

Design and Implementation of a QoS Capable Switch-Router

E. Basturk, A. Birman, G. Delp, R. Guérin, R. Haas, S. Kamat
{basturk,birman,guerin,haas,sanjay}@watson.ibm.com
D. Kandlur, P. Pan, D. Pendarakis, V. Peris, R. Rajan, D. Saha, D. Williams
{kandlur,pan,dimitris,vperis,raju,debanjan,dougw}@watson.ibm.com

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

Abstract

Rapid expansion has strained the capabilities of the Internet infrastructure. Emerging audio and video applications place further demands on already overloaded network elements, especially the routers. 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 signalling, 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.

Keywords: Switch, Router, Networks, QoS, RSVP.

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 RealAudio and Internet Phone, 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 [6] and associated standards being developed by the IETF, namely the Resource Reservation Protocol (RSVP) [19], the Controlled-load service definition [26], and the Guaranteed service [22].

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 [16] for a more comprehensive discussion on this topic). Moreover, when the basic forwarding function is to be enhanced for QoS support and service differentiation with RSVP, 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 (see Section 5 for details on this capability).

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 signalling 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. From there on, i.e., on trunk interfaces, these packets are handled at layer-2 (switched) with appropriate scheduling support provided by the hardware.

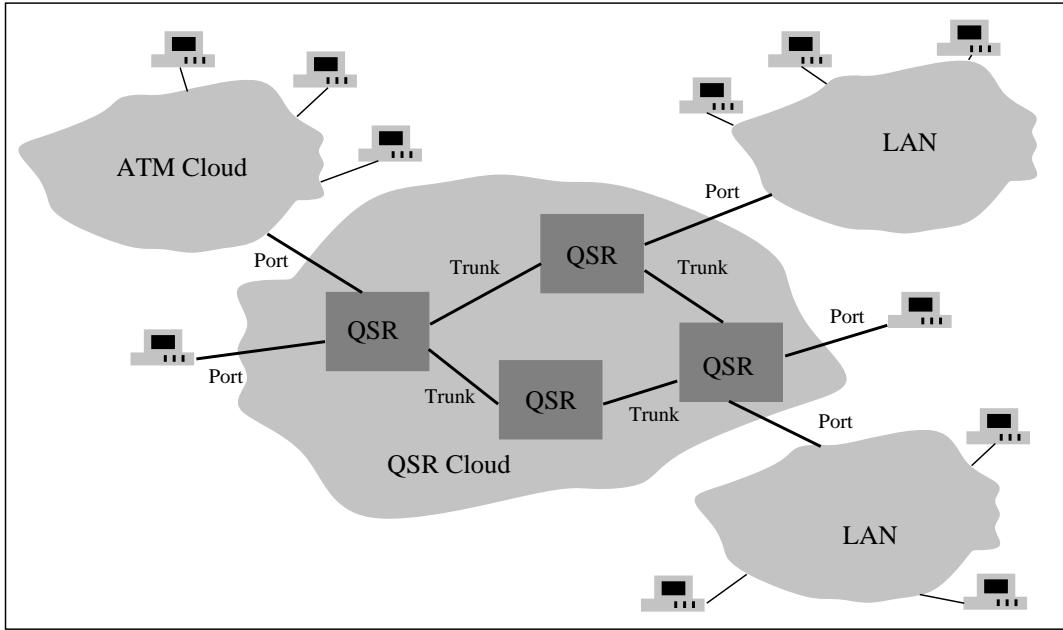


Figure 1: Overview of QSR network.

It should be noted that the formation of a switched path requires the creation of appropriate forwarding state in the QSRs for the flow. Since our focus is on exploiting switched paths for service differentiation, the setup of the switched path must be coordinated with RSVP. Consequently, we have chosen to piggy-back the necessary information on to the RSVP protocol messages. The primary benefit of reusing the RSVP protocol is that the establishment and maintenance of switched pipes is handled using the existing RSVP machinery. In particular, tear-down and reestablishment of switched pipes in the event of route changes is readily handled without the need for additional mechanism, simply by leveraging the existing interfaces between RSVP and routing.

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 [13, 17, 20, 25]. 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 signalling (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

2.1 Functional Requirements

The goals stated in the previous section, namely, providing high throughput and service differentiation, translate into specific design requirements. These requirements, which impact the data paths and the control capabilities, are met through an appropriate choice of components and function placement.

Control Requirements

Providing service differentiation requires that we support signalling 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 addition to signalling, control capabilities are needed to allocate resources to flows in a manner consistent with the requested level of service. Resource availability must be tracked, and the different types of requests – best effort, controlled load service, guaranteed service – must be mapped to the appropriate service level. As with signalling, the computations needed for resource allocation and the maintenance of the necessary state information, also add to the processing and storage requirements of the system.

Data path requirements

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 options. 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 preferred for unicast flows, while frame-based forwarding is needed to allow merging of flows, thereby circumventing the cell interleaving limitation of ATM.

