Dumbbells, Dancehalls, and Beyond: Understanding QoS Prior to Deployment

Kathleen Nichols
kmn@cisco.com

IBAND 3

October 22, 1999

Outline

- One presenter's slant on Differentiated Services QoS
- Understanding performance: how to evaluate the evaluators
- QoS Holy Grails
- Simplicity and conservatism

QoS Contraints

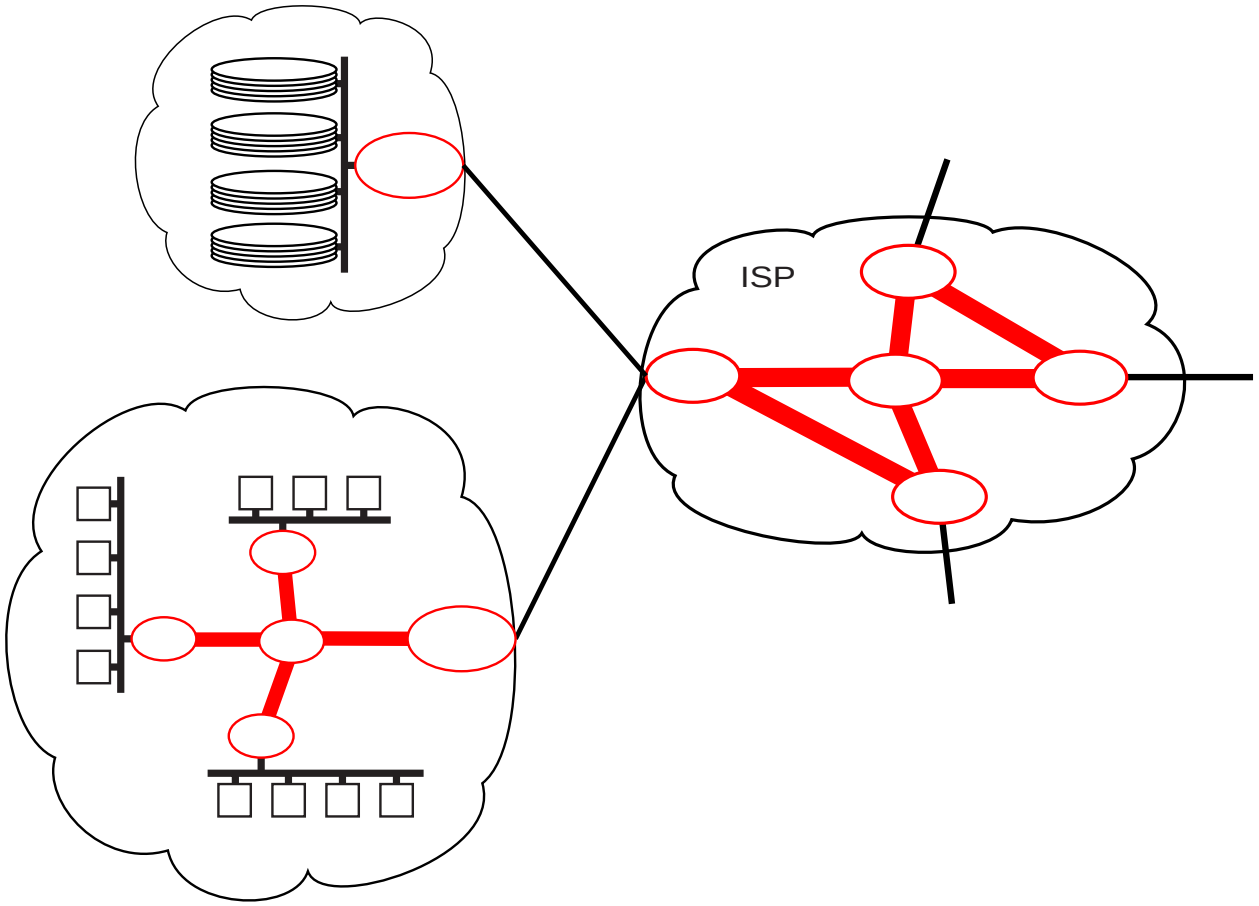
- For scaling, incremental deployment and economic reasons, don’t want to have to do everything everywhere.
- Need some bits in the packet header so upstream can tell downstream about this packet.
- The same bits in a packet may mean different things in different parts of the Internet. Need local control of their meaning.

Differentiated Services provides a framework for delivering QoS with these constraints. In this talk, we are discussing QoS in this framework.

