A scalable intra-domain resource management architecture for DiffServ networks

Haci A. Mantar [a,b], Ibrahim T. Okumus [c], Junseok Hwang [d] and Steve J. Chapin [e]

[a] *Department of Computer Engineering, Gebze Institute of Technology, Gebze, Kocaeli, Turkey*
[b] *Department of Computer Engineering, Harran University, Sanliurfa, Turkey*
Tel.: +90 535 515 1651; E-mail: hamantar@harran.edu.tr
[c] *Department of Electronics and Computer Education, Mugla University, Mugla, Turkey*
[d] *Seoul National University, Seoul, Korea*
[e] *Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA*

Abstract. With the rapid growth of the Internet into a global communication and commercial infrastructure, the need for Quality of Services (QoS) in the Internet becomes more and more important. With a Bandwidth Broker (BB) support in each administrative domain Differentiated Services (DiffServ) is seen as a key technology for achieving QoS guarantees in a scalable, efficient, and deployable manner in the Internet.

This paper presents the design and implementation of a new Bandwidth Broker (BB) model to achieve QoS across a DiffServ domain. Our BB uses centralized network state maintenance and pipe-based intra-domain resource management schemes. The proposed model significantly reduces admission control time and minimizes scalability problems present in prior research while optimizing network resource utilization. The experimental results verify the achievements of our model.

Keywords: Bandwidth Broker, scalability, intra-domain resource management, DiffServ, quantitative QoS

1. Introduction

The diverse service requirements of emerging Internet applications foster the need for Quality of Service (QoS) on IP-networks. The Internet Engineering Task Force (IETF) has defined two QoS models: Integrated Services (IntServ) with Resource ReserVation Protocol (RSVP) and Differentiated Services (DiffServ) [13]. IntServ/RSVP provides QoS guarantees through performing per-flow admission control and resource reservation in all the nodes along the path. While this model can provide end-to-end QoS guarantees, it has a scalability problem in the network core because of per-flow state maintenance and per-flow operation in routers. Because of the scalability problem with Intserv/RSVP, the IETF has proposed DiffServ as an alternative QoS architecture for the network core data forwarding plane, and Bandwidth Broker (BB) [31] for the control plane.

DiffServ requires no per-flow admission control or signaling and, consequently, routers do not maintain any per-flow state or operation. Instead, routers merely implement a small number of classes named Per Hop Behavior (PHB), each of which has particular scheduling and buffering mechanisms. A packet's PHB is identified with the DiffServ field (DSCP) assigned by the ingress router.

To this end, DiffServ is relatively scalable with large network sizes because the number of states in core routers are independent of the network sizes. Thus, DiffServ is considered as the *de facto* QoS standard for the next generation of the Internet. However, unlike the Intserv/RSVP, DiffServ only addresses data forwarding plane functionality, whereas the control plane functions still remain an open issue. Hence, DiffServ alone cannot provide end-to-end

QoS guarantees. In fact, providing end-to-end QoS is not one of the goals of DiffServ architecture [30]. In general, DiffServ has several limitations and open issues. (1) As its name indicates, a PHB defines the forwarding behavior in a single node. Unlike Intserv/RSVP model, there is no QoS commitment for the traffic traversing multiple nodes, or domains; (2) With the exception of Expedited Forwarding (EF) [18], all the PHBs that currently have been defined provide *qualitative* QoS guarantees. Hence, the requirements of real-time applications, which need *quantitative* bounds on specific QoS metrics, cannot be guaranteed even in a single node; (3) There is no admission control mechanism to ensure that the total incoming traffic to a node or domain does not exceed the resources.

From the above issues, it is envisioned that DiffServ needs a control path mechanism to achieve end-to-end QoS guarantees. The Bandwidth Broker (BB) [31] is one of the strongest candidates for this. The BB is a central logical entity responsible for both intra-domain and inter-domain resource management in a DiffServ domain.[1] The goal of the BB is to provide *Intserv-type* end-to-end QoS guarantees in DiffServ-enabled networks.

Upon receiving a reservation request, in general the basic tasks performed by a typical BB [3], [31] are: (1) finding an appropriate QoS path between ingress and egress routers; (2) obtaining up-to-date network QoS state information (which may require communication with all the routers along the path to obtain their QoS state); (3) reserving resources along this path. In this sense, the BB schemes have several appealing aspects such as relieving the core routers from network state maintenance and admission control, optimizing network resource utilization, and alleviating the problems of inconsistent QoS states faced by distributed admission control schemes. However, the BB has its own challenging problems that make the deployment of such a scheme questionable to many researchers:

- It is not clear how a BB can obtain up-to-date network state information under dynamic and non-stochastic network traffic conditions.
- QoS routing itself is a very complex problem [37] and therefore the route computation time is very long [37]. Since a BB needs to find a path for all the requests, the resulting load will be unacceptably high if it performs this for each individual request.
- After finding a path, appropriate resources need to be reserved along the path (adjusting scheduler rate and buffer size). Performing this for each request introduces communication overhead, processing time overhead, and a long admission control time.

In this paper we present a scalable and efficient intra-domain BB resource management model. In particular, this model scalably maintains up-to-date network state information, increases admission probability (and therefore high resource utilization), minimizes admission control time, and provides edge-to-edge quantitative QoS (across its domain) guarantees. Note that the scope of this paper is limited to intra-domain issues. The inter-domain (inter-BB) resource management problems were addressed in our previous work [17,26,32].

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 presents a quantitative QoS service model. In Section 4, we describe the intra-domain resource manager (IDRM) architecture. In Section 5, we provide the IDRM implementation and show the experimental results. A conclusion of this paper and motivation of future work are given in Section 6.

2. Related work

The idea of BB was first proposed by Nichols et al. [31]. Each domain has a BB that is in charge of both intra-domain and inter-domain resource management. The BB model then has been studied by different researchers. Terzis et al. [36] proposed a Two-Tier Resource Allocation Framework. The main idea is that a BB in each domain collects its users' and applications' requests for external resources and sets up a Service Level Agreement (SLA) with its neighboring domains based on aggregated traffic profiles. The BB dynamically re-adjusts the SLA between

[1]Although the BB was originally proposed for DiffServ networks [31], it can also be applied to non-DiffServ networks. Because the BB is independent of the forwarding plane schemes.

domains based on substantial changes in the aggregated traffic demand. In our previous work, we have developed and implemented an inter-domain BB signaling for resource reservation and provisioning (BBRP) [17,26]. We have also used the BB model for inter-domain MPLS paths (LSPs) set up and QoS routing [32,33]. These studies have focused only on the inter-domain BB issues, the intra-domain BB issues have not been addressed.

An application of the BB model for managing VPN across a domain was proposed in [20]. The role of the BB here is to dynamically adjust the VPN between customers of the same domain by taking utilization and accounting issues into account. An interesting intra-domain BB architecture was presented in TEQUILA project [12]. TEQUILA has designed a BB resource management and control architecture for DiffServ-based MPLS networks. In this architecture, the BB has monitoring, traffic engineering, service level specification (SLS) and SLA management, and policy management component. TEQUILA has addressed the important issues of managing a domain resources via a BB. However, this work is still in the initial stage. There is no specific solutions for the defined problems (e.g., resource management and control, traffic engineering, policy control), and no evaluation has been done yet.

Similar approaches were used in [6,25,38]. In [38], a server, which is very similar to the notion of the BB, has been used to manage the LSPs in an MPLS domain. Similar to our approach, the server computes the paths for LSPs on behalf of all the routers, then the path information of the LSPs is downloaded into routers. By taking all the LSP requirements, link attributes, and network topology information into account, the server finds better LSP placement than distributed approach, where every ingress router computes its paths for LSPs separately based on its own information. Another server-based approach named the Routing and Traffic Engineering Server (RATES) [6] was developed by Bell Laboratories for MPLS traffic engineering. RATES uses a centralized database for the network state maintenance, and acts as a link state peer to obtain topology information and link, and node states. In RATES the LSP selection is done using an online routing algorithm, meaning that an LSP set up and its bandwidth is determined based on the current demand or for each individual requests, and thus it cannot be assumed that future requests will arrive. After an LSP is computed, RATES communicates with ingress router by using COPS, and spawn off signaling from ingress router to egress router for LSP setup. The MPLS AutoBandwidth Allocator [5] and the Traffic Engineering Automated Manager (TEAM) [35] also use a server-based centralized paradigm for the LSP management and control in an MPLS domain.

These server-based approaches are in line with our model in the sense that they use a centralized paradigm for QoS routing and LSP setup. However, the QoS routing and LSP setup methods of our model are different. We use pre-computed routing and pre-established LSPs to increase the signaling and state scalability. An LSP, in our model, has a certain bandwidth capacity, and this capacity is dynamically modified with respect to the network conditions. In addition, our model has the following differences. First, we integrate the DiffServ with MPLS. Second, we develop and implement an admission control scheme. Third, we present a new network state maintenance scheme that a BB uses to get the QoS states of the routers in its domain. Fourth, we propose a new class-specific measurement-based scheme to estimate the traffic rate of a link and an LSP. Furthermore, our work provides a quantitative evaluation, which was missing in most of the related studies.

3. Quantitative service model

The DiffServ model provides two type of service guarantees, *qualitative* and *quantitative*. Qualitative services aim to provide per-hop per-class relative service guarantees [16]. There are no worst-case bounds for a given service metric set (e.g., packet loss ratio, queuing delay), meaning that these metrics can vary with temporal network conditions. Instead, each node guarantees that the service variation among different classes is locally maintained. Quantitative services, on the other hand, aim to provide hard guarantees on QoS metrics. In this paradigm, service metrics have worst-case bounds that are independent of the network utilization. Depending on the application, these service guarantees can be either *deterministic* or *statistical*.

