A Flexible and Distributed Architecture for Adaptive End-to-End QoS Provisioning in Next-Generation Networks

Jie Yang, *Student Member, IEEE*, Jian Ye, Symeon Papavassiliou, *Member, IEEE*, and Nirwan Ansari, *Senior Member, IEEE*

***Abstract*—In this paper, a novel distributed end-to-end quality-of-service (QoS) provisioning architecture based on the concept of decoupling the end-to-end QoS provisioning from the service provisioning at routers in the differentiated service (DiffServ) network is proposed. The main objective of this architecture is to enhance the QoS granularity and flexibility offered in the DiffServ network model and improve both the network resource utilization and user benefits. The proposed architecture consists of a new endpoint admission control referred to as explicit endpoint admission control at the user side, the service vector which allows a data flow to choose different services at different routers along its data path, and a packet marking architecture and algorithm at the router side. The achievable performance of the proposed approach is studied, and the corresponding results demonstrate that the proposed mechanism can have better service differentiation capability and lower request dropping probability than the integrated service over DiffServ schemes. Furthermore, it is shown that it preserves a friendly networking environment for conventional transmission control protocol flows and maintains the simplicity feature of the DiffServ network model.**

***Index Terms*—End-to-end performance, quality-of-service (QoS), service granularity, service provisioning.**

I. Introduction

IN TODAY'S Internet, the Internet protocol (IP) becomes the vehicle for delivering various types of data flows and services, including current and emerging real-time and multimedia applications, such as voice, image, and video streams. However, the current IP infrastructure lacks the support of quality-of-service (QoS), as required by many real-time and multimedia applications, thus, significantly hampering its further development.

In order to extend the current Internet service model so that QoS can be better supported, two fundamental frameworks have been proposed, namely, integrated services (IntServs) [1] and differentiated services (DiffServs) [2]. The IntServ model, which aims to provide "hard" end-to-end QoS guarantees to each individual data flow, requires per-flow-based resource allocation and service provisioning and, thus, suffers from the scalability problem due to the huge amount of data flows that may coexist in today's high-speed core routers. The proposed DiffServ model simplifies the design of core routers by aggregating individual flows at edge routers and provisioning only a number of services to the aggregated data flows at each core router. However, in this model, it is difficult to identify each individual flow's QoS requirements at core routers, and to contrive efficient resource allocation mechanisms to guarantee the end-to-end QoS of each individual data flow. Various alternatives have been proposed in order to exploit the benefits of both IntServ and DiffServ, while avoiding their drawbacks. The main challenge here is to find the balance between the implementation simplicity in the network and the service guarantee of the individual flows.

As an example, in [3], the operation of IntServ over DiffServ model was introduced. In this model, the admission control and resource allocation procedures are adopted from those in the IntServ model so that sufficient resources can be reserved to satisfy the data flows' QoS requirements, while the data flows are served in the network domain in a DiffServ fashion, i.e., data flows are aggregated and provided only with a limited number of services. This approach slightly increases the implementation complexity of DiffServ networks by requiring per-flow-based admission control and resource allocation (either at the routers or at some intelligent agents such as "bandwidth brokers") to guarantee each individual flow's QoS requirements, while maintaining the simplicity of DiffServ service provisioning.

However, how to map each individual flow's end-to-end QoS requirements from the IntServ model to the DiffServ model still remains a challenging issue. Currently, some Internet equipment vendors statically map the data flows to a specific service class according to their QoS requirements, and make resource reservation in expedited forwarding (EF) service for flows with "hard" QoS requirements [4]. Thus, along the data path, flows mapped to the same service class may get similar end-to-end delay, loss probability, etc., even though they may actually have quite different QoS requirements. For a DiffServ network with n services at each router, the QoS granularity provided in the network will be $O(n)$, while by static service mapping the granularity that a data flow can actually obtain is $O(1)$. That is, only coarse QoS granularity can be provided in such a QoS mapping and provisioning paradigm, which affects not only the network resource utilization but the user benefit from the network service as well, especially when pricing schemes are introduced into the network service provisioning.

Manuscript received December 1, 2003; revised May 15, 2004. This work was supported in part by the New Jersey Center for Wireless Networking and Internet Security, and in part by a National Science Foundation (NSF) Career Award ANI-0237331.

The authors are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: jxy9918@njit.edu; jxy4864@njit.edu; papavassiliou@adm.njit.edu; nirwan.ansari@njit.edu).

Digital Object Identifier 10.1109/JSAC.2004.839422

A. Paper Objective and Outline

In this paper, we introduce the concept of decoupling the end-to-end QoS provisioning from the service provisioning at each core router to enhance the QoS granularity, which enables a flow to choose different service classes and their associated per-hop behaviors (PHBs) at different routers. Thus, the definition of services and their PHBs may remain the same at each router as in the conventional DiffServ network, while if the path contains m intermediate routers with n available services at each hop, the granularity of the end-to-end QoS provisioned to a data flow by the network can be as fine as $O(n^m)$.

However, there are still some issues in the implementation of such an approach which are associated with the place and methodology of determining the most appropriate services for each data flow. In the current service model, there might be three choices available: the edge routers, the core routers along the data path, and the bandwidth brokers. Since each core or edge router may only have local information on the resource utilization of each service, it is difficult for them to evaluate the end-to-end performance of a data flow and choose the services in an end-to-end fashion. On the other hand, if the corresponding services are determined by one or several bandwidth brokers in the network, the computational overhead of the optimization procedure in finding the services for each flow is prohibitive, due to the possible huge amount of requests.

In [5], the framework of a novel endpoint admission control mechanism [6] referred to as explicit endpoint admission control (EEAC) was proposed, which provides an ideal vehicle for the support and implementation of detecting the resource availability of different services at a router. One of the main features of the EEAC scheme is that the routers along the path are required to explicitly report their service and resource availability upon receiving a probing packet from the end host. Furthermore, the concept of *service vector* which allows a flow to use different services at different routers along its data path was introduced. However, how the EEAC scheme and the service vector concept can be incorporated [referred to as the EEAC with service vector scheme (EEAC-SV)] and deployed in the current DiffServ network architecture, and how the routers can efficiently estimate and report their service performance in the probing packet remain challenging research issues.

In this paper, we further develop and examine the service provisioning architecture and algorithm at both the user and network sides to implement the EEAC-SV scheme, so that the proposed idea of decoupling the end-to-end QoS provisioning from the service provisioning at each router can be efficiently deployed based on the current DiffServ network paradigm. We also demonstrate that by decoupling the end-to-end QoS provisioning from the service provisioning at each router, not only can the user achieve more benefits from the network service, but also the user request dropping probability can be reduced and the network resource utilization can be improved as compared with various service mapping mechanisms in IntServ operations over DiffServ networks.

The rest of the paper is organized as follows. In Section II, we first discuss some related work in the literature, and in Section III, we study and analyze the architecture for decoupling the end-to-end QoS provisioning from the service provisioning at each router. The properties and advantages of the proposed approach over the different service mapping techniques in DiffServ networks are discussed in Section IV, while the performance of the proposed scheme is evaluated in Section V. The concluding remarks are provided in Section VI.

II. Related Work

Different design options of endpoint admission control schemes that have been reported in the literature were discussed in [6], in which probing packet trains are sent to the network to detect the network service performance. The test models for probing packet trains or probing packet pair [7], [8] demonstrate that long probing packet train may be needed to achieve accurate probing results regarding the end-to-end QoS performance [6], which significantly increases the probing overhead and even makes endpoint admission control in some cases impractical [9]. To reduce the probing overhead and enhance the probing accuracy, the performance measurement can be conducted by each router and conveyed to the end hosts by marking or dropping the probing packet [10]. Thus, it is not necessary to send a large sequence of probing packets to the network. Similarly, the explicit congestion notification (ECN) [11] scheme also allows the router to mark the data packet to indicate the network congestion.

However, all these schemes assume that the data flow will use the same service along the data path and, therefore, they cannot be applied conveniently to indicate the performance of each service at each router along the data path, but only the end-to-end performance of a specific service. Although we follow the similar approach of packet marking in this paper, based on the measurement performed by the router, each router can only mark its "own" bits in the probing packet, by which the performance of each service at each router can be represented in the probing packets.

In the literature, user benefit optimization models that introduce pricing schemes into the network service provisioning have been studied to enhance the user benefits from the network services [12], [13]. It is proven in [13] that when throughput (which can be represented by flow data rates and packet dropping probability) is used as the QoS parameter, priority-based differentiated service provisioning can reach max–min fairness among flows in the network. However, to reach the fairness for K flows in the network, the network may need to provide as many as K priorities, i.e., the service granularity may be as fine as $O(K)$. In the current DiffServ network architecture, the service granularity is $O(n)$, where n is the number of service classes in the network. Usually, $n < K$ and the max–min fairness among the flows may not be achieved. On the other hand, in the proposed EEAC-SV scheme, different service vectors may lead to different end-to-end service priorities and performance in the network. As mentioned before, if a data flow passes m routers, the end-to-end service granularity can be as fine as $O(n^m)$. Thus, it is easier and more feasible to achieve the max–min fairness. Moreover, adjusting the data rate sent to the network as adopted in [12] and [13] can also be incorporated into our proposed EEAC-SV scheme and the user benefit optimization model.

