A Simple Pricing Scheme for DiffServ Networks

Liang Ji[1] , Theodoros N. Arvanitis[2] and Nicholas J. Flowers[3]

Department of Electronic Electrical and Computer Engineering
School of Engineering, The University of Birmingham,
Edgbaston, Birmingham, B15 2TT, United Kingdom
[1] liangj@eee.bham.ac.uk [2] t.arvanitis@bham.ac.uk [3] n.j.flowers@bham.ac.uk

Abstract: The Differentiated Service (DiffServ) model has been proposed as an efficient and scalable service architecture to deploy different services on the Internet. In this paper, we propose a new pricing model for DiffServ. We use a marking/dropping mechanism incorporated in our pricing scheme to apply pricing in congested DiffServ networks. The proposed scheme is simple to implement and can give users incentives to control their behaviours by charging the network resources they have used during periods of congestion, while a user does not have to pay fees when the price ceiling has been reached.

Key words: Differentiated Service, Quality of Service (QoS), Pricing, Internet

1. INTRODUCTION

The Internet Engineering Task Force (IETF) proposed the Differentiated Service (DiffServ) as a simple framework to provide Quality of Service (QoS) for various applications in packet-switched networks, and in particular for the Internet [1]. DiffServ is designed to offer services to aggregated traffic, where a number of individual flows are grouped and treated as a single traffic flow throughout transmission in the DiffServ network.

Before entering a DiffServ network, each user has already achieved an agreement with their service provider. Such an agreement is called the Service Level Agreement (SLA) and can be either static or dynamic. Every IP packet is classified by a traffic classifier according to its priority,

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35673-0_28

C. McDonald (ed.), *Converged Networking*

previously set by the user. Following this the processes of metering, admission control and possibly traffic shaping take place. The conditioned traffic flows are then assigned to specific behaviour aggregates (BAs) by marking/identifying the IP header's DS field (Differentiated Services field) with appropriate DSCPs (Differentiated Services Code Points). The Per-Hop behaviours (PHBs) of all the intermediate DiffServ-compatible routers within the DiffServ domain are then achieved when these aggregated traffic flows are forwarded in the interior part of the DiffServ network. DiffServ pushes most of the complexities of the services to the edge of the network, gaining a much better scalability than the Integrated Service (IntServ) that has to keep a large amount of per-flow status information in every interior router.

Because of its favourable features, such as scalability and simplicity, DiffServ has become one of the most important research topics in the area of network QoS. However, if the current approach to a flat-rate pricing remains as the only pricing mechanism for DiffServ, 'the tragedy of the commons' [2] becomes inevitable, where some users take unreasonable large shares of the network resources and cause congestion, while all users are charged at the same price. This is often explained by the lack of proper incentives to control their behaviours.

It is normal to assume that DiffServ should use a differentiated charging scheme, which applies different prices for different classes of services provided to customers, where for instance a higher price corresponds to a higher class of service. Users should be able to subscribe to appropriate service classes, based on their 'willingness to pay' [3]. One of the pricing schemes, based on such an approach is the 'Paris Metro Pricing' [4] proposed by Odlyzko. Inspired by the operation of the Paris Metro network, it partitions the packet-switched network into separate zones with a fixed price for each zone. Although this scheme exhibits simplicity in its implementation, it has no traffic management or service guarantees.

Our research has identified that very few schemes address the issues of pricing details in the DiffServ networks. Moreover, most of to date research only focuses on the theoretical design for pricing schemes [5][6][7], while offers limited discussions on their practical implementations. Another important issue of DiffServ pricing is its simplicity. We believe that a complicate pricing mechanism may not only increase the computational network overheads but may also suffer from the complexities of its implementation. Furthermore, a simple pricing algorithm integrated into a DiffServ network is the only way to keep the scalability of DiffServ meaningful. Therefore, our work aims to combine the aforementioned issues of scalability and simplicity for our pricing scheme.

