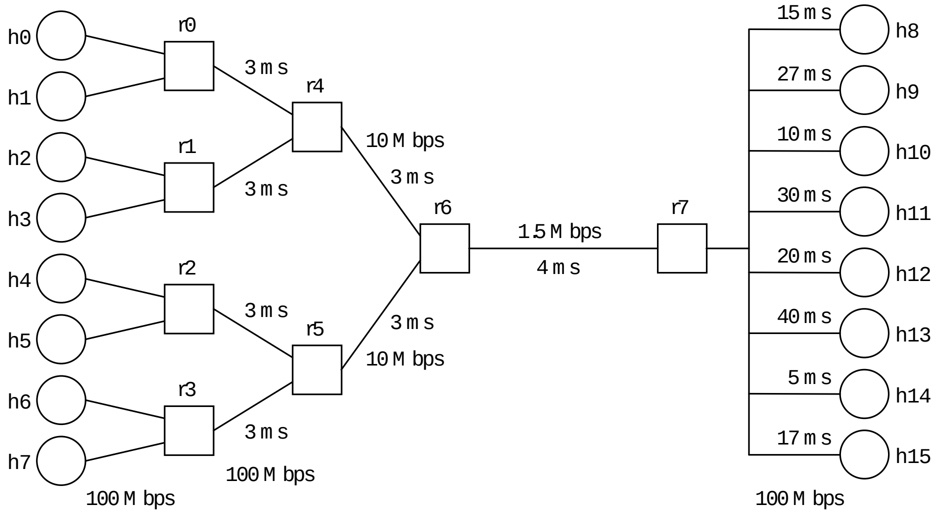
Effect of a Non-responsive Source



Added a non-responsive BE source to 10 RIO, 10 BE FTPs (same RTT, same target rate). The “out” packets degrade the performance.

Preliminary Work with a Branchy Topology

Eight FTPs, each with a target rate of 100 Kbps. Police at merges. Five of the FTPs saw one to three drops of “ins”. 7.5% of “ins” remarked “outs”

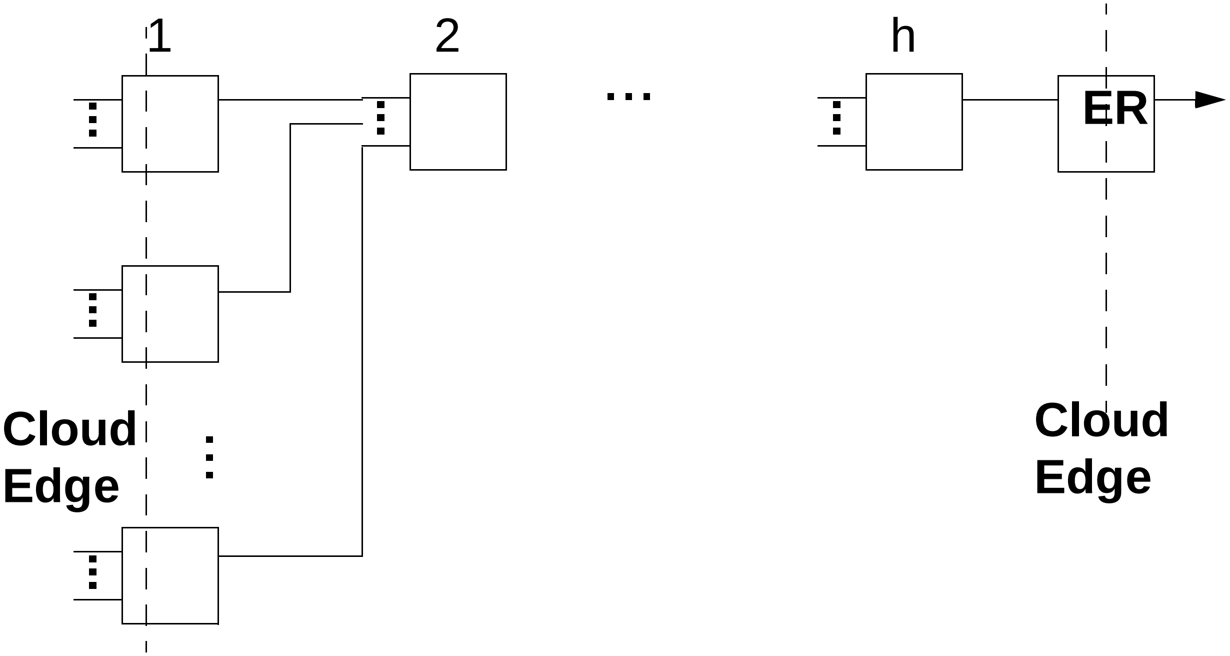


Host id	0	1	2	3	4	5	6	7
RTT (ms)	50	74	40	80	60	100	30	54
Measured ratio	1.9	1.7	1.7	1.7	2.0	1.5	2.2	1.6

Worst Case Train of “In” Packets

Assume: each node has in-degree, i, each internal link has bandwidth B, the egress link has bandwidth L, each input may burst b “in” packets. How large a train of “in” packets can appear at link L?