III. End-to-End Service Provisioning Architecture

A. Implementation of EEAC-SV via Resource Reservation Protocol (RSVP)

The EEAC-SV operation procedure includes two stages at the user side (end host): the probing and admission control stage, and the data forwarding stage. Detailed description of the operation of the EEAC-SV procedure can be found in [5]. Although the core routers in the EEAC scheme are assumed to be capable of providing an "explicit report" of the service performance and availability, thus increasing the complexity of the core router design due to the fact that the core routers may need to support per-flow request (the probing packets), the increased complexity is similar to that of the IntServ operations over the DiffServ network which may require the core routers supporting RSVP PATH and RESV messages [3]. In fact, the EEAC-SV scheme can be implemented by extending the RSVP, such that the RSVP PATH message serves as the probing packet and the RESV message serves as the acknowledgment packet. However, in the EEAC-SV scheme, we do not need the core routers to keep the state of the RSVP PATH and RESV pair as in the conventional RSVP model, thus simplifying the core router design as compared with the IntServ operations over DiffServ networks.

Suppose a flow is going from the source to the destination via m intermediate routers. At each router, the set of services that can be used by the flow is $S = (S_0, S_1, \ldots, S_{n-1})$. A flow may choose service s_i at router i ($s_i \in S$), which may be different from service s_j it chooses at router j. Thus, a service vector is defined as $s = (s_0, s_1, \ldots, s_{m-1})$.

Let us also assume that a user data flow may obtain an end-to-end QoS performance denoted by $R(s)$, which leads to its utility represented by a utility function $U(R)$. The utility functions describe users' level of satisfaction with the perceived QoS and characterize how sensitive users are to the changes in QoS [14]. In general, we assume $U(R) \geq 0$ and the higher the utility the more satisfied the users. To achieve a certain utility, the user pays for a cost represented by a cost function C, which is determined by the pricing policy p and the service vector s the flow chooses. The objective of the end host is to maximize its benefit from utilizing the network services, which is determined by the utility function and the associated user cost function. In this paper, following the similar approach as in [12] and [13], the end host behavior in selecting the appropriate network services is modeled and represented by the following optimization:

$$G = \max_{s} \left(U(R) - C(s,p)\right) \quad (1)$$

$$\text{s.t.} \quad R(s) \in \Re \quad (2)$$

where $\Re$ is the space of R that meets the end-to-end QoS requirements of the data flow, and s is the service vector that a flow may choose. If a feasible service vector cannot be found to satisfy the user's QoS requirements after probing, the end host withdraws the flow request. If a service vector s is found, the EEAC-SV procedure proceeds to the data forwarding stage and the service vector may be represented by a group of labels attached to the data packets. Like RSVP, probing should be performed periodically so that the flow can adjust its service vector to the dynamics of the network, while the route during the probing and data forwarding stages is assumed to remain the same, and each label can, thus, be read by the proper router.

B. Probing Packet Marking Scheme

Since each router in the ideal EEAC model is required to report the performance of each of its service classes in the probing packet so that the end hosts can obtain the service vector that maximizes their benefits, there are two issues associated with the implementation of the reporting scheme at each router: 1) how to evaluate and predict the availability of each service class so that the incoming data flow will receive the performance as described in the probing packets and 2) how to represent the performance of each service class in the probing packet.

1) Assumptions on Service Provisioning in Routers: There are many parameters that can be used to represent a flow's QoS requirements including bandwidth, delay bound, and loss probability. Since in general the capacity of the core network is much larger than the data rate of an individual data flow, i.e., the bandwidth requirement of a data flow is negligible as compared with the bandwidth of the core networks, we adopt the end-to-end delay bound and packet dropping probability as the descriptors of a flow's end-to-end QoS requirements. Correspondingly, the QoS parameters used to describe the service provisioning characteristics at each router are local delay bound and packet dropping probability.

In the DiffServ model, the individual flows are classified and aggregated at the edge routers, while packets are queued and serviced according to their classes at the core routers. Packets with the same class at the router will be served in a first-in–first-out (FIFO) manner. Although it may be difficult to guarantee the end-to-end performance for a data flow in a conventional DiffServ network due to the lack of resource reservation, it is reasonable to assume that performance bounds can be provided by each service class at each router. In the literature, weighted fair queueing (WFQ) [15] is recognized as a scheduling policy that can guarantee the delay bound and fair resource allocation to each service class. Without loss of generality, in this paper, we assume that WFQ is used as the scheduling algorithm among the different service classes at each router, although other schedulers are equally applicable.

We assume that three classes of service are provisioned in an edge or core router: EF service, assured forwarding (AF) service,[1] and best effort (BE) service, i.e., $S = \{\mathrm{EF}, \mathrm{AF}, \mathrm{BE}\}$. We also assume that probing packets use the EF service. Let us denote by R the output link capacity, while the maximum buffer length of each service class is denoted by L_{EF}, L_{AF}, and L_{BE}, respectively.[2] Then, the corresponding delay bound can be represented by $D(S_j) = (L_{S_j}/R_{S_j}) + (L/R)$, where

[1]For simplicity, we only implement the AF class with one dropping precedence, which does not compromise our general conclusions.

[2]Without loss of generality, we assume that the buffer length at different routers for the same service class is the same, while the buffer length of different service classes at each router may be different from each other.

$S_j \in \{\text{EF}, \text{AF}, \text{BE}\}$, L is the maximum packet length, and R_{S_j} is the guaranteed service rate of S_j [15]. To implement the EF, AF, and BE service differentiation, we guarantee $D(\text{EF}) < D(\text{AF}) < D(\text{BE})$. For the EF and AF services, we assume that some random packet dropping schemes such as core-stateless queueing [16] or random early detection (RED) [17] are applied so that the dropping probability at a router has the following feature:

$$P_{S_j}(t) = \begin{cases} 1, & \bar{L}_{S_j}(t) \geq L_{S_j} \\ \leq p_{S_j}^{\max}, & L_{S_j}^{\min} \leq \bar{L}_{S_j}(t) < L_{S_j} \\ 0, & 0 \leq \bar{L}_{S_j}(t) < L_{S_j}^{\min} \end{cases} \quad (3)$$

where $S_j \in \{\text{EF}, \text{AF}\}$, $\bar{L}_{S_j}(t)$ is the average length of the queue at time t, $L_{S_j}^{\min}$ is the threshold above which the dropping algorithm starts dropping packets at a probability determined by the arrival rate [16] or buffer occupancy [17], and $p_{S_j}^{\max}$ is the maximum packet dropping probability of service S_j at a router. For the BE service, we simply perform the drop-tail and the packet dropping probability is given by

$$P_{\text{BE}}(t) \approx P\left[\bar{L}_{\text{BE}}(t) > L_{\text{BE}}\right] \quad (4)$$

where $\bar{L}_{\text{BE}}(t)$ is the average queue length of the BE service at time t. We also assume that $p_{\text{EF}}^{\max} < p_{\text{AF}}^{\max} < P[\bar{L}_{\text{BE}}(t) > L_{\text{BE}}]$. The values of $\bar{L}_{S_j}(t)$, $S_j \in \{\text{EF}, \text{AF}, \text{BE}\}$, are updated upon the arrival of a packet of service S_j and computed by using the exponential moving average [17], as follows:

$$\bar{L}_{S_j}(t) = \left(1 - e^{-\frac{\tau_{S_j}(t)}{K}}\right) L_{S_j}(t) + e^{-\frac{\tau_{S_j}(t)}{K}} \bar{L}_{S_j,\text{old}}(t) \quad (5)$$

where $\tau_{S_j}(t)$ is the interval between the current time t and the arrival time of the previous received packet of service S_j, and $\bar{L}_{S_j,\text{old}}(t)$ is the most recently updated average queue length before t, while K is a constant as described in [16], and $L_{S_j}(t)$ is the queue occupancy at time t.

2) Probing Packet Marking Algorithm: In the EEAC-SV scheme, if a router can accurately predict the performance of its service classes and report them in the probing packet, each flow will receive the guaranteed QoS performance it desires with the help of the service vector. Although the QoS granularity can be greatly enhanced using this approach, the estimation of the performance of each service cannot be accurate and attaching the estimated performance data to the probing packet increases the implementation complexity at the router. At the same time, the packet size grows with the number of routers that the probing packet traverses to the destination. For example, if the delay and packet dropping probability are reported at each router in the format of 4-byte integer or floating number, each router needs 8 bytes (64 bits) to report its performance for each service. If n services are provisioned at each router, and a probing packet traverses m routers, 8 mn bytes (64 mn bits) will be added to the probing packet. When $m = n = 3$, i.e., three routers are traversed, each provisioning three services, the probing field of a probing packet will be 72 bytes (576 bits).

