Cooperative Metering for Receiver initiated Service Level Agreement

Syed Umair Ahmed Shah Peter Steenkiste
Department of Computer Science
Carnegie Mellon University
{umair, prs}@cs.cmu.edu

Abstract-- **The Differentiated Service architecture is an attempt to provide a simple, flexible and scalable model of service differentiation based on the nature of the forwarding behavior required for the traffic. Service Level Agreements are established between the users and the service provider that specify the traffic contract to which both the sides agree to as well as the guarantees promised by the service provider. One such class of agreements can be that the traffic contract and guarantees are provided for the traffic going into a single or group of receivers from multiple sources. For many distributed applications, like a distributed simulation, where results are being gathered by a central server from several points in the network, it would be helpful if the server could control the share of each source or the aggregate traffic from multiple sources in a dynamic and fair manner. In this paper we present a simple strategy to provide a fair and conservative scheme for sharing the resources of the incoming link. The scheme parameters allow the provider to easily configure and adapt to the requirements of a range of traffic agreements.**

Index terms—**DiffServ, SLA, Dynamic Metering,**

I. Introduction And Problem Motivation

The current Internet service model is a simple best-effort network service model that does not provide guaranteed performance or guaranteed fairness. The performance of individual applications is highly dependent upon the demand for network resources by other applications and other users. The resource demands by these applications require that the performance guarantees should be provided in a fair scalable manner. The Differentiated Services (or DiffServ for short) framework is aimed at solving this problem. The basic concept in DiffServ is to have a simplified network core that treats a set or class of packets belonging to the same service class in the "same way". The traffic is policed at the entry point to the network according to the service agreement negotiated between the user and the service provider. Thus by not keeping per-flow state inside the network, the network core becomes simple and scalable.

The Service Level Agreement (SLA) defines the traffic contract between a service provider and a customer (user or another service provider) that specifies the forwarding behavior and guarantees the customer should receive from the network depending on the customer's needs and the provider's policies. The service providers can define a very rich set of SLAs that can be used by the customers.

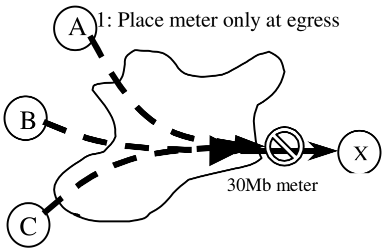
A single-source SLA is one that requests resources for incoming traffic from a single ingress node. It is the easiest to handle and enforce since one can simply place a meter at the ingress point and no further independent policing is necessary inside the network. For example an SLA for an ordinary dialup customer can be simply "Allow A to send X amount of guaranteed traffic to network Y." This prevents any excess traffic from entering the network as well as satisfying the contract. The more interesting case is of considering SLAs that span multiple ingress and egress nodes. For example an SLA contract "Allow A, B and C to send X amount of expedited forwarding traffic to D, E and F". Efficiently managing resources inside the network for all such SLAs becomes necessary as well as quite challenging.

In a multiple-source single-receiver SLA, the receiver might want to place constraints on how much traffic each ingress point can send to it. It can place multiple constraints on the amount of traffic of a certain class that different groups of ingress points can send to it. Consider a simple receiver-based SLA where the receiver says "Do not send more than 30Mb amount of traffic Y to me (X) from all the customers". This SLA is very useful and very likely to be one of the most common ones. A simple example can be a service provider who has several customers connected to its network via some ingress nodes. The service provider would like all the customers (including neighboring service providers) to share the egress links in a fair, controlled and efficient manner. This SLA can also form a basis of VPN service on a DiffServ network in which a group of sources cannot send more than a certain amount of traffic to a single receiver. Another application that can require such an SLA is a distributed simulation or a distributed computation or for a live auction and bidding website or a real-time sampling site.

A naive approach of handling this SLA would be to place a meter at the ingress and egress points with the total SLA limit on all meters (Fig 1). However this scheme has an obvious flaw of allowing too much traffic into the network which will get dropped at the egress node, after having used significant network resources. Such a liberal approach would also impact the guarantees promised to traffic belonging to other SLAs inside the core of the network.

