Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 53(67), Fascicola 2, 2008

Network-wide Proportional Services

Csaba Simon[1], Miranda Naforniţă[2]

Abstract – We proposed an end-to-end relative differentiation scheme to assure Quality of Services of IP network in a predictable manner, called network-wide proportional service. Quality of Service is applied to flow aggregates and the performance parameter of the classes is the goodput of these flows. We also present an algorithm, which computes the bandwidth of the flows required to sustain the model and the considered architecture. We verify then the proposed algorithm by simulations for both UDP and TCP traffic.

Keywords: Quality of Service, Differentiated Services, Proportional Services, UDP, TCP

I. INTRODUCTION

The new innovative services offered in modern time IP networks require a certain Quality of Service (QoS) support from the network side. The Integrated Services (IntServ) architecture [1] was an early answer for such needs, but the inherent scalability problems of this model hindered its widespread application. Opposed to the per-flow handling policy of IntServ, a newer approach, called Differentiated Services (DiffServ)) [2] defines service classes for flow aggregates. The major advantage of the DiffServ architecture is that it provides good scalability, which is very important in large networks. When any subscriber to a certain class sends some traffic into the network, it is tested against the conditions that define that class and only then is it allowed to enter the network. This process is called *admission control*. Later on, during the lifetime of the flow, it is continously monitored and *shaped* to fit the profile of the class and in case it ecxeeds it, the flow is *policed*. Nevertheless, this service model has its drawbacks of its own.

One of the greatest drawbacks comes from the static allocation of the internal queuing parameters. Since the weights are pre-allocated, a change in the offered load among classes cannot be handled in a predictable manner. In the worst case, when subscibers of a premium class are overloaded (but still within the bounds set for that class), and the customers using lower classes are scarce, it may happen that the communication flow in the premium class experiences worse services than the ones directed to lower classes.

To eliminate this problem, the relative service differentiation [3] has been introduced. In this approach, similarly to DiffServ, the traffic flows are grouped in a number of well-ordered service classes. In this context though there is no admission control and resource reservation, it is up to the users and applications to select the class that best meets their requirements, cost and policy constraints.

The proportional differentiation model ‘spaces’ certain class performance metrics proportionally to the differentiation parameters that the network administrator determines [4]. For example, if we consider a differentiation between **m** different classes, and q_i is such a performance measure for class i, the Proportional Service (PropServ) model imposes constrains of the following from all pairs of classes:

$$\frac{q_i}{q_j} = \frac{c_i}{c_j}, \qquad i,j = 1,2,\ldots,m \qquad (1)$$

where $\mathbf{c_1} < \mathbf{c_2} < \ldots < \mathbf{c_m}$ are the generic quality differentiation parameters (QDP). Thus the quality ratio between classes remains fixed, and controllable by the network operator, independent of the class loads. This control comes with the price of relaxing the guarantees offered within a certain class. That means that the actual quality level of each class **varies** with the class loads, the model only guarantees that a „better” class will **always** offer better services to any of its flow than a „worst” class. Additionally, the relative ordering between classes is **consistent** and **predictable** from the perspective of the user.

The relative differentiated services work on a Per Hop Basis [4], therefore it can be categorized as a DiffServ variant, thus the achieved **end-to-end** service differentiation is hard to control. To maintain scalability, algorithmic complexity should be pushed to the edges of the network whenever possible. To correct this inefficiency, a network-wide service differentiation was proposed, called network-wide proportional service [5]. This paper presents detailed simulation results that validate this model for both UDP and TCP traffic.

The paper is organized as follows: section II introduces the network-wide proportional service

[1] Department of Telecommunication and Media Informatics, Budapest University of Technology and Economics, E-Mail: simon@tmit.bme.hu
[2] Electronics and Telecommunications Faculty, ”Politehnica” University Timişoara, E-Mail: monica.nafornita@etc.upt.ro

model. Then an algorithm is given with different variants to UDP and TCP traffic, followed by a section detailing the results of simulated behavior of this algorithm. Finally we conclude our paper.