To alleviate these drawbacks, we first discretize the performance of each service class into several congestion levels. In a probing packet, each router that the packet traverses is allocated several bits to represent the congestion levels of its service classes. A router can indicate its congestion level of a service class by marking these bits. As a result, we can use $\lceil \log_2 Y_{S_j} \rceil$ bits to represent Y_{S_j} congestion levels for service class S_j. For n service classes, we need $(n \lceil \log_2 Y_{S_j} \rceil)$ bits in a probing packet to represent the congestion levels at a router. If the data path consists of m routers, the probing packet needs $(mn \lceil \log_2 Y_{S_j} \rceil)$ bits to represent all the congestion levels along the data path. For example, based on the service provisioning assumptions in this paper, for three congestion levels at each service class in a router, six bits are required to represent these congestion levels in a probing packet. If the path consists of three routers, the total length of the probing bits will be 18, which is only 3% of the length needed (576 bits) in the directly reporting approach described above. Moreover, the packet marking is also less complicated in implementation than writing the integer or floating value into a packet.

TABLE I
PERFORMANCE BOUNDS UNDER DIFFERENT CONGESTION LEVELS FOR THE EF, AF, AND BE SERVICES AT A CORE ROUTER

Congestion Level	0	1	2
EF	$(\frac{L_{EF}^{\min}}{R_{EF}} + \frac{L}{R}, 0)$	$(\frac{L_{EF}}{R_{EF}} + \frac{L}{R}, p_{EF}^{max})$	$(\infty, 1)$
AF	$(\frac{L_{AF}^{\min}}{R_{AF}} + \frac{L}{R}, 0)$	$(\frac{L_{AF}}{R_{AF}} + \frac{L}{R}, p_{AF}^{max})$	$(\infty, 1)$
BE	$(\frac{L_{BE}^{\min}}{R_{BE}} + \frac{L}{R}, 0)$	$(\infty, 1)$	$(\infty, 1)$

The service provider provides users with the performance bounds at each congestion level of a service class in a router as *a priori* knowledge. When the end hosts receive the probing packet with the marked bits to indicate the congestion level of a service class at a router, they can obtain the performance bounds of the service classes at each router that the probing packet traverses. As an example, let $(d_{S_j,i}, p_{S_j,i})$, $(S_j \in \{\text{EF}, \text{AF}, \text{BE}\}, \text{i} = 0, 1, 2)$ represent the delay and packet dropping probability bounds at each congestion level i of service S_j provisioned by a router. The service performance information that can be used as *a priori* knowledge provided to the end hosts is summarized in Table I. In fact, each of the congestion levels corresponds to a unique buffer occupancy, represented by $B_{S_j,q}$, $(q = 0, \ldots, Y_{S_j} - 1)$. For the EF and AF services, $B_{S_j,0} = L_{S_j}^{\min}$, $B_{S_j,1} = L_{S_j}$, $(S_j \in \{\text{EF}, \text{AF}\})$, and we use $B_{S_j,2} = \infty$ to indicate that no more QoS-required flows can be admitted into the service class S_j at the router. For the BE service, $B_{\text{BE},0} = L_{\text{BE}}^{\min}$ indicates that no packet will be dropped at this level, and $B_{\text{BE},1} = B_{\text{BE},2} = \infty$ represents that the packets may be dropped at the tail of the buffer, and no more QoS-required flows should use the BE service at this router. With *a priori* knowledge of the performance bounds of a service class at the end hosts, and utilizing the probing packet marking at each router, the complexity of the router operation and the length of probing packets can be significantly reduced, as compared with directly reporting the performance of each service at a router in the probing packet.

Let $0 \leq q_1 < q_2 \leq Y_{S_j} - 1$ represent the congestion levels from low to high and $B_{q_1} < B_{q_2}$. At the highest congestion level, $B_{S_j, Y_{S_j}-1} = \infty$ indicates that no more QoS-required

```
On receiving packet p
if ((data packet) or (probing acknowledgement))
//we assume probing acknowledgement packets will use EF service and be serviced as a
// regular data packet of the EF service.
    i=classify (p);
    interval=crt_time-old_time[i];
    rate[i]=update_rate (p.length, interval, rate[i]);
    //use Eq.(7)to update the arrival rate of the packets of service i ;
    old_time[i]=crt_time;
    //keep time for next update of service i;
    drop_prob=calculate_drop(average_queue_length[i],rate[i], i );
    //use Eq. (3) to calculate the dropping probability of EF or AF service.
    //use drop-tail to drop packet of BE service
    if (drop_prob>uniform_distrib(0,1))
        packet_drop(p);
    else
        enqueue(i,p);
        queue_length[i]+=p.length;

  average_queue_length[i]=update_queue(queue_length[i],
                                       average_queue_length[i], interval );
  //use Eq.(5) to update the average queue length after dropping or queuing the packet.
```

Fig. 1. Pseudocode for the process of the data packet or probing acknowledgment packet arrival in a router.

flows can be admitted into class S_j. Let us denote by $\bar{r}_{S_j}(t)$ the average packet arrival rate of service S_j at time t. Assuming that the probing period is T for each flow, when a router receives a probing packet from flow k, the average buffer occupancy of service S_j is increased by $(\bar{r}_{S_j}(t) + \hat{r}^{(k)} - R_{S_j})T$ at the time when the next probing packet of flow k arrives, where $\hat{r}^{(k)}$ is the bandwidth requirement of the flow. Thus, during the interval of $[t, t+T)$, the buffer occupancy is estimated to be in the range of $[\bar{L}_{S_j}(t), \bar{L}_{S_j}(t) + (\bar{r}_{S_j}(t) + \hat{r}^{(k)} - R_{S_j})T)$, and the center point is used as the estimation of the average queue occupancy during the interval. Therefore, we can predict $\hat{B}_{S_j}(t)$ as follows:

$$\hat{B}_{S_j}(t) = \begin{cases} B_{S_j,q}, & B_{S_j,q-1} \le \bar{L}_{S_j}(t) + \left(\bar{r}_{S_j}(t) + \hat{r}^{(k)} - R_{S_j}\right)\frac{T}{2} \\ & < B_{S_j,q}, \ \left(0 < q < Y_{S_j} - 1\right) \\ B_{S_j,0}, & \bar{L}_{S_j}(t) + \left(\bar{r}_{S_j}(t) + \hat{r}^{(k)} - R_{S_j}\right)\frac{T}{2} \\ & < B_{S_j,0}, \ (q = 0) \\ \infty, & \bar{L}_{S_j}(t) + \left(\bar{r}_{S_j}(t) + \hat{r}^{(k)} - R_{S_j}\right)\frac{T}{2} \\ & \ge B_{S_j,Y_{S_j}-2}, \ \left(q = Y_{S_j} - 1\right) \end{cases} \tag{6}$$

where $\hat{B}_{S_j}(t)$ is the prediction of the congestion level in the future T time units (seconds) and $Y_{S_j} \ge 2$ is the number of congestion levels of service S_j. Then, the router marks its bits in the probing packet to indicate the congestion level of each service S_j, $S_j \in S$, according to $\hat{B}_{S_j}(t)$. Upon receiving a probing acknowledgment, with *a priori* knowledge of the performance bounds of each service at a router, the end host can evaluate and determine the optimal solution that maximizes its benefit G using relations (1), (2). Note that although each probing packet may contain the bandwidth requirement of the data flow, if we can assume that a single data flow's impact on the network is negligible and, thus, $\hat{r}^{(k)}T \ll B_{S_j,0}$, we can ignore the term $\hat{r}^{(k)}T$ in (6) and the router does not need the flow information from the probing packet, thus, further simplifying the router operation. Similar to how $\bar{L}_{S_j}(t)$ is computed by using (5), $\bar{r}_{S_j}(t)$ is obtained through the exponential moving average [16]

$$\bar{r}_{S_j}(t) = \left(1 - e^{-\frac{\tau_{S_j}(t)}{K}}\right)\frac{l_{S_j}(t)}{\tau_{S_j}(t)} + e^{-\frac{\tau_{S_j}(t)}{K}}\bar{r}_{S_j,\mathrm{old}}(t) \tag{7}$$

where $l_{S_j}(t)$ represents the received packet length at time t and $\bar{r}_{S_j,\mathrm{old}}(t)$ is the most recently updated average rate before t. At each router, both $\bar{L}_{S_j}$ and $\bar{r}_{S_j}$ are updated when a data packet of service class S_j is received. When a probing packet is received, all of the three $\bar{L}_{S_j}$ and $\bar{r}_{S_j}(S_j \in \{\mathrm{EF}, \mathrm{AF}, \mathrm{BE}\})$ are updated, in which $l_{S_j}(t)$ in (7) $(S_j \in \{\mathrm{AF}, \mathrm{BE}\})$ will be 0 and $l_{\mathrm{EF}}(t)$ is the probing packet length, since we assume the probing packet uses the EF service. The pseudo-codes for the arrival process of a data packet and of a probing packet are shown in Figs. 1 and 2, respectively. Since $\bar{L}_{S_j}(t)$ and $\bar{r}_{S_j}(t)$ are based on the measurements performed by the router, they may reflect the performance of the service class better than measuring and modeling the performance of the probing packet train, which is only a small sample of the stochastic service process, while at the same time the EEAC-SV approach significantly reduces the bandwidth overhead of probing as well. For example, the probing overhead of the proposed EEAC-SV scheme is only 10% of the corresponding overhead that is introduced by a ten-packet probing packet train for testing the network performance. Furthermore, the EEAC-SV scheme may still detect an end-to-end packet dropping probability of less than 10%, which a ten-packet probing packet train may not be able to detect.

