"You mean you want *to drop a packet? Why?".*

— The Customer is Always Right![†]

H

Proofs for Chapter 7

H.1 Proof of Theorem 7.2

Theorem 7.2. *(Necessity) To guarantee that a byte is always available in head cache when requested for any memory management algorithm, the head cache must contain at least* $Qw > Q(b-1)(2+\ln Q)$ *bytes.*

Proof. In what follows we will consider a model where data can be written/read from the packet buffer in a continuous manner, *i.e.*, 1 byte at a time. In reality, this assumption results in a more conservative bound on the cache size than what occurs when discrete packets (which have minimum size limitations) are taken into consideration.

Consider the particular traffic pattern with the pattern of requests shown in Figure H.1. We will show that *regardless of the memory management algorithm* the following pattern is applicable. The pattern progresses in a number of iterations, where iteration number $x = 1, 2, 3, \ldots$ consists of $Q(1-1/b)^x$ time slots. Each successive iteration lasts fewer time slots than the previous one. In each successive iteration the pattern focuses on the queues that have not yet been replenished by the MMA in consideration.

[†]There are cases in which packets delayed due to congestion may be dropped from the buffer. Cisco Systems, San Jose, California, Mar 2006.

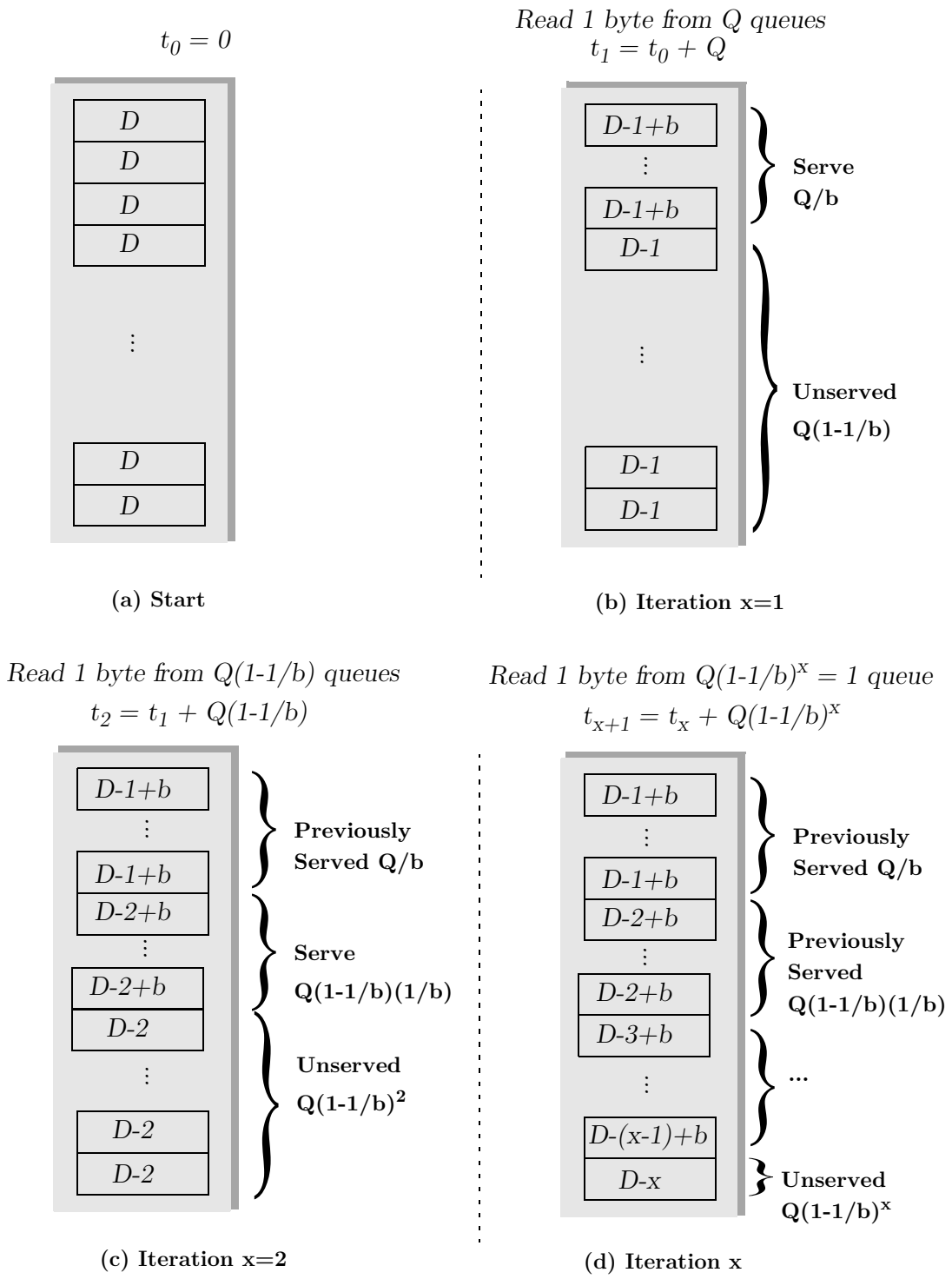

Figure H.1: *Traffic pattern that shows the worst-case queue size of the head SRAM. Starting with completely filled queues with occupancy of D, in every iteration the arbiter requests one byte from the lowest occupancy queues. At the end of iteration x, the last queue has a deficit of $x = \log_{b/(b-1)} Q$.*

