



## Effective Block-Sized Reservation-Based Packet Buffer

Srivalli Namachivayam

Computer Science and Engineering, SRM University,  
Chennai, India

**Abstract**—Router is a device in core networks which process incoming packets and forward them towards their destination. Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination. Routers contain packet buffers to minimize packet loss by holding packets during times of congestion and packet buffers provide a deterministic bandwidth. Two types of memory technologies can be chosen to build a packet buffer. They are Static Random Access Memory (SRAM) and Dynamic Random Access Memory (DRAM). SRAM is more expensive per bit, low capacity and has a short access time. DRAM has high capacity, low cost per bit and a long access time. DRAM is needed to implement large packet buffers but inappropriate DRAM based architecture are too slow to match the bandwidth requirements of high-performance routers. DRAM-based architectures can be categorized currently as prefetching-based and randomization-based for supporting line speed queue operations. They are all based on interleaving memory accesses across multiple parallel DRAM banks for achieving greater memory capacity/bandwidths, but they vary in their packet placement and memory operation scheduling mechanisms. In this paper, we present an effective block-sized reservation-based packet buffer based on the concept of blocks. This concept reduces the memory resources by efficient utilization of bank interleaving using hybrid SRAM/DRAM system which significantly improve scalability and packet scheduling towards higher line rates.

**Keywords**—prefetching-based, randomization-based, packet buffer, packet loss, packet scheduling

### I. INTRODUCTION

High-performance routers need to store temporarily a large number of packets in packet buffer at each line card in response to network congestion. One of the important criteria in designing high performance routers is the implementation of packet buffers that operates or works at tremendously fast line rates. At each time slot  $t$ , a new packet  $p$  arrives can be written to the packet buffer, and an existing packet can be read from the packet buffer for departure. To afford adequate packet buffering at times of network congestion, a common buffer sizing rule is

$$BS = CTT \times L \quad (i)$$

where  $BS$  is the size of the buffer,  $CTT$  is the average round-trip-time of a flow passing through the link, and  $L$  is the line rate [2]. At an average round-trip-time of 250 ms on the Internet [3] and a line rate of 40 GB/s (OC-768), this rule of thumb translates to a memory requirement of over 1.25 GB at each line card, which clearly makes an SRAM implementation impractical. On the other hand, the worst-case random access times of DRAM are not fast enough to match line rates at 40 GB/s or beyond, making DRAM based solution inadequate.

Existing architectures are based on prefetching-based [4], [5] and randomization-based [6], [7]. Prefetching-based architecture involves hybrid of SRAM and DRAM features. This supports line speed queue operations by aggregating packets and prefetching packets for parallel DRAM transfers using fast SRAM caches. It requires complex memory management algorithms and a substantial amount of SRAM for caching to handle worst case access patterns. Randomization-based architectures are based on random placement of packets so that the memory loads across the DRAM banks are balanced.

Switch-memory-switch router architecture is the one where the departure of packet buffers can be deterministically calculated in advance before the packet is inserted in the packet buffer considering best route is chosen. This architecture is based on a set of physically separated packet buffers that are inserted between the crossbar switches. Departure time of the packet is deterministically calculated when a new packet arrives in the incoming queue. Based on the time calculated, it is stored accordingly on the outgoing queue. Implementations of such router architecture include M-series Internet core routers from Juniper Networks.

Another router architecture is the load-balanced router architecture [8], [9]. This architecture is also based on the designed similar to Switch-memory-switch router architecture but only difference is that there is no central scheduler and the two switches have a fixed configuration that is independent of the arrival traffic.

In this paper, we present an efficient/effective reservation-based packet buffer architecture based on the concept of blocks that supports deterministic packet departures. Our architecture differs from the previously proposed reservation-based design in that our solution is based on aggregating packets into blocks so that the amount of book-keeping information in SRAM is minimized. In particular, the total size of the reservation table only grows logarithmically with respect to the total packet buffer size, which results in an order of magnitude reduction in the total size of SRAM. Furthermore, we prove that the required number of interleaved DRAM banks is only a small constant independent of the

arrival traffic patterns, the number of flows, and the number of priority classes. Therefore, our proposed design is scalable to growing packet storage requirements in routers while matching increasing line rates.