Authors address: Computer Science Dept. CMU,
5000 Forbes Avenue, Pittsburgh, PA 15213

An opposing conservative solution can be to place meters at all the ingress nodes with a static equal share of the SLA resources, bandwidth in this case (Fig 2). This changes the semantics of the SLA to that of n identical and independent one-to-one SLAs with an equal share of the bandwidth. Not only that if an ingress point is not sending any data, the other ingress points are unable to use the available bandwidth. It also disallows the senders to send at several valid configurations like 20Mb, 5Mb, 5Mb.



1: Place meter only at egress

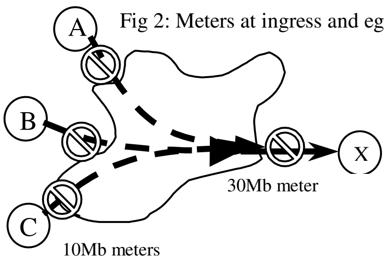


Fig 2: Meters at ingress and egress

Since multiple ingress points to the network can be sending to the same egress domain, dynamically assigning shares among ingress points and limiting extraneous traffic from entering the network is as important as satisfying service guarantees for such an SLA. The ingress nodes should cooperate with each other to decide how much each ingress node can send based on the SLA policies. The decision should be scalable, flexible, resource aware and conservative. In this paper we present a simple scheme that provides a conservative and dynamic share of the network resources among the different ingress nodes in the network for such an SLA. The scheme requires the meters on the ingress nodes to provide traffic statistics to a *Meter Coordinator* that periodically assigns new shares to each of the ingress nodes. It is also highly scalable and easily configurable to suit a wide range of network load conditions. We also provide some design tradeoffs of the scheme and the analysis of the parameters in our scheme. Section 2 describes the design of the meter coordination scheme. Details of the simulation and simulation results are presented in section 3. The status and plans for implementation in section 4 and conclusions in 5.

II. Design of the Meter Coordinator

In general most of traffic entering any ingress node of a network would show characteristics of burstiness. We need to provide a dynamically adapting metering policy that could provide the ingress meters with the necessary metering information based on the trend of the traffic in the network. The scheme that we propose requires that the ingress nodes (where the metering for the SLA is being done) collect statistics for the incoming traffic and forward them at regular intervals to an entity which we call the *Meter Coordinator*. Based on the gathered statistics the *Meter Coordinator Engine* would calculate and distribute new metering information to the ingress nodes. The ingress nodes will meter the traffic with this new profile for the next interval.

A. *SLA Policies*

In a single-receiver SLA, the receiver can place constraints on how the traffic should be shared among the ingress points. These constraints can be used to assign static as well as dynamic shares. Static shares assign one time non-changing shares to the ingress nodes which often waste the resources of the network. However it would be nicer if these shares are dynamically assigned based on the current network load conditions as dictated by the SLA policies. This is achieved by allowing the SLA to place multiple constraints on the amount of traffic that different groups of ingress points can send to a particular egress under different network load conditions. So we provide the customer with a very simple and flexible scheme for specifying a number of constraints in the general form

```
If condition then constraints
```

This form can handle both static as well as dynamic resource sharing. For example if we want a static constraint like a policy that ingress point A should always be able to send at least L_{AX} to egress X under a certain SLA, then we can specify this constraint as (S_{AX} represents the share)

$$\texttt{If true then } S_{AX} >= L_{AX}$$

On the other end of the spectrum, one is able to specify much more complex constraints for certain conditions that may or may not hold. For example a policy requiring that when the aggregate incoming traffic to the network (T) is greater than the limit agreed upon in the SLA (B_X), then each ingress should be assigned at least its weighted share of the receiver bandwidth. At the same time the combined share of traffic from ingress points A and B should be limited by L_{ABX}. This can be represented as

$$\texttt{If } \Sigma T_{iX} >= B_X \texttt{ then}$$
$$\texttt{For } \forall i,\ S_{iX} >= W_{iX} * B_X \texttt{ and } S_{AX} + S_{BX} <= L_{ABX}$$

Note that the customer and the service provider agree upon the weights in the SLA and even the weights can change for different load conditions.

This simple way of representing share assigning policies gives the customer much more flexibility and freedom to customize the traffic policing constraints according to his own requirements. At the same time these enable the service provider a much richer framework to provide service differentiation for satisfying a large set of customer requirements. The service provider can have a different pricing structure for each policy and form customized packages by combining different policies to suit the requirements of different customers or even define certain high demand packages.

