Packet Scheduling with Window Service Constraints

Chunmei Liu, Eytan Modiano

***Abstract*—This paper considers the scheduling problem that minimizes the average response time of two jobs subject to window constraints. By window constraints, we mean that within a fixed time interval, the server can serve at most W packets from one job, where W is called the window size of the job. This paper investigates the effects of the window constraints in detail, and derives the optimal policy. The results shows that both the job lengths (which correspond to the remaining processing times in traditional optimization problems without window constraints) and the window sizes are essential to the optimal policy. Moreover, instead of changing priority of jobs at different times, in most cases the optimal policy gives full priority to one job. The paper also gives the following suboptimal policy: if both window sizes are greater than one half of the time interval, give full priority to the shorter job; Otherwise, give full priority to the job with the smaller window size. The difference between the optimal cost and the cost of this suboptimal policy is less than the maximum of the two window sizes.**

***Index Terms*—scheduling, window constraints, SRPT**

I. Introduction

Traditionally, the performance of scheduling policies is measured by the mean response time, defined to be the difference between the departure and arrival times of a job. Among all policies, for a work-conserving queue, the Shortest-Remaining-Processing-Time (SRPT) scheduling policy is optimal with respect to minimizing the mean response time [3], [5]. The author in [4] further gives the distribution of the response time for M/G/1 queue under the SRPT policy. Recently a number of papers [1], [6], [2] address the fairness property of the SRPT policy using slowdown (also called stretch), which is defined to be the ratio of the response time and the processing time of a job, as the measure. They show that the SRPT policy not only minimizes the mean response time, but also is good in fairness.

All these previous works assume that upon the arrival of a job, any part of the job is available for service. In practice, jobs are often broken into smaller units before being served, and there may exist a limit on the number of units that can be served within a time interval. For example, in data networks, files are broken into messages and then packets before being released from the transport layer. If the transport layer employs TCP, then the number of packets that can be released to the lower layer is limited by the current window size of TCP, denoted by W. That is, at most W packets can be released within one round trip time. Another example is the transmission of frames at the data link layer, where the number of frames that can be transmitted within one round trip time is limited by the window size of the data link layer protocol.

This paper considers the optimal scheduling policy that minimizes the mean response time when there exists a limit on the number of units to be served within a fixed time interval. For convenience, we call the limit window size and the units packets. For simplicity, we consider the scheduling of packets from two jobs. The effects of window constraints are presented in detail, and an optimal policy is developed. Based on this optimal policy, a suboptimal policy which gives further insights on schedule is also derived. Rather than the remaining processing time, our optimal policy takes into account both the job lengths and the window constraints.

The paper is organized as follows: in the next section we provide the problem formulation and discuss the effects of window constraints. In Section III we describe the optimal policy and the suboptimal policy. In Section IV we conclude the paper.

Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, email: mayliu@mit.edu, modiano@mit.edu. This work was supported by NASA Space Communication Project grant number NAG3-2835.

II. Problem Description and Effects of Window Constraints

In this section we first describe the system considered, then explore the window constraints.

A. Problem Description

Consider a system with two jobs to be served by one server. The two jobs are broken into packets before being served. They have L_1 and L_2 packets, respectively. All packets have the same length. The server performs packets-based service. That is, a new packet cannot preempt the packet being served, but after the service is complete, the server can process packets from either job. The processing time of one packet is defined to be one time slot. In this way the system becomes a slotted system, and henceforth all time is measured in time slots.

For convenience, starting from time 0, we index the time slots in order. That is, the kth slot is called slot k. Similarly, we index the packets from job i, $i = 1, 2$, in order, too. That is, the pth packet served from job i is called packet p of job i.

The service is under window constraints. By under window constraints, we mean that within a fixed time interval τ slots, the server can serve at most W_1 packets from job 1 and W_2 packets from job 2 , where W_i, $i = 1, 2$, is called the window size of job i. The interval τ corresponds to the round trip time in data networks, and the window size W_i corresponds to the maximum number of outstanding packets allowed in data networks.