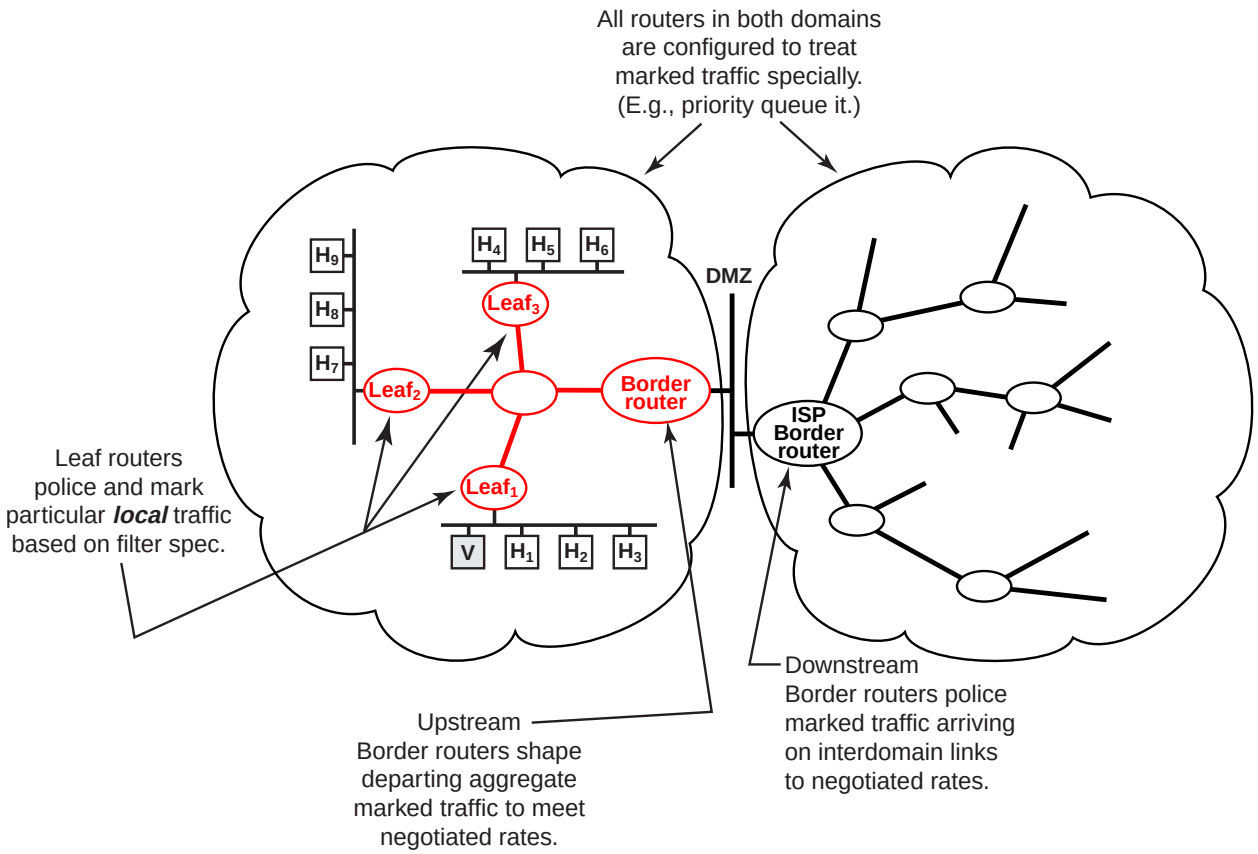
Differentiated Services in a Nutshell

- 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. See www.ietf.org/html.charters/diffserv-charter.html.

Local Control: Who Decides?



The Diffserv Architecture



Components of Diffserv QoS: Forwarding

In the evolution of the Internet, there have been two components:

1. The forwarding path where packets are handled at line rate, accessing a routing table for information on where to steer the packet, and

2. The control which configures that routing table. This happens at a much slower time scale, has and continues to evolve.

Analogously, divide QoS into its forwarding path and control plane behavior.

Forwarding path behavior provides the differential treatment an individual packet receives at forwarding time based on the information configured in a table by the control plane.

For the most part, forwarding path building blocks are well-understood and must be fairly simple as they are done per-packet at line rate.

Diffserv Building Blocks

- **Classification** — take apart (input) packet stream
- **Shaping —** alter temporal behavior to conform to downstream agreements
- **Policing** — ensure behavior conforms
- **Marking** — propagate information (packet identity)
- **Queuing** — isolate traffic stream
- **Scheduling** — construct (output) packet stream based on local policy & downstream agreements

Components of Diffserv QoS: Control

The control plane is used to implement a cloud's policy goals and to configure the forwarding path accordingly.

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

There are a number of proposals for configuring the forwarding path to create services (e.g., RFC 2475, RFC 2598, RFC 2638). Deployment of these will lead to experience that will guide more complex policies and allocations.

Scalability through Aggregation

- Fundamental to the diffserv approach is:
  - relatively small number of ways to handle packets
  - number of traffic flows requiring QoS relatively large
  - traffic subject to a wide range of rules devolved from policy
- Packets are grouped by the forwarding behavior they are to receive within a cloud: a “behavior aggregate (BA)”
- Flows are classified into aggregates and are “conditioned” to meet the rules of that aggregate. Per-flow state is kept at the edges of the network
- Nodes in the center of a network only have to deal with the small number of aggregates rather than keeping track of each separate traffic flow that passes through

Services in the Diffserv Framework

- 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
- Different user-visible services can share the same aggregate
- Services must be sensible and quantifiable under aggregation

Services and Aggregation

- Packets are not marked for the "services" their individual flows receive. Many services may use the same marking.
- As packets transit a network, their BA may be aggregated with other BAs to form a new aggregate. 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

Thus understanding how traffic aggregates is **crucial** to creating services.

About Allocation of 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 domain to meet technical constraints and policy considerations
- Traffic conditioners at boundaries as well as the mechanisms that implement PHBs must be properly configured to the allocation
- For services which cross domain boundaries, must communicate and specify the relevant data about the service between clouds.
- (Initially, the information that crosses boundaries is expected to be simple and static or quasi-static)

Understanding Diffserv Performance

Start by understanding how services will work in the forwarding path.

For a technology that was standardized less than a year ago, there is an impressive quantity (growing daily) of 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.

Evaluating the Evaluators

Most published studies use “dumbbell” topologies and a traffic model consisting *only* of long-lived TCPs with identical RTTs.

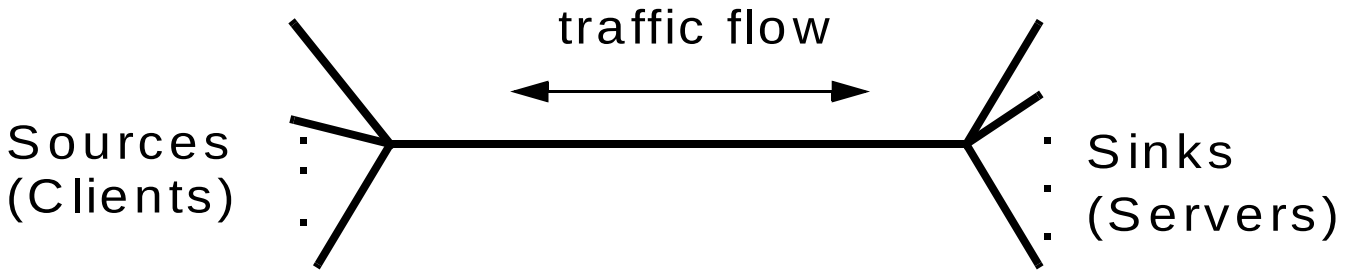
Aggregation issues are invisible in these set ups.