B. Design

We propose that for each ingress node we have a single meter coordinator that can have a database of traffic statistics for all the relevant SLAs at that node. The meters at that node will update this database at regular intervals. The meter coordinator would then periodically take the appropriate data from the database and communicate it to the relevant meter coordinators at the egress nodes. So updates belonging to different SLAs could be packed and transmitted together on the same connection, reducing the communication overhead. The egress node coordinators, on the arrival of the messages, calculate the new shares for the ingress nodes and distribute them back. Note that the egress node can be an egress node for multiple SLAs and so the new share of all these SLAs can be sent back packed together. The meter coordinator at the ingress upon receiving the message updates the metering limits of all the relevant SLAs in the SLA database from where the meters can read their respective shares.

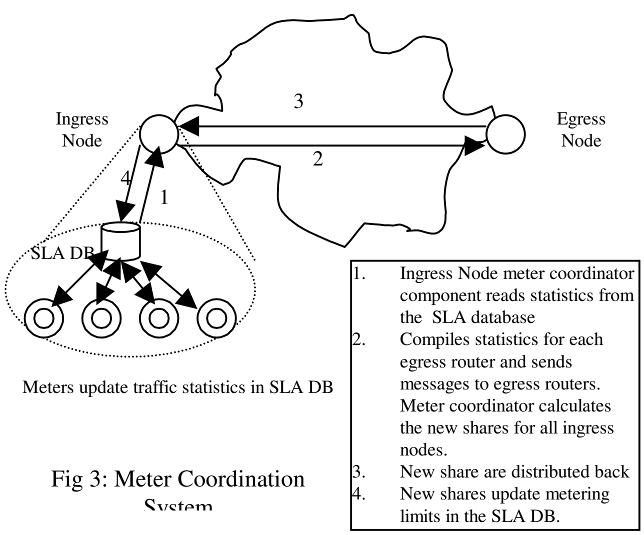


Fig 3: Meter Coordination System

The sensitivity of the incoming traffic will determine how often the metering information needs to be updated. For most traffic profiles the periodic time interval for updating the metering information would be a very small time interval i.e. in orders of 1 sec or less. This interval size should be significantly greater than the maximum delay from an ingress node to an egress node in the network.

For each SLA, the SLA database at the ingress node will store the amount of traffic reaching the ingress node from outside the network. Currently we only transmit the mean incoming traffic rate to the egress but for better estimation we can also provide variance to the egress. The egress node's SLA database would store the algorithm parameters for each ingress node. To remain consistent with the idea of having a simple core and maintaining all state at the edge, we do not include any information about the internal links of the service provider's network in our scheme. Such information could provide more flexibility and perhaps much better dynamic allocation but maintaining and gathering such information would unnecessarily complicate the core of the network. We prefer a simple and flexible scheme to a complex and flexible scheme.

C. Performance Indices

The performance objectives for our meter coordination strategy, were that it should:

- minimize the wasted bandwidth (bandwidth needed by some ingress node but not used)
- minimize excess traffic into the network and maximize throughput for all the ingress nodes while trying to stay as close as possible to the SLA limit.
- adhering closely to the resource sharing policy
- be able to handle any valid traffic load in a quick and stable way.

Another desired feature is flexibility to a range of network load conditions which we achieve by having appropriate parameters in our algorithm that could be used to fine tune the performance of our system. Scalability of the algorithm with the number of SLAs and network size is achieved by requiring only one statistics record for each SLA per ingress node. The egress node will also maintain one record per ingress node for every SLA.

1) Minimize bandwidth wastage

We use *percentage-wasted bandwidth* as our performance index for thruput maximization. For the time intervals when the total incoming traffic does not exceed the SLA bandwidth, the metering limit for each ingress meter is compared with the traffic that it received in that interval. For those ingress nodes for which the traffic was more than their allowed limits, the sum (X) of the excess traffic was calculated. X represents how much more traffic could have entered the network if they had been assigned different limits. We also calculate the unused bandwidth Y available for this SLA in that time interval. The *percentage-wasted bandwidth* is calculated to be the minimum of X and Y as a percentage of the total SLA bandwidth reservation to the egress. This index indicates by how much the QOS of the SLA is not being satisfied by the dynamic allocation of the network resources. If we could allocate a better share to the ingress nodes, we could have satisfied the QOS much more closely. Notice that when the total incoming traffic exceeds the SLA limit, then the traffic dropped due to policing is not considered as negatively affecting the QOS. It is only when the dynamic shares assigned to the ingress nodes are the cause of the bandwidth wastage, then our scheme is affecting QOS of the SLA.

