Smart Box Architecture: A Hybrid Solution for IP QoS Provisioning

Bülent Yener * Gong Su † Eran Gabber ‡

Abstract

This work proposes a hybrid solution, called the Smart Box Architecture (SBoX), that provides Quality of Service (QoS) in Internet Protocol (IP) based networks. SBoX architecture consists of *SBoX servers*, which are located at the network boundary, and *SBoX routers*, which are add-on Label Switching Routers (LSR), which are located at interior network nodes. This approach combines the advantages of three existing technologies: *integrated services* (Intserv), *differentiated services* (Diffserv), and *multi-protocol label switching* (MPLS).

SBoX aggregates traffic in three levels: *commodity-flows*, *macro-flows* and *micro-flows*. Commodity-flows aggregate flow between every pair of edge points. The packets of the same commodity-flow are marked by an MPLS label. Commodity flows are composed of macro-flows, which aggregate the traffic of a particular enterprise. Macro flows are associated with an explicit Service Level Agreement (SLA), which is offered to users. Macro-flows are composed of micro-flows, which are the traffic associated with a particular individual in an enterprise or a particular application.

SBoX servers provide Diffserv like SLA to users. They use Class Based Queuing (CBQ) with a hierarchy of flow aggregation. SBoX servers manage macro-flows and commodity flows only, and leaves the management of micro-flows to the enterprise/users which signed the SLA for the macro-flow. SBoX routers perform MPLS routing of commodity-flows in interior network nodes. SBoX routers can be combined with existing best-effort routers to provide QoS in a network that lacks end-to-end deployment of LSRs.

This paper describes the SBoX architecture and its operation in detail. It also reports experimental results obtained on a prototype network. The results indicate that the SBoX architecture can indeed provide guaranteed performance in a congested network, and that SBoX routers can be combined with commodity best-effort routers to enable QoS in a heterogeneous network.

*Bell Laboratories, Lucent Technologies, 700 Mountain Ave., Murray Hill, NJ 07974 E-mail: yener@research.bell-labs.com, Tel: (908) 582 7087

†Dept. of Computer Science, Columbia University, NY, NY 10027 E-mail: gongsu@cs.columbia.edu

‡Bell Laboratories, Lucent Technologies, 700 Mountain Ave., Murray Hill, NJ 07974 E-mail: eran@research.bell-labs.com

1 Introduction and Motivation

There is a growing demand for providing Quality of Service (QoS) in the Internet for services such as voice-over-IP, streaming video, and various collaborative activities. The current Internet infrastructure cannot support QoS since it is designed for best-effort service model. Best-effort routers employ First-In First-Out (FIFO) scheduling without any guarantees. Increasing the network capacity by adding more bandwidth and routers is not always feasible or efficient. The network must have mechanisms to distinguish QoS requirements of different applications that share the same infrastructure and process them accordingly.

There are two fundamentally different approaches for providing QoS in the Internet: (i) *Integrated Services* (Intserv), and (ii) *Differential Services* (Diffserv). The Intserv approach [SPG97, Wro97, SW97] requires high-end routers to maintain per-micro-flow [1] state information and to perform complex link scheduling algorithms. As the number of flows increases and/or changes frequently, the overhead of this approach does not scale.

The Diffserv approach [Cla97, CW97, NJZ97, SZ98, ea98] strives to reduce the per-flow complexity by providing an aggregated treatment of user traffic that belongs to the same service class. Packets in the Diffserv model are marked at the network entry points in order to indicate whether or not the source follows its Service Level Agreement (SLA). The packets that violate their SLA (i.e., OUT packets) are dropped with a higher probability than the packets that obey their SLA (i.e., IN packets). Several performance studies [BW99, IN98] of the Diffserv approach show that (i) it cannot offer a quantifiable service to TCP traffic, (ii) there is strong dependency between IN and OUT packets (due to the shared queue) and consequently IN packets may be dropped, and (iii) mixing IN and OUT packets in a single TCP connection reduces the connection's performance. Non-deterministic and non-quantifiable QoS cannot be acceptable for certain applications, such as IP telephony, real-time interactive transactions, IP high definition video (e.g., High-Definition Television, or HDTV), and Virtual Private Networking (VPN) for which pricing and QoS are strongly coupled.

Other approaches such as Core-Stateless Fair Queuing (CSFQ) [SSZ98] attempt to provide per-flow handling of packets in routers without incurring the overheads of Intserv. In particular, Dynamic Packet States (DPS) [ea99b] adds flow state information to every packet belonging to that flow. DPS requires checking and updating the state information that is included in each packet. Thus it introduces extra processing complexity at each intermediate node. Furthermore it requires modification of existing routers.

Recently, Multi Protocol Label Switching (MPLS) [RVC98] has become an attractive technology. MPLS provides two main advantages. First, it decouples routing from forwarding. Thus, *explicit routing*, based on metrics different from the ones used by the traditional routing algorithms can be deployed. Label distribution protocols such as the Reservation Protocol (RSVP) or Constraint-based Label Distribution Protocol (CR-LDP) can be used to set QoS based routes. Second, it aggregates traffic flows to Forwarding Equivalent Classes (FEC) using virtual circuit switching. Thus explicit routing enables traffic engineering and QoS based constrained routing, while flow aggregation provides a scalable solution. However, ensuring QoS guarantees may not be possible with MPLS due to lack of resource allocation, scheduling and admission control.

[1] A micro-flow is an application level flow defined by its source, destination address, port numbers and protocol identifier.

This paper is organized as follows. Section 2 contains an overview of the SBoX architecture and its design principles. Section 3 discusses provisioning and configuration issues. Section 4 presents the components of the SBoX architecture. Section 5 describes the prototype implementation and experimental results. Section 6 concludes.

2 Principles and Overview of the SBoX Architecture

Any proposal for providing QoS in the IP networks must address the fundamental tradeoff between scalability and QoS guarantees. In order to provide end-to-end QoS guarantees, some notion of resource reservation is necessary. However, only aggregated flows may have resource reservations in order to minimize the per-flow state information in the routers to ensure scalability. The granularity of aggregation (i.e., the set of the flows that are treated by the network uniformly) impacts the level of service that each micro-flow receives, and it may also create fairness problems. Furthermore, real-time, adaptive resource management and admission decisions increase the complexity of the interior nodes, which may cause performance bottlenecks.

Due to the above tradeoff between scalability and QoS, a separation of the QoS function at the edge and at the core nodes can be realized as follows. Processing and state information overhead that do not scale well at interior nodes should be delegated to the edge nodes. Thus, edge nodes should be in charge of the following three activities: (i) admission control, (ii) QoS based network access, which includes micro-flow management, traffic engineering, and generating aggregate flows, and (iii) service based billing and charging.

In contrast, the responsibility of interior nodes must be limited to a minimum set of operations sufficient to support the ingress/egress QoS mechanisms inside the network. Their complexity can be reduced by the following methods: (i) separating routing from forwarding, (ii) performing pseudo circuit switching (e.g., MPLS, virtual circuit switching) to eliminate per-packet IP lookup and filtering overhead, (iii) using aggregated flow management with Class Based Queuing (CBQ) or Weighted Fair Queuing (WFQ) scheduling, and (iv) using light-weight signaling protocols for aggregate flow reservations.

The SBoX architecture is based on the above design principles. In this way, it combines the best properties of Intserv, Diffserv, and MPLS proposals with QoS provisioning and admission control.

In the following discussion we will refer to an Autonomous System (AS), which is a portion of the Internet under one administrative domain, as "the network". We now describe each of the SBoX components in detail.

SBoX architecture consists of three components: SBoX servers, SBoX routers, and algorithms for provisioning and admission control. *SBoX servers* are intelligent edge routers, that are deployed in Service Access Points (SAP) at the network boundary. SAPs handle all traffic to/from the network end points. An end point may be a user or a connection to another AS (see section 3). SBoX servers perform flow management based on SLAs and traffic labeling for MPLS routine at interior nodes. SBoX servers enforce Diffserv like SLAs. Each agreement refers to a *macro-flow* with a specific bandwidth, where a macro-flow is defined as the traffic between a specific pair of end points. A macro-flow is shared by all the application-level micro-flows between these end points. All macro-flows between a pair of SAPs are aggregated to a *commodity-flow*. The packets of the same commodity-flow are marked by an MPLS label which is globally unique within an AS. SBoX routers enforce the SLAs and pack the traffic into commodity flows by CBQ scheduling.