Questions to ask:

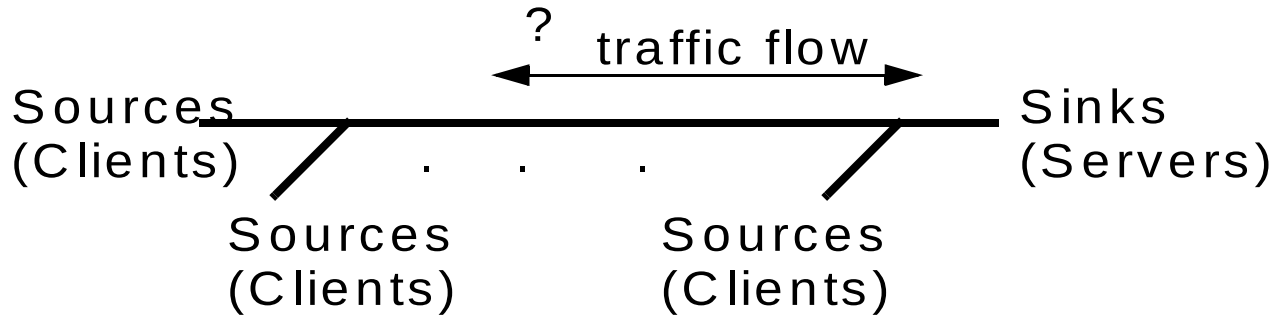
- Were good **simulation practices** used? (number of events in a run; randomly vary starting conditions)
- What **topology** captures the salient features of a realistic network?
- Do the **traffic models** capture the salient features of real loads?
- Were **traffic loads** sufficient to exercise a range of operating points?

Topology: What kind of network?

Dumbbells and Dancehalls

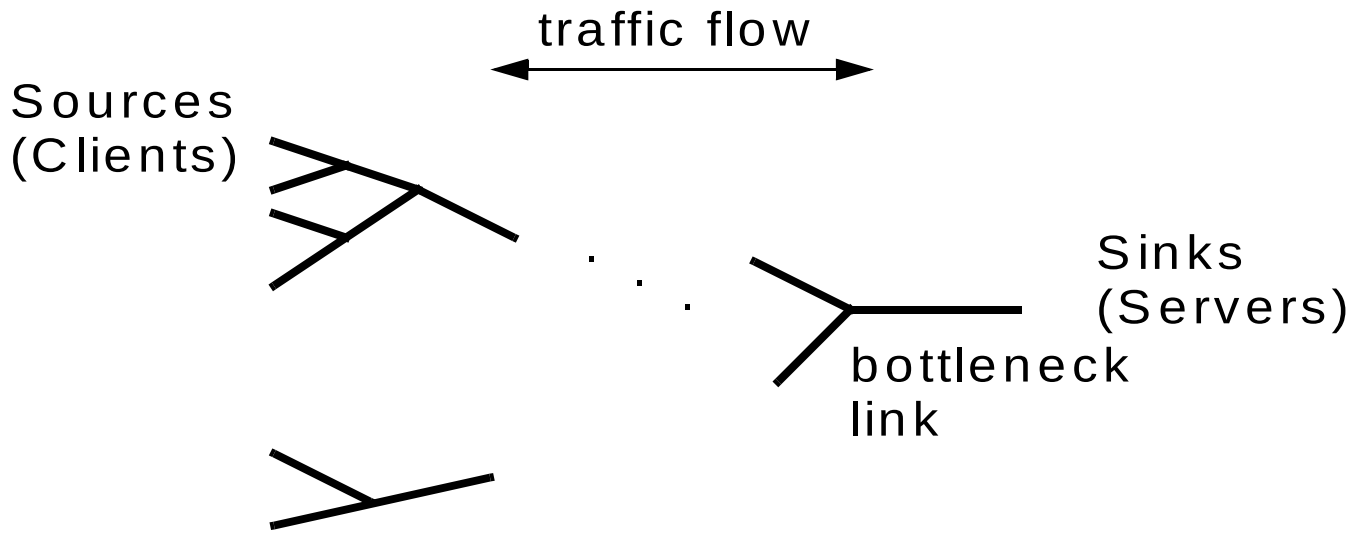


Freeway with On-ramps

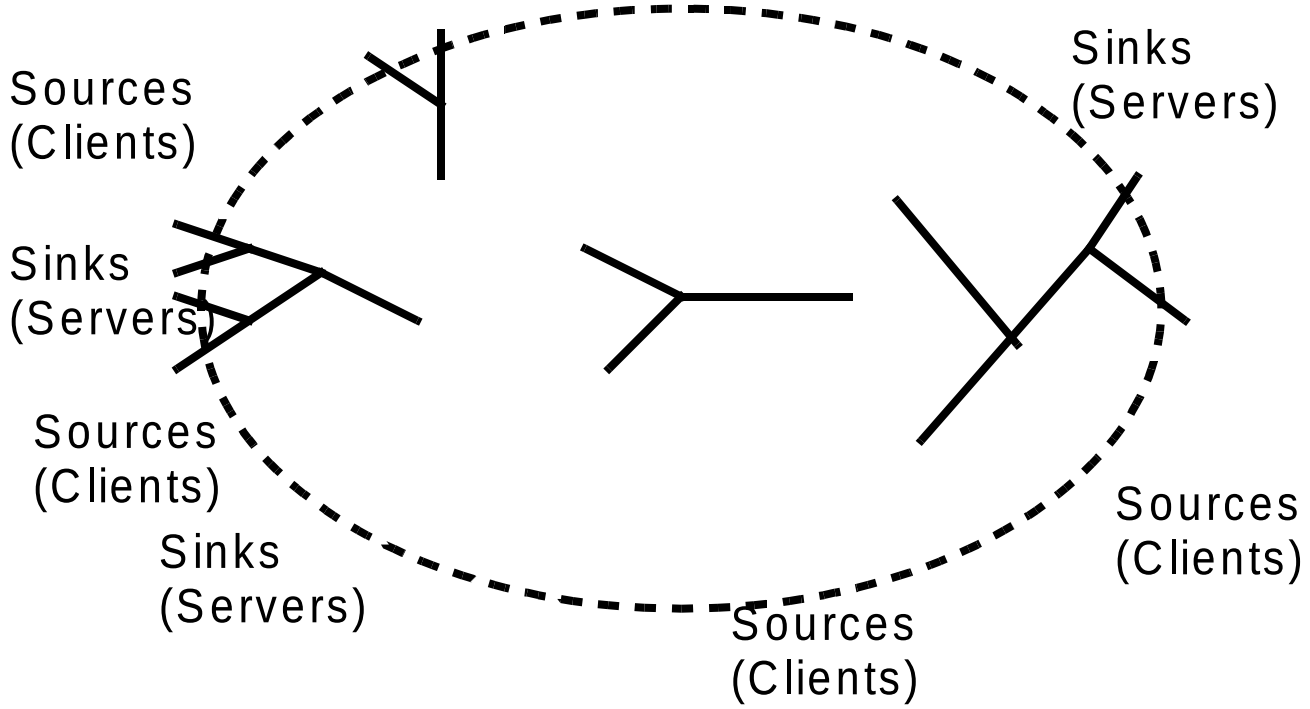


More Topologies

Branch-y (Trees)



Branch-y and Mesh-y



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

This makes it less appealing for those who want to produce results in a short time period.