IV. Properties of the Service Provisioning Architecture

The main features and advantages of decoupling the end-to-end QoS provisioning from the service provisioning at each router, implemented via the proposed distributed end-to-end QoS provisioning architecture, reside in the following two aspects: 1) it may provide the users the flexibility to choose their most desirable QoS provisioned in the network and 2) it distributes the computational overhead between the end hosts and routers, thus maintaining the simplicity of the router and network architecture.

```
On receiving packet p
if (probing packet)
 //probing packet will use the EF service and maybe dropped due to congestion.
    for (i ∈ {EF,AF,BE})
        interval=crt_time-old_time[i];
        if (i==EF) //determine whether the probing packet will be dropped.
            rate[i]=update_rate (p.length, interval, rate[i]);
            //use Eq.(7)to update the arrival rate of the packets of service i ;
             //since probing packets use EF service, it uses p.length to update rate.
            old_time[i]=crt_time;
            //keep time value for next update of EF service.
             drop_prob=calculate_drop(average_queue_length[i],rate[i], i );
              //use Eq. (3) to calculate the dropping probability of EF service.
              if (drop_prob>uniform_distrib(0, 1))
                    packet_drop(p);
               else
                    queue_length[i]+=p.length;
        else
            rate[i]=update_rate (0, interval, rate[i]);
            //for services other than EF service, the received packet length is 0,
            //queue_length[i] is unchanged.
            // And since no data packet of service i is received, old_time[i] is not updated.
       average_queue_length[i]=update_queue(queue_length[i],
                                          average_queue_length[i], interval );
        //use Eq.(5) to update the average queue length after dropping or queuing the
        // packet ;
       congestion_pattern[i]=estimate_congestion(average_queue_length[i], p.rate,
                                  rate[i], i);
       //use Eq.(6) estimate the congestion level
  if (packet_not_dropped (p)= = TRUE)
      marking_probing_field (p, congestion_pattern);
      enqueue(EF, p);
      // We assume probing packets will use EF service.
```

Fig. 2. Pseudocode for the process of the probing packet arrival in a router.

In principle, it is possible that the available bandwidth and buffers can be shared among different classes and can be dynamically allocated to a service class according to the users' demands. Although it is ideal to have the dynamic allocation schemes to share network resources among different classes at each node, the allocation algorithms are rather complex and there is no consensus on how the allocation should be performed due to the various flow characteristics and QoS requirements, as well as different policies the service providers may have. In the trivial case, when the traffic load to a router is light as compared with its output bandwidth, the dynamic resource allocation schemes and the proposed decoupling mechanism may be equivalent from the network perspective. For example, let us consider a scenario where the AF service is idle and the EF service is congested, while the total traffic load to the router is light, such that the AF service can provide to the flow similar QoS as the EF service. In this case, allocating the idle buffer and bandwidth of the AF service to the EF service is equivalent to reassigning the flows from the EF service to the AF service from the perspective of network resource utilization, if the reassigned flows will not congest the AF service queue (e.g., below the congestion level 0). However, when both services have a large number of data flows to serve, the effects are not the same. Flows entering AF queues may encounter higher dropping probability and delay than those entering the EF queues. Thus, if some buffers and bandwidth are dynamically allocated to the EF service from the AF queue, these buffers and bandwidths may accommodate fewer flows than the scheme in which flows use the service vector to choose the AF service. In the following, we ignore the trivial case that some services are almost idle (i.e., below the congestion level 0) and that the total traffic load to a router is light as compared with its capacity, and we demonstrate that the proposed service decoupling scheme (i.e., the EEAC-SV scheme) can reduce the request dropping probability and enhance the resource utilization in the network by using the notion of "effective bandwidth" ([18], [19], etc.).

The request dropping probability is defined as the probability that a flow will not use the network service due to the unsatisfactory end-to-end QoS. Let the delay bound and packet dropping probability of service S_j when the service queue is in the stable condition be $(d_{S_j,\max}, p_{S_j,\max})$. Thus, the effective bandwidth that flow k may consume at node i is $\hat{r}^k_{i,S_j}(\theta_i)$ where θ_i is a function of $(d_{S_j,\max}, p_{S_j,\max})$, and $\hat{r}^k_{i,S_j}(\theta_i)$ has the following properties [19].

Property 1: $\hat{r}^k_{i,S_j}(\theta_i)$ is an increasing function of θ_i.

Property 2: θ_i is a decreasing function of $d_{S_j,\max}$ and $p_{S_j,\max}$.

Property 3: If the number of flows using service S_j at node i is K_{S_j}, the service can provide bounded delay and loss probability of $(d_{S_j,\max}, p_{S_j,\max})$, $S_j \in S$, and the service queue is in the stable condition, if and only if

$$\sum_{k=1}^{K_{S_j}} \hat{r}^k_{i,S_j}(\theta_i) \le R_{S_j} \tag{8}$$

where R_{S_j} is the guaranteed service rate of class S_j, and $\sum_{S_j \in S} R_{S_j} \le R$.

Without loss of generality, assume that service classes in the service set $S = (S_0, S_1, \ldots, S_{n-1})$ at each node can be sorted according to one or several of the performance metrics. In this paper, we assume that the services in S can be sorted such that $k < l$ for service S_k and service S_l implies that $d_{S_k,r} \leq d_{S_l,q}$ and $p_{S_k,r} \leq p_{S_l,q}$, where $1 \leq r < Y_{S_k} - 1$ and $1 \leq q < Y_{S_l} - 1$ are the congestion levels of service class S_k and S_l, respectively. That is, service S_k can always provide better service in terms of delay bound and packet dropping probability than service S_l when both of their queue occupancies are above the congestion level 0 while these service queues are in the stable condition. We represent such a sorting by $S_k > S_l$ (i.e., service S_k has a higher priority than service S_l). Also assume that each flow of type k has a default service $S_D^k \in S$, which is the only service that is acquired and used along the data path by the flow in the static service mapping schemes, while in the EEAC-SV scheme the service vector $s = (s_0, \ldots, s_{m-1})$ must satisfy $s_i \leq S_D^k$, $(i = 0, 1, \ldots, m-1)$. Note that this condition imposes more constraints in finding the optimal service vector as compared with relations (1), (2). However, since we ignore the trivial case that some service is nearly idle (for example, at congestion level 0), according to the service provisioning scheme in this paper, the added condition does not affect the service vector search results for the flows whose default service is the EF service, which is the case for most of today's real-time applications using the IntServ operations over DiffServ networks and requiring quantitative end-to-end QoS guarantees. Then, we can have the following propositions.

Proposition 1: Assume a flow of type k traverses a single link, and the flow's end-to-end QoS requirement is (D_R, P_R), while the end-to-end QoS that the flow's default service S_D^k can provision is $(D_{S_D^k}, P_{S_D^k})$. If there exists a service S_j, $S_j < S_D^k$, which satisfies $D_{S_D^k} < D_{S_j} \leq D_R$, $P_{S_D^k} < P_{S_j} \leq P_R$, and the effective bandwidth of each individual flow is small enough as compared with the output link R, we have $P_{\text{drop}}(S_j) \leq P_{\text{drop}}(S_D^k)$, where $P_{\text{drop}}(s)$ represents the request dropping probability using service vector s; in this single link case, $s = S_j$ and $s = S_D^k$, respectively.

Proof: This can be easily proven by using *Properties 1–3* above.

Proposition 2: Assume a flow of type k traverses a set of m nodes, each of which provides an output link with capacity R in the network. With the same assumptions as in *Proposition 1*, if there exists a service vector s, such that $D_{S_D^k} < D_s \leq D_R$ and $P_{S_D^k} < P_s \leq P_R$, then among the m nodes there exists a node i in which more flows can be admitted by using service vector s than using S_D^k and, thus, $P_{\text{drop}}(s) \leq P_{\text{drop}}(S_D^k)$.

