

Ultra High Speed Packet Buffering using “Parallel Packet Buffer”

Sailesh Kumar, Raja Venkatesh, Joji Philip, Sunil Shukla
Paxonet Communications, CA, USA

Abstract

Modern switches and routers often use dynamic RAM (DRAM) in order to provide large buffer storage space. However, the effective bandwidth of DRAM is frequently a limiting factor in the design of high-speed switches and routers. The focus of this paper is to introduce a packet-buffering architecture called the parallel packet buffering (PPB), which increases the effective memory bandwidth significantly. PPB incorporates n cheaper, narrower and smaller DRAM modules to emulate a common memory buffer. An efficient read/write scheme and memory selection policy ensures the optimum usage of all the memory modules. PPB is particularly useful for extremely high-speed packet buffering applications. We will show that, for most of the traffic profiles, the architecture utilizes the total buffering space much more efficiently than other architectures for high speed packet buffering. In addition, our simulation results show that the PPB architecture achieves much higher bandwidth than a single memory with same aggregate data-width for any traffic.

Introduction

The explosive growth of Internet traffic has created an acute demand for integrated-service networks of ever increasing bandwidth capacity. High-speed networks need the fastest possible switches and routers to provide reliable service. The number of ports and the line (port) rate are parameters that will grow. Port speeds are in the OC-12 to OC-192 (622 Mbps to 10 Gbps) range today, and will grow to OC-768 (40 Gbps) [6] very soon. The number of ports in a single networking device are in the tens to hundreds range today, and will soon number in thousands. Thus the bandwidth requirement of any networking device (particularly routers and switches) is increasing rapidly and terabit devices are on the horizon.

Any device with a store and forward architecture needs to buffer the incoming traffic at the line rate [3][11]. The buffered data then has to be processed and forwarded to the appropriate port at the line rate. All aggregation, scheduling, packet classification and most of the layer-2 & 3 devices belong to this category. A layer-2 or 3 device also needs to perform the header lookup from a common lookup memory table for all the incoming packets. For aggregation and scheduling [4], the device should be capable of buffering large amount of data arriving at the line rate for further processing. Typically, every node in a network should be capable of buffering an amount of data equal to $(R_{tt} \times \text{line rate})$ [5] where R_{tt} is the round trip time of the network. Therefore, as the line rate grows, the buffering space requirement of any node also increases proportionally. This means that the device must have a large, logically single shared memory to buffer all the incoming data at all ports. The bandwidth capacity of this memory should match the line rate of the device. The line rates are ever increasing exponentially but the memory bandwidth is not increasing at that

rate and its growth rate is as low as 8%. Memory access time for the fastest available commercial DRAM is about 30ns, and has made very slight development in recent years. Today, the speed at which the incoming packets can be buffered in the memory is causing a severe bottleneck in the scalability of the networking devices. Memory bandwidth often turns out to be a hard-hitting constraint in the design of high-speed switches and routers [7]. Simply put, network switches and routers are running out of memory bandwidth.

All these factors have contributed to the inception of several new packet-buffering architectures [2][8][9][10][11][12], aimed at maximizing the buffering capability of networking devices. These architectures generally use multiple memory modules to emulate the single buffer of large size and bandwidth. Different memories are accessed simultaneously in parallel for read/write operations thereby increasing the effective data-width of the buffer. Henceforth in this paper, we will call such architectures as “logically single memory” based architectures because even if there are multiple memories, they are accessed simultaneously in parallel just to increase the data-width. However, the logical data bus width of any memory buffer has a limit irrespective of the memory technology or buffering architecture. It shouldn't exceed the minimum packet size arriving at the ingress or data-bus will remain under-utilized for each write operation of such packets. For example if the minimum packet size to be handled by a device is 40 bytes (320 bits), then there is no point in exceeding the data-width of the memory buffer beyond 320 bits. This itself imposes the limitation on the bandwidth scalability of such memory buffers. With the current memory technology, the fastest DRAMs with 320-bit wide data bus can only support OC-192 rates.

Another weakening fact is that the performance of such memory architectures consistently degrades for a random traffic pattern comprising of variable packet sizes. The aggregate bandwidth and single read/write latency doesn't remain constant and deterministic. Also, the effective utilization of the total buffering space decreases considerably. For example, a buffering architecture optimized for buffering 64 byte packets will work perfectly fine for incoming traffic comprising of 64-byte packets. However, the same architecture will impose a severe bandwidth bottleneck for traffic comprising of 65-byte packets. Thus, such traffic generally results in significant wastage of memory space due to lack of a finer granularity in partitioning of the memory.

In this paper, we propose a new packet-buffering architecture called as PPB (Parallel Packet Buffering) that increases the overall buffering capability significantly both in terms of bandwidth and size. It guarantees efficient utilization of the total buffering space for all kinds of traffic pattern and variable packet sizes. The memory modules used are DRAMs with minimal individual bandwidth requirement that results in lower cost and higher capacity in terms of size. The architecture is generic enough to support any bandwidth capacity as high as OC-3072 and above.