II. NETWORK-WIDE PROPORTIONAL SERVICE MODEL

The Proportional Service architecture presented in the previous section has some drawbacks which result from the PHB-based definition of the model. To correct this inefficiency, a network-wide service differentiation was proposed, called network-wide proportional service [5]. In this approach we aimed to obtain **network level** proportional differentiated service, based on the goodputs of the flows.
The goodput, **G**, gives the fraction of the offered load (the data sent) that is actually transmitted through the network (achieved throughput). In order to facilitate the deployment of this approach in practical networking environments we defined it over an administrative domain.

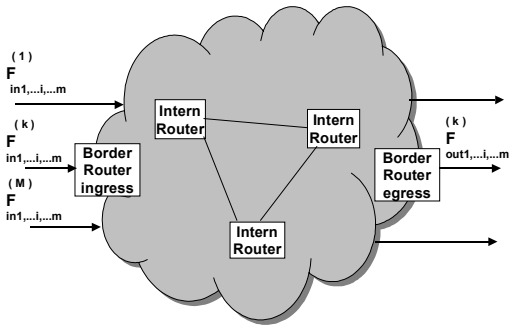

Figure 1. Network domain scenario

Let us consider a network, which forms an administrative domain with **m** different QoS classes. The flows with the same ingress and egress points are aggregated; according to the paths they are crossing the domain. Let us denote with **(k)** a path, where **(k)** connects an ingress **a** with an egress **b**. We note with **N** the total number of **(k)** paths in the network. Over any **(k)** path there are **m** flows (any flow is now a micro-flow aggregate), denoted $\mathbf{F_i^{(k)}}$, where the lower index, **i** = *1,2,...,m*, identifies the QoS class the flow belongs to (see Fig. 1). For each flow, **F**, we have the offered load (input bandwidth, $\mathbf{F_{in}}$) at the ingress and the achieved throughput (output bandwidth, $\mathbf{F_{out}}$) at the egress. With these notations we have the following equation defining the goodput of a flow:

$$G_i^{(k)} = \frac{F_{i,out}^{(k)}}{F_{i,in}^{(k)}} \qquad (2)$$

Thus, based on equation **(1)** for each **(k)** path, the relation between the performances of different classes is:

$$\frac{G_i^{(k)}}{G_j^{(k)}} = \frac{\frac{F_{i,out}^{(k)}}{F_{i,in}^{(k)}}}{\frac{F_{j,out}^{(k)}}{F_{j,in}^{(k)}}} = \frac{c_i}{c_j}, \quad i,j = 1,2,...,m \qquad (3)$$

In our model we define unique network-wide QoS classes, which yield same $\mathbf{c_i}$ parameters for a given class over any **(k)** path. Using unique $\mathbf{c_i}$ parameters for all of the paths makes the model simpler and easier to control. Costumer satisfaction may be reached by carefully choosing the pacing between the classes. Without loosing the generality of the model, in the following we consider only two classes, for the ease of presentation, the higher and a certain lower QoS class, indexed by m (the higher classes index) and **i** (certain classes index).
Based on the notations introduced above we can define the basic formula of the proportional service model:

$$\frac{G_m^{(k)}}{G_i^{(k)}} = \frac{\frac{F_{m,out}^{(k)}}{F_{m,in}^{(k)}}}{\frac{F_{i,out}^{(k)}}{F_{i,in}^{(k)}}} = \frac{c_m}{c_i} = \alpha_i, \qquad (4)$$

$\alpha_i > 1$, $i,j = 1,2,...,m$, $k = 1,2, ..., N$, where, $\mathbf{c_m}$ and $\mathbf{c_i}$ are network wide differentiation parameters for the first (highest) and the second (a certain) QoS classes, and they ratio is represented by $\boldsymbol{\alpha_i}$.
This equation shows us that the goodput of the best quality class (which has the highest index, **m**) is $\boldsymbol{\alpha_i}$ time bigger then the goodput of certain **i** class.