Proof: Assume two identical networks. Along the data path in network 1 the flows are using only the default service S_D^k, while the flows are allowed to use the service vector s in network 2. Since $D_{S_D^k} < D_s \leq D_R$ and $P_{S^k} < P_s \leq P_R$, there must exist a node i at which the flow chooses service $s_i < S_D^k$. From *Proposition 1*, more flows can be admitted into node i of network 2. Therefore, $P_{\text{drop}}(s) \leq P_{\text{drop}}(S_D^k)$.

Note that the request dropping probability P_{drop} in the EEAC-SV scheme is calculated in this paper as the ratio of the number of probing results that do not meet the end-to-end QoS requirements to the total number of probing requests. We assume that once the probing results indicate that the end-to-end QoS requirements cannot be met, the end hosts will drop the request. Moreover, in the proposition, P_{drop} is not limited to the flows that use the entire m nodes of the data path but also includes the flows using a subset of the m nodes and their attached links. As a result, lower request dropping probability may result in higher resource utilization in the network.

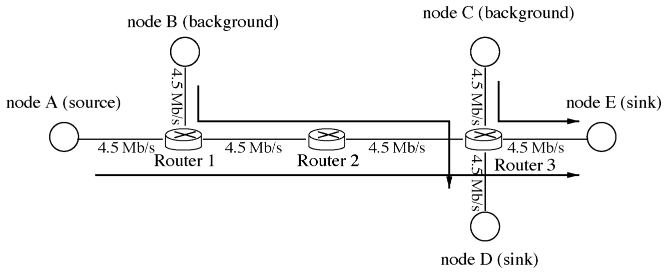


Fig. 3. Simulated network topology.

From Proposition 1 and 2, it can be noted that with same number m of routers along the data path, the network service provider may enhance its benefit in terms of resource utilization when $m \geq 2$, by using the service vector and providing finer end-to-end QoS granularity, as compared with the static service mapping schemes. Moreover, since the end-to-end QoS granularity can be as fine as $O(n^m)$, the larger the value of m, the finer the granularity provided in the network. It should also be noted that the static service mapping for a data flow is a subset solution of the service vector scheme in the user benefit optimization model [(1), (2)]. Therefore, from the user viewpoint, compared with the static service mapping schemes, with the same m along the data path, using the service vector can enhance users' benefits when $m \geq 1$ [5]. However, from Proposition 1 and 2 it can also be noted that both the user and service provider achieve more benefits from the EEAC-SV scheme as compared with the static service mapping schemes, mainly when there exists flexibility between the user required end-to-end QoS and the end-to-end QoS that can be provided by the static service mapping. Since a flow will experience longer end-to-end delay and larger packet dropping probability when m increases, larger m will mainly benefit flows with more flexible end-to-end QoS requirements. For an individual flow with stringent end-to-end QoS requirements compared with the QoS that can be provided by the m routers, the benefits achieved by the EEAC-SV scheme may be limited due to the effect on the performance of the large number of routers utilized.

V. Performance Evaluation and Discussion

In the following, we evaluate the performance of our proposed scheme and compare it with the corresponding performance results of IntServ operations over DiffServ networks through modeling and simulation using OPNET. The network topology used throughout this study is shown in Fig. 3. Specifically, the network consists of three core routers connected by links of 4.5 Mb/s. Three services, EF, AF, and BE service, are provisioned at each router. The normalized weight of EF, AF, and BE services are 0.25, 0.25, and 0.5, respectively. Throughout this study, we use an ON–OFF traffic source to

TABLE II
SUMMARY OF THE BANDWIDTH OCCUPIED BY THE BACKGROUND TRAFFIC

Source	Destination	EF	AF	BE
Node B	Node D	15%	5 %	20%
Node C	Node E	5%	15 %	30%

model some typical types of real-time flows, with an average rate of 12.8 kb/s and the peak rate of 25.6 kb/s. Each data packet is 500 bytes long, while the average on and off period is 0.1 s, respectively. The probing period of each flow is 0.5 s and the length of probing packets is 50 bytes, which is sufficient enough for packet marking. Data flows from node A go to node E only. The background crossing traffic consists of flows going from node B to node D and flows going from node C to node E. The load of the background traffic in each service class with respect to the output bandwidth of the source nodes (node B and node C, 4.5 Mb/s) is shown in Table II.

Three QoS provisioning scenarios are considered and evaluated in this study: EEAC with service vector, IntServ over DiffServ with static resource allocation, and IntServ over DiffServ with dynamic resource allocation.

- EEAC with service vector (EEAC-SV) scheme implements the end-to-end QoS provisioning architecture proposed in the paper. The user will use relations (1), (2) to determine whether or not and what service to use at each router.
- IntServ over DiffServ with static resource allocation (IDSRA) scheme statically maps the flow to a specific service (e.g., the EF service), while at each node the minimum guaranteed bandwidth (i.e., the normalized weight) of the service is kept constant and the buffer allocated to each service queue cannot be shared.
- IntServ over DiffServ with dynamic resource allocation (IDDRA) scheme maps the flow to a service, while both the bandwidth and buffers of each service at each node can be shared among different service classes. For demonstration purposes, we mainly study the EF service provisioning, and compare it with our EEAC-SV scheme. Therefore, we assume that the EF service can share the bandwidth and buffers from BE and AF services, while the other two services cannot share bandwidth and buffers from the EF service. To avoid starvation of AF and BE services, the AF and BE services are guaranteed 50% of their original bandwidth and buffers. Moreover, the dynamic allocation of bandwidth and buffers are nonpreemptive, i.e., only idle bandwidth and buffers can be allocated to the EF service when the EF service needs more resources. Note that the idle bandwidth is defined as $\max(0, R_{S_j} - \bar{r}_{S_j})$, where $S_j \in \{\mathrm{BE}, \mathrm{AF}\}$.

Moreover, for the IDSRA and IDDRA schemes, we assume that the routers can be aware of the flow's end-to-end QoS requirements and make admission control decisions accordingly, although they may map the flows with different QoS requirements to the same service provisioned in the network. For the EEAC-SV scheme, the performance bounds of each congestion level for each service class are summarized in Table III.

TABLE III
PERFORMANCE BOUNDS UNDER DIFFERENT CONGESTION LEVELS FOR THE EF, AF, AND BE SERVICES AT A CORE ROUTER IN THE EEAC-SV SCHEME

Congestion Level	0	1	2
EF	$(0.057s, 0)$	$(0.128s, 0.01)$	$(\infty, 1)$
AF	$(0.142s, 0)$	$(0.284s, 0.02)$	$(\infty, 1)$
BE	$(0.142s, 0)$	$(\infty, 1)$	$(\infty, 1)$

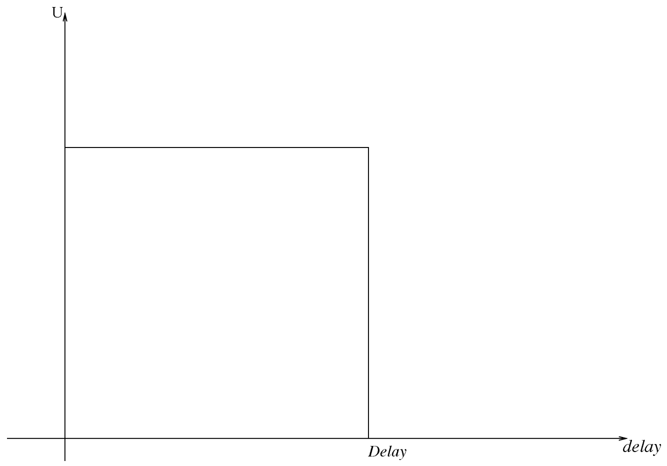


Fig. 4. Utility function of both types of flows.

Three different experiments are performed and presented in this paper. Since the objective of the proposed scheme (EEAC-SV) is to provide fine end-to-end QoS, so that flows with different QoS requirements can achieve different performance, in the first experiment (Experiment 1), we exhibit the performance of the EEAC-SV scheme when heterogeneous QoS requirements of flows coexist in the network. In the second experiment (Experiment 2), the effect of m—the number of routers along the data path—on the performance of flows with different QoS requirements in the EEAC-SV scheme is discussed. Finally, in Experiment 3, we demonstrate that the proposed architecture preserves a compatible and friendly networking environment for conventional TCP flows, which is an essential feature for a QoS provisioning scheme in today's Internet, due to the reason that end-to-end congestion control mechanisms have been widely deployed through TCP [16] and the deployment of the QoS provisioning architecture has to be incremental.