The rest of the paper is organized as follows; Section 2 discusses some background and related work. Section 3 discusses the approach in depth with the detailed design and implementation approach. The conclusion is made in Section 4.

## II. RELATED WORK

### 2.1. Designing Packet buffers

Routers need packet buffers to hold packets during time of network congestion. Congestion occurs when the packets arrive faster than the speed of the outgoing packet. For example, packet may come from multiple lines and output may be directed to the single line. If the output line is overloaded continuously chances are there that the packets will eventually overflow. So, it is very important to design the appropriate packet buffer.

Packet buffers are arranged as a set of one or more FIFO queues. A router typically keeps a separate FIFO queue for each service class at its output; routers that are built for service-providers such as Cisco, Juniper etc. In addition, switches and routers commonly maintain virtual output queues (VOQs) to prevent head-of-line blocking at the input, often broken into several priority levels; it is common today for a switch or router to maintain several hundred VOQs.

Earlier, packet buffers were easy to build: The line card would typically use commercial DRAM to divide it into either statically allocated circular buffers (one circular buffer per FIFO queue), or dynamically allocated linked-lists.

There are four ways to design a fast packet buffer:

- (1) Using Static RAM as SRAM is faster than DRAM. But SRAM is small, expensive and power-hungry.
- (2) Using DRAMs with faster access time.
- (3) Use multiple DRAMs in parallel to increase memory bandwidth.
- (4) Create a memory hierarchy of SRAM and DRAM with the speed of SRAM and the cost of DRAM.

### 2.2. Designing Memory hierarchy

Memory hierarchy should be designed in such a way that it should always accept a packet and always should be able to read a packet when requested. It can be designed properly if we decide on

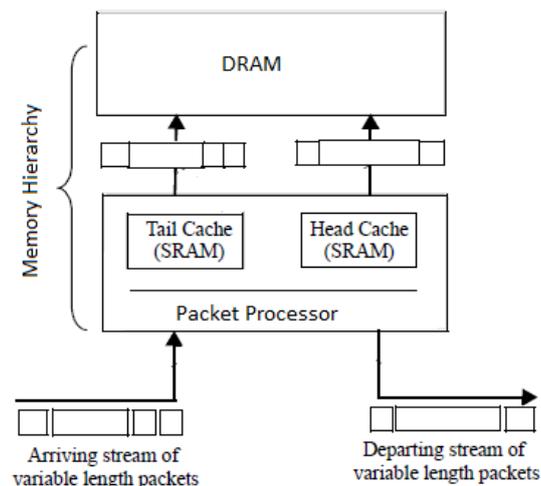
- (1) When to write blocks of packets from tail cache to DRAM.
  - (2) when to read blocks of packets from DRAM so that tail cache SRAM does not overflow.
  - (3) How much SRAM is needed for head cache and tail cache so that we can try to minimize the size of SRAM.
- Memory hierarchy can be designed using various options:

- (1) Guaranteed versus Statistical: Does occasional miss is allowed? When a word is not found, a miss occurs. It is assumed that packet buffer behaves like SRAM and thus no overruns at tail cache or underruns at head cache.
- (2) Pipelined versus Immediate:

Is it allowed to tolerate delay? With the pipeline delay, there is a fixed delay between all read requests and packets getting delivered. With the Immediate option, no delay is allowed.

- (3) Dynamic versus Static Allocation:

Head and tail buffer cache can be managed statically or dynamically based on the design implementation/decision preferred/chosen.



### 2.3. Architectures

As explained in Introduction section, different architectures are studied to improve the efficiency and utilization. Each architecture has its own pros and cons.

- (1) Prefetching-based
- (2) Randomization-based
- (3) Switch-memory-switch router
- (4) Load-balanced

### III. IMPLEMENTATION IDEA

In this section, we describe the efficient block-sized reservation-based packet buffer that we aim to implement. Incoming packets are variable-length packets. Variable-length packets are segmented into fixed-length packets for simplicity while entering the router. Again, these packets are reassembled when leaving the router.

