International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:12, 2008

Expelling Policy Based Buffer Control during Congestion in Differentiated Service Routers

Kumar Padmanabh, Rajarshi Roy

***Abstract*—**In this paper a special kind of buffer management policy is studied where the packet are preempted even when sufficient space is available in the buffer for incoming packets. This is done to congestion for future incoming packets to improve QoS for certain type of packets. This type of study has been done in past for ATM type of scenario. We extend the same for heterogeneous traffic where data rate and size of the packets are very versatile in nature. Typical example of this scenario is the buffer management in Differentiated Service Router. There are two aspects that are of interest. First is the packet size: whether all packets have same or different sizes. Second aspect is the value or space priority of the packets, do all packets have the same space priority or different packets have different space priorities. We present two types of policies to achieve QoS goals for packets with different priorities: the push out scheme and the expelling scheme. For this work the scenario of packets of variable length is considered with two space priorities and main goal is to minimize the total weighted packet loss. Simulation and analytical studies show that, expelling policies can outperform the push out policies when it comes to offering variable QoS for packets of two different priorities and expelling policies also help improve the amount of admissible load. Some other comparisons of push out and expelling policies are also presented using simulations.

***Keywords*—**Buffer Management Policy, Diffserv, ATM, Pushout Policy, Expeling Policy.

I. Introduction

DIFFERENTIATED Services (Diffserv) is a computer networking architecture that specifies a simple, scalable and coarse grained mechanism for classifying, managing network traffic and providing quality of service (QoS) guarantees on modern IP networks. DiffServ can, for example, be used to provide low-latency, guaranteed service (GS) to critical network traffic such as voice or video while providing simple best-effort traffic guarantees to non-critical services such as web traffic or file transfers. Since modern data networks carry many different types of services, including voice, video, streaming music, web pages and email, many of the proposed QoS mechanisms that allowed these services to co-exist were both complex and failed to scale to meet the demands of the public Internet.

Manuscript received May 5, 2007. Some of the result of this paper was published in the proceeding of IEEE International Workshop of High Performance Switching and Routing [29]

Kumar Padmanabh is with Software Engineering and Technology Lab of Infosys Technology Limited, Bangalore India, phone: +91-80-2852-0261 extn. 58936 email kumar_padmanabh@infosys.com .

Rajarshi Roy is with department of electrical and electronics communication engineering, Indian Institute of Technology, Kharagpur, India. e-mail: royr@ece.iitkgp.ece.iitkgp.ernet.in

A. Differentiated Services: The current state of art

The traditional internet offers best effort service and Diffserv is a Class of Service (CoS) model that enhances the best effort services of the Internet. It differentiates traffic by user, service requirements, and other criteria; then, it marks packets so that network nodes can provide different levels of service via priority queuing or bandwidth allocation, or by choosing dedicated routes for specific traffic flows. A policy management system controls service allocation.

Various quality of service techniques have been proposed or developed that attempt to provide predictable service on the Internet. One technique is Integrated Services (IntServ) and its associated RSVP protocol, it will be discussed later on in this section. Some of the concepts in Diffserv grew out of the IntServ model only. However, Diffserv is a CoS approaches rather than a full QoS approach.

There is one fundamental limitations of best effort method being used in internet. The traditional best effort model of the Internet makes no attempt to differentiate between the traffic flows that are generated by different hosts. As traffic flow varies, the network provides the best service it can; but there are no controls to preserve higher levels of service for some flows and not others. What DiffServ does is attempt to provide better levels of service in a best-effort environment. Following is an intuitive analogy of diffserv which is helpful in understanding it better. The class of service provided in diffserv is similar to classes of service provided in a train. Though, the entire train goes from a particular station to another one, however, passenger traveling in first class, second class and general class get different level of service. A part of the analogy we want to stress is that best effort traffic, like coach class seats on the train, is still expected to make up the bulk of internet traffic. While first class and second class carry a small number of passengers, but are quite important to the economics of the department of rail. The various economic forces and realities combine to dictate the relative allocation of the seats and to try to fill the seats of the train. We don't expect that differentiated services will comprise all the traffic on the internet, but we do expect that new services will lead to a healthy economic and service environment.

The next step of QoS architecture is with regard to QoS in the Internet. Intserv is a bandwidth reservation technique that

builds virtual circuits across the Internet. Bandwidth requests come from applications running in hosts. Once a bandwidth reservation is made, the bandwidth cannot be reassigned or preempted by another reservation or by other traffic. IntServ and RSVP are *stateful*, meaning that RSVP network nodes must coordinate with one another to set up an RSVP path, and then remember state information about the flow. This can be a daunting task on the Internet, where millions of flows may exist across a router. The RSVP approach is now considered too unwieldy for the Internet, but appropriate for smaller enterprise networks (or when used with DiffServ and other techniques, discussed shortly).

Diffserv takes a *stateless* approach that minimizes the need for nodes in the network to remember anything about flows. It is not as good at providing QoS as the stateful approach, but more practical to implement across the Internet. Diffserv devices at the edge of the network mark packets in a way that describes the service level they should receive. Network elements simply respond to these markings without the need to negotiate paths or remember extensive state information for every flow. In addition, applications don't need to request a particular service level or provide advance notice about where traffic is going.

In the IntServ/RSVP environment, applications negotiate with the network for service. IntServ is said to be application aware, which allows hosts to communicate useful information to the network about their requirements and the state of their flows. In contrast, Diffserv in present form is not application aware. Since Diffserv does not listen to applications, it does not benefit from feedback that applications could provide. Since it doesn't know exactly what an application needs, it may fail to provide it with an appropriate service level. In addition, Diffserv is not in touch with the receiving host, so it doesn't know whether that host can handle the services it will allocate.

One could say that the Internet needs both RSVP (or some other full QoS model) and Diffserv. RFC 2990 mentions that both Intserv and Diffserv may need to be combined into an end-to-end model, with Intserv as the architecture that allows applications to interact with the network, and Diffserv as the architecture to manage admission and network resources. This is covered further in RFC 2998 (A Framework for Integrated Services Operation Over Diffserv Networks, November 2000). One approach is to use Diffserv to carry RSVP application messages across the core to another RSVP network.

