Implementation of a Quality of Service Feedback Control Loop on Programmable Routers

Abstract— **Current Diffserv architecture lacks mechanisms for network path discovery with specific service performance. Our aim is to introduce an enhanced-Diffserv scheme utilizing a feedback loop to gather path information and allow better flexibility in managing Diffserv flows. We utilize state-of-the-art programmable routers that can host the control loop operation without compromising their normal routing and switching functionalities. Furthermore, the control feedback loop implemented on the control plane of the router can selectively alter the behaviour of a specific data flow in real-time.**

I. Introduction

Over the years the Internet has been transformed into a fundamental infrastructure that both businesses and individuals rely on. However, it is structurally fragmented into large, heterogeneous network domains controlled by different Internet Service Providers (ISPs). The providers have to rely on a complex collection of operational, management methodologies and techniques in order to operate their networks. In this increasingly competitive environment, it is important for service providers to exercise some active controls over their networks. It is important for them to allow customization of network services to differentiate their offerings by rapidly introducing intelligent services on demand such as QoS (Quality of Service) to their clients. In other words, a programmable networking framework is desirable for service/network providers to manage their service/networks intelligently according to the customers' needs.

Until now the Internet only provides a single best effort service without guaranteeing the timeliness or actual delivery of data. Quality of Service (QoS) architectures have been proposed but not deployed due to a fundamental problem: either the architecture is not scalable or it does not guarantee end-to-end QoS. With IntServ [1], [2], the resource requirements for running per-flow resource reservations on routers increase in direct proportion to the number of separate reservations that need to be accommodated. The use of per-flow state and per-flow processing is thus not cost effective, or even feasible across the high-speed core of a network. On the other hand, Diffserv [3] proposes a scalable service discrimination model without requiring any per-flow state management in the core network. Diffserv focuses primarily on aggregate flows and differentiates between service classes rather than providing absolute per flow QoS measures. Diffserv networks classify packets into one of a small number of aggregate flows or "classes", based on the Diffserv codepoint (DSCP) in the packet's IP header [3]. While the aggregated behavior state of the Differentiated Services architecture does offer excellent scaling properties, it does not guarantee end-to-end QoS for applications. By aggregation, the characteristics of individual flows are lost and hence per-flow QoS requirements cannot be observed without adding relevant mechanisms to DiffServ architecture. The lack of end-to-end signaling facilities makes such an approach one that cannot operate in isolation within any environment [4].

There appears to be no single comprehensive service environment that possesses both service accuracy and scaling properties. A framework for RSVP/IntServ operation over Diffserv networks has been proposed to provide both scalability and end-to-end QoS [5]. For this integration framework to be realized, mechanisms for conveying information about resource availability in a Diffserv network region to boundary routers and some form of signaling from the boundary to the client application must be developed. Currently there is no robust mechanism for network path discovery with specific service performance attributes. No existing mechanisms exist within either Intserv or Diffserv architectures to query the network for the potential to support a specific service profile [4].

The paper demonstrates that an intelligent quality of service (QoS) path discovery feedback loop, as proposed in the enhanced-DiffServ [6], can be implemented in a network environment that includes programmable network elements. We will also demonstrate that the feedback information provided by the loop can be used to activate an active service on a router to alter its behaviour in real-time on commercial routers.

The paper is organized as follow: Section II presents a programmable network element along with the software software support for building and deploying network services dynamically. Section 2 briefly reviews our enhanced-DiffServ architecture and describes our implementation of the control feedback loop on our programmable network testbed. Section IV presents an experiment in which we demonstrate how the control loop can activate an active

service for changing the priority of a specific data flow with our programmable framework. Section V concludes the paper with several directions for further research.

II. Programmable Network

Our QoS scheme relies on programmable elements [7] of a network. A programmable network allows value-added services to be deployed across the network efficiently without degrading the overall network performances [8]. Services can be dynamically injected into certain network elements by network administrators or automatically on demand upon satisfying certain network conditions. The later case allows the programmable network to respond quicker to changing network conditions than traditional networks.

Although programmable network elements does not seem to be readily available, many current commercial routers do in fact have programmable elements built in them. Examples include the Procket Pro/8812 router, and Intel IXP network processors. In this paper we focus on another commercial router from Nortel, the Passport.

