In the Proceedings of the 18th International Conference on
Computer Communications and Networks (ICCCN 2009)

Reducing Buffer Requirements in Core Routers using Dynamic Buffering

Girish B.C. and R. Govindarajan
Supercomputer Education and Research Centre,
Indian Institute of Science, Bangalore 560 012, India
girish@hpc.serc.iisc.ernet.in, govind@serc.iisc.ernet.in

***Abstract*—Earlier studies have exploited statistical multiplexing of flows in the core of the Internet to reduce the buffer requirement in routers. Reducing the memory requirement of routers is important as it enables an improvement in performance and at the same time a decrease in the cost. In this paper, we observe that the links in the core of the Internet are typically over-provisioned and this can be exploited to reduce the buffering requirement in routers. The small on-chip memory of a Network Processor (NP) can be effectively used to buffer packets during most regimes of traffic. We propose a dynamic buffering strategy which buffers packets in the Receive and Transmit buffers of a NP when the memory requirement is low. When the buffer requirement increases due to bursts in the traffic, memory is allocated to packets in the off-chip DRAM. This scheme effectively mitigates the DRAM access bottleneck, as only a part of the traffic is stored in the DRAM. We build a Petri net model and evaluate the proposed scheme with core Internet like traffic. At 77% link utilization, the dynamic buffering scheme has a drop rate of just 0.65%, whereas the traditional DRAM buffering has 4.64% packet drop rate. Even with a high link utilization of 90%, which rarely happens in the core, our dynamic buffering results in a packet drop rate of only 2.17%, while supporting a throughput of 7.39Gbps. We study the proposed scheme under different conditions to understand the provisioning of processing threads and to determine the queue length at which packets must be buffered in the DRAM. We show that the proposed dynamic buffering strategy drastically reduces the buffering requirement while still maintaining low packet drop rates.**

***Index Terms*—Buffer management, Network Processors, Internet core traffic**

I. Introduction

Network processors buffer packets to accommodate variations in network traffic and link utilization, and to avoid packet losses in such scenarios. A widely used rule of the thumb states that the buffer size required is equal to *round trip time* $*$ *transmit rate*. The size of packet buffer has a direct impact on cost and performance of a router and hence has been a topic of interest in recent times [1], [2]. Previous studies on NPs [3], [4] have shown that buffering packets in the DRAM is a performance bottleneck due to the large latency of accessing the DRAM banks and limited bandwidth of the data bus. Appenzeller et al. [1] have shown that for core routers, the buffer size can be significantly reduced due to the statistical multiplexing of a large number of flows. They show that the buffer requirement for a router with 10,000 flows, could be as small as 1% of the initial size suggested by the above mentioned rule of thumb, while still keeping the link utilization above 98%.

We observe that due to over-provisioning of links in a core router, their utilization is low during most regimes of network traffic. The low utilization of the output links can be used as an orthogonal source for reducing the buffer requirement. We propose to use the on-chip receive and transmit buffers for storing the network packets whenever there is no buildup of packets in the NP. However when there is an increase in the number of outstanding packets due to bursty traffic, our approach selectively buffers the packets in off-chip DRAM memory. This avoids the overflow of receive and transmit buffers and hence reduces packet drops. At the same time, it reduces the DRAM requirement in the router and hence its cost. An added benefit of the proposed scheme is that it mitigates the DRAM access bottleneck which was shown to be one of the bottlenecks in achieving higher throughput rates in NPs [3], [4].

For evaluating the proposed dynamic buffering strategy under real workloads, we develop a Petri net model that generates network traffic with similar characteristics as observed in the core of the Internet. This traffic is given as input to a Petri net model of IP forwarding application which implements dynamic buffering. With a mean line rate of 5.41Gbps and variation of 5.52%, the dynamic buffering scheme has a packet drop rate of 0.03% whereas the DRAM buffering scheme has a packet drop rate of 4%. In comparison, even when the traffic rate is increased to 7.39Gbps, our dynamic buffering scheme supports the line rate with a link utilization of 90% and packet drop rate of only 2.17%. Also, we measure the effect of system parameters such as the number of processing threads on performance. With 16 or more threads, we show that the NP can effectively process traffic rates of 6.21Gbps with a packet drop rate of 0.7%.