A. Experiment 1—Flows With Heterogeneous QoS Requirements

In this experiment, we use two types of flows with the same traffic arrival patterns as described before and assume that the end-to-end delay requirement of flow type 1 is $D_{\mathrm{type1}} \leq 500$ ms and the end-to-end delay requirement of flow type 2 is $D_{\mathrm{type2}} \leq 750$ ms, while delay is the only QoS parameter of the two types of flows. We further assume that these two types of flows represent inelastic real-time applications [14] whose utility function U in (1) for each type of the flow is as shown in Fig. 4, where Delay$=$ 500 ms for flow type 1 and Delay$=$ 750 ms for flow type 2. Assume that the pricing policy at each router is to charge each packet of the flow according to its service class and the unit price of the packet requiring EF service is normalized as

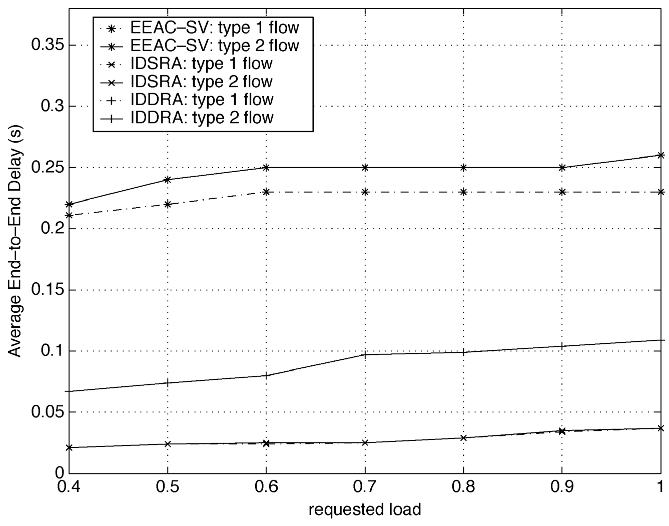


Fig. 5. Average end-to-end delay of flow type 1 and 2 under different operation schemes.

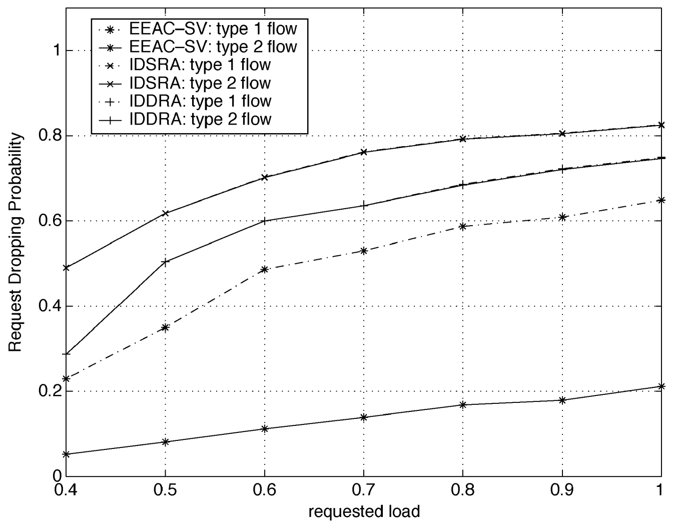


Fig. 6. Request dropping probability of flow type 1 and 2 under different operation schemes.

1 unit, while the unit prices of the AF and BE are assumed to be 0.5 and 0.4, respectively. Then, at the user side the optimization model [(1), (2)] of each type of flows can be simplified to minimizing the total user cost C for delivering the packet from node A to node E, while the end-to-end delay bound of 500 ms and 750 ms can be satisfied, respectively [5]. It can be found that due to the performance bound at each router, the two types of flows have to be mapped to the EF service in both of the IDSRA and IDDRA schemes, and the services provisioned at each router can be sorted as $\mathrm{EF} > \mathrm{AF} > \mathrm{BE}$. Among the traffic generated by node A, 50% of the flows are type 1, while the remaining of the flows are type 2.

Fig. 5 shows the average end-to-end delay of the two types of the flows with different requested load under the three scenarios. The requested load is defined as the ratio of the aggregated bandwidth requirement from the source node A to the backbone speed, i.e., 4.5 Mb/s. In the EEAC-SV scheme, different types of flows have different end-to-end delay due to their different requirements. However, in the IDSRA and IDDRA schemes, the average end-to-end delay of the two types of flows are very close (in the figure, they overlap to each other), due to the fact that the IDSRA and IDDRA schemes cannot provide service differentiation to these two types of flows, both of which are mapped to the EF service. The slight performance difference between the two types of flows in the IDSRA and IDDRA schemes is due to the fact that the admission control procedure may make different admission decisions according to their end-to-end QoS requirements. The results presented in this figure demonstrate that the EEAC-SV scheme can provide finer QoS granularity than the IDSRA and IDDRA schemes.

The effects of the finer QoS provisioning capability of the EEAC-SV scheme are demonstrated in Fig. 6 in which type 1 and type 2 flows received different request dropping probability in the EEAC-SV scheme, while the corresponding values are overlapping under the IDSRA and IDDRA schemes. For the EEAC-SV scheme, the request dropping probability consists of the request dropping probability before a flow starts sending data, and the request dropping probability during the data forwarding stage when the end host finds that no feasible service vector can be found due to the dynamics of the network load. In general, the request dropping probabilities for the two types of flows in the IDSRA and IDDRA schemes are larger than those in the EEAC-SV scheme due to the fact that the finer end-to-end QoS granularity provided by the EEAC-SV scheme can fit the flows' requirements better, and as a result each flow consumes less resources; while the QoS granularity is not enhanced in the IDSRA and IDDRA schemes and each flow may get more service than it actually requires. Since EEAC-SV can provide the finest QoS granularity, it provides the lowest request dropping probability for each type of the flows.

In the experiment of the EEAC-SV scheme, the probing period is assumed to be 0.5 s, while 50-byte probing packets are utilized. This leads to probing overhead (the ratio of the number of bytes for probing sent from node A over the total number of bytes sent from node A) of less than 7%. More specifically, the overhead for type 1 flows is slightly higher (6.1%–6.9%) than that of the type 2 flows (6.0%–6.2%) due to the fact that more requests from type 1 flows are dropped.

The emphasis of this paper is placed in the study of the network performance under the EEAC-SV scheme and the benefit that can be achieved by the service provider in terms of request dropping probability and resource utilization. However, as mentioned before, by utilizing the EEAC-SV scheme, the user can also achieve more benefits from the network services than by utilizing other static service mapping schemes. A detailed description and evaluation of the achieved user benefits is provided in [5]. In the following, for completeness purposes, we provide a brief discussion regarding the benefits that the user can achieve from the proposed strategy. It can be noted that for IDSRA and IDDRA schemes, a user has no choice but pay for 3 units (1 unit at each router) for each data packet it sends to the network (both type 1 and type 2 flows), while the EEAC-SV can reduce the user cost by using the service vector, as compared with the IDSRA and IDDRA schemes. Our numerical evaluation demonstrates that when the requested load is 1.0, the user needs to pay 1.27 per packet for type 1 flows and 1.25 per packet for type 2 flows, under the EEAC-SV scheme. According to our utility function assumption, the reduced user cost

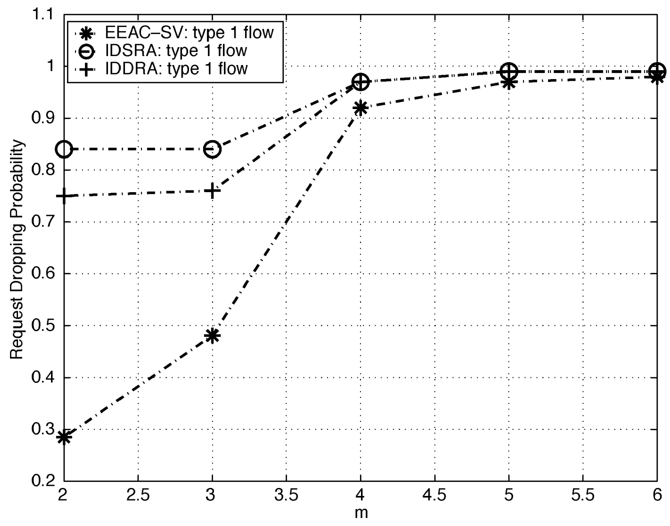


Fig. 7. Request dropping probability of flow type 1 for different values of m.

