

# Adaptive Measurement Based QoS Management in DiffServ Networks

Chamil P. W. Kulatunga, Paul Malone, Mícheál Ó Foghlú  
Telecommunications Software Systems Group (TSSG)  
Waterford Institute of Technology, Ireland.  
Email: {ckulatunga, pmalone, mofoghlu}@tssg.org

## Abstract

*In per-flow based resource reservation techniques, bandwidth allocation for Variable Bit Rate (VBR) applications through real time flow measurements is limited. This limitation is due mainly to scalability problems. In aggregated traffic flow networks such as Differentiated Services (DiffServ), measurement-based bandwidth allocation leads to optimisation of network resources as it links resource allocation to actual usage. With Constant Bit Rate (CBR) applications, simple summation of pre-specified bandwidths is a sufficient solution for resource optimisation in DiffServ networks. However, the pre-specified bandwidth technique is an inefficient approach to DiffServ bandwidth allocation for VBR applications. These applications are generally unaware in advance of their specific bandwidth requirements. This paper describes an adaptive Quality of Service (QoS) management technique that provides an efficient resource allocation scheme for VBR applications in aggregate traffic flows. This method is, in effect, a measurement-based and linearly predicted bandwidth allocation scheme.*

**Index Terms**—DiffServ, Linear Prediction, QoS Manger, Token Bucket Filter.

## 1. Introduction

The technique presented in this work operates through the combination of user agents and centralised QoS managers. The user agent processes metered, real-time bandwidth data averaged over a predefined time interval to forecast expected future bandwidth requirements. These forecasts are calculated through the application of a linear prediction algorithm on the metered data. When a forecast falls beyond a predefined acceptable range above or below the current allocated bandwidth, the user agent informs the local QoS manager of its revised bandwidth requirement.

When a QoS manager performs a bandwidth re-negotiation it signals this change to the next QoS manager responsible for bandwidth allocation on the data path. Using this technique, bandwidth allocations are propagated along the data path to the destination QoS manager.

The method provides an efficient use of resources by taking aggregated predicted traffic requirements into consideration when making decisions in the QoS managers about bandwidth allocations. This efficiency is due to the technique making resource allocations based on actual measured data and as such relates actual system state and usage with resource allocation. Allocation of pre-defined bandwidth leads to poor resource utilisation and can result in customers being charged for resources they have not consumed. The technique under consideration provides a guaranteed service for VBR traffic in DiffServ, as it allocates varying bandwidth rather than a pre-defined and is of particular use when users are charged for actual usage.

The work is presented as part of the CONVERGE project, which provides an architecture to deliver end-to-end QoS guarantees in DiffServ domains through the use of centralised QoS managers. A QoS manager in an ingress domain responds to user bandwidth requests for DiffServ services. With EF (Expedited Forwarding) service, the QoS manager does simple summation of pre-specified bandwidth calculation of the allocated flows to provide admission control for a user request. Upon the availability of bandwidth, the QoS manger configures the edge router to police the user traffic to an agreed bandwidth. The technique demonstrated in this paper is applied as an alternative to simple summation of pre-defined bandwidth for admission control to improve resource utilisation in the CONVERGE end-to-end QoS provisioning architecture in DiffServ networks.

## 2. Context

DiffServ [1], [2] provides a scalable QoS solution based on per class resource reservation avoiding the limitations of IntServ per-flow resource reservation throughout the time period of the flow. But in providing a service using DiffServ, forwarding treatments of the DiffServ classes will not be enough. DiffServ combines with an admission control mechanism and considers Service Level Agreements (SLA) [3] that represent agreement between the customer and the network operator. Topology aware admission control and end-to-end service guarantees have to be fulfilled without end-to-end dynamic signalling. A QoS manager (per domain) addresses the key issues of DiffServ in providing a service for end-to-end guarantees [4].