Most of the real-time and mission critical applications require quantifiable QoS guarantees. However, with the exception of Expedited Forwarding (EF) [18], all PHBs that currently have been defined (by IETF) provide qualitative QoS guarantees, known as Assured Forwarding (AF) [16]. Although EF provides quantitative guarantees, it is envisioned that its applications will be very limited, because it uses peak-rate-based resource reservation and therefore results in poor resource utilization [11,17,29]. Thus, there is an obvious need for new PHBs that can provide quantitative service guarantees (statistically) while increasing resource utilization.

3.1. Quantitative PHB

We define a set of PHBs that provide statistical QoS guarantees. Each PHB, i, is associated with an upper delay bound d_i and/or upper loss ratio bound l_i (e.g., $d_i < 3ms$, $l_i < 0.01\%$) and a certain percentage of link capacity C_i. Each PHB can only use its share of link capacity. The unused capacity can be used by best-effort traffic or qualitative services so there is no waste incurred.

Bounds d_i and l_i are independent of temporal traffic characteristics (e.g., the utilization level of C_i). It is assumed that d_i and l_i are pre-determined at the network configuration stage, which is done in relatively long time intervals (e.g., days, weeks) and downloaded into routers [12,26,30].

3.1.1. QoS Enforcement and provisioning

We use a node enforcement and provisioning algorithm to adjust the scheduler rate (output rate) and buffer size. The algorithm dynamically adjusts the output traffic rate and buffer size of a PHB according to the dynamic network conditions to meet the pre-defined d_i and l_i constraints [9,23,29]. The output traffic rate R_s is simply defined as: $R_s \geqslant buffer\_size/d.$

Under this premise, each router provides the desired QoS regardless of the utilization rate as long as the aggregate traffic rate does not exceed the link rate for that particular PHB. The incoming traffic rate is controlled by a strict admission control mechanism (described in the next section). Note that the worst-case bounds for a PHB are for heavily-loaded conditions; for lightly-loaded conditions delay and loss ratio rate are, of course, less than d_i and l_i.

An important point here is that a router do not perform per-flow or per-class (aggregate) admission control. Hence, there is no signaling and state overhead in core routers. The admission control is performed by a third-party server (BB) on behalf of all the routers.

3.2. Edge-to-edge QoS (PDB)

The next step toward end-to-end QoS is to define QoS behavior across a domain (from the ingress router to the egress router). This is called Per Domain Behavior (PDB) [30] by IETF. A PDB defines the QoS behaviors that identifiable packets will receive as they cross a domain. PDBs, as QoS building blocks, are intended to be useful tools for managing a domain resources. This level of abstraction makes it easier to compose cross-domain services as well as to hide details of a network's internals while exposing information sufficient to enable QoS.

Since the worst-case properties of a PHB (d_i and l_i) are the same in all the nodes within a domain, a PDB for IR-ER (ingress router, egress router) pair and PHB i can be simply computed as: $PDB_{IR,ER,i} = <IR, ER, D_i, L_i>$. Here, $D_i = \sum_{j=1}^{N} d_{ij}$, $L_i = 1 - \prod_{j=1}^{N}(1 - l_{ij})$, where j represents a node, N represents the number of nodes along the path and D_i and L_i are the delay and loss ratio rate of PDB. In this work, a PDB also represents the QoS associated with an intra-domain pipe described in the next section.

An important feature of this service model is that the NP-complete QoS routing problem [37] can be simplified. Since a PHB has the same worst-case constraints, which are not changed with the network utilization, in all the nodes, the routing problem can be simplified to the number of hop counts and bandwidth constraints.

4. Intra-domain resource manager architecture

4.1. Overview

As illustrated in Fig. 1, the intra-domain resource management architecture relies on a *centralized* server called intra-domain resource manager (IDRM). The IDRM maintains *detailed* network QoS information (e.g., PHB-based

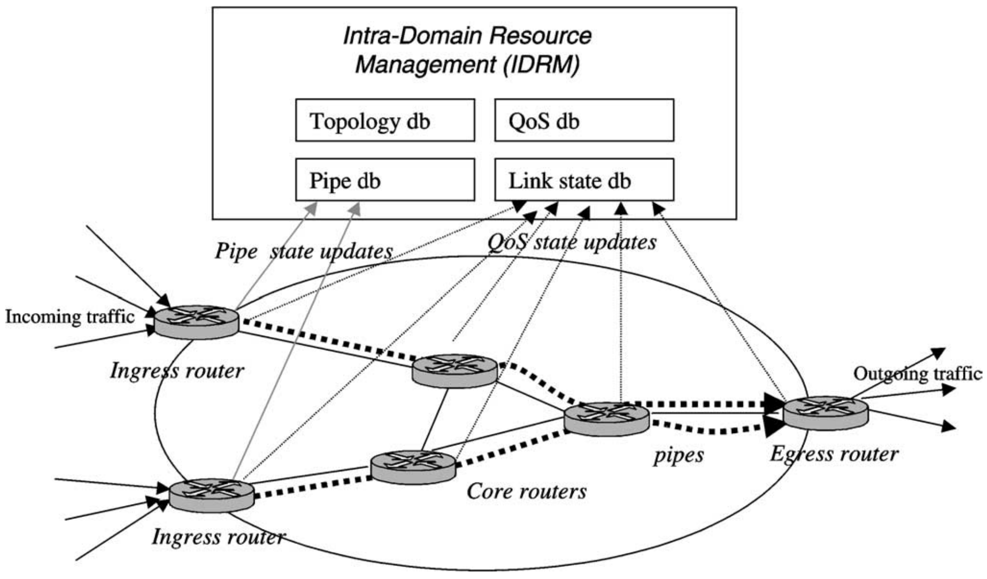


Fig. 1. IDRM functional architecture.

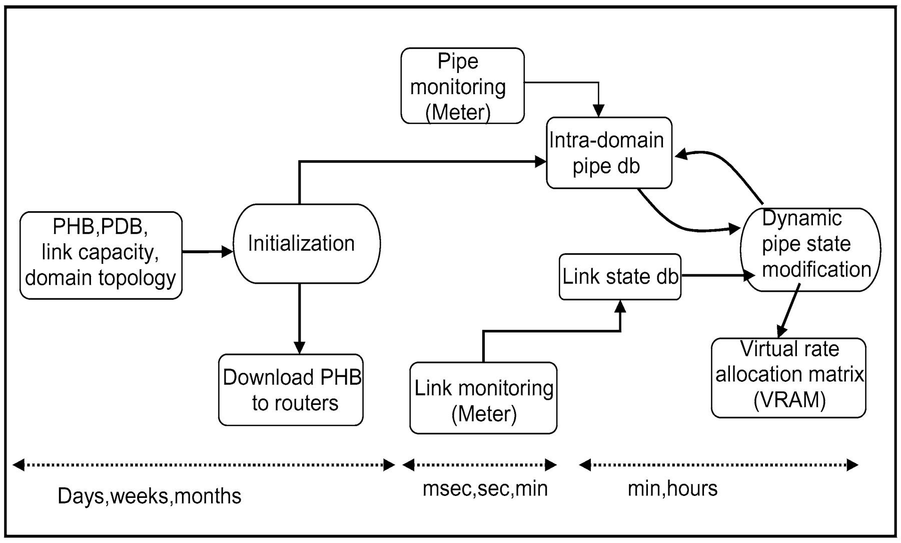


Fig. 2. Intra-domain pipe setup and modification steps.

available bandwidth in each link), performs QoS routing, and presents the status of the network (domain) to the BB in an *abstracted* fashion. By abstracting complex domain structure and performing computationally intensive tasks (e.g., QoS routing) in advance, the IDRM simplifies the online BB tasks for intra-domain admission control.

The IDRM has several relational database components to maintain and manage network state information.

QoS Information Database contains the QoS parameters associated with each PHB (*PHB: <delay bound, loss bound>*). This can be considered as static information in the sense that it is updated only at the network configuration time (Fig. 2).

The Domain Topology Database contains the connection map of routers in the domain. This database is static.

The Link State Database is structurally very similar to the OSPF routing information base (RIB). It keeps class-based (PHB-based) QoS state information of all the routers in the form of: <*interface IP, PHB, total capacity, current traffic rate, cost*>. This database is dynamically updated based on network conditions.

The Pipe Database stores the states of the pipes, established between each ingress router (IR) and egress router (ER) pair, one for each PHB (<*IR-ER, PHB, current traffic rate, pipe size, cost*>). This database is the abstraction of complex domain topology in terms of network state information, and it is dynamically updated with network conditions.

The first step in managing network resources is to obtain accurate link state information. For this, the IDRM uses two different paradigms. First, by having domain topology and the links' QoS state information such as supported PHBs and their associated link capacity in its database, the IDRM centrally updates the link state database as resources are assigned to or released from a reservation without interacting with the links to get their actual QoS state.

Second, the IDRM dynamically obtains links' QoS information from routers. In this paradigm, each router within the domain sends its link state QoS information (for each interface-PHB pair) directly to the IDRM (Fig. 1) rather than flooding to all other routers as done in traditional link state protocol [6,28]. The IDRM is the *only entity* that has complete knowledge of domain topology and each link's QoS state information. Unlike link state protocol, where each router needs to maintain QoS information about all the routers, here a router needs to maintain only its own links' QoS information (i.e., available and reserved bandwidth of PHBs in each interface). Thus, the routers' communication, computation and storage overhead are substantially reduced.

The next and most challenging step is to find a path that has sufficient resources upon receiving a QoS request. As described before, performing QoS routing for each individual reservation results in scalability problem [10,25]. This problem can be even worse for centralized schemes because all the request need to be handled by a single server. To reduce the scalability problem, we propose a *pre-established pipe* model in which the admission control and network state maintenance are performed via pipes.

