Design and Implementation of a QoS Capable Switch-Router

E. Basturk[*], A. Birman, G. Delp, R. Guérin, R. Haas, S. Kamat
D. Kandlur, P. Pan, D. Pendarakis, V. Peris, R. Rajan, D. Saha, D. Williams

IBM T. J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598

Abstract

An important challenge for the future growth of the Internet is to design routers that can forward the exponentially increasing volume of traffic, and at the same time provide the service differentiation needed by new applications. In this paper, we describe the architecture, implementation, and initial experiences with a system designed to meet this challenge. This system, which we call a QoS capable Switch-Router (QSR), combines the salient features of switching and routing technologies to provide high throughput and support the different classes of service being defined by the IETF. It consists of a core (ATM) switch fabric connecting intelligent adapters, each capable of both routing and switching packets. A control engine is responsible for routing, RSVP signaling, and resource management. We have built a prototype network of 3 systems connected to several UNIX hosts, and have conducted preliminary performance measurements on this network.

1 Introduction

The rapid expansion of the Internet has strained the capabilities of the current Internet infrastructure. This is demonstrated by the growth in traffic volume at key Internet exchange points [1]. Emerging audio and video applications, such as Internet Telephony and RealAudio, place further demands on this strained infrastructure. These new applications, many of which do not adhere to the congestion control philosophy of the Internet, have the potential to adversely impact the performance of the network. From the viewpoint of the Internet backbone, an important challenge for the future growth of the Internet is to design routers that can forward the exponentially increasing volume of traffic, and at the same time provide service differentiation for certain traffic types. The latter is especially important to sustain the growth of the Internet as a commercial network. In this paper, we describe the architecture, implementation, and initial experiences with a system that has been designed to meet this challenge. It is based on the Integrated Services model [5] and associated standards being developed by the IETF, namely the Resource Reservation Protocol (RSVP) [6], the Controlled-load service definition [17], and the Guaranteed service [16].

[*]E. Basturk is now with Pluris Networks.

In order to satisfy the potentially conflicting objectives of increasing raw forwarding performance and providing fine-grained control on packet flows for service differentiation, our system draws upon the respective strengths of ATM-style label switching (layer 2 forwarding) and routing (layer 3 forwarding). It is illustrative to understand the differences between layer 2 and layer 3 forwarding, since these differences played a key role in influencing system design. The basic operations of layer 2 and layer 3 forwarding are surprisingly similar. Both involve a look-up based on a pattern contained in a specific field of a packet header, whose outcome determines the course of further processing of the packet, namely, where it is to be sent and with what level of service. The main difference is that a label switch is required to accept and forward only those labels that have been previously assigned, whereas an IP router is required to accept all IP destination addresses. The latter, while providing more flexibility in routing, necessitates a more complex "longest prefix match" lookup on destination address of an IP packet (see [14] for a more comprehensive discussion on this topic). When the basic forwarding function is to be enhanced for QoS support and service differentiation, additional classification and scheduling functions are required which contribute significantly to the per-packet processing overhead. For instance, the classification function is based on source and destination addresses and the transport level port numbers embedded in the packet. Hence, our design focus is on leveraging the benefits of switching for forwarding packets which require some level of service differentiation. This includes packets from individual flows with specific QoS requirements, e.g., RSVP flows, and from aggregate streams, e.g., to a given subnet or set of subnets, for which we want to provide some level of service provisioning.

Our system, which we call a QoS-capable Switch-Router (QSR), consists of a core (ATM) switch fabric to which a number of *intelligent* adapters are attached, with each adapter also supporting an interface to an external (ATM) OC-3 link. The adapters are capable of routing and switching packets, and include hardware support for providing different levels of quality-of-service. The switch and the intelligent adapters are controlled by a "Control Engine" that resides in a separate adapter and is responsible for running the different routing protocols that the system implements, and for handling all aspects of resources management, including signaling which is supported through the RSVP protocol. The QSR adapters are further distinguished by their functions into *port* and *trunk* adapters (see Figure 1). Trunks interconnect adjacent QSRs and are optimized for speed and performance. Ports provide flexible access functions, such as the packet classification functions needed at the periphery of the network. These classifier functions, which precede the regular IP forwarding loop, identify packets that are to be afforded special service and place the matched packets onto particular layer-2 connections.

There has been a significant amount of recent work on topics related to the exploitation of switching for IP forwarding. However, these efforts have focused primarily on handling best-effort traffic [7, 15]. Some of these techniques could, in the future, be employed in the QSR to off-load best-effort IP forwarding. In the current configuration of OC-3 links, based on the forwarding performance observed for standard IP forwarding (see Section 6 for preliminary performance results), we determined that these techniques were not required. We believe that the QSR system described here is original in terms of the scope of what it addresses and its focus on the use of switching for service differentiation. To the best of our knowledge, the system that comes closest to what we have built in terms of overall structure and functionality is the integrated cell switch router from Washington University [3] that was independently proposed.

In the rest of the paper, we provide details on both the QSR design and the different components it relies on, and present some initial results on the performance and capabilities of the implementation. In Section 2, we provide a short overview of the overall system architecture, and highlight its main functions and components. Section 3 is devoted to the control functions implemented to support QoS. These include resource management and signaling (RSVP). Section 4 describes the main data paths that the system supports, i.e., routed (Section 4.1) and switched (Section 4.2). Section 5 describes a new "provisioned IP" service (PSIP), which provides resource reservation for traffic between specific ports across a QSR network. Section 6 presents various test cases and applications that have been used to obtain early feedback on system behavior and performance. Finally, Section 7 summarizes some of the experiences gathered from this work and lists a number of extensions and enhancements that are currently being worked on.

2 System Overview

Providing service differentiation requires that we support signaling facilities that convey traffic and QoS information across a network of QSR boxes. In the Internet model, this function is provided by the RSVP protocol, and the QSR fully supports this function. Moreover, other services such as the "provisioned IP" service (see Section 5) are provided using RSVP to reserve resources on behalf of aggregate traffic streams. Support for RSVP translates into significant control resources for processing and storage in order to efficiently handle a large number of flows.

In order to adequately support best-effort routed traffic and QoS traffic, the system must provide both store-and-forward (routing mode) and cut-through (switching mode) forwarding. In the store-and-forward mode, an efficient interface is required between the packet storage area and the processing unit responsible for making forwarding decisions for the packets. An intelligent interface permits the processing unit to "peek" at the relevant packet headers and avoid memory intensive copy operations. Furthermore, in addition to the traditional IP address look-up, we must provide a range of "classifiers" to identify appropriate flows for forwarding onto switched connections. In the cut-through mode, which is to be used to provide service differentiation, there is need for both cell-based and frame-based forwarding. Cell-based forwarding minimizes latency and storage requirements and is prefered for unicast flows, while frame-based forwarding is needed to allow merging of flows, thereby circumventing the cell interleaving limitation of ATM.