2) Excess Traffic

To satisfy the QOS of the customer, the network needs to minimize the wasted bandwidth. One way to decrease the wasted bandwidth is by allowing more traffic into the network at the ingress nodes than that allowed at the egress node by the SLA. However we would like to minimize injecting excess traffic into the network since it would affect the performance of other SLAs as well as waste core bandwidth since it is going to be dropped at the egress router anyway. We use *percentage excess traffic* (total excess traffic entering the network as a percentage of the SLA limit) as our performance index for the excess traffic. Minimizing both the *excess traffic* and the *wasted bandwidth* is a tradeoff for the service provider to choose the right mix of these two performance measures to maximize the satisfaction of all the customers and to improve the performance of the scheme. Most likely the service provider would place more importance to the client satisfaction (fulfilling QOS). We provide flexibility in this respect also so that the customer and service provider can agree on their desired level of satisfaction at different prices.

3) Policy Adherence

For the purpose of this SLA, different policies will be specified in the contract presenting how the customer wants to distribute the shares among different ingress nodes under different network load conditions. We want to adhere to these policies as closely as possible. This is taken care by our algorithm that all policies are satisfied.

We defined a set of policies that we considered necessary and tested our scheme for those policies. We believe that all SLAs of this type need to have a minimum metering limit for an idle ingress so that it can start sending traffic. An *idle ingress* would be an ingress that is sending less than its minimum reserved share. Idle resources of the SLA should be shared equally only among the active ingress nodes. When the total incoming traffic for the SLA exceeds the SLA limit then only those ingress nodes are penalized that are exceeding their *fair share*. The ingress nodes that are sending below their *fair share* should be able to increase their resource usage so that they can achieve their *fair share*. The *fair share* of the ingress nodes can be based on the long-term traffic usage of the ingress node or maybe just equally dividing the resources among the ingress nodes.

D. Algorithm design and parameters

The outline of the general share-assigning algorithm for such an SLA with any traffic constraints is given below

1. gather all the statistics and determine which of the conditions hold true
2. collect all the constraints whose conditions are true
3. assign minimal shares to all the ingress points satisfying all the constraints
4. assign any remaining unused bandwidth among the ingress nodes according to their respective fair shares
5. distribute the new shares to the ingress nodes.

For the policies that we consider necessary for this SLA as described above, the algorithm would first assign the idle nodes the minimum share. From the remaining nodes those who are sending less than their fair share would be assigned their respective fair share and the remaining nodes will be assigned their weighted fair share of the remaining bandwidth. In the remainder of the document and simulations we will consider the SLA with these specific policies only.

The *fair share* is computed by taking the respective weights of each ingress node. The algorithm takes into account the excess bandwidth the ingress nodes are allowed to send by using a statistical gain parameter. It allows the meter coordinator to assign more bandwidth to the meters than the SLA limit so that the QOS is satisfied for the most time i.e. this is aimed to decrease the *percentage-wasted bandwidth* but it also injects some excess traffic into the network. For all the idle or low bandwidth consuming ingress nodes the minimum reserved bandwidth controls how easily they can start sending traffic. If traffic exceeding the allowed limit of the SLA is entering the network, then ingress nodes which do not exceed their fair share are allowed to increase their limit at the maximum rate of γ the *rate gain parameter*. This allows these ingress nodes to be able to reach their fair share. The *dampening parameter* α is used to account for the burstiness of the aggregate traffic. It reduces oscillations by preventing a sudden burst from causing major shifts in the shares of different ingress points. This allows a more gradual change in the shares and reacts to more consistent bandwidth changes protecting the network from self-induced oscillations.

$$S_i = \alpha S + (1-\alpha) S_i$$

Instead of the traffic arrival in the current time interval we use the weighted average traffic S_i as representing the traffic from an ingress node. S represents the actual traffic received in this interval at ingress node i.