SBoX routers are programmable, add-on Label Switching Routers (LSR). SBoX routers are added only to a subset of the best-effort IP routers inside the network. This subset connects all SAPs and Border Routers (BRs) in the AS. SBoX routers perform MPLS routing with CBQ over IP, which is totally transparent to current IP-based network infrastructure and protocols. When an SBoX router is added to an existing best-effort router, it controls some of the links of the IP router, so that it switches labeled traffic (commodity-flows) directly, and passes other traffic (best effort) to the best-effort router. Label switching can be done by using the global labels (for simplicity) and commodity-flows are protected from each other and from best-effort traffic using CBQ.

Add-on SBoX routers address two problems in current networks: (i) the lack of end-to-end deployment of LSRs (i.e., many routers are not MPLS capable), and (ii) the lack of programmability of the legacy routers. The add-on approach provides service guarantees in a network that contains legacy routers without replacing or changing them.

SBoX architecture proposes new algorithms for provisioning, resource allocation, and admission control. The core of provisioning and resource allocation task is the placement of SBoX routers and the selection of the links that enforce QoS guarantees. The SBoX routers and the links that they manage induce a *virtual premium network* on the existing best-effort network. The virtual premium network is used for label switching of the QoS traffic.

Given the pairwise statistical demands for QoS traffic over a fixed interval, we formulate the problem of designing an optimal virtual network as a multi-commodity problem with integer variables. The objective function is to *pack* the commodity-flows into a minimal feasible subgraph while avoiding bandwidth fragmentation on the links. An SBoX router is associated (activated) with each node that appears in the solution of this optimization problem. The solution to the optimization problem specifies the amount of bandwidth allocation necessary for each commodity. Bandwidth allocation and protection is done by setting CBQ mechanisms in the corresponding SBoX routers. The parameters of the CBQ scheduling and the labels of macro-flows are distributed with a light weight protocol which aims to minimize the communication overhead.

SBoX users must pick SLAs from a predefined service menu for all macro-flows. The predefined service menu enables token based admission control, which in turn simplifies provisioning and billing, and reduces run-time complexity. We believe that the advantages of predefined service menu outweigh its disadvantages, which include over-provisioning and some bandwidth waste. See Section 4.1.2 for details.

3 Network Provisioning

We consider the IP-based Internet as an interconnection of autonomous systems (AS) in which a provider has control of all the resources (i.e., it is a single administrative domain). An AS can be accessed only through entry points called Service Access Points (SAPs). A provider aggregates the user traffic at the SAPs. The AS can be connected to other networks via Border Routers (BRs) that are treated as SAPs in our model. SAPs deal with admission control and perform policy based bandwidth management at network boundary. The discussion in this work

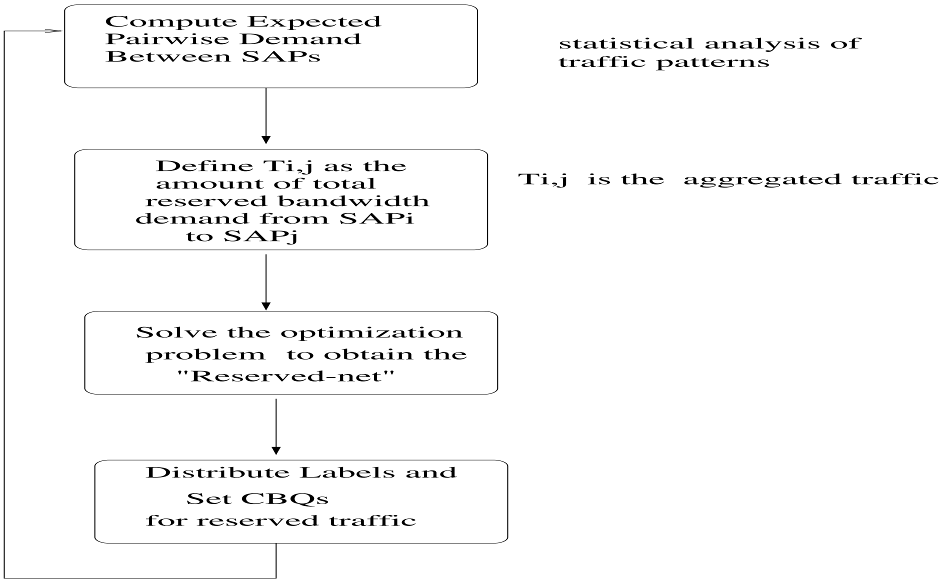

Figure 1: Steps for provisioning and configuration of QoS traffic

is focused on a single AS.

We distinguish between two types of traffic carried by the network: (i) reserved or committed QoS traffic, and (ii) non-reserved or best-effort traffic. Network resources are provisioned and configured to accommodate to the QoS traffic. Thus, provisioning and configuration require estimating pairwise bandwidth demands for QoS traffic between all SAPs and BRs in an AS. There are several techniques for source modeling that can be adapted for this purpose [ML97]. Provisioning and configuration of QoS traffic in the SBoX architecture are achieved by creating a virtual premium network called the *reserved-net.* The reserved-net is a virtual dedicated network that connects all the SAPs and BRs. It protects the reserved traffic from the best effort traffic and forwards QoS traffic using a connection oriented virtual circuit switching protocol such as MPLS. The best-effort traffic uses the network links which are not included in the reserve-net. In addition, best-effort traffic may be permitted to use the reserved-net. However, the reserved traffic has preemptive priority in the reserve-net. Figure 1 depicts the main steps for provisioning and configuration of QoS traffic.

We note that the problem of provisioning a reserved-net is a version of network optimization problem for which a rich literature exists (e.g., [Dov91a, Dov91b, CFZ94, ACL94, Ash95, For96, Lee95, MMR96]). However, the objective function of our problem is different as we explain in the next section.

3.1 Optimum Virtual Network

The network is represented by a graph $G = (V, E)$ where node set V is partitioned into two distinct subsets, $V = R \cup S$. The set R contains the routers, and the set S contains the SAPs. We assume that each node in V has a unique identifier. The set E contains the network links, each of which has capacity (bandwidth) $c_{(i,j)}$ (bits/sec) $\forall (i,j) \in V$. For simplicity we assume that both directions of a link have the same capacity. The reserved-net is a subgraph $G_r \subset G$, represented as $G_r = (R' \cup S, E')$. End-points of a commodity-flow must belong to the set S. Let T be a traffic matrix in which an entry $t_{i,j}$ indicates the aggregated bandwidth request for QoS traffic from i to $j \quad \forall i, j \in S$. In other words $t_{i,j}$ is the sum of the bandwidth of macro-flows from SAP i to SAP j. We assume that T is based on the statistical information capturing a correlated source behavior (e.g., the traffic volume between 8-11 AM). The optimization problem can be formulated as an instance of mixed-integer multi-commodity flow problem, where a commodity k is defined for a pair of nodes $l, m \in S$ such that $t_{l,m} > 0 \quad$ and $m \neq l$. In other words, a commodity k is provided from each SAP l to another SAP m, so that l has non-zero bandwidth request to m. For notational simplicity the total amount $t_{l,m}$ of commodity k will be denoted by θ_k

Our cost measure is to *pack* the *routes* of reserved traffic together so that (i) number of links used by the solution is minimal, and (ii) *bandwidth fragmentation* is minimized for each link included into the reserved-net. There are two motivations behind using bandwidth fragmentation as a measure: (i) support policies for sharing of link bandwidth between best-effort and QoS traffic (i.e., as a part of QoS provisioning), and (ii) minimize the amount of "unused" or "left-over" bandwidth of a reserved link, which is smaller than any expected commodity-flow bandwidth request, if no best-effort traffic is allowed on the reserved-net.

Figure 3 depicts the objective function and the constraints of the optimization problem we wish to solve. The variables and parameters of the optimization problem are as follows:

• Flow variable $f^k_{i,j}$ which takes a real value and indicates the amount of commodity k over the link (i, j).

• A 0-1 integer variable $x_{i,j}$ which indicates the link (i, j) is used by any commodity. It is 1 if used, 0 otherwise. In other words, $x_{i,j}$ denotes whether link (i, j) is included in the reserved-net.

• A 0-1 integer variable $y_{i,j}$ which indicates whether or not the *unused* bandwidth of the link (i, j) is below a given *fragmentation threshold* $\delta_{i,j}$.

• A coefficient α which controls the impact of the bandwidth fragmentation on the objective function.

In Figure 3, constraint (1) ensures the flow balance for each commodity. If the node is an intermediate node (i.e., $i \in R$ is a router), the in-flow should be equal to the out-flow. Otherwise, the node i is an end-node of a macro flow (i.e., it is a SAP). Thus the flow into this node from commodity k is θ_k for each commodity k. The inequality (2) ensures that total load on a link (i, j) does not exceed its capacity. Furthermore, the load is assigned such that the bandwidth fragmentation is minimized by the penalty term $y_{i,j}$. Inequality (3) ensures that the penalty term exists only for the links in the reserved-net. Inequalities (4) and (5) are self explanatory.