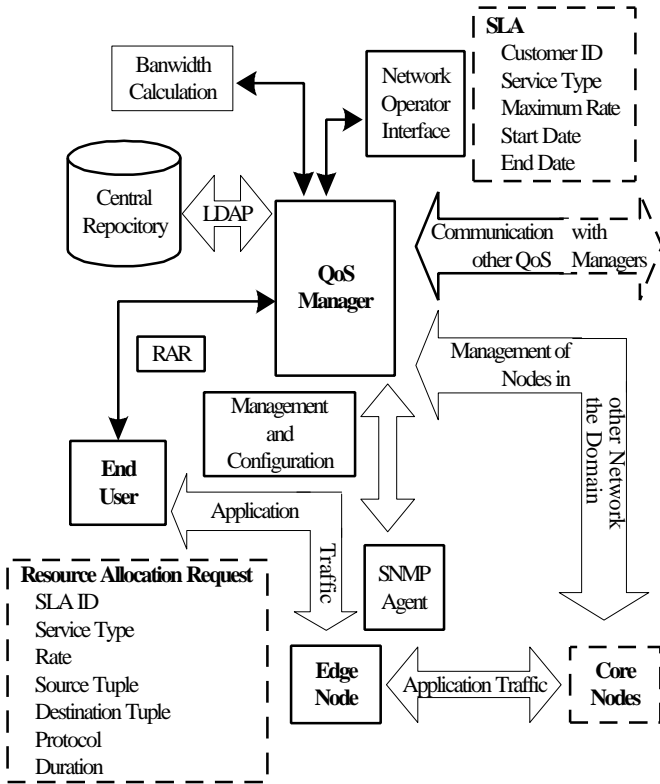


Fig. 1. Inter-Domain DiffServ QoS Management Architecture

As shown in Fig. 1, the user requests the required bandwidth from the QoS manager of his administrative domain. Taking into account the destination, the QoS manager traces the path using the routing tables read from the nodes in its own domain and QoS managers along the data path. It then checks the available bandwidth from its allocated flows in the central repository and other QoS managers. If the new customer has enough resources available to allow the end-to-end provision, it makes an allocation and allows the user to use the service. Billing and authentication can be incorporated into this model quite easily. This briefly describes the way DiffServ provides services to the customers using its forwarding treatments.

## 2.1 Bandwidth Allocation for Pre-Defined Traffic Profiles

In the above architecture the user should request one of the DiffServ service types, EF or Assured Forwarding (AF) [5], [6] as well as the flow specification. Flow specification is described using the Token Bucket Filter (TBF) [1], [7] as indicated in Fig. 2.

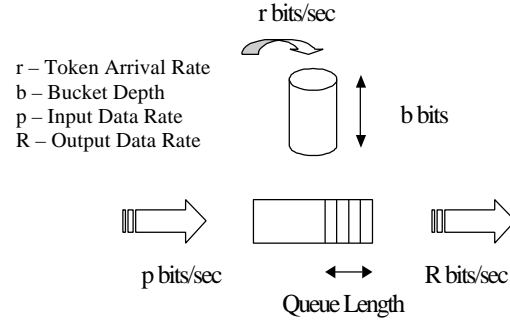


Fig. 2. Token Bucket Filter

When  $p > r$ , packets are stored in the queue until there are enough tokens in the bucket. Once a packet is issued an equal number of tokens to the packet size is removed from the bucket. Long term averaging rate for token arrival rate ( $r$ ) and maximum burst size for bucket depth ( $b$ ) are selected to describe and limit the traffic flowing out of the TBF. The QoS manager allocates bandwidth proportional to these parameters for the requested flows and polices at the edge node.

Once a new request is made, the QoS manager performs the admission control as flows for the bandwidth.

The QoS manager's decision-making engine performs a calculation for the requested bandwidth according to the equation 1.

$$b_k \leq B - \sum_{i=1}^n b_i \quad (1)$$

Where  $b_k$  is the current user requested bandwidth from the QoS manager,  $B$  is the total bandwidth configured for the user requested service by the network operator and  $b_i$  is the allocated bandwidth of the  $i^{\text{th}}$  flow. There are  $n$  flows already allocated on that particular interface.

Similarly, in buffer management, user requested maximum burst size is considered with the ingress router interface buffer size.

Once an allocation is made each QoS manager along the data path keeps a reference to it in its own central repository while the flow is active and considers this for all admission control requests. When a flow is ended, the QoS manager that communicates with the user informs the other QoS managers to release the allocations of that flow.

## 2.2 Adaptive Bandwidth Allocation

The difficulties of pre-specification of traffic can be solved with real time traffic metering [8] by a user agent, which informs the bandwidth changes to the QoS manager in regular intervals as show in Fig. 3.