Traffic Models: Many Ways to Go Wrong

Simulator models often oversimplify any or all of: the application model, the TCP protocol model, packet size selection.

Most published differentiated services simulations used ns-2's basic TCP which uses one size for all packets, does not include the SYNs and FINs of TCP connection set-up and tear-down, and does not permit DATA to flow in both directions. Dropping a SYN packet is different from that of dropping a DATA packet.

Worse yet, some simulations rely on "open-loop" source models that look nothing like real traffic under congestion.

TCP's feedback leads to network behavior which is time-varying and buffer dynamics that will vary with transfer sizes.

Traffic Loads: Reality Check Needed

Most published studies use long-lived TCPs (steady state). Some add CBR as “bad” flows to be restricted relative to the LL TCPs.

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

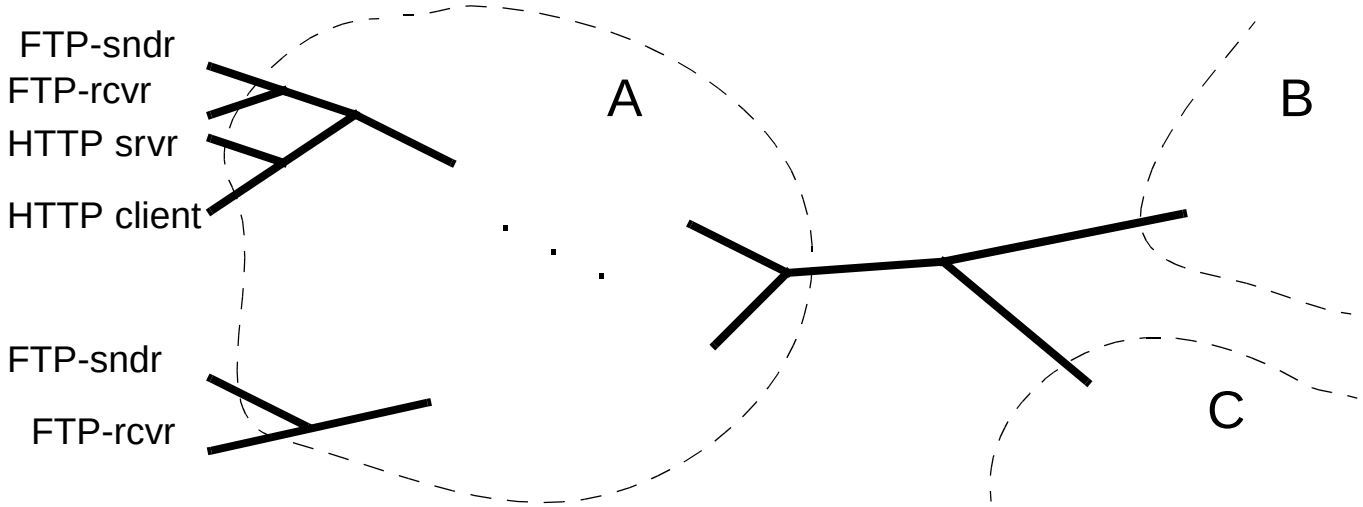
- Measurement studies have found the distribution of file sizes fetched by HTTP transactions are Pareto distributed with most objects fitting into a single packet.
- When a user downloads a web page, typically many small transfers take place. Such flows have different characteristics from a long-lived TCP (like FTPing a large file)

What we know about Internet traffic is that it will change over time, so wise to design for and evaluate over a range of operating points.

Result: Per-flow Fallacies

The use of dumbbells and long-lived TCPs obscures network dynamics and aggregation effects. Perpetuates myth of “single flow”, i.e., a long-lived TCP as a representative of “the customer”.

End up solving the wrong problems.



At each link above: who’s the “customer”? What’s “fair”?

QoS “Holy Grails”

“Better Effort”: A service that has a committed rate (may be zero) for which resources are allocated, but bursts of packets are permitted above that rate

“Virtual Leased Line”: A service that looks like a wire

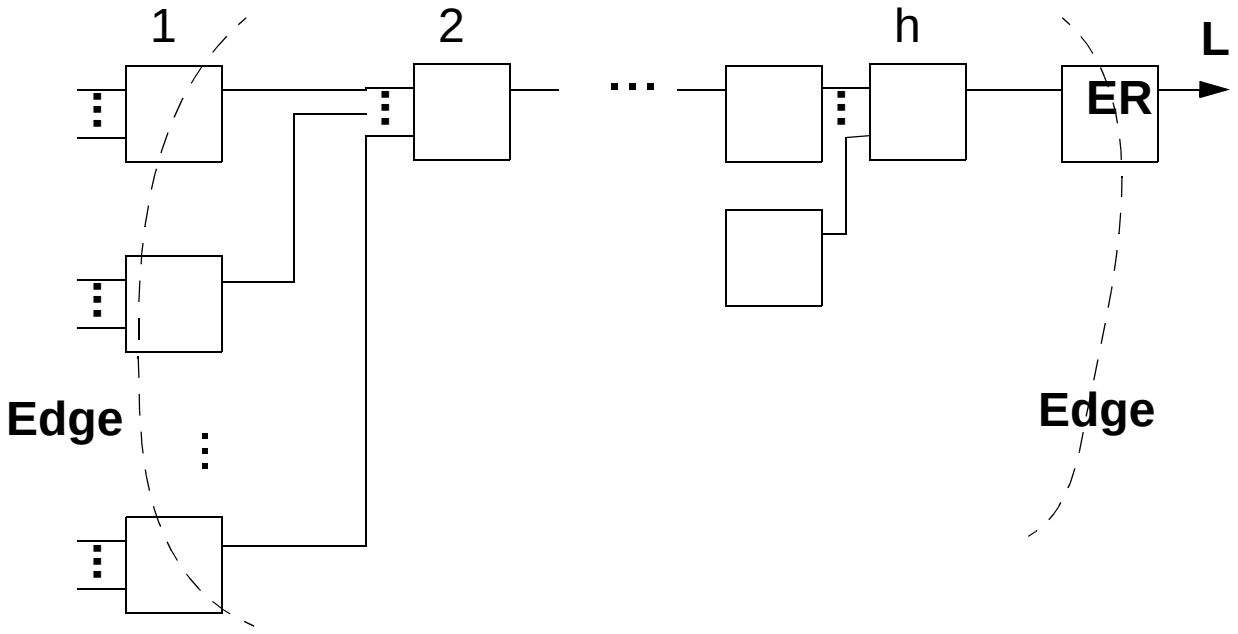
The first is what everyone wants to market, but it’s harder than it looks. The second is achievable, but requires conservatism.