Initially at $t_0 = 0$, each queue has D bytes, where D is the minimum number of bytes required so that every byte request can be satisfied by the SRAM cache.

First iteration (Q time slots): In time slots $t = 1, 2, 3, \ldots, Q$, a request arrives for FIFO t. It takes b timeslots to read b bytes from the DRAM and replenish the SRAM cache of a specific FIFO. At the end of time slot Q, at most Q/b FIFOs will have received b bytes from the DRAM, and so at least $Q(1 - 1/b)$ FIFOs will have $D(i, Q) = 1$. Correspondingly, in the figure we can observe that the number of bytes in the first Q/b queues is $D - 1 + b$, while the remaining queues have a deficit of 1.

Second iteration ($Q(1 - 1/b)$ time slots): In the 2^{nd} iteration, consider the $Q(1 - 1/b)$ FIFOs for which $D(i, Q) = 1$. In the next $Q(1 - 1/b)$ time slots, we will assume that a request arrives for each of these FIFOs. At the end of the 2^{nd} iteration, as shown in the Figure, $Q(1 - 1/b)/b$ of these FIFOs will be replenished, and $Q(1 - 1/b)^2$ will have $D(i, t) = 2$.

x^{th} iteration ($Q(1 - 1/b)^x$ time slots):[1] By continuing this argument, we can see that at the end of x iterations there will be $Q(1 - 1/b)^x$ FIFOs with $D(i, t) = x$. Solving for $Q(1 - 1/b)^x = 1$, we get that

$$x = \log_{b/(b-1)} Q = \frac{\ln Q}{\ln c},$$

where $c = b/(b - 1)$. Since, $\ln(1 + x) < x$, we get

$$\ln c = \ln \left[1 + \frac{1}{b-1}\right] < \frac{1}{b-1}$$

and it follows that $x > (b - 1) \ln Q$.