Given the domain topology and QoS (PHB) supported by each node at the network configuration stage [3,12, 30], the IDRM pre-establishes a pipe between each possible IR-ER pair, one for each PHB (<*IR, ER, PHB*>). A pipe can be an MPLS tunnel (LSP), which is called *trunk* when it is used for DiffServ classes, or it can rely on traditional IP-forwarding where the packets associated with a particular destination address prefix (IP/X, where $X < 32$) are forwarded to the same egress router (similar to aggregated RSVP [7]). All the packets/flows that have the same ingress-egress routers and PHB are aggregated into the same pipe.

Unlike the previous pipe schemes [6,10,22], in our model the pipes do not have an *explicit* bandwidth reservation in the forwarding plane (in the routers). Instead, the IDRM maintains the reserved and available bandwidth of each pipe in its pipe database. Since the admission control is performed (by BB) based on the information maintained by IDRM, there is no need to reserve resources for a pipe in the forwarding path. The IDRM can dynamically resize the pipes with network conditions without reflecting this change to the forwarding path. To have pre-determined QoS (PHB) guarantees, the routers dynamically adjust their buffer size and scheduling rate according to the aggregated traffic rate as described in previous section.

The proposed pre-established pipe model significantly increases the BB admission control scalability. When a BB receives a reservation request, it just determines the ingress and egress router, and then checks the associated pipe resource availability in its pipe database. If there is enough available resource, it accepts the request, otherwise it rejects. No need to perform QoS routing to find a path, no need to check the links' resources along the path, no communication with core routers. Furthermore, the number of reservation states that routers maintain for forwarding is pipe-based (independent of the number of individual reservations).

In this work we assume that each domain has only one IDRM. However, it is better to use multiple IDRMs for large-scale domains due to the signaling and state scalability problems. A domain can be divided into multiple areas each of which has its own IDRM (similar to the notion of areas in OSPF [28]). In this scheme, each IDRM can manage its own area and communicate with other IDRMs for cross-area connectivity. For inter-IDRM communication, we can use the Simple Inter-domain BB Signaling (SIBBS) [3,17] or BB Reservation and Provisioning protocol (BBRP) [26] with minor modifications. Although we developed SIBBS and BBRP for inter-domain BB communication, they can easily be adapted to inter-IDRM communication, because a multi-area domain (each area has one IDRM) is very similar to a multi-domain network (each domain has one BB). In this work we focus only on a single IDRM case, for multi-IDRM case the reader are referred to [3,17,26].

In general, the IDRM has two operation phases:

1. Determine the size of each pipe and store this information in the "pipe database", and readjust the pipes' size when a significant change occurs in their utilization level.
2. Dynamically monitor/estimate links' and pipes' resources (e.g., available bandwidth), and update the link and pipe state databases when needed.

In the next subsections we describe these phases in detail.

4.2. *Dynamic pipe resizing*

This section describes how the IDRM resizes pipes under dynamic network conditions. As shown in Fig. 2, the IDRM dynamically checks each pipe's utilization in the pipe database. When the utilization of any pipe exceeds its utilization target (e.g., 90% of its size), it invokes the pipe resizing process. This is done in two stages: (1) obtaining the QoS state information of all the links along the path (pipe) by accessing the link state database; (2) determining an appropriate pipe size.

Performing these tasks within a short time interval may result in a scalability problem due to the high frequency of database access and computationally intensive pipe resizing process. Thus, as shown in Fig. 2, the resizing process should be done in medium time intervals [12,25,30]. The objective of our pipe resizing scheme is to minimize the frequency of resizing process while achieving high resource utilization.

Consider the simple topology shown in Fig. 3a, where the pipes are mutually disjoint, meaning that a link's resources for a particular PHB are used only by a single pipe. By assuming that a PHB resources in a link can be used only by the traffic of the same PHB or best-effort, the IDRM sets a pipe size to the possible maximum value, that is: $P_{size} = \min(C_1, C_2 \ldots C_N)$, where C_i represents the capacity of a particular PHB in the link L_i along the path. In this scenario, since a link's capacity is used by only one pipe, there is no need to resize a pipe. It can only be resized when the PHBs' capacity is altered at the network configuration time. The IDRM simply updates the pipe database as the resources are assigned to or released from a reservation.

In practice, however, a link capacity might be shared by multiple pipes (Fig. 3b). This makes pre-established pipe schemes difficult in the sense that a pipe's share of a link capacity should be predicted by taking dynamic and non-deterministic future traffic conditions. A network load can fluctuate dynamically, resulting a previously under-loaded pipe to become overloaded over time.

Consider Fig. 3b where the capacity of link $l1$ is shared by pipes $P1$ and $P2$. Setting the pipe size to the maximum value for a long time (network configuration time interval), as done for disjoint pipes, may cause inefficient resource utilization. To clarify the problem, suppose that each link in Fig. 3b has a capacity of 10 Mbps. The size of $P1$ and $P2$ is set to 5 Mbps based on the capacity of $l1$, which is the bottleneck link in this scenario (each pipe has equal share of $l1$). Let say that after some time the traffic rate in $P1$ and $P2$ becomes 3 Mbps and 5 Mbps, respectively. Now, if the BB receives a request that corresponds to $P2$, it will reject the request because of insufficient available resources in $P2$, even though there are available resources along the path. To minimize this problem, the IDRM dynamically resizes the pipes with the following algorithm (Fig. 4).

Steps 1–4 are performed at the network configuration time. Basically, IDRM establishes pipes for each possible ingress-egress pair (one for each PHB). The pipes that compete for the same link's capacity are assigned by a certain share of link capacity, which is called "virtual capacity". The virtual capacity can be determined based on historical data, initial values, or it can be an equal amount for each pipe. Note that since virtual capacity and, in turn, pipe size is modified dynamically based on the traffic load change, there is no need to predict the future traffic demand, which is a difficult problem [34].

In the dynamic traffic-driven operation, the IDRM dynamically re-sizes the pipes with respect to the traffic demand change in the network. When the traffic rate in any of the pipes exceeds utilization target, the pipe resizing process is invoked (steps 6, 7). In this stage, the IDRM distributes the unused links' capacity among the pipes that use the links. For example, suppose m pipes, $P1, P2 \dots Pm$ share a bottleneck link k with capacity c_k. Let P_i^{cl} denote the current traffic load of P_i that uses v_k. The virtual capacity vc_{ik} of pipe P_i on link k is computed as

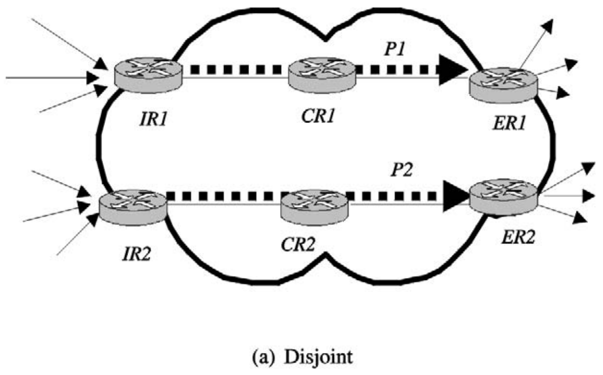
$$vc_{ik} = c_k \frac{P_i^{cl}}{\sum_j^m P_j^{cl}}. \qquad (1)$$

The vc is computed for each link with Eq. (1) and stored in a virtual rate allocation matrix (VRAM) where a column represents a link and the field in the column represents the vc of the link for a particular pipe (Table 1). Each pipe has its own share of a link as if it is the only one that uses that link. After the VRAM is built, the pipe size is determined with Eq. (2).

$$P_i^{size} = \min(vc_{i1}, vc_{i2} \dots vc_{in}). \qquad (2)$$

Table 1
The VRAM matrix of Fig. 4.

pipes	L1	L2	L3	L4	L5
P1	6.25	10	10	X	X
P2	3.75	X	X	10	X



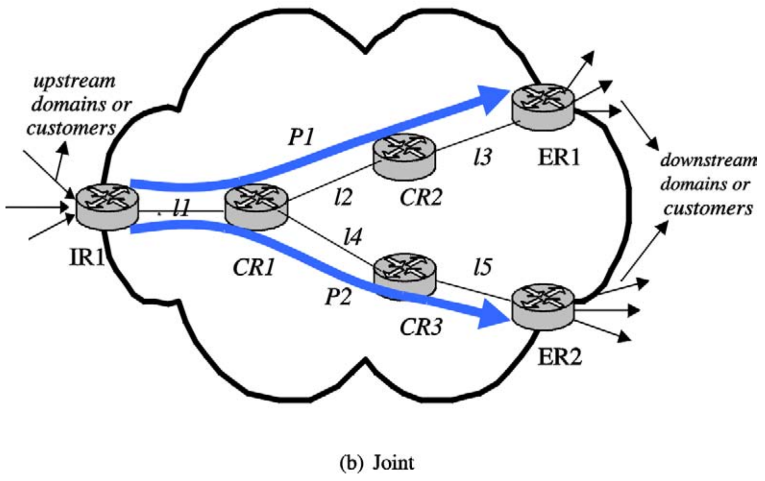

upstream domains or customers
P1
l1
l2
l3
l4
l5
CR2
ER1
IR1
CR1
P2
CR3
ER2
downstream domains or customers
(b) Joint

Fig. 3. Disjoint and joint pipe examples.