A high level illustration of the structure of the QSR system is shown in Figure 2. As shown in the figure, the QSR system consists of a core ATM switch to which a number of intelligent line interface cards are attached. The links currently supported are ATM OC-3 links. The core ATM switch provides high performance connectivity for unicast and multicast flows as required by our design goals. The adapters, referred to as forwarding engines (FE), are responsible for the main data path functions. These include layer 2 and layer 3 forwarding functions, the enforcement of different levels of QoS through scheduling of packet transmissions, and management of buffer space. The FEs are controlled by a separate adapter known as the control engine (CE). RSVP and all routing protocols (OSPF and MOSPF in the context of our current implementation) run in the CE. The CE is also responsible for the resource management functions needed to support quality-of-service.

The internal architectures of the FE and the CE are illustrated in Figure 3. As shown in the figure, both CE and FE consist of a PowerPC 603e based processor complex connected to an ATM segmentation and reassembly (SAR) subsystem (CHARM subsystem [8]) via a PCI bus interface. The 603e processor in the FE provides the flexibility needed to support traditional routing while the CHARM subsystem was chosen since it offers many of the features necessary for the data path, such as scheduling and buffer management. Specifically, the CHARM chip supports per VC queueing and provides facilities for partitioning the packet memory into distinct buffer pools. Each VC can be associated with any one of the buffer pools. The scheduling capability of the CHARM chip consists of three timing wheels that are associated with three priority levels. The top two priority timing wheels operate in a non-work-conserving manner, and only transmit the cell that is in the current time-slot. The third timing wheel is actually work-conserving and provides an approximation to a Weighted Fair Queueing (WFQ) scheduler [9].

The main architectural difference between the CE and the FE is that the CE has only a single CHARM subsystem interfacing to the switch, while the FE has two, one interfacing to the link and the other to the switch.

The general structure of data flows through a QSR is shown in Figure 2. The data flow is comprised of the three basic VC segments: external-in, internal, and external-out. Connections between the segments can be made either at layer 3 (routed VCs) or directly at layer 2 (switched VCs). Figure 4 shows these different connectivity options for data flows through an FE. Connectivity selection is performed for each VC, and all cells/packets arriving on a given VC are either routed or switched. The configuration shown in Figure 4 is most representative of the operations on the input side (from link to switch) of a QSR, but similar data paths exist on the output side.

Cells arriving from the link (switch) on “routed” VCs are buffered and reassembled into packets in the packet memory of the Rx_CHARM (Tx_CHARM) subsystem. The headers of the reassembled IP packets are then transferred over the PCI bus to the PowerPC complex where they are processed (a next hop look-up is performed) and modified. The modified headers are moved back to the CHARM packet memory and the associated packet is then readied for transmission to the switch (link) on the corresponding VC. Transmissions are arbitrated by the scheduling component of the CHARM chip.

Cells arriving from the link (switch) on "switched" VCs can be handled in either one of two ways. They can be buffered and reassembled into packets in the packet memory of the Rx_CHARM (Tx_CHARM) subsystem before being readied for transmission to the switch (link). Alternatively, cells can be made available for transmission immediately after they have been received. These modes correspond to the frame and cell cut-through forwarding modes mentioned earlier, and they can be configured in the CHARM subsystem for each VC. Support for multicast connections (switched or routed) is currently provided through a broadcast VP originating from each input. The VCI field then identifies the flow and the outputs on which it is to be received. While simple, this approach has the drawback of unnecessarily overloading some switch output ports. In the next release, this is avoided by providing a direct interface to the switch control point so that point-to-multipoint connections can be dynamically established.

3 QoS Control Functions

There are two major components involved in supporting QoS capabilities in the QSR: the resource management entity and the RSVP protocol. In this section, we describe their respective roles and functions.

3.1 Resource Management

Resources in the QSR are managed and controlled by the *Resource Manager* running at the CE and the *Resource Controllers* running on each forwarding engine (FE). The resources to be managed include, (1) VP and VC labels (external and internal to the box) and their connectivity; (2) Link Capacity (external as well as internal to the switch); and (3) Buffers in the forwarding adapters (both receive and transmit sides).

The resource manager is responsible for the label spaces of the point-to-point and point-to-multipoint VCs between switch ports as well as the label spaces of the VCs on the links at the trunk FEs. The port FEs use ATM signaling to dynamically setup VCs across the attached ATM subnet. On trunk links, the label space of the VCs on the link is currently owned by the sender. However, tracking of allocated VCs is performed by the resource managers at both the sender and the receiver.

For ease of management, the VCs are divided into three pools: (a) internal point-to-point VCs between switch ports, (b) internal broadcast VCs that are used for multipoint connections and (c) external VCs terminating or originating at the FEs (external-in and external-out). The resource manager maintains three tables to keep track of all the VCs that are free or in use. In addition, it also tracks the resources allocated to each of these VCs at the level of the CHARM chip. Furthermore, the resource manager maintains a table of the active flows passing through the QSR. A flow can be either point-to-point or point-to-multipoint. In general, a point-to-point flow consists of the three segments identified in Figure 2: (a) a VC on the incoming link (external-in); (b) a VC on the outgoing link (external-out); and (c) an internal VC segment connecting them. Each flow is assigned a unique entry in the flow table, with pointers to the different VC segment(s) that constitutes the flow. If the ingress and/or the egress FE is a port side FE, the VC(s) on the incoming and/or the outgoing link is currently not present.

The case of a point-to-multipoint flow is very similar and also consists of at most three segments. The only difference being that the internal VC is a broadcast VC, and each branch of the multicast flow corresponds to a specific VC segment that is created by enabling receipt of the broadcast VC in the associated output port.

In the current implementation, we support, in addition to Best Effort, both the Guaranteed Service [16] and the Controlled Load [17] service. Resources are allocated at the input and output adapters for unicast and multicast flows. In the case of multicast flows, because of the impact of broadcasting multicast traffic, the switch bandwidth is also taken into account.

On the input side, allocation of resources is relatively loose because of the output queueing nature of the switch. Specifically, flows associated with bandwidth reservation, i.e., switched flows, are placed on the highest priority timing wheel, with a transmission rate (peak and average are set equal) set to the link bandwidth. This essentially means that packets are forwarded for transmission as soon as they are ready. Furthermore, because no merging of flows takes place on the input, cell-based forwarding is selected because of its lower latency. Non-reserved traffic, i.e., routed default IP traffic is assigned to the low-priority work-conserving timing wheel, where it is allocated a nominal rate value. This value corresponds to the bandwidth set aside for default IP traffic on each link. Guaranteeing that at least that much bandwidth remains available to default IP traffic is enforced through call admission by limiting the maximum amount of reserved traffic. Note that because of the work-conserving nature of the timing wheel, the default IP traffic will always be able to access any idle bandwidth.