So far, we have proved that if each FIFO can hold $(b - 1) \ln Q$ bytes, then in $(b - 1) \ln Q$ iterations at least one FIFO can have a deficit of at least $(b - 1) \ln Q$ bytes. Imagine that this FIFO is left with an occupancy of $b - 1$ bytes (*i.e.*, it initially held $(b - 1)(1 + \ln Q)$ bytes, and 1 byte was read in each of $(b - 1) \ln Q$ iterations). If in

[1] For example, when discrete packets that have constraints on the minimum size are read, there might be fewer queues that reach their maximum deficit simultaneously, and fewer iterations than the worst case mentioned in this continuous model, where one byte can be read at a time.

successive times slots we proceed to read b more bytes from the most depleted FIFO, *i.e.*, the one with occupancy $b-1$ bytes, it will certainly under-run (because it has never been replenished). Since we do not know *a priori* the queue for which this traffic pattern may occur, we require that $w > (b-1)(1+\ln Q)$ bytes to prevent under-run. But we are not quite done. Imagine that we initially had a head cache large enough to hold precisely $w = (b-1)(1+\ln Q)$ bytes for every FIFO, and assume that the arbiter reads 1 byte from one FIFO then stops indefinitely. After the 1-byte read, the FIFO now contains $(b-1)(1+\ln Q)$ bytes, but is replenished b time slots later with b bytes from DRAM. Now the FIFO needs to have space to hold these additional b bytes. However, since only 1 byte has been read out of the FIFO, it needs to have space for an additional $b-1$ bytes. Therefore, the SRAM needs to be able to hold $w > (b-1)(2+\ln Q)$ bytes per FIFO, and so $Qw > Q(b-1)(2+\ln Q)$ bytes overall.❐

H.2 Proof of Lemma 7.2

Lemma 7.2. *Under the MDQF-MMA, which services requests without any pipeline delay,*

$$F(i) < bi[2+\ln(Q/i)], \forall i \in \{1,2,\ldots,Q-1\}.$$

Proof. The case when $i=1$ is already proved in Lemma 7.1, and when $i=Q$ it is obvious as mentioned in Equation 7.9. For $\forall i \in \{2,\ldots,Q-1\}$, we again solve the recurrence relation obtained in Equation 7.8 to obtain,

$$F(i) \leq i\left[b + b\sum_{j=i}^{Q-1}\frac{1}{j}\right], \forall i \in \{2,\ldots,Q-1\}. \tag{H.1}$$

We can write the summation term in the above equation as,

$$\sum_{j=i}^{Q-1}\frac{1}{j} = \sum_{j=1}^{Q-1}\frac{1}{j} - \sum_{j=1}^{i-1}\frac{1}{j}, \forall i \in \{2,\ldots,Q-1\}. \tag{H.2}$$

Since,

$$\forall N, \ln N < \sum_{i=1}^{N} \frac{1}{i} < 1 + \ln N, \tag{H.3}$$

we can use Equation H.2 and Equation H.3 to re-write Equation H.1 as a weak inequality,

$$F(i) < bi[2 + \ln(Q-1)/(i-1))], \forall i \in \{2, \ldots, Q-1\}.$$

Thus we can write $\forall i \in \{2, \ldots, Q-1\}, F(i) < bi[2 + \ln(Q/(i-1))]$.[2] ❒

H.3 Proof of Lemma 7.3

Lemma 7.3. *(Sufficiency) Under the MDQFP-MMA policy, and a pipeline delay of $x > b$ time slots, the real deficit of any queue i is bounded for all time $t + x$ by*

$$R_x(i, t+x) \leqslant C = b\left(2 + \ln\left(Q\frac{b}{x-2b}\right)\right). \tag{H.4}$$

Proof. We shall derive a bound on the deficit of a queue in the MDQFP-MMA system in two steps using the properties of both MDQF and ECQF MMA. First, we limit (and derive) the maximum number of queues that can cross a certain deficit bound using the property of MDQF. For example, in MDQF, for any k, since the maximum value of the sum of the most deficited k queues is $F(k)$, there are no more than k queues that have a deficit strictly larger than $F(k)/k$ at any given time. We will derive a similar bound for the MDQFP-MMA with a lookahead of x timeslots, $F_x(j)/j$, where $F_x(j)$ is the maximum deficit that j queues can reach under the MDQPF-MMA, and we choose $j = (x-b)/b$. With this bound we will have no more than j queues whose deficit exceeds $F_x(j)/j$ at any given time.