Table 1 shows the VRAM of Fig. 3b for the above scenario in which the traffic rates in $P1$ and $P2$ become 5 Mbps and 3 Mbps, respectively (the sizes of $P1$ and $P2$ are 5 Mbps). Since the utilization of $P1$ reaches its maximum capacity, the pipe resizing process is invoked, and the size of $P1$ and $P2$ will be changed as follows

$$P1^{size} = \min(6.25, 10, 10) = 6.25,$$

$$P2^{size} = \min(3.75, 10, 10) = 3.75.$$

This scheme has several important features that significantly reduce the BB processing load: first, the resizing process is rarely invoked by individual requests; rather, it is invoked by the aggregated traffic rate change in pipes. Due to the complexity of the process (e.g., computing the vc for each pipe on each link), the resizing process may take long time. However, this process is expected to be invoked within a long time period (e.g., minutes, hours). As long as the aggregate traffic rate in a pipe does not exceed the pipe size, the re-sizing process is not invoked. Second, similar to the *trunks* scheme proposed in [22], all the traffic that has the same ingress-egress pair is aggregated into a single pipe, regardless of its source and final destination (e.g., a pipe may carry the traffic originated from multiple source domains and destined to multiple destination domains). This coarse level of aggregation damps short-live traffic fluctuations, consequently reducing the frequency of the resizing process.

A disadvantage of this scheme is that in heavily-loaded network conditions the pipe resizing frequency will be high, making the BB the bottleneck of the system as those presented in [6] and [31]. To minimize this problem, we use a periodic update scheme in our implementation. Under-heavily loaded network conditions, the IDRM invokes the pipe resizing scheme periodically (e.g., every five minute). When the pipe resizing frequency exceeds a pre-defined threshold, the IDRM switches from on demand based update to the periodic update.

Note that in this work we assume that there is only one pipe for each <*ingress, egress, PHB (PDB)*> triple as done in [22]. To further increase the utilization, multi-pipe may need to be established for each <*ingress, egress, PHB (PDB)*> as proposed in [10]. Although, for simplicity, we do not consider this situation in this paper, the IDRM has flexibility to support the multi-pipe paradigm.

```
Input:
Domain topology, PHBs and associated attributes, PDBs for IRER pair

Initialization:
 1. configure routers with given, PHBs
 2. establish a pipe for each <IR,ER,PHB>
 3. for the links shared by multiple pipes, determine the capacity share for each of those pipes
 4. set each pipe size to the maximum possible value
 5. store this information in "intra-domain pipe db"
    /* P:<IR,ER,PDB,size, available_capacity> */

Dynamic traffic-driven operation:
 6. dynamically check the traffic rate in each pipe
 7. if the current traffic rate in a pipe or a new incoming request's BW plus current traffic rate
    reaches to the pipe size
 8. then compute virtual link capacity for each pipe
    /* the shared link capacity is divided to virtual links one for each pipe which use the link */
 9. set the pipe size based on updated link status
10. update intra-domain pipe db with new pipe size
11. end then
12. go to 5
```

Fig. 4. The IDRM pipe management algorithm.

4.3. Network state maintenance

This section describes the methods used to maintain the "link state database" and "pipe database". In particular, it addresses two key issues: how to estimate/determine the traffic rate in a link and pipe, and how to update these databases. We present two possible solutions, parameter-based update and measurement-based update.

4.3.1. Parameter-based update

In this paradigm a BB updates the link state and pipe databases based on the requests' traffic descriptor parameters rather than interacting with routers to get the actual traffic rate. Upon granting a new reservation or releasing an existing one, the BB updates the corresponding pipe resources accordingly. The parameter-based scheme has two common rate estimation methods: peak rate and effective bandwidth.

In the peak-rate method, the traffic rate is updated with a request's peak rate, independent of whether the source transmits continuously at peak rate or not. This method is very simple because only the knowledge of peak rate of reservation is required. However, it results in poor resource utilization. To increase resource utilization, several studies such as [8,21] have proposed effective bandwidth methods. In effective bandwidth methods, a source reserves a particular bandwidth between its mean rate and peak rate by tolerating limited delay and loss. While the effective bandwidth methods can increase resource utilization, it has a computational complexity problem when the sources are heterogeneous (the sources' traffic descriptor parameters are different). In practice, different applications may have different traffic descriptor parameters. Even the requests of the same applications may have different traffic characteristics. This heterogeneity of applications make the implementation of effective bandwidth very difficult [8,24,27].

4.3.2. Measurement-based direct update

We propose a new measurement-based update and rate estimation paradigm. In this paradigm, the IDRM gets the link and pipe QoS states directly from routers. As depicted in Fig. 1, each router dynamically measures the aggregated traffic rates of its links and sends to the IDRM.

Unlike the traditional link state protocol-based update schemes, where a router floods the states of its links to all the routers within the domain, in the proposed scheme a router is configured to send its QoS state directly to the IDRM only. Hence, the IDRM is the only entity that keeps track of the QoS states of all the routers within a domain. A router needs to maintain only its own links' QoS state, no information of other routers is needed.

Compared to the link state protocol update, use of an IDRM significantly increases the scalability. Suppose that a domain has n routers and each router has m interfaces ($m < n$). In the link state protocol based update, since a router propagates its state to all other routers, the number of messages in the network are $n * (n - 1)$. In the IDRM-based scheme this number is n, because a router sends its state only to IDRM. So that the IDRM reduces the communication overhead from $O(n^2)$ to $O(n)$. Another issue that needs to be considered is the storage overhead. While in the link state protocol each router keeps $n * m$ states, in the proposed paradigm only IDRM keeps $n * m$ states–routers keeps only m states. Furthermore, in the link state protocol, when a router receives an update message from other routers, it needs to update its routing information base (RIB). This causes an additional processing load. In the IDRM scheme routers does not have this problem.

4.3.2.1. Class-based traffic rate estimation. In the measurement-based schemes such as [19,21], the traffic samples are collected at regular small time intervals called sampling period, S, in the duration of a measurement window W. Jamin et al. [19] has proposed a maximum sampling rate-based approach, where at the end of each W the highest sampling rate is used as the load estimation for next W, for IntServ controlled load services. Although this model was adopted by many measurement-based schemes, it may not be feasible for DiffServ guaranteed services. This is because it does not explicitly take the delay and loss constraints into account. As illustrated in Fig. 5, the rate estimation over the same traffic samples can vary with respect to QoS constraints.

We propose a class-based measurement paradigm. Our objective is: Given a class' QoS constraints (d_i and l_i), estimate the traffic rate of each class on each link based on the real-time measurement statistics. With the assumption that a large number of reservations are being aggregated into the same queue (the class) and therefore share the same resources, we model the traffic envelope of a link as Gaussian (Normal) distribution under the

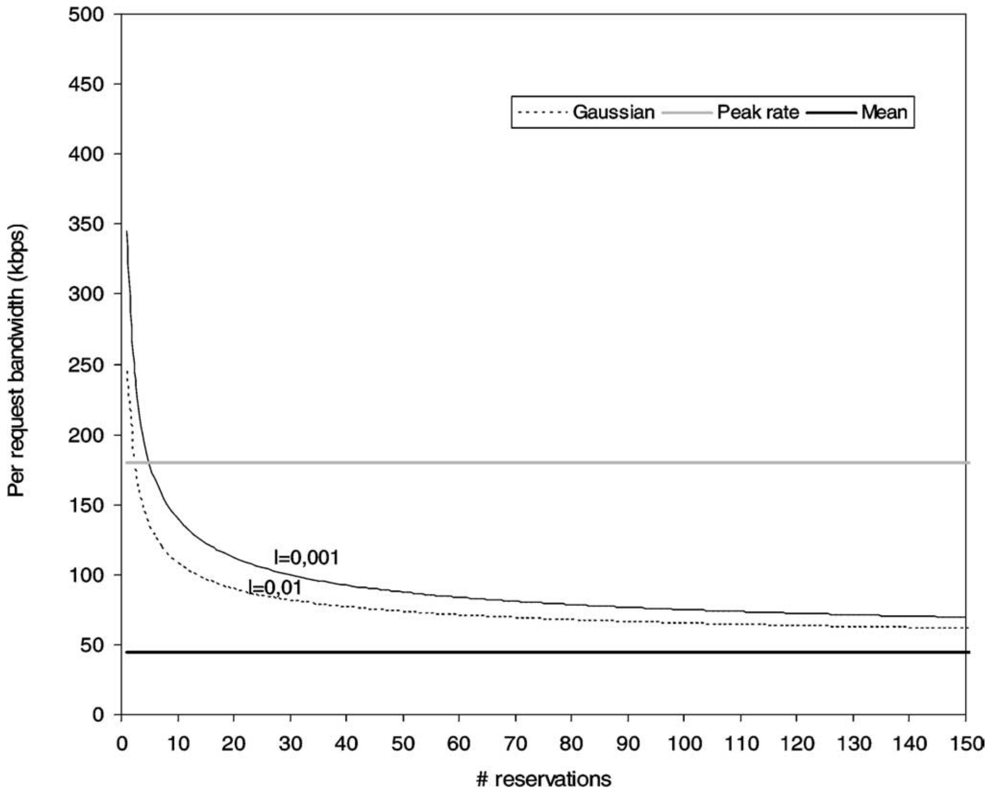


Fig. 5. The bandwidth needed for a request with respect to tolerable loss ratio and the number of aggregated reservations.

conditions of the Central Limit Theorem (CLT) [24,34]. The CLT states that the aggregation rate approaches a Gaussian distribution when the number of aggregated reservations is large [24,34]. There are several advantages of using this model.

- From the given definition, the Gaussian distribution becomes a more realistic model for the DiffServ network to estimate the traffic rate of a class, or link utilization because of the coarse granularity of the aggregation. The individual reservations' traffic rate fluctuations are smoothed due to aggregation.
- The aggregate traffic can simply be characterized by mean and cumulative variance alone.
- The rate can be estimated based on pre-defined QoS metrics [27]. In other words, class-based rate estimation is possible. For example, a class's traffic rate can be estimated based on its tolerable loss ratio (Fig. 5).