In addition to supporting different forwarding modes, the data path must also provide efficient buffer management and traffic scheduling capabilities that are consistent with the Integrated Services definitions for controlled-load and guaranteed service. In particular, traffic shaping and policing is required on a per flow basis, together with the ability to effectively handle excess traffic from different flows. The latter requirement makes it desirable to control buffer allocation and usage on a per flow basis. Similarly, scheduling of packet/cell transmissions must account for

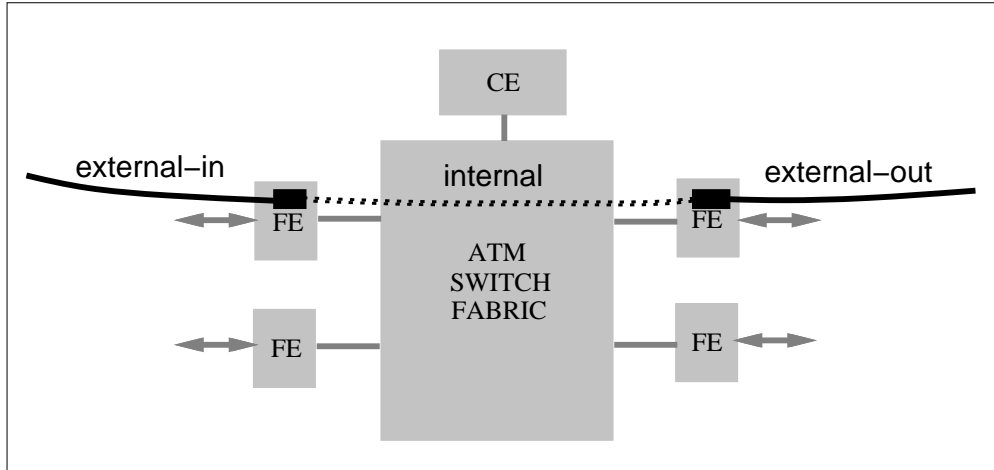


Figure 2: QoS capable Switch-Router.

the service guarantees (bandwidth and possibly delay) of individual flows, while providing excess traffic with “fair” access to any idle bandwidth. Note that this must be achieved without unduly penalizing default IP traffic and depriving it from access to bandwidth, and while maintaining high throughput.

Other requirements on the data path include the ability to support large and fast memory. This is important for both packet and control storage. The presence of different levels of service is likely to introduce wide service-time variations for low priority (best effort) traffic, and support for large buffer space is important to preserve throughput in such situations. Similarly, storage space is required for the state information needed to effectively enforce service differentiation for a large number of flows. This state space may be accessed and updated every time a packet is received. Finally, the data path connectivity between the inputs and outputs of the system must support high throughput and service differentiation for both unicast and multicast flows.

In the next two sections, we outline how the QSR structure and components meet (to a large extent) the above design requirements.

2.2 System Structure

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

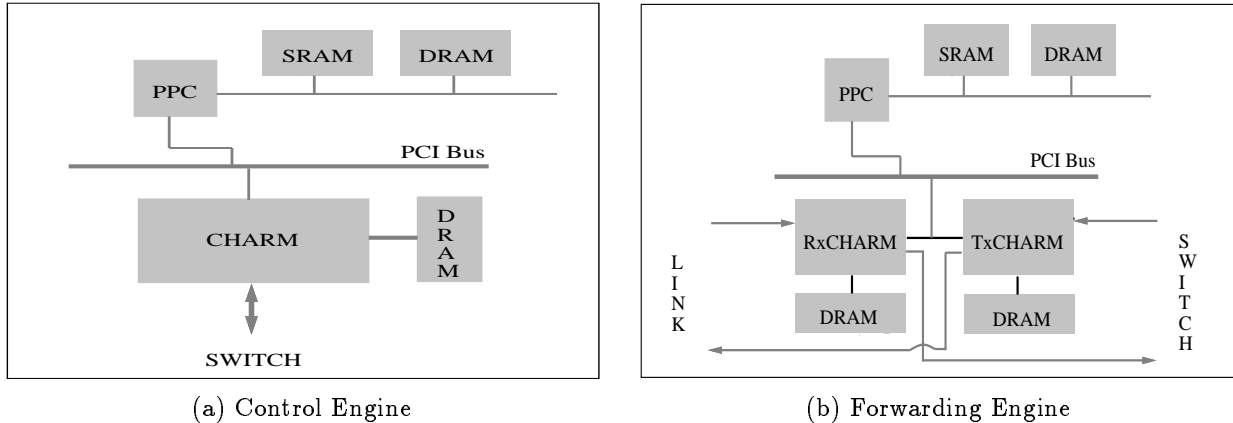


Figure 3: Hardware architecture of control and forwarding engines.

(SAR) subsystem (CHARM subsystem [7]) via a PCI bus interface. The CHARM subsystem was chosen since it offers many of the features necessary for the data path, such as scheduling and buffer management, while the 603e processor in the FE provides the flexibility needed to support traditional routing. 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.