Then we will set the size of the head cache to b bytes more than $F_x(j)/j$. By definition, a queue that has become critical has a deficit greater than the size of the head cache, so the number of unique queues that can become critical is bounded by

[2] Please note that in a previous version of the paper [166], this inequality was incorrectly simplified to $F(i) < bi[2 + \ln(Q/i)]$.

j. This will also lead us to a bound on the maximum number of outstanding critical queue requests, which we will show is no more than j. Since $x \geqslant jb + b$, this gives us sufficient time available to service the queue before it actually misses the head cache. In what follows we will formalize this argument.

Step 1: We are interested in deriving the values of $F_x(i)$ for the MDQFP-MMA. But we cannot derive any useful bounds on $F_x(i)$ for $i < \{1, 2, \ldots, (x-b)/b\}$. This is because MDQFP-MMA at some time t (after taking the lookahead in the next x time slots into consideration) may pick a queue with a smaller deficit if it became critical before the other queue, in the time $(t, t+x)$, ignoring temporarily a queue with a somewhat larger deficit. So we will look to find bounds on $F_x(i)$, for values of $i \geqslant (x-b)/b$. In particular we will look at $F_x(j)$, where $j = (x-b)/b$. First, we will derive a limit on the number of queues whose deficits can cross $F_x(j)/j$ at any given time.

We begin by setting the size of the head cache under this policy to be $F_x(j)/j + b$. This means that a critical queue has reached a deficit of $C > F_x(j)/j + b$, where $j = (x-b)/b$. The reason for this will become clear later. We will first derive the value of $F_x(j)$ using difference equations similar to Lemma 7.1.

Assume that t is the first time slot at which $F_x(i)$ reaches its maximum value, for some i queues. Hence none of these queues were served in the previous time slot, and either (1) some other queue with deficit greater than or equal to $(F_x(i) - b)/i$ was served, or (2) a critical queue was served. In the former case, we have $i+1$ queues for the previous timeslot, for which we can say that,

$$F_x(i+1) \geqslant F_x(i) - b + (F_x(i) - b)/i. \tag{H.5}$$

In the latter case, we have,

$$F_x(i+1) \geqslant F_x(i) - b + C. \tag{H.6}$$

Since $C = F_x(j)/j + b$ and $\forall i \in \{j, j+1, j+2, \ldots, Q-1\}$, $F_x(j) \geqslant F_x(i)/i$, we will use Equation H.5, since it is the weaker inequality.

General Step: Likewise, we can derive relations similar to Equation H.5, *i.e.*, $\forall i \in \{j, j+1, \ldots, Q-1\}$.

$$F_x(i+1) \geqslant F_x(i) - b + (F_x(i) - b)/i \tag{H.7}$$

We also trivially have $F_x(Q) < Qb$. Solving these recurrence equations similarly to Lemma 7.2 gives us for MDQFP-MMA,

$$F_x(i) < bi[2 + \ln(Q/(i-1))], \forall i \in \{j, j+1, \ldots, Q-1\}, \tag{H.8}$$

and $j = (x-b)/b$

Step 2: Now we are ready to show the bound on the cache size. First we give the intuition, and then we will formalize the argument. We know that no more than $j = (x-b)/b$ queues can have a deficit strictly more than $F_x(j)/j$. In particular, since we have set the head cache size to C, no more than j queues have deficit more than $C = F_x(j)/j + b$, *i.e.*, no more than j queues can be simultaneously critical at any given time t. In fact, we will show that there can be no more than j outstanding critical queues at any given time t. Since we have a latency of $x > jb$ timeslots, this gives enough time to service any queue that becomes critical at time t before time $t + x$. The above argument is similar to what ECQF does. In what follows we will formalize this argument.

(*Reductio-Ad-Absurdum*): Let $T + x$ be the first time at which the real deficit of some queue i, $r_x(i, T+x)$ becomes greater than C. From Equation 7.12, we have that queue i was critical at time T, *i.e.*, there was a request that arrived at time T to the tail of the shift register that made queue t critical. We will use the following definition to derive a contradiction if the real deficit becomes greater than C.