For an arbitrary scheduling policy π, let $t_i(\pi)$, $i = 1, 2$, be the time slot that job i is finished and define cost $c(\pi) = t_1(\pi) + t_2(\pi)$. We want to find a scheduling policy that minimizes $c(\pi)$, that is, minimizes the average response time of the two jobs. For brevity, later on when there is no ambiguity, we omit π and write c, t_1 and t_2 directly.

B. Equivalent Constraints

From the above problem description we can see that if there were no window constraints, the problem would be the traditional scheduling problem that minimizes the average response

time, and the SRPT policy is the optimal solution. The essence of the window constraints is to limit the availability of packets for service. One equivalent constraint to the window constraints is on the availability of an arbitrary packet for service at a time slot, and we call it packet constraint. Specifically, at time 0, there are W_1 packets from job 1 (packet 1 to W_1) and W_2 packets from job 2 (packet 1 to W_2) that are available for service. For any other packet $p \geq W_i+1$ from job i, $i = 1, 2$, the packet constraint says that packet p is available for service at time slot z if and only if packet $p - W_i$ was served before and including slot $z - \tau$. This packet constraint is helpful in drawing the service pattern under a scheduling policy with the window constraints.

Another equivalent constraint is on the difference between the slots when packet p and packet $p - W_i$ receive service, and we call it service constraint. This constraint is useful in deriving the optimal policy. Let $u_i(p)$ denote the time slot that packet p from job i is served. Then the service constraint requires

$$u_i(p) - u_i(p - W_i) \geq \tau \text{ for all } p \geq W_i + 1. \quad (1)$$

That is, for packets that are W_i packets apart, the slots when they receive service must be at least τ apart. The equality holds if and only if packet p is served at the time slot when it becomes available. Notice that since each packet takes one time slot for processing, $u_i(p) - u_i(p - W_i) \geq W_i$ holds for all possible policies. Also notice that by definition, $t_i = u_i(L_i)$.

C. Effects of Window Constraints

In this subsection we illustrate how the window constraints affect the service pattern. Start from the simple case when there is only one job, say job i, waiting for service, and consider the work-conserving policy. By work-conserving, we mean that no work is created or destroyed in the system, therefore the server cannot be idle if there are packets available for service. If there were no window constraints, then the server would continue serving packets until the job is finished. When there exists window constraint, by repeatedly using the packet constraint given in the previous subsection, it is straightforward to obtain the service pattern under the work-conserving policy, as shown in Figure 1 (a) and (b) for case $W_i \geq \tau$ and case $W_i < \tau$, respectively. The gray blocks in the figure represent that the server is processing packets, and the letter inside each block represents which job the packets in service are from. The formula above the blocks are the number of packets in the packet-blocks or the lengths of the idle-blocks in slots, and the formula below are the block lengths in slots. Later figures showing service patterns can be explained similarly.

Figure 1 (a) shows that when $W_i \geq \tau$, the service pattern is the same as that without window constraints. That is, the server continues serving packets until the job is finished. Actually, since the server can serve at most τ packets during any τ interval, which cannot be greater than the window size W_i in this case, the window constraints in practice take no effect. Another interpretation is that in this case, $u_i(p) - u_i(p - W_i) \geq W_i \geq \tau$ for all $p \geq W_i + 1$. That is, the service constraint (1) is satisfied automatically. Therefore, when $W_i \geq \tau$, there are always packets available for service, and the service pattern is the same as that without window constraints. Whereas when $W_i < \tau$, the service constraint is not satisfied automatically, and not all time

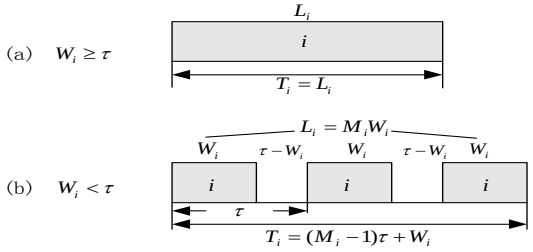

Fig. 1. Service pattern for one job under work-conserving policy and window constraints

slots have packets available for service. The service pattern thus consists of packet-blocks and idle-blocks. Figure 1 (b) shows that their sizes are W_i and $\tau - W_i$, respectively.