Answer: train(b,i,h) = b * i^h



Implications

If the rate allocation of “in” traffic is *a***L*, then can expect that if *a***b* >= 1.0, the entire link can be used by the “in” traffic for the period it takes to transmit train(*b,i,h*) packets at rate *L*

If *L* < *B*, then must buffer a train*(*B*-*L*)/*B* packets to avoid loss. Reasonable values of *B/L* are in the range from 1 to 100. At B/L of 5, buffering 80% of the burst; at 10, 90%; at 100, 99%. (This is a pessimistic estimate

Use of a value of *i* smaller than the actual router in-degree might be used to indicate a more “statistically likely” percentage of sources that are sending their allocation

Other Studies

Claim is that marking non-responsive sources as “way out”, can solve problem. This raises questions many new questions.

“Effect of Number of Drop Precedences in Assured Forwarding”, Mukul Goyal, Arian Durresi, Raj Jain, ftp.isi.edu/internet-drafts/ draft-goyal-dpstdy-diffserv-01.txt. In best case, appears that “in-profile” traffic needs to be limited to 50% of the link and this for a “dumbbell dancehall” topology where there are two levels of mixing, one for 5 senders, the next merges these 10 merged streams. Traffic model of FTPs with unresponsive CBR.

Other published evaluations use a very simple dumbbell and no additional traffic models. Allocated traffic is long-lived FTPs; non-responsive traffic is all considered “undesirable”

Note that this is all very much untried with the primary traffic of the Internet: web surfing

Other Open Questions about DP

- Is it possible to build a service which aggregates well from this PHB? Resources are allocated for service rate, but bursts are permitted. The effect on the applications is not clear if bursts are all remarked to a higher precedence.
- How do you allocate in order to accomodate bursts of lower drop precedence? The effect on the network is not clear if multi-packet bursts of low drop precedence are allowed.
- How much re-marking happens in a branchier network? Do the remarked packets later get dropped? Is the re-marking “fair”? In what sense can a flow’s “expected capacity” be predicted?
- Can one non-reponsive flow take over the high DP (non-allocated) bandwidth?
- What about when “non-responsive” does not equate to “bad”?

Evaluating EF PHB’s Virtual Leased Line

RFC 2598 proposes a “virtual leased line” (VLL) service that can be built from the EF PHB combined with border policers that drop packets that exceed a peak rate where no bursts are permitted.

Shaping is expected to be employed to conform to the policers, where an “upstream” border will shape its traffic to avoid drop.

With these rules, it is easy to aggregate such a service since the peak rates just add.

The notion of a VLL is that is “looks like a wire” to the application in terms of its performance.

Much discussion has centered on the use of VLL for VoIP, since it has the properties of low jitter and guaranteed rate and VoIP is a topic of considerable current interest. It is intended to provide what appears to be a dedicated link which can be used by *any* mixture of network traffic, potentially useful for VPNs.

Evaluating VLLs Forwarding Path Performance

Forwarding path performance of VLLs for general Internet traffic is frankly not that interesting. As long as the traffic is shaped properly (this means sufficiently sized holding buffers and good queue management), the VLL really does look like a dedicated link at the subscribed peak rate.

Most VLL evaluation concerns jitter performance for VoIP type traffic in a VLL. These values are easy to bound in worst case, but we’d like to understand “typical” behavior and how different EF PHB implementations and allocation percentages affect jitter performance.

Kedarnath Poduri simulated EF performance on the six-hop branchy topology with a mixture of flow types and using a mix of BE traffic (HTTP, FTP, CBR). I presented the results in August ‘98 at the NASA NREN QoS Workshop (http://www.nren.nasa.gov/workshops3.html) and the jitter work was included in RFC 2598.

Performance Against a Dedicated Link

(These results are also due to K. Poduri and were also presented at the NASA Workshop.)

The percentage of the bottleneck link allocated to “marked” packets varies: 20, 40, and 60 percent.

Eight FTPs were rate-limited and marked for EF PHB.

Results are the average transfer rate of each flow (all were the same).

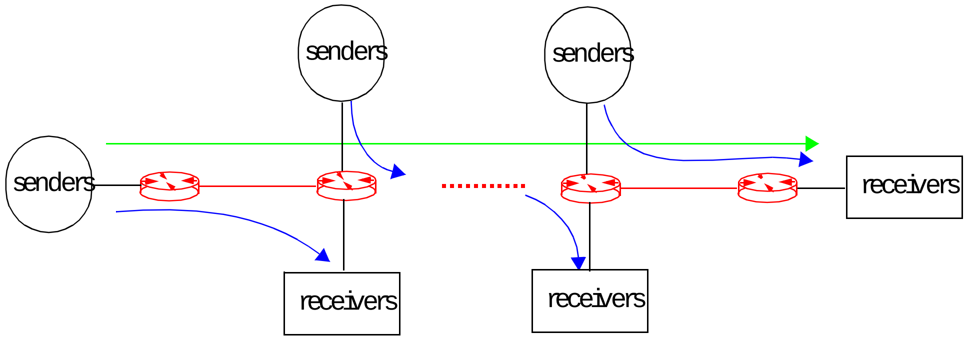
percentage of bottleneck	rate (Kbps)	measured line rate	measured EF rate
20	100	90	90
40	150	143	143
60	225	213	215