A. Fairness considerations

If we rewrite equation (4), we get the following relation:

$$\frac{F_{m,out}^{(k)}}{F_{m,in}^{(k)}} = \alpha_i \frac{F_{i,out}^{(k)}}{F_{i,in}^{(k)}}, \quad \alpha_i > 1, \quad k=1,2,...,N \qquad (5)$$

Equations (5) defines the relation of the class-level flows sharing the same path in the network, but it does not adress the relation between the flows of different paths, sharing a link in the network.
According to the service differentiation approach, the competitive flows should share the common resources equally, proportional with their offered load. This desire is formulated in the next equation, which is called the *fairness condition*, and specifies the relation among concurrent flows, sharing the same link along their path, as they cross the network,

$$\frac{F_{out}^{(k)}}{F_{in}^{(k)}} = \frac{F_{out}^{(l)}}{F_{in}^{(l)}}, \qquad k,l=1,2,\ldots,N \qquad (6)$$

whenever these two paths, (**k**) and (**l**), share the *same bottleneck link.* It has to be emphasized that this last equation, holds only for those flow pairs that

experience the same loss rate at the same bottleneck link in the network.

B. UDP and Proportional Services

The proportional differentiated service model initially was designed to satisfy the needs of real time traffic, which usually is transported over UDP [5]. Since UDP does not restrict the offered load of the source, the traffic sensed at the edges (borders) of the network is exactly the traffic transmitted by the application (supposing that the access capacity is large enough). In these conditions the proposed network-wide proportional service model naturally can be adapted to UDP flows. At any ingress point the measured aggregate throughput for a given class gives the $\mathbf{F_{i,\,in}^{(k)}}$ used in equations (1) – (5).

C. TCP and Proportional Services

TCP flows use a specific end-to-end (host-to host) mechanism to control the traffic, and to avoid congestion collapse [6, 7]. TCP flows guarantee that every sent packet will arrive to the destination and the TCP source elastically adapts to the state of the network. This renders useless the definition of "offered load" of the flows in the case of TCP traffic. To overcome this problem we will use a different definition for the offered load $\mathbf{F_{i,\,in}^{(k)}}$ in case of TCP flows. $\mathbf{F_{i,\,in}^{(k)}}$ will be proportional with the number of micro-flows in class **i** over the path **(k)**. In this approach the entering load of a certain class will be as follows:

$$F_{i,\,in}^{(k)} = n_i \bullet D, \qquad i = 1,2, \ldots ,m \qquad (7)$$

where $\mathbf{n_i}$ is the number of micro-flows in the certain flow-class,

D is a network wide constant,

$\mathbf{F_{i,\,in}}$ is the offered load for a certain, **i** class

Using large D values will certainly lead to overload during the initial phases of the algorithm. Since TCP would anyway rise the sender's throughput to fill even the largest link, it was important to avoid the situation where the theoretical offered load is less than the largest capacity along the path. Once the ratios among flows are correctly established, the absolut value of the troughput is not important, since the algorithm will give us the correct, loss-less value anyway.

III. THE PROPOSED ALGORITHM

Given a network with known offered loads at the ingresses and known paths within the network we can determine the bandwidth allocation among the flows in such a way that a.) there will be no losses in the network and b.) the throughput of each flow will conform to equations (4) and (6), describing the proportional service and fairness criteria, respectively. This section presents an iterative algorithm that computes these bandwidth values and discusses the differences that must be taken into considerations when the algorithm is applied to UDP or TCP traffic.

Assume, that in the network domain scenario presented in Fig. 1 we have **m** different links with different capacities ($\mathbf{C_i}$, where **i = 1,…,m**), and there are **N** different paths for the flow aggregates. The tightest bottleneck along the path of a flow defines the *overload factor*, **γ**, for that given flow. The task of any iterative step is to find the tightest bottleneck within the network and calculate the bandwidths of all flows crossing the bottleneck according to the enumerated criteriae. The bottleneck can be found by searching for the link with the highest utilization. The link utilization is computed by summing up the offered load for all flows that cross the link divided by the capacity of the link, as follows:

$$\rho_i = \frac{\sum_{(k):i\cap(k)\neq O} \sum_{j=1}^{m} \frac{F_j^{(k)}}{\alpha_j}}{C_i}, \text{ for all links } \boldsymbol{i} \text{ in the network} \qquad (8)$$

where $\boldsymbol{\rho_i}$ – represents the utilization of link **i**;