ℵDefinition H.1. $r(t)$**:** The number of outstanding critical queue requests at the end of any time slot t.

Consider the evolution of $r(t)$ till time $t = T$. Let time $T - y$ be the closest time in the past for which $r(T-y-1)$ was zero, and is always positive after that. Clearly

there is such a time, since $r(t = 0) = 0$.

Then $r(t)$ has increased (not necessarily monotonically) from $r(T - y - 1) = 0$ at time slot $T - y - 1$ to $r(T)$ at the end of time slot T. Since $r(t) > 0, \forall t \in \{T - y, T\}$, there is always a critical queue in this time interval and MDQFP-MMA will select the earliest critical queue. So $r(t)$ decreases by one in every b time slots in this time interval and the total number of critical queues served in this time interval is $\lfloor y/b \rfloor$. What causes $r(t)$ to increase in this time interval?

In this time interval a queue can become critical one or more times and will contribute to increasing the value of $r(t)$ one or more times. We will consider separately for every queue, the first instance it sent a critical queue request in this time interval, and the successive critical queue requests. We consider the following cases:

Case 1a: *The first instance of a critical queue request for a queue in this time interval, and the deficit of such queue was less than or equal to $F_x(j)/j = C - b$ at time $T - y$.* Such a queue needs to request more than b bytes in this time interval to create its first critical queue request.

Case 1b: *The first instance of a critical queue request for a queue in this time interval, and the deficit of such queue was strictly greater than $F_x(j)/j = C - b$ but less than C at time $T - y$.* Such queues can request less than b bytes in this time interval and become critical. There can be at most j such queues at time $T - y$.[3]

Case 2 *Instances of critical queue requests from queues that have previously become critical in this time interval.* After the first time that a queue has become critical in this time interval (this can happen from either case 1a or case 1b), in order to make it critical again we require b more requests for that queue in the above time interval.

So the maximum number of critical queue requests created from case 1a and case 2 in the time interval $[T - y, T]$ is $\lfloor y/b \rfloor$, which is the same as the number of critical

[3]Note that no queues can have deficit greater than C at the beginning of timeslot $T - y$ because $r(T - y - 1) = 0$.

queues served by MDQFP-MMA. The additional requests come from case 1b, and there can be only j such requests in this time interval. Thus $r(T) \leqslant j$.

Since we know that queue i became critical at time T, and $r(T) \leqslant j$, it gets serviced before time $T + jb < T + x$, contradicting our assumption that the real deficit of the queue at time $T + x$ is more than C. So the size of the head cache is bounded by C. Substituting from Equation H.8,

$$C = F_x(j)/j + b \leqslant b[2 + \ln Qb/(x - 2b)]. \tag{H.9}$$

This completes the proof. ❒

H.4 Proof of Theorem 7.6

H.4.1 Proof of Theorem 7.6 with Three Assumptions

We will first prove Theorem 7.6 with three simplifying assumptions and derive the following lemma.

Assumption 1. (Queues Initially Full) At time $t = 0$, the head cache is full with $b - 1$ bytes in each queue; the cache has $Q(b - 1)$ bytes of data in it.

Assumption 2. (Queues Never Empty) Whenever we decide to refill a queue, it always has b bytes available to be replenished.

Assumption 3. The packet processor issues a new read request every time slot.

Lemma H.1. *If the lookahead buffer has $L_t = Q(b - 1) + 1$ time slots, then there is always at least one critical queue.*

Proof. The proof is by the pigeonhole principle. We will look at the evolution of the head cache. At the beginning, the head cache contains $Q(b - 1)$ bytes (Assumption 1). Because there are always $Q(b - 1) + 1$ read requests (Assumption 2) in the lookahead buffer, at least one queue has more requests than the number of bytes in head cache, and

so must be critical. Every b time slots, b bytes depart from the cache (Assumption 3) and are always refilled by b new bytes (Assumption 2). This means that every b time slots the number of requests is always one more than the number of bytes in head cache, ensuring that there is always one critical queue. ❒

Now we are ready to prove the main theorem with the three assumptions.