Evaluating Jitter for VoIP using VLLs

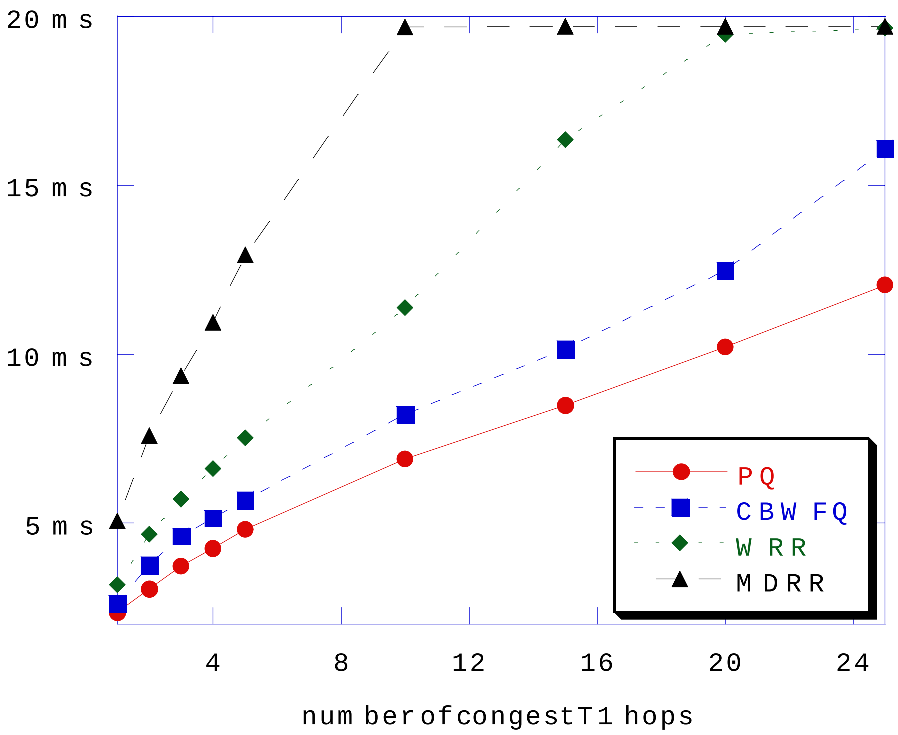
More recent results focused on a VLL over multiple hops. Following results due to Yonghwan Kim of Cisco, for an internal report. Also presented at ETSI QoS Workshop in June '99.

Jitter is the difference between the absolute value of the arrival time differences minus the absolute value of the departure time differences of two adjacent packets, $(|(a_j-a_i)-(d_j-d_i)|)$.

"Freeway" topology of varying numbers of hops. Each hop is 1.5 Mbps.



Median Jitter of VoIP using VLL

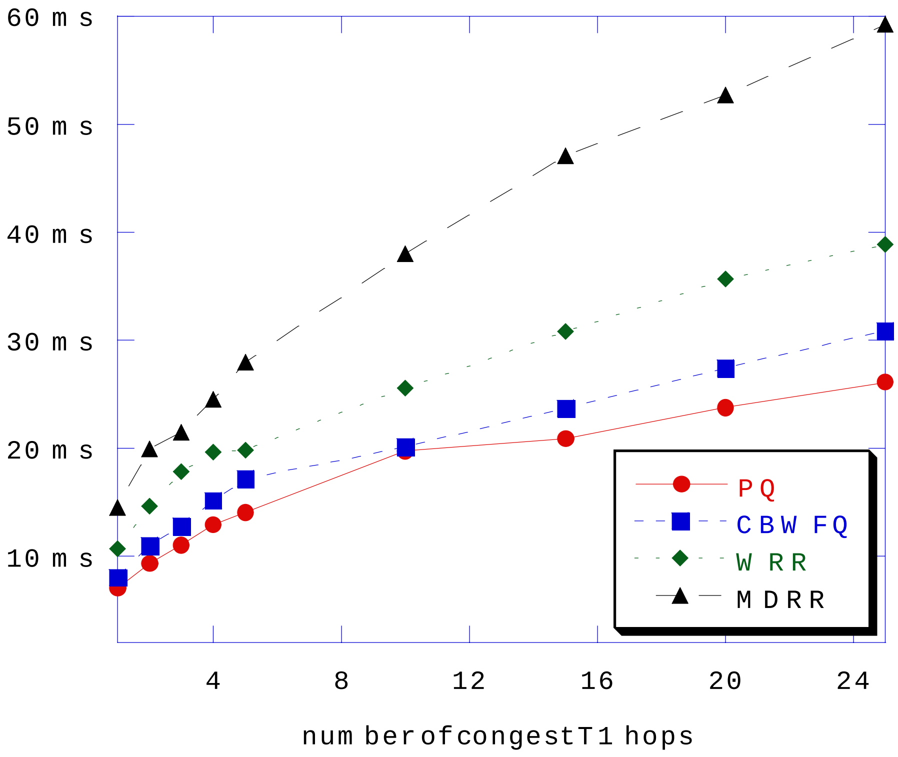


The “VoIP” flows were 24Kbps, 20 ms between 60 byte packets

10% of traffic is EF-marked, 60% gets other “special” treatment

Of the EF packets, half are long (1500 bytes), half short (100 bytes)