Though we evaluate the proposed buffering mechanism in the context of IXP 2400 NP, the dynamic buffering scheme is generic and can be applied to other NPs also. This is because the IXP 2400 has a similar architecture as that of a generic NP [5] and we do not assume any IXP specific features for implementing dynamic buffering.

In the next section, we explain the theory from [1] about buffer size buffer size requirements in core routers. Sec. III provides a brief background on NPs and Sec. IV describes our dynamic packet buffering scheme. The Petri net models for traffic generation and the dynamic packet buffering scheme are discussed in Sec. V. Performance evaluation results are presented in Sec. VI. We present the related work in Sec. VII

and conclude the paper in Sec. VIII.

II. Buffer Sizing for Core Routers

Let S denote the sum of the congested windows of the flows passing through the router. The buffer requirement to keep the output link fully utilized is the difference between S_{max} and S_{min}, where S_{max} and S_{min} are the maximum and minimum values of S. When a small number of large flows pass through a congested link, it is observed that there is synchronization among them [6], [7], [8]; i.e., they experience congestive losses at the same instant. In this case, $(S_{max} - S_{min})$ is equal to the product of the aggregate bandwidth and average round trip time [9], [1], [2]. This is referred to as the Bandwidth Delay product. Previous studies which used simulation [9] have also concluded that the buffer requirement should be equal to the bandwidth delay product in order to maintain full utilization of bottleneck network links.

Next, we explain the reasons delineated by Appenzeller et al. [1] for reduced buffer requirement when the output links are uncongested. In a core router, the start time and the propagation delay of the individual flows (t_{pi}) are independent of each other. As a result the flows in the Internet core have varying RTTs. The senders therefore infer congestion at different points of time, leading to desynchronization among flows. The buffer requirement of the individual flows are now out-of-sync [1], [10]. When the flows are desynchronized, the difference between S_{max} and S_{min} reduces. The aggregate congestion window at time t is the sum of the congestion window sizes of the individual flows (W_i). It has been shown using central limit theorem, that the aggregate congestion window size has a Gaussian distribution around the mean congestion window size [1]. Assuming that individual flows vary uniformly between $[\frac{2}{3}W_i, \frac{4}{3}W_i]$, where W_i is the congestion window size of flow i, it has been shown in [1] that a buffer of size of B_{core} that can support a link utilization of 98.99% is given by

$$B_{core} = \frac{bandwidth\ delay\ product}{\sqrt{number\ of\ flows}} \quad (1)$$

III. IXP 2400 Network Processor

The IXP 2400 (Fig. 1) is a programmable NP that can be used for a number of network applications such as IPv4 forwarding, network address translation etc. It has 8 simple in-order processing units called microengines(MEs). Each ME can support up to 8 threads. IXP 2400 has off-chip SRAM and DDR SDRAM memories. The dual-ported quad data rate (QDR) SRAM [11] is used to store the program data like the lookup trie. The SRAM is connected to two on-chip memory controllers. DDR SDRAM with 4 banks is used to buffer packets during processing. The SDRAM is connected to the on-chip memory controller by a 64-bit channel operating at 200MHz. The SRAM size is usually 8MB whereas the DDR SDRAM size may vary from 64MB to 1GB. There is an on-chip scratchpad memory of size 16 KB, that may be used to store temporary variables or setup communication rings between threads.

The NP is used to perform the data plane operations in a router such as receiving the packets from the ingress ports,