$\mathbf{F^{(k)}_j}$ – represents the offered traffic (load) of flows from path **(k)**, belonging to flow-class **j;** note that we sum up only those paths which contain link **i**

$\mathbf{C_i}$ – is the capacity of link **i,** and

$\boldsymbol{\alpha_i}$ – represents the proportional coefficient between the best flow-class, and flow-class **i.**

Note that a link is overloaded only if is $\boldsymbol{\rho_i}$ value is greater the one. The most heavily loaded link is chosen next, whose index is given by:

$$b = \arg\max \rho_i \qquad (9)$$

The $\boldsymbol{\gamma_j}$ parameter for the link with the highest utilization is the following:

$$\gamma_i = 1 / \rho_i \qquad (10)$$

If $\boldsymbol{\gamma_i}$ *is at least one*, all higher-class traffic demands will de satisfied since the links are under utilized (thus all micro-flows from flow-class **m** will get their resource requirements and they would not experience losses). The remaining task is to distribute the free remaining free bandwidth to satisfy as much lower flow-class requirements as possible.

In the other case, when $\boldsymbol{\gamma_i}$ *is less than one*, link **b** is a tight bottleneck. All flows that cross this link will suffer since the network cannot serve their initial offered load. In this case the achievable throughputs (the bandwidths) of flows sharing the same path, **(x)** (path which contains also the **b** link), are defined by the next equation:

$$F_{i,out}^{(x)} = \frac{\gamma_i}{\alpha_i} F_{i,in}^{(x)} \qquad (11)$$

The excess traffic ($\mathbf{F^{(x)}_{i,\,in} - F^{(x)}_{i,\,out}}$) would be lost at the bottleneck link anyway so it is desired to block this traffic at the ingress. By placing traffic shapers/policers at the edge (border) routers, the congestion collapse from undelivered packets can be avoided. A flow, which has already been shaped to the bottleneck link, is called *fixed*. Thus all the flows that are passing through this bottleneck link will be used in the next iterations of the algorithm with their "fixed" bandwidth values.

As the algorithm iterates, it will 'fix' the flows one after the other. We repeat this procedure, every time when we find a bottleneck link. The algorithm ends, when every flow will be fixed, or when we would not have any more bottleneck links.

A. *Applying the algorithm to UDP flows*

We considered two variants of the algorithm. The first variant achieves global optima in terms of network utilization, because at every change in the offered load of a flow it recomputes the output for all the flows in the network. That is we consider the effects of the change of the load in one flow on all the flows in the network. In the second variant we use a prediction mechanism, where we estimate the offered load at the ingresses based on the traffic measured over previous periods. This makes the implementation of the algorithm much simpler.

B. *Applying the algorithm to TCP flows*

Theoretically, all the values should be recomputed, if a TCP flow enters or exits the network. Note that in normal networking condition, the number of flows presets in the network has a huge value. Thus the algorithm will not be resumed at every single TCP flow entering or exiting, because the modifications introduced by a determined number of flows can be tolerated for the good functionality of the model. In this condition, we will fix a limit for the number of micro-flows. In the moment when, after the last resume of the algorithm, the number of micro-flows entering or exiting the network reaches the determined limit, the algorithm is restarted.
We implemented in the ingress nodes the BLUE active queue management [8], to avoid network congestion, and to assure the calculated bandwidths (throughputs) for the different micro-flows.

IV. PROPORTIONAL SERVICE SIMULATION

A. *Simulated environment*

In this section we present the results of investigation by means of simulations using the *ns2* tool [9], using the network topology as presented in Fig.2.
The scenario shown on Fig.2 is complex enough to validate our proposal. We have four traffic generators, all of them generated both high and low class traffic flows and use UDP or TCP at the transport level.
In case of **UDP traffic** we used two types of simulations, in the first case with constant bit rate (CBR) traffic only (*static* traffic conditions), then with a variable traffic (*dynamic* traffic conditions) where a more complex and thus more realistic traffic pattern is used to reassure the efficiency and viability of the proposed algorithm. Variable traffic can be modeled by the aggregation of a number of CBR sources. In our simulation micro-flows join and leave the aggregate flow according to a Poisson process with the same arrival and departure rate (i.e., $\alpha_{arr} = \alpha_{dep} = 1/4\ sec^{-1}$).