95th Percentile Jitter of VoIP using VLL



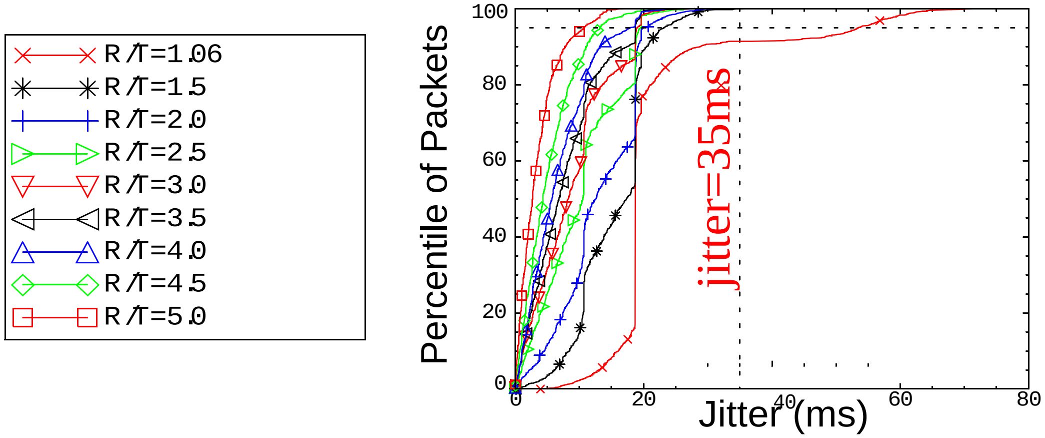
Jitter for a WRR EF Implementation

Uses a single bottleneck topology, six hops, middle is bottleneck

Here there are a total of 5 queues, one of which is EF.

EF gets 20% of each link.

A WRR queue scheduler is used and the ratio of the Rounding weight to the Target rate is varied, R/T.



Simple Analysis to Bound EF Jitter

Recalling our earlier discussion of the two components of jitter, we assume a worst case for both the displacement from other EF microflows and the per-hop wait for a full MTU-sized packet of another aggregate.

Assume the cloud’s bottleneck bandwidth, **B**, is the bandwidth of every link. Worst case is an allocation, **a**, of 50%. Compare to a 30% allocation to see how that affects jitter.

We can refine this to be a sort of “average” worst case by first noting that, at each hop, we wait on average for a half of an MTU-sized packet and that the chances of meeting a packet of another aggregate depend on how highly utilized the link is. The former halves the jitter component due to the number of hops, **h**, and the latter multiplies it times the utilization percentage.

These bounds assume no displacement in the next packet of the stream which is quite unlikely.

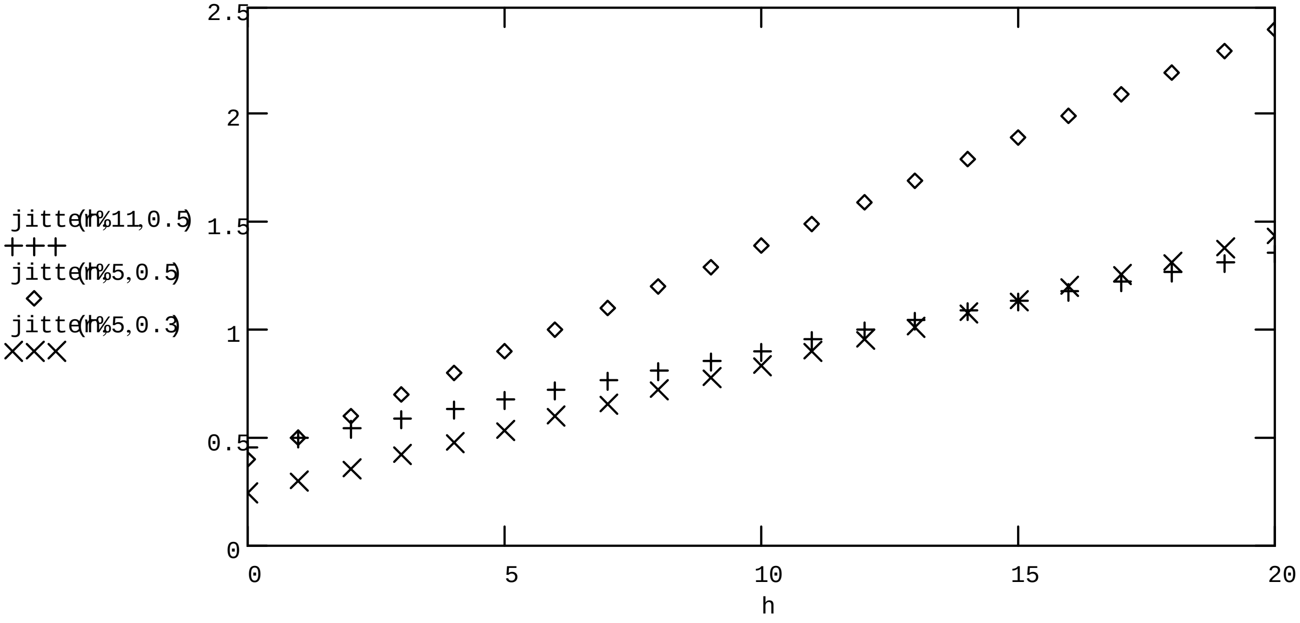
Worst-case Bound on EF Jitter

$$\text{jitter}(h,n,a) := \frac{\left[(n-1)\cdot\left(\frac{MTU}{B}\right)\right] + h\cdot\left(\frac{MTU}{B}\right)}{\left(\frac{MTU}{a\cdot\frac{B}{n}}\right)}$$