In addition to being mapped to different timing wheels, reserved and default traffic are also assigned to different buffer pools. Both the reserved flows and control traffic have distinct buffer pools, each allocated a dedicated amount of memory, but also capable of sharing a common pool of excess buffers. Default IP traffic is constrained to its own buffer pool and is not allowed to share any of the excess buffers. The shared pool of excess buffers is used only as a safety margin for reserved traffic in case backpressure temporarily degrades throughput through the switch. This may occur if large bursts of default IP traffic from multiple inputs are directed towards the same output. In such cases, the switch shared-memory starts filling up, resulting in degraded switch performance.

As mentioned before, because of the output queueing behavior of the switch, resource allocation is most critical at the output. The Guaranteed Service flows (GS flows) are mapped to the high priority non-work-conserving timing wheel with a rate allocation equal to their requested service rate R. The token bucket size is set to the maximum packet size M and the peak rate is set to the link speed. The latter is needed in order to avoid an additional delay term of M/R because of the cell based transmission of packets. The setting of the token bucket size to M instead of the value b specified in the TSpec of the flow, means that we are enforcing reshaping at the service rate R. This is known not to affect the delay [11] and lowers the buffering requirements in the network. As a result, the amount of buffer allocated, i.e., accounted for at time of call admission, to GS flow is simply set to $2M$, which is taken from the buffer pool dedicated to GS flows.

Controlled Load flows (CL flows) are mapped onto the work-conserving timing wheel with a rate r equal to the bucket rate specified in their TSpec. Buffer allocation is made from a separate buffer pool assigned to CL flows and is based on statistical multiplexing assumptions. Specifically, when a new CL flow is to be added, the call admission function checks if enough bandwidth is available on the link. Additionally, it also checks if by adding this flow the probability of running out of buffers remains below an acceptable threshold. This probability is computed based on the number of flows and their associated token bucket depths. In the current implementation, a simple Gaussian approximation is used.

The handling of default IP traffic at the output is essentially similar to what is done on the inputs. Default IP traffic is assigned to a separate buffer pool and is put on the work-conserving timing wheel with a nominal rate value corresponding to the base amount of bandwidth set aside for default IP traffic. Note that although CL flows are also present on the work-conserving timing wheel, their excess traffic (and that of GS flows) is detected when entering the network and forwarded as default IP traffic. Hence, the issue of how to share idle bandwidth between excess reserved traffic and default IP traffic does not arise in the current implementation.

3.2 RSVP Protocol

The CE implements the RSVP protocol as specified in [6], with some extensions needed to support the mapping of RSVP flows onto switched connections. This essentially requires the ability to communicate the identity of the VC that is to be used for a given RSVP flow. In the current implementation, this information is carried in the *PATH* messages[1] as they travel from QSR to QSR. In particular, VCI information is piggybacked into the LIH field of *PATH* messages. Reservations are activated upon receipt of a *RESV* message, and serve as the trigger to the forwarding of data packets onto the switched path (see below).

Upon receiving a *PATH* message, the RSVP protocol first extracts the VCI information from the LIH field. The value carried in the field identifies the VC terminating in the input adapter[2], and on which data packets will eventually be sent. At this point, the RSVP protocol contacts the resource management entity. It first notifies it of the VCI to be used by the new flow on the link terminating at the input adapter. Resource management verifies that this VCI is not already in use. Assuming it is not, RSVP then provides resource management with the identity of the output adapter/link on which the *PATH* message is to be forwarded (RSVP obtains this information from routing). Resource management then returns the VCI values of both the internal VC that will be used through the switch between the input and output, and

[1]As mentioned in Section 3.1, the sender currently assigns the VC.

[2]We are assuming here the case of a trunk adapter as in the case of port adapters, this field is currently unused.

of the VC to be used on the specified output link. The latter is inserted into the LIH field of the *PATH* message that is sent to the next downstream node.

The above applies to unicast flows, but multicast flows are treated similarly, simply by indicating the multicast nature of the flow through a special flag. The use of this flag signals to resource management that the internal VC needs to be allocated from the pool of broadcast VCs. VCs for each of the links corresponding to outputs associated with the multicast flow are allocated one at the time, through repeated `AddParty()` calls that each time specify the identity of a new output. Note that as in the unicast case, the VCs do not carry any data until a reservation request is received.

Upon receiving a *RESV* message specifying a given service class and service parameters, the RSVP protocol communicates this information to resource management. Resource management then performs call admission and resources allocation, and assuming that this step is successful, it then triggers "splicing" of the different VC segments associated with the flow. Specifically, the internal VC is spliced in the output adapter onto the VC assigned to the flow on the output link. Similarly, on the input, the VC assigned to the flow on the incoming link is spliced onto the internal VC. This splicing ensures that from that point on, packets are forwarded directly at layer 2 through the box. However, there is still a need to identify data packets belonging to RSVP flows as they enter the first QSR box on their path. This amounts to updating the classifier on the ingress port adapter of the first QSR box. Triggering this update is again the responsibility of the RSVP protocol in the CE, and is performed upon receipt of the first *RESV* message. It is performed by sending a control message to a client stub residing in the ingress port adapter, that specifies the necessary information (source and destination addresses and port numbers) to identify the corresponding packets. The client stub then updates the data path classifier accordingly, so that packets matching this criteria get immediately forwarded onto the internal VC associated with the flow.

Reservation and flow removals (*RESV_TEAR* and *PATH_TEAR*) are handled in a symmetric fashion. Upon receipt of a *RESV_TEAR* message, the associated VCs are "unspliced" and packet forwarding returns to the default IP data path. In the case of a multicast flow, one needs to determine if the reservation was the last active one, at which point unsplicing of the internal VC at the input is also done. Deallocation of the VCs is only performed upon receipt of a *PATH_TEAR* (or the time-out of the associated path state), at which point both the internal and external outgoing VCs are returned to the pool of free VCs.

One aspect that needs to be pointed out is that even if the data packets of an RSVP flow are eventually carried over a switched connection, this does not apply to control messages such as *PATH* and *RESV* messages. These continue to follow the "hop-by-hop" routed path. Consequently, the path followed by the switched connections remains entirely under the control of the IP routing protocols. For example, when the RSVP protocol detects a route change, the switched path is immediately taken down. It is reestablished along the new route as *PATH* and *RESV* messages change their course.

4 Data Paths

In addition to basic IP forwarding function, ports support a variety of functions including "Classical IP over ATM" (CIP) [12], Multicast Address Resolution (MARS) [4], and ATM UNI [2]. They serve as entry and exit points into switched pipes. Trunk adapters support both routed and switched data paths, that are optimized for performance and service differentiation. The routed path involves the local PPC603 processor for determining the IP next hop using a specialized look-up. Switched paths are under the sole responsibility of the CHARM chip, that determines the next hop (VC) on which to forward packets, and enforces the appropriate scheduling based on the specified service class and traffic parameters for the flow.