Moreover, for the future use when deriving the optimal policy, define function $f_i(l)$ to be the shortest time needed to finish l packets from job i when there is no packets from other jobs. When $W_i \geq \tau$, it is obvious that $f_i(l) = l$. When $W_i < \tau$, further define m and n to be the integers that satisfy $l = mW_i + n$ and $n \in [1, W_i]$. $f_i(l)$ can be obtained by considering the slots when packet $l - kW_i$ receives service for $k = 0, .., m$. Specifically, by the service constraint (1), $u_i(l) - u_i(n) = \sum_{k=0}^{m-1} [u_i(l - kW_i) - u_i(l - (k+1)W_i)] \geq m\tau$. Recall that packet 1 to W_i are available for service at time 0. Therefore $u_i(n) \geq n$ and consequently, $u_i(l) \geq m\tau + n$. Moreover, it is easy to verify that $u_i(l) = m\tau + n$ can be achieved by the work-conserving policy. We thus have that the shortest time $f_i(l) = m\tau + n$ for $W_i < \tau$. Overall,

$$f_i(l) = \begin{cases} l & W_i \geq \tau \\ m\tau + n & W_i < \tau \end{cases} \quad (2)$$

Figure 2 plots function $f_i(l)$ for different window size W_i.

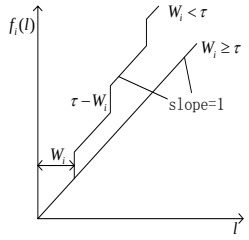

Fig. 2. Time needed for serving l packets from job i

Further define T_i to be the shortest time needed to finish job i when there is no other jobs, and call it shortest processing time of job i subject to window constraint. For simplicity, assume $L_i = M_i W_i$ for some integer M_i when $W_i < \tau$. Then by definition and Equation (2),

$$T_i = f_i(L_i) = \begin{cases} L_i & \text{when } W_i \geq \tau \\ (M_i - 1)\tau + W_i & \text{when } W_i < \tau \end{cases} \quad (3)$$

This is consistent with the T_i shown in Figure 1. Note that for a job with length L_i, a server placing window constraints W_i

and τ, T_i is a constant and independent of the scheduling policy employed.

Now consider the simple case when there are two jobs waiting for service and the server employs the work conserving policy that gives full priority to job 1. By giving full priority to job 1, we mean that whenever packets from job 1 are available for service, the server serves packets from job 1. Under this scheduling policy, the service pattern for job 1 is unaffected and thus the same as that when there is only job 1 waiting for service (c.f. Figure 1).

We classify different cases by whether $W_1 \geq \tau$, $W_1+W_2 > \tau$ and/or $L_1 + L_2 > T_1$. The reasons are explained below. The service patterns under different cases can be obtained straightforwardly by using the packet constraint, and the results are as shown in Figure 3 to Figure 6.

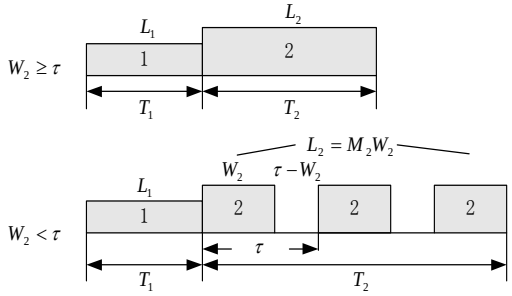

Fig. 3. Service pattern for two jobs under work-conserving policy that gives full priority to job 1, $W_1 \geq \tau$

Specifically, Figure 3 plots the case when $W_1 \geq \tau$. As mentioned before, in this case there are always packets from job 1 available for service and the server continues serving job 1 until it is finished. Afterwards the server begins to serve job 2 in the absence of job 1. Therefore the service pattern can be divided into two parts: the first part is the service pattern for job 1 in the absence of job 2 (takes T_1 slots), and the second part is the service pattern for job 2 in the absence of job 1 (takes T_2 slots).