The paper is organized as follows. Section 2 gives the detailed description of our pricing scheme. Section 3 presents some further analyses

for our proposal. Section 4 provides our critical discussion on the advantages and disadvantages of the proposed scheme and presents our overall conclusions.

2. PROPOSED PRICING SCHEME

Many small-scale network service providers usually change a flat rate to all users for accessing the variety of classes of service available in the network. To avoid any resulting congestion, the service provider may over-provision for the available network resources. However, this solution is not cost-effective. Therefore, to make the most efficient use of available network resources and provide a relatively fair QoS to each user, a pricing scheme is needed to force users to 'behave accordingly'. Similar to Varian's 'smart market' proposal [8], we argue that there should be no charge to the users when there is no congestion in the network. Hence the charging here is solely for the purpose of alleviating network congestion via constrained user behaviours.

2.1 Basic Network Model

Figure 1 illustrates a conceptual network model, used for demonstrating our pricing scheme. Before a user's data flow comes into a DiffServ domain, the user assigns each of their application a priority that is associated with a fixed price. The services mentioned here could be pre-defined services, such as the premium service [9], the assured service [10], or other user-defined service categories specified in the SLA. This assignment gives the user's 'willingness-to-pay' for each packet transmitted. We assume that the users are aware of the price for each service class, as announced by the network provider in advance.

When the data are transmitted across the DiffServ domain, the intermediate routers monitor the current network conditions. If congestion exists, the routers drop low priority packets while at the same time mark high priority packets. In our scheme, we use an agent called 'charging agent' at the egress of the DiffServ domain. It's purpose is to count all the marked packets, and to calculate the charges for every user, while feeding back the billing information to the source, where the users can adjust their behaviours according to this feedback information and their 'willingness-to-pay'.

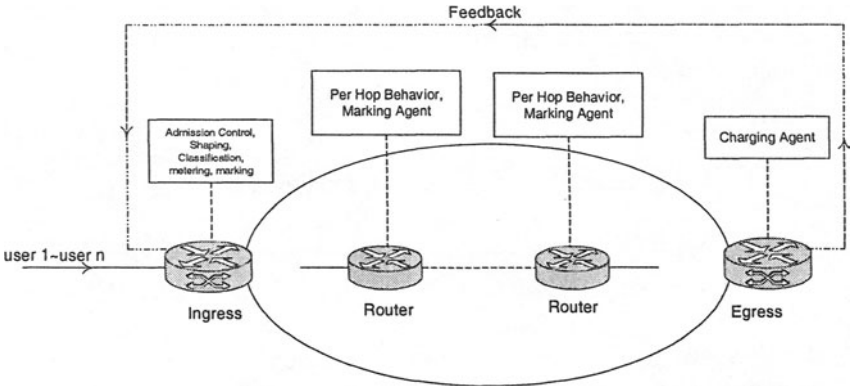

Figure 1. Network Model

2.2 Marking and Dropping

In our proposed scheme, the price of a transferred packet for each service class is fixed, where a packet with a higher priority has a higher 'value' (price) compared to a packet belonging to a lower priority class. We choose this fixed pricing scheme to avoid the high computational overheads of dynamic pricing for obtaining the final charges. Moreover, together with this fixed pricing allocation, we only need a 'two-bit charging algorithm' to acquire the charge for each user. This makes the whole pricing mechanism even simpler.

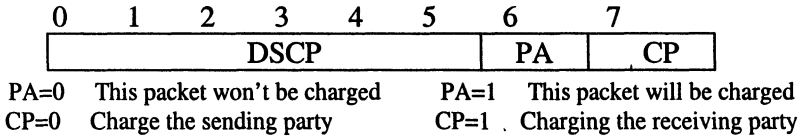

Figure 2. The DS Field

We utilize a new marking mechanism: if a router marks a packet during the transmission, the price of this packet will be charged. We call this marked bit in the DS field the Pricing Active (PA) bit. We propose to use the 6th bit of the DS field in the IP header, one of the undefined bits in DS field, as the PA bit (Figure 2). It is not certain whether the sender or the receiver should be charged; this is normally decided based on the nature of the application. For example, the sender should be charged when he is phoning a party via IP telephony, while the receiver should be billed if he requests to download a file from a server. Hence, we define the 7th bit as the Charging

Party (CP) bit to indicate which party of the communication should be charged.