To preserve the line speed of SRAM, our architecture deploys multiple DRAM banks as an interleaving memory structure to imitate the behavior of a large SRAM. The DRAM bus carries blocks from and to DRAM. The more efficiently it is utilized the narrower it can be dimensioned, while providing the same bandwidth. DRAM banking, i.e., interleaved accessing of DRAM banks, allows using the DRAM data bus efficiently, i.e., leads to a high data bus utilization. However, due to complex timing constraints that have to be met the achieved bandwidth highly depends on the sequence of the accessed memory locations. To keep DRAM accesses deterministic, the memory management algorithms (MMA) have to access the individual banks systematically. E. g., when the MMA accesses the banks in a strict round robin manner there is enough time between two accesses to the same bank to prepare the bank for the new access. Usage of banking implies that the architecture has parallel resources – the individual banks.

Deterministic packet service models exist for the architectures where the departure times of packets from packet buffers can be deterministically calculated precisely in advance, before the packet is inserted into a packet buffer, when best-effort routing is considered. This architecture can efficiently mimic an output queuing switch. The departure time of the packet is deterministically calculated to exactly mimic the packet departure order of an output queuing switch when a new packet arrives at incoming queue. A matching problem is then solved to select a packet buffer to store the packet. Departure time had already been selected when the packet gets inserted into a packet buffer.

To reduce the memory requirement, we propose an effective block-sized reservation-based packet buffer architecture which is built on the concept of blocks that supports deterministic packet departures.

Proposed architecture employs semi-parallel hybrid SRAM/DRAM System (SPHSD). It contains hybrid memory architecture i.e. utilizes both SRAM and DRAM. It also contains a set of parallel DRAMs. It consists of three main parts: the tail part containing the tail buffer (SRAM), parallel DRAMs, and the head part containing the head buffer (SRAM).

#### Parallel DRAMs:

The core of the SPHSD consists of  $k$  parallel DRAMs or DRAM banks that can be operated independently. A DRAM in Figure 1 represents a resource that can be accessed once per random access time  $T$ . Consequently, such a DRAM can be also implemented by an individual bank of a DRAM device. This enables in combination with the proper MMA the utilization of banking.

Each DRAM provides  $1/k$  of the required bandwidth and contains  $Q$  FIFO flow queues, that is, each logical flow queue is spread over all  $k$  DRAMs. The SPHSD aggregates packet data per flow to constant-size blocks. As always full blocks are written to DRAM and read from DRAM the total DRAM bandwidth is dimensioned to  $2R$ . Hence, each DRAM provides a bandwidth of  $2R/k$ , i. e.,  $R/k$  for reading and  $R/k$  for writing. The random access time of a DRAM is  $T$  and so each DRAM performs one read and one write every  $2T$ .

Access time ( $2T$ ) and bandwidth ( $R/k$ ) of a single DRAM define the block size as

$$b = 2TR/ K \quad (ii)$$

#### Tail Part:

The tail part is location on the left side of the DRAMs just below the reservation table in Figure 1. Its task is to write incoming packet data to the DRAMs in a deterministic way. Therefore it aggregates packet data per-flow to constant-size blocks, distributes full blocks to the DRAMs and buffers full blocks when the corresponding DRAM is temporarily overbooked.

#### Head Part:

The head part delivers packets requested by an external scheduler in-order with constant read latency. Therefore, for each packet, it requests the blocks from the DRAMs that contain the packet, reorders these when necessary and finally reassembles the packet from the blocks. The head part is located on the right side of the DRAMs just below Reorder departure buffer in Figure 1. Since the SPHSD is symmetric, the tail part and the components processing requests in the head part are similar. The only difference is that the components in the tail part operate on packet data, while the others operate on requests.