The Passport is an enterprise series class of routers which offer programmability, and yet does not suffer from performance degradation in the forwarding of packets. This is achieved through customization of Application Specific Integrated Circuit (ASIC). The router architecture is essentially separated into two planes, the forwarding plane and the control plane. The ASIC lies in the forwarding plane and it offers the ability is to to change the header contents of packets at wired speed (e.g. for use in DiffServ marking), and to selectively filter certain packets and possibly pass them to the control plane; all without impacting the switch speed of the router. The control plane can set the filters at the forwarding plane to alter their forwarding behaviors or redirect and capture certain packets for processing in the control plane.In the Passport family of routers there are two main model: the 8600 and the 1100 series router, both of which have the programmable control plane built in.

Within the control plane, there is an execution environment called the Oplet Runtime Environment (ORE) developed by the Openet project [9]. ORE is a Java runtime environment in which new network services can be deployed and run. It's implemented in Java because its aim is to be platform independent. Currently ORE can run on the Accelar (an older series of routers from Nortel), and the Passport within their control planes, or it can run on a Linux based router. Thus one can develop a new service/protocol on a separate Linux based testbed and be able to fully test it before deploying it on the actual networks. The organization of ORE is illustrated in Figure 1

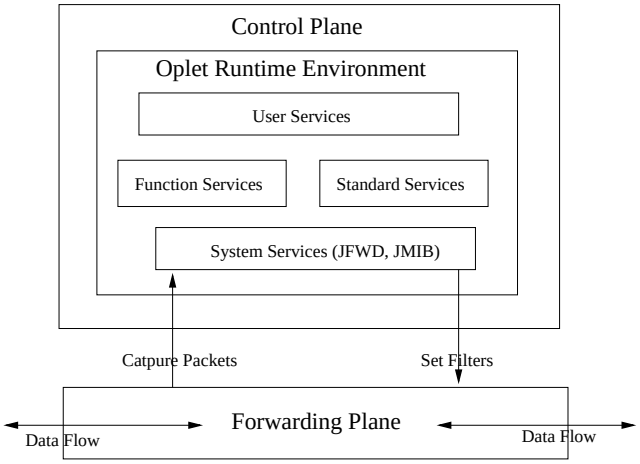

Fig. 1. Oplet Runtime Environment organization in a typical router

At the heart of ORE is a set of APIs for controlling the forwarding plane. Each service deployed in ORE is in the form of encapsulated object called Oplet that is a self-contained, downloadable unit. ORE provides a service hierarchy which separates services into four categories: System, Standard, Function and User. "System Services" are low level services that have direct access to the hardware features. An example is the forwarding service (JFWD), which provides APIs to alter the forwarding behaviour. "Standard Services" provide the ORE standard features for managing service creation and deployment. "Function Services" provide common functionality or utility used to rapidly create user-level services (often these are provided by third-parties). Finally, the "User Services" are application-specific services create by the users. They are built on top of the services from the other three categories. Examples of application services include collecting and analyzing information regarding network conditions, and executing predefined actions such as setting new filters or changing the forwarding behaviors of flows once certain criteria were satisfied.

III. Feedback Control Loop for Enhanced DiffServ

The basic idea of the control and discovery mechanism is depicted in a schematic diagram in the middle (part b) of Figure 2. For each DiffServ traffic class, the source edge router (SER) introduces a Resource Discovery (RD) packet at a rate proportional to the traffic rate for that traffic class destined to the destination edge router (DER). The RD packets contain a vector of QoS parameters as well as traffic parameters for the SED-DER pair. The first core router exercises an intelligent bandwidth allocation algorithm to estimate the bandwidth fair share for this class

based on the traffic parameters contained in the RD packets, and to determine the explicit QoS parameters it can support. The core router then consults its QoS state for that DSCP class and modifies the parameters of the RD packets it can support accordingly. In a lighter version, the core routers only collect relevant statistics and update QoS parameters on the RD packets. The router then forwards the RD packets to the next router along the path.