4.1 Default IP Forwarding

The data path of a "default" IP packet starts with its entrance into the QSR network at a port adapter. Packets arriving at a port adapter are reassembled by the Rx_CHARM chip. The processor is notified of the packet arrival during or after the completion of reassembly. The CHARM chip provides two basic mechanisms that can be used for this purpose. It can setup a DMA transfer across the PCI bus into the system memory of the processor (the CHARM chip can act as either a bus master or slave). Alternatively, it can post an event into one of several event queues that the processor regularly polls. The second approach is preferred since it provides better performance. The processor polls the event queue and when it finds it to be non-empty, it reads the packet header information from the CHARM memory across the PCI bus (it has access to both the packet and control memories of the CHARM chips)

Processing of packet headers proceeds in a number of ordered steps: (1) special packet traps; (2) classifiers (for ports); and (3) core forwarding loop, with some variations between port and trunk adapters. For all three, the first step is to identify whether an incoming packet is one that requires special processing. This includes packets containing IP options, RSVP control messages, and packets addressed to the router. These packets are all forwarded to the CE for further processing. As ports serve as the entry point into switched pipes, their next step is to identify the packets that need to be forwarded onto those pipes. This is done through a classifier which we describe in the next section. Packets that are not selected by the classifier are then handled by the core IP forwarding loop. The core forwarding loop provides similar functionality in both the port and trunk adapters, namely, a next hop look-up based on the IP destination address carried in the packet header.

Upon completion of the lookup, the next hop (output adapter) is identified and the associated VC is retrieved. The processor then updates the packet header (TTL and checksum) and writes it back into the original memory location. It then notifies the Rx_CHARM chip that the packet is available for transmission and provides it with the identity of the outgoing VC. Since we elected to notify the processor of a packet arrival only after the full packet had been received, packets are always available for transmission when the processor returns the outgoing VC information. Although this option increases latency at low load, it avoids checks for packet availability in the forwarding loop. These checks increase the path length of the forwarding loop, and negatively affect performance at high load.

Once the Rx_CHARM chip is provided with the identity of the outgoing VC for a packet, it enqueues it for transmission on the appropriate queue. As described in Section 3.1, the VCs (one to each output adapter) for default IP traffic are assigned to the work conserving timing wheel and share the amount of bandwidth available to them into the switch. VCs to different switch output ports are served in a round robin fashion whenever default IP traffic is provided with a transmission opportunity into the switch. The resulting cell interleaving helps switch performance by breaking up bursts (full packets) headed to a given output.

Cells from default IP packets travel through the switch and are reassembled into packets by the Tx_CHARM chip. Packet reassembly is needed as the default IP VCs from all inputs are merged onto a single outgoing VC on the link. This merging is done done by the Tx_CHARMchip without any involvement from the processor. The outgoing default IP VC is again assigned to the work conserving timing wheel with a nominal transmission rate, that ensures a floor throughput to default IP traffic. Upon arrival at the input (trunk) adapter of the next QSR, the process outlined above is repeated until the packet reaches the last QSR box on its path.

4.2 Forwarding of RSVP flows

RSVP data packets arriving at an ingress port adapter are intercepted by a classifier that precedes the core forwarding loop. The classifier function is needed since the identity of the incoming VC is typically not sufficient to identify the data packets as belonging to an individual flow (multiple flows as well as non-RSVP packets can currently be multiplexed on any incoming port VC). The classifier uses a single hash-based look-up that combines the source address and port number, destination address and port number, and protocol type into a hash key. Collisions in the hash table are handled using simple chaining. The classifier identifies packets belonging to an established (a reservation is in place) RSVP flow and forwards them directly onto the corresponding VC. In the output adapters, for both unicast and multicast flows, the VC on which data RSVP packets arrive is directly spliced onto a point-to-point VC going out on the link. At the input trunk adapter of the next QSR on the path, this VC is again directly spliced onto the VC (point-to-point or broadcast, depending on the type of the flow) that takes it across the switch of this next QSR. This process repeats until the egress port adapter is reached. Hence, the packets are processed only at the switched level until they reach the egress port adapter.

The handling of the VCs used to carry RSVP data packets depends on their reservation style and service class. In the current implementation, individual flows remain assigned to distinct VCs even when they belong to shared reservations. For example, packets from multiple senders destined to a common multicast session will be carried over different VCs even when the receiver has specified a shared reservation style

(Shared Explicit or Wildcard). Keeping flows from different senders on different VCs allows us to merge and *unmerge* flows without incurring any layer three processing. Unmerging of flows is needed because of our use of the MOSPF multicast protocol, that creates source specific trees. This means that while trees from different senders may share a number of links, they can diverge at some later downstream point. Such divergences require that we be able to extract packets from different senders, so as to forward them on their respective trees. Another reason for keeping flows on distinct VCs is that it minimizes latency. Specifically, merging of flows requires the use of the frame-based forwarding mode of the CHARM chip, while keeping distinct VCs allows us to exploit the cell-based forwarding mode. Finally, the use of distinct VCs also allows for a fairer sharing of resources between flows for a given reservation. In particular, the CHARM chip allocates transmission opportunities to VCs sharing the same reservation in a round-robin fashion.

5 Provisioned IP Service

The RSVP protocol enables applications to request a specific quality of service for host-to-host flows. While this level of specificity is appropriate for applications such as real-time multimedia applications, many other applications could benefit from a coarser level of service. For example, it is appropriate to provide some reserved bandwidth for connecting a branch-office to a central operations site, a scenario that is common for many business applications. In these instances, it is neither appropriate nor necessary to create a reservation for each individual flow. The reserved bandwidth may be used in a variety of ways: it may be used for all traffic destined for the central site, it may be used for traffic destined for a specific host in the central site, or it may be used for a specific application. To fulfill these requirements, we have developed a "provisioned IP" service that supports these various levels of granularity.

The "provisioned IP" service is constructed using a special packet classifier function at an ingress port and a switched IP tunnel connecting the ingress port to the appropriate egress port. For the scenario described above, the switched IP tunnel represents the reserved path between the branch office and the central site, while the classifier in the (ingress) router at the branch office is configured to steer an appropriate subset of traffic into the tunnel. The switched IP tunnel itself is created using the RSVP protocol by specifying the two endpoints of the tunnel.

In the QSR system, the establishment of switched pipes and the associated updates of the classifier function on port adapters are implemented through an application, Provisioned Switched IP (PSIP), that resides in the CE. The application is invoked through a simple Web-based interface that allows a network operator to request the establishment of provisioned switched pipes to and from specific subnets (or sets of subnets). The first step in setting up such a pipe is to specify the source and destination subnets (and subnet masks) for the pipe. This step is initiated at the ingress QSR to which the source subnets are attached.

The source subnets identify the local ingress adapter where the switched pipe will start. Likewise, the destination subnets are used to identify (by querying routing information) the address of the egress QSR to which they are attached. At the same time as the address information is specified, the traffic parameters and service type of the connection are also entered. Currently, we only support Fixed Filter and Controlled Load for PSIP pipes, but will shortly extend this to Shared Explicit filters. The use of shared reservation is expected to be useful when setting-up provisioned pipes to destinations such as server farms, where access bandwidth is to be shared across the different users accessing the server farm.