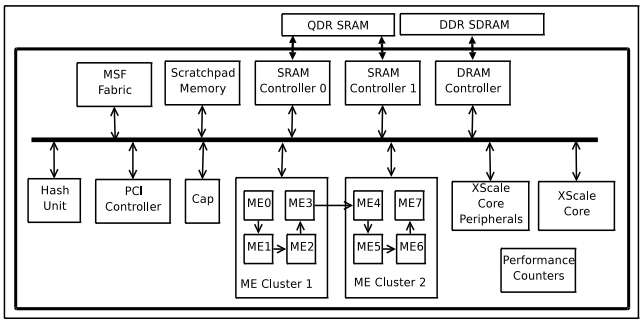


Fig. 1. Architecture of IXP 2400 Network Processor

performing the IP lookup operation and forwarding the packet on the egress ports. It has a Media Switch Fabric (MSF) which is used to connect the MEs to the input and output ports. The on-chip MSF has two buffers called receive buffer (RBUF) and transmit buffer (TBUF), each of size 8KB. An incoming packet is first stored in the RBUF; a free thread is assigned to process it and it is moved to the DRAM. The thread accesses the appropriate part of the packet header, performs a lookup to determine the next hop, modifies the header and moves the packet to the TBUF. The packet is forwarded on the output link from the TBUF. The RBUF is used to store the packet until a thread is assigned for buffering it in the DRAM. The MSF handles the packets in fixed sized cells called mpackets. Further details of this NP are given in [12].

Although we used a specific NP such as IXP 2400, and the traffic rates supported by it, the proposed scheme is applicable in general for other NPs and for larger traffic as well.

IV. Dynamic Packet Buffering

We observe that in the core of the Internet, link utilization is less than 50% [13], [1]. As a result, the network processor can forward packets as soon as the processing is completed and packet buffers could underflow. We use the fact that buffering is required only when there is a backlog of packets to be forwarded and when the output link is fully utilized. Such a scenario occurs only when a burst of packets is received and is infrequent in the core of the Internet due to the presence of a large number of flows and over-provisioning of the links.

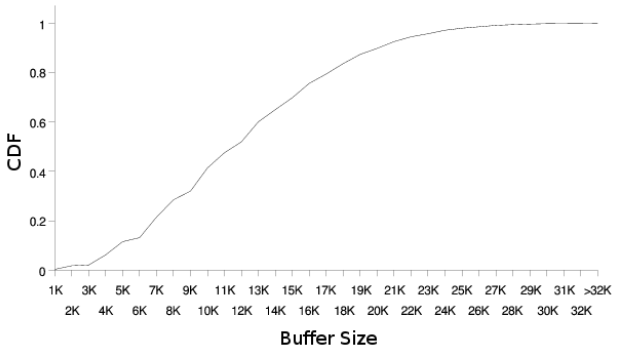


Fig. 2. DRAM buffer utilization in the core Internet

Figure 2 shows the buffer requirements with the traditional DRAM buffering for IPv4 forwarding application with Internet

core like traffic. The x-axis shows the buffer size and the y-axis shows the fraction of time for which the buffer utilization is less than a given value. We used a network core like traffic with a mean input rate of 4.76Gbps. The aggregate bandwidth supported by the output links is 8.1Gbps. This is the typical usage scenario in core routers. We see that buffer utilization does not exceed 36.5KB and is less than 16KB for 77% of the time duration.

This observation shows that, buffering all packets in the DRAM may not be necessary, if the lookup can be performed when the packet is still stored in the on-chip memory. IXP2400 has 8KB of on chip Receive buffer (RBUF) to store packets before they are buffered in the DRAM. Also, there is 8KB of Transmit buffer (TBUF) which is used to store packets before they are transferred to the egress ports. Based on the insight that buffer requirement in core routers can be much smaller than the bandwidth delay product, we propose to use the on-chip memory in the NP with a dynamic buffering strategy wherein packets are stored in the DRAM only when a backlog of packets is developed in the RBUF. When the queue length of the output ports is not very high, the packets could be moved directly from the RBUF to the TBUF. DRAM buffering is necessary only when a burst of packets is received. A packet is buffered in the DRAM when the length of the queue corresponding to its output port increases beyond a threshold, which we call the *high watermark*. Packet build up at the output port implies the need for increased buffering and this larger buffer space is provided by storing the packets in DRAM.

We do not store all packets in the DRAM, but only packets destined to the congested output ports, due to which the traffic to the DRAM reduces substantially. This effectively mitigates the DRAM databus bottleneck that was observed in [4]. When the backlog at the output port reduces, i.e when the queue length for that output port goes below a threshold called the *low watermark*, our scheme reverts to normal processing for that port, wherein packets are directly moved from the RBUF to the TBUF. We refer to the above scheme as dynamic buffering scheme, whereas the original approach where all packets are buffered in the DRAM is referred to as DRAM buffering scheme.