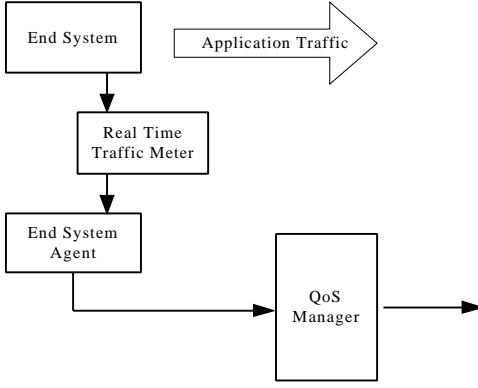


Fig. 3. Meter Agent and QoS Manger

Instead of the meter agent residing at the end-user system it can be located in the ingress of the user's DiffServ network. Buffer management for real time monitoring is not discussed in the work presented here.

In real time bandwidth allocation, a linear prediction technique is used to calculate the next bandwidth. In general  $k$  step linear prediction equation of  $p^{\text{th}}$  order is expressed in equation 2 [9].

$$\bar{x}(n+k) = \sum_{l=0}^{p-1} x(n-l)w(l) \quad (2)$$

Where  $\bar{x}(n+k)$  is the predicted value for  $k$  steps ahead,  $x(n)$  is the  $n^{\text{th}}$  real time measured bandwidth,  $w(l)$  s are the linear prediction coefficients to define or calculate so that the error due to prediction can be minimized.  $p$  is the window size of the moving average.

In our case a simple prediction scheme is used since the data used here are not highly correlated enough to use for calculation of linear prediction coefficients leading to minimising errors. Calculating linear prediction coefficients for highly correlated traffic using an Adaptive Linear Prediction (ALP) is beyond the scope of this paper.

In this analysis we use moving window averaging in calculating predicted bandwidth. In moving average with window size of  $p$ , linear prediction coefficients are:

$$w(0) = w(1) = w(2) \dots \dots \dots = w(p-1) = \frac{1}{p}$$

Only one step ahead is predicted ( $k=1$ ) therefore the predicted bandwidth can be calculated according to equation 3.

$$\bar{x}(n+1) = \frac{1}{n} (x(n) + x(n-1) \dots \dots + x(n-p+1)) \quad (3)$$

The ultimate objective of linear prediction is to minimize the error due to the prediction with the actual values. The error due to prediction is:

$$Error = x(n+1) - \bar{x}(n+1)$$

In analysing the accuracy of linear prediction the Mean Square Error (MSE) is defined as in equation 4 [9].

$$MSE = \sum_{n=0}^{N-1} \frac{(x_n - \bar{x}_n)^2}{N} \quad (4)$$

Where  $\bar{x}$  is the predicted value,  $x_n$  is actual metered bandwidth and  $N$  numbers of samples are considered.  $x_0$  is assumed as the pre-defined bandwidth.

### 3. Testbed

Fig. 4. shows the DiffServ testbed used in the experiments. Mainly it comprised two DiffServ Linux routers and two end user PCs running Windows 2000. Robust Audio Tool (RAT) [10] is running on both Widows PCs.

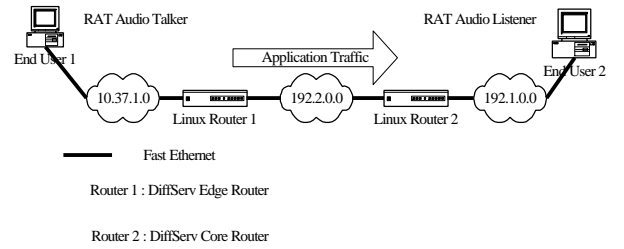


Fig. 4. DiffServ Testbed

Network Traffic Meter (*NeTraMet*) [11] was used for real-time bandwidth monitoring at the two Linux routers. Bandwidths were measured in 1-second intervals. *NeTraMet* Remote Console (*nm\_rc*) with Simple Ruleset Language (SRL) [12] was used to read traffic flows specified according to the rules written in SRL.