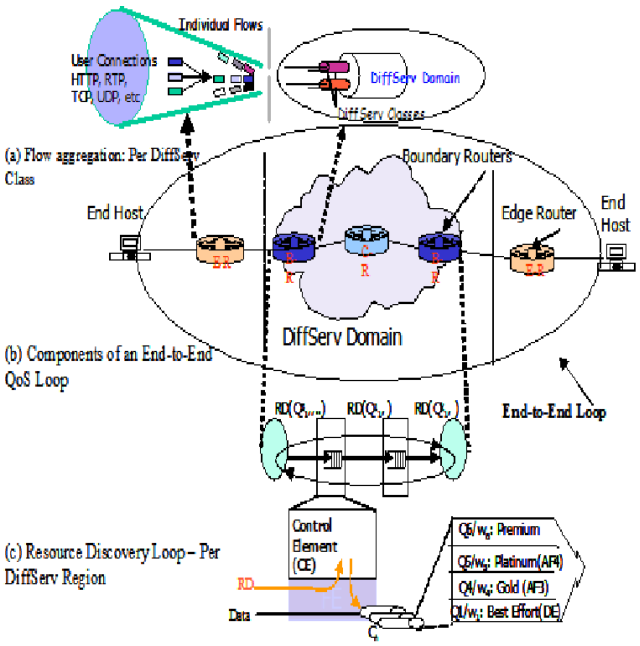

Fig. 2. Enhanced DiffServ Framework

When an RD packet arrives at the DER, that router consults its QoS state (i.e. bandwidth, congestion condition) and modifies the parameters accordingly. The DER returns the RD packet containing the maximum QoS capability that can be supported from the SER to DER. By doing so for each of the DSCP class, the SER discovers the QoS capabilities for this path. Armed with this feedback control mechanism for QoS path discovery, the SER will be able to perform several QoS-related functions. For example, it may want to negotiate an appropriate QoS level for an application or it may perform local admission control.

An implementation of the control loop mechanism is done on the Nortel programmable routers, Passport 8600, 8100, and Linux router utilizing the ORE that are deployed on each type of routers. Figure 3 shows the deployment of the feedback control service on ORE-enabled routers which covers both edge routers as well as core routers. Note that there may be many core routers in the path from SER to DER.

The basic idea behind the feedback control loop implementation is an RD oplet which is deployed at each router on the transmission path. Its function is to generate RD

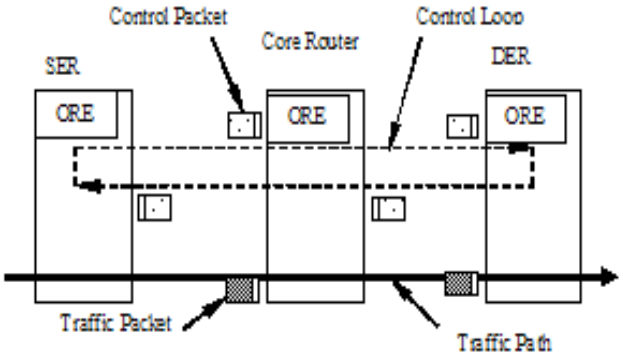

Fig. 3. Implementation of the feedback control loop

packets, process and analyze those RD packets that it receives, and forward them along the path (as is the requirement for core or DER routers).

A key issue with the implementation is the ability of the ORE to capture and process RD packets at each router on the transmission path. On the Passport 8600 router, communication with the ORE is available only through the management port which is kept separate from all other ports for security reason. Packets can be sent from the management port out to the network but it cannot receives packets coming from the normal switching ports. To overcome this restriction, we develop another oplet which offers the ability to set “capturing” filter dynamically based on a packet’s characteristics such as source, destination addresses, ports, protocols, and/or ToS field. Packets matching the said filters are captured from the forwarding plane and passed to the control plane for processing via a system service oplet called JCapture. As a result, RD packets can still be received by the RD oplet for processing by setting the filter to specifically watch out for these RD packets. A limitation with this technique is that the capturing mechanism does not operate at the same wire speed as the normal port switching. Consequently, it is not designed to process every single packet belonging to high bandwidth flow. This is not a severe limitation however, since we only utilize the capturing mechanism for RD control packets whose rate is far less than the data rate.

For the Passport 1100 router, the JCapture mechanism is not fully implemented and a different work around is needed. Rather than using the JCapture system service to capture packet matching the RD filter, these packets are redirected to a dedicated Linux box connected to the router. This dedicated Linux box runs the ORE environment and acts as the control plane for the Passport 1100. The Linux box can communicate with oplets in the router’s ORE via the management port to gather router statistics and set or remove filters to alter traffic behaviors as required.

Finally for standalone Linux router, the capturing mechanism is also available through the use of the pcap library though the restriction on sending and receiving packets to the ORE does not apply here.

IV. Active Flow Manipulation

An important task of RD packets outside the gathering of network QoS path information is the ability to specify actions to dynamically alter the behaviour of traffic flows in real-time. This is referred to as active flow manipulation (AFM) [8]. To this end, we develop an oplet service, JDiffserv, that can set filters to dynamically match the required traffic flow to be altered. Similar to the filters for capturing packets, these filters can match traffic flows based on common packet characteristics such as source, destination addresses, ports range, protocol, and IP ToS field.