Moving the mpackets directly from the RBUF to the TBUF in the dynamic buffering scheme requires a datapath between the two buffers. This path does not exist in the IXP 2400 architecture or in the generic NP architecture [5]. All the mpackets have to be moved explicitly through the micro engines. We propose the addition of this data path as it could lead to substantial improvements in the throughput of applications and frees up the threads for processing. The addition of this path does not incur significant cost as both the buffers are on chip. In the remaining part of the study, we assume the existence of such a databus with a bandwidth of 3.2GB/sec, which is the same as bandwidth of the internal bus in the IXP 2400.

V. Petri Net Model

A. Petri Net Model for Dynamic Buffering

In [4] we developed a detailed model of IPv4 forwarding application on IXP2400. This model takes into account the time needed to perform the IP lookup operation, the contention for resources and the DRAM packet buffering latency. This model was validated against a cycle accurate simulator of IXP2400. We extend this Petri net model to mimic the dynamic buffering scheme, modeling in detail the DRAM access, where required, the SRAM access, RBUF and TBUF accesses and the queue length in the egress port. In order to evaluate the dynamic packet buffering scheme, we simulate the processing for a representative header processing application such as IPv4 packet forwarding on the IXP 2400 network processor. One queue is maintained for every output port. After the packet lookup is performed, the queue length of the corresponding output port is updated. In case the queue length for that port is above the high watermark, the packet is buffered in the DRAM by moving mpackets to the DRAM. Otherwise, the packet is moved from the RBUF to TBUF. The control words for the packet are suitably updated. The queue length corresponding to each port is adjusted whenever a packet enters and leaves the network processor. All packets belonging to a flow are forwarded to the same output port.

Use of colored Petri nets allows us to compose a complex simulation setup using different components. Traffic from the flow based model (described in Section V-B) is given as input to the IPv4 forwarding application model. The SRAM and hash unit that are used for IP lookup are accurately modeled. A validated DRAM Petri net model described in [4] is used for modeling the DRAM.

In the dynamic buffering scheme, RBUF storage is used when threads are processing the packet, i.e., when the thread is determining the output port of the packet. When the processing threads are unable to keep up with the packet arrival rate, the RBUF occupancy increases. A packet is dropped when there is not enough space in the RBUF to store it.

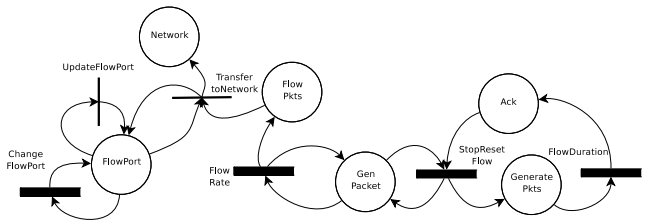
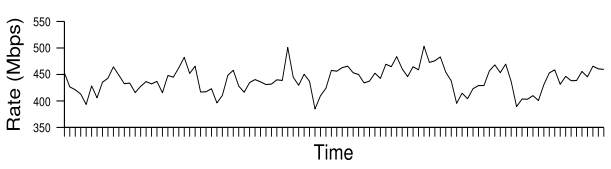


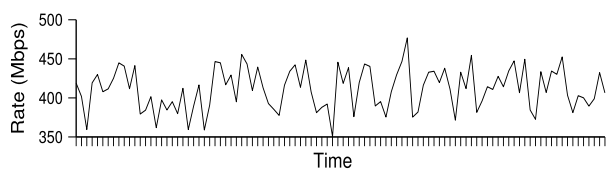
Fig. 3. Petri net model for flow generation

B. Flow Based Traffic Generation Module

In real traces, various characteristics of network traffic such as the packet arrival rate, the packet sizes, the number of packets in a flow and the number of active flows vary with time. Barakat et al. [13] describe an efficient flow based model for uncongested links in the Internet that can capture the dynamics of the traffic at short timescales. The flows have a Poisson arrival rate and the total traffic at any instant of



(a) Real traffic



(b) Simulated Traffic

Fig. 4. Traces from real traffic and simulation

time is the superposition of the active flows. The flow rates are independent of each other. Further, they show that in real traces, the flow sizes and duration are independent of the size and duration of other flows. To keep the analysis simple, they assume each flow as a rectangular Poisson shot-noise process. The *shot* refers to the rate function of the flow. The rectangular shot-noise implies that the rate of each flow is constant.

