Efficient Resource Management for End-to-End QoS Guarantees in DiffServ Networks

Spiridon Bakiras and Victor O.K. Li
Department of Electrical & Electronic Engineering
The University of Hong Kong
Pokfulam Road
Hong Kong

***Abstract*— The Differentiated Services (DiffServ) architecture has been proposed as a scalable solution for delivering end-to-end Quality of Service (QoS) guarantees over the Internet. While the scalability of the data plane emerges from the definition of only a small number of different service classes, the issue of a scalable control plane is still an open research problem. The initial proposal was to use a centralized agent, called Bandwidth Broker (BB), to manage the resources within each DiffServ domain and make local admission control decisions. In this paper, we propose an alternative distributed approach, where the local admission decisions are made independently at the edge routers of each domain. We will show, through simulation results, that this distributed approach can manage the network resources very efficiently, leading to lower bandwidth blocking rates when compared to traditional shortest path admission control. Moreover, its simplicity and distributed implementation make it a very scalable solution for resource management in DiffServ networks.**

I. Introduction

THE Differentiated Services (DiffServ) architecture [1] has been proposed recently by the Internet Engineering Task Force (IETF) as a scalable solution for delivering end-to-end Quality of Service (QoS) guarantees over the Internet. The basic idea of this architecture is that only edge routers should manage traffic on a per flow basis. Core routers should not keep any kind of per flow state, and should process traffic on a much coarser granularity. At the data plane this goal is achieved by specifying different Per Hop Behaviors (PHBs), where packets belonging to the same PHB form a Behavior Aggregate (BA) and receive identical service at the core routers. More specifically, the edge routers will be equipped with flow classifiers, policers, and markers that will properly mark the incoming packets by setting a number of bits on the DiffServ Codepoint (DSCP) [2] field of the IP packet header. The DSCP value will indicate the corresponding PHB, and the core routers will forward the packets based on their DSCP value (by utilizing several scheduling and buffer management techniques).

The IETF has currently specified two different PHBs. The *Expedited Forwarding* (EF) PHB [3] offers the equivalent of a leased line (i.e. low delay, loss, and jitter) between a source and a destination. This is accomplished by giving EF traffic strict priority over the traditional best-effort traffic inside the DiffServ domain. However, each flow has to specify in advance the required bandwidth so that the appropriate resources can be reserved inside the network. In addition, the maximum burst size that is allowed is equal to two Maximum Transmission Units (MTUs). The edge routers will police each flow, and the non-conformant packets will either be dropped or shaped. The *Assured Forwarding* (AF) PHB group [4] does not offer hard QoS guarantees, but instead defines four different AF classes with three different levels of drop precedence within each class. Each AF class is assigned a certain amount of bandwidth at each node, and when the amount of traffic exceeds this bandwidth, packets will be dropped according to their drop precedence value. In this paper, however, we will focus on providing hard end-to-end QoS guarantees, so we will mainly concentrate on the EF PHB.

While the scalability of the data plane emerges from the definition of only a small number of PHBs, the issue of a scalable control plane is still an open research problem. The initial proposal was to use a centralized agent, called Bandwidth Broker (BB) [5], to manage the resources within each DiffServ domain and make local admission control decisions. The centralized approach removes the burden of admission control from the core routers, but there might be some scalability considerations if the BB has to process thousands of requests per second. Moreover, this approach has certain disadvantages that are inherent to any centralized architecture.

- The links around the BB will become very congested when the traffic load from the signaling messages is high.
- The BB must maintain per flow information about every flow that is currently active inside its domain.
- The BB is a single point of failure (i.e. undesirable in reliability considerations).

In this paper, we propose an alternative distributed approach, where the local admission decisions are made independently at the edge routers of each domain. The BB in each domain will be responsible for periodically updating the allocation of the resources inside the domain, according to some measurements of the traffic load at the edge routers. When the allocation of resources is completed, all the edge routers will be able to make instantaneous and independent admission control decisions for new connection requests. We will show, through simulation results, that this distributed approach can manage the network resources very efficiently, leading to lower bandwidth blocking rates when compared to traditional shortest path admission control. Moreover, its simplicity and distributed implementation make it a very scalable solution for resource management in DiffServ networks.

