Dynamic Inter-SLA Resource Sharing in Path-Oriented Differentiated Services Networks

Yu Cheng, *Member, IEEE*, and Weihua Zhuang, *Senior Member, IEEE*

***Abstract*—This paper proposes novel resource sharing schemes for differentiated services (DiffServ) networks, to achieve both high resource utilization and quality of service (QoS) guarantee. Service level agreements (SLAs) are negotiated at network boundaries and supported by path-oriented resource mapping within the network. The recently proposed SLA management scheme based on virtual partitioning (Bouillet *et al.*, 2002) allows overloaded SLAs to exploit the spare capacity of underloaded SLAs for efficient resource utilization, however, at the the cost of possible SLA violation of the underloaders. In the *bandwidth borrowing* scheme proposed here, the dedicated bandwidth for underloaded SLAs is determined and adaptively adjusted at network boundaries according to the actual traffic load and QoS policies; the available spare capacity is then properly distributed to related links for lending to others. On the other hand, the traffic flows admitted with borrowed bandwidth are tagged and may be preempted later when the original bandwidth owner needs to claim back the resources. Through a detailed implementation design and extensive computer simulation results we show that, by bandwidth borrowing, both SLA compliance and high resource utilization can be achieved in various load conditions, with some side benefits such as call-level service differentiation, small admission overhead, and convenience for policy-based management. In addition, we propose a distributed *bandwidth pushing* scheme that can dynamically adjust the spare bandwidth distribution over the network. Combining bandwidth pushing with bandwidth borrowing, the resource utilization can be further improved.**

***Index Terms*—Bandwidth borrowing, DiffServ, inter-SLA resource sharing, resource allocation, service level agreement.**

I. Introduction

THE differentiated services (DiffServ) model [1] has been proposed as a scalable class-based traffic management mechanism to ensure Internet quality of service (QoS). In DiffServ networks, resource allocation (mainly bandwidth allocation) is based on service level agreements (SLAs) and centrally controlled by a *bandwidth broker* [2]–[4]. Neighboring administrative domains make long-term bilateral SLAs on the allocation of resources to different classes of traffic aggregate crossing the domain boundaries. Each domain is allowed to freely choose whatever mechanism it deems proper for internal resource management as long as its SLAs with neighboring domains are met. At the present time, static inter-domain SLAs are mainly used, which are negotiated based on an estimation of the average traffic volume from the up-stream domain, i.e., the *engineered traffic load*. In reality, when the actual traffic load deviates from the estimation, resources will be utilized inefficiently. The actual traffic load may be more or less than the engineered load. We use the terms "overloaded" and "underloaded", respectively, to indicate the loading status of an SLA.

DiffServ itself only defines per-hop behaviors (PHBs) at core routers to coarsely differentiate QoS, and traffic conditioning schemes at network boundaries to limit the traffic volume flowing into the network. It is widely agreed that the basic DiffServ architecture should be augmented with intelligent traffic engineering functions [5], [6] to facilitate accurate internal resource mapping, explicit per-flow admission control to guarantee QoS [7]–[9], and dynamic resource allocation to handle the traffic load variation.

The multiprotocol label switching (MPLS) technique [10], [11] provides an efficient traffic engineering tool for Internet Protocol (IP) networks, by which multiple bandwidth guaranteed label-switched paths (LSPs) can be explicitly set up between each ingress/egress node pair to balance the traffic distribution over the network. With pre-established LSPs and per-class per-ingress/egress pair SLA resource commitments, the optimal (from the perspective of network revenue) internal resource mapping can be achieved through a properly designed network planning or dimensioning procedure [12], [13]. In this paper, we consider a path-oriented DiffServ domain. The bandwidth broker memorizes the network topology and the network resource planning results. Such information is used to support explicit per-flow admission control. Each time when the bandwidth broker receives a new request forwarded from a certain ingress router, it will search for an LSP to admit the new flow according to the routing algorithm and the stored network status information. The admission decision will then be delivered back to the corresponding ingress router. If accepted, the flow related information is stored at the ingress router. It is generally agreed that the edge routers of a DiffServ network are capable enough to keep per-flow information [8], [14]. Trimintzios *et al.* present a resource management architecture for MPLS DiffServ networks in [13]. They propose solutions for operating networks in an optimal fashion through off-line planning and dimensioning, and subsequently through dynamic operations and management functions ("*first plan, then take care*"). This paper focuses on the dynamic resource allocation among SLAs sharing a properly dimensioned DiffServ domain.

Manuscript received November 5, 2003; revised February 17, 2005; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Z.-L. Zhang. This work was supported by a research grant from the Bell University Labs at the University of Waterloo.

Y. Cheng was with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. He is now with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M4S 3G4, Canada (e-mail: y.cheng@utoronto.ca).

W. Zhuang is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: wzhuang@uwaterloo.ca).

Digital Object Identifier 10.1109/TNET.2006.876199

Dynamic resource allocation can be implemented on different time scales. On the largest time scale, dynamic SLA techniques [3], [15] are proposed to adjust long-term bandwidth allocation according to the actual traffic load measured over days or weeks. On the smallest time scale, scheduling algorithms [16], [17] and measurement-based admission control [18], [19] are extensively investigated to utilize the statistical multiplexing gain at the packet-level. In the middle ground, the *virtual partitioning* (VP) technique [20], [21] has been used in circuit-switched and ATM networks to exploit the statistical multiplexing gain on the time scale of call duration. The recent work on effective bandwidth in a priority queueing system [22] and our study on effective bandwidth in a partitioned buffer [9] extend the linear form connection/call admission control (CAC)[1] to DiffServ networks, which makes the call-level QoS (call blocking probability) control possible and the VP applicable in DiffServ networks for call-level dynamic resource allocation [12], [21]. With VP, the free capacity from underloaded classes can be used by the overloaded classes to improve resource utilization, and the trunk reservation mechanism [23], [24] is used to force the overloaded classes to back off when an underloaded class needs to claim its allocated share of the capacity.

In [12], Bouillet, Mitra, and Ramakrishnan propose an SLA management architecture based on VP at each link for efficient resource utilization. The cost of VP is that the QoS of the underloaded SLAs can not be guaranteed. SLA violation for underloaders is a serious problem, which could encourage malicious overloading. Therefore, a *penalty payment* from the service provider to the customer is used in [12] to compensate the possible QoS or SLA violations. However, the penalty scheme is not a completely satisfying solution from the customers' perspective. Customers would always prefer to have guaranteed QoS as well as a fair billing system. To our best knowledge, currently there is no such resource allocation technique available that can achieve a resource utilization close to VP while guaranteeing the QoS of all SLAs involved in the resource sharing. In this paper, we propose a *dynamic inter-SLA resource sharing* scheme, also termed as a *bandwidth borrowing* scheme, for the above objective.

In the bandwidth borrowing scheme, an SLA is negotiated for each traffic class between each ingress/egress pair. Each traffic flow is allocated an effective bandwidth which encapsulates various packet level issues, such as burstiness and QoS (delay, jitter, loss) at network elements. The SLA capacity, at call level as the maximum number of calls that can be served simultaneously, is then properly contracted to satisfy customers' call level QoS requirements at an engineered call arrival rate. The accepted traffic is supported by parallel bandwidth guaranteed LSPs between each ingress/egress pair. Such path-oriented internal resource mapping is achieved by the *network dimensioning* module. During operation, if the traffic monitor finds that an SLA is in the underload status, a *protection bandwidth* smaller than its nominal capacity is calculated according to the QoS policy defined in the SLA. The protection bandwidth is guaranteed for the underloaders to satisfy their QoS requirements during the underloaded periods, and the available spare capacity is then properly distributed to related links to be borrowed by others. On the other hand, traffic flows admitted with borrowed bandwidth are tagged and may be preempted later when the original bandwidth owner needs to claim back the resource.

The methodology adopted in the bandwidth borrowing is that "boundary resource commitment determines link resource sharing," which is consistent with the SLA based management principle; the passive SLA monitoring approach taken in [12] with a predetermined link sharing configuration, however, addresses a resource mapping problem with the inverse QoS analysis approach. In this paper, we show that the newly proposed approach allows dynamic resource sharing, QoS guarantee, policy based management to be achieved simultaneously with a simple resource management architecture.

Through a detailed implementation design and extensive computer simulation results, we demonstrate that with bandwidth borrowing, 1) SLA compliance (namely, the QoS guarantee) and high resource utilization can always be achieved with various link resource sharing schemes (for example VP, complete sharing (CS),[2] or the scheme proposed in this paper) and in various load conditions; 2) per-hop signalling is avoided for a small CAC overhead; 3) policy-based resource management [13], [25] can be conveniently supported; and 4) a call-level service differentiation is achieved. Moreover, performance of the bandwidth borrowing scheme is further strengthened by a *bandwidth pushing* technique, which can dynamically adjust the distribution of the spare bandwidth over the network. As the links (where the bandwidth borrowing happens) and the resource sharing level on a certain link always dynamically change with the SLA traffic load variations, an optimal distribution of spare capacity can result in maximum resource utilization. It is very difficult, if not impossible, to derive an online, centralized optimal distribution algorithm. Therefore, the bandwidth pushing uses a distributed algorithm to adaptively push the spare bandwidth to the paths where the bandwidth borrowing can be successfully executed or where more capacity is required. The efficiency of bandwidth pushing to further improve resource utilization is also demonstrated via computer simulations.

The remainder of this paper is organized as follows. Section II describes the path-oriented DiffServ domain. Section III gives details of the SLA that will be provisioned in a DiffServ domain deploying dynamic inter-SLA resource sharing. Section IV presents the proposed bandwidth borrowing scheme. Section V and Section VI discuss the implementation design, where the proposed data structure and routing/CAC algorithm (with bandwidth pushing) for bandwidth borrowing are presented, respectively. Section VII describes some case studies and presents computer simulation results. Section VIII gives concluding remarks.

[1]The term *call* is often used in the telephone networks and ATM networks. In this paper, we still use this term for convenience. The terms *call*, *connection* and *traffic flow* are used interchangeably.

[2]Two classic schemes for resource sharing are *complete sharing (CS)*, which allows all customers to share the available resources indiscriminately, and *complete partitioning* (CP), which statically divides the resources among the customers. CP can guarantee the resource commitment for each customer, but may underutilize the resources. On the other hand, CS leads to higher resource utilization and statistical multiplexing gain, but the traffic from one customer may overwhelm all the others.