In order to accurately model the variation in characteristics of the network traffic, we use the above flow based model described in [13]. To generate traffic using the above Poisson shot noise process model, we develop the Petri net model shown in Fig. 3. The rate of packet generation for each flow is determined by the color of the token in *GenPacket* place. The traffic rates of these flows are independent of each other. Based on the color of the token in *GenPacket* place the mean firing time of *FlowRate* transition varies. Each colored token corresponds to the generation of a rectangular Poisson shot noise process and their differing firing times simulates different flow rates. The packets are generated with the length distribution observed in the Internet core for Abilene trace I2C-1091276356-2 [14]. The *FlowRate* transition places a token signifying the length of the packet in the *FlowPkts* place. The packets generated by a flow are associated with a single port based on the color of the token in the *FlowPort* place. The transition *TransfertoNetwork* places the tokens at the output place *Network*.

The *StopResetFlow* and *FlowDuration* transitions are responsible for stopping active flows and starting new flows. When a halt token is present in the *Ack* place, the *StopResetFlow* transition removes the packet generating token from *GenPacket* place, thus stopping the generation of packets for that flow. After the completion of the transition, the token is put back in the *GenPacket* place. The flow is again stopped when the halt token is moved from *GeneratePkts* place to *Ack* place by the *FlowDuration* transition. The transition time of *FlowDuration* corresponds to the interval for which the flow is active. Due to the arrival of new flows and completion of active flows, the number of active flows at a given instant of time varies.

Validation of the Traffic Generation Module: We validate the model by checking that traffic generated has similar variation in rate as the one observed in real network traffic in the core of the Internet. We adopted the method in [13], where variation in network traffic is measured by recording the throughput over 200ms windows. A plot of varying traffic rates observed at 200ms intervals in a real Internet core router (Abilene trace I2C-1091276356-2) and the traffic generated by our model are shown in Figure 4. The real trace has a mean rate of 440 Mbps and a variation of 5.40%. The traffic generated by our model has a mean of 411 Mbps and a variation of 6.45%. Since we need to simulate input traffic at higher rates, we increase the number of flows used for simulation.

Input Traffic (Gbps)	DRAM buffering		Dynamic buffering		
	Transmit rate	Packets dropped	Transmit rate	Packets dropped	Buffered in DRAM
4.90	4.74	1.92%	4.90	0.01%	0.60%
5.57	5.10	4.87%	5.57	0.02%	1.14%
5.42	5.08	3.66%	5.42	0.02%	1.28%
5.51	5.09	4.56%	5.51	0.03%	2.41%
5.67	5.16	5.08%	5.66	0.04%	1.35%

TABLE I
THROUGHPUT AND PACKET DROP RATES WITH A MEAN OF 5.41GBPS AND 5.52% VARIATION

VI. PERFORMANCE EVALUATION

In order to evaluate the performance of the dynamic packet buffering scheme, we study the transmit rates and the number of dropped packets in the router under identical traffic conditions. The aggregate bandwidth of the output links is 8.1Gbps. The mean rate of the input traffic is varied with traffic rate ranging from 5Gbps to 7.2Gbps. The over-provisioning in our experiments is much lower than that in an actual setting in Internet core routers where in the link utilization is usually 50% [1], [13]. We note that having lower provisioning for the output link would require more DRAM buffering and could potentially lead to more packet drops. The transmit rate and packet drop rates are measured over 200ms windows. For this study, we considered Internet core like traffic and not *denial of service (DOS)* like traffic. We show in [4] that the hash unit is the bottleneck resource while processing DOS attack only traffic. As a result, improving the buffering scheme for DOS attack only traffic does not improve the throughput supported by the NP.