We are trying to compare diffserv with Inteserv and RSVP. Diffserv can be contrasted with MPLS, which implements connection-oriented virtual circuits on ATM, frame relay, or switched networks. MPLS adds labels (tags) to packets that indicate forwarding behavior, but packets travel across predefined circuits. MPLS is generally more sophisticated and complex than Diffserv, but provides better QoS capabilities.

1) The diffserv architecture:

RFC 2638 states that a differentiated services architecture should keep the forwarding path simple, push complexity to the edges of the network to the extent possible, provide a service that avoids assumptions about the type of traffic using it, employ an allocation policy that will be compatible with both long-term and short-term provisioning, and make it possible for the dominant Internet traffic model to remain best-effort.

Per-Hop Behaviors: A PHB (per-hop behavior) is a basic hop-by-hop resource allocation mechanism. Think of PHB as a particular forwarding behavior that stretches across a network and that provides a particular class of service-being careful not to call it a path, because a path could imply state in the network.

RFC 2475 describes a PHB as a forwarding behavior applied to a particular DS behavior aggregate. A *DS behavior aggregate* is a collection of packets with the same DSCP value crossing a link in a particular direction. When a behavior aggregate arrives at a node, the node maps the DSCP to the appropriate PHB, and this mapping defines how the node will allocate resources to the behavior aggregate. Some example PHBs are described here:

- A PHB that guarantees a minimal bandwidth allocation across a link to a behavior aggregate.
- A PHB similar to the preceding with the added feature of being able to share any excess link capacity with other behavior aggregates.
- A PHB that has resource (buffers and bandwidth) priority over other PHBs.
- A PHB that has low delay and traffic loss characteristics

RFC 2474 and RFC 2475 include sections that describe guidelines for defining PHBs in order to promote consistency and standardization. The guidelines recommend that PHBs be designed to provide host-to-host, WAN edge-to-WAN edge, and/or domain edge-to-domain edge services.

A PHB is implemented with buffer management and packet - scheduling mechanisms. Routers examine the DSCP field, differentiate according to the markings, and then move packets into appropriate queues. An outgoing link typically has multiple queues with different priorities. A scheduling technique is used to move packets in the queues out to the next hop.

Diffserv Network Elements: The Diffserv network consists of a variety of network elements and some specific terminology. Some of the elements are illustrated in Figure-1 and the same is explained next. All of these elements and their associated behaviors are designed to decouple traffic management and service provisioning functions from the

forwarding functions, which are implemented within the core network nodes.

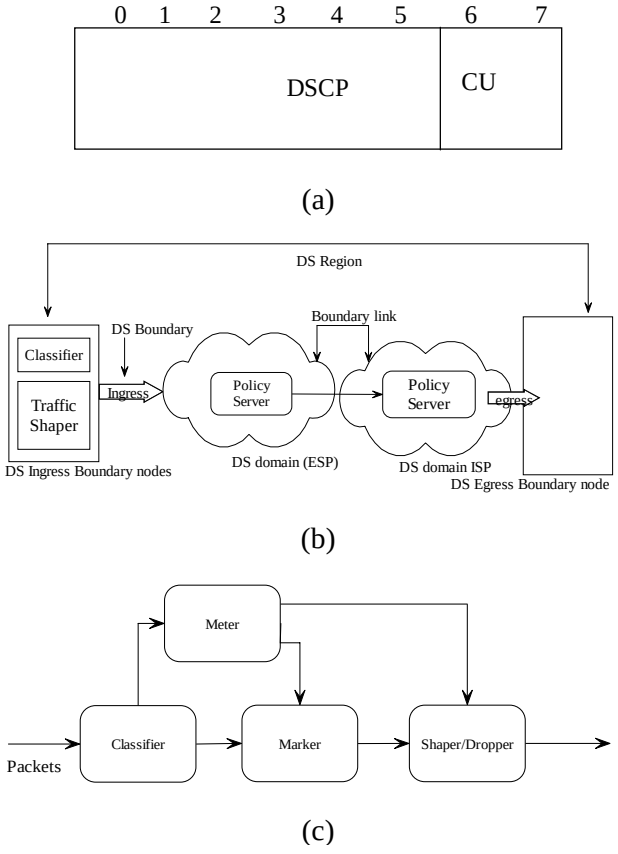

Fig. 1 Various elements of differentiated service (a) Differentiated Service (DS) Field (b) DS network element and (c) traffic conditioner

The most prominent features of Diffserv networks are the *DS domains* and the *DS boundary nodes*. The DS domains may be private intranets, but are typically autonomous service provider networks that have their own service-provisioning policies and PHB definitions. *DS Interior nodes* interpret the DSCP value and forward packets. They may perform some traffic conditioning functions and may remark packets. DS domains interconnect with other domains via boundary links. A *DS region* is a set of contiguous DS domains that offer inter-domain differentiated services.

The DS boundary nodes exist at the edge of the Diffserv network as either ingress or egress nodes. The ingress node is the most important because it classifies and injects traffic into the network. It may also condition traffic to make sure it meets policy requirements. The boundary node contains the following elements.

- **Standard Classifier** Selects packets based on the DS code-point value. Selected packets are then forwarded as appropriate or subjected to traffic conditioning if necessary.
- **Multi-Field Classifier** This classifier selects packets based on the content of some arbitrary number of header fields-typically, some combination of source address, destination address, DS field, protocol ID, source port, and destination port.
- **Marker** An entity that sets the value of the DSCP field.
- **Policy Systems/Bandwidth Brokers** Devices that are configured with organizational policies. They keep track of the current allocation of marked traffic and interpret new requests to mark traffic in light of the policies and current allocation. RFC 2638 provides a broad overview of these system requirements.
- **Traffic Conditioner** An entity that meters, marks, drops, and shapes traffic. A traffic conditioner may re-mark a traffic stream, or may discard or shape packets to alter the temporal characteristics of the stream and bring it into compliance with a traffic profile. The subcomponents of the traffic conditioner are listed here. See "Traffic Management, Shaping, and Engineering" for related details.
- **Meter** Measures the rate of traffic streams selected by the classifier. The measurements are used by the following elements, or for accounting and measurement purposes.
- **Policer** Evaluates the measurements made by the meter and uses them to enforce policy-based traffic profiles.
- **Dropper** Droppers discard some or all of the packets in a traffic stream in order to bring the stream into compliance with a traffic profile. This process is known as "policing" the stream.
- **Shaper** Delays packets within a traffic stream to cause it to conform to some defined traffic profile. A shaper may drop packets if there is not sufficient buffer space to hold the delayed packets.