2.2.1 Data flows

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

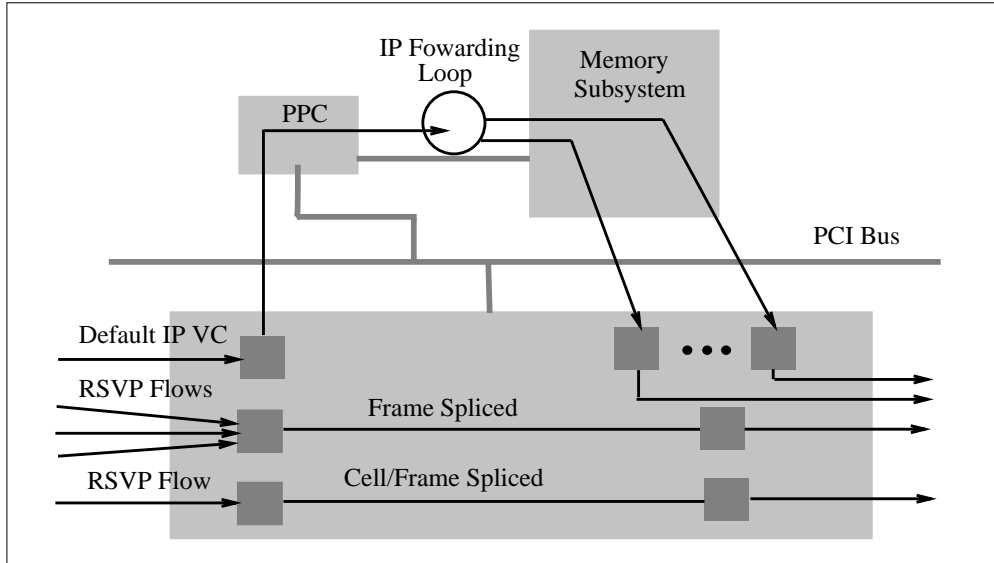


Figure 4: Data paths through the forwarding engine.

direct interface to the switch control point so that point-to-multipoint connections can be dynamically established.

From the above description, it is evident that the CHARM subsystem and the switch interconnect, described below, play a central role in the operation and performance of the system.

2.2.2 CHARM subsystem

CHARM provides many of the key functions needed to satisfy our design goals, namely, standard ATM header processing, support for large (up to 64 Mbytes) packet and control memory, multiple reassembly options, buffer management, and comprehensive scheduling support. The availability of large packet and control memory ensures scalability in throughput and adequate support for best effort traffic. The multiple reassembly options allow the forwarding information (outgoing VC) for a given incoming VC to be either pre-configured in the control memory of the CHARM chip itself (switched mode), or obtained by querying the processor over the PCI bus (routed mode). In addition, in the switched mode, forwarding can be set to take place for each individual cell or only after a full (AAL5) frame has been received. This provides the flow merging capability we need to map multiple incoming VCs onto the same outgoing VC.

Among the QoS related features, the current version of the CHARM chip supports per VC queueing, and provides facilities for partitioning the packet memory in a number of pools. Each pool can be configured with different page sizes, and the order in which pages of different sizes are to be used when reassembling packets can also be set. Pools can be setup so as to be granted a dedicated amount of memory and have access to additional memory shared across multiple pools. The amount of shared memory that a given pool can access is also configurable. Although individual VCs may be assigned to a given memory pool, pool granularity is currently not sufficient to permit the creation of a separate pool for each VC.

The scheduling capabilities of the CHARM chip consist of three timing wheels, that are associated with three different priority levels. The two top priority timing wheels operate in a non-work-conserving manner, moving from

slot to slot and considering for transmission only those cell(s) that are in the current slot. This means transmissions are not scheduled even when there are cells waiting, if these cells are scheduled for later slots. In other words, the two non-work-conserving timing wheels perform reshaping as well as scheduling. However, slots left empty can be used to transmit cells from a lower priority timing wheel. In particular, the third and lowest priority timing wheel operates in a work-conserving fashion, and always tries to schedule a cell for transmission whenever it has an opportunity to do so.

The two non-work-conserving timing wheels schedule cell transmissions on the basis of the traffic parameters assigned to each connection. These traffic parameters are essentially of the form of dual leaky buckets as in [2] and [23], i.e., they allow specification of a token bucket rate, a token bucket depth, and a peak rate. The work-conserving timing wheel only allows specification of a single rate parameter for each VC as well as the enforcement of a maximum bandwidth for all the VCs on the wheel. Its main difference, however, is that it always advances to the next active transmission slot. It provides an approximation of weighted fair queueing (WFQ) [8], and can be shown to have a behavior similar to that of self-clocked fair queueing [11]. In addition, all three timing wheels offer the ability to share a reservation across multiple VCs. This is provided through channel groups that are defined to share transmission opportunities, where the sharing is performed in a round-robin fashion.

The above buffer management and scheduling capabilities are key to the ability of the QSR to support the different types of services allowed by RSVP [19] and the Integrated Services specifications [22, 26]. We detail this further in Section 3.1.

2.2.3 ATM Switch

The switch fabric consists of a shared memory switching chip that supports both unicast and multicast connections. In each switch clock cycle, one cell can be read from the shared memory for each output. When a unicast cell is transmitted, the associated memory location is freed and returned to the pool of free locations. When a multicast cell is transmitted, a transmission count value is decremented (it was initialized to the number of outputs the cell was to be forwarded on) and the memory location is only released when the value reaches 0. On the input side, in each switch clock cycle, any input with cells to transmit can write one cell into the shared memory, provided a free location is available. Whenever an input queue is not able to transmit its cell in a given switch transmission cycle, a “backpressure” signal is applied to the input to notify it of the blocking condition. As soon as the blocking condition is removed, cell transmissions can resume.