- To build these services, we need a PHB and rules that govern the BA at ingress and egress with respect to temporal behavior (e.g., rate and burst)
- Since aggregation behavior is critical, start by understanding how clumps of packets can occur in a BA.
- Hypothetical “SP” PHB: how large a clump of packets can appear in the network?

Clump of “Special” (SP) 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** SP packets. The ingress BAs form a tree toward link **L**. How large a clump of SP packets can appear at link **L**?

If each input has an allocation, then clump(b,i,h) = $b * i^h$



Constraints

The entire egress link can be used by the SP traffic for the period it takes to transmit the clump of $b * i^h$ packets at rate L though ingress edge meters are set up to mark SP at rate $a*L/i^h$ $(a<1.0)$, burst b

If $L < B$, must buffer clump*(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 clump; at 10, 90%; at 100, 99%.

If the total allocation is restricted to a number, $n < a*L/r$, where r is the individual rate allocation, then the maximum clump is limited to $b*n$ packets.

For $b=1$ there is no loss if buffer n packets.

The larger b is, the more packets of the SP aggregate can be delayed and displaced in time.

Considerations for Rules

Restrictions on allocation:

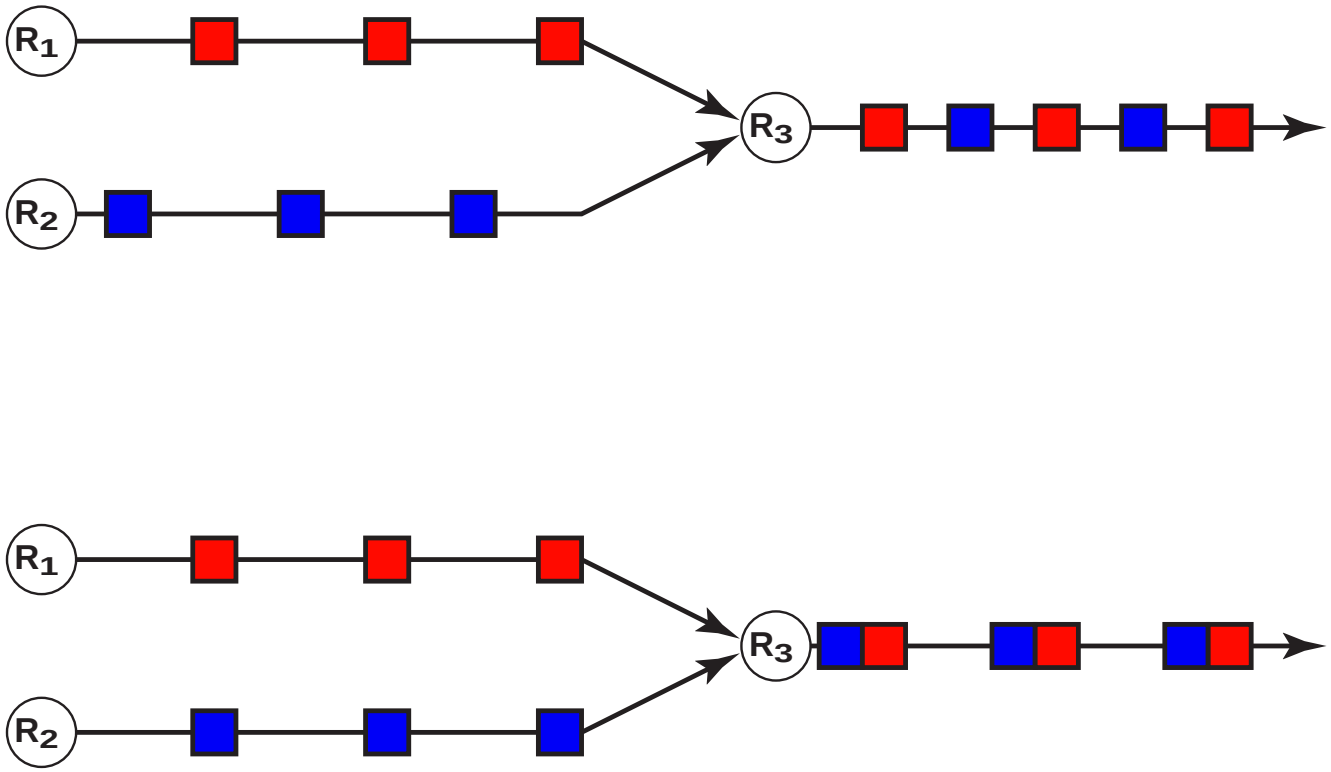
- total rate allocated < a*L
- n < a*L/r, where *r* is the “subscribed rate” for an input
- also N < i^h since no more than one SP BA per ingress link
- so a*L < r*i^h

If *L* is in packets/time, then for *b*=1, the SP burst can take up *a*n/L* time on the link. There **must** be no SP packets for *(1-a)*n/L*

The larger *a* is, the more packets of the SP aggregate can be delayed and displaced in time. But for *b*=1, this is bounded by *a*n/L* (ignoring effects of packets in other aggregates)

If *b>1* and *a*b>L*, a “better best effort” service must deal with excess packets in an aggregatable way

Even CBR Streams can form Clumps



Data traffic is even more interesting: consider a server's response to a web transaction!

