Avoiding congestion through dynamic load control

Vasil Hnatyshin, Adarshpal S. Sethi

Department of Computer and Information Sciences,
University of Delaware, Newark, DE 19716

ABSTRACT

The current best effort approach to quality of service in the Internet can no longer satisfy a diverse variety of customer service requirements, and that is why there is a need for alternative strategies. In order to solve this problem a number of service differentiation models have been proposed. Unfortunately, these schemes often fail to provide proper service differentiation during periods of congestion. To deal with the issue of congestion, we introduce a new load control mechanism that eliminates congestion based on the feedback from the network core by dynamically adjusting traffic load at the network boundary. We introduce four methods for calculating load distribution among the ingress routers and among different flows in each ingress router, and we evaluate these proposed methods through simulation.

Keywords: Quality of service, dynamic admission control, load distribution, network feedback

1. INTRODUCTION

As a diverse variety of applications with different customer service requirements are beginning to access the Internet, the current best effort approach to quality of service in the Internet is no longer sufficient. As people become willing to pay more for services that satisfy their application requirements, the one-service-for-all approach of today's Internet will become obsolete, creating a need for alternative strategies. In order to solve this problem, a number of service differentiation models have been proposed. Differentiated Service architecture [1] introduced by IETF's DiffServ working group, core-stateless fair queuing [13] proposed by Stoica et al, and proportional service differentiation framework [4-6] proposed by Dovrolis et al are currently among the most popular approaches. Unfortunately, these schemes often fail to provide proper service differentiation during periods of congestion. For example, IETF's DiffServ model fails to provide fair resource allocation and fair service degradation during the periods of congestion [7,10-12] because of static resource allocation. On the other hand, the proportional service differentiation model does not violate relative guarantees under any network condition. However in this model, the lack of mechanisms for limiting the amount of data injected into the network can reduce the absolute level of QoS below user expectations [4-6].

We believe that one way to deal with this problem is to introduce a dynamic load control mechanism at the boundary routers. The idea behind dynamic load distribution is different from the idea of admission control. Although both mechanisms adjust themselves based on network feedback, they determine their adjustments differently. While admission control decides whether or not to admit new users into the network, the dynamic load control mechanism admits all the users and only when congestion arises does it adjust user sending rates. Figure 1 illustrates the idea behind the dynamic load control. As the figure shows, the traffic enters the network domain at the boundary router[1] B1, traverses the network in some fashion, and then exits this network domain at the boundary router B2. If congestion arises, the core routers provide feedback to the ingress routers. Based on this feedback, the ingress routers adjust the amount of traffic admitted into the domain.

The idea of adjusting admission control based on network feedback is not new. Chow et al also used such an idea in their work [2]. They proposed to periodically probe the network and, based on the obtained results, adjust admission control at the ingress nodes. However, they only introduced a general framework for the feedback-based dynamic admission control and did not pursue further investigation of this idea. The framework for Explicit Congestion Notification [9] also relies on a similar feedback mechanism. In this approach, when the core routers experience congestion, they set Congestion Experienced (CE) bits in the IP headers of packets flowing through them in order to

[1] Throughout this paper we will also refer to the boundary nodes at which traffic enters a domain as ingress routers and we will call the boundary nodes where traffic leaves a domain as egress routers.

notify the sources to reduce their rates. The Explicit Congestion Notification model assumes that the sources will reduce their rates upon reception of the CE marked packets, which may not always be true.

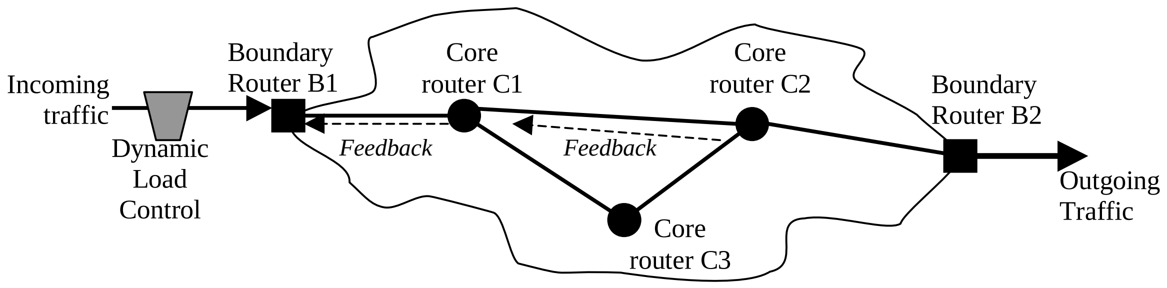

Figure 1. Scenario for Dynamic Load Control

In this paper, we extend these ideas and provide a new protocol for dynamic load distribution based on network feedback. The rest of the paper is organized as follows. Section 2 introduces the general idea of our model, including a detailed explanation of the message exchange mechanism used in this model. Section 3 describes possible techniques by which each ingress node can reduce its sending rate to eliminate congestion at the congested interface. Section 4 provides an initial experimental evaluation of our scheme. We include concluding remarks and present directions for future work in Section 5.

2. DYNAMIC LOAD DISTRIBUTION FEEDBACK CONTROL SCHEME

2.1. General Idea

The main idea of the feedback-based dynamic load control scheme is to dynamically adjust the traffic sending rates at the network boundaries. When the network is congestion-free, the boundary nodes allow users to inject as much data as they desire. However, when the network experiences congestion, the boundary nodes start limiting the amount of traffic that can be admitted into the network. During these periods of congestion, the core routers provide indications to the boundary nodes to slow down. We use an explicit message passing mechanism for providing congestion notifications to the boundary nodes. In our model, these congestion notification messages contain information about which core router interface is congested and its congestion level. This information allows each boundary node to determine by how much it should reduce the overall traffic rate to eliminate congestion. Once the overall reduced rate is calculated, the boundary router can calculate corresponding sending rates for all user flows that contribute to the congestion.