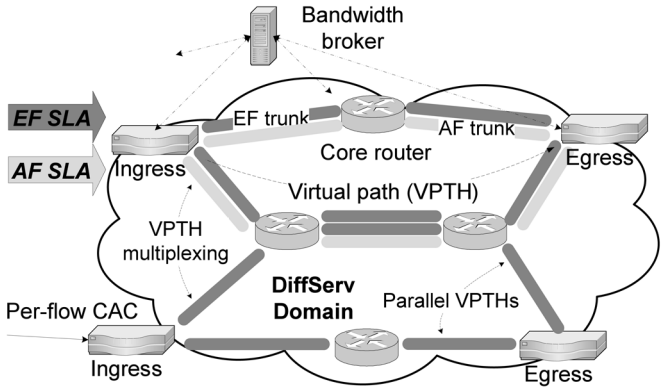

Fig. 1. Path-oriented DiffServ domain.

II. Path-Oriented DiffServ Domain

We consider a path-oriented DiffServ domain, as shown in Fig. 1, where per-class per-ingress/egress pair SLAs are negotiated at network boundaries. SLAs for the standard *premium service* and *assured service*, supported by the expedited forwarding (EF) PHB [26] and the assured forwarding (AF) PHB [27] respectively, are considered as an illustration. The bandwidth broker communicates with all the edge routers in the domain for resource allocation, CAC, and network configuration. We assume there exists an off-line routing algorithm which sets up several parallel paths for each ingress/egress pair. These paths are fixed by MPLS and referred to as virtual paths (VPTHs). All traffic traversing an ingress/egress pair is distributed among the VPTHs. VPTHs for different ingress/egress pairs may share some common links (VPTH multiplexing). An MPLS *traffic trunk* is defined as a logic pipeline within a VPTH, which is allocated a certain amount of capacity to serve a class of traffic. Therefore, a VPTH between an ingress/egress pair may include multiple traffic trunks for different service classes.

In the path-oriented environment, boundary SLA resource commitments are mapped to bandwidth allocation at each traffic trunk by network dimensioning. At each router, the total bandwidth allocation for a PHB is then derived by summing the bandwidth allocation of all the same-class trunks crossing that router. With feasible bandwidth allocation for each PHB, the specific scheduling algorithm can be designed correspondingly to guarantee the resource allocation and packet level QoS requirements. A linear programming procedure can be used for network dimensioning with the objective to maximize the network revenue [28], subject to the constraints: 1) the total bandwidth allocated to traffic trunks associated with an SLA should not be less than the SLA resource commitment; and 2) the total bandwidth allocation at a link does not exceed the physical link capacity. With per-flow CAC, the bandwidth broker puts the newly admitted traffic flow into one of the parallel traffic trunks according to the routing algorithm. All packets of one traffic flow follow the same VPTH.

In the proposed scheme, call level QoS control is decoupled from the packet level QoS control by using the effective bandwidth technique [9], [12], [22]. We assume that each traffic flow of the same service class has the same effective bandwidth for simplicity. The resource allocation to a service class can be equivalently considered in terms of the acceptable number of flows. In practice, the identical bandwidth requirement assumption may not be the case, but it can be validated by the fact that the service class can be defined with a finer granularity considering both the QoS requirements and the flow bandwidth allocation. The identical bandwidth assumption is also adopted in [12] to study the SLA management. Effective bandwidths associated with different classes are generally different.

III. Service Level Agreement: Call-Level Differentiation

This section gives details of the SLA, which is based on a call-level differentiation concept. It is realized that, in practice, an SLA may be negotiated according to very general QoS requirements and policy rules. For the sake of concreteness, we consider an SLA to provision the call-level QoS defined as follows. The call-level differentiation and bandwidth borrowing concept presented in the SLA definition can be applied to a very general scenario where a bandwidth requirement can be determined based on QoS, traffic load and management policies.

- A nominal capacity C is allocated to the SLA according to the engineered call arrival rate, also termed as the *specified rate* and denoted by λ_p, to satisfy the target call blocking probability (CBP), denoted by P_t. Let $E(\cdot)$ be the function characterizing the statistical relation of CBP versus the call arrival rate and SLA capacity. Then C is determined by $E(\lambda_p, C) = P_t$.
- During operation, at the ingress router, a call-level traffic monitor measures the actual call arrival rate for the SLA, denoted as λ_d. Two resource utilization states are defined for the SLA, which are *lendable* state if $E(\lambda_d, C) < P_u$, and *unlendable* state otherwise. P_u is the CBP requirement specified for the underloaded case where the SLA is possible to lend out bandwidth, and $P_u \leq P_t$ to provision better QoS in the underloaded period.
- In the lendable state, a *protection bandwidth* $R(< C)$ is calculated according to $E(\lambda_d, R) = P_u$. The capacity of R is reserved for the SLA. The spare bandwidth, $C - R$, can be exploited by related SLAs, including both lendable and unlendable ones, in a *complete sharing* manner.
- In the unlendable state, the nominal capacity C is guaranteed. The SLA may accept overloaded traffic, by borrowing bandwidth from the lendable SLAs. The traffic flows accepted with the borrowed bandwidth are tagged as *out* profile calls, and the flows accepted with the nominal capacity are considered as *in* profile calls.
- When the traffic monitor detects that the SLA changes back to the unlendable state from the lendable state, the protection bandwidth is then increased to the nominal capacity to claim back resources of the SLA. Some tagged traffic flows from the borrower trunks may be preempted during the bandwidth claiming.

In the above SLA definition, the possible preemption of the *out* profile calls is considered as the QoS differentiation between the *in* traffic and the *out* traffic (The *in* profile calls cannot be preempted). The counterpart differentiation scheme at the packet level is the AF PHB. The differentiation between *in* and *out* traffic efficiently utilizes the spare capacity as well as avoids the malicious overloading. The preemption scheme works well

for the data traffic, where a preempted data transfer may be resumed in the future as a new call. The bandwidth adaption proposed in [29] for multi-layer coded audio/video sources can also be used to alleviate preemption, where the extracted bandwidth from QoS degradation can be returned to the original owner or used to accept new calls.

While the SLA definition focuses on resource utilization efficiency without QoS violation, the fairness issue is not fully considered. As we can see, during operations the unlendable SLAs can occupy more bandwidth than what they buy; the SLA with a higher call arrival rate may grab more spare bandwidth according to CS than those SLAs with a lower rate. The fairness issue may be alleviated through a properly designed billing system, where the payment is proportional to the actual resource usage. To further extend the bandwidth borrowing scheme proposed in this paper for fair resource allocation is an interesting future research topic.

Note that in the SLA definition, for accuracy the *lendable* and *unlendable* are used to differentiate the SLA load status instead of the common terms *underloaded* and *overloaded*. The relationship among the four states is: the overloaded SLA is naturally in the unlendable state; the underloaded SLA may be lendable or unlendable depending on how much capacity needs to be reserved to guarantee the CBP of P_u. In the remainder of this paper, both pairs of the states are to be used, and the meanings should be clear from the context. A normally loaded SLA ($\lambda_d = \lambda_p$) is in the unlendable state according to the definition.

IV. Bandwidth Borrowing Scheme

First, we define some terminologies and symbols. When bandwidth borrowing happens, the related unlendable and lendable SLAs are termed as *borrower SLAs* and *lender SLAs*, respectively. All traffic trunks of a lendable SLA are termed as *lender trunks*. A traffic trunk belonging to a borrower SLA is termed as a *borrower trunk* only when the trunk runs out of its nominal capacity and borrows bandwidth to service traffic flows. A *spare route (path)* is a route (path) along which a flow can be successfully accepted by bandwidth borrowing. Let s denote a service class, σ an ingress/egress pair, and r a route. In the path-oriented DiffServ domain presented in Section II, we use (s, σ) to identify an SLA, (s, r) a traffic trunk, and $\mathcal{R}(s, \sigma)$ the route set or trunk set of an SLA. The nominal capacity of SLA (s, σ) is denoted by $C_{s,\sigma}$, and the bandwidth allocated to traffic trunk (s, r) (determined by the network dimensioning) is denoted by $C_{s,r}$, with $C_{s,\sigma} = \sum_{r \in \mathcal{R}(s,\sigma)} C_{s,r}$.

A. Spare Bandwidth: Calculation and Distribution

For an underloaded SLA (s, σ), the protection bandwidth R is calculated by solving

$$E(\lambda_d, R) = P_u. \tag{1}$$

With the clear context in this subsection, we omit the subscription of (s, σ) in related expressions for convenience. If $R < C$, the SLA is determined to be in the lendable state, and the *spare bandwidth* $C - R$ can be lent out. Otherwise, the SLA cannot lend bandwidth to others. For an unlendable SLA, set $R = C$ to guarantee the nominal capacity. For a lendable SLA, the protection bandwidth, correspondingly the spare bandwidth, is first distributed to each traffic trunk evenly. Let $Y_{s,\sigma}$ denote the number of trunks, and $R_{s,\sigma}$ the protection bandwidth calculated for SLA (s, σ). The protection bandwidth distributed to each trunk is $R_{s,r} = R_{s,\sigma}/Y_{s,\sigma}$. The even distribution may not be the best solution, because the traffic loads and resource sharing levels on different routes, and therefore on different links, are different. Ideally, the protection bandwidth should be distributed in such a way that leads to the maximum resource utilization. This problem will be addressed in Section VI when we discuss CAC.

The function $E(\cdot)$ for CBP calculation in general depends on the statistical characteristics of the call arrival process and the call holding time. The classic call-level modeling in telecommunication networks is the Poisson call arrival process with a mean arrival rate λ and an exponentially distributed call holding time with mean call duration $1/\mu$. The function $E(\cdot)$ is then the well known Erlang-B formula. As $E(\cdot)$ in the Erlang-B formula does not have a close-form inverse function, R can be determined according to (1) by an iterative search:

$$R = \min \{C \in \mathbb{Z} : E(\lambda_d, C) \leq P_u\}. \tag{2}$$

where $\mathbb{Z}$ is the set of non-negative integers. The R value is calculated in terms of the call number. However, in IP networks, exponentially distributed inter-arrival time or call holding time is unlikely the case [30], [31]. It is quite possible that a close-form expression of CBP is not available, and therefore R cannot be determined analytically. In this situation, the relationship $E(\lambda, C)$ can be obtained by off-line computer simulations or measurements from the historical traffic data, and recorded in a table. In online operation for bandwidth borrowing, R corresponding to a certain call arrival rate is determined by looking up the pre-established table. In fact, implementation of the bandwidth borrowing is independent of the CBP calculation details. In the following discussion, we assume a proper approach available to determine the protection bandwidth for an underloaded SLA.