In the proposed scheme, the Head-of-the-Line (HOL) queueing is preferred (Figure 3). When a new packet with priority N comes, it finds its position in the tail of the priority N portion by following a pointer. The reason for choosing the HOL queue is that it reasonably controls the difference of time delays between high and low priority classes of traffic. Schwartz has shown that higher priority packets can gain a much lower average waiting time in a HOL queue than in a FIFO queue at the expense of the average delay for low-priority packets [11]. This is due to the 'conservation law': the weighed sum of wait times for all packets is always reserved [12]. In addition to delay, our scheme takes into account the problem of packet loss. This combination makes the system more scalable for both delay-sensitive and loss-sensitive applications. However, the operational details of the queue still need to be carefully designed to cope with the DiffServ environment.

During congestion, which can be signalled by the queue building up or by the long propagation delays as indicated in [13], the marking is performed from the highest priority class to the lower-priority portion of the queue.

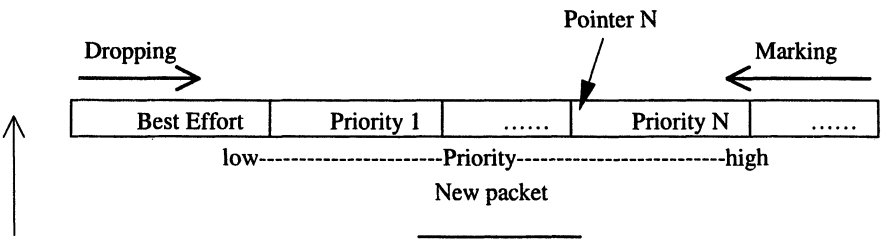

Figure 3. Head-of-the-Line Queue

At the same time, the dropping is performed in the opposite direction: it starts from the lowest priority class—best effort (Figure 3). This is to ensure lower dropping possibilities for the higher priority packets, which conforms to our design goal: those who willing to pay more should enjoy a better service.

As soon as one low priority packet is dropped, one high priority packet is marked, i.e. the PA bit is set to one. As seen in Figure 3, we use two pointers to point to the next packet to be dropped or marked, respectively. After marking a packet, the marking pointer moves towards the low priority end of the queue, pointing to the next unmarked high priority packet. It can be considered that the dropping triggers the marking. And these actions

continue until the congestion in this hop is alleviated. This marking mechanism makes the marked packets with the highest priority to have the lowest 'dropping priorities' in return of the high prices they will be charged.

2.3 Charging

The charging agent shown in Figure 1 calculates the marked packets and charges the users according to the priorities of these packets. Then it feeds back the information for the charges to the users at a specified time period, which is similar to the switch-averaging interval in ATM networks. The agent counts the marked packets during this interval and sends out the changes at the end of each period. We name this interval the 'charging interval'. The feedback can be implemented as in-band or out-band, depending on the length of the charging interval and the traffic overheads it may cause. We describe the behaviour of the charging agent at the egress of the network in the pseudo-code fragment below:

```
At the beginning of each charging interval:
   For i = 1 to n
        /*n is the number of users*/
        {For j = 0 to m
             counter_ij = 0; }
         /*m is the number of available service classes*/
At the end of each charging interval
   {For i = 1 to n
        {calculate the charge for user i in this interval;}
    Send the charging result back;
      /* The result is a n-vector */ }
When a marked packets of user i with priority j come
   counter_ij ++ ;
```

The users can change their service allocations in different classes according to the information contained in a returned n-vector called Char_N. If it is the receivers that receive bills, they can send requests to the senders to change the service allocations via feedback.