III. SIMULATION

We implemented our algorithm in the ns-2 network simulator. Several features of DiffServ were added to ns including traffic conditioning, SLA admissions control, support for different PHBs and SLA types. We developed a Meter coordinator that received the messages from all the relevant meters and calculated and distributed the share of each ingress meter in the manner specified in the previous section. We also added support for generating traffic using a trace file. The performance of algorithms of this type depends heavily on the traffic characteristics. It was essential to use real traffic characteristics for driving our simulation to get meaningful results. We collected traffic traces from National Laboratory for Applied Network Research (NLANR) for evaluating our scheme. We had to convert the traces to approximate the distribution of traffic across multiple ingress domains. We used both ns-2 traffic sources and the trace traffic to drive our simulation. We ran several simulations for a wide range of parameters and configurations. In this section we will give brief details of the trace data conversion and our simulation results.

A. Trace Data

The performance of algorithms of this type depends heavily on the traffic characteristics. It was essential to use real traffic characteristics and we achieved this by collecting traffic traces for driving our simulation. We chose the National Laboratory for Applied Network Research (NLANR) because it has a very large network and also a large measurement infrastructure already in place[1]. The NLANR site gathers and maintains raw packet header data from different points in their network on a daily basis. They gather data in several different formats. We decided to use the 24-byte FR format which included the timestamp (seconds and microseconds), source and destination IP and port, packet size, IP protocol and TCP flags. All the fields except the IP addresses had the original values from the packet header. For security purposes the IP address had to be made completely irreversible using a one-way hash function. This meant that multiple trace data sets could not be used simultaneously to generate a single profile. Before encryption of the IP address we had mapped it to its corresponding BGP address prefix.

We refer to the header data set that was gathered at the SDSC (University of California San Diego) measurement points of the NLANR infrastructure. The 778MB trace data was gathered for about 3hrs and 40 minutes at the location on 7th June 1999. In all it had 1247 different source and destination domains whereas there were a total of 63625 address prefixes. Among the 1247 active domains, only 30 or so domains contributed more than 90% of the traffic. We looked at the traffic destined to some of highly active destination domains and we saw that fewer than 10 source domains were almost always responsible for over 90% of the traffic destined to each domain.
Generating the traffic profile from the trace data required the mapping of the domains into a common ingress node. We relied on an approximate solution by assuming a value for the number of ingress nodes to the network and a value for the bandwidth available at each ingress link. Using the rules that

- if traffic exists between two domains, then those domains must be having different ingress nodes to the network
- if traffic does not exist between two domains, then these domains are very likely to belong to the same ingress
- the total traffic coming from an ingress node should not exceed the assumed bandwidth for any interval of time
- the domains were divided among the assumed 8 ingress nodes

Thus we had assigned different domains to different ingress nodes assuming a star topology. The trace file was converted into a file which could be used to feed the sources in ns-2 where the sources were bound to a particular ingress node according to the above scheme.

B. Simulation Results

We compare the performance of our share-assigning scheme with the simple static scheme of allowing all ingress nodes to send at the SLA rate for different scenarios. This would allow all valid configurations of traffic mix but at the same time would introduce a lot more traffic into the network than that allowed by the SLA.

In the first scenario we developed a simple star core topology as shown in figure 4a. We use three source domains S1, S2 and S3 and one destination domain. In each domain there are 30 ftp traffic sources, each one attached to a separate node. The delays inside the domains are 10usec while the delays and bandwidths in the core are shown in the figure. Thus the round-trip delays for the domains will depend on the delays in the core network. All the sources