Parallel Packet Buffer

The objective is to develop a DRAM based packet buffering architecture having the following set of features:

1. Scalability to achieve extremely high bandwidth capacity with the existing memory technology, not achievable by any logically single memory based architectures.
2. Comparatively higher bandwidth capacity than logically single memory based architectures with similar memory and aggregate data-width.
3. Finer granular accesses to memory modules for reading and writing packets. This ensures that even for a large variation in the packet sizes, bandwidth of the buffer doesn't deteriorate.
4. Efficient utilization of the total buffering space available. This is to ensure that the total available memory is utilized completely and efficiently for packets of any random size.
5. Fast and deterministic memory module arbitration logic and efficient read/write logic without any contention for each individual memory access. This ensures that the device performance doesn't deteriorate for the fluctuating traffic pattern and random read/write sequence.

PPB incorporates n individual DRAM memory modules to emulate the common memory buffer. Each module has same size, data-width and access time. Value of n can be chosen depending on the bandwidth and latency requirements of the packet buffer. All the n memory modules are accessed simultaneously and independently. Total amount of data that can be buffered is the aggregate buffering capacity of all the n memory modules since the incoming packets are buffered in each of the memories. However, a complete incoming packet is always written into one memory module only. The read and write operations are performed in a pipelined manner in each individual memory module. Thus, while a packet is getting written into one memory module, a newly arrived packet can be written in some other memory module, which is not being accessed currently. This pipelined and simultaneous access to the individual memory modules boosts the aggregate bandwidth while reducing the load on each individual memory. The aggregate bandwidth capacity of all the n memories must equal the line rate of the device. For a given aggregate bandwidth, as the value of n increases, bandwidth requirement of the individual memory modules goes down. Therefore, the data-width of the individual memories is inversely proportional to n . Thus for higher values of n , write and read operations to each individual memory can be performed with a finer granularity. Each write access is of the same size as the data-width of each individual memory. This makes the PPB architecture immune to varying packet sizes. Our simulation results will show that as n increases, the effective utilization of buffering space and overall bandwidth increases significantly for all traffic profiles.

A memory module in the PPB is said to be "idle" when neither read nor write operation is being performed on it else it is termed as "busy". A packet is written into one of the "idle" memory modules only. There is an arbiter that selects one out of all the "idle" memory modules for each of the incoming packets. For each incoming packet, PPB Arbiter (ref. Figure 1) selects a memory module and starts writing the packet into that memory module. One complete packet is always written in one memory module only. For each incoming packet, PPB performs the following sequence of operations.

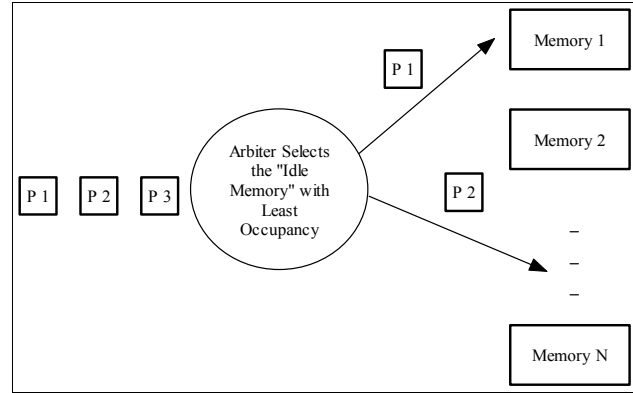


Figure 1 Top-level schematic diagram of the "Parallel Packet Buffering" architecture

1. Select an "idle" memory module, where the complete packet will be written.
2. Update the status of the chosen module as busy and start writing the packet into it and look for the new packets arriving.
3. When a packet has to be read out, select the appropriate memory module where the packet is residing, update the status of the module as "busy" and start reading out the packet.
4. As soon as the read/write operation on any memory module is over, update the status of the module as "idle".

Figure 2 shows the PPB packet buffering belonging to a single queue in the n memories.

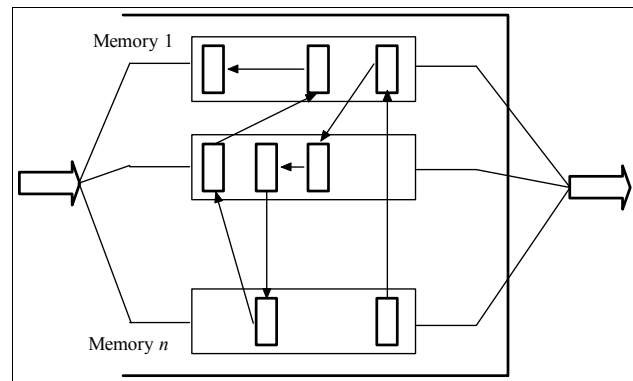


Figure 2 Packet queuing in an n memory based PPB

Selection of the memory module

A fast, efficient and deterministic policy has been devised to select the memory module where the incoming packet has to be written. PPB arbiter maintains the status and occupancies of each of the n memory modules.

For any incoming packet, PPB selects an "idle module" with the least occupancy level. And if none of the modules are found idle at that time, the PPB waits for one of the module to become idle.

As the line rate of device is equal to the aggregate bandwidth capacity of the n modules, the bandwidth capacity of an individual

module is less than the line rate by a factor of n . Hence, it is possible that whilst a packet is being written into the selected module, a new packet arrives before the current packet gets written completely. And this will happen more often for higher values of n .

But we prove that PPB arbiter selection policy always finds an “idle” memory module for each arriving packet if the ingress data rate doesn’t exceed the aggregate capacity of the PPB memory.