A. Impact of Dynamic Buffering

Table I shows the performance of the two schemes over five 200ms intervals at an input rate of 5.29Gbps and variation of 5.52%. In this experiment, we used 40 threads for processing. We observe that when the network rate is high, the DRAM packet buffering scheme drops more packets. This is because of the DRAM bottleneck discussed in [4]. As all packets are transferred over the bottleneck data bus, the DRAM does not have extra bandwidth to absorb bursts of traffic and the network processor experiences packet drops. On the other hand, with dynamic buffering scheme the packet drops are negligible and the scheme is able to support higher throughput. Also, the percentage of packet data that is buffered in the DRAM, due to bursts in flows, is shown in column 6. This percentage is very small as only packets to be output on congested ports are buffered in the DRAM. The processing threads are able to forward the packets at the given input rate.

Input Traffic (Gbps)	DRAM buffering		Dynamic buffering		
	Transmit rate	Packets dropped	Transmit rate	Packets dropped	Buffered in DRAM
5.80	5.18	6.23%	5.78	0.25%	1.43%
5.63	5.10	5.53%	5.61	0.22%	2.88%
5.50	5.07	4.43%	5.47	0.24%	3.37%
7.50	5.86	9.06%	7.27	1.75%	8.36%
6.53	5.78	8.72%	6.44	0.80%	7.05%

TABLE II
THROUGHPUT AND PACKET DROP RATES WITH A MEAN OF 6.19GBPS AND 13.48% VARIATION

Input rate (Gbps)	Threads	Transmit rate	Buffered in DRAM	Packets dropped
6.22	8	5.89	2.88%	2.94%
6.22	16	6.13	10.69%	0.72%
6.22	24	6.18	2.79%	0.34%
6.22	32	6.18	4.02%	0.34%
6.22	40	6.18	3.18%	0.33%

TABLE III
TRANSMIT RATES WITH DIFFERENT NUMBER OF THREADS

Input rate (Gbps)	Threads	Transmit rate(Gbps)	Buffered in DRAM	Packets dropped
7.39	8	6.60	5.95%	5.66%
7.39	16	7.09	11.75%	2.25%
7.39	24	7.10	13.10%	2.17%
7.39	32	7.09	16.04%	2.35%
7.39	40	7.06	14.10%	2.58%

TABLE IV
TRANSMIT RATES WITH DIFFERENT NUMBER OF THREADS AT HIGHER TRAFFIC RATE

Input rate (Gbps)	High Watermark(bytes)	Transmit rate(Gbps)	Buffered in DRAM	Packets dropped
6.33	1600	6.18	27.96%	1.39%
6.33	2240	6.22	16.93%	0.97%
6.33	2880	6.23	12.94%	0.85%
6.33	3520	6.24	8.73%	0.79%
6.33	4160	6.25	4.33%	0.73%

TABLE V
TRANSMIT RATES WITH DIFFERENT THRESHOLD FOR HIGH WATERMARK

As expected, with higher traffic rates, the contention for the output ports increases and therefore the percentage of packets stored in the DRAM increases slightly. Table II shows the performance of the DRAM buffering and dynamic buffering schemes under a mean traffic of 6.19Gbps and higher variation of 13.48%. We see that the dynamic buffering scheme supports higher throughput and lower packet drop rates even with high variation in traffic.

B. Buffer Requirement with Dynamic Buffering

In order to find the buffer utilized during processing, we measured the buffer requirement when a packet arrives at and departs from the NP. A packet arrival at the NP or departure from the NP was referred to as an event. With a mean traffic of 6.19Gbps, the buffer requirement was less than 10KB for 94% of the events and with a mean traffic rate of 5.42Gbps, the buffer requirement was less than 10KB for 99% of the events. The maximum buffer requirement was 36KB and 24KB respectively. We observe that more buffer space than the RBUF and TBUF memory is necessary to store packet only during bursts in traffic. With dynamic buffering, this additional memory is provided in the DRAM.

C. Effect of System Parameters

The number of threads dedicated for processing and the value of the high water mark that is used to determine the need for DRAM buffering, affect the packet drop rate and the amount of traffic that is buffered in the DRAM.