Let denote m_i as the mean traffic rate of sampling period (S) i, N as the number of samples in a window W, m as the mean traffic rate of a W ($m = (1/N)\sum_{i=1}^{N} m_i$), σ^2 as the average variances of N samples ($\sigma^2 = (1/(N-1))\sum_{i=1}^{N}(m_i - m)^2$), R as the predicted traffic rate, and X as the instantaneous traffic rate. To meet the pre-defined l and d, the following equation must be satisfied:

$$Pr(X > R) < l \tag{3}$$

(3) can be solved with the well-known normalized Gaussian approximation [34].

$$Q\left(\frac{R-m}{\sigma}\right) \leqslant l \tag{4}$$

where

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp^{-y^2/2} \, \mathrm{d}y = \frac{1}{2} erfc\left(\frac{x}{\sqrt{2}}\right),$$

in order not to exceed l

$$R = m + Q^{-1}(l)\sigma. \tag{5}$$

Equation (5) computes a link's traffic rate (for a class) with respect to l. The multiplier $Q^{-1}(l)$ controls the estimation rate to meet the constraint l. As seen, the estimated traffic rate of a class varies with respect to its l value.

Since, the delay is one of the constraints, the value of W is set based on the class's delay bound, d. That is, $W = d - \triangle d$; $\triangle$ is a cushion to prevent delay violation that may come from the scheduler.

Estimating the traffic rate of a pipe is the same as estimating a link rate. The only difference is that in pipe's rate estimation, the samples are collected from the traffic that belongs to the pipe, while in link rate estimation the samples consists of all the traffic of a particular class regardless of their destination (egress router). Also a pipe rate is measured only by the corresponding ingress router. The reason of measuring at ingress router is that the traffic may get smoother as it traverses towards the egress router due to buffering effect, and therefore the rate measured in the egress router might be less than the one in the ingress router. Thus, relying on egress router measurement for a pipe may not give us accurate results.

4.3.2.2. Update conditions. An update condition determines when a node should update the link state and pipe database (send its link and pipe state to the IDRM). The most accurate approach would be to update at the lowest possible time intervals such as at the end of every window, W. However, this is very expensive in terms of signaling and processing overhead.

To minimize the above problem, we use a threshold-based triggering scheme, inspired by [15]. A new update is sent when the change in available bandwidth since the last update exceeds the predetermined threshold (th). A node sends the update messages to the link state database when the current traffic rate R^c is significantly different from the previously advertised value of traffic rate R^p. That is

$$\frac{\|R^c - R^p\|}{R^c} \geqslant th. \tag{6}$$

The value of th has a significant effect on the update frequency. While the smaller th results in more accurate network state information, it causes a scalability problem due to high communication and processing overhead. On the other hand, the larger th may increase scalability, but it results in a more inaccurate network state information (and hence low resource utilization). Thus, the key issue here is to choose an appropriate th value.

In our model, the value of th varies with respect to the utilization level. That is, when the link is lightly loaded, th is high, meaning that the frequency of the update messages is low. An inaccurate link state will not have much effect on the link utilization (when the load is low). When the link is heavily loaded, th is low, meaning that the frequency of update messages is high. This is because in a heavily loaded case, the more accurate link state may result in higher utilization. Based on these conditions, we formulate th as

$$th = \lambda \frac{C - R^c}{C}, \tag{7}$$

where C is the link capacity for that particular class and λ is a constant.

The format of a link update message is as follows: *<interface IP, PHB ID, available bandwidth, current traffic rate, cost>*. The above conditions are the same for pipe database updates. The format of a pipe update message is: *<Pipe ID, current traffic rate>*.

5. Implementation results

We implemented the IDRM in conjunction with our inter-domain BB scheme described in [17,26,27]. In general, the implementation consists of the following main components.

Network Monitoring Tool (NMT) estimates online link and pipe utilization with the measurement-based method described in the previous section and updates the corresponding databases (link and pipe) in IDRM. The NMT has three parameters: sampling period S (this, to some extent, defines the tolerable delay/buffer-size of a node), window size W, and loss control parameter η ($\eta = Q^{-1}(l)$). These parameters are class-specific and predetermined at the configuration time. The NMT is located in both edge/border and core routers. While the core routers estimate only the aggregate traffic rate in their interfaces, the ingress and egress routers estimate the traffic rate of pipes in addition to their interfaces' rate.

Relational Databases: The IDRM uses relational databases to maintain pipe and link states. Currently, each router sends its state directly to the link and pipe databases (Fig. 1). The communication between routers and the databases is handled via a long TCP session.

Forwarding path: We used iproute2 [1], which has advanced traffic control functionalities, for traffic conditioning and queuing. In all the experiments we applied class-based queuing (CBQ) [14]. The traffic that exceeds its profile is simply dropped.

For the first part of the experiments, we used the topology shown in Fig. 6. Each ingress router ($IR1$ and $IR2$) has 10 source nodes that request reservations to any node in the destination domain. Each link in the domain has a 10 Mbps capacity for priority traffic, and all the links are unidirectional. There are two intra-domain pipes, $P1$ and $P2$, which span over the paths $L1 - L3$ and $L2 - L3$, respectively. The $P1$ and $P2$ traffic rate estimations were performed at the corresponding ingress routers $IR1$ and $IR2$, respectively.

The source nodes connected to IR1 and IR2 were configured to make reservation requests. A request was characterized with minimum, mean and maximum traffic rates. The mean rate varied according to a rate profile created based on the traced data from a real network [2]. We set the minimum and maximum traffic rates to 70% and 140% of mean rate, respectively. Once a reservation was granted, the host immediately triggered the traffic generator [4] to send traffic. Each node had multiple simultaneous reservations between 5 and 10. The average number of simultaneous reservations for each source network was 35. The duration of a reservation was exponentially distributed with a mean of one minute.

In the first set of experiments, we explored the effectiveness and the accuracy of the our class-based measurement paradigm for link state maintenance and update. Figure 7 shows the effect of W in traffic rate estimation and the

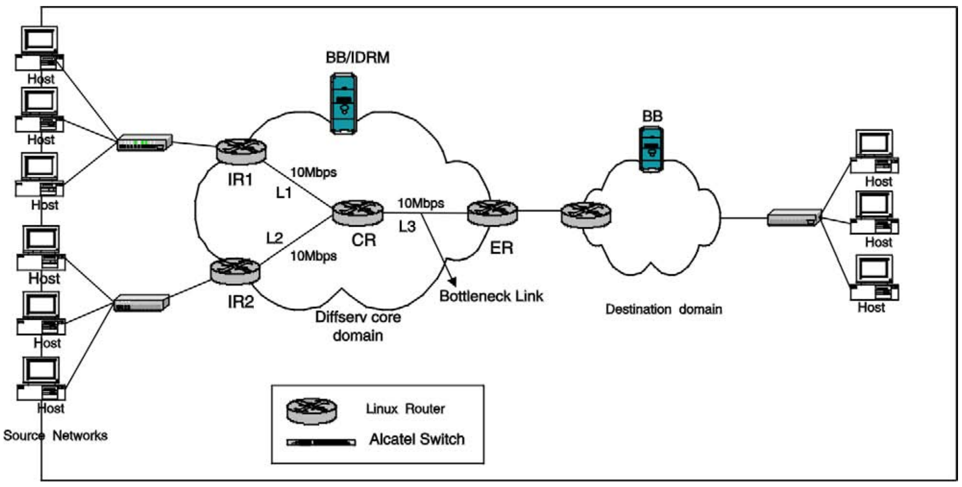


Fig. 6. Experimental network topology.

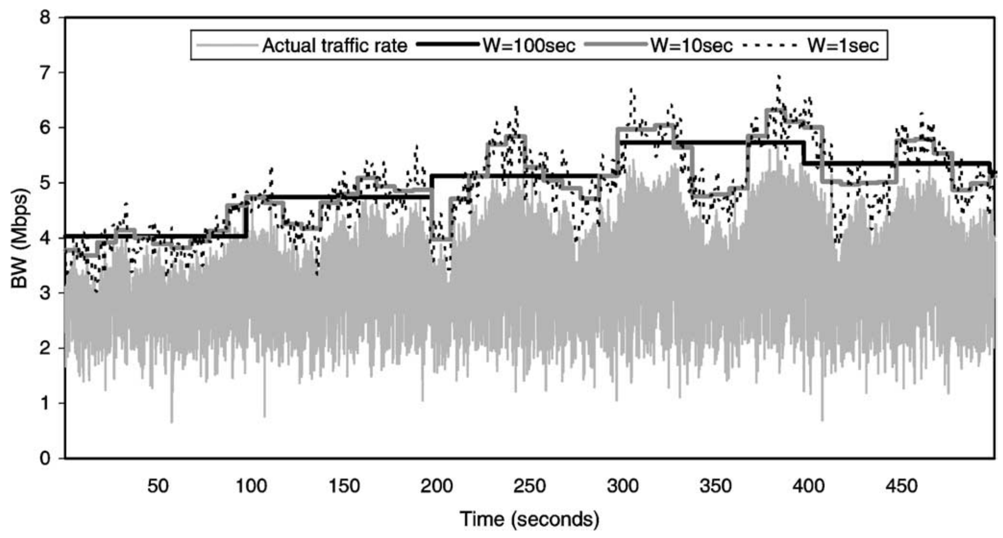


Fig. 7. The effect of W in a link traffic rate estimation.