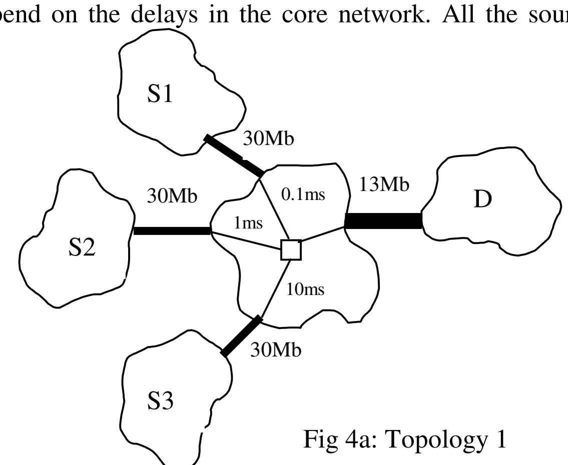


Fig 4a: Topology 1

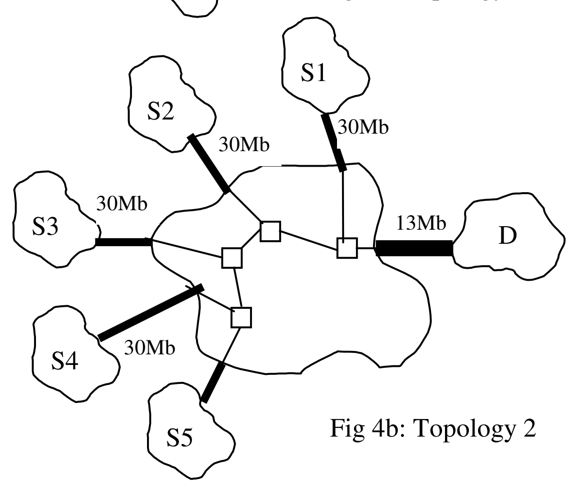


Fig 4b: Topology 2

[1] See http://moat.nlanr.net

send traffic to nodes in the destination domain D. A simple many-to-one SLA is setup with the policies as described earlier in the paper. The receiver bandwidth is set to 10Mbs traffic to enter on any of the ingress points for D. Meters are placed at all the ingress and egress points. All the meters are initialized to a limit of 10Mb. Traffic from S1 starts at time 10, from S2 at 0 and from S3 at 30. The total simulation duration was 80 sec. No background traffic was used. We compared the performance of our scheme with coordinated metering to the one with absence of coordinated metering. We use a metering limit of 10Mb as metering limit at each ingress node for the entire duration of the simulation.

Figure 5 shows how the throughput used by each of the three domains varied with time in the absence of coordinated metering. We see that the *fair share* policy is being violated for this SLA as whenever there is contention between the ingress nodes, the low end-to-end delay domain takes more egress node bandwidth as compared to the high end-to-end delay domains.

Next we repeated the same scenario but this time with the presence of coordinated metering. The parameter values used for the algorithm were: γ was 2, the time interval was set to 400msec, α was set to 0.5, SG was 1.05 and K to 0.001. This time (fig 6) the *fair share* policy among the ingress nodes is adhered to more closely and during periods 10 to 70 sec, the difference between the thruput shares of different source domains is much similar. The assigned metering for different ingress nodes exactly matched the policy requirement. The difference that exists is because of an unequal capture of the extra bandwidth assigned by the statistical gain parameter. For an SG value of 1.0 this unfairness will be minimized while wasting some bandwidth (i.e. not satisfying QOS). We repeated similar simulations on topology 2 that provides for different levels of traffic aggregation from different ingress nodes. We saw a similar performance pattern with the simulation.

The comparison of the variation of mean *percentage excess traffic* and mean *percentage wasted bandwidth* for the two schemes shows that our scheme introduces much lesser excess traffic with slightly more wasted bandwidth than the other scheme depending on the statistical gain parameter (Figure 7 & 8 for topology 1). The data labeled *wo* in the graphs represents the performance without coordinated metering. These graphs show the tradeoff between these two performance indices and a statistical gain of 1.05 to 1.1 seems to give the best of both. The wasted bandwidth remains around 1%, which has very good implications for the QOS. It also shows the inaccuracy of our prediction of future bandwidth usage. If we could more accurately predict the future bandwidth needs, then the wastage would have been less and the SLA QOS would have been met.

Next we took a star topology with 9 domains and used the trace data to drive the simulation. We had already divided the trace data domains across different ingress points. A

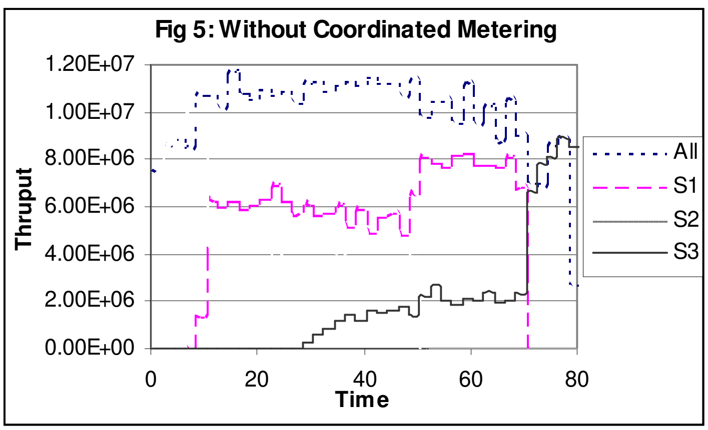


Figure 5: The traffic being sent to each ingress node from the respective source domains without having any support for cooperative metering.