Chasing the Holy Grail: Marking Packets for Drop Preference

Dave Clark of MIT proposed a service based on RIO droppers and edge marking by “time sliding window” markers in “Explicit Allocation of Best Effort Packet Delivery Services” by Clark and Feng, in IEEE/ACM Transactions on Networking.

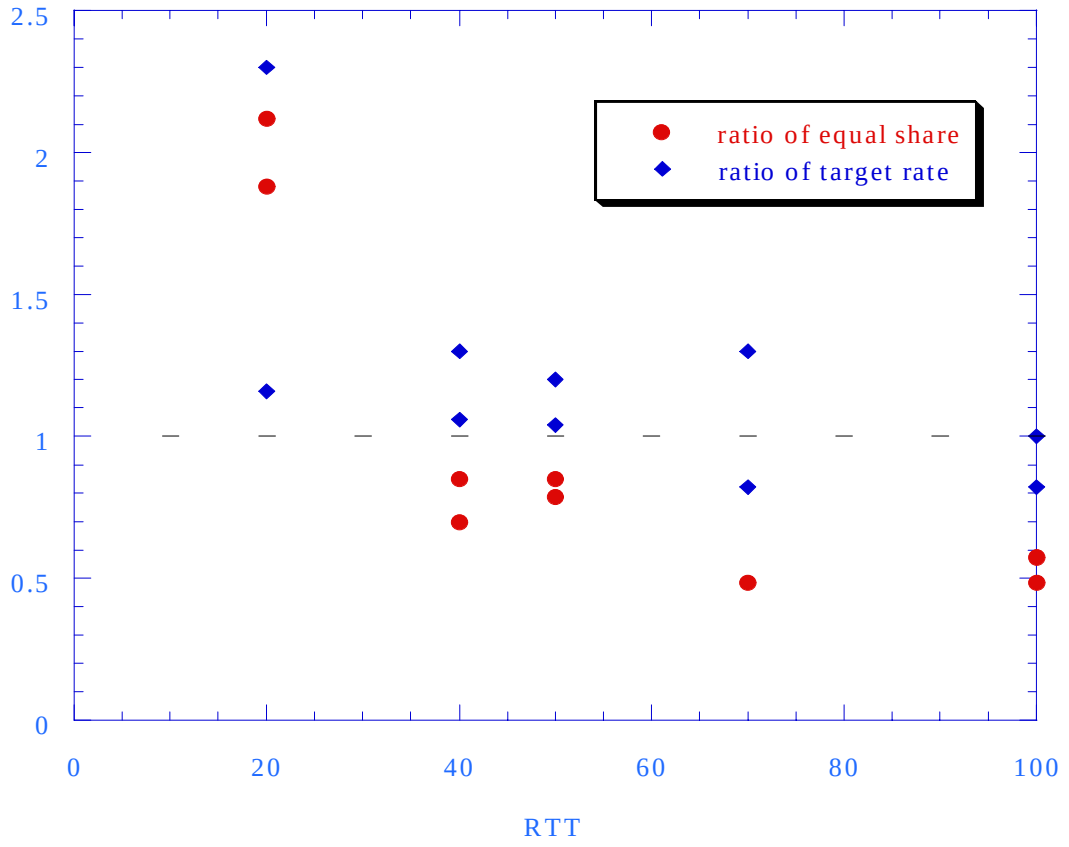
RIO droppers use two ‘93 RED algorithms running on the same queue. For “outs”, dropping starts at a lower average queue size and has a higher probability.

Table 1 showed 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). For better understanding, I plotted this table.

Achieved-to-Target vs. RTT

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

Diamond/blue are the RIO results divided by target rate



What do the Results Mean?

Does use of “ins” really overcome RTT effects? Not conclusive, clear, or quantifiable. How can a service built from this be quantified against its subscribed rate?

How do these drops effect TCPs?

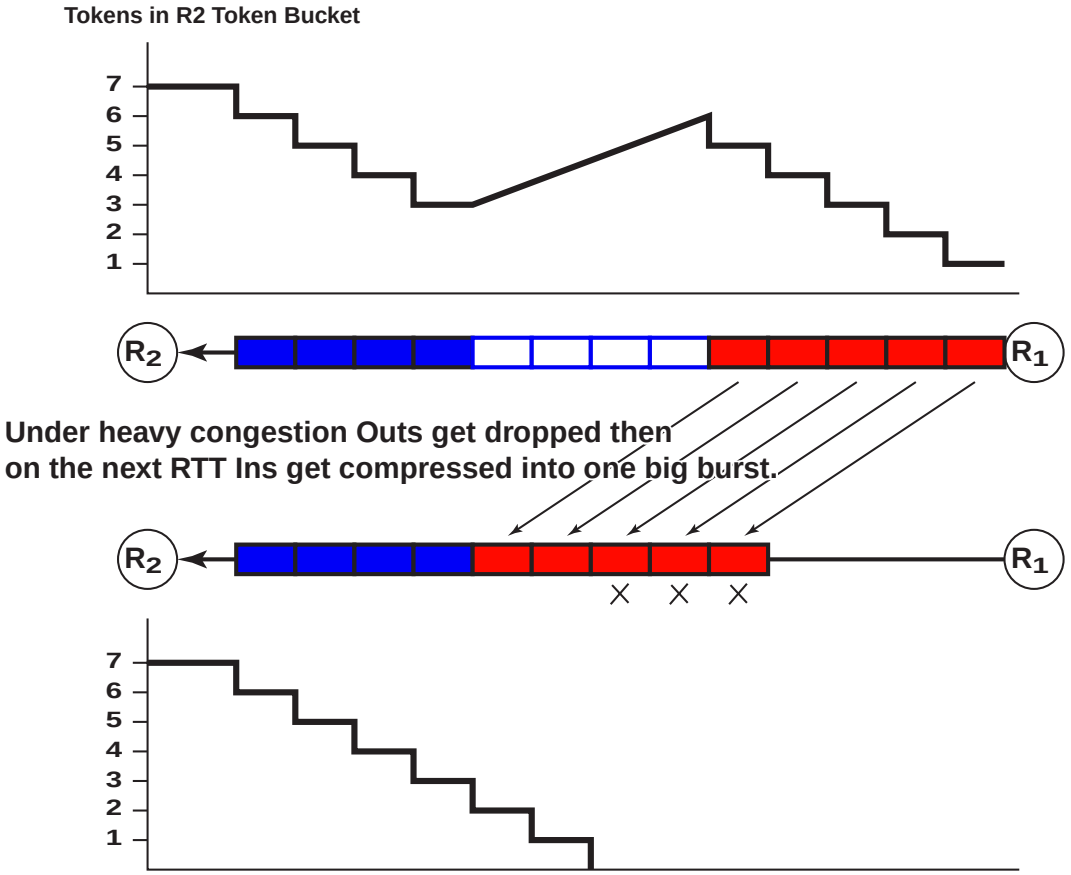
What if more experiments were run? (Is the point at 20 ms anomalous?)