The remainder of this paper is organized as follows. In Section II we present the overall system design, and discuss various implementation issues of our resource management scheme. In Section III the results of the simulation experiments are presented, while in Section IV some related work is discussed. Finally, Section V concludes our work.

II. System design

In this section we will describe in detail all the network functions that need to be performed, in order to provide end-to-end QoS guarantees in DiffServ-based networks. In particular, we will address the issues of routing, packet forwarding, and both intra- and inter-domain admission control. Throughout this paper we will assume that the Internet consists of several independently administered DiffServ domains, that are interconnected in order to provide global connectivity. One typical example is shown in Fig. 1, where the source and the destination are connected through domains A and B. Each domain consists of a BB, and the core, edge, and leaf routers. The BB will exchange control messages with the edge and leaf routers for the purpose of resource management. From this point on, for simplicity, we will refer to the leaf routers as either leaf or edge routers.

0-7803-7400-2/02/$17.00 © 2002 IEEE

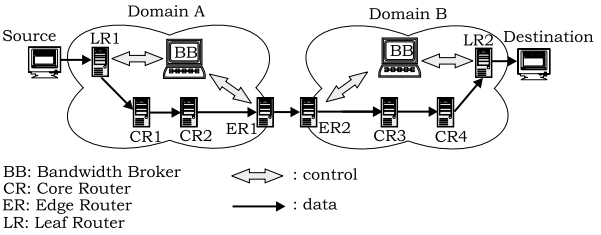

Fig. 1. The DiffServ architecture.

A. Routing

Routing is the process of correctly identifying the next hop at each node (router) so that a packet will be able to reach its final destination. In this work we focus on intra-domain routing, and we assume that all the DiffServ domains use a standard inter-domain routing protocol, such as the Border Gateway Protocol (BGP) [6], to exchange reachability information with their neighbor domains. All the edge routers in each domain will participate in this information exchange.

Providing hard QoS guarantees in any kind of network requires the reservation of enough resources along the path from the source to the destination. Therefore, unlike best-effort traffic routing, a path has to be established in advance between the source and destination nodes, and all the packets should follow the same path. In our resource management scheme we perform static intra-domain routing, where k different paths are pre-computed by the BB to carry the traffic between each pair of edge routers. In the case of link or node failure, the BB will re-compute the paths, and send control messages to the domain routers so that they can update their routing tables accordingly. In the simulation results of Section III we used the well known k-shortest path algorithm [7] for the path selection, where the hop count was used as the link metric.

The static routing approach was adopted for several reasons. First, it facilitates fast packet forwarding which will be further discussed in the following subsection. Second, it follows the general principles of the DiffServ architecture, by completely isolating the core routers from the admission control procedure. Finally, it provides the means for implementing a completely distributed admission control mechanism.

B. Packet forwarding with IPv6

The slowest process in the forwarding path of an IP router is the multi-field classification and routing procedure. When a packet is received at a router, the next hop is decided by looking into several fields on the IP header (e.g. IP addresses, TCP/UDP port numbers, etc.), and then finding the appropriate entry at the local routing table. This operation will be even more complicated for QoS-based flows, since their packets must follow exactly the same path. Clearly, this procedure will become the bottleneck in a multi-Gbps router.

The IPv6 packet header contains a new 20-bit field, which does not exist in the earlier IPv4, called *flow label*. In our scheme we will utilize this field, in order to increase the speed of the forwarding path. Since the routing tables are fixed, the core routers will not need to perform any complex multi-field classification in order to make the routing decision. For each pair of edge routers inside the domain there will be k pre-computed paths connecting them. We may then assign one flow label value to each one of them, and construct new (much smaller) routing tables inside the core of the domain, based only on flow labels. In this way, each packet will be forwarded according to the flow label and DSCP values. During the intra-domain admission control the edge router will select one of the k paths for each new flow, and then mark all the packets that belong to this flow with the corresponding flow label value.