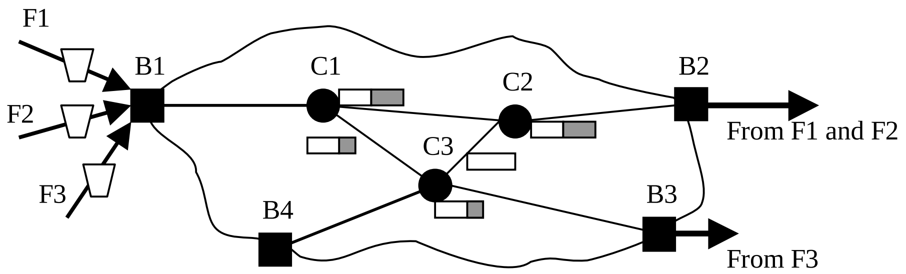

Figure 2. Congestion-free network

Consider the situation shown in Figure 2. This figure shows a congestion-free network, where the boundary node B1 accepts traffic flows F1, F2, and F3 and forwards them towards their corresponding destinations. In particular, flows F1 and F2 pass through core routers C1 and C2 and then exit this domain at the egress node B2. Flow F3 passes through the routers C1 and C3 and then exits the network domain through the boundary node B3. Figure 2 depicts a situation where all the interfaces of the core routers are congestion-free and no load control at the boundary nodes is required. Figure 3 shows what happens when one of the core link interfaces becomes congested. Suppose that flow F4 becomes active and enters the domain at boundary node B4. Flow F4 follows the path C3, C2, B2 and creates congestion on the link between C2 and B2. C2 identifies this situation and generates congestion notifications to all the boundary nodes that send their traffic through the congested link. Congestion notifications travel through the routers C1 and C3 to reach boundary nodes B1 and B4 respectively.

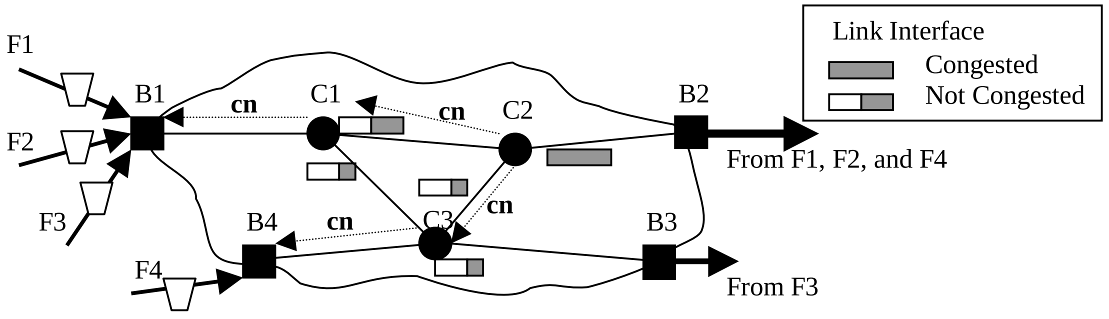

Figure 3. Node actions during the congestion

When B1 and B4 receive their congestion notifications, they calculate by how much they should reduce their traffic rate to eliminate congestion. Then B1 and B4 identify the flows that contribute to the congestion between C2 and B2 (F1, F2 and F4). After that, B1 and B4 distribute the calculated rate reduction among these flows, by controlling each flow's load accordingly. Figure 4 illustrates how the network situation changes after the activation of load control for the selected flows.

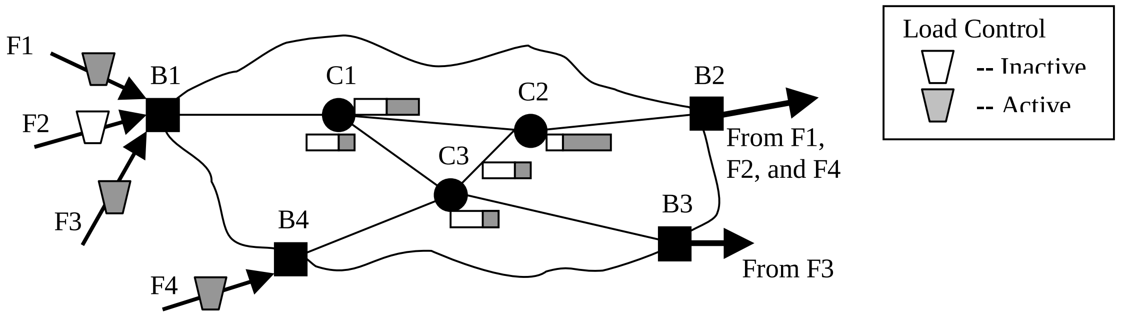

Figure 4. Congestion-free network after the adjusting load control

2.2 Overview of the message exchange mechanism

In our model, we assume that for each source host that sends traffic through a given ingress node, the ingress node knows the list of destination hosts for which the traffic is intended. We record this information in the Service Level Agreement (SLA) established between each source and the network domain. The SLA will also include information about the minimum provisioned rate for each source-destination host pair. During congestion, the boundary node will not reduce a flow's sending rate below this minimum provisioned rate specified in the SLA. In our scheme we assume that the network is well provisioned, and that congestion does not occur when all the users send traffic at their minimum provisioned rates. Throughout the paper, whenever we refer to a flow, we will mean packets flowing between a source-destination host pair specified in the SLA, as contrasted with the traditional definition of a flow, as stream of packets flowing between a pair of applications. Table 1 shows an example of Service Level Agreements established between users (source hosts) and a network domain.

When congestion occurs, the boundary node receives a congestion notification that specifies the address of the interface where congestion occurred and the level of congestion on that interface. Not all the flows that enter the domain at a particular ingress node travel through the congested interface; therefore, it is impossible to determine which flows should slow down based only on the information provided in the congestion notification message. To remedy this situation, we maintain two data structures in addition to the SLA in each boundary node, called ***path table*** and ***router table***. These tables allow the boundary node to identify the flows that should be slowed down when congestion occurs. These data structures are generated and maintained based on information collected by probe messages, which the ingress nodes periodically transmit on each currently active path, in order to discover the path changes. Probe messages are also generated when a flow becomes active for determining the flow's path within this domain. The probes collect the list of routers they traversed within the domain and the ingress nodes use this information to maintain the ***path*** and ***router tables***. The ***path table*** contains a complete route within the domain that the flows follow to their destination. The ***router table*** keeps the router interface addresses and the list of flows that pass through those interfaces. Thus, whenever a flow becomes active or the path timer expires, the ingress boundary node generates a probe message that follows the route toward a particular destination. The probe terminates its progress at the egress node, which is the last router within the domain on the path to the destination. The egress node forwards the probe back to the ingress node that generated it. We refer to the message forwarded by the egress node as a probe reply message.