The Linux *iproute2 tc* utility [13] was used to control the traffic in DiffServ implementation. In this configuration Router 1 is the DiffServ edge router and the Router 2 acts as a pure DiffServ core router [14]. The edge router polices the incoming traffic to the specified TBF rate and burst size. It also remarks the DS-field to the DiffServ CodePoint (DSCP) used in EF Per Hop Behaviour (PHB) [15]. All other traffic that does not intend to use EF traffic class is remarked with Best Effort (BE) PHB traffic class. Fig. 5 shows the script that causes Router 1 to function as a DiffServ edge router [13].

```
Link='dev eth1'
Rate1='rate 15Kbit'
Burst='burst 2K'
Action='drop'
Match1='match ip src 10.37.1.39 match ip dst 192.1.0.20 match ip dport 5020 0xffff
match ip protocol / 17 0xff'
Match3='match ip src 0/0'
Meter1='police $Rate1 $Burst $Action'
Meter3='police $Rate3 $Burst $Action'
tc qdisc add $Link handle 1:0 root dsmark indices 64
tc class change $Link classid 1:1 dsmark mask 0x3 value 0xb8
tc class change $Link classid 1:2 dsmark mask 0x3 value 0x0
tc filter add $Link parent 1:0 protocol ip prio 1 handle 1: u32 divisor 1
tc filter add $Link parent 1:0 prio 1 u32 $Match1 $Meter1 flowid 1:1
tc filter add $Link parent 1:0 prio 1 u32 $Match3 $Meter3 flowid 1:2
```

Fig. 5. DiffServ Edge Router Configuration Script

DSMARK queue and U32 filter are used for packet remarking and per-flow classification respectively. According to the script, UDP traffic from End User 1 to End User 2 at destination port 5020 will be policed at a rate of 15Kbps and maximum burst size of 2Kbytes. Non-conformant traffic is dropped. Packets on this traffic stream are marked with DSCP 0xb8, which is the codepoint used in DiffServ EF PHB. All other traffic that does not match with the above flow are marked with DSCP 0x0 the codepoint use with BE PHB.

Linux Router 2 acts as a DiffServ core router and its functions are limited only to forward packets based on DS-field and reserve bandwidth to DiffServ traffic classes EF PHB and BE PHB only the DiffServ traffic classes we use in these experiments. Fig. 6 shows the script that causes Router 2 to behave as a DiffServ core router [13].

```
Link='dev eth0'
tc qdisc add $Link handle 1:0 root dsmark indices 64 set_tc_index
tc filter add $Link parent 1:0 protocol ip prio 1 tcindex mask 0xfc shift 2
tc qdisc add $Link parent 1:0 handle 2:0 cbq bandwidth 100Mbit cell 8 avpkt 1000
/mpu 64

EFRate='rate 200Kbit'
tc class add $Link parent 2:0 classid 2:1 cbq bandwidth 100Mbit $EFRate avpkt 1000
/prio 1 bounded isolated allot 1514 weight 1 maxburst 10
tc qdisc add $Link parent 2:1 pfifo limit 5
tc filter add $Link parent 2:0 protocol ip prio 1 handle 0x2e tcindex classid 2:1
/pass_on

BERate='rate 70Mbit'
tc class add $Link parent 2:0 classid 2:2 cbq bandwidth 100Mbit $BERate avpkt 1000
/prio 7 allot 1514 weight 1 maxburst 21 borrow split 2:0 defmap 0xffff
tc qdisc add $Link parent 2:2 red limit 60KB min 15KB max 45KB burst 20 avpkt 1000
/bandwidth 100Mbit probability 0.4
tc filter add $Link parent 2:0 protocol ip prio 2 handle 0 tcindex mask 0 classid
/2:2 pass_on
```

Fig. 6. DiffServ Core Router Configuration Script

DSMARK queue and TCINDEX filter are used with Class Based Queue (CBQ) in packet classification based on DS-field and bandwidth reservation to EF and BE classes. According to the script 200Kbps of bandwidth is reserved to EF PHB traffic that has DSCP marked with 0xb8. All other traffic other than DSCP 0xb8 uses the bandwidth that forward packet with DSCP with 0x0 and bandwidth of 70Mbps.

#### 4. Tests

Three audio samples of two men and a woman were used in the tests. These samples of lengths of 180S, 90S and 45S were run on the testbed using *RAT*. End User 1 was used to play the samples and End User 2 acted as the listener. While the samples played the bandwidth was monitored on the routers using *NeTraMet*.