For each filter, the oplet can define a Diffserv traffic profile to apply to traffic flow(s) matching it. All the common Diffserv functions are supported. These include marking traffic (admission marking), or re-marking traffic (condition marking) to specific Diffserv class, and policing traffic according to their agreed traffic profile. Also supported is the ability to drop all packets from a flow matching the filter which is useful in situations such as a Denial of Service attack. RD control packets can make use of the services offered by the JDiffserv oplet to perform flow manipulation according the dynamic network information it gathered.

To demonstrate the effectiveness of AFM, we deploy our oplets on a prototype system using the Passport 8600 and Linux routers. The layout for the network topology is shown in Figure 4. There are two incoming flows both destined to the same output link representing the congested link. We want to demonstrate that AFM can be activated through the use of RD control packets which instruct the JDiffserv oplet to perform the required actions. In all our experiments, we utilize the network measurement tool Iperf [10] to generate the required traffic and measure the results.

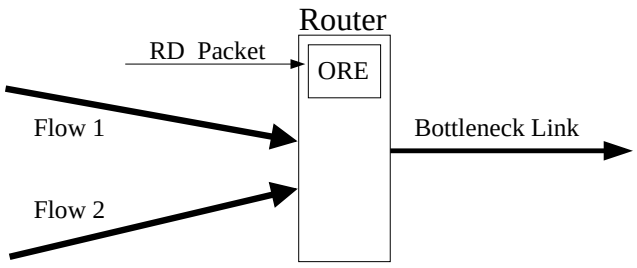

Fig. 4. Experiment Layout

In the first experiment (Figure 5 a), we look at two competing UDP flows which are both set to send at 80Mb/sec. Since the outgoing link is limited to 100Mb/sec, there will be losses experienced by both flows. This can be seen during the initial period (0-35 sec) where both UDP flows fluctuate greatly. At time 33 sec, an RD control packet is sent to the JDiffserv oplet residing in the ORE of the router instructing it to give higher priority to Flow 2 with a traffic profile of 78Mb/sec. and discarding any out of profile traffic. Immediately, we see that Flow 2 reaches its targeted traffic profile while Flow 1 is left with the remaining link bandwidth. At time 100 sec, Flow 2 terminates and we see that Flow 1 quickly utilizes the available bandwidth to transmit at 80Mb/sec.

The second experiment (Figure 5 b) looks at two competing TCP streams. Unlike UDP flows, TCP test stream generated by Iperf does not have a targeted bandwidth but tries to find the maximum available bandwidth. Due to the TCP congestion control mechanism, both flows share the same bandwidth during the initial period (0-32 sec). Once the AFM is activated at time 31 sec through the use of an RD control packet to set a higher priority of Flow 2, the same results as the UDP case can be seen.

Finally we look at the case of a mixed UDP and TCP flow. As before, the UDP flow is set to send at 80Mb/sec. Due to the congestion control mechanism of the TCP and the inherent greediness of the UDP, we see that the TCP flow quickly loses out during the initial period (0-36 sec). At time 35 sec, an RD packet was sent to give a higher priority to the TCP flow and this also quickly takes effect with the TCP flow quickly reaching its targeted traffic profile. Thus AFM can be used to shield badly behaved flows from well behaved ones.

These experiments showed that our RD packets can be used to enable AFM dynamically with the desired results near instantaneously. Another key observation is that the Passport forwarding plane carries out the AFM identification and action of packet flows at the wire-speed, without obvious performance reduction. The reason is because the control service does not require packet processing in the forwarding plane. The forwarding engine of the network node processes and forwards packets as normal while the CPU executes the Java control code implementing the AFM-based service. This application service indicates an immediate benefit of active detection of flows and dynamic adjustment of packet priorities on commercial-grade routers.

V. Conclusion

In this paper, we presented an architecture for enhanced Diffserv which utilizes a feedback control loop for QoS path discovery. This mechanism is lacking in the current Diffserv framework. We showed that such a control loop can be deployed on programmable routers by building a

