Differentiated Services on the Internet

CS 851 - Statistical Multiplexing

Nicolas Christin

christin@cs.virginia.edu

Outline

- Differentiated Services on the Internet
  - Explicit Allocation of Best Effort Packet Delivery Service, *D. Clark and W. Fang*
  - A Two−bit Differentiated Services Architecture for the Internet, *K. Nichols, V. Jacobson, L. Zhang*
- LIRA: How to Reconciliate DiffServ and IntServ
  - LIRA: An Approach for Service Differentiation on the Internet, *I. Stoica and H. Zhang*
- Proportional Differentiated Services
  - Proportional Differentiated Services: Delay Differentiation and Packet Scheduling, *C. Dovrolis, D. Stiliadis, P. Ramanathan*

State of the Art in Differentiated Services

- Clark, Jacobson: two complementary approaches that were the starting point for DiffServ.
- Resulted − after modifications − in an IETF draft.
- Clark: *expected capacity*
  - "Different levels of best−effort service in times of network congestion"
- Jacobson: *premium and assured service*
  - Defines three classes of service (premium, assured, best effort)

Clark's Framework

- Right now: best−effort service, therefore unpredictable at a congestion point (relies on the total capacity of the congestion point, regardless of the traffic)
- Expected capacity: in times of congestion, all connections should slow down to an expected rate
  - Different users => Different expectations => Different levels => Different rates
- Push the complexity at the edges of the network (only simple modifications to the core routers).

Sender−Based Scheme

- The expected capacity framework:

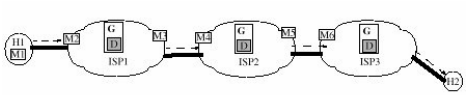


Figure 1. the "Expected Capacity" framework (sender-based)
Host 1, H1 has a sender-based profile, and is sending traffic to host H2 (dotted line). The traffic traverses three ISPs. The gateways, G's, in the figure are all augmented with preferential dropping algorithms, Ds. There are profile meters, Ms, at each interface between a customer and an ISP, or between two ISPs. M1 is a profile meter inside a host; M2 is on the access link from H1 to ISP1; M3-M6 are profile meters on the boundaries of ISPs.

Sender−Based Scheme (Cont'd)

- A meter exists at the entry of the network (ie, right at the connection point of the user).
- This meter tags the packets as being *in−profile* or *out−of−profile* according to the profile defined for the user.
- The gateways along the path drop then the packets differently if they are *IN* or *OUT*.
- Need to design the meter carefully: the traffic can be really bursty=> more complexity

Issues in Defining a Profile

- Traffic Specs: what is exactly provided to the customer (e.g., bandwidth)
  - Guaranteed rate
- Geographic Scope: to which destination(s) does this service apply?
  - Set of IP addresses
- Statistical Assurance: with what level of assurance is the service provided?
  - Packets losses

Two Types of Assurance

- Absolute: needs some reservation of the path, hard guarantees: more IntServ−like.
- Low probability of failure: approach adopted by Clark and Wang.

Receiver−Based Scheme

- Problem with Sender−Based: what if the sender cheats? => need for another scheme, based on the receiver (e.g., the ISP)

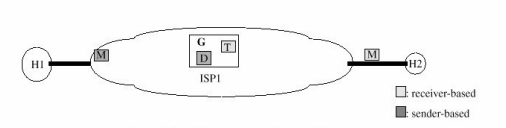


Figure 2. Simplified "Expected Capacity" framework

The simplified framework to illustrate the different levels of service provided for bulk-data TCP transfers. Both sender-based and receiver-based are shown. The darkly shaded boxes, profile meter M, and dropper D are for sender-based control. The lightly-shaded boxes, tagger T in the router and profile meter M at the receiver side, are for receiver-based control. G are all the gateways in ISP1. There are no cascaded profile meters.

Receiver−Based Scheme

- Uses an add−in to TCP: the *Explicit Congestion Notification* (ECN) bit
  - The ECN bit is set by the gateways if the network is loaded (in a normal scheme, they would drop the packet).
  - This ECN bit is then copied in the acknowledgement packet to inform the sender that it must slow down.