The solution to this optimization problem is a set of $x_{i,j}$ and $y_{i,j}$ values that minimize the cost function while connecting each pair of ingress/egress points. The links with $x_{i,j} = 1$ are called *reserved-links* and the nodes incident to reserved-links are called *reserved-nodes*. The union of the reserved-links induces a connected graph G_r which is our reserved-net as shown in Figure 2.

A path from SAP_k to SAP_l on the reserved-net is called *reserved-path*. It is the Label Switched Path (LSP) that carries the commodity-flow for commodity k. It may be desirable to over allocate the capacity for providing fault-tolerance and dynamic admission control. A simple way to achieve this is to add slack variables to the $t_{i,j}$ and solve the above optimization problem. Note that the exact solution of the above optimization problem is computationally expensive; thus we may either use heuristics to obtain suboptimal solutions, or solve the optimization

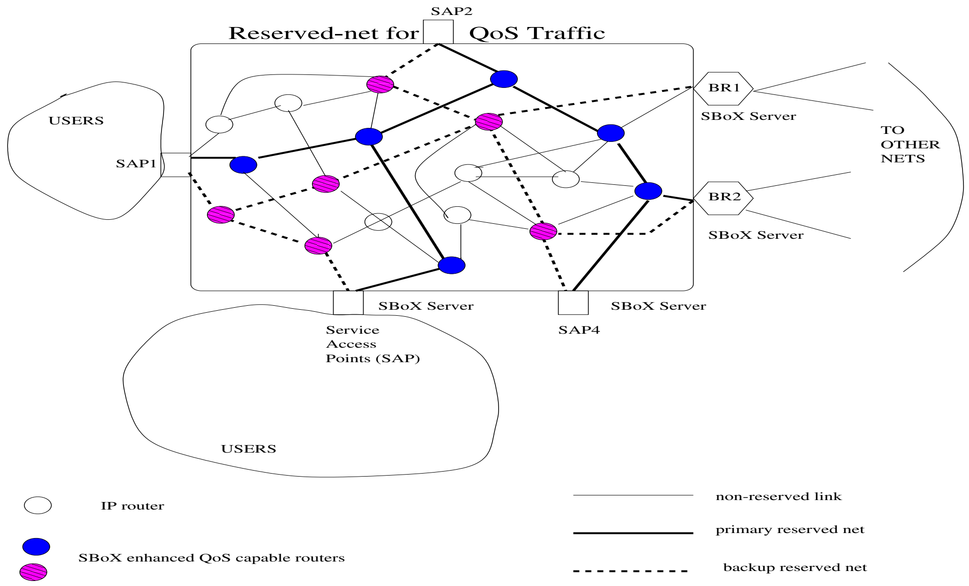

Figure 2: Reserved-Net: virtual network for MPLS traffic.

Minimize: C= $\sum_{i,j} x_{i,j} + \alpha(y_{i,j}) \quad \forall (i,j) \in E$ and $\forall k$

Constraints:

s.t.

(1)

$$\sum_{j\neq i} f^k_{(j,i)} - \sum_{j\neq i} f^k_{(i,j)} = \begin{cases} \theta_k & \forall k, i \;\; s.t. \;\; k \neq i \\ 0 & \forall i \in R \end{cases}$$

(2) $\sum_k f^k_{i,j} \leq (c_{i,j} - \delta_{i,j})x_{i,j} + \delta_{i,j} y_{i,j} \quad \forall (i,j) \in E$

(3) $y_{i,j} \leq x_{i,j}$

(4) $x_{i,j} \in \{0,1\} \quad \forall (i,j) \in E$

(5) $f^k_{i,j} \geq 0$

Figure 3: Mixed-integer formulation of the problem

problem off-line and store it for different traffic matrices T in advance. In the next section we propose two heuristics with polynomial time complexity.

3.2 Heuristic Approaches

The heuristics are based on building multiple routing trees of G to connect the SAPs. This problem can be treated as an instance of the Steiner Tree problem and several heuristics are known for finding approximate solutions for the minimum cost Steiner Tree problem [HRW92].

Heuristic 1

The first heuristic produces two Steiner trees which have the following two properties: (i) trees are edge disjoint, and (ii) they share minimal number of Steiner vertices (i.e, vertices other than the SAPs). One tree is used as the primal routing tree, while the other tree is kept for backup. Of course, the minimum capacity on tree links must be at least the maximum bandwidth provisioned for the reserved traffic.

The algorithm has two phases. In the first one the following steps are performed:

• Preprocessing: eliminate all the links that do not have the capacity to handle aggregate bandwidth demand of the reserved traffic.

• Construct a Steiner tree (call it T_1) to connect all the SAPs with minimum number of Steiner vertices.

• Remove all the Steiner vertices used by T_1 and the links associated with them from the network graph. If the SAPs remains pairwise connected then build another Steiner tree T_2 and stop. Else build a spanning tree of the connected components using some of the deleted Steiner vertices and their links.

Assuming that reserved-net is obtained from the above heuristic approach, there is a single reserved-path between any pair of SAPs. Thus, the QoS based explicit routing for MPLS is loop-free and obvious. However, the same simplicity is also the main drawback of this heuristic since the links may not have enough capacity to carry all the aggregated traffic. Next we present another heuristic algorithm to remedy this problem.

Heuristic 2

The second heuristic is based on constructing a shortest path tree for each SAP. Each tree is directed and obtained from the union of the shortest paths from an SAP to all others. The distance metric can be hop count, delay or physical distance. The objective of the second heuristic is to pack the routing trees so that minimum number of trees sharing an edge is maximized. Since our focus is on the QoS architecture, we will not further elaborate on the optimization issues.

3.3 Resource Allocation

The solution to the optimization problem determines reserved-links and the amount of flow that can be assigned to these links. In particular the $f^k_{i,j}$ value indicates the bandwidth request of commodity-flow on link (i, j) that resides in the reserved-path from SAP_k to SAP_l. In order to enforce service guarantees, $f^k_{i,j}$ must be reserved and protected on link (i, j). Resource allocation in the reserved-net is done at the commodity-flow level and enforced by CBQ.

Different policies may be adapted to adjust the interaction between the reserved and the best-effort traffic. We propose a *movable boundary* scheme, in which the best-effort can use the available reserved bandwidth, where reserved traffic has preemptive priority as shown in Figure 4. The position of the boundary can be dictated by the network economics and utilization. For example, an Internet Service Provider (ISP) may decide to reserve only 60% of link capacity to the aggregated reserved-traffic.

The policy for setting such a boundary can be enforced in the optimization problem by the threshold variable $\delta_{i,j}$ and the coefficient α.

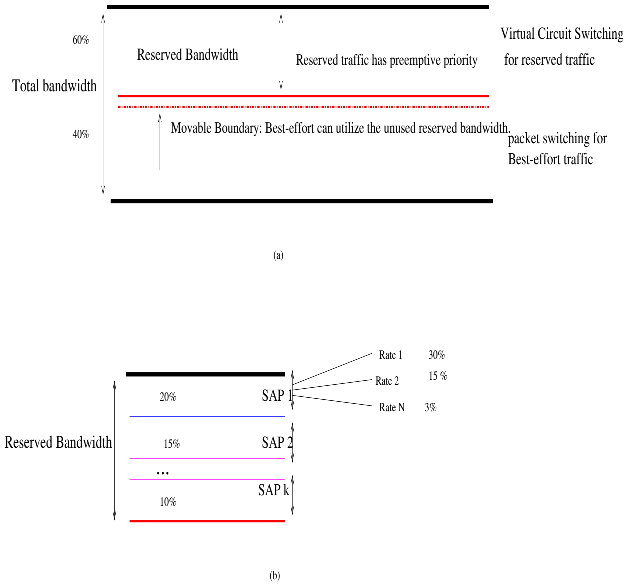

Figure 4: Bandwidth sharing

Distribution of MPLS labels and setting the CBQs can be done using a simple signaling protocol as presented in the next section.

3.4 A Light Weight Signaling Protocol

Currently two protocols are being discussed to set a LSP (i.e., to distribute the labels and make the reservations at LSR): RSVP extension, and CR-LDP [ea99a]. The RSVP maintains a soft state which has the overhead of refreshing state. As a result of its response delay is higher. Furthermore RSVP lacks a reliable protocol (it is based on IP) for distribution. In contrast, the CR-LDP uses TCP for reliable delivery but it is subject to all the problems associated with the performance of TCP under increasing traffic load.

We propose a lightweight signaling protocol called the SBoX Reservation Protocol (SBRP) for establishing paths in the reserved-net. SBRP protocol sends messages only along the pre-computed reserved-paths, and it does not have to try alternate paths in the reserved-net, since constructing a virtual premium network (i.e., the reserved-net) ensures that for any reserved-link (i, j), $\sum_k f^k_{i,j} \leq c_{i,j}$. Figure 5 describes the operation of SBRP in detail.