The lent-out spare bandwidth is shared by trunks associated with different SLAs that can access it according to the CS scheme. As the protection bandwidth is enough to guarantee the lendable SLA a CBP of P_u, we can limit a lender trunk's access to the spare bandwidth so that more bandwidth can be exploited by borrowers. At the moment that a lender trunk successfully captures some spare bandwidth to accept a new flow, the flow will be admitted into the network with an access probability, denoted by P_a. By properly choosing the values of P_u and P_a, the service provider can control the tradeoff between statistical multiplexing gain and QoS for lender SLAs. For example, the configuration of $P_u = P_t$ and $P_a = 1$ lead to high statistical multiplexing gain according to CS; $P_u < P_t$ and $P_a = 1$ bring lenders a CBP much smaller than P_t; $P_u = P_t$ and $P_a = 0$ correspond to allocating all the spare capacity to borrowers.

B. Trunk Resource Sharing at a Link

In the bandwidth borrowing, the dynamic resource sharing is implemented at the trunk level. When a class s traffic flow arrives, a trunk (s, r) between the ingress/egress pair is selected according to a routing algorithm. The traffic trunk first tries to

admit the traffic flow using its nominal capacity or protection bandwidth depending on the SLA state; if such bandwidth is used up, the traffic trunk then tries to grab the spare capacity from the lender trunks according to the CS scheme; otherwise, the traffic flow is rejected. Let $U_{s,r}$ denote the current bandwidth usage of trunk (s,r) and $R_{s,r}$ the protection bandwidth. A route or a trunk passing link ℓ is represented as $r \supset \ell$ or $(s,r) \supset \ell$. A class-s flow with effective bandwidth e_s can be accepted at link ℓ by exploiting the spare capacity, if

$$U_{s,r} + e_s \le C_\ell - \sum_{(s',r'):\,(s',r')\neq(s,r),r'\supset\ell} \max(U_{s',r'}, R_{s',r'}) \tag{3}$$

where C_ℓ is the link capacity and $\max(U_{s',r'}, R_{s',r'})$ guarantees that the capacity of $R_{s',r'}$ is dedicated to trunk (s',r') so that bandwidth borrowing happens without SLA violation. The right-hand-side of (3) implies that the spare capacity is contributed by all the lender trunks passing link ℓ. For convenience of expression, the algorithm for trunk resource sharing at a link is referred to as the `trunkshare` algorithm.

In fact, the borrowing sharing (BS) scheme given in (3), VP and CS[3] are all special cases of the general trunk reservation scheme

$$U_{s,r} + e_s \le C_\ell - t_{res} - \sum_{(s',r'):\,(s',r')\neq(s,r),r'\supset\ell} U_{s',r'} \tag{4}$$

where the reservation parameter t_{res}, when $U_{s,r} \ge R_{s,r}$, is $\sum_{(s',r'):(s',r')\neq(s,r),r'\supset\ell} \max(0, R_{s',r'} - U_{s',r'})$ for BS, $\max_s(e_s)$ for VP [12], and 0 for CS; when $U_{s,r} < R_{s,r}$, $t_{res} = 0$ in all the three cases. Among the three schemes, BS is normally the most conservative scheme with the largest trunk reservation[4] for QoS guarantee; CS is a greedy scheme for high resource utilization, while the underloaders may be overwhelmed by overloaders; VP is a trade-off approach where the overwhelming can be alleviated but not eliminated. With the call preemption approach as detailed in Section VI-A, the dilemma between high resource utilization and QoS guarantee can be successfully solved; VP and CS can also be used in the `trunkshare` algorithm for higher resource utilization. Any aggressive resource usage from the borrowers, which invades the protection bandwidth reservations of other SLAs, will then be preempted by the original owners when necessary, so that the aggressiveness does not lead to SLA violation.

C. Bandwidth Borrowing Along a Path

An out profile flow can be accepted only when bandwidth borrowing via `trunkshare` is successful at all links along the selected path. For a new out profile request associated with an unlendable SLA (s,σ), the borrowing procedure is as follows. The bandwidth broker begins from the first hop link on each VPTH $r \in \mathcal{R}(s,\sigma)$. If `trunkshare` can admit the class s flow in this link, the bandwidth broker steps forward to the second hop link. If the `trunkshare` rejects the flow at any link along a VPTH, this VPTH is denied. After all the links along all VPTHs are checked, we can get a *lendable route set*. All links along a lendable route r can lend bandwidth to the borrower trunk (s,r). The leftover bandwidth along the lendable route r is $\min_{\ell:\ell\in r}(C_\ell - t_{res} - \sum_{(s',r'):r'\supset\ell} U_{s',r'}) \ge e_s$, where the specific value of t_{res} depends on whether BS, VP or CS is used for resource sharing. From the lendable route set, the route with the largest lendable bandwidth (the random selection principle will be used when multiple such routes exist) will be selected to hold the new traffic flow. Such an approach is taken to protect the out profile call from future preemption as much as possible. In the case that all the routes do not have enough bandwidth to admit the flow, the bandwidth broker rejects the flow. The above admission procedure is also applied to an lendable SLA when its protection bandwidth is used up, where the admitted flow is still in profile as long as $R_{s,\sigma} < U_{s,\sigma} \le C_{s,\sigma}$.

It is noteworthy that the hop by hop checking of resource availability here is not through signaling, but through looking up route table and resource usage information stored in the bandwidth broker (to be explained in the next section). Hence, there is no scalability problem and the CAC overhead time is expected to be small. The `trunkshare` checking at each link and the route selecting procedure are summarized as a `sparest route` subroutine process to be used in the CAC procedure, which returns the selected route r, or -1 when no lendable route exists.

V. Data Structure

A. Data in the Bandwidth Broker

For bandwidth borrowing, all the information stored in the bandwidth broker is organized into three tables: **Route Table**, **Trunk Status Table**, and **SLA Status Table**.

Route Table: The topology and routing information is organized into a route table. Assume that each traffic trunk and each link is assigned a unique ID in the DiffServ domain. Each row and each column of the route table is indexed with the traffic trunk ID and the link ID, respectively. Searching along a row, we can find all the links of a traffic trunk. Searching along a column, we can get all the trunks crossing a certain link.

Trunk Status Table: The network planning results, current network resource usages, and bandwidth borrowing information are organized into the Traffic Trunk Status Table. Each record in the table is indexed with the traffic trunk ID and has four items: Traffic Trunk Capacity $C_{s,r}$ allocated by the network planning, Traffic Trunk Usage $U_{s,r}$ which is updated with call arrival and completion, Trunk Protection Bandwidth $R_{s,r}$ (For the trunks associated with an unlendable SLA, $R_{s,r} = C_{s,r}$; for those with a lendable SLA, $R_{s,r} = R_{s,\sigma}/Y_{s,\sigma}$ is initially set, and may be dynamically adjusted during the bandwidth borrowing), and Trunk Utilization Status (TUS) Flag. The TUS flag indicating the current trunk utilization status. Three states are possible, namely, *notfull*, *full*, and *borrowing*. When $U_{s,r} < R_{s,r}$, TUS flag is set as *notfull*; when $R_{s,r} \le U_{s,r} \le C_{s,r}$, flag as *full*; when $U_{s,r} > C_{s,r}$, flag as *borrowing*.

SLA Status Table: The call-level traffic monitor inside the edge routers measures the call arrival rate for each SLA, and

[3]Note that in bandwidth borrowing, the CS scheme is adopted to exploit the spare capacity $(C - R)$ on each link. When we compare different link resource sharing schemes (BS, VP, and CS), CS is then referred to as one approach for sharing the *whole* link capacity.

[4]It is obvious that examples where $\sum_{(s',r'):(s',r')\neq(s,r),r'\supset\ell} \max(0, R_{s',r'} - U_{s',r'}) < \max_s(e_s)$ can be easily constructed. However, in a multiclass network with multiple lenders, a separate resource reservation for each lender is normally more conservative than the single reservation of $\max(e_s)$ used in VP.

forwards the measurement results to the bandwidth broker when the rate changes (according to a predefined threshold, for example, increase or decrease by 10%). The bandwidth broker will then update the SLA status and the protection bandwidth R. The SLA Status Table is just a simple array indexed with the SLA ID, where each element indicates the status of an SLA, *lendable* or *unlendable*.

B. Data in the Edge Routers

Flow Record Table: To support per-flow CAC, the ingress routers memorize the per-flow information in the Flow Record Table. The information is organized and saved according to the traffic trunks where the traffic flows are placed.

Flow Information Record: A flow information record includes the Traffic Flow ID, the Effective Bandwidth allocated to the flow, and the Flow Admission Status (FAS) Flag. The FAS flag indicates what bandwidth is used to admit the traffic flow, which is set as *nor-admit* (normally admitted) if the traffic flow is admitted into a *notfull* trunk by the nominal or protection bandwidth, and set as *bor-admit* (admitted with borrowed bandwidth) if admitted through bandwidth borrowing. A *bor-admit* flow is considered as an *out* profile flow.

VI. Routing, CAC, and Bandwidth Pushing

The flowchart of the proposed routing/CAC algorithm for a class-s traffic flow is given in Fig. 2, which has the following procedure:

1) The dedicated bandwidth $R_{s,\sigma}$ available in the *notfull* trunks is always used first to accept the traffic flow. The random trunk selection is adopted for load balancing over the network. Upon a *nor-admit* admission, the need of bandwidth claiming is checked. One reason is that the SLA under consideration may just increase its dedicated protection bandwidth due to the increase of traffic load, where the former spare capacity (but not available now) may have been borrowed by other SLAs and should be claimed back if itself needs to use it. The other reason is that VP or CS based `trunkshare` may lead to aggressive bandwidth usage. The steps to claim back bandwidth are summarized in the subroutine `claimbackband`.
2) When the dedicated bandwidth is used up, that is, no *notfull* trunk is available, the inter-SLA sharing is evoked. The subroutine `sparestroute` is called to search for the lendable route $r \in \mathcal{R}(s,\sigma)$ with the largest leftover bandwidth. If no lendable route exists, the flow is rejected.
3) For an *unlendable* SLA, the inter-SLA sharing is a borrowing process. The FAS flag is set as *bor-admit*.
4) For a *lendable* SLA, the inter-SLA sharing is for better QoS. Here, the lender trunk is not allowed to go to the *borrowing* state ($U_{s,r} > C_{s,r}$) for a stable resource management scheme. The access probability, P_a, is used to control the extent that the lender trunk accesses the spare capacity. The FAS flag for an admitted flow is set as *nor-admit*.
5) Upon each admission via the inter-SLA resource sharing, the subroutine `pushprotband` is called to dynamically adjust the protection bandwidth distribution among related lender trunks, so that the spare bandwidth can concentrate on the routes where it can be more efficiently utilized.