C. Intra-domain admission control

In order to avoid the potential scalability problems that will emerge by having one centralized agent responsible for the resource management inside each domain, our goal is to provide a fully distributed approach where all the edge routers will independently participate in this procedure. The BB in each domain will be responsible for periodically updating the allocation of the resources inside the domain, according to some measurements of the traffic load at the edge routers. When the allocation of resources is completed, all the edge routers will be able to make instantaneous and independent admission control decisions for new connection requests.

The overall system model inside each DiffServ domain may be described as follows:

- There are M edge routers in the domain.
- There are $N = M(M-1)$ different router pairs to which we shall refer to as Source-Destination (SD) pairs.
- For each SD pair there are k pre-computed paths.
- The source of each SD pair i will periodically measure (e.g. every second) the total amount of reserved bandwidth r_i towards the destination. This means that each of the M routers will keep $(M-1)$ different measurements.
- Every T seconds the BB will receive the following information about every SD pair i:
  - The current reserved bandwidth x_i^j on each of the k paths, $j = 1, 2, \ldots, k$. Clearly, $r_i = \sum_{j=1}^{k} x_i^j$.
  - The average reserved bandwidth w_i over the measurement window T.
- After receiving all the information, the BB will run an algorithm which will calculate the new amount of resources that should be allocated to each path. Specifically, each path j of SD pair i will be allocated an amount of bandwidth equal to y_i^j, where $y_i^j \geq x_i^j$.

One possible improvement that takes into consideration the rejected traffic load at the edge routers, is to introduce one more variable v_i which we call *virtual reserved bandwidth*. This variable is normally updated (just like r_i) whenever a new connection is accepted or an existing connection is terminated. However, when a new request is rejected, the edge router will update v_i as if the request was accepted. It will also assign a virtual finish time, which may be based on some measured average call holding time. When those virtual connections are terminated, v_i will be updated accordingly. Then, w_i will represent the average virtual reserved bandwidth measured over the time period T. In this way, when the rejection rate for a particular SD pair is increasing, the BB will try to allocate (if possible) more resources to accommodate the increased traffic load.

Suppose the source node at domain A in Fig. 1 wants to establish a connection to the destination node at domain B. Then the intra-domain admission control at domain A will be performed as follows (with an RSVP-like [8] signaling protocol):

1) The source node will send a PATH message to LR1 which will include the required amount of bandwidth b.
2) LR1 will know that the destination node is in domain B, so it will check whether there are enough resources to carry this flow towards ER1. In particular, for every path j of this SD pair i (i.e. the pair LR1-ER1), it will check whether $x_i^j + b \leq y_i^j$, for $j = 1, 2, \ldots, k$.
3) If none of the paths satisfies this inequality, the request will be rejected. Otherwise the path with the largest amount of residual bandwidth will be selected, and the PATH message will be forwarded to ER1.

To complete the resource management scheme we need to introduce our proposed algorithm which will periodically allocate the resources among the different paths. It is a simple heuristic algorithm which aims

to find a near-optimal solution for the resource allocation. We could not use any of the standard optimal routing algorithms [9] (e.g. the gradient projection method), because these algorithms might produce a solution where for some paths $y_i^j < x_i^j$. This is unacceptable in our case, since the bandwidth x_i^j has already been allocated.