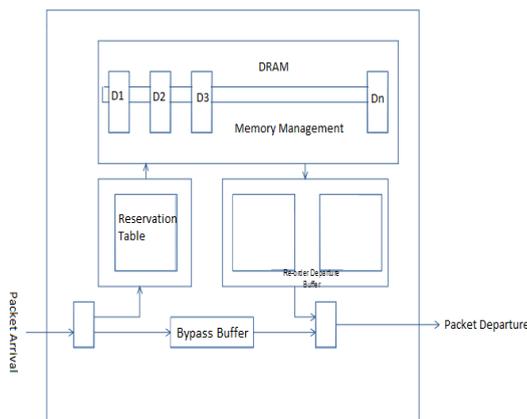


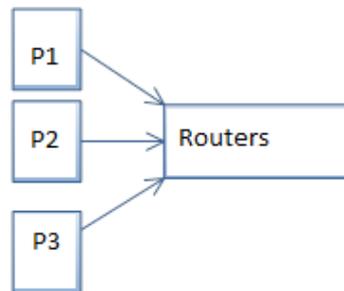
Figure 1 Detailed design of efficient block-sized reservation- based packet buffer

Modules include,

- Data Transmission
- Reservation Table
- Memory Management
- Data Retrieved

*Data Transmission:*

Data transmission involves segmenting all the incoming variable-size packets into fixed-length payload segments/packets/cells when a router is forwarding the packet. Packets are reassembled when leaving the router. Packet buffer architecture is said to implement a time deterministic packet buffer that is, at every time slot, a new packet can be written into the packet buffer and a new packet can be read from the packet buffer in accordance to its pre-determined departure time.



*Reservation Table:*

Reservation table is necessary to keep track of the packets in DRAM banks. Reservation table serves as a memory management module to select one bank out of all available banks to store an arriving packet such that it will not cause any conflicts at the time of packet arrival or departure. Once the reservation table has been updated with the value, the packet can be written to the one of the parallel DRAM banks or the bypass buffer, which can occur simultaneously with the reservation table processing for the next packet. If the packet is stored in the corresponding packet location in DRAM bank, then it is marked as 1. Otherwise it is marked as 0. Even though only 1 bit is required per packet, it grows with the number of DRAM banks location.

Reservation table grows with the size of the banks where size of the bank is represented as  $sizeR = N_{max} \cdot \log_2 N / N$ . (iii)

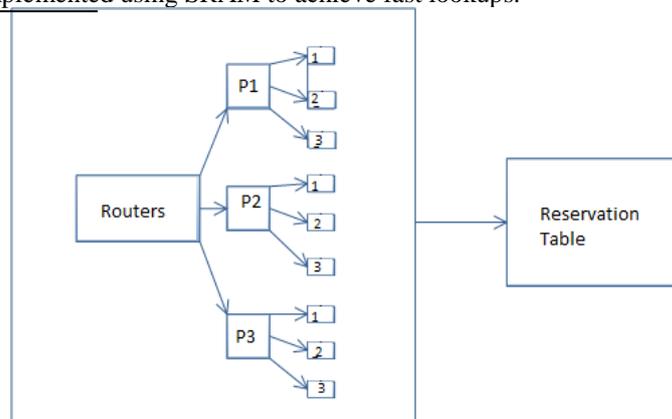
If  $N = \alpha * N_{max}$ , where  $\alpha$  is a constant and  $\alpha < 1$ , then

$sizeR = N_{max} \cdot \log_2 (\alpha * N_{max}) / (\alpha * N_{max})$  (iv)

Therefore the size of the reservation table grows logarithmically as the DRAM bank size grows. The size of the reservation table can be reduced at the cost of larger departure reorder buffers implemented in SRAM.

The number of interleaved/interlaced DRAM banks required to implement the proposed packet buffer architecture which is independent of the number of logical queues, yet the proposed architecture can achieve the performance of an SRAM implementation and scale to growing packet storage requirements in routers while matching increasing line rates.

Enough DRAM banks should be available to store all the incoming packets and to depart all the outgoing packets. Reservation table can be implemented using SRAM to achieve fast lookups.



*Memory Management:*

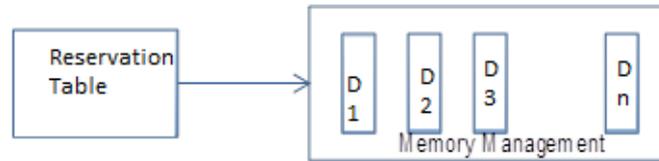
Packets are reserved and send to respective time slot. Advanced memory interleaving schemes share the idea of utilizing multiple DRAM banks to achieve SRAM throughput, while reducing or eliminating potential bank conflicts. Existing DRAM based designs can be classified into two types – Prefetching-based [4], [5] and Randomization-based [6], [7]. Prefetching-based architectures support line speed queue operations by aggregating and prefetching packets for parallel DRAM transfers using fast SRAM caches.