Once this initial information has been entered, the PSIP application at the ingress QSR communicates with its local RSVP function through a standard host RSVP-API, and requests the transmission of a *PATH* message destined to the identified egress QSR. Specifically, the destination address is set to that of the egress QSR and the port number is chosen to ensure that the remote PSIP application is notified upon arrival of the *PATH* message. Upon receiving the *PATH* message, the PSIP application in the egress QSR immediately triggers the transmission of a *RESV* message. As described in Section 3.2, the *RESV* message triggers the splicing of the different VCs assigned to the flow in the course of the *PATH* message.

At the egress QSR, this process is slightly modified so that the spliced connection terminates in the port adapter to which the destination subnets are attached. This modification is needed to avoid terminating the switched connection at the CE, which is the end-point associated with the destination address specified in the RSVP messages. This modification is implemented through a simple extension of the API between PSIP and RSVP, that allows specification of different local end-points to be used by RSVP when communicating with Resource Management. Likewise, on the ingress side, the extended interface between RSVP and PSIP is used to specify the ingress port adapter as the starting point for the switched pipe.

Once the switched pipe is set up, the network operator can specify the filters that are to be used to select packets for this pipe. These filters are inserted into the forwarding code at the ingress port. In the current implementation we have chosen to use CIDR prefixes as filters for the PSIP application, which enables us to tune the granularity of the search from a single host address up to a large subnetwork. Since a switched pipe can have one or more prefixes associated with it, and it is possible to have several PSIP pipes originating at an ingress port, the PSIP classifier is implemented as a longest prefix match lookup. The packet forwarding code has been modified to perform this lookup after the standard RSVP classifier. The ingress port forwarding function thus consists of (1) traps for special (control) packets, (2) the RSVP classifier, (3) the PSIP classifier, and (4) the standard IP forwarder. This ordering enables us to give preferential treatment to the "provisioned" traffic over the best-effort traffic. Furthermore, since we anticipate a relatively small number of such provisioned pipes, the impact of this additional check on the forwarding performance for default IP traffic remains minimal.

It should be noted that in our system, the IP tunnel is constructed by creating the switched pipe through the QSR network. The concept of provisioned IP service may, however, also be realized using UDP/IP encapsulation of packets at the ingress port. The elimination of overheads for encapsulation and decapsulation is one of the benefits of the QSR design.

6 Test Scenarios and Applications

The testbed consists of 3 QSRs as shown in Figure 5, to which several UNIX workstations are attached. The workstations are connected to the Port FEs through an ATM Switch using IP over ATM (RFC 1577, [12]). Note that some of the workstations like `elvis` and `clash` have 155 Mbits/sec ATM adapters, whereas the others are connected through 100Mbits/sec ATM TAXI interfaces. Despite the fact that we had reasonably high-performance workstations, we were unable to generate more than 30Kpps out of each of these UNIX workstations. Therefore, in order to generate more traffic and stress the system, we connected two routers (`who` and `what`) that were modified to simply serve as packet generators. These were connected to QSR 2 as show in Figure 5.

6.1 IP Forwarding

We measured the forwarding performance of a Trunk FE over a range of different packet sizes. Figure 6 depicts the forwarding performance of the trunk FE. As described in Section 4.1, the trunk forwarding lookup is implemented using a DP-trie structure [10] and, therefore, the forwarding performance will vary with the actual depth of the lookup. We have plotted for different packet sizes (plain line curves), the two extreme cases which correspond to depths of 1 (F-dp-1) and 32 (F-dp-32) in the DP-trie. For small packets, the Trunk FE can forward 126 Kpps for a 1-deep lookup, and around 96 Kpps for a lookup that is 32 deep. For 1-2 cell packets the forwarding loop is the bottleneck, as the throughput is significantly smaller than what the link can sustain. For packets that are larger than 2 cells the forwarding loop is no longer the bottleneck. Rather it is the CHARM chip which cannot receive small (3-5 cell) frames at link speed. For packet sizes in excess of 1Kbyte, we achieve a throughput of approximately 150Mbits/sec (this figure includes ATM overheads), which is close to the maximum the link can support. The throughputs obtained for the different packet sizes are plotted in dashed lines for the 1-deep (T-dp-1) and 32-deep (T-dp-32) lookups.

6.2 QoS Support

The next and more important tests focused on the service differentiation capabilities of the QSR. These capabilities were exercised by running a video stream between `clash` and `elvis`, and observing the qualitative and quantitative variations in performance of displayed video under a number of configurations. Both `clash` and `elvis` were equipped with Parallax cards, that perform hardware JPEG compression/decompression providing good quality video at a bit rate of 3 Mbits/sec. A video camera was attached to `clash` and the playback was on `elvis`, requiring the video stream to cross all 3 QSRs as shown in Figure 5. Control of the video stream was done using the *vic* application since it was readily available [13]. We also extended *vic* to provide an interface with our host RSVP daemon, enabling it to reserve resources through the QSR network using the RSVP protocol.

To characterize the impact of resources contention on the quality of the video stream when its packets are sent as default IP traffic, we "blasted" packets destined to **1.1.3.1** from the two routers `who` and `what`. Depending on the size and rate of the packets transmitted by the routers, we observed either one of the following two scenarios: (1) the forwarding engine at **FE 1.3** could not keep up, or (2) the link from **FE 2.2** to **FE 1.3** was congested.

Overloaded FE 1.3: In this scenario, packets arrive at **FE 1.3** at a rate higher than what the IP forwarding loop can handle. As a result, the receive buffer pool soon fills up and arriving packets are dropped indiscriminately. When the video packets from `clash` are sent as default IP, i.e., follow the routed path, they are also subject to this congestion and dropped. The amount of dropped video packets can be varied by adjusting the rate of packets sent from the two routers, but the qualitative degradation of the video is quite significant even when only around 10% of the packets are lost. This is because each video frame is composed of several packets and the loss of a single packet invalidates the entire frame.

Congestion on link from FE 2.2 to FE 1.3: In this scenario, the bottleneck is not the IP forwarding loop, but instead the link between QSR 2 and QSR 1. This is achieved by decreasing the rate of packets sent from the two routers `who` and `what` but increasing packet sizes, so that they each generate a load close to their link speed, i.e., 100 Mbits/sec. As a result, **FE 2.2** now receives over 200 Mbits/sec which exceeds the capacity of its OC-3 link to **FE 1.3**. The receive buffer pool in **FE 2.2** then starts filling up and severe packet losses occur, resulting in poor video quality on `elvis`.

By using RSVP we can setup a reserved switched pipe from **FE 3.1** to **FE 1.1** to provide QoS to this video stream. The RSVP Path and Resv messages are generated by an extension to the *vic* interface. A Controlled Load reservation is requested with a peak rate of 155 Mbits/sec, a token bucket rate equal to the maximum rate set by *vic* (about 3 Mbits/sec), and a token bucket depth of about 2Kbytes. The establishment of the switched pipe ensures that the packets from the video stream altogether avoid the congested IP forwarding loop at **FE 2.2**, have access to buffers from a different pool than the default IP traffic, and are also allocated the necessary amount of bandwidth at each link. As expected, the moment the pipe is setup, video quality improved instantaneously with losses returning to zero.