Receiver−Based Scheme (Cont'd)

- The receiver chooses then to turn off the ECN bit if the packet remains in profile or to leave it on if the packet is out−of−profile.
- No packets are actually dropped in this scheme. It relies on the "good will" of the sender that has to slow down if it receives some acknowledgements with the ECN bit on.

Sender−Based vs. Receiver−Based

- Advantages of sender−based:
  - Explicit treatment (i.e., possible dropping) of packets
  - Distinction between in−profile and out−profile packets is straightforward.
- Advantages of receiver−based:
  - No possibility of cheating for the sender
  - Conveys dynamic information about congestion level along the path

The Dropping Algorithm

- Need to keep it simple (scalability) but general enough so that it will not be modified over time
- Need to efficiently use TCP: two possible behaviors when some packets get dropped
  - Fast−Recovery: only a couple of packets got dropped. TCP cuts its window size in half, and then increase it by one packet every round−trip time.
  - Slow−Start: occurs when a large number of packets got dropped. TCP waits for the retransmission timer, the window size is set to one packet and increases linearly after a threshold. => **SHOULD BE AVOIDED!!!**

RIO and RED

- RED: Random Early Drop
- RIO: Random Early Drop with In/Out Bit.
  - RIO actually uses twin RED algorithms, one for packets that are in−profile, one for packets that are out−of−profile.

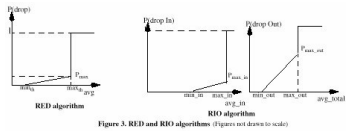


Figure 3. RED and RIO algorithms (Figures not drawn to scale)

The TSW Tagger

- TSW: Time−Sliding Window.
- Two components:
  - A rate estimator used to smooth TCP's inherent burstiness. It estimates the rate upon each packet arrival and forgets the past history over time. The goal is to have a rate estimator that is independent of the speed of the TCP connection itself.
  - A tagging algorithm

The TSW Tagger (Cont'd)

- Two approaches are possible for the tagging algorithm:
  - Use a long past history (one full sawtooth between 0.66Rt and 1.33Rt, Rt is the target rate) and tag packets as being out when: $\frac{(avg_{rate} - Rt)}{avg_{rate}} > 0$
  - Use a short past history of one RTT and look for a sawtooth exceeding 1.33Rt => the packets are then tagged as being out.
- The second approach is useful when the tagger is close to the host

The TSW Tagger (Example)

- The actual algorithm used is:

Initially:

Win_length = a constant;
Avg_rate = connection's target rate, R_T;
T_front = 0;

Upon each packet arrival, TSW updates its state variables as follows:

*Bytes_in_TSW = Avg_rate * Win_length;*
New_bytes = Bytes_in_TSW + pkt_size;
Avg_rate = New_bytes / (now - T_front + Win_length);
T_front = now;

Whereas, ***now*** is the time of current packet arrival; and ***pkt_size*** is the packet size of the arriving packet.

Figure 4. TSW algorithm

Issues in designing TSW/RIO

- Different RTT imply different behavior for the TCP window => closer connections are able to recover more quickly.
  - Need to know the RTT to achieve fairness, or to guess it
- Need to avoid tagging a full cluster of packets as being out in order to stay in FastRecovery mode
  - Need to use a probabilistic function to space the packets being tagged out

Jacobson's Approach

- Proposes three different levels of service:
  - Premium Service
  - Assured Service (similar to Clark's approach)
  - Best−Effort Service
- Two−bit architecture:
  - Packets gets differentiated by two bits in their header (e.g., the IP Type Of Service (TOS) byte)
  - Premium bit (P−bit)
  - Assured Service bit (A−bit)

The Premium Service

- Hard−limited to its peak provisioning rate
- Shaped to prevent the traffic bursts from being injected into the network
- Specified by a desired peak bit−rate
- Equivalent to a virtual leased line

The Premium Service (Cont'd)

- How does it work?
  - The first−hop router (leaf router) turns the P−bit on and shapes/smoothes the traffic
  - Modifications needed in the routers: send Premium packets first, and the rest after
    - This implies two levels of priorities, and possibly two queues in the routers to avoid sorting.
  - The policy is set up at the edges of the network with a token−bucket mechanism

The Premium Service (Cont'd)

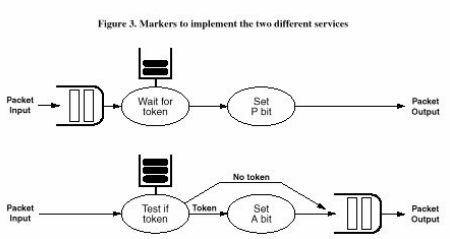
- What does happen when the packets exceed the rate? Two possibilities:
  - Discard them.
  - Downgrade their priority: this is actually bad since it can enttail out−of−order packet delivery.
- => Need to correctly size the queues of Premium packets to prevent them from being dropped
- Low delay variation than Clark's approach if the queue is correctly sized.