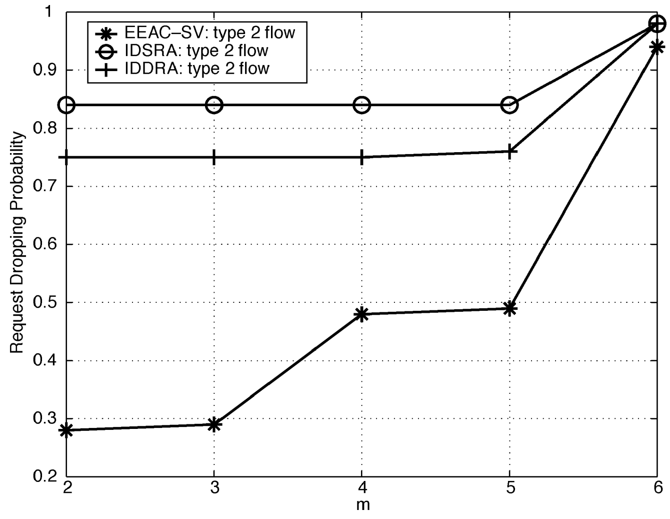


Fig. 8. Request dropping probability of flow type 2 for different values of m.

implies that user may get more benefit (i.e., larger G in (1)) from the EEAC-SV scheme, as compared with IDSRA and IDDRA schemes.

B. Experiment 2—Impact of m on the Performance of Flows With Different QoS Requirements

From the discussion in the previous section, we can conclude that larger m in principle provides finer end-to-end QoS granularity, while it may mainly benefit flows with more flexible end-to-end QoS requirements. To demonstrate and study the impact of m on the flows with different QoS requirements, in this experiment, we vary the number of routers between router 1 and 3 in Fig. 3 from 0 to 4, i.e., $m = 2, 3, 4, 5, 6$, respectively, and the same experiment is performed twice: the first time we assume that there are only type 1 flows from node A, while the second time we assume that there are only type 2 flows from node A. The corresponding request dropping probabilities of the type 1 and type 2 flows for different values of m, when the requested load is 1.0, are shown in Figs. 7 and 8, respectively.

From the results presented in Figs. 7 and 8, it can be seen that although the EEAC-SV scheme can provide smaller request dropping probability, the difference in the achievable performance among the EEAC-SV, IDSRA, and IDDRA schemes becomes smaller, as m increases. This happens because for a given end-to-end QoS requirement, as m becomes large, the flexibility of choosing different services at different routers is limited and/or saturated. As a result, there exist a threshold value of m for each type of flow, after which the achievable end-to-end performance is mainly dominated by the number of routers that a flow has to go through. In these cases, due to the stringent end-to-end requirement, the flexibility in choosing the services at each router is reduced, and most of the flows only use the EF service in order to satisfy the required QoS. The actual value of m where such saturation may be observed, depends mainly on the type of flows and whether they present stringent or flexible QoS requirements. Specifically, the more flexible the QoS requirements of a flow, the larger the corresponding threshold of m. For instance, as can be observed from Figs. 7 and 8, the corresponding values of m for the type 1 flows with strict QoS requirements, and for the type 2 flows with more flexible QoS requirements are 4 and 6, respectively. For values of m less than these thresholds, the EEAC-SV scheme significantly reduces the achievable request dropping probability for both types of the flows. However, for values of m larger than the thresholds, most of the flows using EEAC-SV scheme will choose the EF service at each router along the path, which is similar to the static service mapping under the IDSRA and IDDRA schemes.

C. Experiment 3—TCP Friendly

We expect that the proposed service provisioning architecture with the EEAC-SV scheme will be mainly used by flows with guaranteed QoS requirements, while these flows may not have end-to-end TCP-friendly congestion control mechanisms. Since the service vector mechanism will allow these flows to use the AF and BE services, which are conventionally reserved for TCP flows, it is important that the aggregated flows adopting our proposed service provisioning architecture will not affect the performance of TCP flows negatively when they use the AF or BE service. To demonstrate the TCP-friendly feature of our proposed architecture, in this experiment, we change the type 2 flows to TCP Reno flows[3] that use the AF service only from the source (node A) to the destination (node E). When the requested load is 1.0 (i.e., the heaviest load), the throughput of each type of the flows are shown in Figs. 9 and 10. It can be seen that in the EEAC-SV scheme, TCP flows and type 1 flows receive similar throughput and their performance is stable. Although, in the IDSRA scheme, the TCP flows have larger throughput due to the static resource allocation to the AF service at each node, the throughput of type 1 flows is low due to their strict QoS requirements and lack of the resource they need, which is unfair when both types of flows are heavily requesting service from the network. In the IDDRA scheme, although the throughput of both types of flows are similar, their performance changes dramatically over the time, which is an undesirable feature for the

[3]Our experiments and results demonstrated that other TCP versions, such as Tahoe or SACK, can be equally applied here without affecting the general conclusions of this paper, although the actual values may be slightly different. Therefore, in this paper, without loss of generality, we only present the results obtained from TCP Reno flows.

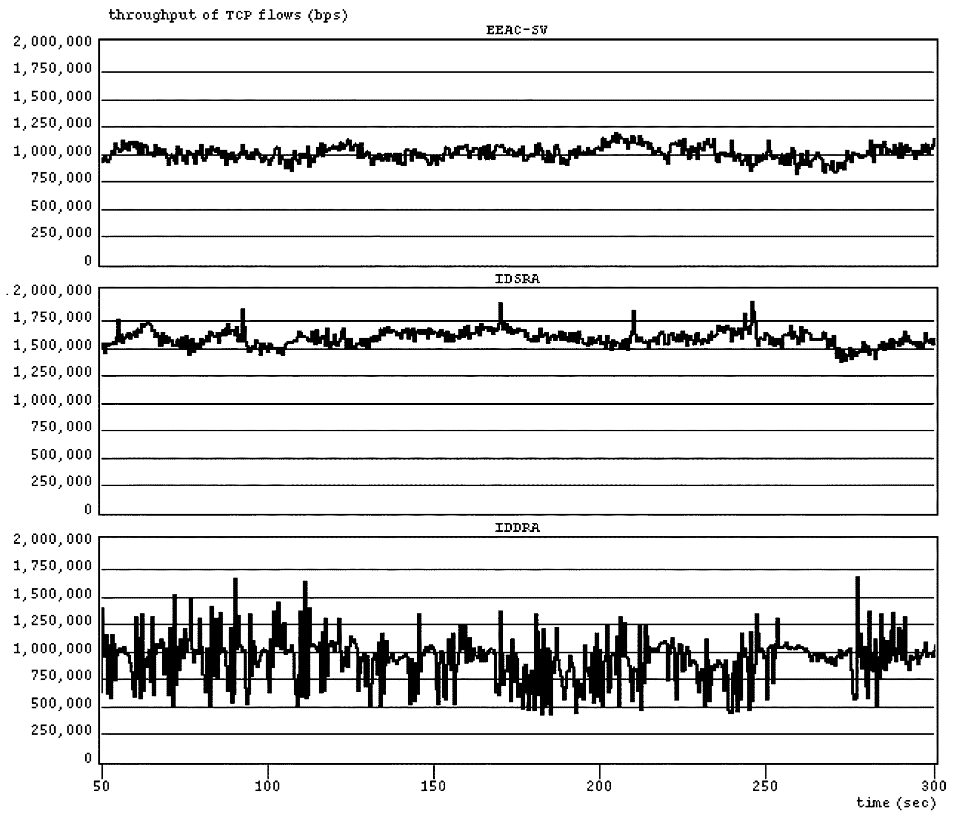


Fig. 9. Throughput of TCP flows under the requested load of 1.0.

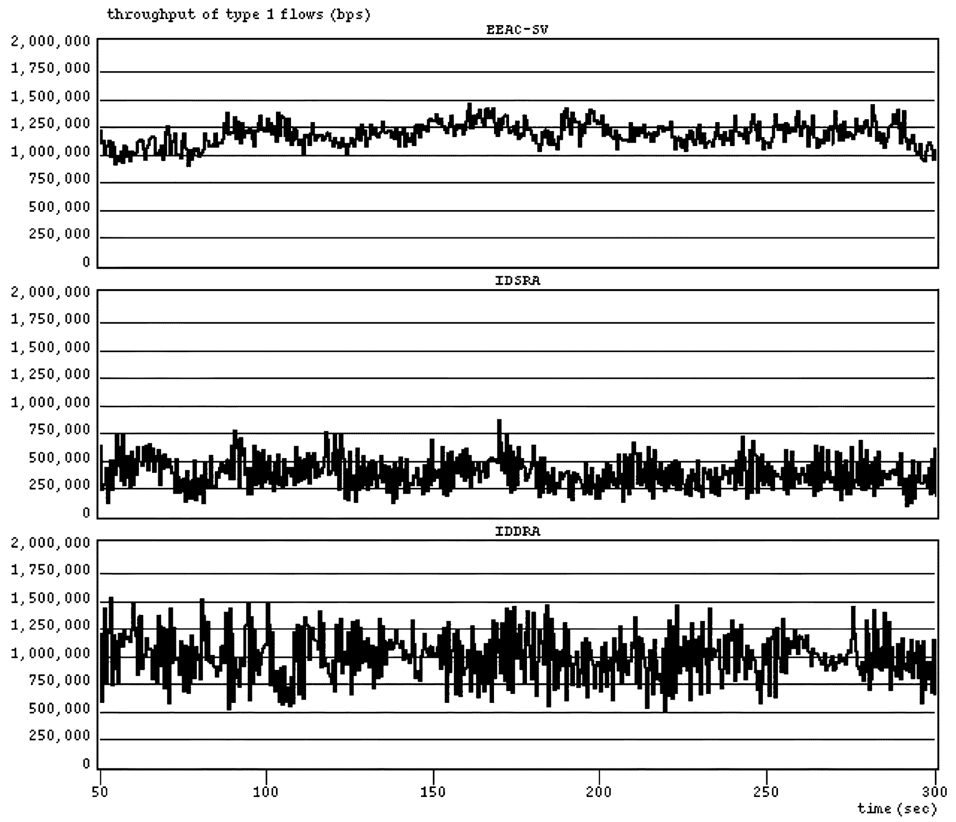


Fig. 10. Throughput of type 1 flows under the requested load of 1.0.

network. From this experiment, it can be seen that our proposed QoS provisioning architecture with the EEAC-SV scheme will provide a TCP-friendly environment in the network for TCP flows. It should be noted that in the simulation we do not assume any end-to-end congestion control mechanism for the type 1 flow. However, our probing packet marking scheme and algorithm will control the aggregated load to each service so that these flows will not use up all the network resources and starve the TCP flows.