At the initialization phase of the algorithm we determine which SD pairs require an additional amount of bandwidth to satisfy the offered traffic load. We will name those SD pairs *eligible*. For every eligible SD pair i we determine the additional amount of resources b_i that must be allocated. Specifically, $b_i = w_i - r_i$, which means that an SD pair i is eligible if $b_i > 0$. We then add, in a round-robin manner, a very small amount of bandwidth dx to each eligible SD pair. This amount of bandwidth will be added to the path which minimizes a certain cost function. The above procedure will continue until all SD pairs become either non-eligible (i.e $b_i \leq 0$) or *saturated* (i.e. no more resources can be allocated to them). The cost function that we used was the average number of packets in an M/M/1 system [9], given by

$$\text{cost}_j = \sum_{(l,m)} \frac{F_{lm}}{C_{lm} - F_{lm}}$$

where (l, m) are the links that belong to path j, F_{lm} is the flow on link (l, m), and C_{lm} is the capacity of link (l, m). This is a very good cost function for spreading the traffic load among different paths.

After this procedure is terminated there might be some resources left that have not been allocated to any path. These unused resources will be allocated equally among all the non-saturated SD pairs, in a manner similar to the one described above. The complete resource management algorithm is shown in Fig. 2.

```
Procedure resource_management()
{
/* Calculate bi, for all sd pairs i */
        for (i=0; i<N; i++) {
                sd[i].b = sd[i].w - sd[i].r;
                sd[i].saturated=0;
        }
/* Initialize yij, for all paths j and sd pairs i */
        for (i=0; i<N; i++) {
                for (j=0; j<k; j++)
                        y[i][j] = x[i][j];
        }
/* Allocate necessary resources */
        while (exists i with (sd[i].b>0 and sd[i].saturated==0)) {
                for (i=0; i<N; i++) {
                        if (sd[i].b>0) {
                                for (j=0; j<k; j++)
                                        calculate cost[j] for additional bandwidth dx;
                                if (cost[j]==infinity, for all j)
                                        sd[i].saturated=1;
                                else {
                                        select j, where cost[j]=min;
                                        y[i][j] += dx;
                                        sd[i][j].b -= dx;
                                }
                        }
                }
        }
/* Allocate unused resources */
        while (exists i with sd[i].saturated==0) {
                for (i=0; i<N; i++) {
                        if (sd[i].saturated==0) {
                                for (j=0; j<k; j++)
                                        calculate cost[j] for additional bandwidth dx;
                                if (cost[j]==infinity, for all j)
                                        sd[i].saturated=1;
                                else {
                                        select j, where cost[j]=min;
                                        y[i][j] += dx;
                                }
                        }
                }
        }
}
```

Fig. 2. The resource management algorithm.

D. Inter-domain admission control

Having introduced the intra-domain admission control protocol, we move on to illustrate how to perform admission control over multiple domains. We will continue the example from the previous subsection, where the source node at domain A (Fig. 1) wants to establish a connection to the destination node at domain B. We start from the point where the PATH message is sent to the edge router of domain A (ER1).

1) ER1 will forward the PATH message to ER2.
2) ER2 will perform the intra-domain admission control in a way identical to the one described in the previous subsection.
3) If the request is accepted, it will forward the PATH message to the destination node.
4) The destination node will send the RESV message back to the source node.
5) While the RESV message travels back to the source node, all the intermediate edge routers will configure their traffic shapers, policers, and markers to account for the new connection.

Therefore, the response time for a connection request is very small, and it is equal to the round-trip delay between the source and the destination. It is clear that our resource management algorithm is based on dynamic per flow admission control, and not on aggregate resource reservation. This fact, however, does not affect the scalability of this design, since the admission control is performed on the fly while the the PATH message travels from the source to the destination node. The algorithm in Fig. 2 can be easily modified to account for other resource allocation policies as well. For example, two neighbor domains may choose to limit the amount of traffic that is exchanged between them. This may be realized by limiting the capacity of the inter-domain link in the resource management algorithm. Another example is the more permanent connection set up of a Virtual Private Network (VPN). This may also be realized by setting up these connections in advance, and then limiting the capacity of the corresponding links.