One, not totally unexpected, observation made during the above tests is that severe congestion on the link from **FE 2.2** to **FE 1.3** or the forwarding loop at **FE 1.3** can result in a loss of the RSVP reservation state. The fundamental problem here is that the RSVP Path and Resv messages are sent in-band and, therefore, are likely to be lost at times of severe congestion unless they are sent very frequently. To alleviate this problem, we set up special VCs, called network control VCs, that are provided with a dedicated buffer pool in CHARM/ and are reserved for network control messages, e.g., RSVP control messages, route updates, etc.

Packets arriving on the network control VC at an FE are directly spliced in hardware onto the internal network control VC that terminates in the CE. Similarly network control messages sent from the CE are forwarded on a network control VC that is again directly spliced onto the outgoing network control VC on the trunk FE. After this separate control "channel" was put in place, the RSVP signaling was reliable and could not be subverted by any amount of load on the trunk FEs or the links.

7 Conclusion

In this paper, we have presented the design of an experimental system that integrates switching and IP routing capabilities. Our focus has been on leveraging the benefits of switching in the context of providing service guarantees. We have done this not only for RSVP flows, but also for packet streams headed to specific sets of destinations, e.g., a given CIDR prefix. In both cases, the RSVP protocol is used to establish provisioned switched pipes that carry packets through the network. This use of switching is achieved while preserving all the functionality of the RSVP and Int-Serv models. Furthermore, standard IP forwarding is also supported, so that the flexibility that layer three forwarding affords is preserved. The current system clearly has limitations and its design occasionally reflects the influence of implementation "shortcuts." However, it demonstrates some basic capabilities in integrating switching and routing, in particular to address scalability requirements of the Internet infrastructure while also providing support for service differentiation.

Acknowledgments

The authors would like to gratefully acknowledge the contributions of Steven Blake, Mike Johnson, Liang Li, Jim Rubas, John Shaffer, and Tim Wivel.

References

[1] Internet Performance Measurement and Analysis (IPMA). `http://www.merit.edu/ipma/reports/`.

[2] *ATM UNI Specification, Version 3.1,* ATM Forum, September 1994.

[3] H. Adiseshu and G. Parulkar. Scalable cell switched integrated services routers. Washington University Workshop on Integration of IP and ATM, November 1996.

[4] G. J. Armitage. Support for multicast over UNI 3.0/3.1 based ATM networks. RFC: 2022, November 1996.

[5] B. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. RFC: 1633, June 1994.

[6] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) - version 1 functional specification. *Request for Comments RFC 2205*, September 1997.

[7] R. Callon, G. Swallow, N. Feldman, A. Viswanathan, P. Doolan, and A. Fredette. A framework for multiprotocol label switching. Internet Draft, `draft-ietf-mpls-framework-01.txt`, August 1997. Work in progress.

[8] G. Delp, J. Byrn, M. Branstad, K. Plotz, P. Leichty, A. Slane, G. McClannahan, and M. Carnevale. ATM function specification: CHARM introduction - version 3.0. Technical Report ROCH431-CHARM Intro, IBM AS/400 Division, February 1996.

[9] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queuing algorithm. In *Proceedings of ACM SIGCOMM'89*, pages 3–12, August 1989.

[10] W. Doeringer, G. Karjoth, and M. Nassehi. Routing on longest matching prefixes. *IEEE/ACM Trans. Networking*, 4(1):86–97, February 1996.

[11] L. Georgiadis, R. Guérin, V. Peris, and R. Rajan. Efficient support of delay and rate guarantees in an internet. In *Proceedings of ACM SIGCOMM'96*, pages 106–116, Stanford University, CA, August 1996.

[12] M. Laubach. Classical IP and ARP over ATM. RFC: 1577, January 1994.

[13] S. McCanne and Van Jacobson. vic: a flexible framework for packet video. In *Proceedings of ACM Multimedia'95*, pages 511–522, San Francisco, CA, 1995.

[14] N. McKeown. An overview of hardware issues for IP and ATM. Washington University Workshop on Integration of IP and ATM, November 1996.

[15] P. Newman, W. L. Edwards, R. Hinden, E. Hoffman, F. C. Liaw, T. Lyon, and G. Minshall. The transmission of flow labelled IPv4 on ATM data links - version 1.0. Technical report, Ipsilon Networks, February 1996.

[16] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. *Request for Comments RFC 2212*, September 1997.

[17] J. Wroclawski. Specification of controlled-load network element service. *Request for Comments RFC 2211*, September 1997.

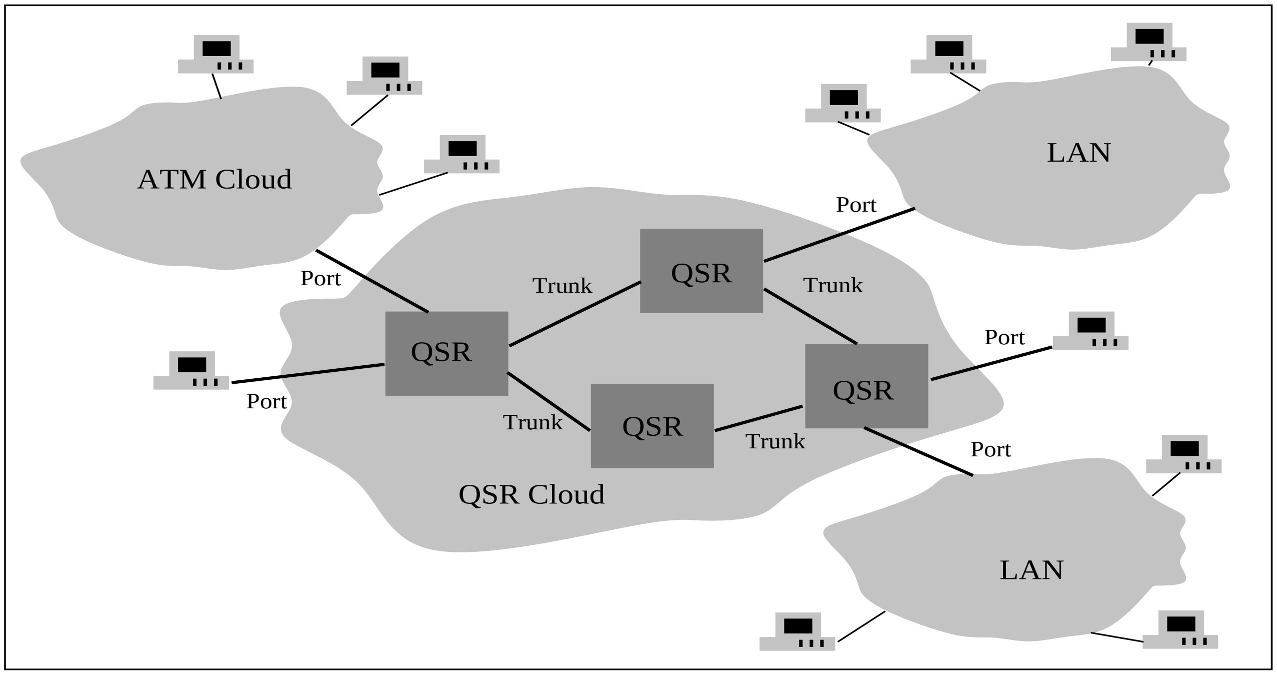


Figure 1: Overview of QSR network.