The Complete Framework

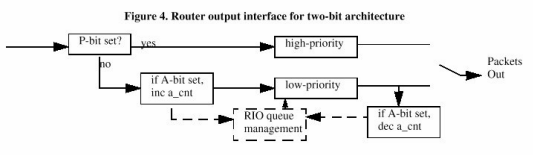
- The routers shall provide three different types of service (Premium, Assured, or Best−Effort)
- Packets pass through markers: their A and P−bits are cleared, then set if the flow is in conformance with its profile
- The P−bit is turned on possibly after the packet has been hold to remain in conformance with the peak−rate: *Non−work conserving approach*
- This shall be handled at the leaf router

The Packet Marker

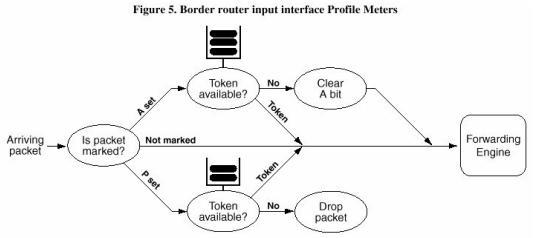
- Described by the following picture:



Output Configuration of the Router



The Border Router



The Implementation

- Leaf routers need a flow specification and a general classifier. The flow specification can be assured by a signaling protocol such as RSVP or SNMP.
- Bit−pattern classifier
- Bit−setter
- Priority queues
- Shaping (Token−Bucket)
- Policing (Border routers)

The Marked Traffic

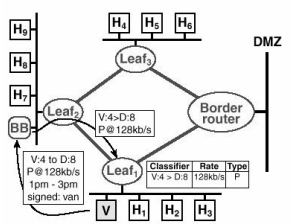
- Two ways of allocating the level of marked traffic throughout the Internet:
  - Provisioning: static guarantees
  - Call set−up: flow specifications are used for a given flow.
- The scheme can evolve into a more complex architecture if needed
  - Hierarchical filters to differentiate the flows among the different levels of service provided.

The Bandwidth Brokers

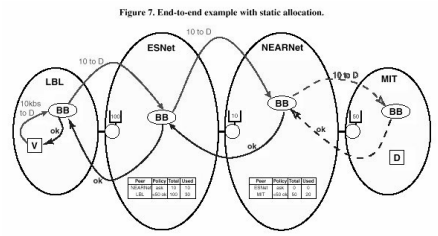
- Needed to automate the differentiation
- Two functions:
  - Setting up the local routers (leaf routers) to allow the differentiation.
  - Managing the messages passed at the edges of the network (ie, domain boundaries)
- The user "talks" with the BB to set up the property of a given flow. The BB is actually a kind of policy server (possibly using authentication methods)

The Bandwidth Brokers (Example)

- Let us consider the following example:



The Complete Architecture



Conclusion of the first part

- We have shown two complementary schemes.
  - The second is more general, and uses the results of the first one. Notion of different levels of importance => has evolved into a concept of classes.
- Share some characteristics with IntServ, but:
  - less overhead: the signaling is not performed on a per−hop basis, but only at the edges of the network
    - Interesting in the sense that the core routers should not be affected
  - works on aggregate traffics, and *not* on a specific flow in general (contradiction with Jacobson's analysis?)