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

Can a non-responsive flow take over?

Is it possible to employ DP to build a service which aggregates well?

How Does DP Aggregate?



The red “in” packets can become “outs”: this is not aggregatable behavior. On-going evaluation work thus far employs simplistic models.

Simplicity and Conservatism: The EF PHB and VLL Service

Expedited Forwarding (EF) PHB (RFC 2598)

In simple terms, a rough equivalent of priority queueing, 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. "

(Where "configured rate" means the configured rate of the EF aggregate for that output link)

Implications of the EF Definition

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

The worst case displacement (which can lead to jitter) 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 another aggregate at every hop along its path.

To see this as jitter, one packet of a stream must get worst case displacement, the next packet must wait for *no* packets

A Virtual Leased Line Service

RFC 2598 proposes a “virtual leased line” (VLL) service built from the EF PHB combined with border policers that drop packets exceeding a peak rate. Bursts are not permitted.

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

These rules make it easy to aggregate the service: peak rates just add.

A VLL “looks like a wire” to the application or aggregate.

Its properties of low jitter and guaranteed rate, make VLL a good means to provide QoS to VoIP. However, it is intended to provide the equivalent of a dedicated link, useful for *any* mixture of network traffic, potentially useful for VPNs.

VLL's 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 (sufficiently sized holding buffers and good queue management), the VLL acts like a dedicated link at the subscribed rate. (http://www.nren.nasa.gov/workshops3.html)

For VoIP using VLL, care about jitter. 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.

Simple Analysis to Bound EF Jitter

Earlier, noted the two components of jitter: assume a worst case for both the displacement from other EF microflows and a per-hop wait for a full MTU-sized packet of another aggregate at every hop.

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, ($|(a_j-a_i)-(d_j-d_i)|$). The worst case jitter is the worst case displacement if the next packet of the stream doesn't queue. (Very unlikely!)

Assume the bottleneck bandwidth, **B**, is the bandwidth of every link in the domain. Worst case is an allocation, **a**, of 50%. Compare to a 30% allocation.

This is an excessively pessimistic bound, but gives some feel for what conditions have to occur to have jitter exceed one MTU at the subscribed rate.

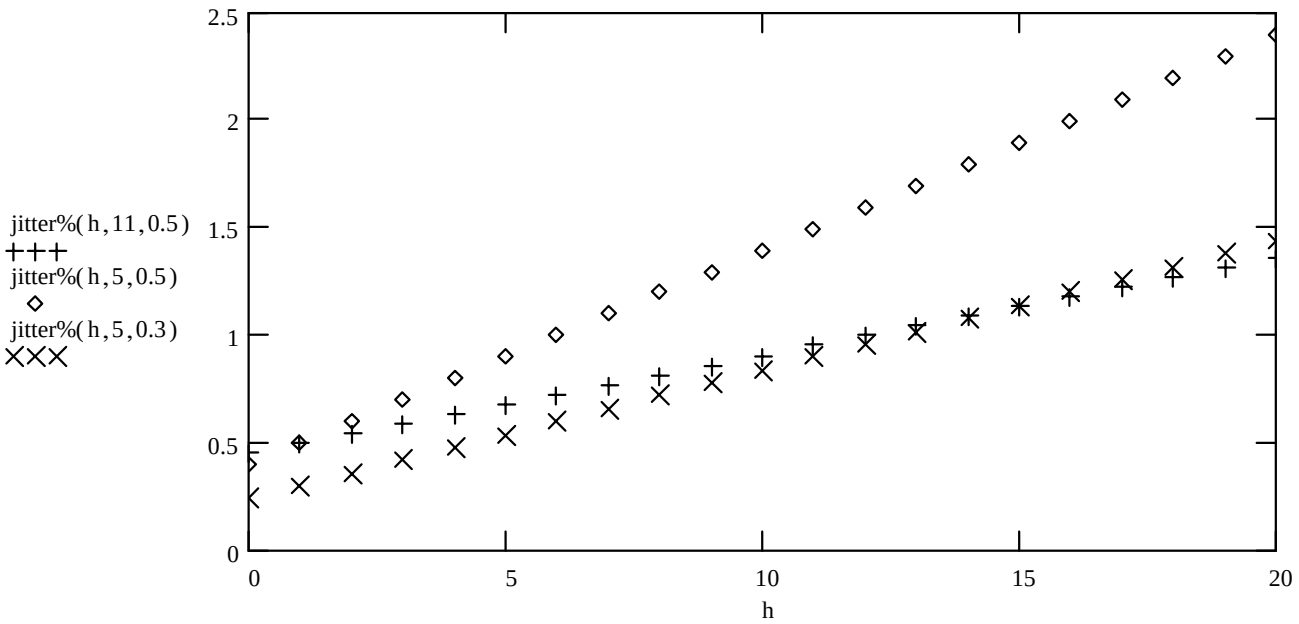
Bounding EF Displacement

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

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

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

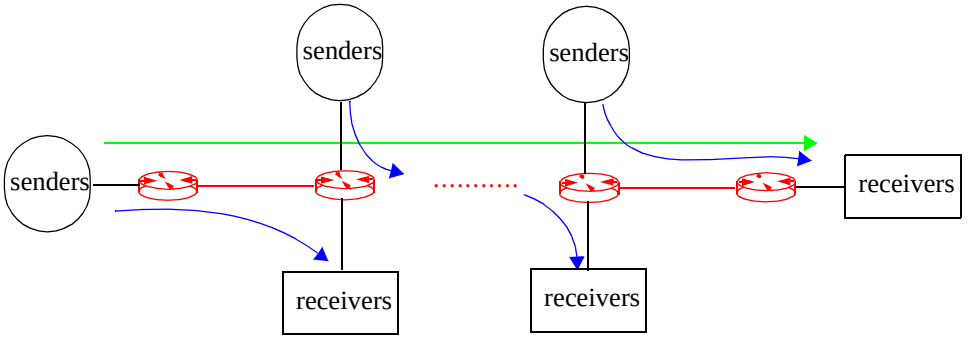
The maximum allocation of a=0.5 gives the largest jitter



Evaluating Jitter for VoIP using VLLs

These simulation results focused on a VLL over multiple hops (from Yonghwan Kim of Cisco).