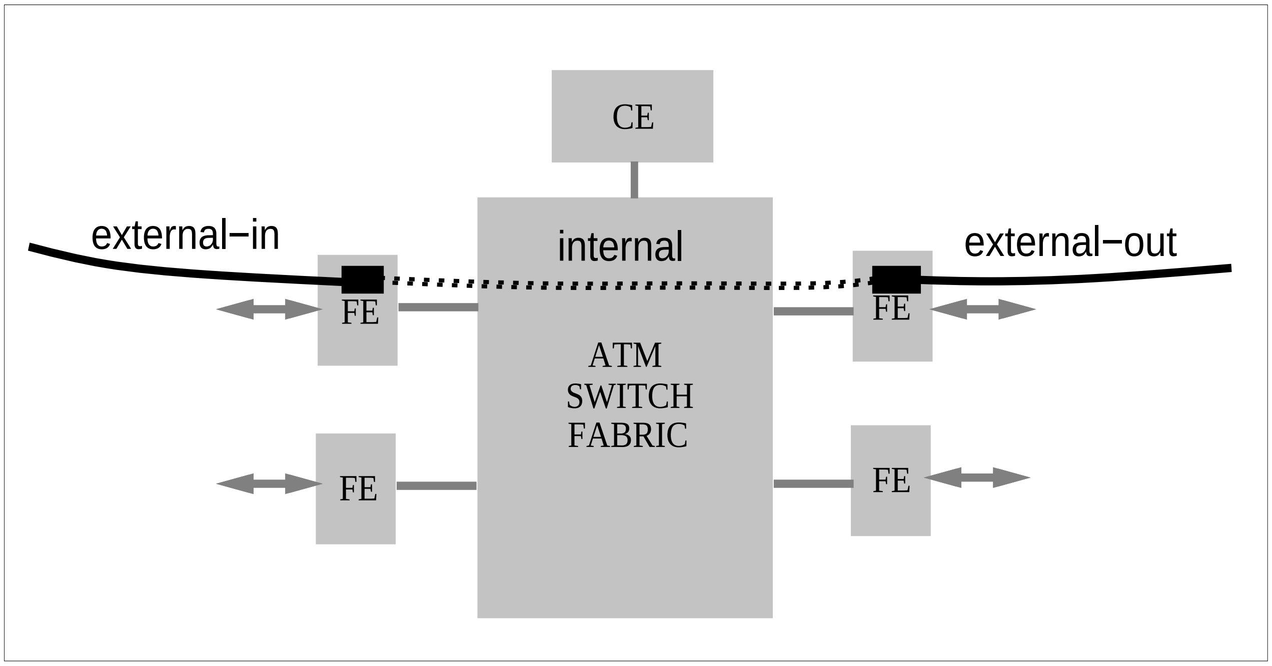
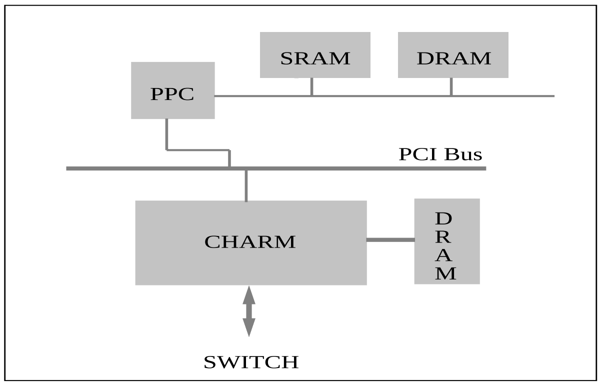
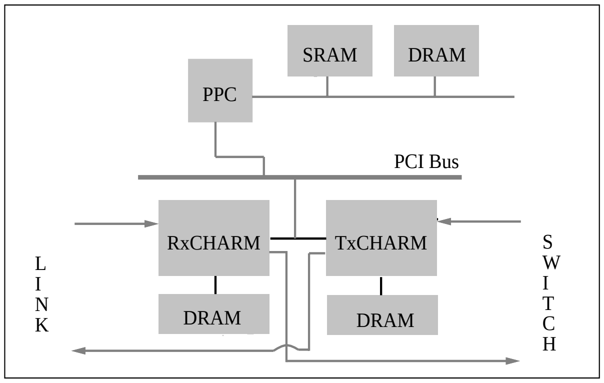


Figure 2: QoS capable Switch-Router.



(a) Control Engine



(b) Forwarding Engine

Figure 3: Hardware architecture of control and forwarding engines.