$$\text{Char\_N} = \{P_{sub\_0}, P_{sub\_1}, \ldots P_{sub\_i}, \ldots, P_{sub\_n}\} \quad (1)$$

P_{sub_i}: The subtotal charge for user i in the last charging interval

If Char_N = 0, it means that there is no congestion. Then the users may increase their rates or give higher priority to some applications. If timeout,

the feedback may be lost or delayed in the congested network. In this case, users should control their traffic or prepare to be charged with higher fees. The following pseudo-code fragment describes the actions that users can take when timeout happens:

```
If (not timeout) Then
  {If (Char_N = 0) Then
    {Users may increase their rate or
     remain the current states}}
   Else          /* Char_N≠0*/
    {Users take appropriate actions after
     knowing the fees they have to pay;}

  Else           /* Timeout*/
  {Users control behaviours or are ready to receive
higher charges if they do not change their allocations }
```

There is a need to alleviate the timeout problem, in the future.

3. FURTHER ANALYSIS

Let $S=\{0,1,...,\}$ denote a finite service set provided by the DiffServ domain, where priority 0 means the best effort service and m stands for the highest service priority. Let $P = \{0, p_1,..., p_m\}$ be the price set where each price gives the expected charging per unit transmitted for a corresponding service class, and

$$0<p_1<...<p_m \tag{2}$$

where,

$$p_i<p_j, \quad \text{if } i, j \in S, i < j \text{ and } p_i, p_j \in P \tag{3}$$

3.1 Price ceiling

Note that the prices introduced here may not be actually charged for every transmitted packet, if the network is not congested as mentioned before. Hence the price set P and a user's own choice of service allocation actually provide a 'pricing ceiling', which gives them a rough idea of what they might have to pay at most. This is important because normal users like to have some idea about how much they would have to pay to benefit from the services. The 'pricing ceiling' can make the aggressive users to control their behaviours and reconsider the submission of high bandwidth-

consuming applications. On the other hand, common users will not overreact to the possible charges of their services and restrain appropriately their usage of the network resources, if they aware of these ceilings. For a certain user $i \in N = \{1,...n\}$, the price ceiling can be given by:

$$A_{max\_i} = \sum_{j=0}^{m} p_j z_j \quad \text{for } i \in N, j \in S \tag{4}$$

Where z_j is the size of the data block that user i plan to transfer in priority j.

Sometimes the users may not be able to figure out the approximate data volume they are about to consume. Hence, they may either overestimate or underestimate the charges they will receive afterwards. However, we believe that they would be willing to adjust their rates or service allocations in their preferred ways after receiving some feedback information.

3.2 Final charge to each user

As we described in section 2, in our pricing scheme not all transmitted packets are charged. Instead, when congestion occurs only high-priority packets are charged. The sub-total charges for users are sent back from the egress point at the end of each charging interval. The subtotal charge for user i in a charging interval is included in the feedback vector *Char_N*:

$$P_{sub\_i} = \sum_{j=1}^{m} p_j C_{i\_m\_j} \tag{5}$$

where the counter is the marked packet counter for priority j packets of user i

And the total charge for user i for the whole session:

$$C_{t\_i} = \sum_{t} (P_{sub\_i})_t \qquad \text{t: charging interval } t \tag{6}$$

The price ceiling of user i can be given as:

$$C_{pc\_i} = \sum_{t} (P_i)_t \qquad \text{t: charging interval } t \tag{7}$$

Where,

$$P_i = \sum_{j=1}^{m} p_j C_{i\_a\_j} \tag{8}$$

$C_{i\_a\_j}$ is the number of all priority j packets from user i:

Obviously, $C_{t\_i} \leq C_{pc\_i}$ (9)

Actually, if the network is not heavily congested, the real charge is expected to be much less than the ceiling price.

3.3 User's Goal

Suppose user i has $r \in R$ applications to transfer simultaneously, where R is the application set. We denote the concave increasing utility function $U_i(T)$, which only depends on the effective bandwidth T for user i. Here we may regard the utility of user i as the degree of satisfaction in the sense of monetary.