$$\text{jitter}(h,n,a) := \frac{(n-1)+h}{\frac{n}{a}}$$

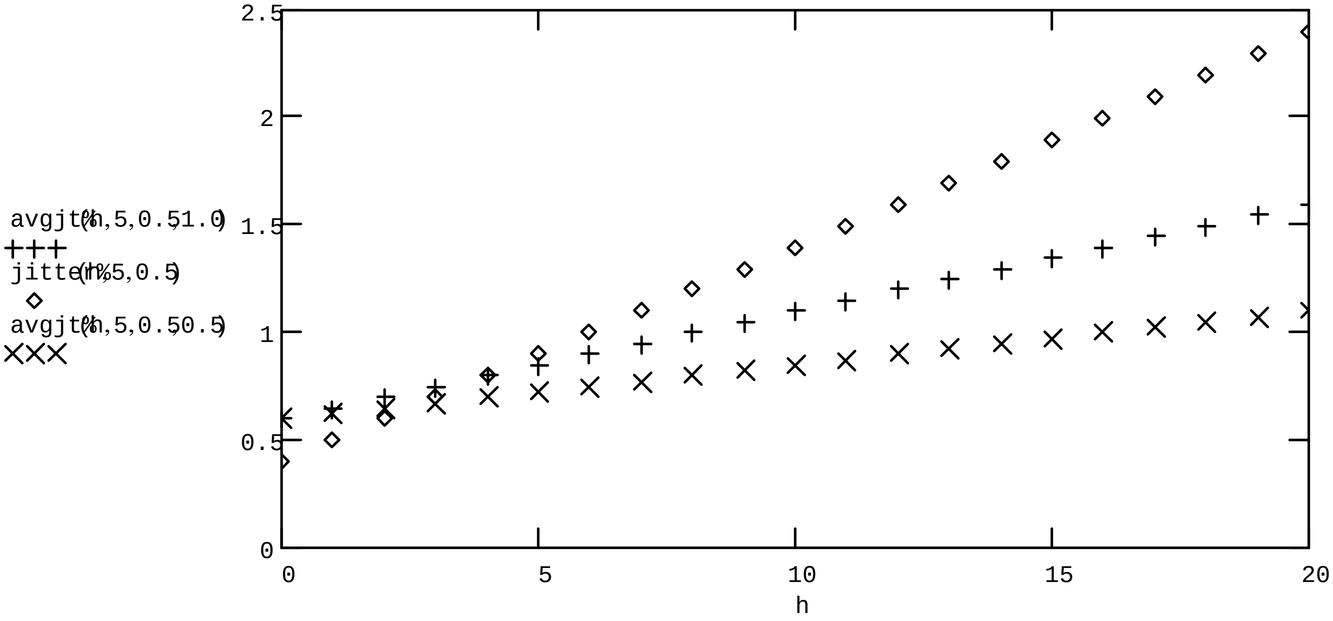
Jitter could be larger as a frac a smaller packet size, but the number of bytes and thus the tim would remain constant

The maximum allocation of a=0.5 gives the largest jitter



Relaxing the Bound with Averaging

$$\mathrm{avgjt\%}(h,n,a,u) := \frac{(n+1)+\frac{u}{2}\cdot h}{\frac{n}{a}}$$



Using the average wait of half an MTU makes a larger difference than varying the link utilization, but both give some feel for what conditions have to occur to have jitter exceed one MTU at rate

Allocation of QoS Services

A technical specification of a particular service first must make sense and must perform as expected