Finally, we will summarize below some advantages of our resource management scheme:

- The signaling overhead for connection set up is spread across multiple links, avoiding the congestion of the links around the centralized BB.
- The BB does not need to maintain per flow information about every flow that is currently active inside its domain. This information will be distributed across the edge routers of the domain.
- It offers a scalable solution for dynamic connection set up with very fast response time (equal to the round-trip delay between the source and the destination).
- A temporary failure of the BB will not affect the functionality of the domain, since the admission control is performed at the ingress routers. The failure of an ingress router will have no effect, since no traffic should be routed through this node (i.e. no admission control is required at this node).

III. Simulation results

We simulated our resource management scheme in the MCI internet topology of Fig. 3, which has also been used in [10]. We assumed that all the links have a capacity of 2.4 Gbps, and that all the capacity may be allocated to EF traffic. In a real network, though, the service provider will allocate a portion of the available bandwidth to EF traffic, according to some policy. This topology represents an autonomous DiffServ domain, so we have only simulated the intra-domain resource management scheme. As we have shown in the previous section, inter-domain admission control is performed as a series of independent intra-domain admission decisions.

New requests may arrive at any one of the 19 routers, which means that there were $N = 342$ different SD pairs. The new requests arrive at each SD pair i, according to a Poisson process with mean λ_i.

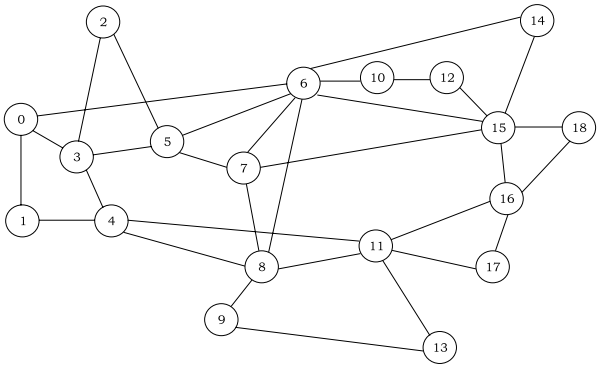

Fig. 3. The simulated network topology.

The arrival rates for the different SD pairs were chosen from a uniformly distributed random variable in the interval [1,λ_{max}]. The value of λ_{max} was properly selected to control the average arrival rate per SD pair. Every new request was either for a voice or video connection with equal probability. The duration of each voice connection was exponentially distributed with mean 5 min, and the required bandwidth was 64 kbps. For video connections the duration was exponentially distributed with mean 20 min, and the required bandwidth was uniformly distributed in the interval [300,2000] kbps. These exponential holding times are quite realistic for voice and video connections, and they have been shown [10] to produce higher blocking rates when compared to long-tail distributions. We simulated 24 hours of real time, and collected the results after an initial warm-up period. The bandwidth blocking rate was used as the perfomance metric, i.e. the percentage of actual bandwidth that was rejected due to insufficient resources inside the domain.

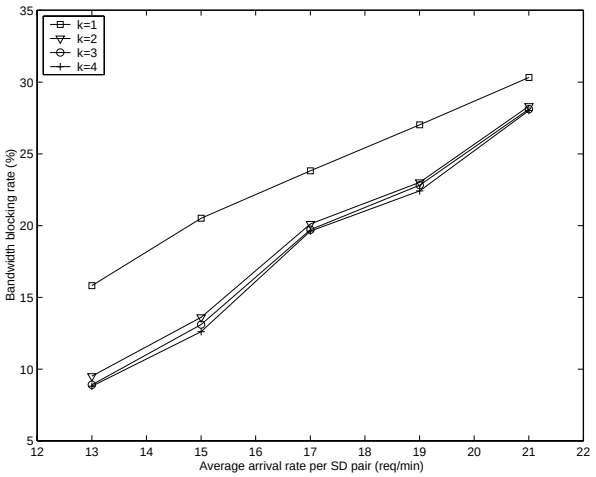

Fig. 4. Bandwidth blocking rate as a function of the average arrival rate for different values of k.