Where,

$$T = \sum_{j=1}^{r} b_j \qquad j \in R \tag{10}$$

b_j : average bandwidth taken by user i for application j

We denote user i's utility in charging interval t as $U_t(T_t)$, then we can get:

$$T_t = \sum_{j=1}^{r} b_{j\_t} \qquad j \in R \tag{11}$$

b_{j_t} : average bandwidth taken by user i for application j during charging interval t.

Hence for user i, their aim is to maximize the utility after reducing the cost:

$$\max\left\{U_i(T) - \sum_{j=1}^{r} q_j z_j\right\} \tag{12}$$

where z_j : number of marked packets of application j for user i and q_j: price of a marked packet from application j

$$q_j = p_{j'} \qquad j \in R,\ j' \in S, p_{j'} \in P \tag{13}$$

Because the charging is performed periodically, we should consider the utility in one charging interval t. Suppose the charging interval is a fixed period of time T_c. For charging interval t, the problem of (12) becomes:

$$\max\left\{U_i(T_t) - \sum_{j=1}^{r} q_j b_{j\_t} T_c\right\} \tag{14}$$

Hence, the solution is:

$$U_i'(T_t)\Big|_{b_{j\_t}} = q_j T_c \tag{15}$$

Equation 15 shows us that the selections of priorities that set the prices of packets from application j is closely related to the utility maximization goal that the user want to achieve during charging interval t.

3.4 Equilibrium

In the application of game theory [14], a cooperative game may get a better payoff for each user because Nash equilibria may be pareto inefficient in many situations. In this paper, we only consider the relationship between users as a non-cooperative game. The reason for this is that it is not possible to make users have any 'agreements' among them in packet-switched networks, while they can hardly have any information of other users' behaviours. Theoretically, for finite number of users, if each user has finite number of allocation choices and mixed strategies are allowed, there exists at least one Nash equilibrium in this game. However the users can only choose pure strategies to play this game. So they may not be able to achieve Nash equilibria.

Assume that there exists a Nash equilibrium called E_n. At E_n, each user chooses a pure strategy from one of the available service allocations. The user has no intention to change the service allocation while no other users keep their status unchanged. To achieve the equilibrium, there are two prerequisites: a) There are no user/application joining or dropping off dynamically, which is unrealistic in many situations b) there exists at least one user behaviour pattern that can achieve a Nash equilibrium in the finite number of possible pure strategies. The number of possible patterns is:

$$\sum_{i=1}^{n}(app_i)^m \qquad \text{app}_i \text{ :the number of applications of user } i \tag{16}$$

For bursty elastic traffic [15] in a single link model, Marbach [15] has shown that there is an equilibrium for non-cooperative games. However, this work does not consider any impacts from queuing, propagation delays, and possible users/applications joining/dropping dynamically. Furthermore, we believe that we should take the whole network as a single entity to consider this issue. To analyse equilibria in a multi-hop DiffServ domain with different user applications, further work is needed. For more realistic traffic patterns where users can add/drop any of their applications dynamically at any time, we doubt that an equilibrium could possibly be achieved in such a situation. We plan to investigate this through simulation studies for such versatile environment. However, this does not affect our conclusion that users can gain right incentives to control their behaviours when a usage-based pricing scheme is applied.

4. DISCUSSION AND CONCLUSION

In our model, it is the users who decide which class of service they should subscribe to for each of their applications. Some may argue that

delay-sensitive applications, such as real time video/audio programs, should have higher priorities than relatively delay-insensitive applications, such as email. However, we believe it is the users' own interest to determine the degree of importance for each application they are employing. To a certain user, an email carrying critical company data may be far more important than a real-time streaming video being shown for pure entertainment. With the emerging of new applications, it is also unpractical to keep the network aware of everything new and rearrange the priority orders, in time. Therefore, we believe that making the users to choose their own service allocations is more practical, while exploiting the good adaptability of our proposed pricing scheme.