Proof:

In PPB, there are n memories (all makes up the common memory buffer) and the aggregate bandwidth capacity of PPB has to be at least twice the line rate R . This is because; data would be simultaneously written into them as well as read out from them. Thus at any point of time, on an average, reads will be performed on at most $n/2$ memories if the egress rate doesn’t violates the line rate. So, we can say that at any point of time, there will exist at least $n/2$ memories where read operation is not taking place. Thus we get our Lemma 1.

Lemma 1: At any time read operation will be performed on $n/2$ memories only.

Now, we have to prove that we will be able to write incoming data packets into the $n/2$ idle memories, provided ingress rate doesn’t violates the line rate.

Line rate = R .
 Average packet size = S .
 Average packet arrival time = S/R .

Bandwidth capacity of each module = $2R/n$.
 Time to write packet of size S in module = $(S) / (2R/n)$
 = $Sn/2R$.

Now if we consider S/R as one time slot then,
 Average packet arrival time = 1 time slot.
 Time to write packet of size S in a module = $n/2$ time slots.

This combined with Lemma 1 proves that if we have $n/2$ modules, packets can be buffered in the memory with the selection policy of selecting the memories where reads are not getting performed.

PPB selection policy is to select, out of the $n/2$ modules, the one with the lowest occupancy.

Operation of PPB

PPB read/write operations are performed simultaneously in a pipelined manner over each individual memory modules. If the packet arrival interval is assumed to be of one time slot, each individual read/write operation in the PPB will take at least $n/2$ time slots. Thus during each complete read/write operation, $n/2$ new packets will arrive at the PPB and all of them have to be buffered. We have already proved that the arbiter selection policy ensures that for each new arriving packet, PPB finds an “idle” module and the throughput remains unaffected even if a single write operation takes a longer time.

The complete operation of PPB for n equal to 4 is depicted in figure 3. The read/write operation pipeline length will be 2 ($N/2 =$

$4/2 = 2$). Therefore, a single read/write operation on any memory module lasts for two time slots.

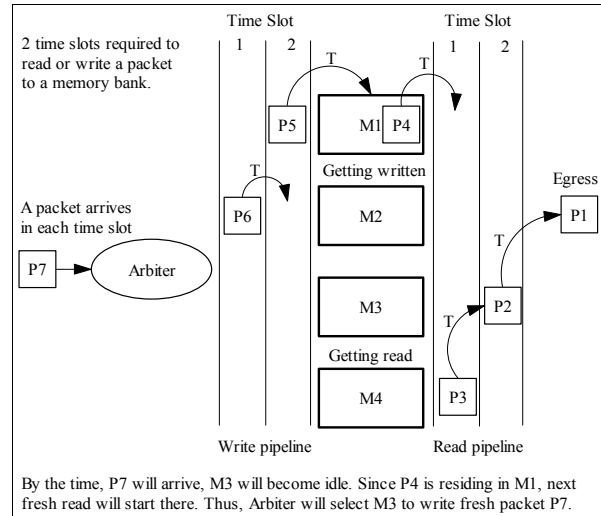


Figure 3 Operation of PPB with $n = 4$

Advantages of PPB

We prove that in any practical application, PPB architecture has the following advantages over any logically single memory based architecture.

1. Scalable enough to support extremely high line rates with the existing memory technology

Traditionally, the bandwidth of the memory buffer is increased by a proportionate increase in its data width. As we reach very high speeds, the data bus keeps on becoming becomes wider proportionally and as soon as we reach the figure of 320, there is a limit. Since most of the IP switches and routers are optimized to process minimum 40-byte sized (320 bit) packets, if data width is increased beyond 40 bytes (320 bits), then for such packets, there is no improvement in the bandwidth and the wastage of memory space will also increase. PPB architecture can achieve a much higher bandwidth than what will be achieved by a 320 bit wide logically single memory by choosing higher values of n and appropriate data-width D . We can have multiple narrower memory modules operating simultaneously. For example, we can choose n to be equal to 8 and data width to be 160 (less than 320), thus getting bandwidth comparable to a logically single memory with data width 1280. Achieving such high bandwidths with any logically single memory is not at all possible.

Thus, it is possible for PPB to achieve extremely high bandwidth capacity irrespective of the memory technology and the bandwidth limit supported by a single memory. It is also independent of the minimum packet size that is to be buffered into PPB.

Theoretically, it seems a logically single memory based architectures with wider data bus width same as $N*D$ (until $N*D$ is less than 320) will give same bandwidth as the PPB architecture. However, this ideology doesn’t hold true for all kind of traffic profiles. Actually for any practical traffic profile, we will prove

that logically single memory based architectures will not give the deterministic and constant bandwidth as compared to the PPB.

2. Higher bandwidth with the same data width

The data-width of any memory buffer is generally chosen to be an integral factor of the average packet size in order to maximize the bandwidth and reduce the buffer wastage. For example, if the device is optimized to process minimum 40 bytes packet, then the data width can be chosen to be 320-bits or $320/i$, where i is a positive integer. For high-speed routers and switches, to achieve very high bandwidth, generally the data width of the memory has to be made very wide. To achieve highest bandwidth, generally i is chosen to be equal to 1 or the data bus width is made the maximum. Thus, one 40-byte packet will be written in just one time slot in the memory and the read/write operation will achieve the fastest possible rate.