Most of the packet buffer architectures assume a very general model where a buffer consists of many logically separated FIFO queues that may be accessed in random orders. But in our proposed architecture, instead of random order access, we go with sorted order access based on deterministic departure.

The number of blocks in the packet buffer numberBlock is

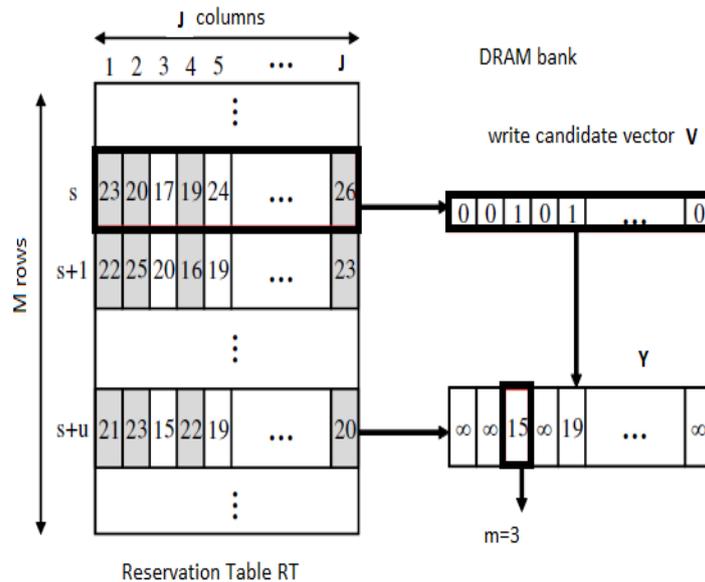
$$\text{numberBlock} = \text{CTT} * L / \text{bPN} \quad (v)$$

where CTT is round trip time, L is line rate, P is packet size, b is time slots and each row of the reservation table represents N frames.



To store the incoming packet in selected DRAM bank, we need to follow the below steps:

- (1) Use the write candidate vector V to check incoming arrival conflicts and outgoing departure conflicts in the DRAM banks. The vector can be maintained either separately or constructed upon necessity.
- (2) Reservation table row RT(s+u) is accessed.
- (3) To find the memory bank for storing the incoming packet, compare vector V and reservation table row RT(s+u) to find the most empty free bank using bit-wised operation
 
$$Y = \text{RT}(s + u) \cdot 1/V \quad (vi)$$
- (4) If a bit in vector V is 1, then the corresponding entry in vector Y is the same as the entry in RT(s + u). Otherwise, the entry in vector Y is infinite.
- (5) Find the smallest entry Y(m) in vector Y with ties broken arbitrarily.



**Data Retrieved:**

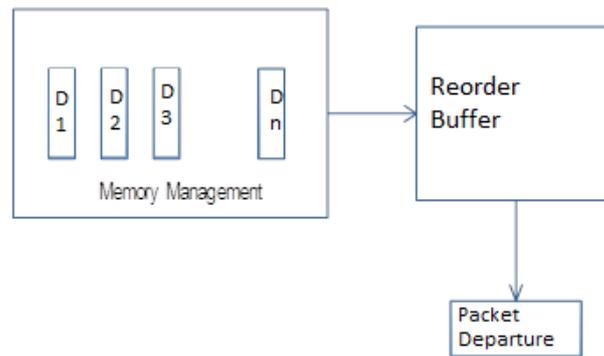
Memory hierarchy gives statistical guarantees to access a packet to interleaving or prefetching used in computer systems. A simple technique to obtain high throughputs using DRAMs is to stripe a packet across multiple DRAMs.