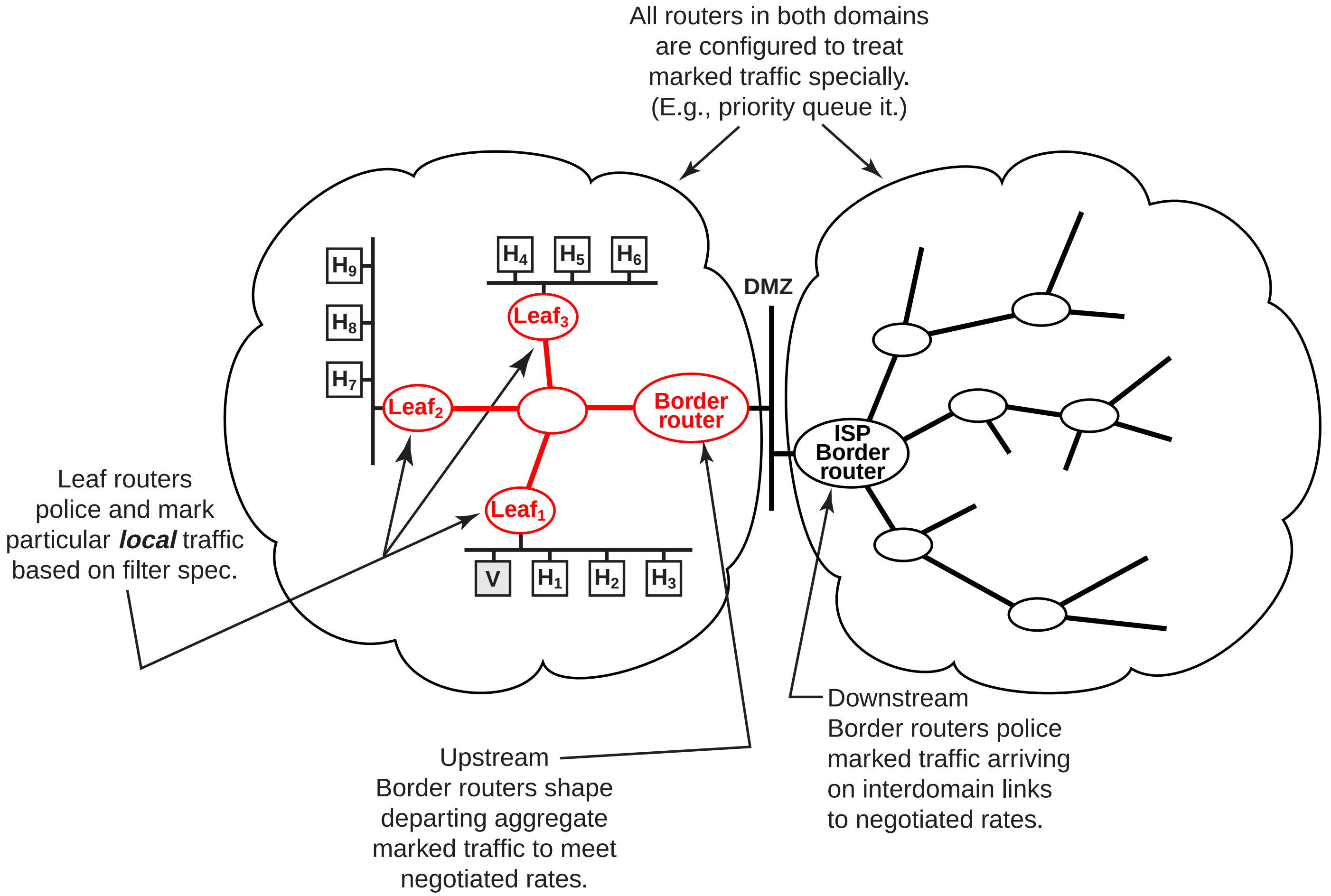
Then we can begin to explore how it can be allocated within a cloud to meet technical constraints and policy considerations. Ideally, start with simple and conservative allocations

Traffic conditioners at boundaries as well as the mechanisms that implement PHBs must be properly configured to the allocation

For services which cross clouds, must address how to communicate and specify the relevant data about the service between clouds. Ideally, the information that crossed boundaries is simple and static or quasi-static

As an application to be given QoS, voice has some nice properties: a known and relatively low bandwidth, self-shaped

Example: Putting Together a VLL Service



All routers in both domains are configured to treat marked traffic specially. (E.g., priority queue it.)

Leaf routers police and mark particular ***local*** traffic based on filter spec.

Upstream Border routers shape departing aggregate marked traffic to meet negotiated rates.

Downstream Border routers police marked traffic arriving on interdomain links to negotiated rates.

How do the leaf and border routers know what to mark and drop?

Allocation by Controlling Clouds and Boundaries

- A repository of of policy is needed to keep track of priorities and limits on QoS allocations for individual users, projects, and/or departments.

- An entity needs to receive requests for QoS, consult and update the database, and send configuration information to the routers, where indicated.

- Call this entity a bandwidth broker (BB) (Van Jacobson).BB is part of the network infrastructure and must authenticate requests from users. Some information can be configured.

- Intradomain policy decisions and implementations remain up to each domain. Share information about aggregates at boundaries.

- BBs in a domain may be organized in a hierarchy and/or for redundancy

Bandwidth Broker Background

- Van Jacobson has given a number of talks on BBs; slides of some are available at www-nrg.ee.lbl.gov/talks