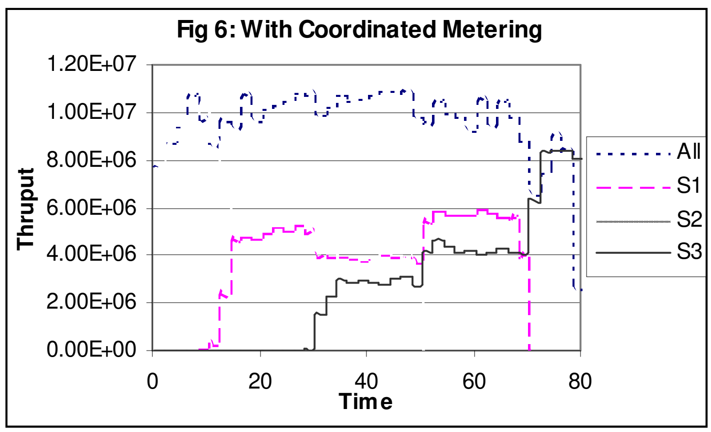
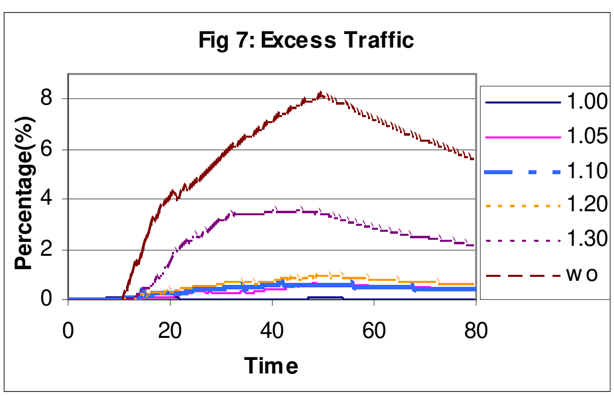
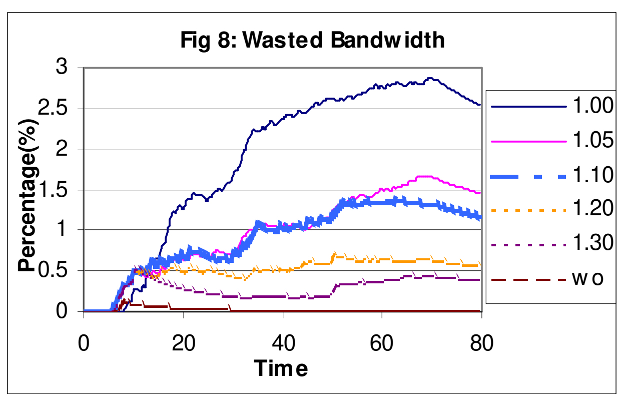


Figure 6: The traffic being sent to each ingress node from the respective source domains but this time with cooperative metering.





single traffic source for each domain was used to generate the aggregate traffic by reading their respective trace file lines and for each line injecting appropriate packets into the network. The core bandwidth used was 10Mb and the SLA bandwidth was assigned to be 3Mb. We tested both the schemes and observed similar behavior. The packet trace data was much more bursty and rapidly fluctuating as compared to the simulation sources. Thus it gave us a much more accurate picture of the performance of the scheme. The burstiness added more error to the traffic predictions, wasting more bandwidth and lowering the QOS performance of our scheme. Making the weight parameter lower improved the performance.

IV. IMPLEMENTATION STATUS

We have implemented DiffServ on Darwin[6], our local CMU Active Networks project. Darwin provides an infrastructure for the network to actively respond to the needs of the applications based on the current network conditions. Darwin did not provide traffic conditioning facility. We added this support to the Darwin kernel at the input port. We provided support for dynamic metering by making appropriate daemons which exchange the metering information and update the SLA DB which is stored inside the kernel. The basic router architecture that we used is shown in the figure below. Our brief implementation experience shows the proof of our concept and we will explore fully how we can learn various lessons from the implementation.