Traffic conditioners are usually located within the DS ingress or egress boundary nodes, but may also be located in interior nodes within the DS domain. The ingress node of the source domain is the first to mark packets. An egress node that leads to another DS domain may re-mark packets if necessary.

Traffic conditioning rules are specified in a TCA (traffic conditioning agreement) and enforced by the traffic conditioner. TCA rules correspond to SLAs (service-level agreements) made between customers and service providers. These agreements specify the type of service a customer will receive. Note that DS domains within a region include ISPs that are peering with one another and have established peering SLAs.

Diffserv performs traffic conditioning to ensure that the traffic entering the DS domain conforms to the rules specified in the TCA, in accordance with the domain's service provisioning policy. The traffic classifier forwards packets to appropriate traffic conditioning elements.

Traffic conditioners use *traffic profiles* to determine how to condition traffic. A traffic profile defines the rules for determining whether packets are in-profile or out-of-profile. Out-of-profile packets may be queued until they are in-profile (shaped), discarded (policed), marked with a new code-point (re-marked), or forwarded unchanged while triggering some accounting procedure. Out-of-profile packets may be mapped to an "inferior" behavior aggregate.

As mentioned in the preceding list, traffic conditioners contain meters, markers, shapers, and droppers. The meter measures traffic streams against the traffic profile, and the state of the meter affects whether a packet is marked, dropped, or shaped. The following illustration shows packets coming into the classifier. The meter measures the stream and passes information to other elements that trigger a particular action. The marker sets the DSCP value of a packet, effectively adding it to a particular behavior aggregate.

RFC 2597 (Assured Forwarding PHB Group, June 1999) defines a method for defining drop precedence. IP packets are marked by customers or other ISPs with one of three possible drop precedence values. When congestion occurs, the congested DS node protects packets with a lower drop precedence value by discarding packets with a higher drop precedence value.

RFC 2598 (An Expedited Forwarding PHB, June 1999) describes an expedited forwarding (EF) PHB that can be used to build a low-loss, low-latency, low-jitter, assured-bandwidth, end-to-end service through DS domains. Such a service appears to the endpoints like a point-to-point connection or a "virtual leased line." It is useful for voice over IP because it minimizes latency.

RFC 2697 (A Single Rate Three Color Marker [srTCM], September 1999) describes a way to mark packets according to three traffic parameters: Committed Information Rate, (CIR), Committed Burst Size (CBS), and Excess Burst Size (EBS). The srTCM is useful for ingress policing of a service, where only the length, not the peak rate, of the burst determines service eligibility.

RFC 2698 (A Two Rate Three Color Marker [trTCM], September 1999) describes a way to mark packets based on two rates, Peak Information Rate (PIR) and Committed Information Rate (CIR). The trTCM is useful for ingress policing of a service, where a peak rate needs to be enforced separately from a committed rate.

RFC 2859 (A Time Sliding Window Three Color Marker, June 2000) describes a method of marking packets based on the measured throughput of the traffic stream, compared to the Committed Target Rate (CTR) and the Peak Target Rate (PTR). The marker is intended to mark packets that will be treated by the Assured Forwarding (AF) PHB in downstream routers.

Thus diffserv is the future technology of internet where various class of service can be provided with its inherent capabilities. However, the router used in the diffserv based traffic will have to deal with severe versatility. The various policies designed for a typical IP traffic may not be the right candidate to be used for it. The careful designs as well as modifications are needed for this scenario's. Buffer management is a typical policy to enhance the qualities of service. Let us summarize the relevant points on buffer management for diffserv.

B. Buffer Management Policies:

In this paper the shared memory switches has been taken into consideration. It is proved [1, 2] that complete memory sharing with proper buffer management can provide better throughput performance than complete partitioning of memory among output ports or complete sharing without buffer management; however for complete memory sharing, careful design of buffer management is essential. The buffer management policy has to decide whether to accept or reject new incoming packets. The buffer management policy may also decide to drop a few packets, which are in the buffer waiting to be drained by the output ports and were accepted in the buffer in a previous decision epoch. The buffer management policy can trigger the dropping action only when there is not sufficient space available for the new incoming packet or it may even be triggered when the buffer is not full and all the new incoming packets are successfully accepted.

The prime purpose of an ATM switch is to route incoming cells (packets more generally) arriving on a particular input link to the output link, which is also called the output port, associated with the appropriate route [3]. Three basic techniques have been proposed to carry out the switching (routing) function: space-division, shared-medium, and shared-memory [4]. The basic example for a space-division switch is a crossbar switch, which has also served circuit-switched telephony networks for many years. The inputs and outputs in a crossbar switch are connected at switching points called cross-points, resulting in a matrix type of structure. The operation of a shared-medium switch, on the other hand, is based on a common high-speed bus. Cells are launched from input links onto the bus in round-robin fashion, and each output link accepts cells that are destined to it.