Our scheme is similar to the 'smart market' proposal in the sense that all users place their 'willingness-to-pay' to the network where they obtain much lower charges, in most cases. However, our scheme has several important features that make it different from the 'smart market' approach. Firstly, the price selections are limited to N-1 choices (N is the number of priorities available). On the contrary, within the smart market method the number of choices is infinite, thus increasing substantially the implementation complexity. Secondly, the final charges to the users are based on the number of marked packets and their associated prices. In contrast, the final charges for 'smart market' are the same for all the admitted packets, i.e. the highest bid of the rejected packets. This may actually introduce more computing overheads, to be able to identify the specific 'clearance price' by a comparison process, each time the bidding is taking place.

The selection of the charging interval is also a non-trivial issue because it in fact determines the delay between the time of congestion and its effects being brought back to the users. It also decides the frequency by which users can change their service allocations. If congestion happens, the faster the feedback the higher possibilities can be obtained for users' controlled behaviours, to further reduce the congestion downstream. Meanwhile, it also means the creation of heavier feedback overheads to the already congested network. The interactions between feedback and the user reactions are not clear at this stage and require further investigation.

In this paper, we have proposed a simple and practical pricing mechanism for the DiffServ networks. We have in detail described our pricing model and discussed some further issues that are important for the scheme. We have also shown that the proposed pricing scheme may give users the appropriate incentives to adjust their behaviour under congestion conditions, by means of linking the final charges to their choices of service allocations. To gain more understanding on the operations of the whole system and the related user behaviour in real networks, further investigations such as realistic simulations are needed.

REFERENCES:

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

[2] Garret Hardin, "The Tragedy of the Commons", *Science*, 162: 1243-1247,1968.

[3] F. Kelly, A. Maulloo, and D. Tan, "Rate Control for Communication Networks: Shadow Prices, proportional Fairness and Stability", *Journal of the Operational Research Society*, 49(3): 237-252,1998.

[4] A.M.Odlyzko, "Paris Metro Pricing for the Internet", in *Proceedings of the ACM Conference on Electronic Commerce*, pp. 140-147, 1999.

[5] A.Orda and N. Shimkin, 'Incentive Pricing in Multi-Class Communication Networks', *INFOCOM*, 1997.

[6] Ayalvadi Ganesh, Koenraad Laevens and and R. Steinberg, 'Congestion pricing and user adaptation', *INFOCOM*, 2001.

[7] Ioannis Ch. Paschalidis and J.N. Tsitsiklis, 'Congestion-Dependent Pricing of Network Services', *IEEE/ACM Transactions on Networking*, 2000.

[8] J. MacKie-Mason and H. Varian, "Pricing the Internet", in *Public access to the Internet*, B. Kahin and J. Keller, eds., Englewood Cliffs, NJ, Prentice Hall, 1995.

[9] V. Jacobson, K. Nichols and K. Poduri. "An Expedited Forwarding PHB". *IETF Request for Comments, RFC2598*, June 1999.

[10] J. Heinanen, F. Baker, W. Weiss and J. Wroclawski. "Assured Forwarding PHB Group". *IETF Request for Comments, RFC 2597*, June 1999.

[11] M. Schwartz, *Telecommunication Networks: Protocols, Modeling and Analysis*, Addison-Wesley, 1987.

[12] L. Kleinrock, *Queueing systems, volume II: Computer Applications*, Wiley, New York, 1975.

[13] Fendick KW, Rodrigues MA and Weiss A, "Analysis of a rate-based feedback control strategy for long-haul data transport", *Performance Evaluation*, 16: 67-84, 1992.

[14] R. Gibbons, *Game Theory for Applied Economics*, Princeton University Express, 1992.

[15] Shenker S., "Fundamental Design Issues for the Future Internet", *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 7, pp. 1176-1188, September 1995.

[16] Peter Marbach. "Pricing Differentiated Services Networks: Bursty Traffic", in *Proceedings of the IEEE Infocom 2001 Conference*, Anchorage, Alaska USA , April 2001.