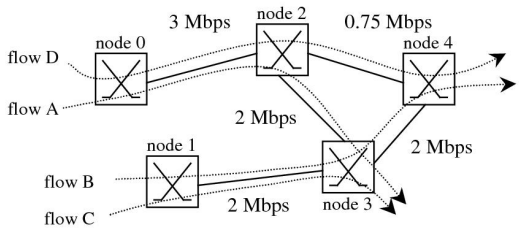

Figure 2. Simulated topology

The average bandwidth of each aggregate is set to 1 Mbps, and the bandwidth of each micro-flow is exponentially distributed with mean of 32 kbps. We used a class based queuing (CBQ) scheme to shape the flows at the ingress. In all simulations we set the target V ratio to a value close to one (i.e., $\alpha = 1.1$).
For **TCP traffic** we simulated a static and dynamic scenario as well. The network topology and the flow paths were the same. For the static scenario we generated 50 TCP streams for the high-class flow *A* and 60 streams for the low-class flow A (50/60 micro flows in flow *A*). For the rest of the flows these values were 40/30 micro flows for flow *B*, 30/50 micro flows for flow *C*, and 60/40 micro flows for flow *D*.
In order to generate variable traffic load in the TCP scenarios, we modified the number of the above enlisted TCP streams. We changed the number of streams at one second intervals. The numbers of micro flows within a certain class were changed by stopping an existing or starting an additional TCP stream. In order to decide whether to stop, start or leave unmodified the TCP streams, we relied on the random generator of the Linux OS.

B. *Simulation with static traffic conditions*

The correctly working Proportional Services architecture drops the packets that would overload the core links at the ingress. That is, there should be no packet loss inside the network.

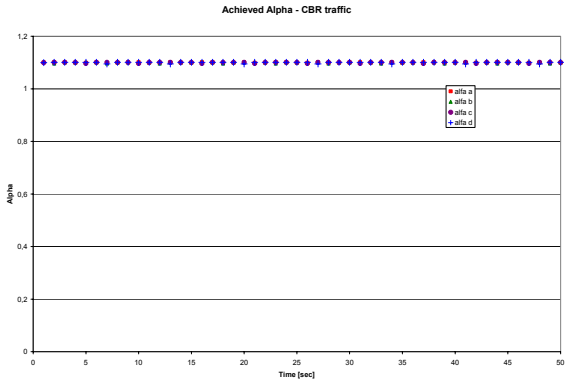

Figure 3. Static UDP traffic scenario

This is the first criteria that should be verified. The way we implemented the simulator, the losses appear on the first, incoming links. The rest of the links should be loss free. We run the simulation for the static traffic scenarios with different α values: one very close to 1 (as a worst case scenario) and one larger one (to test the behavior of the protocol in a different parameter range). We selected α = 1.1 for UDP and α = 2 for TCP transport protocols and analyzed these links inside the network, and they had no losses. Therefore the model passed the first test.
Then we took the simulated UDP scenario and verified the resulting proportionality coefficients (α). The results are shown in Fig.3. As one can see, the simulation results were in good agreement with the theoretical results.

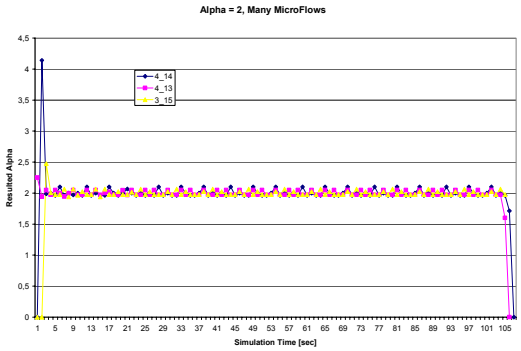

Figure 4 Static TCP traffic scenario

In the case of TCP flows we did the same analysis, with α = 2 used to loss test, as well. Fig. 4 shows the result of this simulation. The variance of the monitored value appears because the monitoring interval separates some packets, starting to arrive very close to the sampling period's end, and a part of it will arrive in the next second. From practical point of view we considered that the result shows that under static traffic conditions the architecture achieves the targeted behavior, thus the simulation validates the proposal.