The SBoX Reservation Protocol is essentially a 3-way handshake. In order to reserve a path from SAP_x to SAP_y, SAP_x first sends a *REQ* message along the reserved path to SAP_y. All intermediate SBoX routers mark the new path. In the second phase, SAP_y responds to SAP_x by sending a *REP* message, which commits the path in the intermediate SBoX routers. To complete the reservation, SAP_x should either start sending data or send a *CONF* message to SAP_y. All intermediate SBoX routers and SAP_y employ timers to release the reservation if no follow up message arrives in time.

Notice that QoS requirement is expressed simply as the amount of the bandwidth. The traffic specifications such as burst size burst length are omitted, since they are controlled at the

1. SAP_x makes a request of R bps to a SAP_y for the corresponding commodity-flow by
   1.1 sending $REQ(Label, R, seq\#)$ message.
   1.2 starts a timer for reply.
2. each SBoX router on the reserved-path
   2.1 marks the forwarding table for rate R and *Label* and
   2.2 appends its ID to the REQ message.
   2.3 marking is not a commitment but a tentative state. It is associated with a timer and if no reply (REP) message is received, then there will be timeout and the tentative commitment will be reset.
3. upon receiving the REQ message, the SBoX server at SAP_y
   3.1 sends $REP(Label, R, seq\#)$ message back by reversing the reserved-path recorded into the REQ message.
   3.2 sets a timer for confirmation (CONF) message
4. each SBoX router on the path
   4.1 identifies the REQ/REP using the Label,
   4.2 *commits* for the rate R that REP message carries,
   4.3 unmarks the REQested rate
   4.4 starts a timer for confirmation
5. upon receiving the REP message, the SAP_x confirms the reserved-path by
   5.1 sending a $CONF(Label, R)$ message to SAP_y or
   5.2 starts transmitting data.
6. if the timer in SAP_x times out for the REQ message then it resends a REQ message with an incremented seq #.
7. if an SBoX router receives a redundant REQ message and
   7.1 it has marked the forwarding table then assumes that previous REQ was lost at the segment between itself and SAP_y
   7.2 it has also committed then assumes that REP got loss at the segment between itself and SAP_x
   7.3 takes no action and waits the edge routers to response.
8. upon committing to a REQ message if an SBoX router receives no CONF message or data, then it times out and releases the commitment.
9. each router maintains the commodity-flow until a TEARDOWN message is received.

Figure 5: SBoX reservation protocol

SAPs by SBoX servers. Each commodity flow is associated with an MPLS label which can be global within the AS [2]. Global labels can be assigned by using the distinct SAPs identifying each commodity-flow and packets of the commodity-flow are marked by this MPLS label.

[2]Note that 20 bits from the 4-byte MPLS label can support 2^{20} commodities which corresponds to approximately 10^3 SAPs within an ASP. If the number of SAPs exceeds such number, then labels must be swapped and cannot be global.

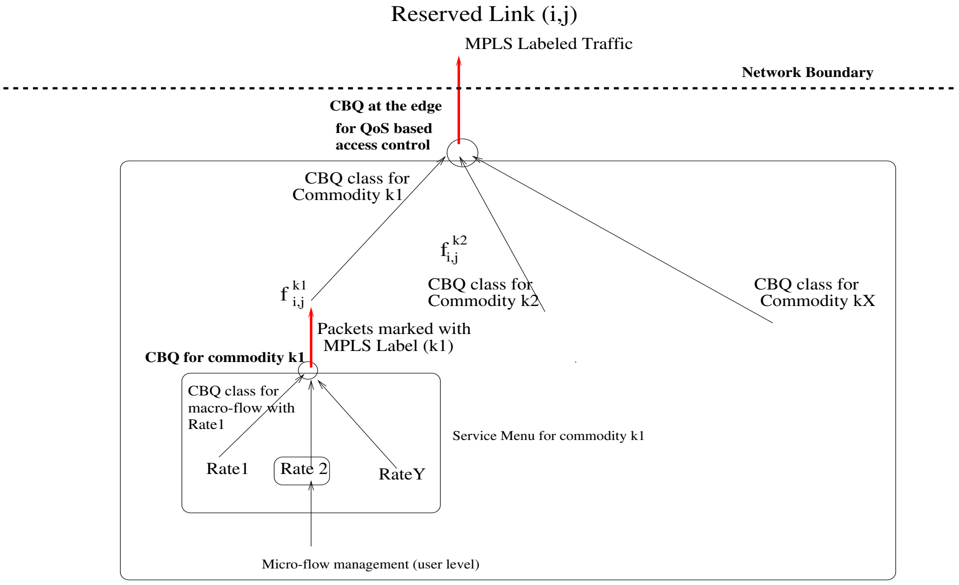

Figure 6: CBQ hierarchy and labeling maintained by an SBoX Server

4 SBoX Architecture

The SBoX hardware architecture comprises two components: (i) SBoX servers, and (ii) SBoX routers. SBoX servers deal with admission control and management issues at the network edge. SBoX routers are the interior network nodes that perform only label switching of macro flows for QoS based routing. We will describe the SBoX servers and routers in detail in this section.

4.1 SBoX Servers

The SBoX architecture deals with the complexity of QoS support by delegating it to the SBoX servers. An SBoX server is an *intelligent edge-router* that resides at network ingress/egress points (i.e., SAP) and provides QoS based network access. The SBoX server is a programmable device with an open architecture, and it is in charge of managing the resources and requests from SAPs. It separates QoS traffic from best-effort and labels the QoS traffic for MPLS routing inside the network. The SBoX server uses hierarchical traffic management mechanisms which can be configured dynamically to adapt to the changing policies of an ISP. We note that since the adjacent AS are treated as SAPs, their resource management is also handled by SBoX servers.

4.1.1 Flow Management and Labeling

SBoX server is a configurable device which has hierarchical traffic management capability. Figure 6 depicts the hierarchical management of a reserved-link (i, j) by an SBoX server. At the top of the hierarchy, the bandwidth of (i, j) is shared by multiple commodity-flows whose bandwidth allocation is determined by the optimization problem presented in section 3.1. Link (i, j) needs to be managed to ensure the protection of commodity-flows from best effort and the protection of commodity-flows from each other. The second level of the hierarchy deals with the management of the components of each commodity-flow, namely the macro flows. Each commodity-flow is divided into a set of macro-flows, which are offered as SLAs to users of a SAP for this commodity. Thus, each SLA provides a predefined service rate which is offered by a *service menu*. In the bottom of the hierarchy, macro-flows are managed in application level flow (i.e., micro-flow) granularity. We will explain the service menus and management of macro-flows in the next section.

SBoX servers performs the IP to MPLS mapping at the commodity-flow level. As shown in Figure 6, the SBoX server maintains a CBQ scheduling mechanism for each commodity-flow that uses edge (i, j). Each commodity is considered as a class and classes are strictly isolated from each other (i.e., fire-walled). The resource sharing between best-effort and QoS traffic is based on the movable boundary scheme as explained above. Packets of each commodity are filtered and scheduled based on their MPLS labels.

An SBoX server also maintains billing and accounting information at different levels of granularity. It can provide Call Description Record (CDR) type of information at the commodity-flow or at the macro-flow level. We will explain billing and charging issues in the next section.

4.1.2 Admission Control with SLAs

The committed traffic is accepted to the network through admission control, which ensures that bandwidth allocation satisfies the service descriptions of user requests. An important part of admission control is monitoring and managing the available bandwidth for each commodity.

The SBoX architecture provides certain data rates called *service rates*, that are a multiple of a base rate (e.g., 1Mbps, 5Mbps, 10Mbps, 20Mbps) in a service menu. Each data rate represents a different SLA. Users sign a contract with their provider by picking an SLA from a service menu.

As shown in Figure 6, each SLA corresponds to a macro-flow. It is left to the users/enterprises to manage the access to the pipe. There are several commercial products available to differentiate business critical traffic based on enterprise policies (e.g., Xedia's Accesspoint, Packeteer's Packetshaper, Checkpoint's Flood Gate, Allot's AC 200-330).

The challenge is how to decide which service rates to offer and how many requests to grant for each (macro-flow) rate. We believe that the solution should have low overhead and also address the network economics. For example, each service rate in the service menu can be assigned a number of *tokens* (permits) during provisioning. The weighted sum of the tokens is equal to the total amount of reserved traffic that the provider is willing to accept (where the weights represent the data rates). Each SAP and BRs are given a set of tokens based on (i) the expected number of connections, and (ii) the bandwidth demands between them. Thus, each SAP/BR can make a local decision how much traffic to accept to the reserved-net using the token/permit pool it has. For example, we refer to Figure 6. Suppose that the SAP is given x tokens of 512Kbps and y tokens of 1Mbps for commodity k_1. Then this SAP can accept at most $x + y$ requests for this commodity such that $x512 + y1000 \leq f_{i,j}^{k_1}$ Kbps.