Cells are enqueued for transmission into the switch at the inputs in FIFO order, so that the presence of the backpressure signal creates the familiar head-of-the-line (HOL) blocking problem. In other words, the switch starts behaving as an input queueing switch. The frequency of this behavior and, therefore, the performance degradation it implies is a function of two factors: the size of the shared memory and the speed ratio between the switch and the links. The current implementation combines a moderate size shared memory (of the order of 100 cells) with a speed-up factor of more than 50% over the input links, to achieve an efficient trade-off between performance and complexity¹.

¹ Using the results of [12] the saturation throughput of the switch can be found to be significantly higher than the link speed even with reasonably bursty traffic.

2.3 Software Components

Software components are distributed between the CE and the FEs. The CE implements the main control functions, while the FEs contain software entities involved with data path operations and client stubs for several of the control functions present in the CE. In both the CE and the FEs, the control functions are implemented on the PowerPC 603e processor that runs a real-time operating system. The different control components in the CE and FEs are listed in Table 1 and we now briefly review their functions and interactions.

	Init. & Cnfg.	Int. Comm.	Res. Mgmt.	RSVP	PSIP	Routing Protocols	fwdng loop	classifier	MARS clnt/serv	ATM sig. ARP
CE	✓	✓	✓	✓	✓	✓	✓			
Trunk FE	✓	✓					✓			
Port FE	✓	✓					✓	✓	✓	✓

Table 1: Software Components.

Initialization and configuration tasks are present in both the CE and the FEs, and are responsible for providing basic connectivity between the adapters. This is achieved by establishing a mesh of virtual paths (VPs) across the switch, so that the VPI field alone determines the switch outputs to which cells need to be delivered. The VCI field is then used to identify individual flows. In particular, some VCs are assigned to carry internal box control flows, while others are used for the forwarding of default IP traffic across the switch (see Section 4.1). Similarly, pre-assigned VCs are activated at configuration time to transmit/receive default IP traffic between boxes.

The internal communication component enables exchange of control information between the FEs and the CE. The current implementation provides a lightweight, unreliable, in-order message transfer service between peer software components in the FEs and the CE. For example, this is used by the routing protocol to download forwarding information to the FEs, by resource management to configure the CHARM chip when adding or removing flows, and by RSVP and PSIP to update their respective classifiers in port FEs.

Resource management is implemented in the CE which maintains information about overall system resources, namely, VCs, bandwidth and buffers, and their allocation to individual flows. Such a centralized approach may be argued to have implications in terms of scalability, but it also affords significant simplifications since individual requests typically involve resources from several entities, e.g, input and output adapters, switch, memory to store state information, etc. Client stubs in the FEs are then used to perform the necessary hardware configuration in the FE, based on “commands” transmitted from the CE. These operations are detailed further in Section 3.1.

Implementation of the RSVP protocol is also centralized, so that all RSVP control messages are forwarded to the CE by the FEs. The RSVP entity maintains state information for all the flows in the box, and is also responsible for state maintenance operation, e.g., timers, refreshes, handling of error conditions. In addition, the RSVP entity interfaces to resource management to request allocation of the resources requested by each flow. Note that this includes VC labels in order to map RSVP flows onto switched connections. The RSVP entity in the CE is also responsible for communicating with a client stub in port FEs, in order to update the corresponding classifier when flows come and go. Additional details on the operation of the RSVP entity are provided in Section 3.2.

The provisioned IP service is implemented using components in the CE and FE, and is described in Section 5. The other software components of the QSR are involved with more traditional router functions such as routing protocols, forwarding loops, and various ARP-like functions (for both unicast and multicast addresses) on port FEs connected to ATM subnets. The routing protocols (OSPF and MOSPF) run entirely in the CE which is responsible for downloading forwarding information to the FEs. Each FE in turn runs a forwarding loop whose structure and

operation is described in Section 4. In the case of ports, the forwarding loop is preceded by classifiers whose purpose is to intercept incoming packets that are to be forwarded onto switched connections through the network. Details on the classifiers can be found in Sections 4.2 and 5.

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 policies and mechanisms employed by the Resource Manager for resource allocation are based on the type of flow and the location of the resources. For example, there are differences in allocation of buffer space at inputs and outputs, as well as in ports and trunks.

3.1.1 Label (VPI/VCI) Space Management

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 signalling 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
- (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.

3.1.2 Bandwidth and Buffer Management

In the current implementation, we support, in addition to Best Effort, both the Guaranteed Service [22] and the Controlled Load [26] 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.

Switch Inputs: On the input side, allocation of resources is relatively loose because of the output queuing behavior of the switch, that results in most of the congestion taking place in the outputs. 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.

This setting is used for both Guaranteed Service and Controlled Load flows and applies to unicast as well as multicast flows. The only difference between unicast and multicast is that the transmission rate assigned to multicast flows is set to a lower value (approximately one-half of the link speed). This is because multicast flows have a greater impact on the switch, and the pacing can help improve switch performance. In addition to reserved flows, the VCs assigned to carrying control information between the CE and the FEs are also mapped to the highest priority timing wheel with a transmission rate set to the link bandwidth as well.