Theorem 7.6. *(Sufficiency) If the head cache has $Q(b-1)$ bytes and a lookahead buffer of $Q(b-1)+1$ bytes (and hence a pipeline of $Q(b-1)+1$ slots), then ECQF will make sure that no queue ever under-runs.*

Proof. The proof proceeds in two steps. First, we will prove the theorem with the three assumptions listed above. Then we relax the assumptions to show that the proof holds more generally. The proof for the first step (with the three assumptions) is in two parts. First we show that the head cache never overflows. Second, we show that packets are delivered within $Q(b-1)+1$ time slots from when they are requested.

Part 1 We know from Lemma H.1 that ECQF reads b bytes from the earliest critical queue every b time slots, which means the total occupancy of the head cache does not change, and so never grows larger than $Q(b-1)$.

Part 2 For every request in the lookahead buffer, the requested byte is either present or not present in the head cache. If it is in the head cache, it can be delivered immediately. If it is not in the cache, the queue is critical. Suppose that q' queues have ever become critical before this queue i became critical for byte b_i. Then, the request for byte b_i that makes queue i critical could not have arrived earlier than $(q'+1)b$ time slots from the start. The DRAM would have taken no more than $q'b$ time slots to service all these earlier critical queues, leaving it with just enough time to service queue i, thereby ensuring that the corresponding byte b_i is present in the head cache.

Hence, by the time a request reaches the head of the lookahead buffer, the byte is in the cache, and so the pipeline delay is bounded by the depth of the lookahead buffer: $Q(b-1)+1$ time slots. ❒

H.4.2 Removing the Assumptions from the Proof of Theorem 7.6

We need to make the proofs for Theorem 7.6 and Lemma H.1 hold, without the need for the assumptions made in the previous section. To do this, we make two changes to the proofs – (1) Count "placeholder" bytes (as described below) in our proof, and (2) Analyze the evolution of the head cache every time ECQF makes a decision, rather than once every b time slots.

Removing Assumption 1: To do this, we will assume that at $t = 0$ we fill the head cache with "placeholder" bytes for all queues. We will count all placeholder bytes in our queue occupancy and critical queue calculations. Note that placeholder bytes will be later replaced by real bytes when actual data is received by the writer through the direct-write path as described in Figure 7.3. But this happens independently (oblivious to the head cache) and does not increase queue occupancy or affect the critical queue calculations, since no new bytes are added or deleted when placeholder bytes get replaced.

Removing Assumption 2: To do this, we assume that when ECQF makes a request, if we don't have b bytes available to be replenished (because the replenishment might occur from tail cache from a partially filled queue that has less than b bytes), the remaining bytes are replenished by placeholder bytes, so that we always receive b bytes in the head cache. As noted above, when placeholder bytes get replaced later, it does not increase queue occupancy or affect critical queue calculations.

Removing Assumption 3: In Lemma H.1, we tracked the evolution of the head cache every b time slots. Instead, we now track the evolution of the head cache every time a decision is made by ECQF, *i.e.*, every time bytes are requested in the lookahead buffer. This removes the need for assumption 3 in Lemma H.1. In Theorem 7.6, we replace our argument for byte b_i and queue i as follows: Let queue i become critical when a request for byte b_i occurs. Suppose q' queues

have become critical before that. This means that queue i became critical for byte b_i, no earlier than the time it took for q' ECQF requests and an additional b time slots. The DRAM would take exactly the same time that it took ECQF to issue those replenishment requests (to service all the earlier critical queues), leaving it with at least b time slots to service queue i, thereby ensuring that the corresponding byte b_i is present in the head cache.

So the proofs for Lemma H.1 and Theorem 7.6 hold independent of the need to make any simplifying assumptions.