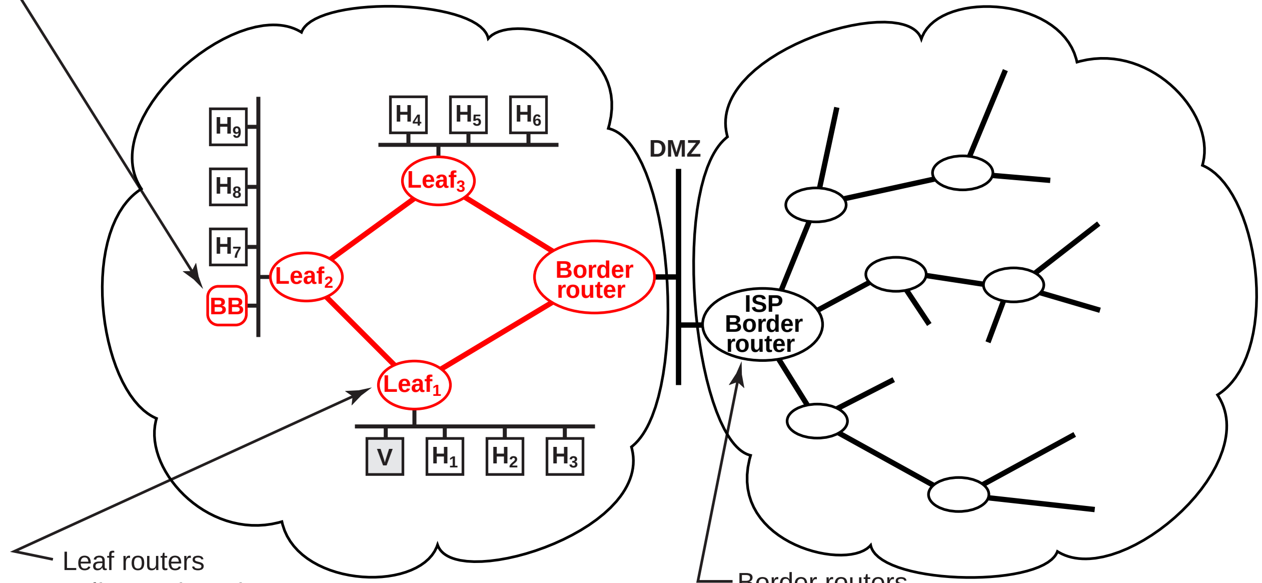
- In November ‘97, an Internet-draft with BB discussion was written. Though the forwarding path part has been superseded, it is available as informational RFC 2638 at ftp://ftp.ee.lbl.gov/papers/dsarch.pdf

- A talk on diffserv and bandwidth brokers by Jacobson given at the Internet2 QoS workshop is available at: www.internet2.edu/media/qos8.ram and the proceedings from that workshop are at www.internet2.edu/qos/may98Workshop/9805-Proceedings.pdf

- In the Internet2 talk, Jacobson made the case for BBs to allocate based on their smallest bandwidth link. He shows how the clouds should put such links at their boundaries

Bandwidth Broker in the Enterprise

Bandwidth Brokers tell
Leaf Routers what to mark
based on user requests, current
commitments and site policy.



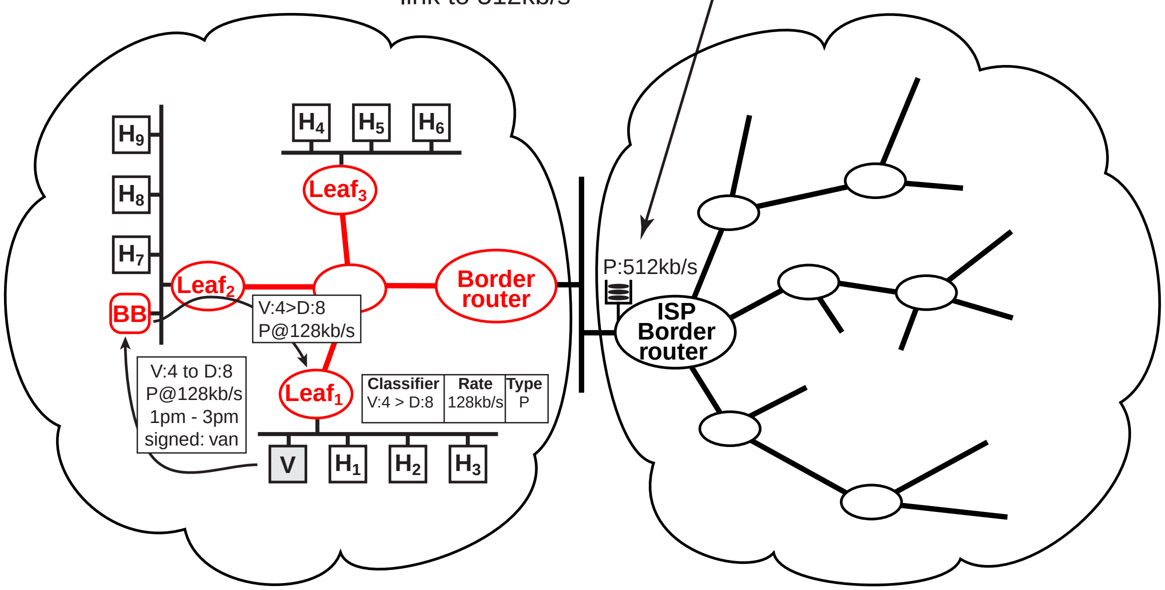
Leaf routers
police and mark
particular ***local*** traffic
based on tspec from BB.

Border routers
police marked traffic
arriving on interdomain
links to negotiated rates.

In a static or configured implementation, BB configures leaf/ edges with soft state information (COPS-PR?)