Next the edge router policing TBF parameters were set at 20Kbps, 25Kbps, 30Kbps and 35Kbps with the burst size set to 2Kbytes. The core router EF PHP bandwidth was set to the policing rate in each case. The three samples were run in each configuration. The packet loss percentage was calculated at the interfaces of Router 1 and Router 2 using *tc*. The bandwidth was also measured with *NeTraMet*.

Finally the results were analysed by calculating the MSE for different pre-defined bandwidths and for moving average bandwidth with the actual bandwidths.

## 5. Results

Table 1 shows the Mean and the Standard Deviation (SD) of the three different audio samples run through the network with *RAT*.

Sample	Mean (bps)	Standard Deviation
Sample 1 (180S)	27096.56	13134.94
Sample 2 (90S)	31439.72	11885.13
Sample 3 (45S)	32322.34	11227.18

Table 1. Mean and SD of Three Samples

According to Table 1 there is a considerable difference in the three samples even though they are in the same category of Internet traffic. The SD gives a general indication of the burstiness of traffic and the difference between the samples used.

Table 2 shows the effect of these different behaviours of traffic in DiffServ domains.

Sample	Loss Percentage (%)	
	Edge Router Policer	DiffServ Core Router
Police at 20 Kbps and Burst Size 2 Kbytes; EF Bandwidth Reservation 20 Kbps		
Sample 1	18.65	3.79
Sample 2	20.52	1.96
Sample 3	22.56	1.26
Police at 25 Kbps and Burst Size 2 Kbytes; EF Bandwidth Reservation 25 Kbps		
Sample 1	4.51	3.39
Sample 2	9.96	2.84
Sample 3	12.23	2.32
Police at 30 Kbps and Burst Size 2 Kbytes; EF Bandwidth Reservation 30 Kbps		
Sample 1	0.00	0.00063
Sample 2	0.31	0.00070
Sample 3	0.78	0.00062
Police at 35 Kbps and Burst Size 2 Kbytes; EF Bandwidth Reservation 35 Kbps		
Sample 1	0.00	0.00
Sample 2	0.00	0.00
Sample 3	0.00	0.00

Table 2. Traffic Behaviour in DiffServ Domain

Table 2 shows the effect of three samples at the edge and the core routers in the DiffServ testbed. The burst size was kept the same for all four different bandwidths and the burst size was not considered in this analysis. The table shows the poor quality the traffic stream got if the user did not have a prior knowledge of the bandwidth behaviour of each of the three samples. Changing the policing from 20Kbps to 25Kbps the packet loss for Sample 1 is reduced considerably to a better figure. This is not the case for Samples 2 and 3, both of which still have a reasonably high packet loss ratio. However, when the policing is increased to 30Kbps Sample 1 encounters

no packet loss but may be over-provisioned. At this policy level Sample 2 and Sample 3 still suffer from packet loss.

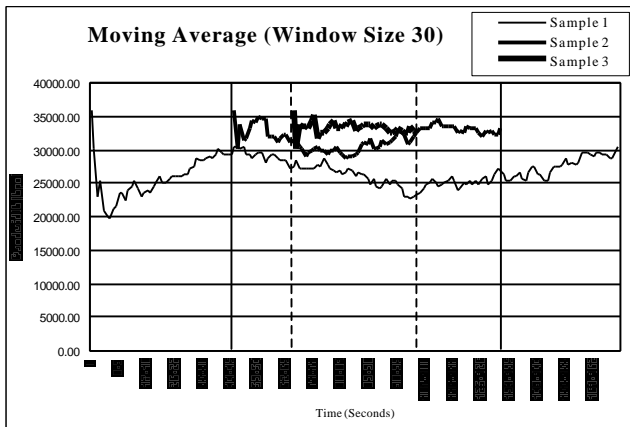


Fig. 7. Moving Average with Window Size 30.

Fig. 7 shows the moving average bandwidth for all three samples with a window size of 30. It shows the changes in the moving average, which fluctuate above and below the long-term mean. In moving average calculations initial user requested bandwidth was considered as the initial point.