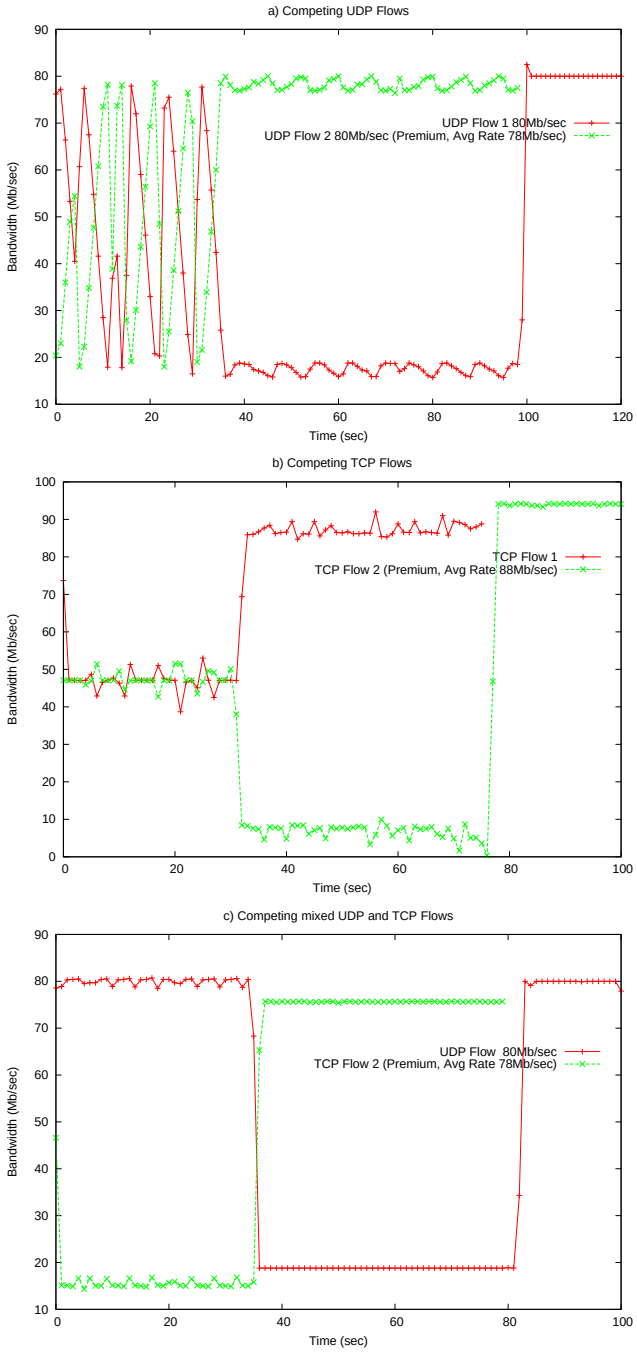

Fig. 5. Results demonstrating active flow manipulations for various competing flows.

prototype, utilizing the Nortel Passport router and its programmable environment, ORE. The advantage of the programmable router is that the capturing and execution of the control loop packets do not interfere with the switching performance of the router. Besides gathering network path QoS, we demonstrated that the control loop can be used to dynamically alter the flow of traffic. This active flow manipulation (AFM) allows greater flexibility to administrators in managing their networks. We showed that AFM can be deployed and activated near instantaneously on demand. New services can be develop and deployed which will be activated on demand based on changing network conditions without manual intervention. Our future work focuses on utilizing QoS path information from the control loop to perform fair admission control at the edge, avoid congestion, and to provide end-to-end QoS negotiation.

REFERENCES

[1] R. Braden, D. Clark, and S. Shenker, "Internet RFC 1663: Integrated Services in the Internet Architecture: An Overview," June 1994.

[2] A. Mankin, F. Baker, R. Braden, S. Bradner, A. Weinrib M. O'Dell, A. Romanow, and L. Zhang, "Internet RFC 2208: Resource Reservation Protocol (RSVP) Version 1: Applicability Statement," September 1997.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "Internet RFC 2475: An Architecture for Differentiated Services," December 1998.

[4] G. Huston, "Internet RFC 2990: Next Steps for the QoS Architecture," November 2000.

[5] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, B. Braden, B. Davie, E. Felstaine, and J. Wroclawski, "Internet RFC 2998: Integrated Services Over Diffserv Networks," Novebер 2000.

[6] D. B. Hoang, Q. Yu, M. Li, and D. Feng, "Fair Intelligent Congestion Control Resource Discovery Protocol on TCP Based Network," in *Interworking 2002 Symposium (IFIP '02)*, October 2002.

[7] T. Lavian, D. B. Hoang, F. Travostino, P. Wang, and S. Subramanian, "A Programmable Service Platform for Internet Service Architecture," To appear in the special issue of IEEE Transactions on Systems, Man, and Cybernetics on technologies promoting computational intelligence, openness and programmability in networks and Internet services.

[8] T. Lavian, F. Travostino P. Wang and, D. Hoang, S. Subramanian, V. Sethaput, and D. Culler, "Enabling Active Flow Manipulation in Silicon-based Network Forwarding Engines," *Journal of Communications and Networks*, vol. 3, 2001.

[9] "Openet project," http://www.openetlab.org.

[10] "Iperf," http://dast.nlanr.net/Projects/Iperf.