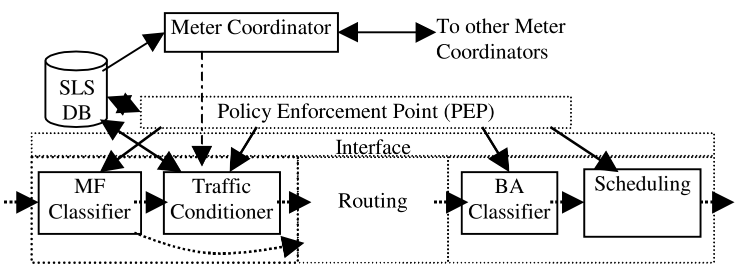


Fig 9: Router Architecture

V. CONCLUSION AND PLANS

We have shown how a simple strategy can be used to provide bandwidth guarantees in a receiver based many-to-one SLA in a conservative and flexible manner. Coordinated dynamic metering is necessary for the network provider for providing better service to the end users as well as efficient use of network resources. Our scheme enables the network provider to control the performance of the SLA along many different dimensions thus presenting a wide range of services for a client to choose from. We believe that this class of SLAs will form the basis of widely used traffic contracts by a DiffServ service provider.

The next step is to see how this basic scheme for a many-to-one SLA can be carried forward to a many-to-many SLA where the egress routers also share the resources based on similar policies. Another aspect is to see how once can play around with the different parameters in this simple scheme to give best performance for different SLAs.

VI. ACKNOWLEDGEMENTS

Special thanks to Hans-Werner Braun <hwb@nlanr.net> for gathering the packet header trace data in the NLANR network.

VII. REFERENCE:

[1] D. Clark and J. Wroclawski, "*An Approach to Service Allocation in the Internet*", Internet Draft <draft-clark-diff-svc-alloc-00.txt>, July 1997.

[2] K. Nichols, V. Jacobson, and L. Zhang*, "A Two-bit Differentiated Services Architecture for the Internet*", Internet Draft <draft-nichols-diff-svc-arch- 00.txt>, November 1997.

[3] Borje Ohlman*, "Receiver control in Differentiated services*", Internet Draft <draft-ohlman-receiver-ctrl-diff-01.txt>, 30 September 1998 .

[4] Marty Borden and Christopher White, "*Management of PHBs*", Internet Draft <draft-ietf-DiffServ-phb-mgmt-00.txt> August, 1998.

[5] Mandis Biegi, Raymond Jennings, Srinivasa Rao and Dinesh Verma, "*Supporting Service Level Agreements using Differentiated Services*", Internet Draft, <draft-verma-DiffServ-ntimplem-00.txt>, 18 Nov. 1998

[6] Prashant Chandra, Allan Fisher, Corey Kosak, T. S. Eugene Ng, Peter Steenkiste, Eduardo Takahashi, Hui Zhang, "*Darwin: Resource Management for Value-Added Customizable Network Service"*, extended draft of the ICNP paper

[7] Hungkei Chow, Alberto Leon-Garcia, "*A Feedback Control Extension to Differentiated Services*", Internet Draft <draft-chow-DiffServ-fbctrl-00.txt, March 1999

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

[9] Yoram Bernet, James Binder, Mark Carlson, Steven Blake, Mark Carlson, Srinivasan Keshav, Elwyn Davies, Borje Ohlman, Dinesh Verma, Zheng Wang and Walter Weiss*, "A Framework for Differentiated Services*" Internet Draft <draft-ietf-DiffServ-framework-01.txt>, October, 1998.

[10] Y. Bernet, D. Durham and F. Reichmeyer, "*Requirements of Diff-serv Boundary Routers*", Internet Draft draft-bernet-diffedge-01.txt, Nov 1998

[11] Z. Wang, "*User-Share Differentiation (USD), Scalable bandwidth allocation for differentiated services*", Internet Draft <draft-wang-diff-serv-usd-00.txt>, May 1998.

[12] Walter Weiss, *"Providing Differentiated Services through Cooperative Dropping and Delay Indication*", Internet Draft <draft-weiss-cooperative-drop-00.txt>, March 1998