C. *Simulation with variable flows*

Then we turned to test the behavior of the model in dynamic traffic conditions, as well.

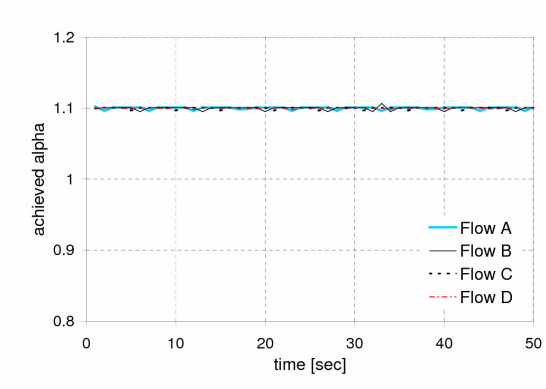

Figure 5 Dynamic UDP scenario

First we analysed the case of *variable UDP flows*. The traffic model is described in section IV.A and tries to approximate an aggregated flow, which is dynamically joined and left by micro-flows. Using this scenario we tested whether the algorithm can respond to small changes in offered load. We considered a network, where each flow provides its own traffic profile, thus we applied the first variant of the algorithm described in section III.A.
Fig. 5 presents the achieved α ratio. It can be seen from the plot that the α ratio is kept very close to the 1.1 value. (The α values were computed for every second.) The highclass component of flow *A* matches the offered load, and the components of flow *D* suffer the most severe limitation, similarly to the previous case.
Then we made the same test for *variable TCP flows*, as well. In this case we used the α = 1.1, in order to detect any disfunctionality of the model that eventually may appear. The traffic model is different for the UDP model due to the particularities of TCP and is described in section IV.A. The results are presented in Fig.7. As one can see the proportional coefficients are close to the target value of 1.1. The explanations for the variations are the same as for the static TCP traffic scenario.

D. *Simulation for UDP flows with measurement-based predictions*

Next we refined our architecture, and introduced a measurement-based prediction to approximate the offered load as input for the algorithm, as mentioned in section III.A. We used a simple prediction scheme: the prediction takes the average load of the last five seconds as an approximation for the next five seconds.

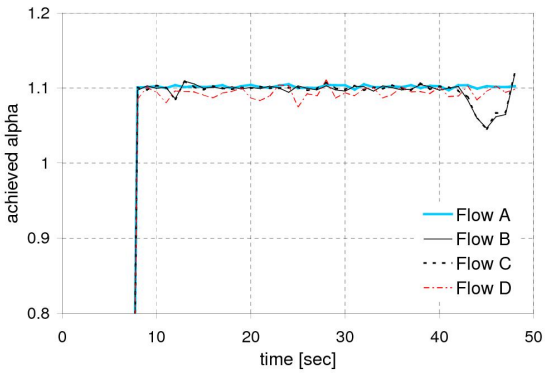

Figure 6 Measurement-based predictions

The first prediction interval starts from 2 seconds when all components of the flows are active. (Thus the relevant result starts after 7 seconds of simulation time.) As it can be seen on Figure 4 the output is mostly correct, it keeps the parameter between 1.11 and 1.07. In the last period the 0.1 deviance from the expected value should be further studied and eliminated by refining the prediction schemes.

V. CONCLUSION

In this work an end-to-end relative differentiation scheme, called network-wide proportional service has been proposed. Apart from the already available proposals for per-hop performances, this one is defined over an administrative domain. The flows are aggregated based on their ingress and egress points and the performance parameter of the classes is the goodput of these flows.