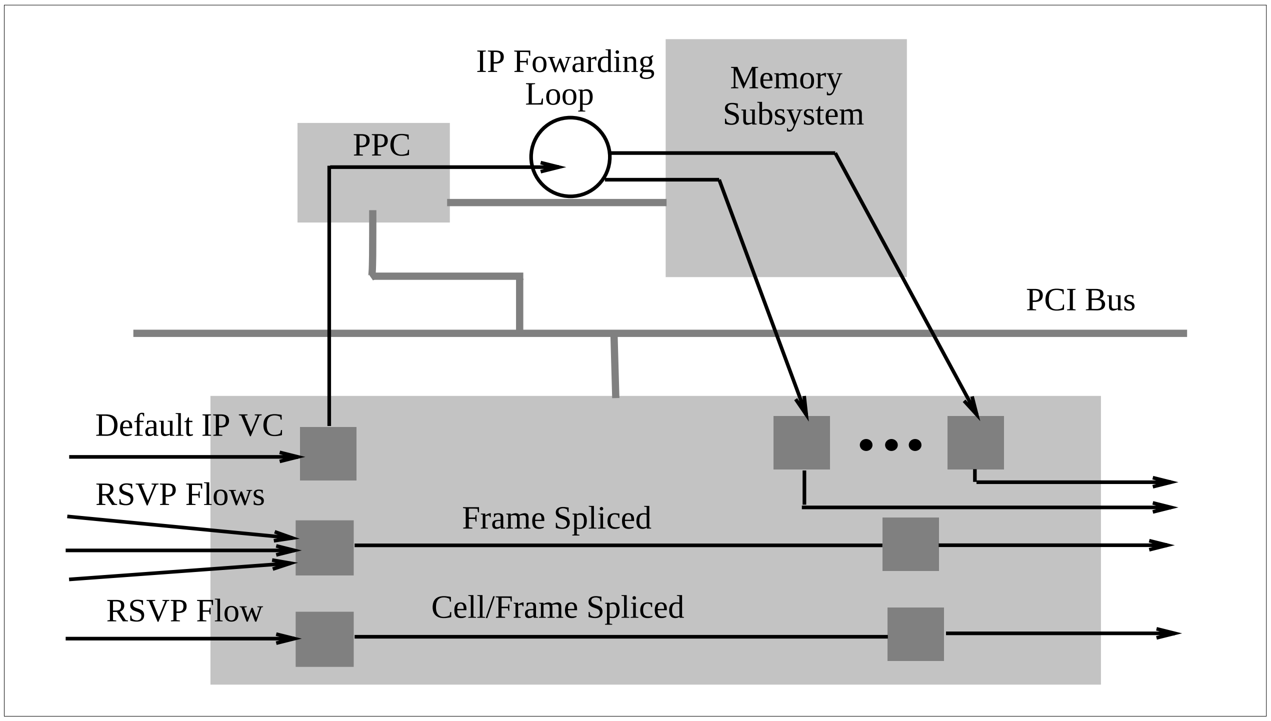


Figure 4: Data paths through the forwarding engine.

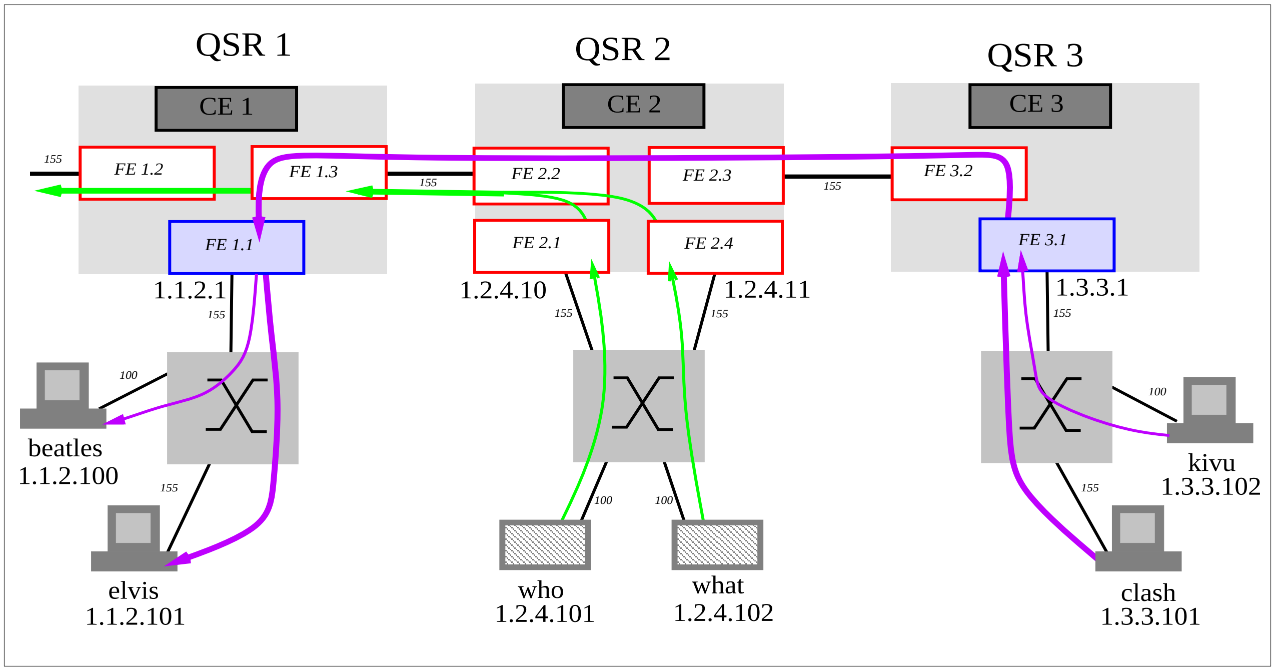


Figure 5: The QSR testbed

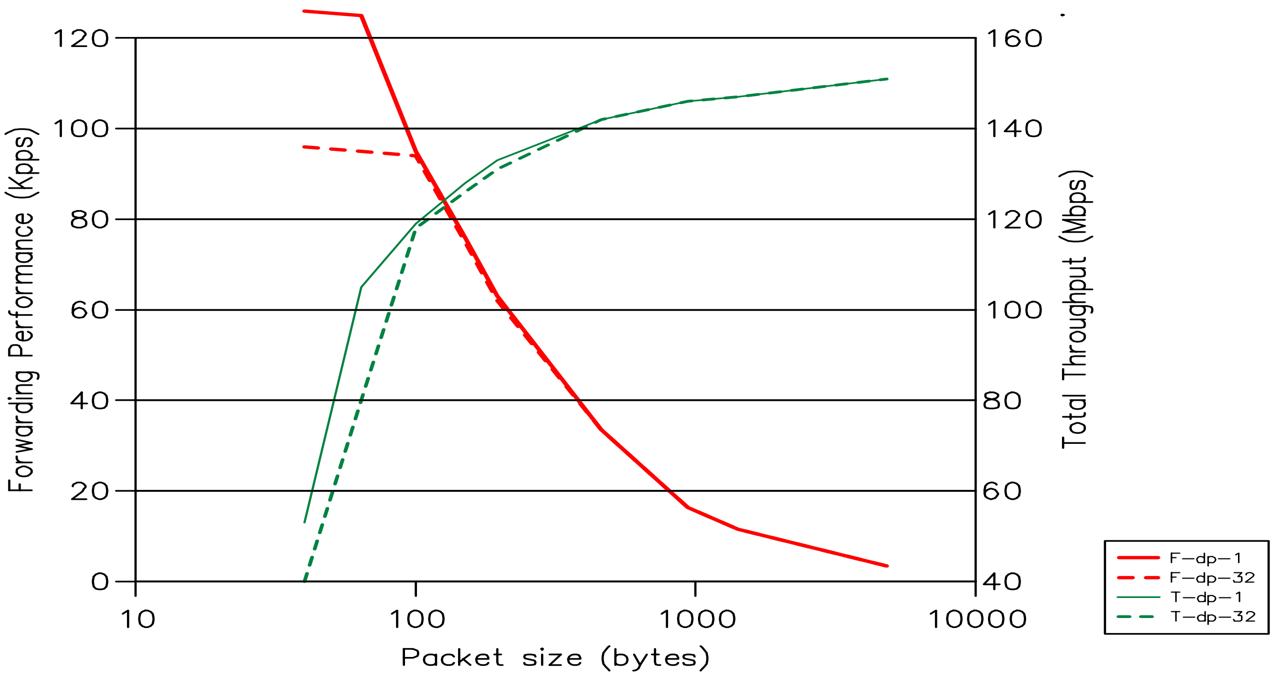


Figure 6: IP Forwarding performance of the Trunk FE