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

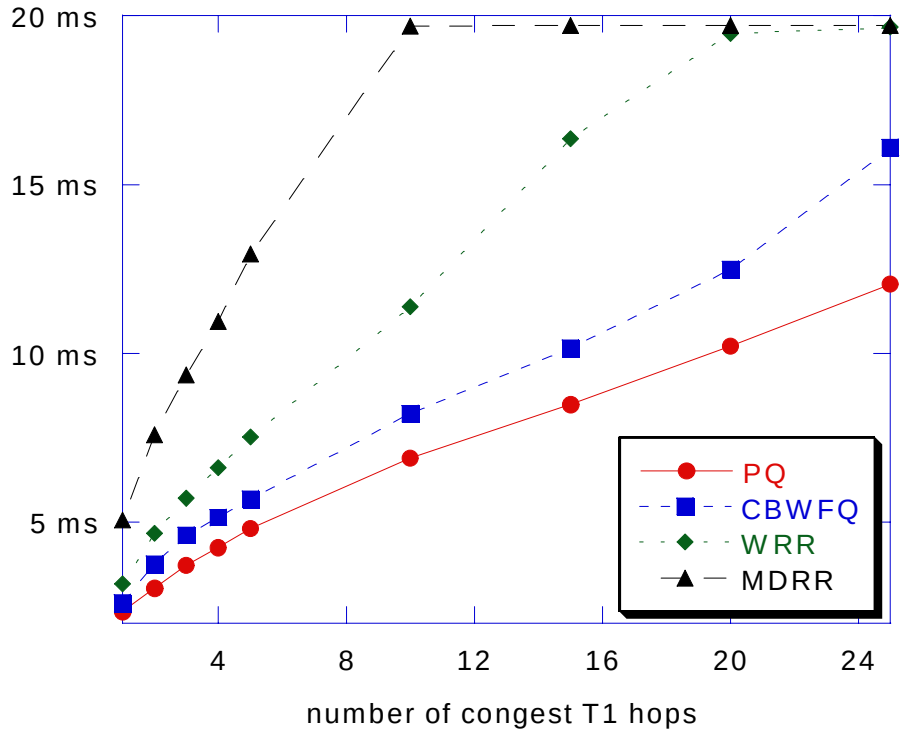


The “VoIP” flows were 24Kbps, 20 ms between 60 byte packets

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

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

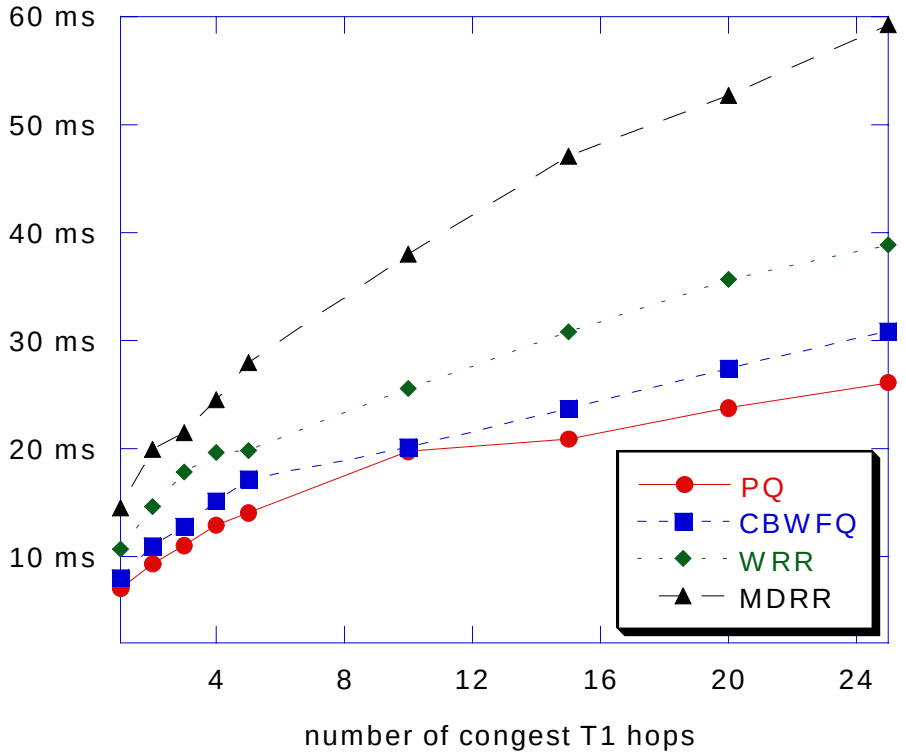
Median Jitter of VoIP using VLL



The PQ case corresponds to the simple analysis.

An MTU at 24 Kbps is 500 ms so the median value is no where near that.

95th Percentile Jitter of VoIP using VLL



PQ jitter is bounded at slightly more than the cycle time of the VoIP stream.

Parting Observations

Marketing folks have driven a demand for a service that guarantees a minimum that can be exceeded when there is capacity in the network. This is a lot harder to do than folks simulating dumbbells think.

Must understand the behavior of traffic under aggregation before we start to develop a lot of PHBs and services. Fortunately, if we stay simple, we can deliver a quantifiable VLL with current knowledge.

Flow-based thinking hampers deployability by introducing complexity and limiting the ability of solutions to scale.

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.

Words to Live By for QoS

"If you expect to see the final results of your work you simply have not asked a big enough question." I.F. Stone