In order to understand the number of threads that must be provisioned for network processing, we ran simulations for 400ms traffic with different number of threads with a fixed input traffic rate of 6.22Gbps. The throughput and packet drop rates achieved with different number of processing threads are shown in Table III. With only 8 threads, the packet drop rate is about 3%. This is because the small number of threads are unable to keep up with the rate of packet arrival. Due to overflow in the RBUF, packets are dropped even before they can be buffered or forwarded. Consequently, the throughput achieved drops. However, when the number of processing threads is increased to 16 or more, each of the packets experiences a lower waiting delay in the RBUF. As a result, packet drop rate reduces. Packet bursts lead to buffering in the DRAM and there is a slight increase in the DRAM traffic with 24 and 32 threads. However, when there are 40 threads, IPv4 processing is more resilient to packet bursts as packet drop rate is the lowest. Table IV shows the performance of dynamic buffering with a higher input traffic rate of 7.39Gbps. We observe that packet drop rates are around 2.5%. This is due to larger packet build up in the RBUF. However, the packet drop rate with dynamic buffering is still less than the drop rate observed with DRAM buffering which dropped more than 5% of the packets with a traffic rate of 6.19Gbps (refer to Table II). Interestingly, the packet drop rate does not decrease even with an increase in the number of threads because the high utilization of the RBUF and TBUF.

Next, we assess the effect of the high watermark on the amount of traffic buffered. We simulated the performance of the network processor with different values for this threshold. A low threshold could excessively buffer all packets in the DRAM due to which the DRAM bottleneck previously observed would reappear. Table V shows the throughput and packet drops with various values of the high watermark. Input traffic with a mean of 6.33Gbps and 32 threads (4MEs with 8 threads enabled) was used. The high watermark in column 2 is the length of the queue associated with each port, beyond which all packets belonging to this queue are buffered in the DRAM. When the high watermark is set to 1600 bytes, about 28% of the traffic is buffered in the DRAM, requiring the RBUF to DRAM and DRAM to TBUF transfer of packets, reducing the bandwidth necessary to absorb traffic bursts. This, in turn causes more packets (1.39%) to be dropped. However, when the value of the high watermark is increased, the amount of traffic through the DRAM decreases. In this case the available DRAM bandwidth is able to absorb the packet bursts, leading to a decrease in the packet drop rate. The packet drop rate and transmit rate do not change substantially when the value of the high watermark is above 2880 bytes. This is due to the small fraction of traffic that is buffered in the

Resource Utilization	Thread Configuration (ME X Threads)				
	1 x 8	2 x 8	3 x 8	4 x 8	5 x 8
Hash Unit	14.2%	15.0%	15.0%	15.0%	15.0%
Thread	27.9%	14.5%	9.1%	6.8%	5.1%
ME	54.3%	28.6%	19.1%	14.3%	11.4%
Data Bus	3.2%	6.0%	4.8%	5.5%	3.40%
Avg. RBUF usage	1444B	1356B	1293B	1284B	1249B
Avg. TBUF usage	1432B	2250B	2310B	2327B	2368B
Packet Drop Rate	2.94%	0.72%	0.34%	0.34%	0.33%

TABLE VI
UTILIZATION OF RESOURCES IN THE NP WITH MEAN INPUT TRAFFIC OF 6.22GBPS

DRAM.

D. Resource Utilization with Dynamic Buffering

The Petri net model allows us to analyze the utilization of resources in order to understand the hurdles in achieving higher throughput rates. Table VI shows the resource usage with a mean input traffic of 6.22Gbps and different configurations of MEs and threads. We observe that the hash unit utilization remains constant around 15%. As expected, the utilization of threads decreases as their total number increases. Data bus utilization is below 6% in all the configurations. The data bus bottleneck that previously manifested when all packets where buffered in the DRAM [4], has been effectively removed. The RBUF and TBUF occupancy has a bearing on the ability of the NP to absorb sudden increase in traffic. A low utilization of these resources enables the NP to store packets when there is a traffic burst.

The role of the RBUF is to provide storage to packets when there is a burst of packets. The number of threads available for processing determines the duration for which the packet is stored in the RBUF. Fewer processing threads implies that the RBUF could overflow leading to packet drops. Thus, we find that the size of the RBUF and the number of threads provisioned for processing affect the packet drop rate.