In the first experiment we investigated the impact of the number of paths k that are pre-computed for each SD pair. The results are depicted in Fig. 4, where it is clear that setting up at least two paths per SD pair is very important for efficient resource management. The additional performance gain, however, by adding more paths is very small. This can be explained from the fact that the connectivity of the selected network topology is relatively sparse. In a network with a large average node degree we may gain more in performance by establishing more paths between each SD pair.

We then investigated the impact of the measurement window T on the performance of the resource management scheme. The 'static resource assignment' curve in Fig. 5 corresponds to a system where the resource allocation is manually configured according to some long term traffic measurements. In the simulations, this curve was produced by running the resource management algorithm of Fig. 2 at the start of each experiment (using the known values of λ_i for each SD pair), and then using the resulting $\{y_i^j\}$ values for admission control. Another promising result in Fig. 5 is that the performance of our scheme is not affected much by the length of the measurement window, as long as it is kept at reasonably small values. Therefore, the control overhead imposed by the resource management scheme is almost insignificant.

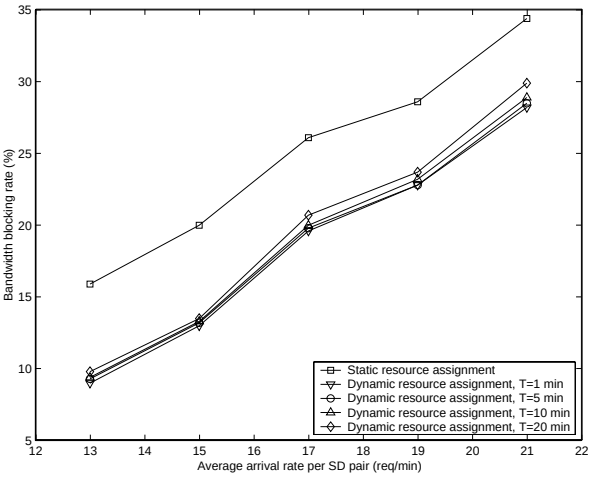

Fig. 5. Bandwidth blocking rate as a function of the average arrival rate for different values of T.

Finally, we compared our scheme with a few other alternatives, and the results are shown in Fig. 6. The 'shortest path routing' curve was produced with typical centralized shortest path routing. Whenever a new request arrived, the BB performed the admission control, based on up to date information about the reservation status on every link. The hop count was used as the metric for finding the shortest path, because in our simulations it had better performance compared to residual bandwidth or average delay. Our scheme clearly outperforms shortest path routing, and its bandwidth blocking rate was more than 10% lower. This result was anticipated, since shortest path routing tends to overload the links that belong to the shortest paths, and further connections have to be established over longer paths, which will consume a lot of the network resources. In the same figure, an obvious disadvantage of static resource allocation is also illustrated. In the 'incorrect static resource assignment' curve, the resources were allocated with the assumption that $\lambda_i = \lambda$ for all SD pairs. In the experiments, though, the λ_i's were uniformly distributed with an average value of λ. In general, static resource allocation results in poor performance, and it should be avoided.

IV. Related work

Most of the research on scalable resource management focuses on aggregate resource reservations between DiffServ domains. The BB is still the centralized agent responsible for resource reservation, but the scalability is achieved by reserving resources for aggregate traffic between different domains. In [11] a two-tier model is introduced, where each domain is assumed to have long-term bilateral agreements with each of its neighbors, specifying the amount of traffic that will be exchanged between them. Whenever there is an increase in the traffic

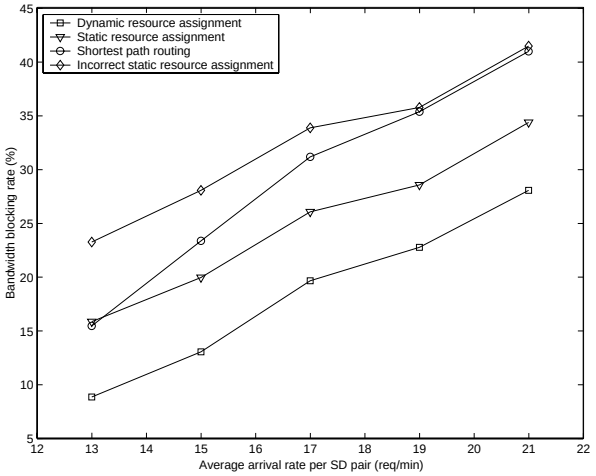

Fig. 6. Bandwidth blocking rate as a function of the average arrival rate for different resource management schemes.