Flow ID	**Source**	**Egress**	**Min. Provisioned Rate**
1	192.0.1.1	B2	200 Kbps
2	192.0.1.1	B3	250 Kbps
3	192.0.3.5	B2	190 Kbps
4	192.0.3.5	B3	300 Kbps

Table 1. Example of Service Level Agreement

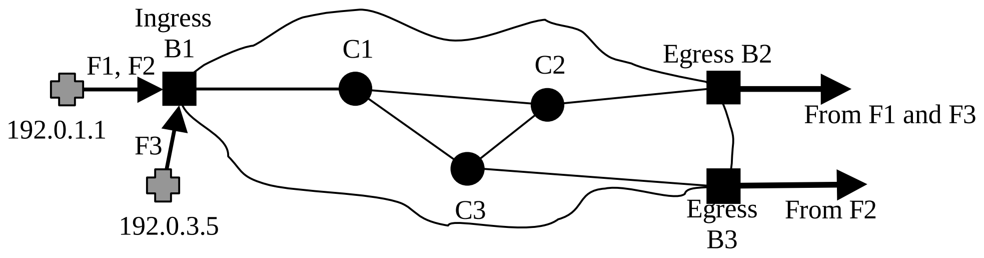

Figure 5. Network Topology

Destination	**Path**	**List of Flows**	**Timer**	**Min. Provisioned Rate**
B2	B1, C1, C2, B2	F1, F3	0.39	390 Kbps
B3	B1, C1, C3, B3	F2	1.17	550 Kbps

Table 2. ***Path Table*** for the boundary node B1.

Router Interface	**List of Flows**
B1-C1	F1,F2,F3
C1-C2	F1, F3
C1-C3	F2
C2-B2	F1, F3
C3-B3	F2

Table 3 ***Router Table*** for the boundary node B1.

Tables 2 and 3 show an example of the ***path*** and ***router tables*** respectively at node B1 for active flows within the network of Figure 5. We index the ***path table*** using the full path between the current ingress node and the egress router where the flow exits the domain. In addition, each entry in the ***path table*** contains a timer and a list of active flows that follow that path. The timer specified in the ***path table*** is used to determine when the next probe for that path should be generated. A ***router table*** kept in each ingress node is indexed by the router interface address and contains the list of the flows that pass through that interface. Both of these tables are built based on the data collected by the probe messages.

When a probe reply message arrives at the ingress node, it contains a list of the interface addresses and their current arrival rates. Interface arrival rates are used to update load control for the flows that pass through those interfaces. The interface addresses are used to update the ***path*** and ***router tables***. If the probe reply message was generated due to timer expiration and contains a new route, then the old route entry in the ***path table*** should be replaced with the new route information. Moreover, the ***router table*** should be updated. First, we identify the list of flows that follow a new path, let us call it *flows-to-update* list. Then, we remove the list of *flows-to-update* from each router interface entry of the ***router table*** that was part of the old route. Similarly, we add the list of *flows-to-update* to each router interface entry of the ***router table*** that is part of the new route. In the case when the probe message was generated due to activation of a flow, we update the ***path table*** as follows. If returned path already existed we simply add a new flow to the list of current flows for that entry, otherwise we create a new entry in the ***path table***. Similarly, we update the ***router table*** by either modifying the list of flows for all existing interfaces, or by adding new entries into the ***router table*** if unknown interfaces were encountered.

Now, let us consider an example of ***path*** and ***router tables*** updates. Consider that, in addition to the active flows shown in Figure 6, flow F4 becomes active and starts sending traffic. When the first packet of the flow F4 arrives at the boundary node B1, a probe is generated and forwarded towards its destination. When the egress node B3 receives the probe generated by the ingress node B1, it copies collected data into the probe reply message and sends it to B1. Upon arrival of the probe reply message at the boundary node B1, the corresponding entries in the path and router tables are updated. Figure 6 illustrates how the probe message exchange works and Tables 4 and 5 show how ***path*** and ***router tables*** are updated.

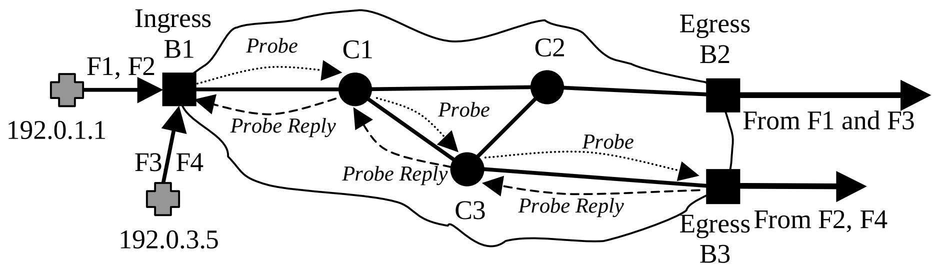

Figure 6. Network Topology

Destination	Path	List of Flows	Timer	Min. Provisioned Rate
B2	B1, C1, C2, B2	F1, F3	1.09	390 Kbps
B3	B1, C1, C3, B3	F2, F4	0.00	550 Kbps

Table 4. Updated ***path table*** for the boundary node B1.

In general, the message exchange mechanism of the feedback-based dynamic load control scheme consists of two parts. The first part called probe message exchange keeps track of the routing changes and allows boundary routers to identify the flows to which load control should be applied. The second part called congestion notification exchange, provides indication to the boundary nodes about when and how to adjust the load control.

The probe message passing accomplishes two goals. First of all it allows core routers to identify which ingress routers should be notified during the congestion, and it allows ingress routers to identify the flows that should reduce their rates. Since boundary nodes generate probe messages each time a flow becomes active, these messages allow a core router to keep track of the boundary nodes that are sending traffic through it. When the core router receives a probe message, it records the address of the boundary node that generated this message and the address of the outgoing interface on which the probe message departed. So for each outgoing interface the core router keeps a list of ingress routers that send traffic through this interface. During congestion, the core routers consult this table and send congestion notifications to all the boundary nodes that sent the traffic through the congested interface.

Router Interface	**List of Flows**
B1-C1	F1,F2,F3,F4
C1-C2	F1, F3
C1-C3	F2, F4
C2-B2	F1, F3
C3-B3	F2, F4

Table 5. Updated ***router table*** for the boundary node B1.

The core nodes generate congestion notification when an outgoing interface becomes congested. Core nodes estimate incoming traffic rate for each outgoing interface. If the estimated rate on a particular outgoing interface reaches a certain threshold then it signifies congestion and notification messages are sent to corresponding boundary nodes. A congestion notification message carries interface address, estimated arrival rate on that interface, and the capacity of that interface. Based on this information along with the local flow information, the boundary nodes can calculate the new sending rate for each of the flows that send its traffic on the congested interface.