Towards Proportional Differentiated Services

- The initial approach was perhaps too greedy
  - "any" destination in the Internet is still a problem− depends on the RTT: need to evaluate it.
  - Or other problem: underutilization of the link in case of large spatial granularities.
- New approach: provide proportional differentiated services
  - LIRA, PDS

LIRA

- Takes Clark's approach as its starting point:
  - "The key difference between Intserv and Diffserv is that while Intserv provided end−to−end QoS service on a per−flow basis, Diffserv is intented to provide service differentiation among the *traffic aggregates* to different users *over a long timescale*"
- LIRA=Location Independent Resource Accounting.
  - Main goal: alleviate the "any" destination problem

The Big Picture

- Problem: The network needs to provision resources to *all* destinationsbecause it does not know where the packets shall go => difficult to support a fixed bandwidth profile and a high utilization of the link at the same time
- LIRA's idea (based on shaping): Use a token−bucket to determine which packets can go through

The Resource Token Bucket

- Idea: the number of tokens needed to admit a preferred packet is a dynamic function of the path it traverses
- Example of function: the number of tokens can be equal to the cost of the link, defined by: $C = \frac{a}{(1-u(t))}$ with $u(t) = \frac{R(t)}{C}$, C being the capacity of the link
  - The cost function has to diverge when the utilization tends to 100%, and has to be a constant "idle cost" when the link is not utilized.

The Resource Token Bucket (Cont'd)

- Problem with the previous function: not adapted to a real environment:
  - Such a function has to be computed periodically because of the overhead.
  - Therefore, results might be obsolete when they are used!
  - Critical problem, since the function diverges and the link utilization can drastically change in a short amount of time

A "Better" Cost Function

- The solution: use an iterative formula to minimize the variations
- Advantages: $c(t(i)) = a + c(t(i-1))\frac{R(t(i),t(i-1))}{C}$
  - Defined even if the link is congested.
  - Smoother when the link utilization approaches 100% (increases by *a* every iteration).
- Problem:
  - Actually too smooth! When the link becomes congested, only a couple of iterations should be needed, otherwise we'll need to drop tagged packets. Solution: use $C' = \beta C$ for the marked traffic

Multipath Routing in LIRA

- Need to ensure that all packets belonging to the same flow are forwarded along the same path.
- Otherwise: possible out−of−order delivery of the packets => bad performance, increases overall instability.

Multipath Routing in LIRA (Cont'd)

- Solution: the "XOR" scheme:

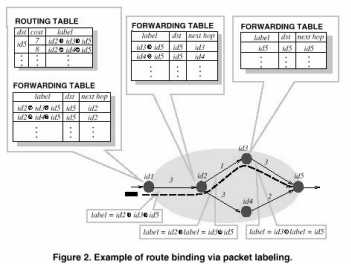


Figure 2. Example of route binding via packet labeling.

Path Selection

- How to choose a route?
  - Use a probabilistic function to select a route:
  - Each time a cost is updated, the routes are split in two sets:
    - Higher cost
    - Lower cost
  - Then the probability of picking a route in the lower cost set is increased by δ.
  - Converges to the optimal solution within δ in a steady−state system

Route Aggregation

- Generalization of the previous algorithm for *egress nodes* instead of *destinations*.
- Need to perform cost aggregation:

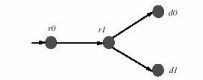


Figure 3. Topology to illustrate the label and cost aggregation.

$$cost(r_1, d_0, d_1) = \frac{(cost(r_1, d_0) \cdot R(r_1, d_0))}{(R(r_1, d_0) + R(r_1, d_1))} + \frac{(cost(r_1, d_1) \cdot R(r_1, d}{(R(r_1, d_0) + R(r_1, d}$$

Problem with this approach: need to maintain per flow state and perform packet classification at a core router (r1 in the previous example)

Simulation Results

- Load balancing and dynamic routing may hurt when the load is already balanced:
  - Overhead due to the computation of the cost function and the probabilistic choice may actually pick a longer route
- Otherwise we achieve a better link utilization
  - Better throughput
  - More fairness

Conclusion on LIRA

- Abandon the idea of absolute bandwidth
- Simple mechanism (token−bucket like) manages to achieve better link utilization
- First step towards Proportional Differentiated Services

Proportional Differentiated Services

- Ongoing research: the paper has been written in January 1999!
- Based on the assumption that "*some form of route pinning may be necessary for implementing such services* [Premium, Assured]".
- Does not offer absolute bandwidth guarantees, only relative guarantees
  - Two levels of differentiation:
    - Queuing delays (we will focus on this issue)
    - Packets losses