1) Shared Memory Switch

The subject of this article, the shared-memory (SM) switch, consists of a single dual-ported memory shared by all input and output lines. Packets arriving on all input lines are multiplexed into a single stream that is fed to the common memory for storage; inside the memory, packets are organized into separate output queues, one for each output line. Simultaneously, an output stream of packets is formed by retrieving packets from the output queues sequentially, one per queue; the output stream is then demultiplexed, and packets are transmitted on the output lines [4]. The block

diagram of an SM ATM switch is depicted in Fig. 1. Examples of early shared-memory ATM switches are CNET's Prelude [5] and Hitachi's ATM switch [6]. Packet switches have another major functionality besides switching, namely, queuing. The need for queuing (also called buffering) arises since multiple cells arriving at the same time from different input lines may be destined for the same output port [4]. There are three possibilities for queuing in a packet switch: buffer cells at the input of the switch (input queuing); buffer at the output (output queuing); or buffer internally (shared-memory) [3]. Shared-memory ATM switches gained popularity among switch vendors due to the advantages they bring to both switching and queuing. In fact, both functions can be implemented together by controlling the memory read and write appropriately [7]. As in output buffered switches, SM switches do not suffer from the throughput degradation caused by head of line (HOL) blocking, a phenomenon inherent in input buffered switches [3, 7]. Moreover, modifying the memory read/write control circuit makes the SM switch flexible enough to perform functions such as priority control and multicast [7]. Issues regarding the routing function of the SM architecture are outside the scope of this article. Likewise, we will not discuss the details of how the memory is organized into logical queues, and how the cells are written in and read out. Our focus is on the problem of buffer allocation. Buffer allocation determines how the total buffer space (memory) will be used by individual output ports of the switch.

2) Static and Dynamic Threshold Policy

In the early research on buffer management policy, there were two policies broadly evolved, namely static threshold policy and dynamic threshold policy. In the static threshold policy, the length of a particular queue was not supposed to cross the predefined threshold. This policy does not let the inputs with higher traffic to consume most of the buffer memory and make sure that input with lower data rate have sufficient memory available in its corresponding queue. However, if one cannot predict the very nature of the traffic of all inputs, this policy will introduce un-fairness among the queue allocation. At particular moment some of the queue will have plenty of memory available in its queue whereas other will be overcrowded. Dynamic threshold policy avoids this type of fairness problem and length of the queue is variable and depends upon the nature of the traffic at particular instance. The Irland [8], and Kamoun and Kleinrock [9] and authors in [10-12] studied these types of policy.

The buffer sharing policies explained in the preceding sections have a common philosophy. An arriving packet is dropped at the instant of arrival, if the switch is at a certain predetermined state in order to accept future arrivals from some other link which promises better throughput than the current arrival. However, there is always a chance that the decision to discard a packet to save space for another link may be a wrong one, and that the saved free space may not be used by other arrivals. In order to eliminate these situations, a delayed resolution policy (DRP) is proposed by Thareja and Agrawala in [14]. The DRP does not discard an arriving packet if there is space in the common buffer. If a packet arrives and the common buffer is full, the arriving packet, or some other packet that was already accepted, is discarded. The decision to drop a packet from a certain port can be made based on the state of the system or based on different priority classes. If the arriving packet is always dropped, then of course the policy is equivalent to CS. Wei et al. propose to drop from the longest queue in the switch, when the memory is full [18]. They call their algorithm drop-on-demand (DoD). This class of policies, in which a previously accepted packet can be dropped, is more commonly known as push-out (PO), and it has been studied with various different queuing systems. For example, push-out schemes have previously been used to provide service to multiple classes of traffic through one output buffer (and link) in an ATM switch [19]. A comparison of schemes in this type of buffer-sharing systems has been provided in [20]. In our context, where multiple output links compete for buffer space, the PO policy, as defined in [18], is appealing for the following reasons: It is fair, as it allows smaller queues to increase at the expense of longer queues. It is efficient, as no space is ever held idle while some queue desires more; thus, overall system throughput should be high.

It is naturally adaptive. When lots of queues are active, their rivalry keeps their queue lengths short; when only one queue is active, it is allowed to become long [12].

3) Buffer management in Diffserv based traffic

If we focus ourselves only at the ATM type scenario we will have packets of same size with perhaps two or more space priorities. However, in the Diffserv [21] model there are versatilities in data packets. The data packets come from difference sources, which create two fundamental situations. There may be two packets, which have same values (cost), but their lengths are different one. Alternatively, there may be two packets whose values are different but have the same lengths. Thus the packets with variable sizes having different space priorities or values will be there. One of the authors of this paper had studied the relative merits and demerits of expelling policy over push out policies in the context of shared memory based ATM switches and demultiplexer before [22, 23]. Therefore, in this work focus is on the relative performance study of push out and expelling policies for the situation when the packets of variable size with two different space priorities is there.

There has been considerable prior work in this area [24]-[28]. Moreover a VLSI implementation has also been done for expelling policy in ATM switches [31]. Our contribution differs from previous works in that it focuses on multi port devices, packets with priorities, packets with variable sizes and considers broad policy classes. The queuing analyses of different buffer management schemes were present in [24]. One of the early papers in this area is [25], which considered optimal memory sharing within the class of blocking based policies. In [26] the optimality of push out from the longest queue under symmetric traffic situation is discussed. Optimality of the push out with threshold policy is discussed in [1] and their treatment includes traffic asymmetry while

only memory less traffic and service model is considered. In [2] authors extend the work of [1] in the context of ATM switches. Hung. et al in [27] provided optimal policies within discarding, push out and expelling classes for the case of a single output port using dynamic programming and sample path based techniques. Researchers including one of the co-authors of this paper have extended the work in [27] in the context of shared memory ATM switches already in [22] and [23]. This work can be viewed as a further extension of that previous work in the context of IP based Diffserv as here packets of variable sizes and of two different space priorities are considered. In [28] competitive analysis techniques were used to evaluate merits of various buffer management policies for such a situation, however, policies like expelling class of policies is there where the packet drop even when the buffer is not full is not considered.

C. Contribution of this paper:

With analyses and simulation studies following two points are found to be the major contributions:

i. This is the first paper on buffer management which extend the idea of expelling class of buffer management policy in the case of in the context of IP based Diffserv routers previous expelling policies studied only for ATM switch. Thus this paper identifies the advantages of expelling policy for a traffic that consists of packets of variable lengths and multiple priorities.

ii. This is the first paper that proposes a method to control QoS of different class of packets using expelling policy in the context for DiffServ routers.