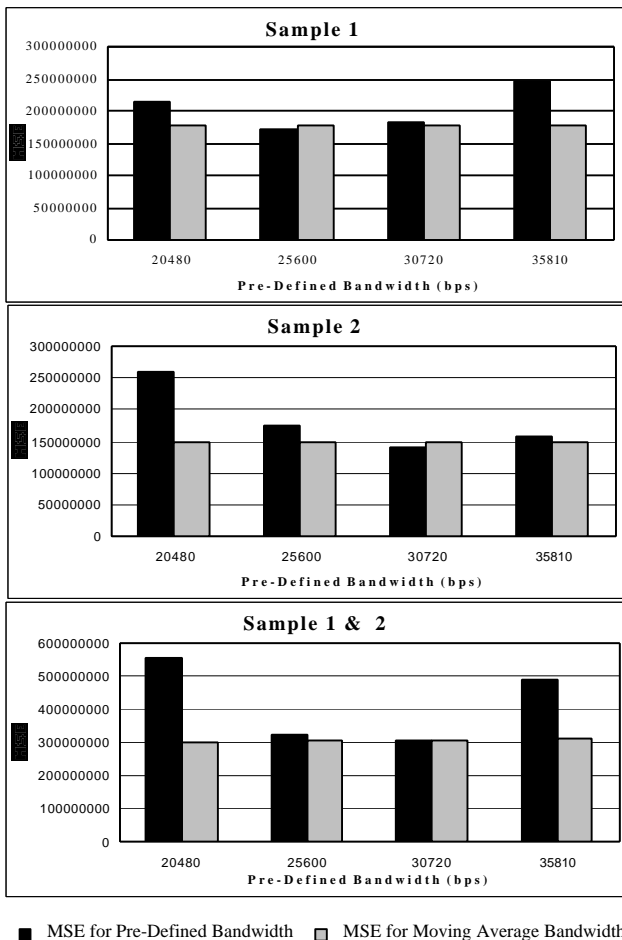


Fig. 8 MSE for Pre-Parameterised Bandwidth and Moving Average Bandwidths with Window Size 30.

Fig. 8 shows a comparison of the MSE (in  $\text{bps}^2$ ) due to pre-defined bandwidth and moving average bandwidths with a window size of 30 for Sample 1 and Sample 2 and also in traffic aggregation of two samples. The figure shows this comparison for pre-defined bandwidths of 20Kbps, 25Kbps, 30Kbps and 35Kbps.

It can be seen that both Sample 1 and Sample 2 incur a large MSE at 20Kbps and 35 Kbps. It can also be seen that there is a high risk in allocating a bandwidth that is not close to the mean of the traffic flow. This risk is to be expected if the bandwidth is allocated due to the pre-defined traffic specification rather than adaptive self-determination of the bandwidth requirement. This figure also shows that in DiffServ, traffic aggregation does not reduce this error for pre-defined traffic allocation.

At a bandwidth allocation of 25Kbps, Sample 1 it can be seen that the MSE of the moving average allocation technique is greater than the MSE for the pre-defined policy. Sample 2 at this bandwidth, shows that the moving average technique reduces the MSE.

At 30Kbps this situation is reversed, with Sample 1 showing a reduction in the MSE for moving average allocation and Sample 2 showing an increase.

In aggregation the moving average allocation technique shows a reduction in the MSE for both 25Kbps and 30Kbps. At bandwidth 25Kbps Sample 2 shows poor quality and at 30Kbps Sample 1 shows network resource over provisioning.