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 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 happen if large bursts of default IP traffic from multiple inputs is directed towards the same output. In such cases, the switch shared-memory starts filling up, resulting in degraded switch performance.

Severe degradations are avoided through a simple flow control mechanism between the inputs and outputs. This mechanism relies on the ability of the CHARM chip to gather usage statistics on individual VCs. This allows us to measure the overall incoming load of default IP traffic from different inputs and determine the maximum amount of bandwidth each is entitled to. This amount is then communicated out-of-band to the input adapters, where it is used to set the maximum amount of bandwidth that the work-conserving timing wheel can use. The operation of the algorithm while local is very similar to that of virtual-source and virtual-destination of the ATM ABR service specifications [21].

Switch Outputs: This is where resources allocation is most critical, and it depends on the type of service requested by the flow.

For Guaranteed Service flows (GS flows), the following allocation procedure is used: 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 [10] 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. One exception is in the first QSR on the path of a GS flow, where the buffer allocation is set to the original token bucket size b specified in the TSpec. This higher allocation is necessary there because the flow has not yet been reshaped and it may, therefore, be necessary to accommodate an entire burst while it is being transmitted at rate R .

Another aspect of importance for GS flows is their delay guarantee. While the use of the high priority timing wheel ensures that GS flows have precedence over all others and are also guaranteed their requested service rate, some care must be exercised to identify the associated delay guarantees. Specifically, each VC is provided with transmission opportunities on the timing wheel, that are a function of its traffic parameter, i.e., upon transmission of a cell from a VC, the next transmission opportunity for this VC is set to the first future slot on the wheel where the VC will be compliant with its traffic parameters. However, transmission need not actually occur in the scheduled time slot. This is because multiple VCs can end-up being enqueued on the same slot. In such a case, cells from those VCs are transmitted in FIFO order in the next available slots, and the next transmission opportunity for a VC can then be selectively (per VC) set based on either the original or the actual transmission slot.

The fact that multiple flows can be scheduled on the same transmission slot of the timing wheel needs to be accounted for when specifying delay guarantees. The impact of this behavior can in the worst case introduce a delay of NM/C , where N is the number of active GS flows, M is the maximum packet size (for simplicity, we assume it is the same for all flows), and C is the link speed, i.e., 155Mbps. This possibility is accounted for through the specification of a delay error term $D \propto NM/C$ that is used to update the ADSPEC of RSVP *PATH* messages for GS flows. As a result, we also limit the number N of GS flows that are admitted, in order to keep the value of the error term reasonably low². This is obviously a limitation and a source of inefficiency, but we expect it to be acceptable initially since there should not be too many GS flows. Note also that the presence of the error term D affects buffer requirements as well (in many instances by up to DR for each flow). This is accounted for by sizing the GS buffer to be larger than $2NM$ (but less than $2NM + NDR$), so that the probability of running out of buffers for GS flows is of the order of the line bit error rate.

Controlled flows (CL flows) are easier to deal with. They 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 not only if enough bandwidth remains on the link, but also 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. In the future, we plan to experiment with other functions and also use measured usage information that the CHARM chip provides.

The handling of default IP traffic is essentially similar to what is done on the inputs. Default IP traffic is assigned to a separate buffer pool and is again 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

²Note that the actual value used for D is somewhat less than NM/C , as it is only necessary to specify a value whose occurrence is of the order of the bit error rate on the link.

the network (in the classifier) 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. In the next version, the better buffer management capabilities of the CHARM chip will allow us to handle excess traffic directly on the switched paths. In that case, the WFQ approximation implemented by the work-conserving timing wheel will result in default IP traffic getting a share of the idle bandwidth, that is proportional to its nominal bandwidth allocation.

3.2 RSVP Protocol

The CE implements the RSVP protocol as specified in [19], 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* messages³ 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 adapter⁴, and on which data packets will eventually be sent. Note that the identity of the input adapter on which the *PATH* message arrived is preserved through the identity of the VC on which the message is actually delivered to the CE, e.g., a message received on VPI:0/VCI:1 arrived at input port 1. This VC identity information is made available to the RSVP protocol for that purpose.

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 obtained 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 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.

Specifically, 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, note that there is still the need to identify data packets belonging to RSVP flows as they enter

³As mentioned in Section 3.1, the sender currently assigns the VC.

⁴We are assuming here the case of a trunk adapter as in the case of port adapters, this field is currently unused.

the first QSR box on their path.

As mentioned in Section 2, this amounts to updating the classifier on the ingress port adapter of the first QSR box. Triggering this update is again under 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.

Note that the *RESV* messages currently do not carry the assigned VCI value back in the LIH field. This is because a single *RESV* message may be carrying multiple reservations, and including all the corresponding VCI values cannot be done through a single LIH field. In addition, this would unnecessarily overload the *RESV* messages. Furthermore, returning those values has little benefit since the receipt of a *RESV* message implicitly acknowledges receipt of the associated *PATH* messages and, therefore, of the VCI value it carried. The only instance where such an acknowledgment may not be provided, is with Wildcard Filters (except when a *SCOPE* object is used) as the set of senders from whom *PATH* messages have been received is not explicitly identified. In this case, we rely on the RSVP refreshes to ensure that the VCI information is eventually received for all senders.