The organization of this paper is as following: The Push out policy is described in section-II. In section-III the statement of expelling policy its analysis is presented. Thereafter, the method to control QoS of two priority classes is describes. In section-IV simulation studies based on real life data is presented.

II. Pushout Policy

In this section, heuristic push out policy is developed for packet dropping, which is defined by the following rules:

(a) If a low priority packet comes and can not find sufficient room in the buffer then the longest low priority packet from the longest queue is picked as a primary candidate for expulsion. If that is not sufficient for accommodation of the new packet then the next longest low priority packet from the same queue for expulsion is also considered. If such a packet is not available in that queue then we go for the longest low priority packet in the next longest queue. However, if more than two packets are required to be dropped for the insertion of the new packet then instead of dropping them the new incoming packet is discarded. A low priority packet is never allowed to push out a high priority packet.

(b) If a high priority packet comes and finds that sufficient room is not available in the buffer then it targets the longest low priority packet from the longest queue for expulsion. If sufficient space is still not there, it targets the next longest low priority packet from the longest queue and so on. If it exhausts all the low priority packets in the longest queue then it goes for the longest low priority packet in the next longest queue and so on. If after expulsion of all the low priority packets, the new high priority packet still needs some more packets to be dropped or if only high priority packets are present in the system then it targets the longest high priority packet from the longest queue for expulsion and then may go for other high priority packets in the longest queue or other queues in decreasing order of length. However, if more than two high priority packets are required to be dropped by this new incoming packet then this new packet itself is dropped instead.

This heuristic prefers packets of longer length for expulsion compared to packets of shorter length, which tends to create more space for future incoming packets. This heuristic also tries to put a bound on the amount of packet loss at the expense of byte loss.

III. Expelling Policies

In this section the heuristic expelling policy is presented. Here, following are the two important rules to execute this policy.

(a) A new incoming packet is treated the same way as it would be treated under the push out policy.

(b) However, while serving a particular output queue if it is found that the amount of high priority data in that queue is more than some threshold then all low priority packets from the head of that queue are dropped till the first available high priority packet and that high priority packet is put into service. If a high priority packet is at the head of the queue it is however always put into service.

A. QoS Control by expelling policy:

Let $L(P_i)$ represents the length of the i_{th} packet represented by P_i. Let us consider that C_1 be the cost (value) of a unit length of lower priority packet and C_2 be the cost of unit length packet of higher priority packet with $C_1 \le C_2$. Further, let us consider that before applying expelling policy, M_1 number of low and M_2 number of high priority packets are available in the buffer. Let us consider that due to a particular value of threshold say, Th_1, X_1 number of extra lower priority order packets gets lost and Y_1 number of higher priority packets get into the queue. So the over all cost (value) of the all packets present in the buffer before applying the expelling policy can be given as $C_1\{\sum_{M_1} L(P_l)\} + C_2\{\sum_{M_2} L(P_h)\}$. Where as after applying the expelling policy the total cost (value) of all packets present in the buffer can be given as

$$C_1\{\sum_{M_1} L(P_l) - \sum_{X_1} L(P_l)\} + C_2\{\sum_{M_2} L(P_h) + \sum_{Y_1} L(P_h)\}.$$

So the total cost of packet is saved by applying expelling policy can be expresses as:

$$\Big\{C_1\{\sum_{M_1} L(P_l) - \sum_{X_1} L(P_l)\} + C_2\{\sum_{M_2} L(P_h) + \sum_{Y_1} L(P_h)\}\Big\}$$
$$-\Big\{C_1\{\sum_{M_1} L(P_l)\} + C_2\{\sum_{M_2} L(P_h)\}\Big\}$$
$$= C_2 \sum_{Y_1} L(P_h) - C_1 \sum_{X_1} L(P_L).$$

Here X_1 the extra lower priority order packets lost and Y_1 the additional higher priority order packets saved. These parameters X_1 and Y_1 depends upon the expelling threshold Th. If decrease expelling threshold is decreased, loss of low priority packets will get increased but at the same time saving of high priority packet also will increase. Thus X_1 and Y_1 both are decreasing function of expelling threshold Th. Thus due to expelling policy we can achieve the saving of total weighted cost given by:

$$C_S(Th) = C_2 \sum_{Y_1} L(P_h) - C_1 \sum_{X_1} L(P_L) \quad (1)$$

With this analysis we have following results to claim:

Claim-1: By applying the expelling class of buffer management policy, we could save the extra "total weighted cost" given by equation-1.

Claim-2: QoS of high priority packets can be improved by dropping more number of low priority packets.

Claim-3: Since in this process number of low priority packets lost also increases, a trade-off between the numbers of high priority packet saved and number of low priority packets lost by setting a proper expelling threshold can be done depending upon the QoS requirement of the packets of both priority class.

Thus it is claimed that by using expelling policy, one can have extra saving of total weighted cost (value). We can satisfy improved QoS requirements of high priority packets and a trade-off between QoS of high priority packets with the same of low priority packet can be done. These show that expelling policy can be used to control QoS of different class of packets. This point is further explained in the simulation studies are presented in next section.

In what follows, it will be also shown that in doing so, loss probability of the packet with high priority improves dramatically while loss probability of low priority packets increases but remains of the same order.

IV. Simulation Studies

For the experimental set up an 8x8 system is simulated with each input line connected to a two stage on off source with probabilistically distributed ON-OFF period as shown in figure-1. The source can generate a packet in ON period and from ON period it can go to off period with a probability p_1. Source does not generate any packet in OFF period. From OFF period source can go to ON period with a probability p_2. The probability distribution of the length of packet generated is directly taken from [30]. Distribution of the sizes of the packet is as following:

a) Size of 50% of the packet is 0.02KB,
b) Sizes of 15% packet are uniformly distributed between 0.02KB to 0.58 KB,
c) Size of 20% of the packet is exactly 0.6KB
d) Size of rest 15% of the packet is uniformly distributed between 0.62KB to 1.5KB.