3. RATE REDUCTION TECHNIQUES

3.1 Calculation of the congestion reduced aggregated rate

The problem of adjusting traffic load at the boundary nodes may be divided into two parts. The first part deals with the problem of calculating the total load that a particular ingress router can generate without causing congestion in the network core. The second part of the problem deals with the question of how the total load should be divided among the individual flows. In this section we will introduce three simple mechanisms for solving the first problem. Each ingress router must compute the new aggregate rate at which it transmits traffic through the congested link. This rate, which we call Congestion Reduced Aggregated Rate (CRAR), should satisfy two requirements:

1. The sum of all CRARs from all the ingress nodes should not exceed the congested link’s capacity.
2. The congested link’s utilization should remain high after all the ingress nodes reduce their sending rates.

It is very difficult for the ingress nodes to compute the CRARs that satisfy the above requirements without additional information about the rate distribution among the various ingress nodes that send traffic on the congested interface. Unfortunately, this information is not available to the ingress nodes, which only have the estimated value of their own sending rate and the information about the arrival rate and capacity of the congested interface.

A possible solution to this problem is to have the core router monitor the rates of traffic reaching it from each ingress node and then provide this information to the boundary nodes when congestion arises. In order for the core router to estimate the arrival rate of the boundary node, it should know which flows enter the domain through that ingress node. One way to achieve this is for the core router to pre-compute and keep information about all the flows that enter this domain and their corresponding ingress nodes. Another approach would be to require all the packets to be marked in such a way that the core router could uniquely identify the ingress node through which the packet entered this domain. Both of these approaches seem to be too “heavyweight” and would introduce too much additional complexity into the core router processing. Because of this additional complexity associated with distribution of the boundary node traffic sending rates among the ingress routers, we decided to estimate the value of the Congestion Reduced Aggregated Rate.

We will first examine a set of naive methods for calculating the CRAR at each boundary node. In the first method, we reduce the rate at each ingress node proportionally to its sending rate. We will use the following notation in our subsequent discussion:

- R_i – total traffic rate on the congested interface i from all ingress nodes.
- C_i – capacity of the congested interface i.
- E_i – excess traffic received at the congested interface i.
- R_i^{new} – CRAR that arrives at interface i.

- R_{ji} – total traffic rate from the ingress node j to the interface i.
- S_{ji} – total provisioned traffic rate from the ingress node j to the interface i.
- E_{ji} – excess traffic sent from the ingress node j to the congested interface i.
- R_{ji}^{new} – CRAR from ingress node j to interface i.

In the first method, called naive method 1, we calculate the CRAR R_{ji}^{new}, as follows, reducing sending rates only of those ingress nodes that send traffic above their minimum provisioned rate.

$$R_{ji}^{new} = R_{ji}\left(1-\frac{E_i}{R_i}\right), \text{only if } R_{ji} > S_{ji} \qquad (3.1.1)$$

Using this method, the boundary nodes will reduce their aggregated sending rates to R_{ji}^{new}. However such rate reduction may not completely eliminate congestion. Consider the situation where some ingress nodes send traffic below their minimum provisioned rate and therefore do not slow down during the congestion. For this case, we can show that reducing the sending rate in accordance with naive method 1 will not eliminate the congestion. We define congestion as the situation when the estimated arrival rate at the link is above the link's capacity, $R_i > C_i$. The excess traffic is given by:

$$E_i = R_i - C_i \qquad (3.1.2)$$

The total rate U_i from all nodes sending traffic below their provisioned rates is:

$$U_i = \sum_{j \ni R_{ji} \le S_{ji}} R_{ji} \qquad (3.1.3)$$

Whereas the total rate O_i from all nodes that exceed their provisioned rates is:

$$O_i = \sum_{j \ni R_{ji} > S_{ji}} R_{ji} \qquad (3.1.4)$$

Obviously,

$$R_i = \sum_i R_{ji} = O_i + U_i \qquad (3.1.5)$$

When load control is applied according to naive method 1, the new rate from the previously over-provisioned nodes is adjusted to:

$$O_i^{new} = \sum_{j \ni R_{ji} > S_{ji}} R_{ji}^{new}, \text{ where } R_{ji}^{new} < R_{ji} \qquad (3.1.6)$$

Thus, the CRAR is:

$$R_i^{new} = \sum_j R_{ji}^{new} = U_i + O_i^{new} \qquad (3.1.7)$$

By re-writing (3.1.7) and by using definitions (3.1.2) – (3.1.6) we will obtain that:

$$R_i^{new} = O_i\left(1-\frac{E_i}{R_i}\right) + U_i = R_i - E_i\frac{O_i}{R_i} > C_i \qquad (3.1.8)$$

Inequality (3.2.8) holds since E_i is the exact amount by which we should reduce rate R_i in order to eliminate congestion. However, since the ratio $\frac{O_i}{R_i}$ is smaller than 1 because of the assumption that there are some ingress nodes that send traffic below their minimum provisioned rate, we have reduced the rate R_i by an amount smaller than E_i. Therefore, in this case, we will not eliminate congestion at the core router i. However, if all the ingress nodes send above their minimum provisioned rates, such rate reduction will eliminate congestion. Another problem of this approach is that it computes rate reduction based only on the ingress node's sending rate and congestion level in the core router. Because of that, this rate reduction method might cause certain ingress nodes to reduce their sending rates below their corresponding minimum provisioned rate S_{ji}.

To eliminate the last problem, we propose two other rate-reduction techniques, which we will call naive methods 2 and 3. Similarly to the naive method 1, we will reduce sending rate only at those ingress nodes that send traffic above their minimum provisioned rate. Both of these approaches guarantee that the ingress nodes will not reduce their sending rates below their minimum provisioned rates. In naive method 2, the ingress nodes reduce their sending rate only by a fraction of the excess rate, and therefore can reduce their sending rate by excess rate at the most.

$$R_{ji}^{new} = R_{ji} - E_{ji}\frac{E_i}{R_i}\text{, only if } R_{ji} > S_{ji} \qquad (3.1.9)$$

On the other hand, in naive method 3, we explicitly prohibit reducing sending rates below corresponding minimum provisioned rates.

$$R_{ji}^{new} = \max\left(R_{ji}\left(1-\frac{E_i}{R_i}\right), S_{ji}\right)\text{, only if } R_{ji} > S_{ji} \qquad (3.1.10)$$

Unfortunately, none of the naive methods can reduce sending rates to a level that will eliminate congestion after only a single congestion notification. Using these rate reduction approaches would require multiple congestion notification messages before congestion in the core router's interface is eradicated.