However, in any practical scenario, the arriving packet size varies over wide range. If a 41-byte packet will arrive, the packet will take 2 time slots to get written since writing first 40 bytes and next 1 byte will both take 1 time slot each. Thus, in two time slots only 41-bytes data has been written and the write bandwidth has gone down by a factor of $[41 / (40*2)]$ 51%. Now if such odd size packets will keep on coming, the buffer will be able support only 51% of the peak bandwidth capacity. This happens because the granularity at which the memory is accessed is quite large.

PPB eradicates such a situation by making the access in the memory in finer granularity. For example if we choose $n = 8$, then with $D = 320/8 = 40$, the peak bandwidth capacity will match the logically single memory based architecture described above. Now, if 40-bytes packet will come, it will take 8 time slots to get written completely and the bandwidth utilization will be 100%. However, for 41-byte packet, it will take 9 time slots to write the one complete packet. In first 8 time slots, effective bandwidth utilization is 100%, in the last time slot, 1 byte has been written in same time in which ideally 5 bytes (40 bits) could have been written. So the bandwidth utilization in last time slot is $1 / 5 = 20\%$. Therefore, the effective bandwidth utilization for such packet is $(8*100\% + 1*20\%) / (8 + 1) = 91.1\%$. Thus, even in this demanding scenario, in which the potential performance of a logically single memory based system could significantly deteriorate, PPB consistently gives 91% of the ideal bandwidth. And this figure can be extrapolated further by increasing the value of n .

We have devised the following set of equation representing the percentage bandwidth advantage of PPB over an equivalent logically single memory based buffering architecture of same aggregate data-width.

$$A = \left(1 - \frac{1}{L}\right) \times \left(1 - \exp\left(\frac{-k * D}{(D - n)}\right)\right) \times 100\%$$

Where, L = Rate at which the bandwidth of single memory based buffer architecture degrades for a given traffic profile.

D = Aggregate data width of PPB or Single memory.

k = Constant depending on L , D and the traffic profile.

As soon as n becomes equal to D , the PPB achieves the maximum possible ideal bandwidth irrespective of the traffic profile and the packet size. This happens because the granularity of memory

access reaches the minimum limit of 1 bit and for any packet size, no bandwidth will be wasted.

Simulation Results for Bernoulli Traffic

We generated a traffic pattern with sustained data rate (100% load) and the packet size varying as per the Bernoulli distribution function. The value of p is chosen to be 0.75 and mean packet size 40 bytes. But the packets of size less than 40-bytes were filtered out to keep the minimum packet size at 40 bytes. Figure 4 shows the effective bandwidth achieved from the PPB as a function of n with the constraint that the aggregate data-width ($n*D$) is equal to 320. This is made to match the PPB bandwidth with logically single memory based architectures of same data-width of 320.

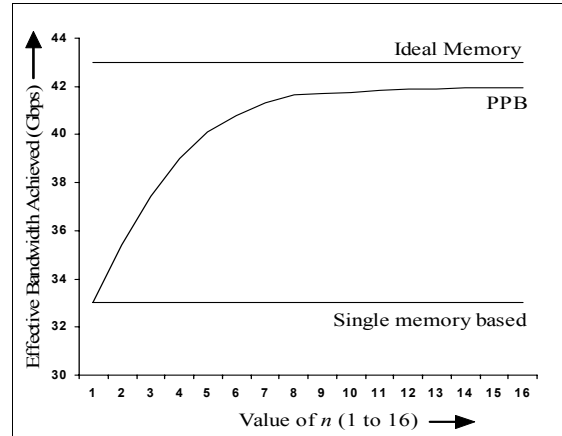


Figure 4 Bandwidth of PPB as a function of n

The graph in figure 4 shows that as the value of n increases, the effective bandwidth of PPB will also increase. As n approaches 320, the data-width of individual memories becomes 1 and effective bandwidth of PPB equals the ideal maximum possible bandwidth irrespective of the varying packet sizes.

3. More available memory (efficient utilization of buffering space):

Now we will show that as n increases, the effective memory utilization also increases. By that we mean that if a logically single memory of same data-width accommodates M packets, PPB will accommodate $(M + \Delta)$ packets, where Δ will depend on n and can be consistently very large for a real traffic pattern.

We will take the same example of logically single memory based buffering architecture as shown in point 2. A 41-byte packet has arrived at such memory buffer with data bus width 320. Thus in first address space we will be storing full 320 bits representing the 100% utilization of the memory space. In second address space, we will be storing 8 bits where ideally 320 bits of data could be stored. Thus in the buffering space of 640 bits, only 328 bits has been stored and the effective memory space utilization is 51% again.

In our architecture with $n = 8$ and $D = 40$, the same 41 byte packet will be stored in 9 locations or address spaces. First 8 locations will be utilized completely while last location will be 20% utilized resulting in the effective buffering space utilization as high as 91%.

Thus, we can say that if the classical memory implementation will accommodate 1 million such packets, PPB will accommodate approximately 1.5 million such packets with the same total size of memory.