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. This is worth emphasizing as it implies that the path followed by the switched connections remains entirely under the control of the IP routing protocols. For example, when the RSVP protocol detects⁵ a route change, the switched path is immediately taken down from that point on, and will be reestablished along the new path as *PATH* and *RESV* messages get received. This means that the management of the routed and switched paths is under the responsibility of a single entity.

4 Data Paths

Section 2.2.1 described the two basic types of data paths: routed and switched data paths. We now describe how these paths are supported by the QSR components described in the previous sections by following the various alternatives shown in Figure 4. The main components of these data paths are their start and end points, namely, the ingress and egress “port” adapters, and the segments internal to the QSR network that involve “trunk” adapters.

In addition to basic IP forwarding function, ports support a variety of interface functions and serve as entry and exit points into switched pipes. Interfaces functions consist of the protocols needed to exchange data across the ATM subnet to which the port adapter attaches. Connectivity can be provided either through permanent or switched virtual circuits (PVCs and SVCs), with SVCs being created using standard protocols such as “Classical IP over ATM” (CIP) [14, 18] for unicast connections and Multicast Address Resolution Server (MARS) [4] for multicast

⁵This is currently implemented in a straightforward fashion, simply by checking with the routing protocol before sending any refresh. Better solutions are clearly possible, and this is one area of future work.

connections. Note that since SVC based connectivity is supported, port adapters all implement the ATM UNI signalling stack. In addition to providing basic connectivity to and from its attached ATM subnet, the (ingress) port adapter also provides the classifier function needed to extract packets that are to be forwarded onto 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 603 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. Connectivity to this port adapter is provided through an ATM VC being setup using the CIP protocol for unicast traffic, or the MARS protocol for multicast traffic. Packets arriving at a port adapter are reassembled by the Rx_CHARM chip. Once a complete packet has been received, the processor needs to be notified. 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. An event posting approach is used as it was found to provide better performance than using the DMA capability of the CHARM chip. This is in part because of the overhead in scheduling DMA transfers and, since packet headers are being written in various memory location, the PCI bus burst transfer mode is not effective. 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 both, 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, but its implementation is different in the two environments.

In port adapters, the forwarding loop is partitioned into fast and slow paths. The fast path consists of a full address hash-based look-up into a software managed cache. In case of a cache miss, the slow path is invoked. The slow path is based on a balanced tree longest prefix match routine, and includes triggers for the necessary cache updates. The use of a cache was deemed beneficial for ports because of the higher potential for address locality. However, based on experiences reported by ISP backbone operators, it was felt that address locality may be weak for trunks inside the network. As a result, the trunk forwarding loop only consists of a longest prefix match look-up. This look-up is implemented using the DP-trie structure of [9], with the entire look-up table stored in system memory. The benefit of this structure is that its performance is independent of address locality, and its worst case is bounded. Some initial benchmarking tests of this forwarding loop are provided in Section 6.

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. Note that because 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 by the Tx_CHARM chip 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.

The handling of default IP packets on the last (egress) QSR box differs slightly from the processing at other nodes. The main difference is that routing of the packet takes place not only on the input, but also on the output port adapter. This second routing function is required because port adapters interface to ATM (NBMA) subnets on which multiple next hops can exist. As a result, another IP look-up is needed to identify the appropriate next hop across the ATM subnet, and the associated VC. This determination follows the same steps as in the ingress port adapters, with one difference: the checking for packets with special processing requirements is not needed since this was done in the input adapter.

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.

As was outlined in Section 3.1, 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 initial goal of the testing was to verify the functional operation of the system and provide initial performance measures. 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 based on RFC 1577 [14]. 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.