Requests to a BB and the Result

ISP BR will police "P" traffic arriving on this link to 512kb/s



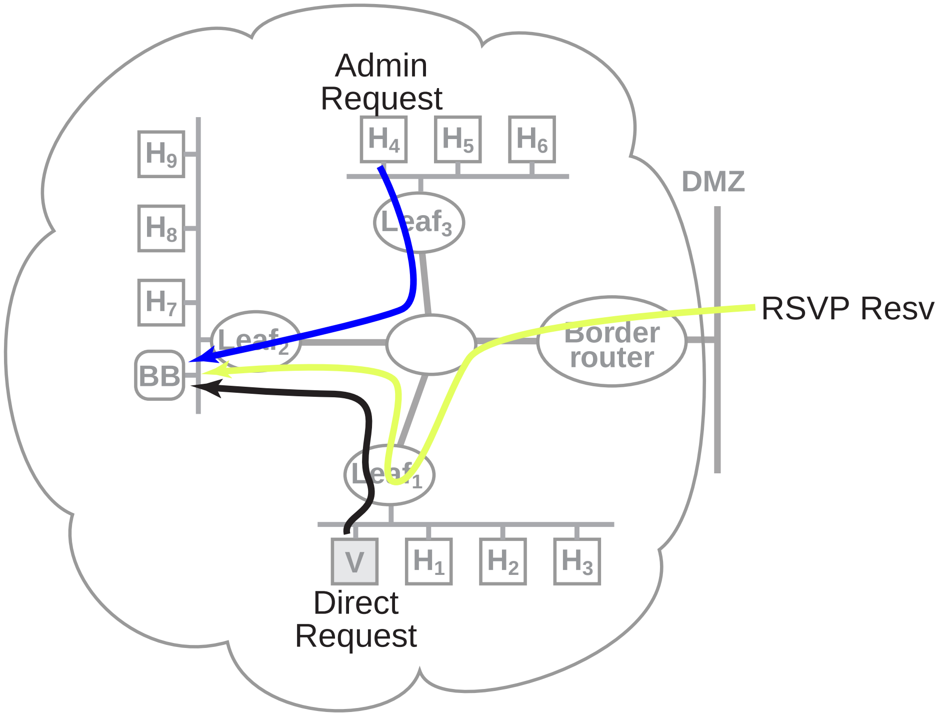
V:4 to D:8
P@128kb/s
1pm - 3pm
signed: van

V:4>D:8
P@128kb/s

Classifier	Rate	Type
V:4 > D:8	128kb/s	P

Here the BB responds to user requests

Requests Can Come from Many Sources



"Agnostic about signaling"

Voice in this Framework

Inside a high-bandwidth cloud, VoIP flows are trivial and not worth tracking

1. Assume configured EF at each network node sufficient to handle all calls
2. When a call is initiated, check if the destination is within the cloud
3. If so, just admit the call
4. May want to set up a classifer on the edge router to ensure no "spoofing"

If required, the VoIP flows within the cloud can be limited. The limit should be picked to keep the EF utilization of the cloud's smallest bandwidth links below some percentage. Then, in the third step above, check whether there is sufficient allocation for this call and, if so, decrement the allocation and admit the call

Voice between (Administratively Same) Clouds

Track connections only in the areas of limited resources, boundaries between clouds. For two bandwidth-rich clouds connected by a low-bandwidth link

- Assume the low-bandwidth link is configured for an EF rate that gives a sufficiently low probability of “busy” (bw_available)
- When a call is initiated, check destination
- If it’s in the other cloud, check: (bw_available - call_bw) >=0?
- If not, refuse call
- If yes, bw_available -= call_bw and proceed

Voice across Clouds

Locally, track connections only in the areas of limited resources, tail to “next cloud”.

- Assume the tail is configured for an EF rate that sufficient to handle all outside calls most of the time (bw_available)
- When a call is initiated, check destination
- If it’s “not-me”, check: (bw_available - call_bw) >=0?
- If not, refuse call
- If yes, signal/message “next cloud” and wait for reply
- If reply is positive, bw_available -= call_bw and proceed

Related Work on Resource Control

A Two-Tier Resource Management Model for Differentiated Services Networks: F. Reichmeyer et. al., ftp://ftp.isi.edu/internet-drafts/draft-rotzy-2-tier-management-00.txt

Discusses how the separation of the intradomain functions of and the interdomain functions of a bandwidth broker mean that the internal allocation method can be anything, including RSVP

Performance of QoS Agents for Provisioning Network Resources: Olav Schelen et. al., www.cdt.luth.se/~olov/publications. Presents topology-aware QoS agents which it presents as a type of bandwidth broker. Focus is on “path-sensitive admission control and maintains per-link resource reservations in a link state routing domain”. Starts with the assumption that per-flow admission and per-link state are required. Authors have implemented topology-tracking based on OSPF

Related Work on Resource Control (cont'd)

A Flexible Model for Resource Management in Virtual Private Networks: N. Duffield et. al., Sigcomm '99

This paper examines allocating VPNs based on the total ingress and egress amount of traffic of any one customer site. The "hose" abstraction is described as encompassing provisioning with no additional state in routers as well as keeping some "per-hose" state. Trace-driven simulations are used. Its allocation policy could be used by a bandwidth broker, but the concept of resizing dynamically lacks an incentive structure for both customer and provider

Parting Thoughts

Many questions are still open about building services that are “better effort”. There’s a desire for services that guarantee a minimum but allow for exceeding of this minimum when there is capacity in the network. This is a lot harder to do than folks simulating dumbbells think.

Analysis to give more realistic bounds on EF jitter would be nice to have (look at the probabilities between two adjacent packets). Analysis of the effects of permitting bursts in “guaranteed” traffic that would also be useful.

Allocation is in its infancy. It would be nice if we started simply and asked ourselves “what problem are we solving?”

QoS and its allocation are needed in order to express desired organizational policies. We should be focused on building the tools to make that possible.