We have again generated the same Bernoulli traffic profile with mean data packet size 40 bytes and p to be 0.75. The packets having size lower than 40-bytes has been filtered out. The effective buffering space utilization or the maximum number of packets that could have been buffered will have similar pattern as depicted in figure 4. It can be seen that as the value of n goes up, the effective buffering space utilization also goes up because of finer partitioning of buffers in the memory.

We have also generated constant sized packets with minimum limited to 40 bytes and buffered in a PPB architecture and an equivalent single memory based architecture. n has been chosen to be 8 and D is equal to 40 for PPB while the data-width of logically single memory was made 320. Figure 5 shows the percentage utilization of the memory buffering space as a function of the packet size. It can be seen that as the packet size crosses the integral multiplicity of the data width, the space utilization changes considerably. But this is very less in the PPB again because of finer partitioning of the memory buffers.

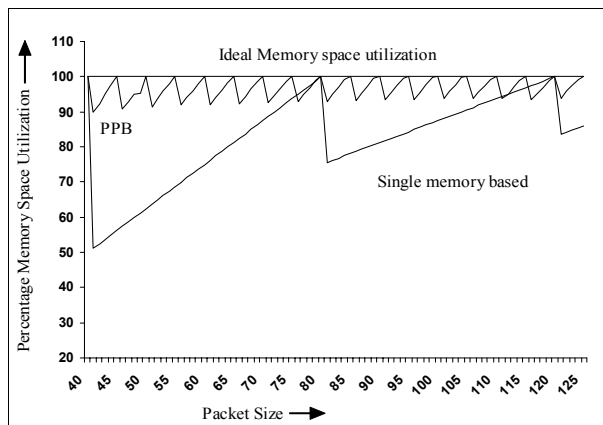


Figure 5 Buffering space utilization as a function of packet size

Thus, it can be seen that any buffering architecture employing wider data bus works at higher buffering granularity that results in non-predictive performance varying considerably as the packet size varies. Also it results in wastage of a lot of available memory space depending on the traffic pattern and packet size variations. In PPB, data width of each of the memory has been reduced by a factor of n while increasing the number of memory modules. Thus, the buffer granularity has been reduced while keeping the effective bandwidth constant. This results in a number of advantages inherent to the lower granularity processing. It can handle a wide range packet sizes without any deterioration in the effective performance for a random traffic with wide variation in packet size.

Disadvantages

Apart from the numerous advantages listed above, PPB comes with one disadvantage, increased overflow rate [13][14][15]. We have

done the complete analysis of the overflow in the PPB for different kind of traffic profiles.

Increased overflow rate

PPB architecture can lead to an early overflow even when there is some space available in one of the memory modules. If the less occupied memory module is being read, then all the incoming packets will be directed to another “idle” modules, which may already be full or having higher occupancy, causing an overflow or increased imbalance in the occupancies of the memories. The situation is described below.

Let the occupancies of the memory modules 1, 2, and n be X_1 , X_2 , and X_N respectively. Then the state of the PPB at any time can be represented by the vector $\{X_1, X_2, \dots, X_N\}$. Due to random read and write sequences, there can be small discrepancies in the value of X_i 's. However, ideally, in the steady state, the values of X_i should remain almost same for all i . Statistically all the X_i 's should tend towards each other owing to the selection policy of the PPB to select the memory module with the least occupancy for writing any new arriving packet. However, if all the memory modules with lesser occupancies are getting read consistently, then the arriving packet will be directed to a memory module, which already has the higher occupancy. This will lead to an increase in already existing discrepancies in the memory occupancies. These increasing discrepancies in the memory occupancies might ultimately lead to the overflow of the PPB even when there is some space left in some of the memory modules. An early overflow state in a PPB with $n=2$ is depicted in figure 6.

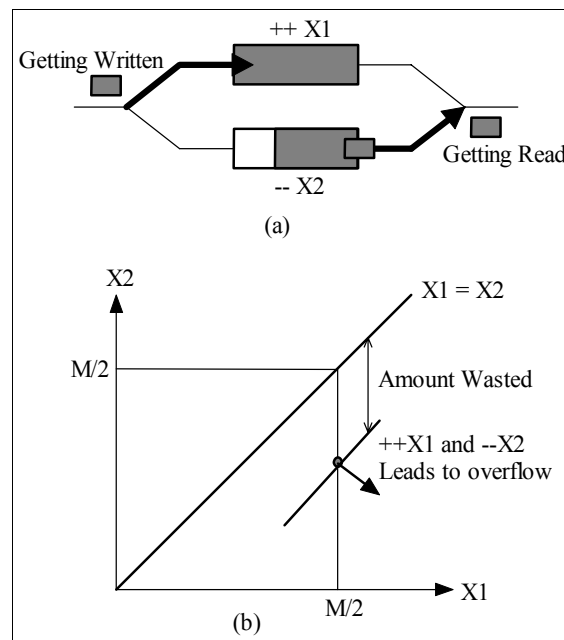


Figure 6 (a) Overflow in PPB with $N=2$ (b) Overflow on occupancy plane

As a worst case in PPB, it is quite possible that one of the memory module overflows while remaining $(n - 1)$ modules remains completely empty. Thus a PPB at worst may perform like an n times faster buffer with n times less buffering space. Fortunately, in practice, such cases are very rare. Even we had difficulty

creating such a situation in our simulations. Our simulation result shows that the amount of memory getting wasted due to increased overflow rate is independent of the total buffer size and the bandwidth. It depends on the traffic profile and is inversely proportional to the value of n and is generally very less.