3.2 Two-iteration method for computing CRAR

We expect that none of the naive techniques for computation of the CRAR at the ingress nodes can guarantee fast convergence to the sending rate that would eliminate congestion. That is why we propose an alternative method that is able to reach a sending rate that will eliminate congestion after the second congestion notification. In this scheme, when the first congestion notification arrives at the ingress nodes, the sending rate is lowered to a level which reduces but does not eliminate the congestion. Because of that, the congested interface of the core router generates another congestion notification. In our scheme, after receiving the second congestion notification, the ingress nodes are able to calculate the exact value of the CRAR, such that congestion in the core router's interface will be completely eliminated. In this section we will use the same notations as in Section 3.1, except that we will distinguish between the rates before the first and the second computation of the CRAR. For example, we will call the amount of excess traffic that interface *i* receives before the first and the second rate reductions as E_i^1 and E_i^2 respectively. Also we will use symbol d_i^1 to denote the rate reduction ratio used to compute the new rate after receiving the first congestion notification. Thus, when the first congestion notification is generated, the congested interface has an amount of excess traffic that is equal to E_i^1. After the rate is reduced using reduction ratio d_i^1, the congested interface will have an amount of excess traffic that is equal to E_i^2. When an ingress node receives a congestion notification, it computes the first rate reduction ratio as follows:

$$d_i^1 = \frac{E_i^1}{R_i^1} \qquad (3.2.1)$$

Those ingress nodes that send traffic below their provisioned rates have excess traffic rate equal to zero. Then each ingress router that receives a congestion notification will compute the CRAR as follows:

$$R_{ji}^2 = R_{ji}^1 - E_{ji}^1 d_i^1 \qquad (3.2.2)$$

The excess traffic after this rate reduction is:

$$E_{ji}^2 = E_{ji}^1\left(1-d_i^1\right) \qquad (3.2.3)$$

We can easily show that this reduction will not eliminate congestion and therefore a second congestion notification will be generated. The first rate reduction caused the following decrease in the arrival rate at the congested interface:

$$d_i^1 \sum_j E_{ji}^1 = E_i^1 - E_i^2 \tag{3.2.4}$$

Based on the knowledge that, for ingress node *i*, the reduction by $d_i^1 \times E_{ji}^1$ caused an overall rate reduction at the congested interface in the amount of $E_i^1 - E_i^2$, we can calculate the reduction ratio d_i^2, such that it will cause an overall rate reduction at the congested interface in the amount of E_i^2. Based on this observation and equation (3.2.3), we can construct the following equality and solve it for d_i^2. In order to be able to construct this equality, we assume that the number of the ingress nodes that adjust their rates remains the same for the period of the rate reduction.

$$d_i^1 \frac{E_{ji}^1}{E_i^1 - E_i^2} = d_i^2 \frac{\left(1 - d_i^1\right) E_{ji}^1}{E_i^2} \tag{3.2.5}$$

$$d_i^2 = \left(\frac{d_i^1}{1 - d_i^1}\right)\left(\frac{E_i^2}{E_i^1 - E_i^2}\right) \tag{3.2.6}$$

Selecting the value of d_i^2 according to equation (3.2.6) ensures that the previously congested interface will become congestion-free. To verify the correctness of equation (3.2.6), we will show that the overall rate reduction after the second congestion notification equals the amount of excess traffic at the congested interface.

$$d_i^2 \sum_j E_{ji}^2 = E_i^2 \tag{3.2.7}$$

By applying equation (3.2.3) to the left side of equation (3.2.7) and then by using equations (3.2.4) and (3.2.6) we obtain:

$$d_i^2 \sum_j E_{ji}^2 = d_i^2 \left(1 - d_i^1\right) \sum_j E_{ji}^1 = d_i^2 \left(1 - d_i^1\right) \frac{E_i^1 - E_i^2}{d_i^1}$$

$$d_i^2 \sum_j E_{ji}^2 = \left(\frac{d_i^1}{1 - d_i^1}\right)\left(\frac{E_i^2}{E_i^1 - E_i^2}\right)\left(1 - d_i^1\right)\left(\frac{E_i^1 - E_i^2}{d_i^1}\right) = E_i^2$$

Therefore, equation (3.2.7) holds, and selection of the reduction ratio for the second iteration according to equation (3.2.6) will eliminate congestion on the interface that generated the congestion notification.

3.3 Reduction of the flow rates during the congestion

When the boundary node receives a congestion notification, it calculates the CRAR, using one of the methods presented in Sections 3.1 and 3.2. After that, the boundary node calculates the new traffic rate, r_n^k, for each flow that contributes into the aggregate that passes through the congested link. In this section, we will use a slightly different notation in order to make the formulas more readable.

- R_{new} – Congestion Reduced Aggregated Rate.
- R – estimated traffic rate generated by the flows passing through the congested interface.
- S – sum of the minimum provisioned traffic rates for all the flows that travel to their destination through the congested interface.
- r^k – estimated traffic rate of flow k that passes through the congested interface.
- s^k – minimum provisioned traffic rate of flow *k* that passes through the congested interface.
- r_{new}^k – new traffic rate of flow *k* that passes through the congested interface.

The idea of this approach is very simple: we calculate a fair share for each flow and we reduce the flow's sending rate only if it sends traffic above its fair share. We calculate the fair share of the flow as follows:

$$f^i = s^i\left(1+\frac{R_{new}-S}{S}\right) \qquad (3.3.1)$$

where,

$\left(R_{new}-S\right)$ is the amount of excess bandwidth available for all the flows.

Then, we calculate a new sending rate for each flow that sends above its fair share as follows:

$$r^i_{new} = s^i\left(1+\frac{R_{new}-R^{under}-S^{over}}{S^{over}}\right) \qquad (3.3.2)$$

where,

$R^{under} = \sum_{i \ni r^i \le f^i} r^i$, is the total sending rate of those flows that send below their fair share.

$S^{over} = \sum_{i \ni r^i > f^i} s^i$, is the total provisioned rate of those flows that send above their fair share.

$\left(R_{new}-R^{under}-S^{over}\right)$ is the excess bandwidth left for sharing among the flows that send above their fair share.