Table 2
The effect of variable window size on the number of accepted reservation requests under different network loads

window size	$\rho = 1$	$\rho = 0.9$	$\rho = 0.8$	$\rho = 0.7$
10	563	661	691	700
50	461	641	687	700
100	359	611	681	700

scalability of link database update. In this experiment we set $\eta = 2.5$, which corresponds to a 0.621% loss ratio and $S = 1$ millisecond. The measurements were performed for $W = 1$, $W = 10$ and $W = 100$ seconds under a heavy load network condition. As depicted in the figure, the accuracy of rate estimation varies with W. While the 1-second window has the most accurate rate estimation, it is sensitive to small time-scale traffic rate fluctuations and therefore results in signaling scalability problem (high frequency of update messages). On the other hand, the 100-second window increases the scalability, but it has the least accurate rate estimation. As a consequence, it degrades QoS performance and causes poor resource utilization.

The Table 2 shows the trade-off between window size and the number of admitted reservation requests during a 20 minute period. (ρ represents the utilization rate of a pipe.) From these results, it is observed that, under lightly-loaded network conditions, setting W to a smaller value does not significantly increase the link utilization compared to the update overhead that it causes. For example, while for $\rho = 0.8$ the number of admitted requests for 10-second and 100-second windows are very close to each other, the update frequency of 10-second window is 10 times larger than the update frequency of 100-second window. The effect of window size in resource utilization is significant only when the network is heavily loaded.

In the next experiment (Fig. 8), we evaluated the effect of W in a pipe resource estimation and pipe-based admission control scheme, which is a combination of both parameter- and measurement-based methods. The pipe size was fixed (set to 2 Mbps), meaning that no pipe-resizing process takes place during the experiment. The average reservation demand of a pipe was 3 Mbps, which was the aggregated demand from 35 sources. The mean, minimum, and maximum traffic rates of a reservation were set to 100, 70 and 140 kbps, respectively. This setting allows us to examine the pipe utilization under heavily-loaded network conditions. The ingress router (R1) sent the up-to-date pipe utilization state to the IDRM with an interval of W. When an IDRM granted a new reservation, it simply updated the pipe database according to the peak rate of the reservation. When there were no available

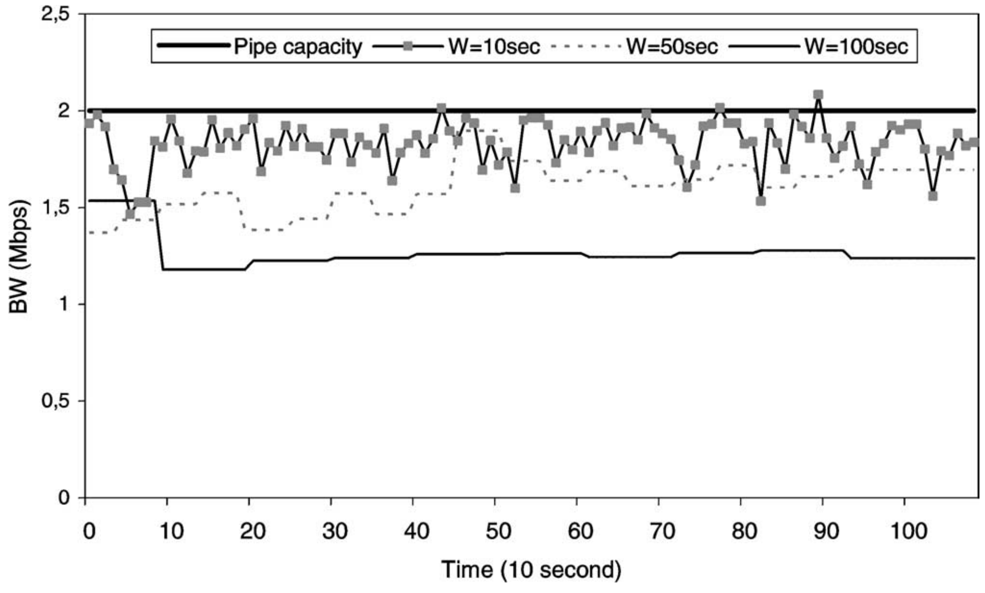


Fig. 8. A pipe utilization with variable window size under heavily loaded network conditions.

Table 3
The effect th on the numbers of database update

window size	$th = 1\%$	$th = 5\%$	$th = 10\%$	$th = 15$
10	115	93	71	48
20	54	41	32	27
40	30	25	22	13

resources left, the new incoming requests were rejected. Once the measurement results arrived, the pipe traffic rate was immediately updated with the new results.

Figure 8 depicts the pipe utilization with respect to variable window sizes. As expected, increasing W decreases the pipe utilization. During the experiments, 611, 439, and 357 reservations were accepted for 10, 50, and 100 seconds window sizes, respectively. As seen, there is a clear trade-off between the frequency of database update and resource utilization. While a 10-second window estimated the most accurate link and pipe utilization, it introduced the highest update overhead. On the other hand, a 100-second window had the least update frequency but caused inefficient resource utilization (under heavily loaded network conditions), because the available resource on a link or in a pipe remained unknown during W, and thus the admission control decisions during this period were performed based on worst-case conditions (peak rate based).

Figure 8 also shows that with the 10-second window size, the traffic rate exceeds the pipe size several times. In this case the router simply drops the excessive packets, and therefore causes QoS degradation. However, the average packet loss ratio is still lower than the pre-defined packet loss ratio l. So that these packet drops do not violate the pre-defined QoS commitment. (As mentioned earlier, in the measurement-based schemes QoS guarantees are statistical.)

Table 3 shows the scalability of our threshold-based (th) database update scheme, in which a router use small window sizes in traffic rate estimation, but updates the database only when a significant change occurs in link/pipe utilization (when the utilization exceeds th). As described in Section 4.3 (Eqs (6) and (7)), the value of th is changed with respect to the network load. As expected, th has a significant effect on the database update frequency. As the th increases (long update intervals) the number of update frequency decreases and in turn scalability in-

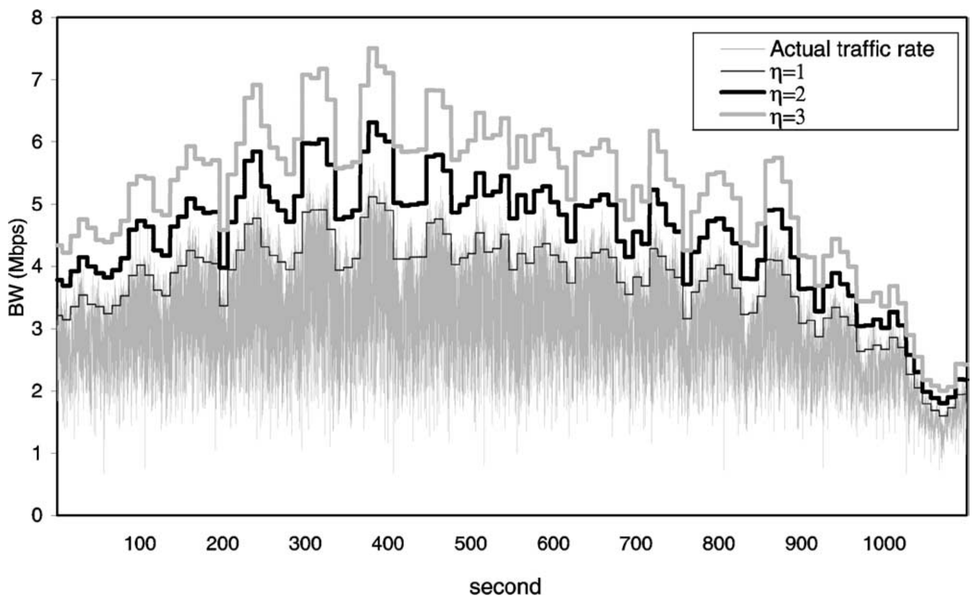


Fig. 9. Class-based traffic rate estimation.

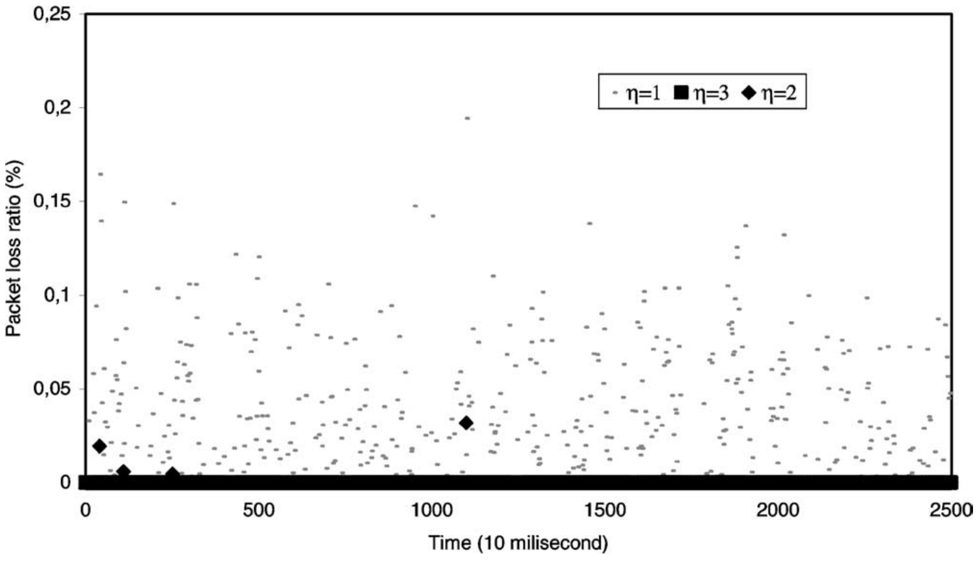


Fig. 10. The classes' loss ratio with respect to η.

creases. As Table 2 shows, when the network is lightly-loaded, reducing update frequency (increasing th) does not result in poor resource utilization.