There are two advantages for providing SLAs with token based admission control. First, it reduces the traffic management overhead at the commodity-flow level. Since users must pick an SLA from a service menu in advance, the policing, shaping and scheduling in the network can be set ahead of time, even before the SLAs are offered. In contrast, if we replace the predetermined rates with a dynamic management scheme, then requests for macro-flows must be accommodated on demand basis. This requires setting the traffic filters and shapers on demand as well. Thus, it

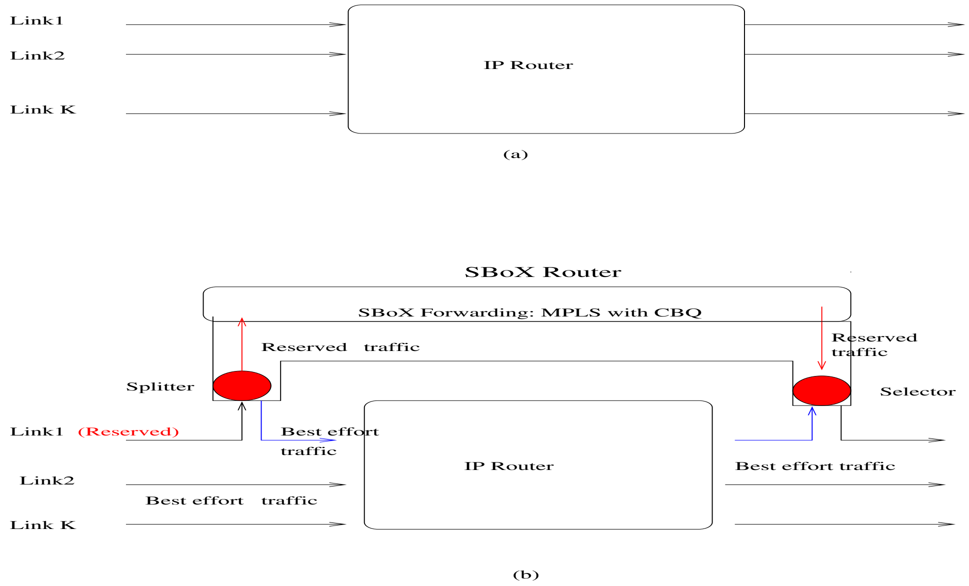

Figure 7: SBoX router operation

may increase the processing time for requests and response delay. Second, token based admission control enables efficient billing and accounting mechanism to ISPs. For example, each token can have a price based on the rate that it is associated with. Thus charging can be reduced to the token holding time multiplied with the price of this token. The price of a token can be changed based on the demand for bandwidth. For example a discount price can be offered during off-peak hours while price can be increased during peak hours.

The disadvantage of picking from a predefined menu of service rates is that it may result in over-allocation and bandwidth waste. However, we believe that the advantages of simplified management, billing and reduced run-time complexity outweigh the disadvantages of the proposed approach.

4.2 SBoX Routers

An SBoX router is a programmable, stand-alone, label switching router with a limited number of network interfaces. An SBoX router cannot perform IP-based routing and its routing capability is limited only to label switching with CBQ scheduling.

Since legacy routers are vertically integrated closed boxes, an SBoX router may be deployed as an add-on device [3]. An SBoX router controls the reserved links, identified by the reserved-net, incident to a legacy router, as depicted in Figure 7. Figure 7 also shows that QoS traffic has precedence over best-effort traffic. Best-effort traffic uses either links that are not a part of the reserved-net or the idle portion of reserved link. Combination of SBoX and best-effort router provides functionality found in a *dual-switch* [Kum74, Zaf74, OY94, NBS95].

The number of network interfaces and the line speeds that an SBoX supports are dictated

[3]Certainly, it would be the best if an SBoX can be transplanted into a legacy router; however current routers do not provide such modular and open architectures.

• *Fget:* intercept the traffic into the *reserved node*;
• *Fchk:* check the header to determine its type;
• *FcuthruS:* IF the packet belongs to a macro flow then cut through to the SBoX;
• *FcuthruR:* ELSE forward the packet to the same input link incident to the router.
• *Fdrop:* (optional. The best-effort traffic may be deflected back to the router instead of dropping.) if the packet is not a IP control packet.
• *Fnext:* check the label and the outgoing link;
• *Fvc:* perform MPLS compliant virtual circuit switching with per commodity-flow CBQ scheduling.
• *Fblock:* prevent the router to access the output link if the SBoX has a packet to be transmitted to it as well.

Figure 8: SBoX router actions

by the network economics and considered to be outside the scope of this paper. However, such constraints can be included in the formulation of the optimization problem to limit the number of reserved links incident to a legacy router that can be in the solution. The main actions of the SBoX router are summarized in Figure 8.

The actions in Figure 8 are performed by two network interface modules called the *splitter* and the *selector*. The splitter is functionally a 1x2 demultiplexer. It intercepts all incoming packets, examines their header information, and decides where to forward the packet (to an outgoing link for MPLS cut-through, or to an input link of the best-effort router). The splitter performs the *Fget*, *Fchk*, *FcuthruS*, *FcuthruR*, and *Fdrop* actions as defined in Figure 8. The splitter stores the initial portion of incoming packets in a buffer until the label is processed. If the packet does not carry a label, then it is considered as a best-effort, and it is forwarded immediately to the best-effort router without waiting and storing the rest of the packet. The size of the splitter's buffer is a function of the link rate and the splitter processing speed.

The selector handles two kinds of contention on the outgoing reserved link: (i) contention between best effort traffic sent by the best effort router and reserved traffic on the same reserved link. This contention is handled by an arbitration circuit, which ensures that as long as there is data in the SBoX's output buffer, the reserved link cannot be used by the legacy router. (ii) contention caused by reserved traffic that is sent to the same outgoing reserved link from several incoming links. This contention is always temporary, since the reservation algorithm ensures that the reserved traffic never exceeds the capacity of the outgoing reserved link. In this case, the selector keeps the outstanding reserved traffic in a buffer of a limited size, which is proportional to the size of the largest packet and the number of incoming reserved links. The first kind of contention is resolved by a 2x1 multiplexer circuit, which chooses between the SBoX and the best effort router. The multiplexer selects the SBoX as long as it has data. The splitter and selector reside inside the SBoX and their functions can be combined in an interface card with two ports and a limited processing capability.

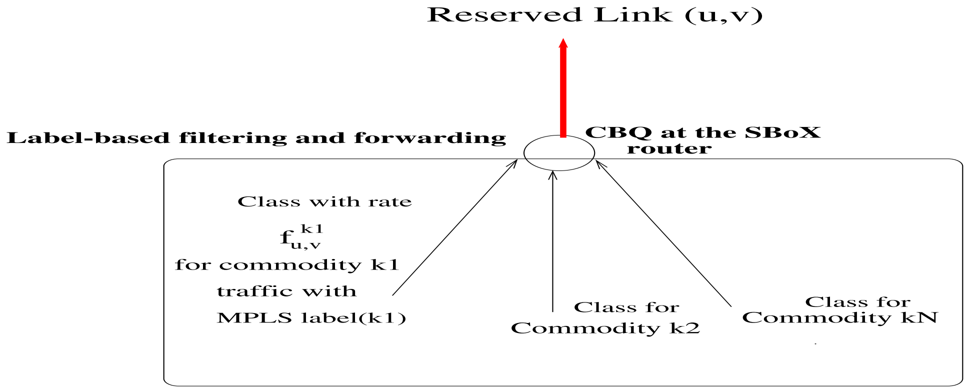

Figure 9: CBQ class hierarchy in an SBoX router

4.2.1 Flow Management and Forwarding

While the best-effort router performs IP-based routing, packet forwarding in an SBoX is based on label switching using MPLS. Upon entering the network, QoS traffic is handled exclusively by the SBoX routers. Thus it bypasses all best effort traffic. Each SBoX router has a *forwarding table*, associated with each link. An entry in the forwarding table indicates the outgoing link and the reserved bandwidth for a commodity-flow identified by its MPLS label.

The protection and enforcement of service guarantees is done with CBQ as shown in Figure 9.

5 SBoX Testbed Prototype

The testbed for SBoX project consists of seven Intel Pentium III PCs and a Cisco 2621 router connected with the topology depicted in Figure 10. All the links except the bottleneck one (which is 10Mbps) are 100Mbps Ethernet crossover links.