We assert that reducing flow rates according to equation (3.3.2) will reduce the aggregated rate to the calculated value of R_{new}. To prove this assertion, we need to show that:

$$\sum_i r^i_{new} = R_{new} \qquad (3.3.3)$$

By applying the definition of R^u, S^o, and (3.3.2) to (3.3.3) we get:

$$\sum_i r_{new}{}^i = R^{under} + \sum_i s^i\left(1+\frac{R_{new}-R^{under}-S^{over}}{S^{over}}\right) = R^{under} + \sum_i s^i + \left(\frac{R_{new}-R^{under}-S^{over}}{S^{over}}\right)\sum_i s^i$$

$$\sum_i r_{new}{}^i = R^{under} + S^{over} + \left(\frac{R_{new}-R^{under}-S^{over}}{S^{over}}\right)S^{over} = R_{new}$$

4. EVALUATION OF THE LOAD DISTRIBUTION FEEDBACK CONTROL SCHEME

4.1 Simulation set-up

To evaluate the proposed load control scheme, we built a simulation model using the OPNET [7] network simulator. We implemented the message exchange mechanisms described in Section 2, and compared the performance of the different rate reduction schemes for the network topology shown in Figure 7.

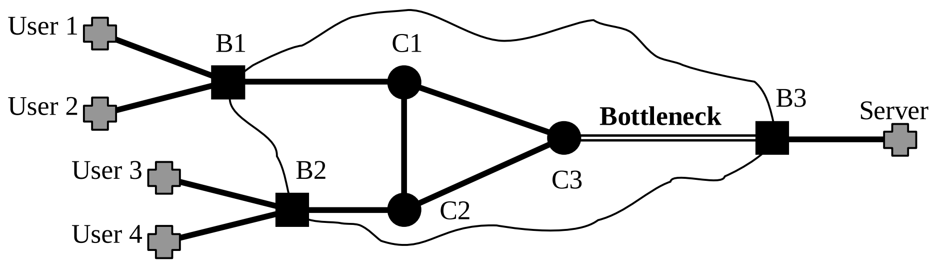

Figure 7. Simulation topology.

We connected all the nodes in this topology with T1 links with a capacity of 1.544 Mbps. Only the bottleneck link between nodes C3 and B3 was provisioned with a different amount of bandwidth, varying its capacity to simulate different congestion situations. We simulated one-directional traffic from the user nodes (sources) to the server node (destination). All the sources generated FTP traffic with an average inter-request time of 0.1 second and data size of 800 bytes. Including all the headers, this added up to a total sending rate of 66,240 bits/second for each user. Also the boundary nodes B1 and B2 established SLAs with their corresponding users as shown in Table 6.

We used the Time Sliding Window Metering (TSW) mechanism [3] for estimating the arrival rate in the core and boundary nodes. The boundary routers estimated a value of the arrival rate for calculation of new reduced rates, while the core router used the estimated rates for determining if there was congestion. We configured the TSW meter with a window size of 5 seconds. During the congestion, we passed user traffic through a simple token bucket that reduces the sending rate of the user as needed. We ran our simulations for 300 seconds and activated users according to the following schedule:

- User 1 and user 3 start sending traffic at time instant chosen randomly during the first 40 seconds of simulation.
- User 4 starts sending traffic at time 100 seconds.
- User 2 starts sending traffic at time 300 seconds.

	User 1 – Server	**User 2 – Server**	**User 3 – Server**	**User 4 – Server**
Minimum Provisioned Rate	10 Kbps	20 Kbps	30 Kbps	40 Kbps
Boundary node	B1	B1	B2	B2

Table 6. Service Level Agreement

4.2 Evaluation of the 2-step rate reduction method for computing CRAR.

During our experimental evaluation, we observed that the 2-step rate reduction method was not able to properly estimate the CRAR. After close examination of the results, we noticed that the value of the aggregated arrival rate at the congested interface as reported to the ingress nodes was not very accurate. The 2-step rate reduction method is based on the assumption that we can clearly observe the results of the first rate reduction. Any inaccuracy of the current arrival rate values causes the estimate of the new CRAR to fail during the second iteration. For this reason, we evaluate the performance of the 2-step rate reduction method separately from the other methods.

We further observed that the value of the arrival rate at the congested link reported during the first iteration was usually smaller than the value reported during the second iteration. Closer examination of that phenomenon revealed that the Time Sliding Window rate estimator was not reporting the current rate accurately enough and required additional time to converge to the actual value of the arrival rate. Therefore, the ingress node was observing that the rate reduction after the first congestion notification caused an increase instead of decrease in the arrival rate at the core link. Because of that we slightly modified our implementation of the 2-step rate reduction mechanism. Instead of generating congestion notification immediately after the congestion was observed, we delay this notification. Introduction of the delay significantly improved the performance of the 2-step rate reduction mechanism. This method was now able to eliminate congestion in exactly two steps.

A second problem with this method was that it reduced total sending rates of the ingress nodes below their minimum provisioned rates. Again, the reason for this behavior was the inaccuracy of the reported values by the TSW rate estimator. In one case, we noticed that although all the ingress nodes reduced their total load by 24 Kbps, the TSW rate estimator reported a rate reduction of only 18 Kbps. Such inaccuracy forced the ingress nodes to reduce their loads by an amount larger than was needed to eliminate the congestion. We then explicitly added a condition that does not allow the reduction of the total load at the ingress node to be below the minimum provisioned rate. This solved the problem partially but still resulted in larger than necessary rate reduction, causing the congested link to become underutilized. In our simulation, the 2-step method with these modifications was properly reducing the ingress node sending rates only when the interval between the congestion notifications was about 17 seconds.

4.3 Evaluation of the naive rate reduction methods for computing CRAR.

In our evaluation, we used the number of congestion notifications needed to completely eliminate congestion in the core router as a measure of how well the method performs. For each method, we ran the experiment 10 times, counted the number of notifications required to completely eliminate congestion, and averaged these counts over the 10 runs.

In this set of simulations, we provisioned the bottleneck link with 105 Kbps of bandwidth, which is only 5 Kbps above the arrival rate at the link in the case when all of the users send their traffic at the minimum provisioned rate. We also set the minimum time interval between two consecutive congestion notifications to be 1 second. This interval was introduced in order to avoid sending too many congestion notifications.