Estimating the overflow rate

We focus the study of PPB with n equal to 2 and its application to input-queued switches [16][17]. Later on the results can be extrapolated for any value of N . We assume that the PPB memory buffering space is used to emulate m virtual packet queues in a common memory buffer. Q_i represents the i th queue and event of arrival of a packet to this queue is represented by A_i . For the arrival process, we first consider A_i 's distributed as uniform independent identically distributed (i.i.d.) Bernoulli function [19]. The selection policy of scheduler (arbiter) to select one of the virtual queues for reads is made RANDOM. It selects a queue out of the n queues each with uniform probability of selection ($1/n$) independent of the queue size. Finally, we only present simulation results for the case when offered load is high or the virtual queues are written and read very rapidly, utilizing the maximum possible bandwidth. This is because we are interested in the overflow rate and overflows only occur in significant numbers (and therefore only become measurable) when the offered load is high. Also we define the wastage factor (ω) as the ratio of memory buffering space left in the memory when overflow occurs to the total memory size.

$$\omega = \frac{\sum_{i=0}^N \left(\frac{M}{n} - O_i \right)}{M}$$

Where O_i is the occupancy of the i^{th} memory module when the overflow occurs.

The simulation results for Uniform i.i.d. Bernoulli traffic without any burstiness has been shown in figure 7. It appears that the wastage factor ω is inversely proportional to the total PPB size M . However, the total memory wastage in terms of number of buffers remains almost same irrespective of the PPB size (M). In practice, it can be concluded that beyond certain M , the total memory wastage remains almost constant and we have found that it is as low as 10 packets worth space and wastage factor ω keeps on decreasing as the total buffer size increases. Thus by putting few additional buffers in the PPB, the overflow rate will become comparable to that of a logically single memory based architecture.

Next we consider, whether burstiness in the arriving traffic cause higher rate of overflows. Intuitively, it can be said that the traffic arrival rate can lead to burstiness in the traffic departure, if the device is not performing any traffic shaping function. This kind of burstiness can result in higher imbalances in the memory occupancies and hence higher overflow. To see the effect of burstiness on the performance of the PPB, we consider 2-state Markov-modulated Bernoulli arrivals [18]. Often used as a simple model of burstiness, it models a simple on-off process that alternately produces a burst of packets, all with the same destination, followed by an idle period. The length of the bursty and idle periods is geometrically distributed.

We find that, contrary to our expectation, burstiness in the incoming traffic *reduces* the fraction of buffering space wasted by

the PPB architecture. This is shown by Figure 7, which also compares the wastage factors for Markov-modulated Bernoulli arrivals (emulating the bursty traffic), with the uniform i.i.d. random arrivals as described previously.

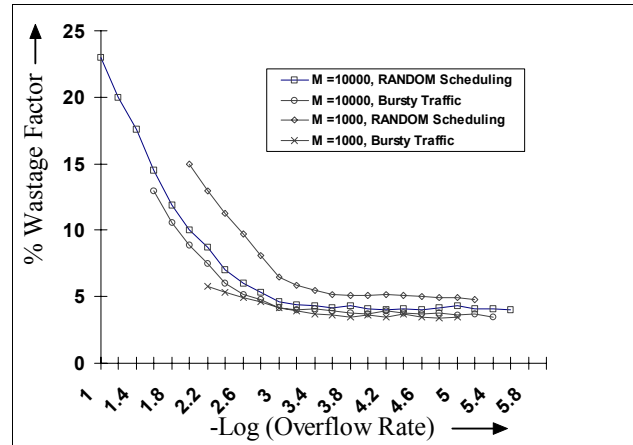


Figure 7 Wastage Factor Vs -Log(Overflow Rate) for i.i.d. Bernoulli traffic

We believe that increased burstiness will decrease the wastage factor, but as yet have been unable to prove this result in general. Instead, we offer an intuitive explanation. First, recall the example in figure 6 that shows how discrepancy (and hence increased wastage factor) can arise in the PPB architecture. The discrepancy was caused by a sequence of constrained writes (while other memory modules were being read out) that forced an arriving stream of packets to be written into only one of the memory module. Later, when this sequence of packets departed (all from one memory), they forced another sequence of arriving packets to be written into the other memory. In other words, when a sequence of arriving packets causes a discrepancy in the occupancy distribution, the same sequence of packets will cause further discrepancy when they depart. We say that in this way, discrepancy *propagates* in a FIFO PPB buffer queue. Now instead, suppose that the packets had not departed in FIFO order; in fact, let's assume that they departed in a random order. This would mean that the first sequence of arriving packets would not force all of the packets in a future sequence to be written into the same memory module. In other words, the random departure order breaks this propagation of discrepancy from one sequence of packets to the next. Now consider what happens when the buffer is arranged as a set of Virtual output queues rather than a single FIFO queue. The scheduling algorithm will cause the packets to depart in non-FIFO order, and so we can expect the wastage factor to be decreased, as shown by our results. Furthermore, we find that if the arrivals are burstier, then their departure order is even further from FIFO kind of arrival and departure. Since the scheduling algorithm is independent of the queue size, it is unlikely to service the same queue twice in a row. This means that packets arriving in a burst are unlikely to depart consecutively, or even close to each other in time. If, on the other hand, the arrivals are not bursty, then successively arriving packets are likely to be destined for different queues. It is therefore quite possible that the scheduler will cause them to depart consecutively, or close to each other in time. In summary, we believe that a FIFO departure order leads to the propagation of discrepancy, and hence increased wastage factor. Independent arrivals lead to departures that are closer to FIFO

order than bursty arrivals. Thus, as the arrivals become burstier, the discrepancy is less likely to propagate. Hence, bursty arrivals lead to a smaller wastage factor.