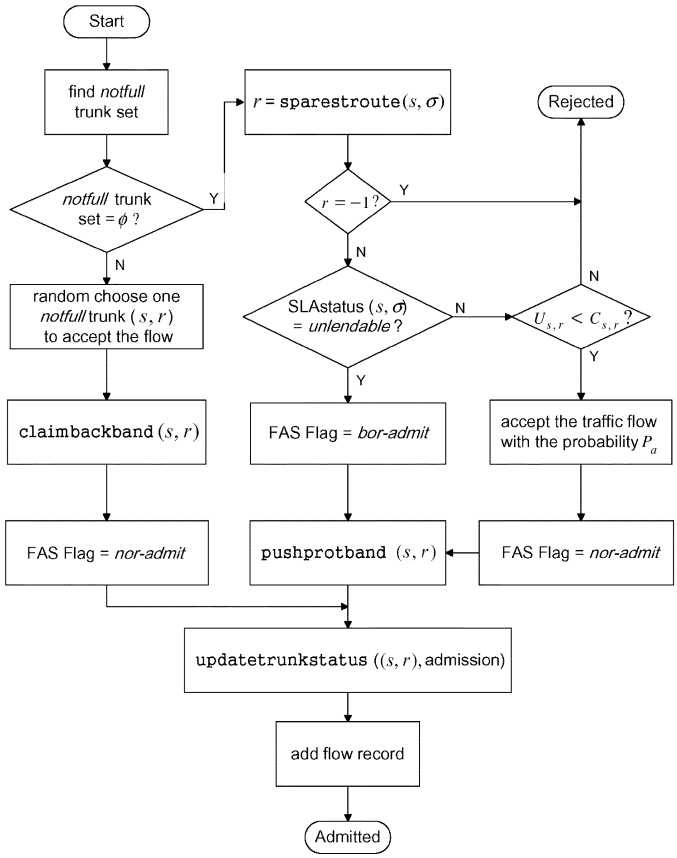

Fig. 2. Routing and admission procedure with bandwidth borrowing and bandwidth pushing.

6) If the traffic flow is accepted, the trunk status is then updated by the subroutine `updatetrunkstatus`. Specifically, $U_{s,r} = U_{s,r} + e_s$, and the TUS flag is updated correspondingly. Also, the flow information is added to the flow record table at the corresponding ingress router.

In the above CAC procedure, several subroutines are called. Among them, the `sparestroute` subroutine is described in Section IV-C. The `updatetrunkstatus` ((s,r), event) subroutine is to update the status of traffic trunk (s,r) for the specified event. The specified event may be admission of a call, completion of a call, preemption of a call, or adjustment of the trunk protection bandwidth. An event leads to the change of R or U in the specified traffic trunk, and the TUS flag is then updated depending on the current relationship among R, U and C of the trunk. The details of the `claimbackband` and `pushprotband` subroutines are given in the following subsections.

A. Claiming Back Bandwidth

In the `claimbackband` subroutine, if the new request is to be admitted in the *notfull* trunk (s,r), the leftover bandwidth is calculated as $C_\ell - \sum_{(s',r'):r' \supset \ell} U_{s',r'}$ on each link that the trunk passes.[5] On any link, a leftover bandwidth less than e_s means that the link is an *overused* link, where borrower trunks exist and trunk (s,r) needs to claim back its lent-out bandwidth by preemption. On each overused link, the associated borrower

[5]Note that the calculation of $C_\ell - \sum_{(s',r'):r' \supset \ell} U_{s',r'}$ for preemption checking, other than $C_\ell - t_{res} - \sum_{(s',r'):r' \supset \ell} U_{s',r'}$ for admission control, can fully exploit the statistical multiplexing to avoid unnecessary preemption.

trunk set is searched, from which certain *bor-admitted* traffic flows are preempted to release enough bandwidth to hold the new request. The pseudocode of the subroutine `claimbackband` is as follows:

```
subroutine name: claimbackband
Input: ID of trunk (s, r) holding the new flow
Return: no return value
linkset = all the links along route r;
for i = 1: size(linkset) %foreachlink
    leftband = leftover bandwidth on the ith
    link;
    while leftband< e_s %preemptionrequired
      borrowertrunkset = all the borrower
      trunks passing the ith link;
      select one borrower trunk, (s_b, r_b), from
      the borrowertrunkset with a probability
      that is proportional to the borrowed
      bandwidth;
      preempt the newest bor-admitted traffic
      flow in (s_b, r_b);
      leftband = leftband + e_{s_b};
      updatetrunkstatus ((s_b, r_b), flow
      preemption);
    end
end
```

When bandwidth borrowing is implemented based on the BS trunk sharing scheme, the traffic flow preemption does not have a severe impact on the long term call-level QoS, due to three reasons: 1) The preemption happens only during the short period just after a certain lender trunk increases its protection bandwidth. 2) When the protection bandwidth of a lender trunk increases, the admission rate of new *bor-admitted* flows reduces immediately due to the decrease of borrowable bandwidth, which can speed up the bandwidth returning. 3) At each link, unused bandwidth in all trunks is fully exploited to avoid preemption. But with VP or CS trunk sharing scheme, the aggressive flow admission may result in an unneglectable preemption probability to those out profile calls. Preemptions brought by the BS, VP, and CS trunk sharing schemes are compared in Section VII.

Call preemption should be considered as a type of cost in the resource sharing. It may not be a problem for non-realtime data application, but may be annoying in realtime video/audio applications. Based on the proposed CAC algorithm, a message can be sent to the customer before the actual service regarding the SLA load status and flow admission status. The customer can then determine to continue or try at a later time. With customers' awareness of the status, the resource sharing with preemption is expected to be a reasonable service model.

B. Dynamic Bandwidth Pushing

The even distribution of spare bandwidth (and protection bandwidth correspondingly) over lender trunks is not the most efficient approach. For example, a lender SLA has two parallel lender trunks, but borrowers associated with one lender trunk are only slightly overloaded while heavily overloaded with the other one. In this case, most of the SLA spare bandwidth should be distributed to the heavily overloaded route for higher resource utilization. Generally, the spare bandwidth distribution should be properly determined and dynamically adjusted according to the network status for maximum resource utilization. It is very difficult, if not impossible, to derive a centralized, optimal on-line distribution technique. Therefore, we propose a distributed *bandwidth pushing* scheme to approximate the optimal distribution, which is summarized in the subroutine `pushprotband`. The pseudocode of `pushprotband` is as follows:

```
subroutine name: pushprotband
Input: ID of trunk (s, r) holding the new flow
Return: no return value
linkset = all the links along route r;
pushedtrunks= 0; %storealreadyprocessedtrunks
for i = 1: size(linkset) % ateachlink
    lendertrunkset = all the lender trunks
    passing the ith link;
    for j = 1: size(lendertrunkset)
      if (lendertrunkset (j) ∉ pushedtrunks &
      R_{lendertrunkset(j)} > B_push)
        (s_j, σ_j) = SLA that lendertrunkset(j)
        belongs to;
        from (s_j, σ_j)'s pushable trunk set,
        randomly choose one trunk, x;
        R_{lendertrunkset(j)} = R_{lendertrunkset(j)} − B_push;
        R_x = R_x + B_push;
        % updating trunk status due to new R
        updatetrunkstatus (lendertrunkset(j), new
        R_{lendertrunkset(j)});
        updatetrunkstatus (x, new R_x);
        add lendertrunkset(j) to pushedtrunks;
      end
    end
end
```

In the bandwidth pushing scheme, an adjustment of the protection bandwidth distribution is triggered each time a new traffic flow is put into a full trunk (s,r), according to the CAC procedure given in Fig. 2. To do the adjustment, in `pushprotband` the lender trunk set is searched at each link along route r, and the detected lender trunks are called *lend-on* trunks for expression convenience. Each *lend-on* trunk will then try to push some of the protection bandwidth to its fellow lender trunks, referred to as *push-to* trunks, belonging to the same lender SLA, so that the spare bandwidth can concentrate on the trunks where bandwidth borrowing is taking place. Let e_{lender} denote the effective bandwidth associated with a lender SLA. The amount of the protection bandwidth to be pushed is $B_{\text{push}} = \lceil e_s/e_{\text{lender}} \rceil e_{\text{lender}}$, to reserve more bandwidth for future arrivals possibly with the effective bandwidth e_s

(which is estimated from the previous arrivals). A push-to trunk $x = (s_{\text{lender}}, r_{\text{lender}})$ is a pushable trunk only when conditions

$$R_x + B_{\text{push}} \le C_x \tag{5}$$

$$\min_{\ell:\ell\in r_{\text{lender}}} \left(C_\ell - \max(U_x, R_x + B_{\text{push}}) - \sum_{(s',r'):(s',r')\neq x, r'\supset \ell} \max(U_{s',r'}, R_{s',r'}) \right) \ge \max_s(e_s) \tag{6}$$

are satisfied. Condition (5) indicates that the protection bandwidth can never exceed the nominal allocation. Condition (6) means that over-pushing should be avoided so as not to block the future bandwidth borrowing along the push-to paths. In other words, the bandwidth pushing should not change the "spare property" of other paths. Within a lender SLA, all pushable trunks form a *pushable trunk set*, and one of them is randomly picked out to accept the pushed-in protection bandwidth. The *lend-on* trunk can push away its protection bandwidth down to zero.

It should be noted that the bandwidth pushing is operated in the whole network in a distributed manner. Each spare route tries to push the protection bandwidth to other routes but with different pushing force. With the proposed pushing algorithm, those routes with a heavier traffic load generate stronger pushing force and obtain a larger part of the spare capacity in the pushing competition. At the same time, the over-pushing is limited by (6) and each spare route has a chance to be the winner of the pushing competition when the traffic load changes dynamically. The simulation results in Section VII demonstrate that the bandwidth borrowing scheme can always be enhanced, independent of the link resource sharing schemes (BS, VP, or CS), by the bandwidth pushing technique with a higher resource utilization as compared with the no-pushing case.

C. Processing for Call Completion and Preemption

The data processing for call completion is simple. One operation is to delete the flow information from the flow record table in the corresponding ingress router. Another is to update the status of the trunk where the traffic flow resides. When a traffic flow completes on the trunk (s, r), $U_{s,r} = U_{s,r} - e_s$ and TUS flag is updated accordingly by subroutine `updatetrunkstatus` ((s, r), completion). The processing for a call preemption is the same as that for a call completion.