VII. RELATED WORK

A number of papers have recently studied the size of packet buffers for Internet links. Dhamdhere, et al. [2] provide a generalized analytical model to determine the size of packet buffers in routers, under various constraints on link utilization, total packet loss rate and queuing delay. They show that when the output link is congested, the buffer size should be equal to the bandwidth-delay product. However when the output links are not congested, as in the case of core Internet routers, the buffer requirement is given by the Stanford model [1]. These models assume that the output link has to be fully utilized. Dynamic buffering strategy is motivated by the observation that in core routers, the link utilization is low as the output links are over-provisioned. As a result buffer requirement can be further reduced. Here DRAM memory is used for buffering only when there is a burst of packets in the traffic or there is congestion in the output link.

Shorten et al.[10] discuss changing the back-off factor in the additive increase multiplicative decrease (AIMD) congestion control algorithm of TCP in order to reduce the size of buffering required. By changing the responsiveness of TCP to infer congestion, it can be made to adapt to a small buffer size. While this scheme requires changes in TCP implementation, our scheme does not envisage any change in the sender's TCP implementation.

VIII. CONCLUSIONS

In this paper, the low link utilization in core Internet links has been exploited to reduce the buffer requirement. We show that different on-chip storage locations present in IXP 2400 can be used to buffer packets during most regimes of the input traffic. The proposed dynamic buffering strategy effectively utilizes the on-chip memory for buffering and avoids the slow DRAM access: a bottleneck in network processing applications. We use a Petri net model to generate traffic with the same characteristics as real traces and evaluate the dynamic buffering strategy in detail. With network traffic of 5.41Gbps, the proposed dynamic buffering scheme has a lower packet drop rate than the traditional DRAM buffering scheme. With 16 or more threads for processing, the application can be more resilient to packet bursts and has low packet drop rates. The choice of the high water mark determines the amount of traffic that is buffered in the DRAM. The utilization of the resources under different traffic rates are measured. DRAM buffering bottleneck that was previously observed has been removed.

REFERENCES

[1] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *SIGCOMM '04: Proc. of the 2004 conf. on Appn, technologies, arch, and protocols for comp. comm.*, 2004, pp. 281–292.

[2] A. Dhamdhere, H. Jiang, and C. Dovrolis, "Buffer sizing for congested internet links," in *INFOCOM 2005: 24th Annual Joint Conf of the IEEE Comp and Comm Societies*. Proceedings IEEE, 2005, pp. 1072–1083.

[3] S. Govind and R. Govindarajan, "Performance modeling and architecture exploration of network processors," in *QEST '05: Proc. of the Second Intl. Conf. on the Quantitative Evaluation of Systems*, 2005, p. 189.

[4] B. C. Girish and R. Govindarajan, "A petri net model for evaluating packet buffering strategies in a network processor," in *QEST '07: Proc. of the 4th Intl. Conf. on the Quantitative Eval. of Systems*, September 2007, pp. 19–28.

[5] T. Wolf and M. A. Franklin, "Design tradeoffs for embedded network processors," in *ARCS '02: Proc of the Intl Conf on Arch of Computing Systems*. London, UK: Springer-Verlag, 2002, pp. 149–164.

[6] S. Schenker, L. Zhang, and D. D. Clark, "Some observations on the dynamics of a congestion control algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 20, no. 5, pp. 30–39, 1990.

[7] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.

[8] L. Zhang and D. D. Clark, "Oscillating behaviour of network traffic: A case study simulation," *Internetworking: Research and Experience*, pp. 101–112, 1990.

[9] C. Villamizar and C. Song, "High performance TCP in ANSNET," *SIGCOMM Computer Communications Review*, vol. 24, no. 5, pp. 45–60, 1994.

[10] R. N. Shorten and D. J. Leith, "On queue provisioning, network efficiency and the transmission control protocol," *IEEE/ACM Tran. on Net.*, vol. 15, no. 4, pp. 866–877, 2007.

[11] "QDR SRAM - The High Bandwidth SRAM family," http://www.qdrsram.com.

[12] *Intel IXP2400 Network Processor Hardware Reference Manual*, PDF file, Intel, November 2003.

[13] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, and P. Owezarski, "A flow-based model for internet backbone traffic," in *IMW 02: Proc of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 35–47.

[14] "National Laboratory for Applied Network Research," http://pma.nlanr.net.