Variation of wastage factor with offered load

All of the results that we have presented have been for the case when the offered load is high. Although we see that the *wastage factor* is low for a reasonably larger sized memory, it is worth asking whether the *wastage factor* is still small when we reduce the offered load. Indeed, we find that as we reduce the offered load for different types of traffic, the *wastage factor decreases*. For brevity, we do not reproduce lots of results here; a typical result is shown in Figure 7 shows that as the offered load decreases, so does the *wastage factor*. Intuitively, the relationship between offered load and *wastage factor* is simple. Recall that wastage in the PPB architecture is caused by discrepancies in the occupancy level of the memory modules, which in turn is caused by reads and writes in a particular order. When the offered load is high, there are a large number of reads, which constrain the writes of the (plentiful) new packet arrivals. Therefore, the probability that an incoming packet faces a constrained write increases with the offered load. Discrepancy is increased, leading to larger wastage.

Therefore, we can infer that when the offered load is reduced, the *wastage factor* will also decrease. In fact, we found that the *wastage factor* is inversely proportional to the offered load on the PPB.

Estimating the memory discrepancy

Our findings are in agreement with the qualitative argument presented, where we considered the discrepancy of the memory occupancies at the times of overflows. The discrepancy is independent of memory size. However, if the buffer is small, then it constrains the discrepancy to small values. A small increase in memory size significantly reduces the probability that a discrepancy will cause an overflow, but increases the maximum discrepancy. This is illustrated in Figure 9. On the other hand, if the buffer is large, a small increase in memory size barely changes the probability of a discrepancy causing an overflow. And the maximum discrepancy remains almost same.

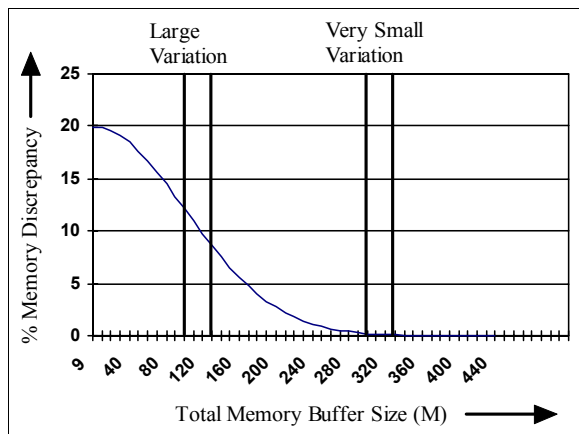


Figure 8 Memory Discrepancy as a function of Total Buffer size (M)

Wastage factor as function of n

Recall that all the simulation results have been shown for a PPB with n equal to 2. It can be proved that as the value of n increases, *wastage factor* decreases rapidly in an inverse relation. This can be hypothetically proved by the following argument:

1. It is known fact that the *wastage factor* is directly proportional to the memory discrepancy.
2. The memory discrepancy in PPB occurs when a write is constrained to happen on a memory with higher occupancy. This happens because the memories with lower occupancy may be getting read while the new packet arrives.
3. Consider that a packet to be read out of PPB has equal probability to exist each of the n memory modules. Thus, the probability that the packet to be read exists in memory module i is $1/n$ for all i . Therefore for each read operation, PPB can select a memory module i with probability $1/n$.
4. Earlier we have proved that with the highest possible offered load on PPB, $n/2$ out of n memory modules will be getting read at any point of time. Therefore for arriving packet, PPB will select one memory module out of the $n/2$ idle memory modules. Now the discrepancy will increase only if all the $n/2$ idle modules have higher occupancy than all the $n/2$ modules that are getting read. From point 3, it can be concluded that probability of such an event is $1/(2 \times n \times (n-1))$. Thus the probability that the discrepancy will increase has an inverse relation with n .

Logically speaking, as the value of n increases, for each write operation PPB arbiter has more number of memory modules ($n/2$) to select from. The PPB arbiter will select the memory modules will the least occupancy, thereby for increasing values of n , PPB arbiter selection policy will start acting and will have a reverse effect on the memory discrepancies (if exists), since it will be able to select the least occupant memory modules out of the lot of $n/2$ modules.

Choosing the appropriate n

The value of n can be chosen appropriately depending on the following parameters:

1. Bandwidth requirements.
2. Traffic profile (mean and variance of packet size).
3. The available memory technology.

As the value of n goes up, the memory bandwidth requirement of a single memory module goes down. Actually each memory is required to support n times lesser bandwidth than the line rate. Thus to achieve very high bandwidth that can't be supported by a logically single memory, there is no option but to use PPB. The value of n can be chosen depending on the buffering bandwidth required and the maximum bandwidth capacity of the memory technology.