Then we presented an algorithm proposal, which computes the flow-shares required to sustain the model and the considered architecture. An important aspect of the model is that the resource intensive flow handling is pushed to the edges of the network domain. We verified then the proposed algorithm by simulations, for both UDP and TCP flows.

For UDP we tested a measurement-based prediction scheme to increase the scalability of the model. The model works even with this simple prediction sheme, but is some cases a more advanced prediction might be required. We also proposed a solution to serve TCP traffic, using a new shaping mechanism at the ingress nodes, the BLUE AQM, initially developed for congestion control. The implementation results validated the proposals for both, the algorithm and shaping mechanism.

With the proposed solutions, the original Proportional Service architecture can be deployed in IP networks to support the novel multimedia services.

REFERENCES

[1] R. Braden, D. Clark, and S. Shenker**,** *Integrated Services in the Internet Architecture: An Overview*, IETF RFC 1633, June 1994

[2] S. Blake et al., *An Architecture for Differentiated Service*, IETF RFC 2475, December 1998

[3] C. Dovrolis, P. Ramanathan, *A Case for Relativ Differentiated Services and the Proportional Differentiated Model*, IEEE Network 1999, September 1999

[4] C. Dovrolis, D. Stiliadis, P. Ramanathan, *Proportional Differentiated Services: Delay Differentiation and Packet Scheduling*, ACM SIGCOMM 1999, September 1999

[5] Cs. Simon et al., *End-to-End Relative Differentiated service for IP Networks,* Proceedings of IEEE ISCC 2002, June 2002

[6] W. Stevens, *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*, RFC 2001, January 1997

[7] J. Postel, *Transmission Control Protocol,* IETF RFC 793*,* September 1981

[8] W. Feng, D.Kandlur, D. Saha, K. Shin, *BLUE: A New Active Queue Management Algorithms*, Project Report CSE-TR-387-99, University of Michigan, April 1999

[9] DARPA, *The Network Simulator – ns-2*, http://www.isi.edu/nsnam/ns, SAMAN Project, 1995.

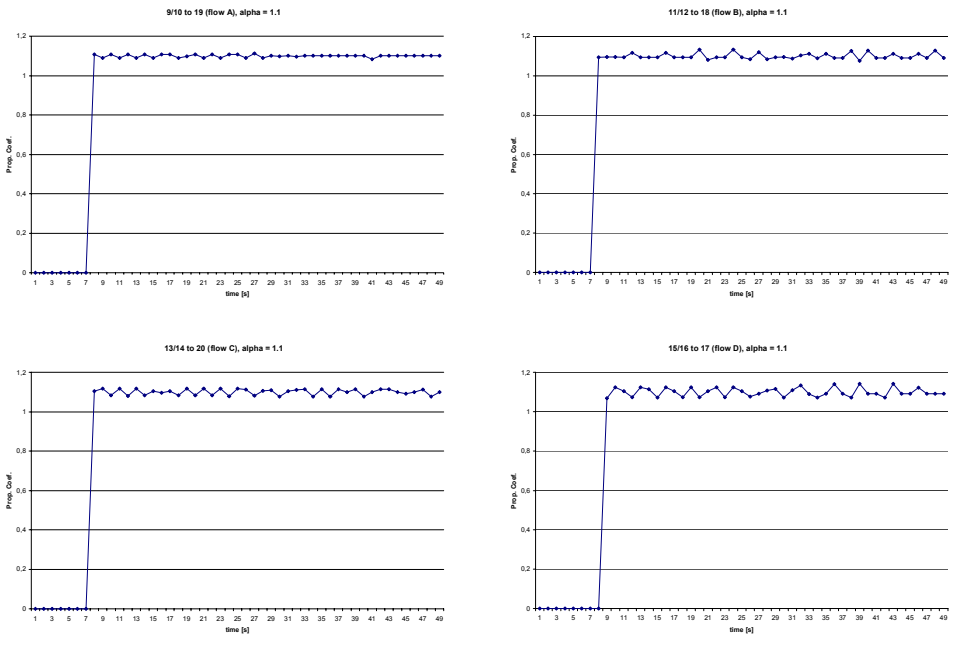

Figure 7 Dynnamic TCP scenario