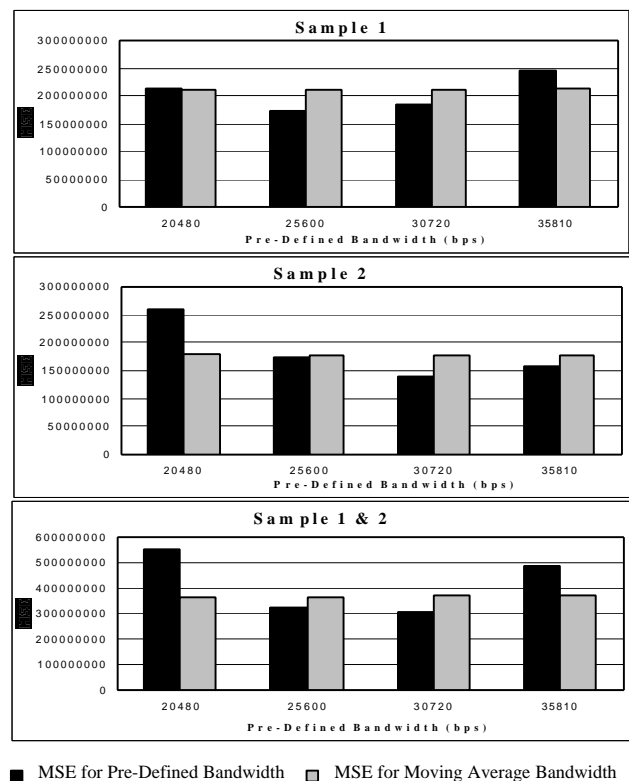


Fig. 9. MSE for Pre-Parameterised Bandwidth and Moving Average Bandwidths with Window Size 5.

Fig. 9 shows that when the window size is reduced to 5 the advantage of the moving average technique no longer proves to be a great success. The risk of larger MSE when allocating bandwidth that is not close to the mean remains.

Comparing Fig. 8 and Fig. 9 shows that when the moving average window size is larger the MSE of the sample gets lower and thus provides improved performance.

In consideration of scalability issues, a lesser MSE was observed for 2-second than 1-second sampling intervals with moving average window of 20. This indicated that a higher sampling period did not violate the advantages of the concept of adaptive bandwidth allocation technique.

## 6. Conclusions

This paper addresses the risks in resource allocation for pre-parameterised traffic flows in managed DiffServ domains through centralised QoS managers. Instead of pre-parameterised traffic profiles, adaptation for self-determination of required bandwidth is proposed. This leads to improving quality of service from the customer's point of view. Furthermore, from the network operator's point of view, this can reduce network resource under-utilisation. Combining real-time traffic metering, linear prediction technique and DiffServ traffic aggregation behaviour achieves this goal. This can be shown to be advantageous, especially for Variable Bit Rate (VBR) applications in DiffServ networks.

With the deployment of QoS in the Internet users might be given an interface to request the bandwidth for their application for example listening to an online lecture. This work has proved the risk of completely giving that responsibility to the end user. This implementation is expected to provide more effective means of bandwidth management irrespective of the additional network management overhead and the complexity of the QoS management architecture in DiffServ domains.

## 7. References

- [1] Zheng Wang, "Internet QoS, Architecture and Mechanism for Quality of Service", Morgan Kaufmann Publishers, 340, Pine Street, San Francisco, CA 94104-3205, USA, 2001.
- [2] S. Blake, D. Black, M. Carlson, E. Davies, "An Architecture for Differentiated Services", IETF RFC 2475, December 1998.
- [3] Afrodite Sevasti, "SLA Definition for the Provision of an EF-based Service", 16th International Workshop on Communications Quality & Reliability, CQR 2002, Okinawa, Japan, May 2002.
- [4] Internet2 QoS Working Group,  
<http://www.internet2.edu/qos/wg>.
- [5] B. Davie, A. Charny, J.C.R. Bennett, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", IETF RFC 3246, March 2002.
- [6] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group", IETF RFC 2597, June 1999.
- [7] J. Glasmann, M. Czermin, A. Riedl, "Estimation of Token Bucket Parameters for Videoconferencing Systems in Corporate Networks", 8th International Conference on Software,

Telecommunications and Computer Networks, SoftCOM 2000, Split, October 2000.

- [8] N. Brownlee, C. Mills, G. Ruth, "Traffic Flow Measurement Architecture", IETF RFC 2722, October 1999.
- [9] A. Adas, "Supporting Real Time VBR Video Using Adaptive Linear Prediction", Department of Electrical and Computer Engineering, Georgia Institute of Technology, IEEE Infocom, San Francisco, March 1996.
- [10] Robust Audio Tool (*RAT*), Available: <http://www-mice.cs.ucl.ac.uk/multimedia/software/rat>.
- [11] Real Time Traffic Flow Measurement Tool (*NeTraMet*), Available: <http://www.caida.org/tools/measurement/netramet>.
- [12] N. Brownlee, "SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups", IETF RFC 2723, October 1999.
- [13] Linux DiffServ Implementation (*tc*), Available: <http://diffserv.sourceforge.net>.
- [14] Y. Bernet, S. Blake, D. Grossman, A. Smith, "An Informal Management Model for DiffServ Routers", IETF RFC 3290, May 2002.
- [15] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", IETF RFC 2474, December 1998.