It also depends on the traffic profiles and the packet sizes. If there is little variation in the packet sizes, then lower values of n will suffice the purpose. For very high fluctuation in the packet sizes, to provide constant and deterministic access times and better utilization of memory buffering space, higher values of n has to be chosen. Also as the n goes up, the data-width of individual memory modules goes down. Their bandwidth requirement also goes down. Thus cheaper memories with narrower data bus and

lower bandwidth can be used. The appropriate value of n can be judiciously chosen taking into account all the above parameters.

Conclusion

We have discussed the limitations inherent to a classical logically single memory based buffering architecture. Not only does the PPB introduced here eliminate these limitations, it also has its own advantages to offer over other buffering architectures, which make it an exciting choice for packet buffering at very high speeds. These advantages were brought out in the sample simulation using Bernoulli and bursty traffic. Inherent to the PPB is an architectural flaw, described in terms of the wastage factor, which at an initial glance may prove to be a deterring factor in the PPB's usage. However, it has been shown that for real life traffic patterns occurring at high-speed network device nodes, this factor is reduced to an insignificant value, thereby making the PPB's deployment very conducive.

References

- [1] Obara, H., "Optimum architecture for input queueing ATM switches", IEE Electronics Letters, pp. 555 - 557, March 28, 1991.
- [2] Marwan Krunz, Herman Hughes, and Parviz Yegani, "Design and analysis of a buffer management scheme for multimedia traffic with loss and delay priorities," in Proceedings of the IEEE GLOBECOM '94 Conference, pp. 1560-1564, San Francisco, November 1994.
- [3] J. Liebeherr and N. Christin, "JoBS: Joint Buffer Management and Scheduling for Differentiated Services", In Proceedings of IEEE/IFIP International Workshop on Quality of Service (IWQoS '2001). Karlsruhe, Germany, June 2001
- [4] H. Fu and E. Knightly, "Aggregation and Scalable QoS: A Performance Study", In Proceedings of IWQoS '01, Karlsruhe, Germany, June 2001.
- [5] M. Shor, K. Li, and J. Walpole, "Application of Control Theory to Modeling and Analysis of Computer Systems", in Proceedings of Japan-USA-Vietnam Workshop on Research and Education in Systems, 2000.
- [6] Eric M. Manton, "Tomorrow's Internet Broadband Capillaries, Need Terabit Arteries", Cahners IN-STAT Group, August 2001.
- [7] Gideon Kaempfer, "The Challenges of Building a Terabit Router", Charlotte's Web Networks.
- [8] Youngmi Joo, and Nick McKeown, "Doubling Memory Bandwidths for Network Buffers", IEEE Infocom April 1998, Vol 2, pp. 808-815, San Francisco.
- [9] Sundar Iyer, Ramana Rao Kompella, and Nick McKeown, "Analysis of a Memory Architecture for Fast Packet Buffers", IEEE Workshop on High Performance Switching and Routing, Dallas, Texas, May 2001.
- [10] D. K. Hunter, M. C. Chia, and I. Andonovic, "Buffering in optical packet switches," J. Lightwave Technol., vol. 16, no. 12, pp. 328--337, Mar. 1998.
- [11] M. I. Irland, "Buffer Management in a Packet Switch," IEEE Trans. Commun., vol. 26, pp. 328--337, Mar. 1978.
- [12] D.K. Hunter, B. Qiu, K. M. Guild, M.C. Chia, A. C. Bryce, M. H. M. Nizam, Y. Qian, I. Andonovic, J. S. Aitchison, J. Marsh, M. J. O'Mahony: "Buffering Strategies for Optical Packet Switches", OSA Topical Meeting on Photonics in Switching, 21-23 July 1999, Santa Barbara, CA

- [13] Jin Cao and Kavita Ramanan, "A Poisson Limit for Buffer Overflow Probabilities", Proc. INFOCOMM, 2002.
- [14] C. Courcoubetis and R. Weber, "Buffer overflow asymptotics for a buffer handling many traffic sources", *Journal of Applied Probability*, vol. 33, pp. 886-903, 1996.
- [15] Michel Mandjes and Sem Borst, "Overflow behavior in queues with any long-tailed inputs", *Advances in Applied Probability*, vol. 32, pp. 1150-1167, 2001.
- [16] Marco Ajmone Marsan, Andrea Bianco, Paolo Giaccone, Emilio Leonardi, Fabio Neri, "Packet Scheduling in Input-Queued Cell-Based Switches", IEEE INFOCOM'01, Anchorage, Alaska, USA, April 2001.
- [17] N.McKeown, "Scheduling algorithms for input-queued cell switches", Ph.D. Thesis, Un. of California at Berkeley, 1995.
- [18] A. B. DRAGUT, M. Dragut, "Upper Bound Decision Process for a Time-Constrained Radical New Product Development Process (TR-NPD)-A Finite Horizon Discrete-Time Markov Nonstationary Approach", Proceedings of the X-th International Symposium on Applied Stochastic Models and Data Analysis, June 12-15, 2001, Compiègne, France, pp.392-397.
- [19] B.D.Choi, Y.C.Kim, D.I.Choi, D.K.Sung, "An Analysis of M, MMPP/G/1 Queues with QLT Scheduling Policy and Bernoulli Schedule", IEICE Trans. on Commun., Vol.E81-B, pp.1-10, No.1, 1998.