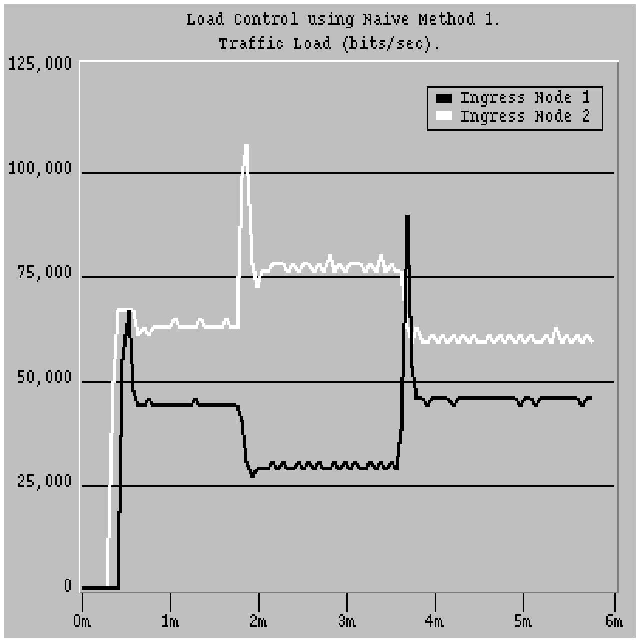

Figure 8. *Naive method 1*: ingress node sending rate variation

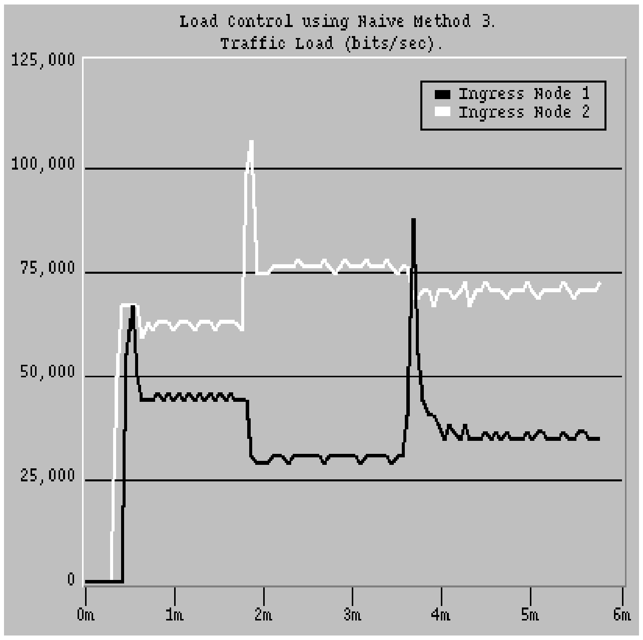

Figure 9. *Naive method 3*: ingress node sending rate variation

Due to the user activation schedule, our simulation had three distinct periods of congestion.

- The first period of congestion occurred at time 30 seconds, when both users 1 and 3 became active. Since all users send their traffic at the rate of 66.24 Kbps, the core router C2, had (2 * 66.24 – 105) = 27.48 Kbps of excess traffic arriving on its interface.
- The second period of congestion stated at time 105 seconds, when the user 4 started sending traffic. At this point, core router C2 had about 66.24 Kbps of traffic arriving above the capacity of the interface.
- The third congestion period started at time 210 seconds when user 2 began sending traffic. The core router C2 again was observing 66.24 Kbps of excess traffic arriving on its link.

Table 7 displays the number of congestion notifications required to eliminate congestion in the core for each period of congestion.

As expected, naive method 2 performs very poorly. This method requires about two and a half times as many congestion notifications as naive methods 1 and 3. The main reason for that is the fact that naive method 2 always reduces sending rate only by a fraction of the excess rate. As experiments showed, this reduction technique is unable to converge to the value of the sending rate that would eliminate congestion completely. As a result, with naive method 2 the ingress nodes receive congestion notifications almost throughout the whole simulation. Naive methods 1 and 3, on the other hand, converge to the correct sending rate fairly quickly, and as expected they require almost the same number of congestion notifications to converge.

Rate Reduction Method	***Congestion Period***		
	1-st	***2-nd***	***3-rd***
Naive 1	6.6	14.3	20.2
Naive 2	14.0	40.8	52.3
Naive 3	6.6	15.7	22.4

Table 7. Comparison of the methods for rate reduction

Also, as we expected, naive method 1 failed to avoid reducing aggregated rates below the minimum provisioned rates. However, both naive methods 2 and 3 were able to eliminate this problem. Figures 8 and 9 illustrate this by showing how the sending rates of the ingress nodes change during the congestion. As Figure 8 shows, naive method 1 reduces traffic load at ingress node 2 to 59 Kbps, while this node is provisioned for 70 Kbps on the path to the server. Naive method 3 on the other hand, does not reduce traffic loads below their minimum provisioned rates. As shown in Figure 9, both, ingress node 1 and ingress node 2 send traffic at rates slightly above their provisioned rates.

4.4 Evaluation of the per-flow rate distribution.

As we expected, using the method proposed in Section 3.3 allows the ingress nodes to fairly distribute excess bandwidth among their users. To illustrate how the ingress nodes distribute excess bandwidth among the users, we conducted a different set of simulations. In this simulation set, we allocated 140 Kbps of bandwidth at the bottleneck link, and we used naive method 3 for CRAR computation. In this scenario, we had only two congestion periods because the bottleneck link had more bandwidth allocated to it. As the results in Tables 8 and 9 show, the excess bandwidth is always distributed fairly among the users of the ingress nodes in proportion to their minimum provisioned rates. It should be noted that the fair share distribution among flows is only across flows in a given ingress node and is not valid for flows in different ingress nodes. Figures 10 and 11 illustrate these results in a graphical form.

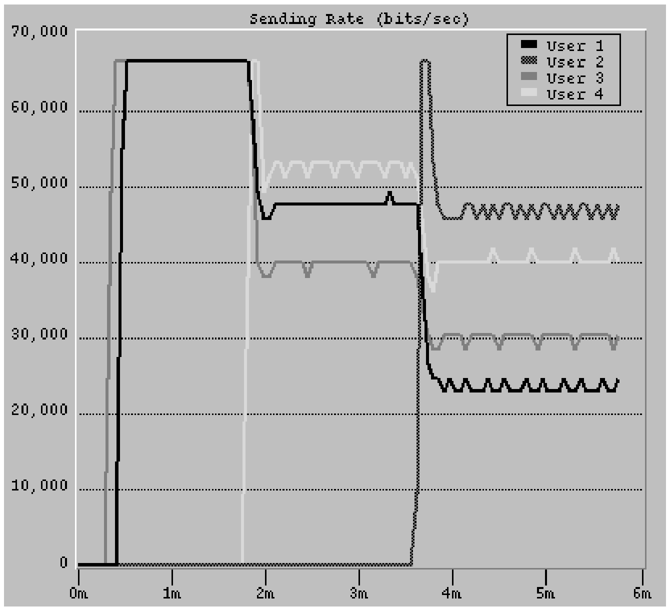

Figure 10: *Naive method 3:* Per-flow rate distribution.