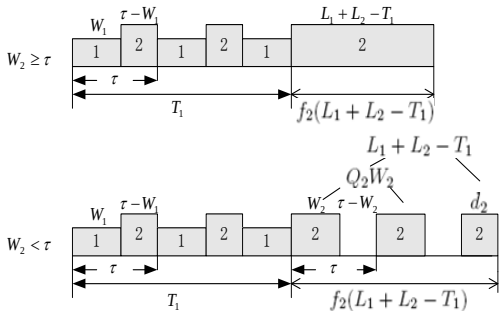

Fig. 4. Service pattern for two jobs under work-conserving policy that gives full priority to job 1, $W_1 < \tau$, $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_1$

Figure 4 shows the service pattern when $W_1 < \tau$, $W_1+W_2 > \tau$ and $L_1 + L_2 > T_1$. $W_1 < \tau$ means that packets from job 1 are not always available for service, and there are gaps between packet-blocks of job 1. Moreover, during any time interval with length τ, the server can serve at most τ packets. But according to the window constraints, there can be as many as $W_1 + W_2 > \tau$ packets available for service before either job is finished. Therefore, at any time slot before either job is finished, there exists at least one packet, from either job 1 or job 2, available for service. Consequently, a work-conserving policy has no idle slot before either job is finished, and hence packets from job 2 fill up the gaps between the packet-blocks of job 1, as shown in Figure 4.

Furthermore, since each packet takes one time slot, $T_1 - L_1$ packets are needed to fill up all the gaps between packet-blocks of job 1. Therefore, inequality $L_1 + L_2 > T_1$ means that there are enough packets from job 2 to fill all the gaps between packet-blocks of job 1, and at the time job 1 is finished (slot T_1), there are $L_1 + L_2 - T_1$ packets left from job 2. The service pattern in Figure 4 can thus be summarized as follows: the pattern for job 1 is the same as that in Figure 1 (without job 2). Before job 1 is finished, packets from job 2 fill up all the gaps between packet-blocks of job 1, and there are no idle slots. After job 1 is finished, the service pattern follows the pattern of job 2 for its remaining packets.

For convenience, for this case, define Q_2 and d_2 to be the integers that satisfy

$$L_1 + L_2 - T_1 = Q_2 W_2 + d_2, \tag{4}$$

and $d_2 \in [1, W_2]$. The physical meaning of Q_2 and d_2 are shown in Figure 4. By symmetry, when $W_2 < \tau$, $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_2$, define Q_1 and d_1 to be the integers that satisfy $d_1 \in [1, W_1]$ and

$$L_1 + L_2 - T_2 = Q_1 W_1 + d_1. \tag{5}$$

These parameters will be used later when we describe the optimal policy.

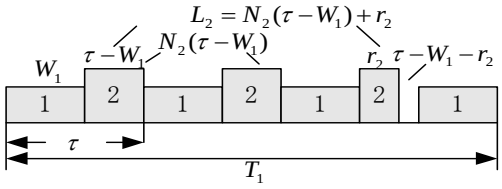

Fig. 5. Service pattern for two jobs under work-conserving policy that gives full priority to job 1, $W_1 < \tau$, $W_1 + W_2 > \tau$ and $L_1 + L_2 \leq T_1$

Figure 5 plots the service pattern when $W_1 < \tau$, $W_1+W_2 > \tau$ but $L_1 + L_2 \leq T_1$. Similar to the previous case, packets from job 2 fill up the gaps between packet-blocks of job 1. Differently, in this case $L_1 + L_2 \leq T_1$. This means that there are no enough packets from job 2 to fill up all the gaps ($T_1 - L_1$ packets needed to fill up all the gaps), and job 2 is finished before job 1. For convenience, define N_2 and r_2 to be the integers that satisfy

$$L_2 = N_2(\tau - W_1) + r_2, \tag{6}$$

and $r_2 \in [1, \tau - W_1]$. Their physical meanings are shown in Figure 5. By symmetry, when $W_2 < \tau$, $W_1 + W_2 > \tau$ but $L_1 + L_2 \leq T_2$, define N_1 and r_1 to be the integers that satisfy $r_1 \in [1, \tau - W_2]$ and

$$L_1 = N_1(\tau - W_2) + r_1. \tag{7}$$

These parameters will be used later when we describe the optimal policy.