VII. Simulation Results

This section presents the computer simulation results from some case studies to demonstrate the efficiency of the bandwidth borrowing, dynamic bandwidth pushing, and QoS guarantee with preemption. For convenience of calculating the protection bandwidth, we assume Poisson arrivals for each SLA and exponentially distributed call holding times in all simulations. Also, the network dimensioning procedure is simplified by tailoring the link capacity to exactly hold all the traffic trunks crossing the link. This approach provides us the flexibility to configure networks. The units used for related measures are time unit (t-unit) for time, capacity unit (c-unit) for link/trunk/SLA capacity (it will be explicitly stated, when the call level capacity is used), and efficient bandwidth usage, call/t-unit for call arrival rate and call level throughput, and c-unit/call for effective bandwidth.

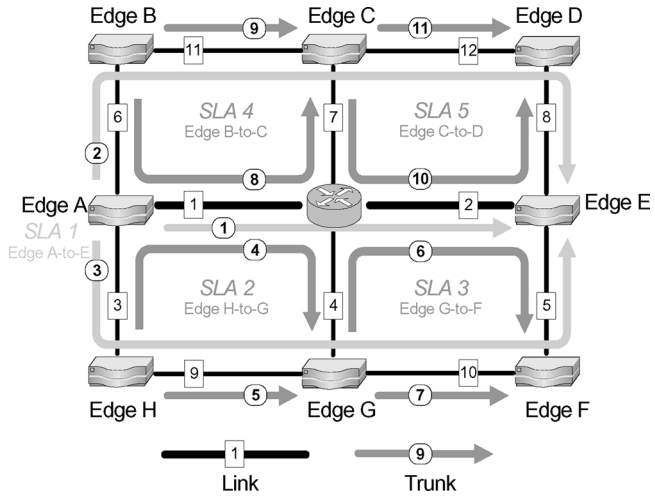

Fig. 3. Network topology, SLAs, and trunk deployment for bandwidth borrowing.

TABLE I
SLA QoS Requirements and Network Capacity Planning Results

SLA	Ingress/egress	Traffic trunks	λ_p	C_{SLA}	P_t
1	Edge A/E	1, 2, and 3	46.9	60	10^{-2}
2	Edge H/G	4 and 5	29	40	10^{-2}
3	Edge G/F	6 and 7	29	40	10^{-2}
4	Edge B/C	8 and 9	29	40	10^{-2}
5	Edge C/D	10 and 11	29	40	10^{-2}

A. Operation and Performance of Bandwidth Borrowing

The network for this simulation study is shown in Fig. 3. We use three examples to illustrate various aspects of the proposed resource sharing schemes.

Example 1: Resource Utilization Improvement: Five SLAs are negotiated and served with parallel traffic trunks, as shown in Fig. 3. SLA QoS requirements, engineered call arrival rates and network capacity planning results are given in Table I. The SLA capacity is evenly distributed over the related traffic trunks. A homogeneous case is considered, where the effective bandwidth associated with each SLA equals to 1, and the arrivals in each SLA have an average call holding time of 1. The link capacity is 60 for links 1 and 2, and 40 for other links. The BS trunk resource sharing scheme is used for bandwidth borrowing.

The simulation starts at $t = 0$, and ends at 48000. Traffic for each SLA starts with the specified call arrival rate, and the call arrival rates for some SLAs are changed at certain moments to create the overloaded and underloaded periods. We assume that the call-level traffic monitor can detect the rate variation timely and accurately, as the operations of the bandwidth borrowing are independent of the specific measurement methods [12], [32]. The actual call arrival rate for each SLA and the corresponding protection bandwidth for each trunk are given in Table II. $P_u = P_t$ and $P_a = 1$ are set for bandwidth borrowing. The measured call blocking probability and call throughput for

TABLE II
CALL ARRIVAL RATE AND PROTECTION BANDWIDTH FOR EACH SLA

t	λ_d^1	(R_1, R_2, R_3)	λ_d^2	(R_4, R_5)	λ_d^3	(R_6, R_7)	λ_d^4	(R_8, R_9)	λ_d^5	(R_{10}, R_{11})
0	46.9	(20,20,20)	29	(20,20)	29	(20,20)	29	(20,20)	29	(20,20)
6000	62.6	(20,20,20)	29	(20,20)	29	(20,20)	29	(20,20)	29	(20,20)
12000	62.6	(20,20,20)	14.4	(11,12)	29	(20,20)	29	(20,20)	14.4	(12,11)
36000	62.6	(20,20,20)	14.4	(11,12)	29	(20,20)	29	(20,20)	29	(20,20)

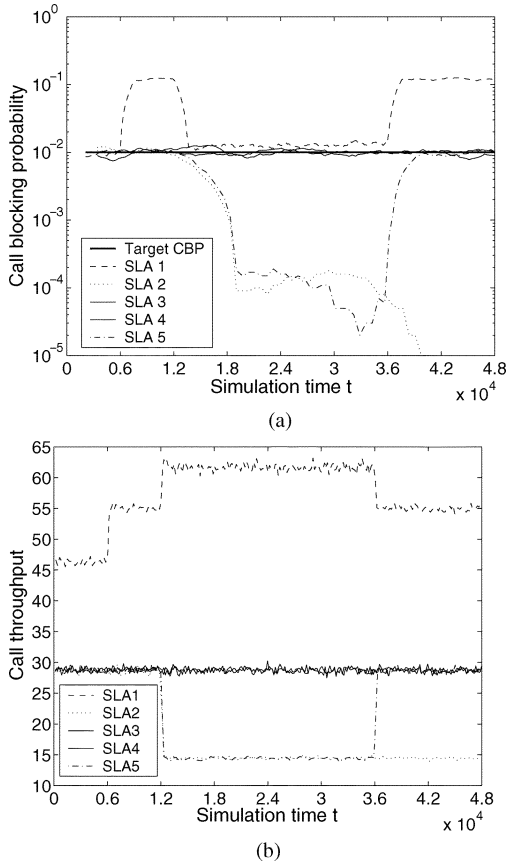

Fig. 4. Performance with bandwidth borrowing. (a) The call blocking probability of each SLA. (b) The call throughput of each SLA.

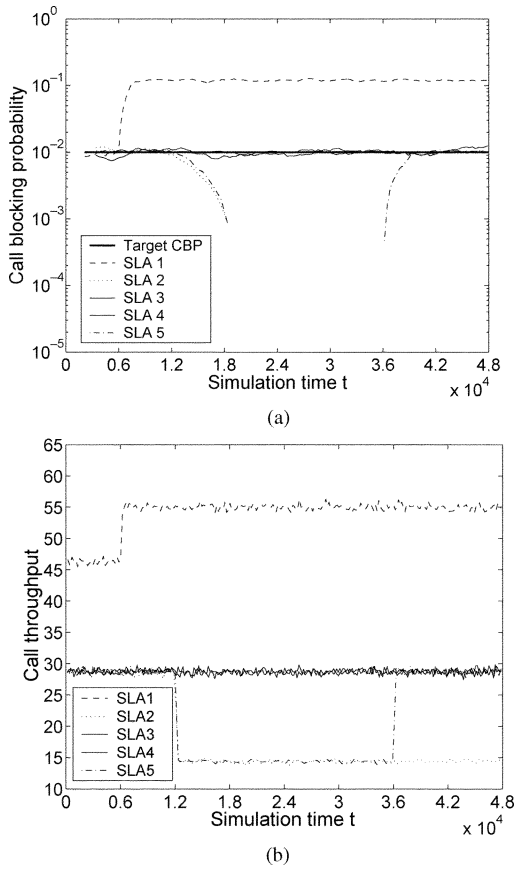

Fig. 5. Performance without bandwidth borrowing. (a) The call blocking probability of each SLA. (b) The call throughput of each SLA.

each SLA with bandwidth borrowing are presented in Fig. 4(a) and (b), respectively, and those without bandwidth borrowing are presented in Fig. 5(a) and (b), respectively. To estimate the throughput, each 10000 inter-arrival times are averaged and then taken the reciprocal to get a throughput sample. To estimate the CBP, a slide-window algorithm is used. Each 10000 arrivals compose a slide period, and 10 slide periods compose a window. The number of lost calls in one window is used to get an CBP sample. After one CBP sample is obtained, the window then shifts forward by one slide.

From Table II and Figs. 4 and 5, we have the following observations:

1) Each SLA starts with the engineered rate and achieves the target CBP, without inter-SLA resource sharing. After $t = 6000$, SLA-1 becomes overloaded. The bandwidth borrowing does not happen until $t = 12000$ when SLA-2 and SLA-5 become underloaded. During the time period of (6000, 12 000), SLA-1 has the CBP of $E(62.6, 60) \approx 0.1208$, and other SLAs continue with the specified rate and achieve the target QoS.
2) During the time period (12000, 36000), SLA-2 and SLA-5 become underloaded, where both their protection bandwidths are calculated as 23 and first evenly distributed between the two parallel trunks as 11 and 12 (fractional capacity unit not allowed). With bandwidth borrowing, SLA-1 can utilize the spare capacity from SLA-2 and SLA-5 along trunk-1, but not along trunk-2 and trunk-3, because there is no spare bandwidth on link-6, link-11, link-10 and link-5. Therefore, trunk-1 is the borrower trunk, trunk-4 and trunk-10 the *lend-on* trunks, and trunk-5 and trunk-11 the *push-to* trunks. The `pushprotband` procedure then pushes protection bandwidth from trunk-4 to trunk-5 for SLA-2 and from trunk-10 to trunk-11 for SLA-5, until the protection bandwidth on trunk-5 and trunk-11 reaches 19, with 1 unit of spare bandwidth reserved to maintain their "spare" property. In the steady state, the spare capacity on both trunk-4 and trunk-10 is then $20 - (23 - 19) = 16$. The spare capacity of 16 on link-1 is shared between SLA-1 and SLA-2 according to the CS principle, and on link-2 between SLA-1 and

SLA-5. SLA-1 grabs almost all the spare capacity due to the large call arrival rate to achieve a CBP approximately of $E(62.6, 60 + 16) \approx 0.0125$. The simulation results match very well with this numerical estimations. The spare bandwidth of 1 unit on link-9 and link-12 can only be accessed by SLA-2 and SLA-5, respectively. In addition to having some spare capacity grabbed from trunk-4 and trunk-10, both SLA-2 and SLA-5 can achieve a CBP much smaller than $E(14.4, 23 + 1) \approx 5.7 \times 10^{-3}$, which are also demonstrated by the simulation results.