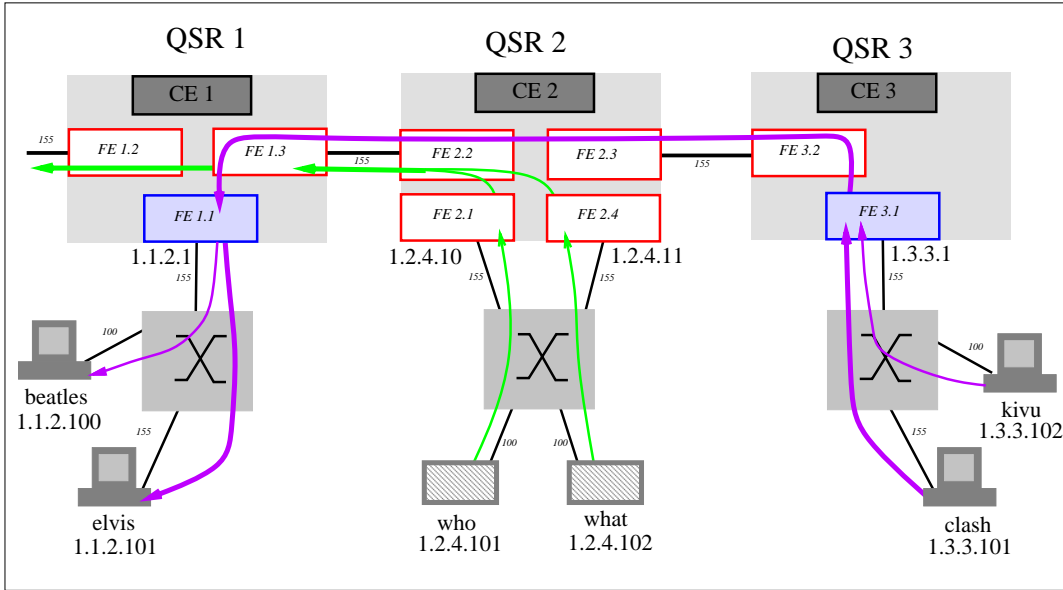


Figure 5: The QSR testbed

6.1 IP Forwarding

The setup shown in Figure 5 is now fully operational, and after the initial bring-up period we have been able to execute some initial test cases. The first one involved evaluating the basic IP forwarding performance of the Trunk and Port FEs. It should be noted, that these numbers are preliminary as no performance “tuning” of the forwarding has yet taken place.

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 [9] 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). 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. When packet size starts increasing, the reason for the slight decrease in forwarding performance for packet sizes in the range of 3-5 ATM cells is that the current version of the CHARM chip is not optimized for handling small AAL5 frames (a few cells). The next version of the CHARM chip will eliminate this problem. As the packet size increases further, the processor is no longer the limiting factor and the depth of the lookup is no longer relevant. 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. 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.

6.2 Provisioned Service

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

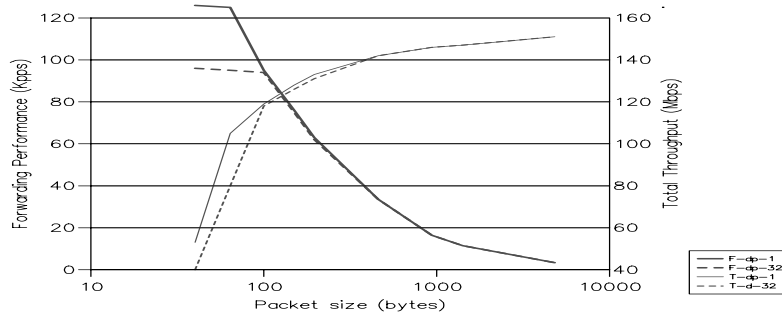


Figure 6: IP Forwarding performance of the Trunk FE

with Parallax cards, that perform hardware JPEG compression/decompression so that the quality of the video was quite good (the associated bit rate was around 3 Mbits/sec). A video camera was attached to `clash` and the playback was on `elvis`, so that the video packets had 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 [15]. We also extended `vic` so that it was able to interface with our host RSVP daemon, that had been developed as part of the project. As a result, the `vic` application was able 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.
2. The link from **FE 1.3** to **FE 2.2** 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., also 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 1.3 to FE 2.2: 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**. As in the previous scenario, the receive buffer then starts filling up and severe packet losses occur, that again affect the quality of the video displayed on *elvis*.

As described in previous sections, the QSR provides two mechanisms, PSIP and native RSVP, to remedy the problems caused by congestion in the above two scenarios. Our next tests were aimed at evaluating both the functionality and performance of these two schemes.

1. In the first test, we use PSIP to setup a provisioned switched pipe from **FE 3.1** to **FE 1.1**. The target subnet is specified as **1.1.2.0** with a mask of **255.255.255.0**. 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 (our maximum packet size). 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. The test exercises our router implementation of the RSVP protocol. It also demonstrates that the PSIP application is capable of interacting with RSVP to request the reservation, that the associated switched pipe originates and terminates in the correct ingress and egress FEs, and that the classifier is updated on **FE 3.1**. As expected, the moment the pipe is setup, video quality improved instantaneously with losses returning to zero.

Next we test the impact of the shared nature of the switched PSIP pipe by starting a packet stream from *kivu* (**1.3.3.102**) to *beatles* (**1.1.2.100**). Because the PSIP pipe is set with subnet **1.1.2.0** as its destination, the packets destined to *beatles* are also forwarded on the pipe and, therefore, compete for resources with those generated by *vic* in *clash*. As expected, some performance degradation is observed, but less severe than in the previous cases because of the more limited level of contention⁶, that only takes place on the link between **FE 2.2** and **FE 1.3**.

2. The second test involves the use of an individual RSVP reservation for the *vic* application in *clash*. This test exercises the interface to RSVP from *vic*, the host RSVP implementation, and the RSVP classifier in **FE 3.1**. Similar results are observed as with PSIP. However, one important difference is the demonstration that because of the individual nature of the RSVP pipe (dedicated to packets from the *vic* application), starting the packet stream from *kivu* does not affect performance of the video stream.

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**, leads to problems because of the loss of control messages. This took a number of forms, from the basic inability to get *PATH* or *RESV* messages through, to the dropping of reservations due to state time-out caused by the loss of several consecutive *PATH* or *RESV* messages.

The fundamental problem here is that the control messages are 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 mapped to a dedicated buffer pool 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

⁶For this experiment we disable the policing function on the classifier to ensure that all the traffic was forwarded on the switched pipe.

control VC on the trunk FE. After this separate control “channel” was put in place, the RSVP signalling 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. In addition, we are also in the process of extending and improving the QSR in a number of directions.