In this approach each incoming packet is split into smaller packet and each packet is written into different DRAM banks. DRAM banks reside in a number of parallel DRAMs in memory management. To decrease the access rate to each DRAM, packet interleaving can be used, consecutive arriving packets are written into different DRAM banks. When we write the packets into the buffer, we determine the time in advance and we store it in DRAM banks on order and so it can happen that consecutive departing packets reside in the same DRAM row or bank, causing row or bank conflicts and momentary loss in throughput. There are other techniques which give statistical guarantees where a memory management algorithm (MMA) is designed so that the probability of DRAM row or bank conflicts is condensed. These designs include that arbitrarily select memory locations so that the probability of row or bank conflicts in DRAMs are considerably reduced.

Packets from the memory block are moved to a departure reorder buffer. Reorder buffer has to be selected in such a way that it has to be large enough to accommodate departure packets in-order after a constant read latency.

Therefore, the reservation-based designs are scalable to growing packet storage requirements in routers while matching increasing line rates.

Number of the vital interfaced DRAM banks is a small constant independent of the arrival traffic pattern, the number of flows, and the number of priority classes, making it scalable to the growing packet storage requirements in future routers while matching increasing line rates.



#### IV. CONCLUSIONS

In this paper, we described a packet buffer architecture based on interleaved memories that takes advantage of the known departure times of arrival packets to achieve simplicity and determinism. A key contribution of this paper is an efficient block-based reservation table design that only grows logarithmically with the line rates, which leads to more than an order of magnitude reduction in SRAM requirements for this class of reservation-based architectures. Further, in our proposed architecture the number of the required interleaved DRAM banks is a small constant independent of the arrival traffic pattern/configuration, the number of different flows, and the number of precedence classes, making it scalable/accessible to the growing packet storage requirements in future routers while matching increasing line rates.

We have also shown that SRAM is used to minimize in block-sized reservation-based packet buffer to improve efficiency and utilization. As shown in Reservation Table section, as number of DRAM banks increases, size of the reservation table decreases and bypass buffer increases. And once the reservation table has been updated, packet can be written into DRAM banks concurrently while reading next packet thus improving the performance.

#### REFERENCES

- [1] Hao Wang, Bill Lin, Reservation-Based Packet Buffers with Deterministic Packet Departures, Volume 25, 2014.
- [2] C. Villamizar and C. Song, High performance tcp in ansnet, SIGCOMM Comput. Communication. Rev., volume 24, page 45 to 60, no. 5, 1994.
- [3] CAIDA, Round-trip time measurements from caida's macroscopic internet topology monitor. [Online]. Available:<http://www.caida.org/analysis/performance/rtt/walrus0202>.
- [4] S. Iyer, N. McKeown, R. R. Kompella, Designing packet buffers for router linecards, IEEE/ACM Trans. Netw., volume 16, no. 3, page 705 to 717, 2008.
- [5] J. Garcia, J. Corbal, L. Cerda, M. Valero, Design and implementation of high-performance memory systems for future packet buffers, page 373, 2003.
- [6] G. Shrimali, N. McKeown, Building packet buffers using interleaved memories, Workshop on High Performance Switching and Routing, page 1 to 5, May 2005.
- [7] S. Kumar, P. Crowley, J. Turner, Design of randomized multichannel packet storage for high performance routers, page 100 to 106, August 2005. 264, 2002.
- [8] C.S. Chang, D.S. Lee, Y.-S. Jou, Load balanced birkhoff-von neumann switches, part i: one-stage buffering, Comput. Communication, volume 25, page 611 to 622, 2002.
- [9] I. Keslassy, S.T. Chuang, M. Horowitz, O. Solgaard, N. McKeown, K. Yu, D. Miller Scaling internet routers using optics, SIGCOMM Comput. Communication. Rev., page 189 to 200, 2003.
- [10] Alper T. Mzrak, Keith Marzullo, Stefan Savage, Detecting Malicious Packet Losses, page 191 to 206, 2009.
- [11] Amogh Dhamdhere, Constantine Dovrolis, Open issues in router buffer sizing in 2006.
- [12] K. Ramakrishnan, S. Floyd, D. Black, Addition of Explicit Congestion Notification (ECN) to IP, RFC 3168, IETF, 2001.
- [13] Ben Kenwright, Fast Efficient Fixed-Size Memory Pool, Computation Tools, 2012.
- [14] Manjunath Reddy, Efficient and Novel Distributed Packet Buffers and High-Bandwidth Switches and Routers, IJERT, 2013.