3) After $t = 36000$, SLA-5 changes back to the *unlendable* state, and both protection bandwidths of trunk-10 and trunk-11 are then reset to 20 to claim back the lent-out bandwidth. Since the spare capacity on link-2 is not available anymore, the bandwidth borrowing along trunk-1 stops. However, in the simulation, we did not observe preemptions due to the protection provided by the statistical multiplexing as explained in Section VI-A. When the bandwidth borrowing stops, SLA-2 exclusively utilizes the spare capacity, achieves a CBP of $E(14.4, 40) \approx 1.48 \times 10^{-8}$.
4) The throughput simulation result of each SLA is consistent with the call arrival rate and the achieved CBP. For the CAC without bandwidth borrowing, each SLA always exclusively uses its nominal capacity. The difference between the borrowing and non-borrowing scenarios exists during (12000, 36000), where the underloaded SLA-2 and SLA-5 in the latter achieve the CBP of near 0, but the throughput only increases approximately from 14.3 to 14.4 as compared with the former. On the other hand, the bandwidth borrowing can trade the slightly (almost unnoticeably) degraded QoS of underloaded SLAs for an obvious throughput increase in the overloaded SLA-1, approximately from 55.0 to 61.8.

Example 2: Dynamic Bandwidth Pushing: In this example, we show that the bandwidth pushing can adaptively adjust the spare bandwidth distribution in the network when traffic load changes. Also, the heterogeneous effective bandwidth allocation and the effect of P_a control are demonstrated. We basically use the same network dimensioning and configurations as those used in Example 1, with the following changes:

- An SLA-6 between ingress G and egress E is added. SLA-6 is supported by trunk 12 passing link-10 and link-5. The engineered call arrival rate for SLA-6 is also 29, with average call duration of 1. The nominal capacity of trunk-12 is 40 calls to guarantee a CBP of 0.01.
- Flow effective bandwidth is 2 for SLA-1 and 1 for other SLAs. Link capacity is adjusted to keep the network well dimensioned, where $C_{\ell 1} = C_{\ell 2} = 80$, $C_{\ell 4} = C_{\ell 7} = 40$, $C_{\ell 5} = C_{\ell 10} = 100$, and 60 for the other links.
- When SLA-2 becomes underloaded, the initial protection bandwidth is set as $(R_4, R_5) = (17, 6)$.
- SLA-5 changes back to the *unlendable* state at $t = 24\,000$. SLA-6 goes to underloaded at $t = 36\,000$ with $\lambda_d^6 = 14.4$, and $R_{12} = 23$.
- $P_a = 0.95$.

The simulation starts at $t = 0$ and ends at 48000. The measured CBP for each SLA is shown in Fig. 6(a). As SLA-4 is not involved in bandwidth borrowing, it is not shown in the figure.

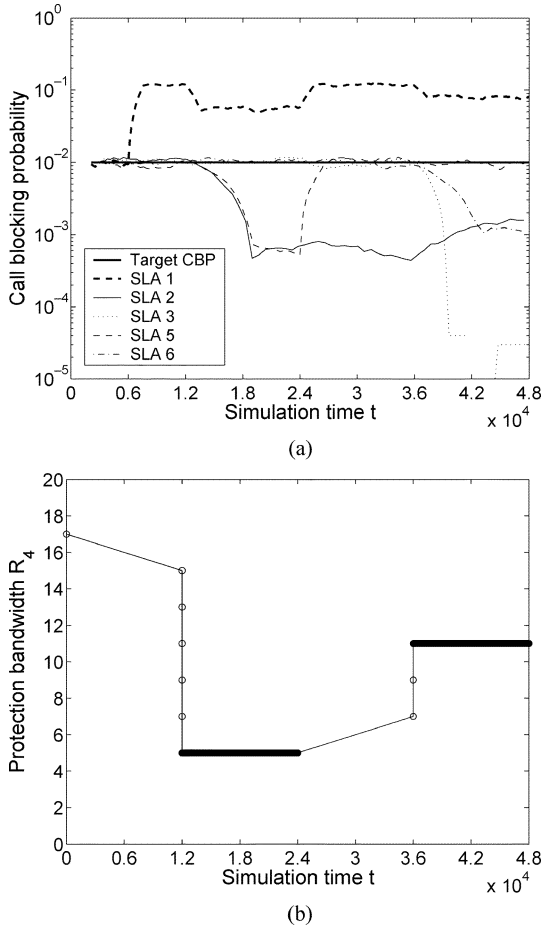

Fig. 6. Performance of dynamic bandwidth pushing. (a) The call blocking probability of each SLA. (b) Dynamic adjustment of the protection bandwidth.

During the time period (0, 12000), we have the same observations as those in Example 1. During (12000, 24000), bandwidth borrowing happens between SLA-1, SLA-2 and SLA-5, and the protection bandwidth distribution is adjusted to $(R_4, R_5) = (5, 18)$ and $(R_{10}, R_{11}) = (5, 18)$ by bandwidth pushing, independent of the initial distribution. Along trunk-5 and trunk-11, $\max(e_s) = 2$ is reserved to keep their "spare" property. Along trunk-1, only 14 out of the 15 spare units is accessible to SLA-1 due to the discreteness. Therefore, the CBP of SLA-1 can approximately goes back to $E(62.6, 60 + 14/2) \approx 0.0568$ as shown in the figure. During this period, the lenders SLA-2 and SLA-5 have a CBP with the order of 10^{-3}, larger than the order of 10^{-4} in Example 1, due to the spare capacity access control with $P_a = 0.95$. During the interval (24 000, 36000), the bandwidth borrowing stops as SLA-5 claims back its spare capacity. In this period, although SLA-2 can exclusively access its spare capacity, its CBP does not decrease obviously, again due to the P_a control.

Interesting things happen during (36000, 48000) when SLA-6 becomes underloaded. In this period, SLA-2 and SLA-6 are lenders, and the SLA-1 out profile calls can be accepted along trunk-3. Between SLA-1 and SLA-2, the borrower trunk-3 needs to borrow bandwidth from both lenders trunk-4 and trunk-5. It is obvious that in this case an even distribution is the best choice of the SLA-2's spare capacity for maximum resource utilization. In simulation, the bandwidth pushing does

TABLE III
PERFORMANCE RELATED TO TRUNK RESOURCE SHARING, BANDWIDTH PUSHING AND FLOW PREEMPTION

Configurations				SLA-1	SLA-2	SLA-3	SLA-4	SLA-5	EBU
CP	No pushing	No Preempt	CBP	0.3769	1.4758e-8	0.0100	0.0100	1.4758e-8	144.6668
BS	No pushing	No preempt	CBP	0.3457	0.0006	0.0106	0.0102	0.0006	147.5547
	Pushing	No preempt	CBP	0.2251	0.0017	0.0106	0.0102	0.0015	158.8393
VP	No pushing	No preempt	CBP	0.1718	0.0356	0.0688	0.0690	0.0367	159.4450
		Preempt	CBP	0.1689	0.0014	0.0104	0.0097	0.0013	N/A
			OCPP	0.0652	N/A	0.0647	0.0684	N/A	N/A
			CBPP	0.2123	0.0014	0.0105	0.0098	0.0013	160.0647
	Pushing	No preempt	CBP	0.1334	0.0137	0.0486	0.0502	0.0146	164.8128
		Preempt	CBP	0.1303	0.0029	0.0103	0.0102	0.0025	N/A
			OCPP	0.0383	N/A	0.0076	0.0160	N/A	N/A
			CBPP	0.1563	0.0029	0.0103	0.0102	0.0025	165.2703
CS	No pushing	No preempt	CBP	0.0870	0.0912	0.1765	0.1767	0.0936	159.5373
		Preempt	CBP	0.0909	0.0022	0.0085	0.0085	0.0020	N/A
			OCPP	0.1458	N/A	0.0968	0.0875	N/A	N/A
			CBPP	0.2025	0.0022	0.0089	0.0089	0.0020	161.0257
	Pushing	No preempt	CBP	0.0696	0.0543	0.1370	0.1335	0.0552	164.6505
		Preempt	CBP	0.0711	0.0028	0.0085	0.0086	0.0027	N/A
			OCPP	0.0986	N/A	0.0223	0.0201	N/A	N/A
			CBPP	0.1459	0.0028	0.0086	0.0087	0.0027	166.3356

adjust the distribution to $(R_4, R_5) = (11, 12)$. The dynamic adjustment of R_4 is plotted in Fig. 6(b), which clearly shows the two best values during the two bandwidth borrowing intervals and the adaptive transition between them. Along the path composed of link (3, 9, 10, 5), link-9 is the bottleneck in bandwidth borrowing, so there is enough spare capacity from trunk-12 to serve SLA-3's traffic. Therefore, SLA-3 achieves a very small CBP during this period. Again, the lender SLA-6 achieves a CBP around 10^{-3} due to the P_a control.

Example 3: Trunk Resource Sharing and Preemption: In this example, the impact of the link-level trunk resource sharing schemes on resource utilization and efficiency of the preemption scheme on QoS guarantee are investigated. In the simulation, network dimensioning, effective bandwidth, P_u and P_a configurations are the same as those used in Example 1, and we here focus on the scenario where $(\lambda_d^1, \lambda_d^2, \lambda_d^3, \lambda_d^4, \lambda_d^5) = (2 \times 46.9, 14.4, 29, 29, 14.4)$, and $(R_4, R_5) = (17, 6), (R_{10}, R_{11}) = (17, 6)$ are initially set. We consider different configurations that specify which trunk resource sharing scheme (CP, BS, VP or CS) is used and whether bandwidth pushing or flow preemption is applied.

The CBP is measured for each configuration and compared in Table III. When the flow preemption is involved, the out profile call preemption probability (OCPP) is measured as

$$\mathrm{OCPP}_i = \frac{\text{number of preempted SLA-}i\text{ calls}}{\text{number of } \textit{bor-admitted}\text{ SLA-}i\text{ calls}} \tag{7}$$

and a total call blocking and preemption probability (CBPP) is calculated by

$$\mathrm{CBPP}_i = \frac{(\text{No. of blocked} + \text{No. of preempted})\text{SLA-}i\text{ calls}}{\text{number of arrived SLA-}i\text{ calls}}. \tag{8}$$

As a more straightforward metric to measure the resource utilization, the total efficient bandwidth usage (EBU) over the network, according to [12], is calculated by

$$\mathrm{EBU} = \sum_{(s,\sigma)} \rho_{s,\sigma}(1 - \mathrm{CBPP}_{s,\sigma})e_s \tag{9}$$

where $\rho_{s,\sigma} = \lambda_{s,\sigma}/\mu_{s,\sigma}$ is the Erlang traffic load to the SLA (s, σ). Calculation in (9) in fact gives a conservative result, because traffic served within an out profile call before its preemption is not taken into account.