In the OFF period the source does not generate any packet. Packets are marked as packets of high or low space priorities probabilistically. Simulation runs were set up so that 95 percent confidence interval for packet losses is always less than 10 percent of the measured value.

We have considered two cases of loading the queue with incoming packet. In the first case; when each packet goes to an output queue with equal probability and gets drained by the output line at a rate of 0.4KB per time slot. In the second case the hotspot scenario is considered where there are chances of getting one queue overloaded. The packets generated have following chance pattern to enter into the queues:

(i) There is 30% chance that a packet will join 1st queue.
(ii) There is 20% chance that it will join 2nd queue.
(iii) There is 10% chance that it will join 3rd queue.
(iv) There are 8% chances for a packet to join remaining 5 queues uniformly.

We are never allowed to drop the partially delivered packet, which is in the process of draining by the output line. FCFS order is maintained within each logical queue. We have not put any restriction on the length of the queue. Depending upon the traffic any queue can grows up to any size depending provided the total size of the buffer is within limit.

In what follows, the simulation results for two distinct cases of loading the queue are presented. In first case an incoming packet can join any queue out of 8 queues available with equal probability. In second case the hotspot scenario is taken into consideration where some of queues are suppose to heavily loaded, where as other queue don't have so much loaded. We follow the hotspot scenario as it is described earlier in this section.

A. Case-I uniform loading scheme:

In figure-2, the variation of packet loss probability of packets of various priorities under different buffer management policies versus the load on the system is presented. Here, the total buffer size is kept at 1200 KB and high priority traffic consists of 80 percent of the total load.

In figure 3, buffer size is kept at 1200 KB and load was fixed at 0.83 while the traffic mix was varied. Under the expelling policy as the proportion of high priority packet increases, high priority packets are less likely to get a low priority packet to push out and that causes high priority loss to increase. Low priority loss also increases because the expelling action gets triggered more often and the lack of low priority cells in the system increases the probability that a low priority cell will get pushed out. Here in expelling policies total packet loss is

slightly greater but it remains of the same order, while high priority packet loss decreases dramatically. This improvement of the performance of high priority packet loss at a little compromise of low priority packet loss gives a special merit to expelling policies.

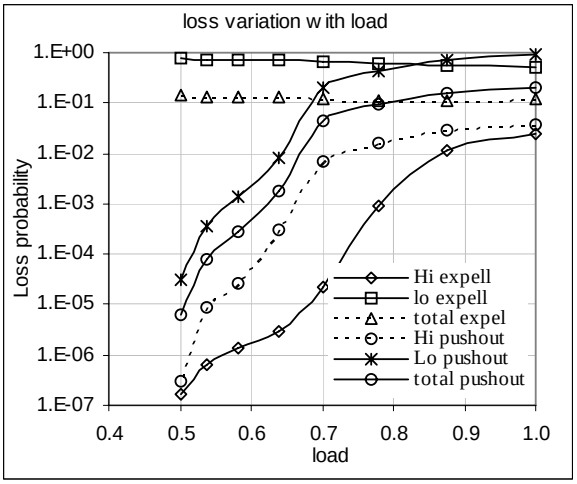

Fig. 2 Loss variation with loads, Buffer Size =1200, percentages of high mix 80%, for Case-I where packets can join any queue with equal probability.

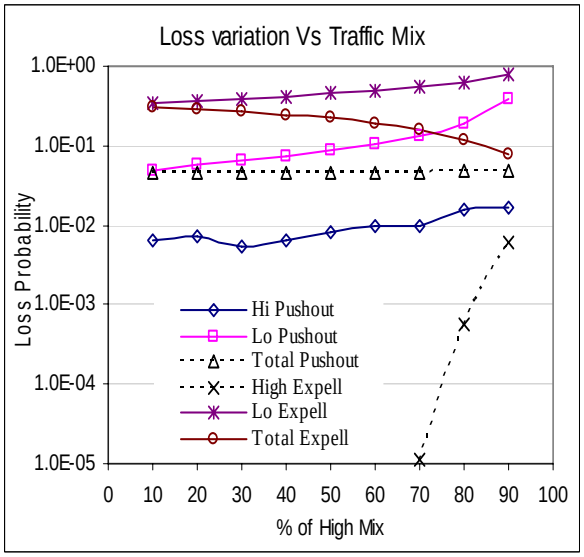

Fig. 3 Loss variation with % of high mix. Buffer size=1200, Load=0.83, for Case-I where packets can join any queue with equal probability

The merit of expelling heuristic policy over heuristic push out policy is demonstrated in figure 5. For example, assume a required value of probability of high priority packet loss of 10^{-5} and that of low priority loss of 10^{-2}. With a high priority mix of 80 percent, it was observed that the maximum admissible load is higher in expelling policy than in push out policy. We have plotted buffer size vs. maximum admissible load in this figure-4.

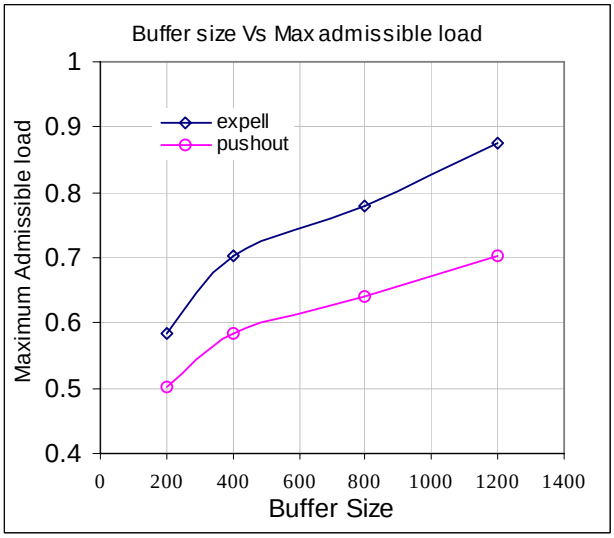

Fig. 4 Maximum Admissible Load Vs Buffer size. Expelling policy can bear more load, for Case-I where packets can join any queue with equal probability