between two domains, the BBs will re-negotiate and make new agreements. In our scheme, though, this is done automatically without the need for an inter-BB signaling procedure. Moreover, the authors did not investigate the efficiency of their intra-domain resource management.

In [12] a Clearing House architecture is proposed, where multiple basic domains are clustered to form a logical domain. In this way, a hierarchical tree is created, where the BB of the logical domain is responsible for resource reservation across the basic domains. The BBs at the basic domains forward only aggregation of inter-domain requests to the BB of the logical domain, thus enhancing the scalability of this architecture. However, the authors only consider the inter-domain resource allocation, and they do not address the problem of resource allocation within a basic domain.

Unlike these two architectures which are more concerned with inter-domain admission control, our scheme tries to optimize the resource management within each domain, which in turn will improve the performance of inter-domain resource management. In addition, we eliminated the centralized BB from both the intra- and inter-domain admission control procedure, which greatly improves the scalability of our design.

Finally, the authors in [13] propose an MPLS-based approach for intra-domain resource reservation, where RSVP signaling is used to make aggregate reservations between each pair of edge routers. Specifically, a sink tree is maintained towards each egress router, and all the traffic follows the paths specified in the sink tree topology. However, as we have shown in our simulation results, having only one path for each pair of edge routers is not very efficient. In addition, the authors do not give any details on how the resources are allocated among the different sink trees or how admission control is actually performed.

V. Conclusions

We have proposed a new distributed resource management scheme for end-to-end QoS guarantees in DiffServ-based networks. Our approach offers a scalable solution to resource management, by completely eliminating the centralized BB from the admission control procedure. The BB in each domain is only responsible for periodically updating the allocation of the resources inside the domain, according to some measurements of the traffic load at the edge routers. Each new request is then forwarded from the source node to the destination node, and admission control is performed instantaneously at the ingress router of each domain. We have shown that this distributed approach can manage the network resources very efficiently, leading to lower bandwidth blocking rates when compared to traditional shortest path admission control.

Acknowledgments

This research is supported in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-01/99).

References

[1] S. Blake, D. Black, M. Carlson, E. Davis, Z. Wang, and W. Weiss, "An architecture for differentiated services," *Internet RFC 2475*, December 1998.

[2] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the differentiated services field (DS field) int the IPv4 and IPv6 headers," *Internet RFC 2474*, December 1998.

[3] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB," *Internet RFC 2598*, June 1999.

[4] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," *Internet RFC 2597*, June 1999.

[5] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," *Internet RFC 2638*, July 1999.

[6] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," *Internet RFC 1771*, March 1995.

[7] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, July 1971.

[8] R. Branden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP) – version 1 functional specification," *Internet RFC 2205*, September 1997.

[9] D. Bertsekas and R. Gallager, *Data Networks, 2nd edition*, Prentice-Hall, 1992.

[10] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *Proceedings International Conference on Network Protocols (ICNP)*, October 1997, pp. 191–202.

[11] A. Terzis, L. Wang, J. Ogawa, and L. Zhang, "A two-tier resource management model for the Internet," in *Proceedings Global Telecommunications Conference (GLOBECOM)*, December 1999, pp. 1779–1791.

[12] C. Chuah, L. Subramanian, R. H. Katz, and A. D. Joseph, "QoS provisioning using a clearing house architecture," in *Proceedings International Workshop on Quality of Service (IWQoS)*, June 2000, pp. 115–124.

[13] T. Li and Y. Rekhter, "A provider architecture for differentiated services and traffic engineering (PASTE)," *Internet RFC 2430*, October 1998.