In the considered configurations, when VP or CS trunk resource sharing scheme is used and flow preemption is not applied, the routing/CAC procedure proposed in Section VI is slightly adjusted. In such cases, an overwhelmed in profile call is rejected instead of preempting out profile calls. The sample path (call arrival time and call holding time) is identically reproduced in all the simulations. The simulated number of flows that contribute to the statistical measures is sufficiently large to make the confidence intervals negligibly small.

From Table III, the following observations can be made:

1) With the CP trunk resource sharing, traffic service in each SLA works independently. There is no bandwidth pushing or flow preemption issue in this case. The CBP is directly obtained from the Erlang-B formula. Obviously, CP leads to the worst resource utilization. As we consider a fixed load scenario, there is no preemption issue either when BS scheme is used;
2) For a certain pushing and preemption setting, the VP or CS scheme can further improve the EBU as compared with the BS scheme. However, when flow preemption is not applied in the VP or CS scheme, the aggressive resource usage in the overloaded SLA-1 leads to QoS violation in all the other SLAs. The dilemma between high resource utilization and QoS violation, never effectively solved before, can be overcome by the flow preemption scheme. Comparing the results in all the paired "X-X-NoPreempt" and "X-X-Preempt" settings, we can observe the QoS guarantee by preemption for the normally loaded and underloaded SLAs. Also, the CBPP of SLA-1 in the VP and CS schemes is obviously smaller than that in the CP and BS cases due to a higher statistical multiplexing gain, even with the preemption cost being taken into account. In addition, "N/A" of OCPP means no *bor-admitted* call and therefore no preemption is observed in simulation;

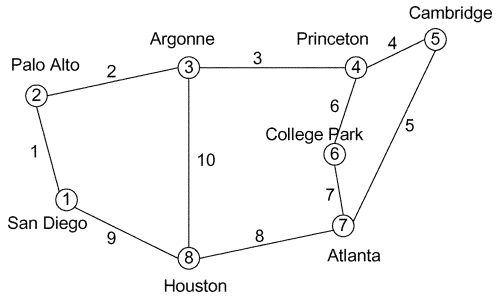

Fig. 7. Network configuration for robustness study.

3) In most cases, the configuration with the CS scheme achieves the highest EBU, as compared with the corresponding ones with the BS or VP scheme. An exception happens between "VP-Pushing-NoPreempt" and "CS-Pushing-NoPreempt". As the exact same sample path is used in each simulation, the above exception is not due to a statistical error. The exception indicates that the greedy CS scheme sometimes does not lead to the highest resource utilization in a network.
4) For any combination of preemption and trunk resource sharing settings, the bandwidth pushing technique can further improve the resource utilization, as compared with the "NoPushing" case.

B. Robustness of the Performance

The above examples comprehensively illustrate the operation and performance of the proposed resource sharing techniques. Here, we demonstrate that the bandwidth borrowing and bandwidth pushing techniques have robust performance of improving resource utilization, by simulating a larger scale network supporting tens of SLAs.

The network topology, shown in Fig. 7, is the same as that used in [12], where eight nodes are connected with 10 links. In this study, three service classes are considered. The effective bandwidth associated with classes 1, 2, and 3, are 1, 6, and 24, respectively. For example, if 1 c-unit corresponds to 16 Kb/s, the three classes can be used to support voice, medium-rate data, and video streaming services, respectively. Correspondingly the mean duration of calls of the three classes are set as 1, 4, 6.67, where one t-unit corresponds to 3 minutes. Thus, a video flow lasts on average for 20 minutes.

There are 32 SLAs installed in the network and the service classes associated with each SLA are arbitrarily set and given in Table IV. SLAs are negotiated between each pair of the nodes. For simplicity, only node pairs 1/5 and 4/7 have SLAs negotiated for all the three classes, and a single SLA is negotiated between other node pairs. Furthermore, an SLA contracts resources for traffic in both directions. Each SLA between a node pair is supported by parallel traffic trunks. For each SLA, all parallel paths not longer than 4 hops are searched and form the trunk set. Altogether, there are 66 traffic trunks to serve the 32 SLAs. Each SLA has an engineered call arrival rate for a target Erlang load of 41.5 and a nominal capacity of 54 calls to guarantee a target CBP of 0.01. The SLA capacity is evenly distributed to related traffic trunks. Each link capacity is tailored to exactly hold all the traffic trunks crossing that link to get a well dimensioned network.

TABLE IV
SLAs BETWEEN EACH NODE PAIR AND ASSOCIATED SERVICE CLASS

	①	②	③	④	⑤	⑥	⑦	⑧
①	-	2	3	2	1, 2, 3	1	2	3
②	-	-	1	3	1	2	1	3
③	-	-	-	3	2	2	1	3
④	-	-	-	-	1	2	1, 2, 3	1
⑤	-	-	-	-	-	1	3	2
⑥	-	-	-	-	-	-	1	2
⑦	-	-	-	-	-	-	-	3
⑧	-	-	-	-	-	-	-	-

Underloaded SLAs and overloaded SLAs are created to observe the operations of the bandwidth borrowing and bandwidth pushing. For convenience, the SLAs in Table IV are identified as SLA-1 to SLA-32 from left up to right down, row by row. The overloaded SLAs are 1, 5, 7, 11, 13, 14, 15, 16, 18, 19, 20, 21, 24 and 30, underloaded ones are 2, 3, 8, 10, 12, 17, 22, 23, 25, 26, 27, 28, 31 and 32, and normally loaded are 4, 6, 9 and 29. With this setting, all links have both lender trunks and borrower trunks. In such an environment, an intuition may be that the bandwidth pushing is unnecessary as spare capacity can be exploited on each link. However, the simulation results demonstrate that bandwidth pushing can still clearly increase the resource utilization in this case. To show the robustness of the performance, we run simulations with different trunk sharing schemes and under various load conditions. All the underloaders are tuned to have $\lambda_d = 0.49\lambda_p$, and the corresponding $R = 30$ calls (for a $P_u = P_t = 0.01$) which is at first evenly distributed among traffic trunks. Also, $P_a = 1$ is used. Three load conditions are simulated by varying the call arrival rates of overloaded SLAs as $\lambda_d = 2\lambda_p$, $3\lambda_p$, and $4\lambda_p$, respectively, where the network as a whole is about 22%, 67%, and 112% overloaded as compared with the engineered traffic load, and referred to as lightly, medially, and heavily loaded, correspondingly.

Table V presents the simulated call level throughput (CLT) and the EBU obtained from (9) when the BS, VP, or CS scheme is used as the trunk resource sharing scheme under the three load conditions. The preemption scheme is applied in all cases to enforce the SLA compliance. The CLT and EBU of the CP case are calculated from the Erlang-B formula as the performance benchmark. The sample path is identically reproduced for all the simulations in a given load condition. For each run of simulation, a sufficient number of flows are generated to make the confidence intervals negligibly small.

With the traffic load changes from lightly loaded to heavily loaded, the CLT and EBU under CP increase only marginally due to the saturation in overloaded SLAs and resource waste in underloaded SLAs. In all the load scenarios, BS, VP and CS consistently achieve improved resource utilization as compared with CP. The bandwidth pushing also has robust performance to further increase resource utilization in all the cases. As mentioned earlier, it is not obvious what factors lead to the good performance of bandwidth pushing in a basically balanced network, where lenders and borrowers have a roughly uniform distribution over the network and bandwidth borrowing happens on each link. Simulation results show that the bandwidth pushing dynamically adjusts the protection/spare bandwidth distribution on the time scale of call inter-arrival time and efficiently utilizes the call-level statistical multiplexing. The dynamic protection

TABLE V
ROBUST PERFORMANCE OF BANDWIDTH BORROWING AND BANDWIDTH PUSHING

			CP	BS	VP	CS
Lightly overloaded	No pushing	CLT	0.5619×10^3	0.7608×10^3	0.7669×10^3	0.7682×10^3
		EBU	1.1789×10^4	1.4273×10^4	1.5288×10^4	1.5297×10^4
	Pushing	CLT	N/A	0.7644×10^3	0.7681×10^3	0.7689×10^3
		EBU	N/A	1.4778×10^4	1.5436×10^4	1.5419×10^4
	EBU increase		N/A	3.54%	0.97%	0.8%
Medially overloaded	No pushing	CLT	0.5678×10^3	0.9987×10^3	1.0025×10^3	1.0161×10^3
		EBU	1.1902×10^4	1.4873×10^4	1.6360×10^4	1.6433×10^4
	Pushing	CLT	N/A	1.0089×10^3	1.0088×10^3	1.0197×10^3
		EBU	N/A	1.6010×10^4	1.6894×10^4	1.6843×10^4
	EBU increase		N/A	7.64%	3.26%	2.49%
Heavily overloaded	No pushing	CLT	0.5696×10^3	1.2131×10^3	1.2178×10^3	1.2471×10^3
		EBU	1.1936×10^4	1.5179×10^4	1.6838×10^4	1.6830×10^4
	Pushing	CLT	N/A	1.2276×10^3	1.2250×10^3	1.2509×10^3
		EBU	N/A	1.6365×10^4	1.7311×10^4	1.7120×10^4
	EBU increase		N/A	7.81%	2.81%	1.72%

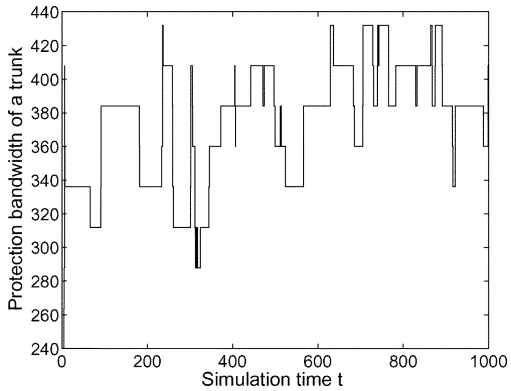

Fig. 8. Dynamic bandwidth pushing on call-level time scale.

bandwidth adjustment for one of the lender traffic trunks associated with SLA-25, passing link-6 and link-7, is illustrated in Fig. 8, where the simulated scenario is "Medially loaded-VP-Pushing". The call-level bandwidth pushing is not fully reflected in Fig. 6(b), due to the single spare path configuration used in the example.