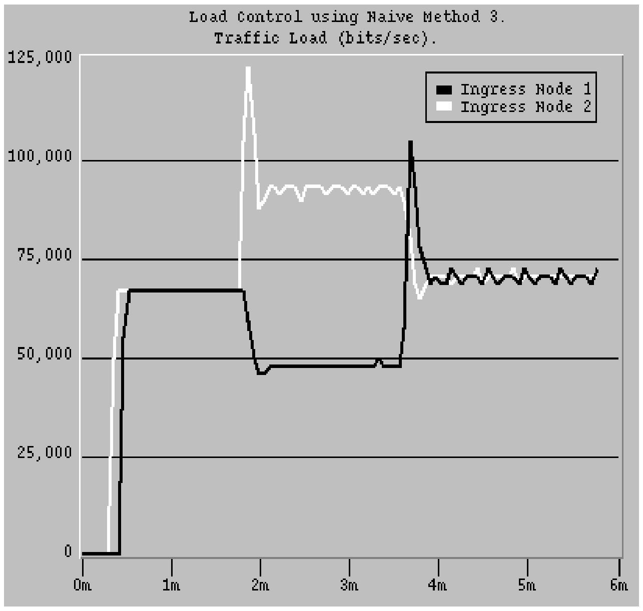

Figure 11: *Naive Method 3:* Ingress node sending rates variation.

	Minimum Provisioned Rates	**First Congestion Period**		**Second Congestion Period**	
		Sending rate	*Excess BW*	*Sending Rate*	*Excess BW*
Ingress Node 1	30 Kbps	47.3 Kbps	17.3 Kbps	68.5 Kbps	38.5 Kbps
Ingress Node 2	70 Kbps	92.6 Kbps	22.6 Kbps	71.4 Kbps	1.4 Kbps

Table 8. Rate distribution among ingress nodes.

		Provisioned Rate	**First Congestion Period**		**Second Congestion Period**	
			Sending rate	Fair share	Sending rate	Fair share
Ingress Node 1	User 1	10 Kbps	47.3 Kbps	37.3 Kbps	22.8 Kbps	12.83 Kbps
	User 2	20 Kbps	--	--	45.7 Kbps	25.67 Kbps
Ingress Node 2	User 3	30 Kbps	39.7 Kbps	9.68 Kbps	30.6 Kbps	0.6 Kbps
	User 4	40 Kbps	52.9 Kbps	12.91 Kbps	40.8 Kbps	0.8 Kbps

Table 9. Rate distribution among the users.

5. CONCLUSIONS AND FUTURE WORK

In this paper we introduced an architecture for load distribution in the ingress nodes of a network domain during periods of congestion. We also presented and evaluated four methods for distributed computation of the sending rates at ingress nodes. We found that the complex method for precisely estimating new sending rates does not work well because of the inaccuracy in the reported arrival rate at the congested interface. However, simple methods for reducing rate can eliminate congestion within a reasonable time and would not reduce traffic rates at ingress nodes below their corresponding minimum provisioned rates. Furthermore, we proposed a mechanism for fair rate distribution among the individual flows of the ingress node, and we showed using the OPNET [7] network simulator that this mechanism achieves fair load distribution among the flows. However, to better understand possible advantages of the proposed dynamic load control protocol, we need to investigate further the performance of the rate reduction mechanisms under a variety of network conditions.

6. ACKNOWLEDGEMENTS

We would like to thank Dr. Constantinos Dovrolis for the long hours of discussion and for helping to realize this idea.

7. REFERENCES

1. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss. "An Architecture for Differentiated Services", December 1998. IETF RFC 2475
2. H. Chow, A. Leon-Garcia, "A Feedback Control Extension to Differentiated Services", March 1999. Internet Draft: draft-chow-diffserv-fbctrl.txt
3. D. Clark, W. Fang, "Explicit Allocation of Best Effort Packet Delivery Service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362-373, August 1998
4. C. Dovrolis, P. Ramanathan, "A Case for Relative Differentiated Services and Proportional Differentiation Model," *IEEE Network*, Vol. 13, No. 5, pp. 26-34, Sep./Oct. 1999.
5. C. Dovrolis, D. Stilliadis, "Relative Differentiated Services in the Internet: Issues and Mechanisms," In *ACM SIGMETRICS*, May 1999.
6. C. Dovrolis, D. Stilliadis, P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," Proc. ACM SIGCOMM '99 Conference, Cambridge, MA, Sep. 1999, pp.109-120.
7. Wu-chang Feng, Dilip D. Kandlur, Dabanjan Saha, and Kang G. Shin "Understanding and Improving TCP Perfromance over Networks with Minimum Rate Guarantees," IEEE/ACM Transactions on Networking, Vol. 7, No. 2, pp. 173-187, April 1999.

8. OPNET Modeler. OPNET Technologies Inc. http://www.mil3.com
9. K. K. Ramakrishnan, Sally Floyd, D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", March 2001. Internet Draft: draft-ietf-tsvwg-ecn-03.txt
10. Rezende, J. F., "Assured Service Evaluation", In *Proceedings Globecom'99*, March 1999.
11. B. Nandy, N. Seddigh, P. Pieda, J. Ethridge, "Intelligent Traffic Conditioners for Assured Forwarding Based Differentiated Services Networks,“ In *Proceedings of IFIP High Performance Networking (HPN 2000)*, June 2000.
12. Peter Pieda, Nabil Seddigh, Biswajit Nandy, "The Dynamics of TCP and UDP Interaction in IP-QOS Differentiated Services Networks," In *Proceedings of the 3rd Canadian Conference on Broadband Research*, November 1999.
13. Ion Stoica, Scott Shenker, Hui Zhang, "Core-Stateless Fair Queuing: Achieving Approximately Fair Bandwidth allocations in High Speed Networks," Proc. ACM SIGCOMM'98 Conference, Vancouver, B.C., Sept. 1998, pp.118-130.
14. A.F. Lobo and A.S. Sethi, "A cooperative congestion management scheme for switched high-speed networks," Proc. ICNP-96, International Conference on Network Protocols, Columbus, Ohio (Oct.-Nov. 1996), pp. 190-198.