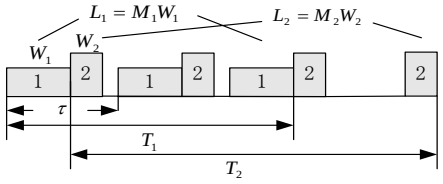

Fig. 6. Service pattern for two jobs under work-conserving policy that gives full priority to job 1, $W_1 + W_2 \leq \tau$

Finally, Figure 6 plots the service pattern when $W_1+W_2 \leq \tau$. In this case, during each τ time slots, there are at most $W_1 + W_2$ packets served and the rest $\tau - (W_1 + W_2)$ slots are idle.

From the above analysis we see that under the work-conserving scheduling policy that gives full priority to job 1, whether $W_1 \geq \tau$ determines whether the server continues serving job 1 until it is finished, or there are gaps between packet-blocks of job 1. Whether $W_1 + W_2 > \tau$ determines whether there are idle slots before either job is finished. Moreover, when $W_1 < \tau$ and $W_1 + W_2 > \tau$, whether $L_1 + L_2 > T_1$ determines whether job 1 is finished first. The service pattern for other policies under different window sizes and job lengths can be obtained in a similar way.

The above analysis and figures also show that due to the window constraints, the service pattern for jobs are quite different from traditional problems without window constraints. As will be shown later, the optimal policy is quite different as well.

III. Optimal Policy and Suboptimal Policy

In this section we derive the optimal policy for the problem described and the more insightful suboptimal policy. The difference between the costs of the optimal policy and this suboptimal policy will be given as well. When deriving the optimal policy, we classify the problem into two cases:

Case 1: $W_1 + W_2 \leq \tau$;
Case 2: $W_1 + W_2 > \tau$.

In the case when $W_i < \tau$ and $M_i = 1$ ($L_i = W_i$), the window constraints take no effects. Since we are concerning only the window constraints, henceforth when $W_i < \tau$, we only consider the cases with $M_i \geq 2$.

A. Optimal Policy When $W_1 + W_2 \leq \tau$

Theorem 1: If $W_1 + W_2 \leq \tau$, the policy π^* that gives full priority to the job with smaller window size is optimal.

Proof: Since $W_i > 0$ for $i = 1, 2$, the inequality $W_1 + W_2 \leq \tau$ means that $W_i < \tau$ for both $i = 1$ and 2. From the service constraint (1), we have $u_i(kW_i) - u_i((k-1)W_i) \geq \tau$ for $k \in [2, M_i]$, where as defined before, M_i is the integer that satisfies $L_i = M_iW_i$. Equality holds if and only if packet kW_i is served at the time slot when it becomes available. By summing this inequality over k, we obtain

$$\begin{aligned}(M_i - 1)\tau &\leq \sum_{k=2}^{M_i}[u_i(kW_i) - u_i((k-1)W_i)] \\ &= u_i(M_iW_i) - u_i(W_i) \\ &= t_i - u_i(W_i),\end{aligned}$$

where the last equality comes from the definitions of t_i and u_i. We therefore have

$$c = t_1+t_2 \geq (M_1-1)\tau+(M_2-1)\tau+u_1(W_1)+u_2(W_2). \quad (8)$$

Recall that packet 1 to W_1 from job 1 and packet 1 to W_2 from job 2 are available for service from time 0. Hence the problem of minimizing $u_1(W_1)+u_2(W_2)$, called the suboptimal problem, is the traditional optimization problem with no window constraints and the SRPT policy that gives priority to the job with shorter W_i is optimal. Without loss of generality, assume $W_1 \leq W_2$. Then by applying the result of the SRPT policy, the resulting optimal sum of the suboptimal problem is $2W_1 + W_2$, which is a lower bound for $u_1(W_1)+u_2(W_2)$. After plugging it into the inequality (8), we obtain

$$\begin{aligned}c &\geq (M_1 - 1)\tau + (M_2 - 1)\tau + 2W_1 + W_2 \\ &= T_1 + T_2 + W_1,\end{aligned}$$

where the equality follows from Equation (3). The above inequality gives us a lower bound for c for any policy that satisfy the window constraints.