Under a certain pushing setting and load condition, CS always achieves the largest CLT due to its greedy feature, but not always the largest EBU. This further strengthens the observation (3) given in Example-3 in Section VII-A. In fact, in most cases, VP achieves the largest EBU. It is not difficult to understand that the greedy approach is not the optimal admission policy from the bandwidth usage perspective, especially in the heavily loaded heterogeneous environment. For example, assume that at a certain moment a link has the leftover capacity of 24. Then a new class-1 call $(e_1 = 1)$ arrives followed by a class-3 call $(e_3 = 24)$. According to CS, the class-1 call is admitted and the leftover capacity changes to 23, where the following class-3 call has to be rejected, leading to low resource utilization. Our observations regarding the resource sharing policies are consistent with Borst and Mitra's conclusion in [20] that the revenue (proportional to bandwidth usage) generated by VP is extremely close to the maximum achievable value. Moreover, the improvement from the bandwidth pushing is also mainly shown in the EBU rather than the CLT. In the design, the pushing procedure intends to reserve the spare bandwidth for high-bandwidth flows, which can be considered as a form of ungreedy resource sharing.

It should be emphasized that the robust performance of efficient resource utilization is achieved without any SLA violation. In operation, the bandwidth borrowing, bandwidth pushing and preemption may lead to an extra overhead during the CAC. However, with the proposed data structure, the above procedures require only addition and table looking-up operations, which are very convenient for computer execution. Therefore, the impact of bandwidth borrowing on the CAC overhead time should be acceptable.

VIII. CONCLUSION

We have proposed a bandwidth borrowing scheme for dynamic inter-SLA resource sharing in path-oriented DiffServ networks, where the spare capacity from underloaded SLAs can be efficiently exploited without SLA violation. The basic methodology is "boundary resource commitment determines link resource sharing". In addition to QoS specifications for an engineered traffic load, each SLA explicitly specifies the QoS that should be guaranteed in an underloaded period, where a protection bandwidth smaller than the nominal capacity is adjusted dynamically according to the actual traffic arrival rate. The protection bandwidth is guaranteed for the underloaders, and the available spare capacity is then properly distributed to related links for lending to others.

A call-level service differentiation concept is proposed for resource sharing. Traffic flows admitted with borrowed bandwidth are tagged as *out* profile calls, possible to be preempted later when the original bandwidth owner needs to claim back the resources. The call-level differentiation scheme provides the freedom of using any link resource sharing (such as the BS, VP, or CS) for maximum resource utilization without SLA violation, as any aggressive resource usage from borrowers is then preempted by the original owners when necessary. Furthermore, a distributed bandwidth pushing scheme is proposed, which further exploits the call-level statistical multiplexing by adaptively adjusting the spare capacity distribution over the network. Efficiency and robustness of the proposed resource sharing techniques are demonstrated through extensive simulation studies.

For future work, we are investigating the implementation of the bandwidth borrowing scheme using a fully distributed bandwidth broker. For inter-SLA resource sharing, messaging between the edge routers and the bandwidth broker is required for

SLA status update and per-flow admission control. If the bandwidth broker is a physical central controller, it is then prone to become a congestion point in the network. Therefore, we are trying to achieve the bandwidth broker as a *logical* central entity, whose functionalities are physically distributed to all the edge routers. In addition, we plan to further study the fairness issue involved in the bandwidth borrowing.

ACKNOWLEDGMENT

The first author would like to thank Prof. A. Leon-Garcia for stimulating discussions on the computer simulations. The authors are grateful to the anonymous reviewers for their constructive comments and suggestions which helped to improve the quality of this paper.

REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," IETF, RFC 2475, Dec. 1998.
[2] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," IETF, RFC 2638, Jul. 1999.
[3] A. Terzis, L. Wang, J. Ogawa, and L. Zhang, "A two-tier resource management model for the Internet," in *Proc. IEEE GLOBECOM*, 1999, vol. 3, pp. 1779–1791.
[4] Z.-L. Zhang, Z. Duan, L. Gao, and Y. T. Hou, "Decoupling QoS control from core routers: a novel bandwidth broker architecture for scalable support of guaranteed services," in *Proc. ACM SIGCOMM*, 2000, pp. 71–83.
[5] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni, "Traffic engineering with MPLS in the Internet," *IEEE Network*, vol. 14, no. 2, pp. 28–33, Mar./Apr. 2000.
[6] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS," Internet RFC 2702, Sep. 1999.
[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] E. Mykoniati, C. Charalampous, P. Georgatsos, T. Damilatis, D. Goderis, P. Trimintzios, G. Pavlou, and D. Griffin, "Admission control for providing QoS in DiffServ IP networks: the TEQUILA approach," *IEEE Commun. Mag.*, vol. 41, no. 1, pp. 38–44, Jan. 2003.
[9] Y. Cheng and W. Zhuang, "Effective bandwidth of multiclass Markovian traffic sources and admission control with dynamic buffer partitioning," *IEEE Trans. Commun.*, vol. 51, no. 9, pp. 1524–1535, Sep. 2003.
[10] G. Armitage, "MPLS: the magic behind the myths," *IEEE Commun. Mag.*, vol. 38, no. 1, pp. 124–131, Jan. 2000.
[11] G. Swallow, "MPLS advantages for traffic engineering," *IEEE Commun. Mag.*, vol. 37, no. 12, pp. 54–57, Dec. 1999.
[12] E. Bouillet, D. Mitra, and K. G. Ramakrishnan, "The structure and management of service level agreements in networks," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 691–699, May 2002.
[13] P. Trimintzios *et al.*, "A management and control architecture for providing IP differentiated services in MPLS-based networks," *IEEE Commun. Mag.*, vol. 39, no. 5, pp. 80–88, May 2001.
[14] S. Wang, D. Xuan, R. Bettati, and W. Zhao, "Providing absolute differentiated services for real-time applications in static-priority scheduling networks," in *Proc. IEEE INFOCOM*, 2001, vol. 2, pp. 669–678.
[15] C. Dou and F.-C. Ou, "Performance study of bandwidth reallocation algorithms for dynamic provisioning in differentiated services networks," in *Proc. 15th Int. Conf. Information Networking*, 2001, pp. 700–705.
[16] J. Qiu and E. W. Knightly, "Inter-class resource sharing using statistical service envelopes," in *Proc. IEEE INFOCOM*, 1999, vol. 3, pp. 1404–1411.
[17] R. R. Boorstyn, A. Burchard, J. Liebeherr, and C. Oottamakorn, "Statistical service assurances for traffic scheduling algorithms," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 12, pp. 2651–2664, Dec. 2000.
[18] C. Cetinkaya, V. Kanodia, and E. W. Knightly, "Scalable services via egress admission control," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 69–81, Mar. 2001.
[19] J. Qiu and E. W. Knightly, "Measurement-based admission control with aggregate traffic envelopes," *IEEE/ACM Trans. Netw.*, vol. 9, no. 2, pp. 199–210, Apr. 2001.
[20] S. C. Borst and D. Mitra, "Virtual partitioning for robust resource sharing: Computational techniques for heterogeneous traffic," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 5, pp. 668–678, Jun. 1998.
[21] R. Garg and H. Saran, "Fair bandwidth sharing among virtual networks: A capacity resizing approach," in *Proc. IEEE INFOCOM*, 2000, vol. 1, pp. 255–264.
[22] A. W. Berger and W. Whitt, "Effective bandwidth with priorities," *IEEE/ACM Trans. Netw.*, vol. 6, no. 4, pp. 447–460, Aug. 1998.
[23] R. J. Gibbens and F. P. Kelly, "Network programming methods for loss networks," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 7, pp. 1189–1198, Sep. 1995.
[24] P. B. Key, "Optimal control and trunk reservation in loss networks," *Prob. Eng. Info. Sci.*, vol. 4, pp. 203–242, 1990.
[25] P. Flegkas, P. Trimintzios, and G. Pavlou, "A policy-based quality of service management system for IP DiffServ network," *IEEE Network*, vol. 16, no. 2, pp. 50–56, Mar.-Apr. 2002.
[26] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB," Internet RFC 2598, Jun. 1999.
[27] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," Internet RFC 2597, Jun. 1999.
[28] D. Mitra and K. G. Ramakrishnan, "A case study of multiservice, multipriority traffic engineering design for data networks," in *Proc. IEEE GLOBECOM*, 1999, vol. 1B, pp. 1077–1083.
[29] Y. Cheng and W. Zhuang, "Diffserv resource allocation for fast handoff in wireless mobile internet," *IEEE Commun. Mag.*, vol. 40, no. 5, pp. 130–136, May 2002.
[30] V. Paxson and S. Floyed, "Wide area traffic: the failure of Poisson modeling," *IEEE/ACM Trans. Netw.*, vol. 3, no. 3, pp. 226–244, Jun. 1995.
[31] A. Feldmann, A. Gilbert, W. Willinger, and T. G. Kurtz, "The changing nature of network traffic: scaling phenomena," *Comput. Commun. Rev.*, vol. 28, pp. 5–29, Apr. 1998.
[32] S. Jamin, P. B. Danzig, S. J. Shenker, and L. Zhang, "A measurement-based admission control algorithm for integrated service packet networks," *IEEE/ACM Trans. Netw.*, vol. 5, no. 1, pp. 56–70, Feb. 1997.

Yu Cheng (S'01–M'04) received the B.E. and M.E. degrees from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2003, all in electrical engineering.

From September 2003 to August 2004, he was a Postdoctoral Fellow in the Department of Electrical and Computer Engineering at the University of Waterloo. Since September 2004, he has been a Postdoctoral Fellow in the Department of Electrical and Computer Engineering at the University of Toronto, Toronto, ON, Canada. His research interests include Internet QoS, traffic engineering, service and network management, and wireless/wireline interworking.

Dr. Cheng received a Postdoctoral Fellowship from the Natural Sciences and Engineering Research Council of Canada (NSERC) in 2004.

Weihua Zhuang (M'93–SM'01) received the B.Sc. and M.Sc. degrees from Dalian Maritime University, Liaoning, China, and the Ph.D. degree from the University of New Brunswick, Fredericton, NB, Canada, all in electrical engineering.

Since October 1993, she has been with the Department of Electrical and Computer Engineering, University of Waterloo, ON, Canada, where she is a Professor. She is a coauthor of the textbook *Wireless Communications and Networking* (Prentice Hall, 2003). Her current research interests include multimedia wireless communications, wireless networks, and radio positioning.

Dr. Zhuang is a licensed Professional Engineer in the Province of Ontario, Canada. She received the Outstanding Performance Award in 2005 from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2001 from the Ontario Government. She is an Editor/Associate Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and *EURASIP Journal on Wireless Communications and Networking*.