Four of the seven PCs are running Windows NT. Two of them run the the Lucent MPEG-4 video streaming server (`mpegsrvr`) and its client (`mpegclnt`). The other PCs run the Lucent Cineblitz [MNO+97] video streaming server (`cnbzsrvr`) and its client (`cnbzclnt`). The remaining three PCs are running Linux with MPLS and CBQ support compiled into the kernel. Two of these PCs serve as SBoX servers (`sboxsvr1` and `sboxsvr2`) and remaining PC is the SBoX router (`sboxrtr1`). The Cisco 2621 router (`sboxrtr2`) plays the role of a conventional IP router for best-effort traffic.

One MPEG video stream of approximately 4Mbps flows from `mpegsrvr` to `mpegclnt`. Multiple Cineblitz video streams, each about 1.5Mbps, flow from `cnbzsrvr` to `cnbzclnt`. These video streams are the premium QoS traffic and can be selectively protected with guaranteed bandwidth. Premium traffic is label-switched across the network between `sboxsvr1` and `sboxsvr2`. In addition, best-effort UDP traffic, with adjustable bandwidth consumption, flows from `sboxsvr1` to `sboxsvr2` as the background traffic to create congestion. The background traffic is not label-switched; it is routed through regular IP routing.

The background traffic entering `sboxrtr1` is diverted to the Cisco router and then looped back to `sboxrtr1` before it is routed to its final destination. This operation simulates (i) the splitter and selector operation, and (ii) the ability of SBoX to inter-operate with existing IP

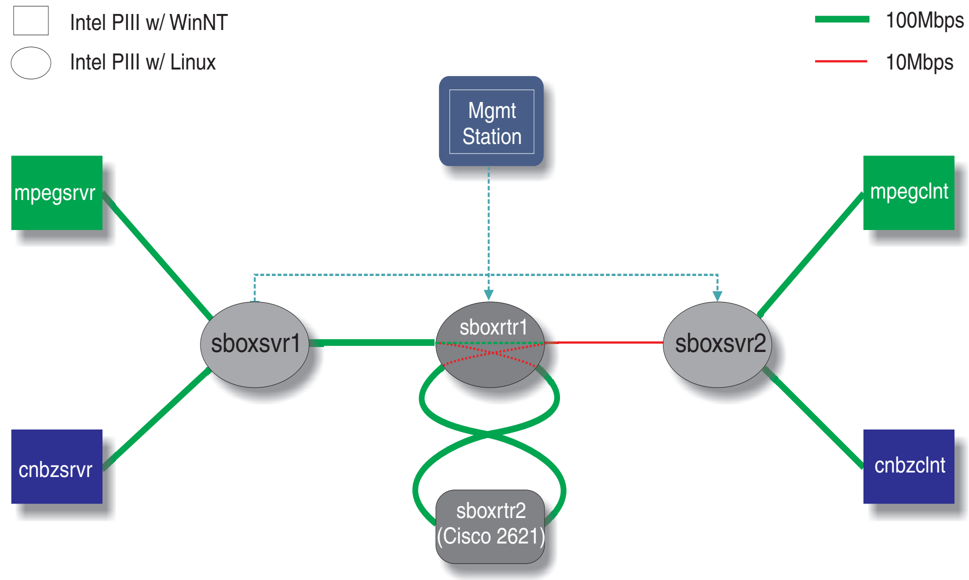

Figure 10: SBoX prototype testbed.

routers without affecting the existing IP traffic. Note that all traffic goes through the bottleneck link, the 10Mbps link between `sboxrtr1` and `sboxsvr2`.

5.1 SBoX Software

This section describes the components of the SBoX software in the testbed.

5.1.1 SBoX Daemon and SBoX Management Client

The SBoX daemon is a user-level program that runs on all SBoX servers and routers. The SBoX daemon listens on a well known port and responds to configuration, control, and management requests from a management station through the SBoX Management Protocol (SBMP). SBMP allows the management station to:

1. configure token tables on SBoX servers.
2. configure routing tables on SBoX servers and routers.
3. configure MPLS on SBoX servers and routers
4. configure CBQ on SBoX servers and routers

The SBoX daemon also implements the SBoX Reservation Protocol (SBRP) as described in Section 3.4 for setting up an end-to-end QoS guaranteed path and distributing the labels. Figure 11 depicts the functional components of an SBoX daemon. The SBoX management client is a command-line driven tool that can be used by a management station to communicate with SBoX daemons. The tool is built on top of the SBoX Management API which implements the

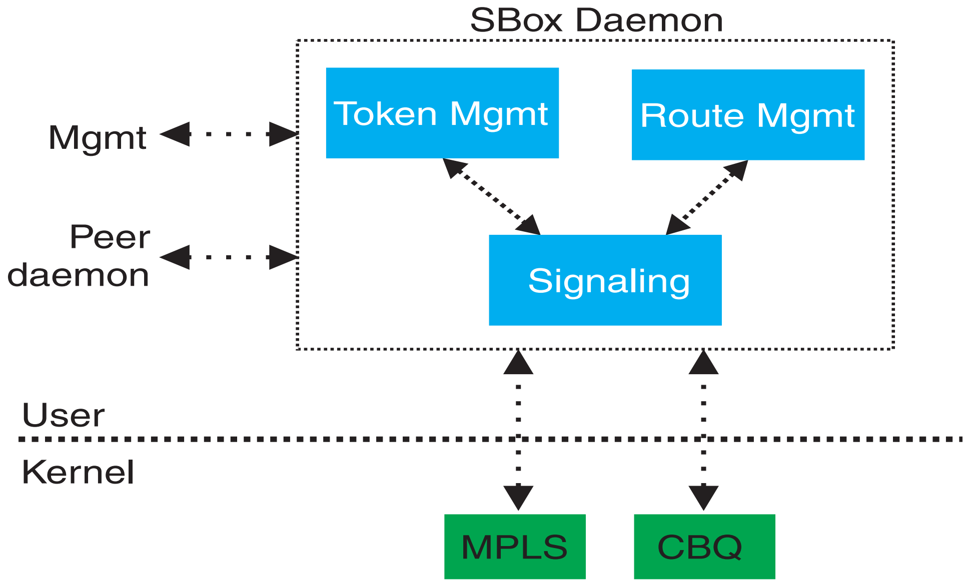

Figure 11: SBoX software components

SBoX Management Protocol. The separation of the tool and the API allows more sophisticated management tools such as a GUI front-end to be built easily on top of the API. The following is a list of a few typical API calls.

1. *token_set, token_get, token_del.* On an SBoX server: assign, list, and remove tokens, respectively, for links that comprise the path from the (source) SBoX server to another (destination) SBoX server.

2. *route_set, route_get, route_del.* On an SBoX server or an SBoX router: install, list, and remove routing information, respectively, for traffic flowing between two end-stations that will traverse the SBoX server or the SBoX router.

3. *resv_rqst.* On an SBoX server: initiates the SBoX Reservation Protocol for reserving bandwidth for a macro-flow.

5.1.2 SBoX Client Reservation Web Interface

Clients on end-stations such as `mpegclnt` and `cnbzclnt` use a simple Web form served from their SBoX servers to make bandwidth reservations for the video stream between the video server and the client. Note that the reservation can be used for any type of traffic, not just videos. The Web form executes the CGI program (written in C) on the SBoX server, which dynamically generates another web form with options of current available bandwidth for different video servers. When the client commits the reservation, the CGI program on the SBoX server initiates the SBoX Reservation Protocol by calling the *resv_rqst* call of the SBoX Management API.

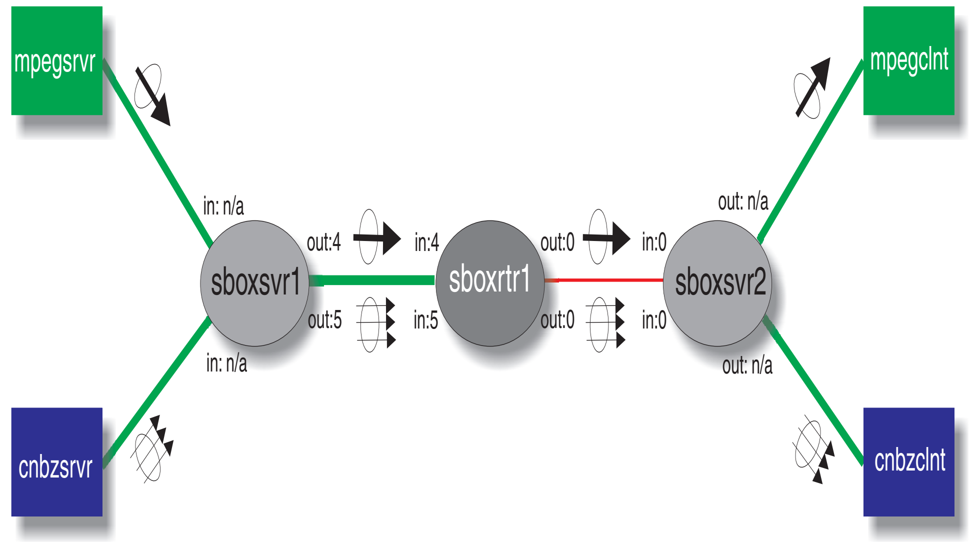

Figure 12: MPLS setup

5.2 MPLS and CBQ Settings

This section describes the implementation of MPLS and CBQ in the testbed and reservations used in the experiments.