Now consider the policy that gives full priority to the job with the shorter window size, which is job 1 under our assumption. From Figure 6 we can easily obtain that under this scheduling policy, $c = t_1 + t_2 = T_1 + T_2 + W_1$, which achieves the lower bound of c. Therefore, the policy that gives full priority to the job with the shorter window size is optimal.

■

From the proof we see that the lower bound $T_1 + T_2 + W_1$ can be achieved if and only if the equality in (8) is achieved and $u_1(W_1) + u_2(W_2)$ achieves its lower bound. It can be easily verified that our optimal policy meets both of the two requirements.

B. Optimal Policy When $W_1 + W_2 > \tau$

Now let's consider the second case, when $W_1 + W_2 > \tau$. We first give five lemmas that characterize the optimal policy, then based on these lemmas, we develop the optimal policy.

Lemma 1: If $W_1 + W_2 > \tau$, the optimal policy has no idle slots before either job is finished.

Proof: Suppose policy π is an optimal policy and has at least one idle slot before either job is finished. Since $W_1+W_2 > \tau$, at this idle slot there is at least one packet from job 1 or job 2 available for service. Without loss of generality, assume the packet available is packet p from job 1. By doing the following, we obtain policy π': keep the position of each packet from job 2. Serve packet p from job 1 at this idle slot, and for all $k \in [p + 1, L_1]$, serve packet k at $u_1(k - 1)$, where $u_1(k - 1)$ is defined for policy π. That is, move all packets after packet p from job 1 one packet backward. Then for the new policy π', $t_1(\pi') =$

$u_1(L_1-1) < u_1(L_1) = t_1(\pi)$ and $t_2(\pi') = t_2(\pi)$, which results in $c(\pi') = t_1(\pi') + t_2(\pi') < t_1(\pi) + t_2(\pi) = c(\pi)$. That is, policy π cannot be optimal, which leads to a contradiction. ■

Define policy set $\Pi_i = \{$policy $\pi|$ job i is finished first and there is no idle slot before job i is finished$\}$, $i = 1, 2$, and optimal policy set $\Pi^* = \{$policy that has the lowest cost over all policies$\}$. The above lemma shows that $\Pi^* \subseteq \Pi_1 \cup \Pi_2$.

Lemma 2: If $W_1 + W_2 > \tau$, then $L_1 + L_2 > \min\{T_1, T_2\}$. Furthermore, if $L_1 + L_2 \leq T_1$, then $\Pi_1 = \varnothing$ and $\Pi^* \subseteq \Pi_2$; if $L_1 + L_2 \leq T_2$, then $\Pi_2 = \varnothing$ and $\Pi^* \subseteq \Pi_1$.

Proof: First show that $L_1 + L_2 > \min\{T_1, T_2\}$. If at least one of W_1 and W_2 is greater than or equal to τ, say $W_i \geq \tau$, for $i = 1$ and/or 2, then $T_i = L_i$ (Equation (3)). Thus $L_1 + L_2 > T_i \geq \min\{T_1, T_2\}$. If $W_1 < \tau$ and $W_2 < \tau$, we show $L_1 + L_2 > \min\{T_1, T_2\}$ by contradiction, as follows.

Suppose $L_1+L_2 \leq T_1$ and $L_1+L_2 \leq T_2$. If $M_1-1 \leq M_2$, the first inequality $L_1+L_2 \leq T_1$ gives us $M_2W_2 = L_2 \leq T_1-L_1 = (M_1-1)(\tau-W_1) \leq M_2(\tau-W_1)$. This gives $W_1+W_2 \leq \tau$. Contradiction.

On the other hand, if $M_1 - 1 > M_2$, the second inequality $L_1 + L_2 \leq T_2$ gives us $M_1W_1 = L_1 \leq T_2 - L_2 = (M_2 - 1)(\tau - W_2) < M_1(\tau - W_2)$. Again this gives $W_1 + W_2 < \tau$. Contradiction.