Requirements

- Relative differentiation between classes should be:
  - Predictable: Differentiation independent of the variation and distribution of the class loads.
  - Controllable: Should be adjusted by tuning any selected criteria
- In terms of queuing delays, we want to have something of the form:

$$\frac{(\bar{d}_i(I))}{(\bar{d}_j(I))} = \frac{\delta_i}{\delta_j} \quad (1)$$

The Dynamics

- Let λ be the aggregate arrival rate in the system, and λ_i the individual arrivals of each flow.
- We have: $\sum_{i=1}^{N} \lambda_i \bar{d}_i = \lambda \bar{d}(\lambda)$ (2)

Where d() is the average queuing delay that would result is the aggregate traffic was services by a work−conserving FCFS server of the same capacity as the scheduler that enforces the proportional delay model.

(1) and (2) imply: $\bar{d}_i = \frac{\delta_i \bar{d}(\lambda)}{\delta_1 \cdot \frac{\lambda_1}{\lambda} + \delta_2 \cdot \frac{\lambda_2}{\lambda} + \ldots + \delta_N \cdot \frac{\lambda_N}{\lambda}}$

The Dynamics (Cont'd)

- (3) in turns leads to the following properties:
  - The average delay of a class is increasing with the arrival rate of *every* class
  - Increasing load in higher classes cause larger increases in the class average delays than increasing load in lower classes
  - If the delay differentiation parameter of a certain class increases, the average delay of all other classes decreases while the average delay of that class increases

Feasibility of a Set of Deltas

- Is there a work−conserving sched that satisfies (1)?
- We need to have an "feasible" set of average delays $\sum_{i\in\Phi}\lambda_i\cdot\bar{d}_i \geq (\sum_{i\in\Phi}\lambda_i)\cdot\bar{d}(\sum_{i\in\Phi}\lambda_i)$

This set of inequalities should hold for all Φ in the set of nonempty proper subsets of {1,2,...,N}. The second term on the right is the average delay that the aggregate traffic of the classes in Φ would experience in a work−conserving FCFS server. To summarize: **The average backlog of a subset of classes cannot be lower than the backlog of these classes in a FCFS server.**

Existing Schedulers

- Static Priority: Does not work because one cannot adjust the delay between the classes. Besides the lower classes can experience service starvation
- Weighted−Fair Queuing: Proportional bandwidth does not imply proportional delay differentiation, unless the characteristic of the flow are known a priori
- Earliest Deadline First: could work but in an absolute mode only

The BPR Scheduler

- Backlog Proportional Rate
- The service rate allocation satisifies:

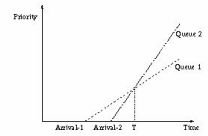
$$\frac{r_i(t)}{r_j(t)} = \frac{s_i}{s_j}\cdot\frac{q_i(t)}{q_j(t)}$$

Where {si} are the Scheduler Differentiation Parameters, directly linked to the {i} (they are the inverse of the latter, r is the rate and q is the queue length.

Claim: the relative waiting times tend to $\frac{s_i}{s_j}$
Drawback: Need to be approximated (fluid flow model)

The WTP Scheduler

- Waiting−Time Priority
- Proposed by Kleinrock in 1964
- Characterized by: $p_i(t) = w_i(t)s_i$
- Problem: need a dynamic reordering of the priority queues:



The WTP Scheduler (Cont'd)

- Second drawback:
- If R<Ri (peak input rate) and: $1-\frac{R}{R_i} > \frac{s_i}{s_j} (with s_i < s_j)$ then a sequence of *n* consecutive class *j* packets that starts arriving at time *t* will be services before any class *i* packets that arrived at *t* or later
- Can lead to starvation in the lower classes until a burst in a higher class is completely serviced!

Evaluation

- Both BPR and WTP satisfy the criteria when *the link utilization tends to 100%*.
- Actually the results seem valid for a link utilization *greater than 90%*.
- It seems that WTP is more appropriate (tends more quickly to the desired differentiation)
- Does it work also for short−lived flows? (short timescales)

Conclusion on PDS

- Interesting approach, more likely to be feasible and scalable than Absolute Differentiated Services.
- However, the studies are incomplete (high utilization of the link only)
- Current work in progress
- Probably need to use some of the IntServ capabilities (e.g., RsvP) to achieve all of these goals.