Figures 9 and 10 illustrates the performance of our measurement-based scheme under class-specific QoS constraints. We defined three different classes that have $\eta = 1$, $\eta = 2$, and $\eta = 3$, which in turn represent a 6%, 2.27% and 0.1% packet loss ratio ($\eta = Q^{-1}(l)$). The window size was set to 10 seconds for the entire duration of the experiment.

As shown in Fig. 9, the rate estimation over the same set of traffic samples varied with classes' loss ratio constraints. The trade-off here is between resource utilization and QoS. While the lower η (higher tolerable loss

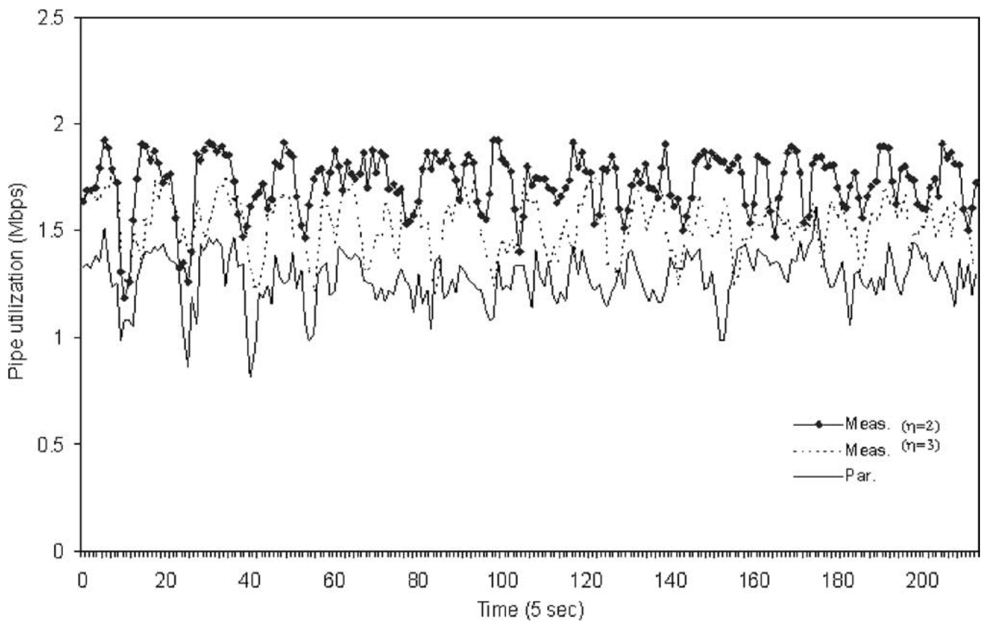


Fig. 11. Measurement-based vs. parameter-based in pipe utilization.

ratio) achieves higher resource utilization, it substantially degrades the QoS by dropping some packets (Fig. 10). On the other hand, the higher η increases the QoS, but results in poor resource utilization. As seen, each class' (PHB) traffic rate estimation in turn resource utilization depends on their tolerable loss rate (η). This is one of the feature that make our measurement-based method different from previous ones [19,21]. A provider can choose the η based on the PHB QoS characteristics. For example, it can choose high value for EF service [18], which requires minimum packet loss ratio, and small value for loss-tolerant services.

Figure 10 shows the importance of Gaussian approximation in achieving the given QoS constraints. For the measurement intervals of 10 milliseconds, Class 1 ($\eta = 1$) and class 2 ($\eta = 2$) met their given loss-ratio constraints with 98.1% and 99.92%, respectively. There was no packet lost in class 3 ($\eta = 3$). This means that the QoS commitments are satisfied. Each class's packet loss ratio is less than its pre-defined tolerable rate.

Figure 11 depicts the efficiency of the measurement-based scheme compared to the parameter-based scheme in pipe utilization. The pipe size and the average incoming reservation demand were set to 2 Mbps and 2.5 Mbps, respectively. As expected, the measurement-based scheme has much higher utilization than the parameter-based. Even when the loss ratio was 0.1% (which is less than the tolerable loss ratios of most of real-time applications), the average pipe utilization was increased about 20%. This validates the efficiency of using the measurement-based admission control scheme for the services that can tolerate statistical QoS guarantees.

In the next set of experiments, we evaluated the IDRM performance in terms of admission control time and BB processing load. To do this, a large network size was needed. Unfortunately, it was difficult to build such a network in a lab environment. We therefore evaluated these features with a simulation scenario. We used our BB software tool presented in [17,26].

For these experiments, we had a network topology consisting 20 edge routers (ingress, egress) and 35 core routers. Edge routers had mesh connections, so there were about 380 intra-domain pipes. As described before, in our model, upon receiving a reservation request, the BB performs the admission control based only on the current utilization of the corresponding pipe. It does not check the capacity of the links constituting the pipe, because pipes' capacities are determined in advance. In on demand scheme (the traditional approach of a BB such as [6,35]), the BB first determines the path, and then check the capacity of all the links along the path.

As depicted in Fig. 12, the IDRM significantly increases the admission control time compared to the on demand scheme. Figure 13 shows the performance of IDRM with variable network sizes (variable number of intra-domain pipes).

As mentioned above, one of the goals of the IDRM is to minimize the pipe size modifications. Figures 14 and 15 illustrate the IDRM pipe resizing scalability under different network loads (the numbers represent the sum of updates for the duration of 20 minutes). As shown, when the network is lightly-loaded, the resizing frequency is very small (Fig. 14). However, when the network is heavily-loaded (e.g., more than 80%), the pipe resizing increases exponentially. This can result in serious scalability problem (the IDRM needs to check all the links's capacity and resize all the pipes). To reduce this problem, the IDRM uses periodic update scheme for heavily-

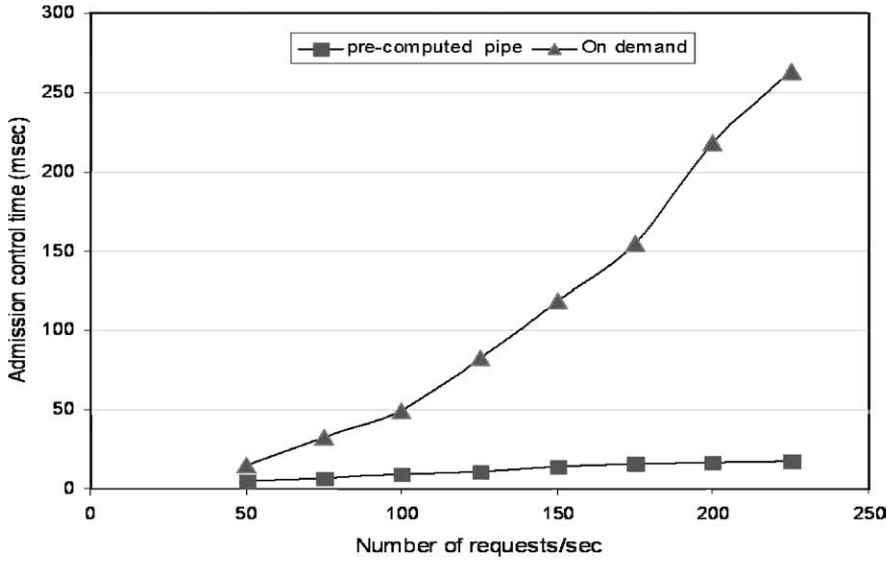


Fig. 12. Admission control time of a BB/server.

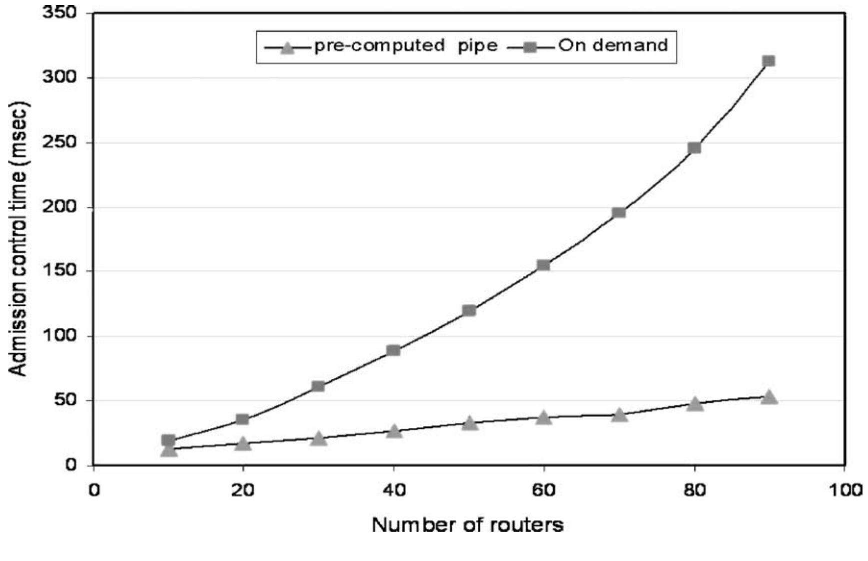


Fig. 13. The admission control time of a BB/server with variable network size.

loaded network conditions. For example, when the pipe resizing frequency exceeds some pre-determined threshold, the IDRM resizes the pipes periodically. (In this experiment, the IDRM started periodic update when the average of resizing intervals becomes less than one minute.) As shown in Fig. 15, using periodic updates keeps the resizing frequency in a reasonable range. The resizing frequency can vary with the network size and traffic characteristics. For example, in stub networks the traffic rate fluctuation is more than the one in transit networks [27], and thus the periodic resizing frequency in stub networks should be higher than the one in transit networks.