5.2.1 MPLS Setup

The MPLS support for Linux is version 0.011 of MPLS/LDP implementation by James R. Leu from the University of Wisconsin. Labeling and switching for the premium video traffic is currently statically configured via automated scripts using `mplsadm`. Implementation of dynamic MPLS configuration by the SBoX daemon is in progress. Figure 12 shows the label assignment for the video traffic.

A label-switched path corresponds to a commodity-flow. Note that although there can be multiple types of micro-flows, such as Telnet, FTP, HTTP, etc., flowing between the two end points that fall into the same macro-flow, all the traffic of all those micro-flows is assigned the same MPLS label. This is the main mechanism of SBoX architecture to deal with scalability issues of flow management.

5.2.2 CBQ Setup

The CBQ support for Linux is the kernel implementation on Linux version 2.3.3. Setting up CBQ consists of two parts. The initialization part sets up the queuing discipline and the root class. It is done automatically at system startup time via a script using `iproute` tool. The dynamic part sets up the class and filter for each premium traffic. It is done by the SBoX daemon through SBoX Reservation Protocol, which also makes use of the `iproute` tool. The Lucent Cineblitz video streaming protocol requires certain amount of bandwidth in the opposite direction of the video stream, i.e., from `cnbzclnt` to `cnbzsrvr`. Since the 10Mbps link between `sboxrtr1` and

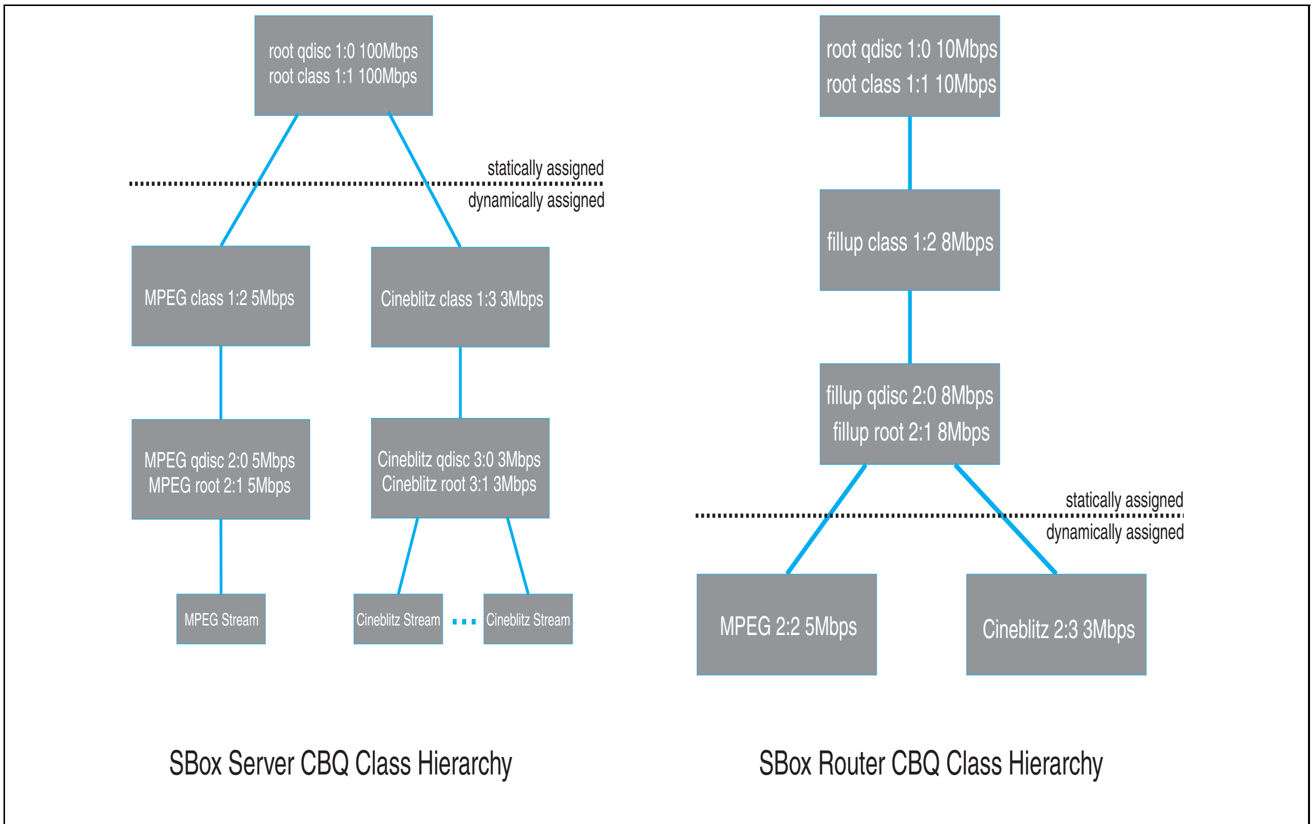

Figure 13: CBQ class hierarchy

`sboxsvr2` is a simplex cross-cable, we created a "fill up" CBQ class of 8Mbps for all the traffic going from `sboxrtr1` to `sboxsvr2`. This effectively makes the 10Mbps link appear to be an 8Mbps link for traffic going from `sboxrtr1` to `sboxsvr2` and leaves 2Mbps for the traffic going in the opposite direction. The CBQ class hierarchy on `sboxrtr1` is shown in Figure 13.

Note that Figure 13 depicts the class hierarchy used by `sboxrtr1` inside the network. It contains only two types of traffic, Lucent MPEG video stream and the Lucent Cineblitz video stream, regardless of the number of micro-flows within each type of traffic. The SBoX servers at the network boundary may employ additional classes and filtering rules (such as matching port numbers) to control and manage micro-flows within a macro-flow.

5.3 Experimental Results

We experimented with the performance of the SBoX architecture in the testbed described above. The SBoX router `sboxrtr1` manages two commodity-flows and it is connected to the `sboxsvr2` with a full-duplex bottleneck link with capacity 10 Mbps as mentioned above.

Figure 14 shows the behavior of the MPEG-4 and Cineblitz reserved traffic as both sources increase their sending rate. The horizontal lines are the maximal receive rate of each traffic type. Note that the maximal receive rate of the MPEG-4 traffic is close to its reservation (5Mbps) and the maximal received rate of Cineblitz traffic is also close to its reservation (3Mbps). This experiment indicates that MPLS compliant QoS traffic belonging to both commodities is isolated from the other commodity traffic. Furthermore, their total sending rate of both commodities is