VI. Conclusion

In this paper, we have proposed a novel distributed end-to-end QoS provisioning architecture to increase the QoS granularity provided by the DiffServ network. We have devised the EEAC-SV mechanism that enables the end hosts to choose different services at different nodes, and proposed a probing packet marking mechanism to be incorporated into the EEAC-SV scheme in order to effectively estimate the service performance at each router and reduce the probing overhead significantly. The main feature of this new QoS provisioning paradigm is that it decouples the end-to-end QoS provisioning from the service provisioning at each router, thus enhancing the end-to-end QoS granularity in the DiffServ network, while maintaining the simplicity of the service implementation at each router. It can be seen that the definitions and implementations of each service can remain the same at each core router as in the conventional DiffServ networks and, therefore, the deployment of the proposed architecture can be incremental.

By providing finer end-to-end QoS, the network can provide better services to the user, and enhance its resource utilization by reducing the request dropping probability. The numerical results presented here demonstrate that the proposed QoS provisioning architecture can have better service differentiation capability than the IntServ over DiffServ schemes via "customizing" the QoS offered to each user, reduce the request dropping probability, and provide a compatible and friendly networking environment for current TCP flows with end-to-end congestion control mechanisms.

References

[1] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," IETF, RFC 1633, Jun. 1994.

[2] S. Blake, D. Black, M. Calson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," RFC 2475, Dec. 1998.

[3] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine, "A framework for integrated services operation over DiffServ networks," RFC 2998, Nov. 2000.

[4] Cisco, *Internetworking Technology Handbook*, ch. 49.

[5] J. Yang, J. Ye, and S. Papavassiliou, "A new differentiated service model paradigm via explicit endpoint admission control," in *Proc. SCC2003*, Jun. 2003, pp. 299–304.

[6] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint admission control: Architectural issues and performance," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 69–81, Aug. 2000.

[7] F. P. Kelly, P. B. Key, and S. Zachary, "Distributed admission control," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 12, pp. 2617–2628, Dec. 2000.

[8] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 879–894, Aug. 2003.

[9] T. Timotijevic and J. Schormans, "Bandwidth overhead of probe technology guaranteeing QoS in packet networks," *Inst. Elect. Eng. Electron. Lett.*, vol. 39, no. 10, pp. 816–818, May 2003.

[10] G. Bianchi, A. Capone, and C. Petrioli, "Throughput analysis of end-to-end measurement-based admission control in IP," in *Proc. IEEE INFOCOM'00*, 2000, pp. 1461–1470.

[11] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification ECN to IP," RFC 2481, 1999.

[12] X. Wang and H. Schulzrinne, "Pricing network resources for adaptive applications in a differentiated services network," in *Proc.IEEE INFOCOM*, 2001, pp. 943–952.

[13] P. Marbach, "Priority service and max-min fairness," in *Proc. IEEE INFOCOM*, vol. 3, Oct. 2002, pp. 23–27.

[14] J. Hou, J. Yang, and S. Papavassiliou, "Integration of pricing with call admission control to meet QoS requirements in cellular networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 9, pp. 898–910, Sep. 2002.

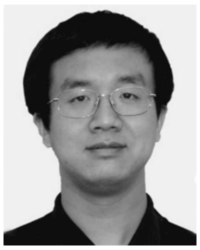
[15] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, Jun. 1993.

[16] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 33–46, Feb. 2003.

[17] S. Floyd and V. Jacobson, "Random early detection for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, pp. 397–413, Jul. 1993.

[18] G. Kesidis, J. Walrand, and C.-S. Chang, "Effective bandwidth for multiclass Markov fluids and other ATM sources," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 424–428, Aug. 1993.

[19] C. Chang, "Stability, queue length, and delay of deterministic and stochastic queueing networks," *IEEE Trans. Autom. Control*, vol. 39, no. 5, pp. 913–931, May 1994.



Jie Yang (S'99) received the B.S. degree in information engineering and the M.S. degree in communication and information systems from Xidian University, Xidian, China, in 1996 and 1999, respectively. He received the Ph.D. degree in electrical engineering from the Department of Electrical and Computer Engineering, New Jersey Institute of Technology (NJIT), Newark, 2004.

His current research interests are end-to-end QoS provisioning, admission control, resource allocation and traffic engineering, and Internet security.

Jian Ye received the B.S. and M.S. degrees in automation from the University of Electronic Science and Technology, Chengdu, China, in 1993 and 1996 respectively, and the Ph.D. degree in computer engineering from the New Jersey Institute of Technology (NJIT), Newark, in 2004.

His current research interests include network design and management, genetic algorithms, intelligent and mobile agents, and optimization of stochastic systems.

Symeon Papavassiliou (S'92–M'96) received the diploma in electrical engineering from the National Technical University of Athens, Athens, Greece, in 1990 and the M.Sc. and Ph.D. degrees in electrical engineering from Polytechnic University, Brooklyn, NY, in 1992 and 1995, respectively.

From 1995 to 1999, he was a Senior Technical Staff Member at AT&T Laboratories, Middletown, NJ. From June 1996 to August 1999, he was also an Adjunct Professor in the Electrical Engineering Department, Polytechnic University. Since August 1999, he has been an Assistant Professor in the Electrical and Computer Engineering Department, New Jersey Institute of Technology (NJIT), Newark, NJ. He has an established record of publications in his field of expertise. He is the Director of the Broadband, Mobile, and Wireless Networking Laboratory, NJIT, and one of the founding members of the New Jersey Center for Wireless Networking and Security (NJWINS). His main research interests lie in the areas of computer and communication networks with emphasis on wireless communications and high-speed networks.

Dr. Papavassiliou was awarded the Best Paper Award in INFOCOM 1994, the AT&T Division Recognition and Achievement Award in 1997, and the National Science Foundation (NSF) Career Award in 2003.

Nirwan Ansari (S'78–M'83–SM'94) received the B.S.E.E. (*summa cum laude*) from the New Jersey Institute of Technology (NJIT), Newark, in 1982, the M.S.E.E. degree from University of Michigan, Ann Arbor, in 1983, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 1988.

He joined the Department of Electrical and Computer Engineering, NJIT, as an Assistant Professor in 1988, and has been a Full Professor since 1997. He authored *Computational Intelligence for Optimization* (Norwell, MA: Kluwer, 1997) with E.S.H. Hou and translated into Chinese in 2000, and coedited *Neural Networks in Telecommunications* (Norwell, MA: Kluwer, 1994) with B. Yuhas. He is a Technical Editor of the *Journal of Computing and Information Technology* and the *ETRI Journal*. He has frequently been invited to give talks and tutorials. He has contributed over 200 publications in journals, edited books, and conferences. His current research focuses on various aspects of multimedia communications and high-speed networks.

Dr. Ansari was a distinguished speaker at the 2004 Sendai International Workshop on Internet Security and Management, and a Keynote Speaker at the IEEE/ACM cosponsored International Conference on E-Business and Telecommunication Networks (ICETE 2004). He initiated (as the General Chair) the First IEEE International Conference on Information Technology: Research and Education (ITRE 2003), was instrumental, while serving as its Chapter Chair, in rejuvenating the North Jersey Chapter of the IEEE Communications Society, which received the 1996 Chapter of the Year Award and a 2003 Chapter Achievement Award, served as the Chair of the IEEE North Jersey Section and in the IEEE Region 1 Board of Governors during 2001–2002, and currently serves in various IEEE committees. He was the 1998 recipient of the NJIT Excellence Teaching Award in Graduate Instruction, and a 1999 IEEE Region 1 Award. He is a Technical Editor of the *IEEE Communications Magazine*.