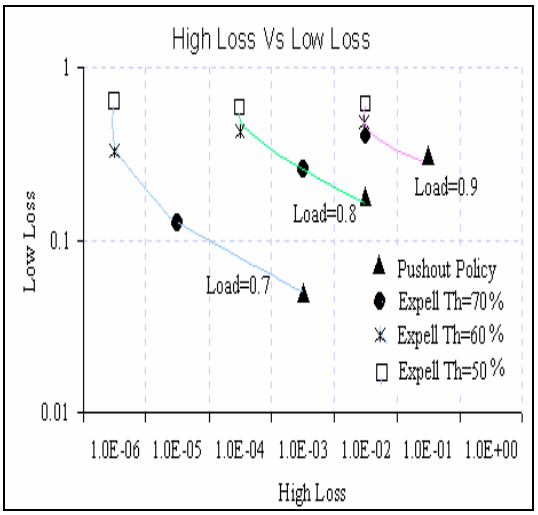

Fig. 5 Although low priority loss remains of the same order high priority loss reduces drastically in in expelling policy, for Case-I where packets can join any queue with equal probability.

However, using expelling policy one can achieve orders of magnitude improvement in the high priority performance and therefore can eventually satisfy its QoS requirement at the cost of moderate low priority performance degradation. The low priority loss remains at the same order. We can achieve this by varying the expelling threshold.

In figure 5 the demonstration of the relative merits of push out and expelling policies regarding this issue is presented. It shows that we have one more control parameter i.e. the expelling threshold. If threshold is decreased, keeping the load at fixed value the high priority packet loss decreases very

rapidly, though the low priority loss increases too. It can be observed that for a load of 0.7 if decrease the threshold up to 50% the loss of low priority packet increases but remains in the order of 10^{-2} while loss probability of the high probability packet decreases from 10^{-3} to 10^{-6}. Thus if one can compromise a bit for low priority packet loss, then high priority packet loss can be reduced up to a great extent.

B. Case-II: Hotspot scenario: Some of the queue is heavily loaded:

Here the same scenario as described earlier in this section is being followed. Here an incoming packet has 25% chance that it will join first queue, 15% chance that it will join second queue, 10% chance that it will join third queue and 8% chance to join remaining 5 queues. Size of the packets and other assumptions are as it is given for case-I.

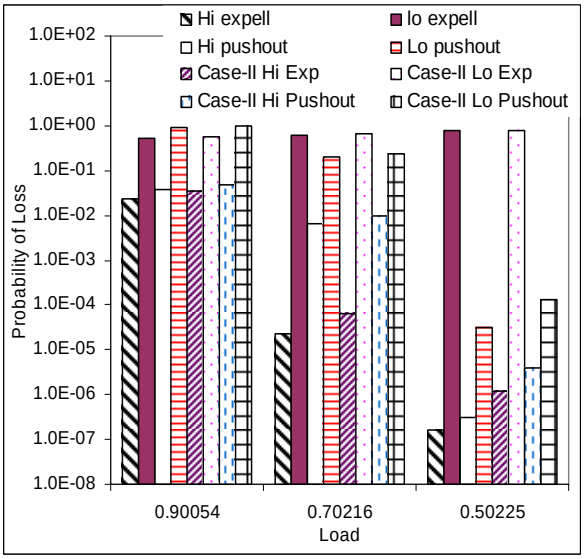

Fig. 6 Probability of loss of low priority and high priority packets in case-I and in case-II. In hot spot scenario where there is chances of losing more packets, expelling policy still lowers the probability of loss of high priority packets.

There are two results out of this simulation study. Due to hotspot scenario there are more chances of loss of both high priority and low priority packets with respect to the same of case-I. But still it can be found that in expelling policy probability of loss of high priority packet is less in comparison of the same of push out policy. Thus in this scenario also, one can easily conclude that by applying expelling policy probability of loss of high priority packets can be lowered if one can bear a little more loss of low priority packets.

V. Conclusion

In Diffserv router have to deal with heterogeneous traffic with multiple priority class of variable packet lengths. The buffer management policy designed for ATM switch may or may not be relevant here. In this paper, the simulation and analysis based studies is presented to explore the relative merit of expelling policies over push out policies is studied in a situation where multi port shared memory systems getting traffic made of variable size packets with two loss priorities is there. Sample path based studies show some sort of dominance results for the expelling policies over push out policies and further exploration of some expelling policy using competitive analysis remains topics of future study.