Some of these enhancements are concerned with the ability to extend switched connections across ports, in the case of RSVP flows. In particular, we are currently working on allowing SVCs setup to/from ports on behalf of RSVP flows, to be directly spliced onto the VC assigned in the QSR. Similarly, we are also looking at extending the use of switching to some default IP flows as has been proposed in some of the other IP switching proposals, that have been brought forward [5, 17, 20, 24, 25]. Another area of ongoing investigation is the interface between RSVP and the routing protocols, in particular in the context of QoS routing. As far as basic improvements to the system are concerned, we are looking at a number of different designs for the port forwarding loop to determine how to best combine the classifier and basic forwarding functions. We are also adding an interface between the CE and the switch control point, in particular to allow dynamic setup of point-to-multipoint VCs across the switch. This will allow us to eliminate the overhead caused by the use of a broadcast VC. Finally, we are also working on replacing the current version of the CHARM chip by the next generation that provides better scheduling and buffer management support. In particular, we have designed a per VC buffer accounting mechanism and a deadline-based scheduler which will be incorporated into this new version.

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). Home Page of the Internet Performance Measurement and Analysis (IPMA) project, a joint effort of the University of Michigan and Merit Network. <http://www.merit.edu/ipma/>.
- [2] *ATM UNI Specification, Version 3.1*, ATM Forum, September 1994.
- [3] H. Adishesu and G. Parulkar. Scalable cell switched integrated services routers. Presentation at Washington University Workshop on Integration of IP and ATM. Session 5: Hardware Issues in ATM+IP Networks, November 1996. <http://www.arl.wustl.edu/arl/workshops/atmip/session5/hari.ps>.
- [4] G. J. Armitage. Support for multicast over UNI 3.0/3.1 based ATM networks. Request for Comments: 2022, IETF Network Working Group, November 1996.

- [5] F. Baker and Y. Rekhter. Tag switching with RSVP. Internet Draft, IETF, December 1996. `draft-baker-tags-rsvp-00.txt`.
- [6] B. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. Request for Comments: 1633, IETF Network Working Group, June 1994.
- [7] 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, LAN Development, February 1996. IBM Confidential.
- [8] 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.
- [9] W. Doeringer, G. Karjoth, and M. Nassehi. Routing on longest matching prefixes. *IEEE/ACM Trans. Networking*, 4(1):86-97, February 1996.
- [10] 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.
- [11] S. J. Golestani. Network delay analysis of a class of fair queueing algorithms. *IEEE J. Selected Areas Commun*, 13(6):1057-1070, August 1995.
- [12] I. Iliadis and W. Denzel. Analysis of packet switches with input and output queueing. *IEEE Trans. Commun.*, 41(5):731-740, May 1993.
- [13] Y. Katsube, K.-I. Nagami, and H. Esaki. Router architecture extensions for ATM : Overview. Internet Draft, IETF, October 1996. `draft-rfced-info-katsube-00.txt`.
- [14] M. Laubach. Classical IP and ARP over ATM. Request for Comments: 1577, IETF Network Working Group, January 1994.
- [15] 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.
- [16] N. McKeown. An overview of hardware issues for IP and ATM. Presentation at Washington University Workshop on Integration of IP and ATM. Session 5: Hardware Issues in ATM+IP Networks, November 1996. <http://www.arl.wustl.edu/arl/workshops/atmip/session5/nick.ps>.
- [17] 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.
- [18] M. Perez, F. Liaw, A. Mankin, E. Hoffman, D. Grossman, and A. Malis. ATM signaling support for IP over ATM. Request for Comments: 1755, IETF Network Working Group, February 1995.
- [19] R. Braden (Ed.), L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reSerVation Protocol (RSVP) version 1, functional specification. INTERNET-DRAFT, Internet Engineering Task Force - RSVP WG, November 1996. `draft-ietf-rsvp-spec-14`. [`ps`,`txt`].
- [20] Y. Rekhter, B. Davie, D. Katz, E. Rosen, and G. Swallow. Tag switching architecture overview. Internet Draft, IETF, September 1996. `draft-rfced-info-rekhter-00.txt`.
- [21] S. Sathaye. ATM Forum Traffic Management Specification Version 4.0. ATM Forum 95-0013, December 1995.
- [22] S. Shenker, C. Partridge, and R. Guérin. Specification of guaranteed quality of service. Internet Draft, Integrated Services WG, IETF, November 1996. `draft-ietf-intserv-guaranteed-svc-06.txt`.
- [23] S. Shenker and J. Wroclawski. Specification of General Characterization Parameters. Internet Draft, Integrated Services WG, IETF, October 1996. `draft-ietf-intserv-charac-02.txt`.
- [24] D. Williams, R. Guérin, and D. Kandlur. Virtual circuit identification support for an RSVP-based service. Internet Draft, IETF, September 1996. `draft-williams-iss11-vcuse-00.txt`.
- [25] R. Woundy, A. Viswanathan, N. Feldman, and R. Boivie. ARIS: Aggregate route-based IP switching. Internet Draft, IETF, November 1996. `draft-woundy-aris-ipswitching-00.txt`.
- [26] J. Wroclawski. Specification of the controlled-load network element service. Internet Draft, Integrated Services WG, IETF, November 1996. `draft-ietf-intserv-ctrl-load-svc-04.txt`.