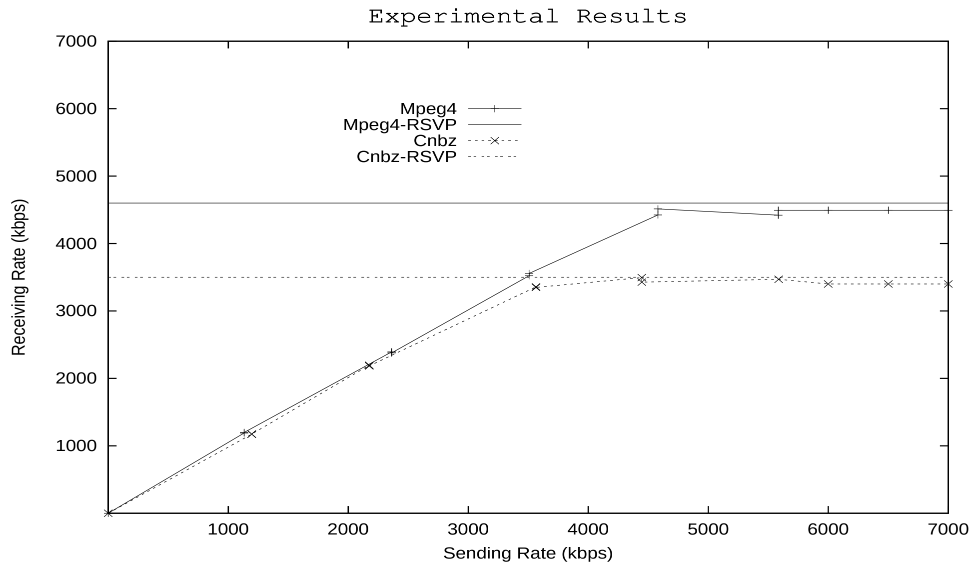

Figure 14: Behavior of reserved traffic

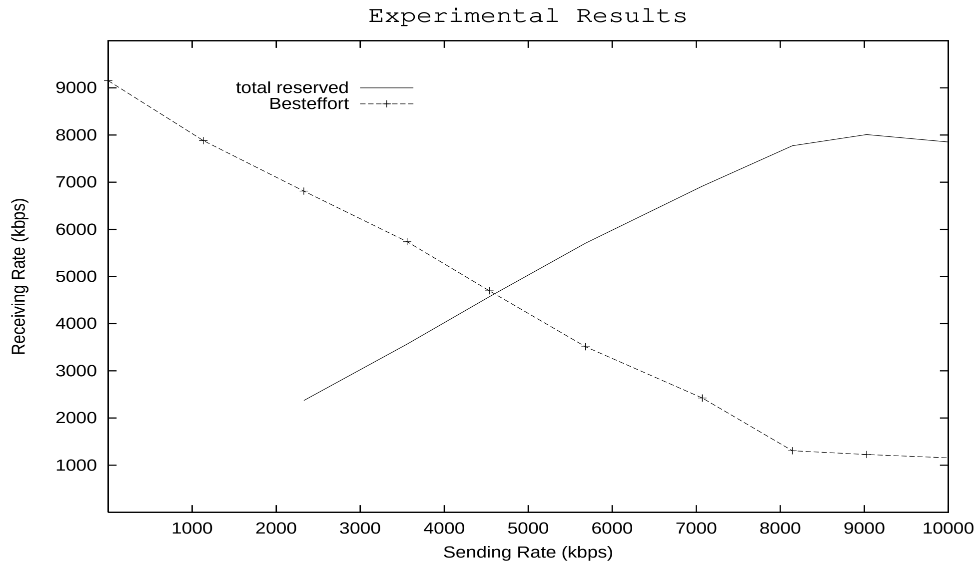

Figure 15: Behavior of best effort and reserved traffic under increasing contention

bounded by 8Mbps, which is the total reserved bandwidth on this bottleneck link.

Figure 15 shows that as the total amount of reserved traffic increases, the best-effort rate is reduced to the unreserved portion of the link bandwidth traffic, which consumes all of the available unreserved bandwidth. This behavior verifies that the *movable boundary* scheme works efficiently for providing high utilization. Furthermore it also shows that QoS traffic is protected from the best-effort.

6 Summary and Conclusions

In this work we presented a hybrid architecture to provide deterministic QoS on the IP-based networks called Smart Box (SBoX). SBoX combines the best features of Diffserv, Intserv, and MPLS proposals. The core of SBoX is precise provisioning and admission control algorithms. These algorithms determine a virtual premium network (the reserve-net), that routes QoS traffic. The traffic in the reserve-net consists of commodity flows, which are the aggregated traffic between service access points. Each commodity flow consists of macro-flows, which contain the users' traffic. Macro flow traffic is managed by service level agreements. The macro flows consists in turn of micro-flows, which are the traffic generated by particular applications.

The SBoX architecture defines two types of active components: SBoX servers, which are located at the network boundary, and SBoX routers, which are located at the network interior. SBoX servers perform the complex micro-flow management at the network boundary. SBoX servers also enforce the service level agreements on macro-flow traffic, and attach MPLS labels to the commodity flow traffic. SBoX routers are add-on label switching routers, which are combined with existing best-effort routers to provide label switching capabilities in addition to best-effort routing. The SBoX routers handle the MPLS labeled traffic, and they delegate the best-effort traffic to the existing router.

We described an experimental testbed that illustrates the operation of the SBoX architecture. We presented the results of several experiments on the testbed, which indicate that the SBoX architecture provides QoS guarantees under contention. Moreover, the testbed indicates that the SBoX router can indeed be combined with existing best-effort routers to provide QoS routing.

References

[ACL94] G. R. Ash, K. K. Chen, and J-F. Labourdette. Analysis and design of fully shared networks. *Proc. of 14'th International Teletraffic Congress*, pages 1311–1320, 1994.

[Ash95] G. R. Ash. Dynamic network evolution, with examples from at&t's evolving dynamic network. *IEEE Communications Magazine*, 33(7):26–39, 1995.

[BW99] A. Basu and Z. Wang. Fair bandwidth allocation for differentiated services. *in Protocols for High Speed Networks VI, J. Touch and J. Sterbenz ed. Kluwer Academic Publishers*, 1999.

[CFZ94] I. Chlamtac, A. Farago, and T. Zhang. Optimizing the system of virtual paths. *IEEE Trans. on Networking*, 2(6):581–587, Dec. 1994.

[Cla97] D. Clark. Internet cost allocation and pricing. *Internet Economics, L. W. McKnight and J. P. Bailey (eds)*, pages 215–252, 1997.

[CW97] D. Clark and J. Wroclawski. An approach to service allocation in the internet. *Internet Draft, draft-clark-diff-svc-alloc-00.txt*, July 1997.

[Dov91a] R. D. Doverspike. Algorithms for multiplex bundling in a telecommunications network. *Operations Research*, 39:925–944, 1991.

[Dov91b] R. D. Doverspike. A multi-layered model for survivavility in intra-lata transport networks. *Proc. IEEE GLOBCOM'91*, pages 2025–2031, 1991.

[ea98] S. Blake et al. A framework for differentiated services. *draft-ietf-diffserv-framework-0.1.txt*, October 1998.

[ea99a] B. Jamoussi et al. Constraint-based lsp using ldp. *Draft-ietf-mpls-cr-ldp-03.txt*, September 1999.

[ea99b] I. Stoica et. al. Per hop behaviours based on dynamic packet states. *internet draft. http://www.cs.cmu.edu/ istoica/DPS/draft.txt*, February 1999.

[For96] ATM Forum. P-nni specification, version 1.0. March 1996.

[HRW92] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem.* Annal of Discrete Math. 53, North-Holland, 1992.

[IN98] J. Ibanez and K. Nichols. Preliminary simulation evaluation of an assured service. *internet draft : draft-ibanez-diffserv-assured-eval-00.txt*, August 1998.

[Kum74] K. Kummerle. Multiplexer performance for integrated line and packet switched traffic. *Int. Conf. Comput. Commun. Record*, pages 507–515, August 1974.

[Lee95] W. C. Lee. Topology aggregation for hierarchical routing in atm networks. *Comput. Commun. Rev. (USA)*, 25(2):82–92, 1995.

[ML97] H. Michiel and K. Laevens. Teletraffic engineering in a broad-band era. *Proc. of IEEE*, 85(12):2007–2033, December 1997.

[MMR96] D. Mitra, J. A. Morrison, and K. G. Ramakrishnan. Atm network design and optimization: A multirate loss network framework. *Proc. IEEE INFOCOM'96*, pages 994–1003, 1996.

[MNO+97] C. Martin, P.S. Narayan, B. Ozden, R. Rastogi, and A. Silberschatz. The fellini multimedia storage system. *Journal of Digital Libraries*, 1997.

[NBS95] A. Nguyen, N. Bambos, and M.H. Sherif. Adaptive (t1, t2)-multiplexing transmission schemes for voice/data integrated networks. *IEEE Syp. Comp. and Communications*, pages 430–435, 1995.

[NJZ97] K. Nichols, V. Jacobson, and L. Zhang. A two bit differentiated services architecture for the internet. *Internet Draft, draft-nicolas-diff-svc-arch-00.txt*, November 1997.

[OY94] Y. Ofek and M. Yung. The integrated MetaNet architecture: A switch-based multimedia LAN for parallel computing and real-time traffic. *IEEE INFOCOM'94*, 1994.

[RVC98] E. C. Rosen, A. Viswanathan, and R. Callon. Multi protocol label switching. *draft-ietf-mpls-arch-02.txt*, July 1998.

[SPG97] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. *rfc2211*, September 1997.

[SSZ98] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks. *Proc. of ACM SIGCOMM'98*, August 1998.

[SW97] S. Shenker and J. Wroclawski. General characterization parameters for integrated services network elements. *rfc2215*, September 1997.

[SZ98] I. Stoica and H. Zhang. Lira: An approach for service differentiation in the internet. *Proc. of NOSSDAV'98, Cambridge, England*, 1998.

[Wro97] J. Wroclawski. Specificiation of the controlled-load network element service. *rfc2211*, September 1997.

[Zaf74] P. Zafiropolo. Flexible multiplexing for networks supporting line switched and packet switched data traffic. *Int. Conf. Comput. Commun. Record*, pages 517–523, August 1974.