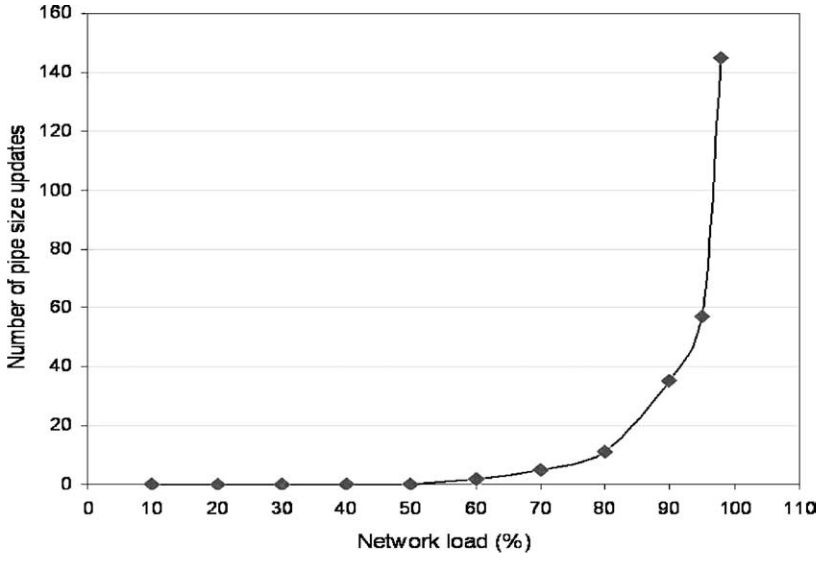


Fig. 14. The IDRM pipe resizing scalability.

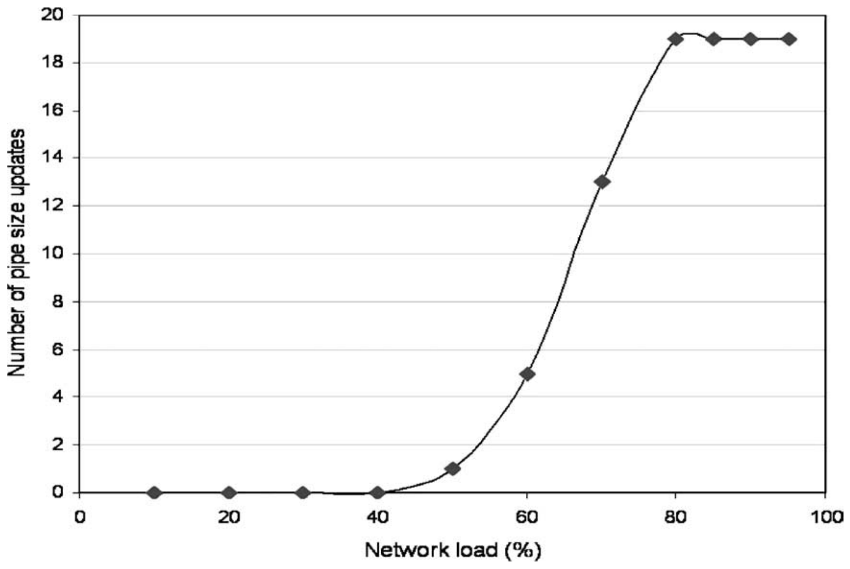


Fig. 15. The IDRM pipe resizing with periodic update.

6. Conclusion and future work

In this paper we presented the design and implementation of an intra-domain resource manager (IDRM) for quantitative QoS guarantees across a BB-supported DiffServ domain. The IDRM uses centralized network state maintenance and pipe-based intra-domain resource management schemes. The proposed model significantly reduces admission control time and minimizes scalability problems present in prior research while optimizing network resource utilization.

In the future we will work on the intra-domain traffic engineering problems for BB-supported DiffServ model. We will also investigate the reliability of the BB, which is an important problem of all the centralized schemes.

Acknowledgement

This project is supported by NSF award NMI ANI-0123939.

References

[1] http://snafu.freedom.org/linux2.2/.

[2] http://www.caida.org/dynamic/analysis/workload/sdnap/.

[3] QBone Signalling Design Team: Simple Inter-domain Bandwidth Broker Signaling (SIBBS), http://qbone.internet2.edu/bb/index.shtml.

[4] Traffic generator software, http://www.postel.org/tg/.

[5] MPLS AutoBandwidth Allocator for MPLS traffic Engineering, *Cisco White Paper*, June 2001.

[6] P. Aukia, M. Kodialam and P. Koppol, RATES: A server for MPLS traffic engineering, *IEEE Network Magazine* (March) (2000).

[7] F. Baker, C. Iturralde, F. Faucheur and B. Davie, Aggregation of RSVP for IPv4 and IPv6 Reservations, *RFC 3175*, September 2001.

[8] C.-S. Chang and J.A. Thomas, Effective bandwidth in high speed networks, *IEEE Journal on Selected Areas in Communications* **13** (1995), 1091–1100.

[9] L. Cruz, Quality of service guarantees in virtual circuit switched networks, *IEEE Journal of Selected Areas in Communications* (special issue on).

[10] A. Elwalid, C. Jin, S. Low and I. Widjaja, MATE: MPLS adaptive traffic engineering, in: *IEEE The Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Infocom 2001*, Alaska, USA, 2001.

[11] G. Armitage, et al., A Delay Bound alternative revision of RFC 2598, *RFC 3248*, March 2002.

[12] P. Trimintzios, et al., An architectural framework for providing QoS in IP differentiated services networks, in: *Proceedings of the International Symposium on Integrated Network Management*, Washington, USA, 2001.

[13] S. Blake, et al., An architecture for differentiated services, *RFC2475*, December 1998.

[14] S. Floyd and V. Jacobson, Link-sharing and resource management models for packet networks, *IEEE/ACM Transactions on Networking* **3**(4) (1995).

[15] R. Guerin, A. Orda and D. Williams, QoS routing mechanisms and OSPF extensions, in: *Proceedings of 2nd Global Internet Miniconference (joint with Globecom'97)*, Phoenix, AZ, USA, 1997.

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

[17] J. Hwang, S. Chapin, H. Mantar and I. Okumus, An implementation study of a dynamic inter-domain bandwidth management platform in DiffServ networks, in: *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, 2004.

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

[19] S. Jamin, P.B. Danzig, S.J. Shenker and L. Zhang, Measurement-based admission control algorithm for integrated services packet networks, *IEEE/ACM Transactions on Networking* **5**(1) (1997), 56–70.

[20] I. Khalil and T. Braun, Edge provisioning and fairness in vpn-diffserv networks, *Journal of Network and System Management* **10**(1) (2002).

[21] E. Knightly and J. Qiu, Measurement-based admission control with aggregate traffic envelopes, in: *IEEE IWDC '98*, Italy, 1998.

[22] T. Li and Y. Rekhter, A provider architecture for differentiated services and traffic engineering, *RFC 2490*, January 1999.

[23] R. Liao and A. Campbell, Dynamic edge provisioning for core IP networks, in: *Proc. of the 8th International Workshop on Quality of Service (IWQoS)*, PA, USA, 2000.

[24] D.N.C.M. Grossglauser, Tse: A framework for robust measurement-based admission control, *IEEE/ACM Transactions on Networking* **7**(3) (1999), 293–309.

[25] H.A. Mantar, J. Hwang, S. Chapin and I. Okumus, A scalable intra-domain resource management scheme for DiffServ Networks, in: *The Second International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HETNETs-04)*, Yorkshire, UK, 2004.

[26] H.A. Mantar, J. Hwang, I.T. Okumus and S.J. Chapin, A scalable model for interbandwidth broker resource reservation and provisioning, *IEEE Journal on Selected Areas in Communications* **22**(10) (2004), 2019–2034.

[27] H.A. Mantar, Scalable resource management framework for QoS-enabled multi-domain networks, PhD Dissertation, Syracuse University, August 2003.

[28] J. May, OSPF Version 2, *RFC 2178*, July 1997.

[29] J. Liebeherr, N. Christin and T.F. Abdelzaher, A quantitative assured forwarding service, in: *IEEE the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, Infocom 2002*, New York, NY, 2002.

[30] K. Nichols and B. Carpenter, Definition of differentiated services per domain behaviors and rules for their specification, *RFC 3086*, April 2001.

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

[32] I.T. Okumus, H.A. Mantar, J. Hwang and S.J. Chapin, Inter-domain qos routing on diffserv networks: a region-based approach, *Computer Communications* **28**(2) (2005), 174–188.

[33] I.T. Okumus, J. Hwang, H.A. Mantar and S.J. Chapin, Inter-domain LSP setup using bandwidth management points, in: *Proc. IEEE Global Communications Conference, Globecomm 2001*, 2001.

[34] A. Papoulis, *Probability, Random Variables, and Stochastic Process*, 3rd edition, McGraw-Hill, New York, 1991.

[35] C. Scoglio, T. Anjali, J.C. Oliveira, I.F. Akyildiz and G. Uhl, TEAM: A traffic engineering automated manager for DiffServ-based MPLS networks, *IEEE Communication Magazine* (October) (2004).

[36] A. Terzis, L. Wang, J. Ogawa and L. Zhang, A two-tier resource management model for the Internet, in: *Proceedings of Global Internet*, 1999.

[37] Z. Wang and J. Crowcroft, Quality-of-Service routing for supporting multimedia applications, *IEEE Journal of Selected Areas in Communications* **14**(7) (1996), 1228–1234.

[38] X. Xiao, A. Hannan, B. Bailey, S. Carter and L.M. Ni, Traffic engineering with MPLS in the Internet, *IEEE Network Magazine* (March) (2000).