Reference

[1] Cidon, I.; Georgiadis, L.; *Guerin, R.; Khamisy, A.;* "Optimal buffer sharing" IEEE Journal on Selected Areas in Communications, ,Volume: 13 , Issue: 7 , Sept.1995 Pages:1229 – 1240.
[2] Sharma, S.;Viniotis,Y.; "Optimal buffer management policies for shared-buffer ATM switches", IEEE/ACM Transactions on Networking ,Volume: 7 , Issue: 4 , Aug.1999 Pages:575 – 587
[3] M. Schwarz, Broadband Integrated Networks, Prentice Hall, Inc. 1996.
[4] F. A. Tobagi, "Fast Packet Switch Architectures for Broadband Integrated Services Digital Networks," Proc. IEEE, vol. 78, no. 1, Jan. 1990, pp. 13367.
[5] M. Devault, J. Cochennec, and M. Servel, "The Prelude ATD Experiment : Assessments and Future Prospects," IEEE JSAC, vol. 6, no. 9, Dec. 1988, pp. 157686.
[6] T. Kozaki et al., "32x32 Shared Buffer Type Switch VLSIs for B-ISDN," Proc. IEEE ICC '91, June 1991, pp. 7115.
[7] N. Endo et al., "Shared Buffer Memory Switch for an ATM Exchange," IEEE Trans. Commun., vol. 41, no. 1, Jan. 1993, pp. 23745.
[8] M. Irland, "Buffer Management in a Packet Switch," IEEE Trans. Commun., vol. COM-26, no. 3, Mar. 1978, pp. 32837.
[9] F. Kamoun and L. Kleinrock, "Analysis of Shared Finite Storage in a Computer Network Node Environment under General Traffic Conditions," IEEE Trans. Commun., vol. COM-28, no. 7, July 1980, pp. 9921003.
[10] G. Latouch, "Exponential Servers Sharing a Finite Storage: Comparison of Space Allocation Policies," IEEE Trans. Commun., vol. COM-28, no. 6, June 1980, pp. 9105.
[11] G. J. Foschini and B. Gopinath, "Sharing Memory Optimally," IEEE Trans. Commun., vol. COM-31, no. 3, Mar. 1983, pp. 35260.
[12] A. K. Choudhury and E. L. Hahne, "Dynamic Queue Length Thresholds for Shared-Memory Packet Switches," IEEE/ACM Trans. Commun., vol. 6, no. 2, Apr. 1998, pp. 13040.
[13] H. G. Perros and K. M. Elsayed, "Call Admission Control Schemes: A Review," IEEE Commun. Mag., Nov. 1996, pp. 8291.
[14] A. Erramilli and J. L. Wang, "Monitoring Packet Levels," Proc. IEEE GLOBECOM '94, vol. 1, Dec. 1994, pp. 27480.
[15] B. R. Collier and H. S. Kim, "Efficient Analysis of Shared Buffer Management Strategies in ATM Networks under Non-Uniform Bursty Traffic," Proc. IEEE Magazine, Mar. 1996, pp. 6718.
[16] A. K. Thareja and A. K. Agrawala, "On the Design of Optimal Policy for Sharing Finite Buffers," IEEE Trans. Commun., vol. COM-32, no. 6, June 1984, pp. 73740.
[17] I. Cidon et al., "Optimal Buffer Sharing," IEEE JSAC, vol. 13, no. 7, Sept. Page 17 1995, pp. 122939.
[18] A. Baiocchi et al., "Loss Performance Analysis of an ATM Multiplexer Loaded with High-Speed ON-OFF Sources," IEEE JSAC, vol. 9, no. 3, Apr. 1991, pp. 388-92.
[19] Cisco LightStream 1010 product documentation, available at http://www.cisco.com/univercd/cc/td/doc/pcat/
[20] S. X. Wei, E. J. Coyle, and M. T. Hsiao, "An Optimal Buffer Management Policy for High-Performance Packet Switching," Proc. IEEE GLOBECOM '91, vol. 2, Dec. 1991, pp. 92428.
[21] K. Nichols, V. Jacobson and L.Zhang, ``A Tow-bit Differentiated Services Architecture for the Internet", Internet Draft, July, 1999.
[22] Rajarshi Roy and S. S. Panwar, "Optimal Space Priority Policies for Shared Memory ATM Systems", Proceedings of the 35th Annual Allerton Conference on Communications, Control and Computing, pp.604-613, September-October, 1997.
[23] Rajarshi Roy and S. S. Panwar, "Efficient Buffer Sharing in Shared Memory ATM Systems With Space Priority Traffic", IEEE Communications Letters, Vol. 6, No. 4, April 2002.

[24] Analysis of Shared Finite Storage in a Computer Network Node Environment Under General Traffic Conditions *Kamoun, F.; Kleinrock, L.;* IEEE Transactions on Communications,Volume: 28, Issue: 7 , Jul 1980, Pages:992 – 1003.
[25] G.J.Foschini and B.Gopinath, " Sharing Memory Optimally", IEEE Transactions on Communications,Vol. COM-31, No.3, March 1983.
[26] An optimal buffer management policy for high-performance packet switching *Wei, S.X.; Coyle, E.J.; Hsiao, M.-T.T.;* Global Telecommunications Conference, 1991. GLOBECOM '91. Pages:924 - 928 vol.2.
[27] L. Tassiulas, Y. C. Hung and S. S. Panwar, ``Optimal Buffer Control during congestion in an ATM network Node", IEEE/ACM Transactions on Networking, August 1994, Vol. 2, No. 4, pp. 374-386.
[28] "Competitive Algorithms for High-Speed QoS Switches," PhD Thesis, Tel Aviv University, 2004 by Dr. Alexander Kesselman.
[29] Kumar Padmanabh and Rajarshi Roy, "Expelling Policies for Shared Memory Fast Packet Switches with Variable size Packets of Multiple Priorities", *IEEE, High Performance Switching and Routing HPSR-2005.*
[30] http://www.caida.org/analysis/AIX/plen_hist/
[31] H.J.Chao and N Uzun, "A VLSI sequencer chip for ATM traffic shaper and queue manager" IEEE Journal of Solid state circuits pp. 1634-1643, nov 1992.
[32] Internet Engineering task Force (IETF) http://www.ietf.org/

Kumar Padmanabh : Kumar Padmanabh was born in India in year 1976. He obtainer his PhD degree in 2007 from the department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur, India. Prior to this he obtained Master of Technology in Digital Systems from National Institute of Technology, Allahabad, India and Bachelor of Engineering in Electronics and Communication Engineering from MJP Rohillkhand University, Bareilly, India. He was selected up to the final round of Indian Mathematics Olympiad and he was among top 500 candidates who received certificate of proficiency in mathematics. During his master's course he worked in Bhabha Atomic Research Center, Mumbai in Biomedical Signal Processing. Presently, he is working as a Research Associate at Software Engineering and Technology Labs of Infosys Technology Limited, Bangalore, India. Presently he is working on the issues of wireless sensor networking research to provide based business solution using this technology.

Rajarshi Roy: Rajarshi Roy obtained his PhD degree from Polytechnic University, Brooklyn, NY, USA, and Master of Science from Indian Institute of Science, Bangalore, India and Bachelor of Engineering degree from Jadavpur University, Kolkata, India. Presently he is a faculty in the department of Electronics and Electrical Communication Engineering of Indian Institute of Technology, Kharagpur, India. Prior to this, he worked with Helsinky Institute of Technology, Espoo, Finland; Indian Statistical Institute, Kolkata; Lucent Technology, Bangalore, India; Comverse System, NY, USA, and Bell-Labs USA.