Therefore when $W_1+W_2 > \tau$, the two inequalities $L_1+L_2 \leq T_1$ and $L_1 + L_2 \leq T_2$ cannot hold together. Hence, $L_1 + L_2 > \min\{T_1, T_2\}$.

Now consider the rest of the lemma. Since T_1 is the shortest processing time of job 1 subject to window constraint, for any policy π, the time when job 1 is finished $t_1 \geq T_1$. If $L_1 + L_2 \leq T_1$, then all policies that finish job 1 first have at least one idle slot among the first T_1 slots, thus among the first t_1 slots, that is, before job 1 is finished. Therefore by the definition of Π_1 and Lemma 1, $\Pi_1 = \varnothing$ and $\Pi^* \subseteq \Pi_2$.

Similarly, when $L_1 + L_2 \leq T_2$, $\Pi_2 = \varnothing$ and $\pi^* \subseteq \Pi_1$. ■

The next three lemmas characterize policies in Π_2 which leads to optimal policies. Here we consider set Π_2 instead of Π_1 in order to use Figure 1 to 6 as illustration.

For an arbitrary policy $\pi_2 \in \Pi_2$, let $v_1(\pi_2)$ denote the number of packets from job 1 that were served before job 2 is finished. For brevity, later on when there is no ambiguity, we omit π_2 and use v_1 directly.

Lemma 3: When $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_2$, for any policy $\pi_2 \in \Pi_2$, we have $v_1(\pi_2) \in [v_1^L, v_1^U]$, where

$$v_1^L = \begin{cases} 0 & \text{when } W_2 \geq \tau \\ T_2 - L_2 & \text{when } W_2 < \tau \end{cases} \tag{9}$$

$$v_1^U = \begin{cases} L_1 - 1 & \text{when } L_1 + L_2 > T_1 \\ (N_2+1)W_1 & \text{when } L_1 + L_2 \leq T_1 \end{cases} \tag{10}$$

Proof: First consider the lower bounds. A trivial lower bound is 0. For the case when $W_2 < \tau$, the tighter lower bound given in the lemma can be shown by contradiction as follows. Suppose $v_1 < T_2 - L_2$. Then the total number of packets served before job 2 is finished is $v_1 + L_2 < T_2 \leq t_2$, where the last equality and the last inequality come from the definition of T_2 (see Equation (3)). Therefore, there must be idle slots before job 2 is finished, which contradicts to the definition of Π_2.

Next consider the upper bound. A trivial upper bound is $L_1 - 1$, otherwise job 1 would be finished first. For the case when $L_1 + L_2 \leq T_1$, we show $v_1 \leq (N_2 + 1)W_1$ by contradiction. Suppose $v_1 > (N_2 + 1)W_1$ and let slot s denote the slot where the first v_1 packets from job 1 are finished. Since for every τ slots, the server can serve at most W_1 packets from job 1, before slot s there are at least $(N_2 + 1)(\tau - W_1)$ slots where the server is not serving packets from job 1. In addition, by the definition v_1, at least one packet from job 2 is served after slot s. That is, before slot s there are at most $L_2 - 1$ packets from job 2 that were served. But from Equation 6, $L_2 - 1 = N_2(\tau - W_1) + r_2 - 1 \leq (N_2 + 1)(\tau - W_1) - 1$. Therefore, before slot s, thus before t_2, there are at least one idle slot, which contradicts to the definition of Π_2. ■

Intuitively, v_1^L corresponds to v_1 under the policy that gives full priority to job 2, and v_1^U corresponds to v_1 under the policy that gives full priority to job 1. Then Equation (9) and (10) can be easily obtained from Figure 1 to 5.

Lemma 4: When $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_2$, for all policies in Π_2 with $v_1 = v \in [v_1^L, v_1^U]$, the finish time t_1 and t_2 and the cost c are lower bounded by

$$t_2^L(v) = L_2 + v, \tag{11}$$

$$t_1^L(v) = t_2^L(v) + f_1(L_1 - v) + b, \tag{12}$$

$$c^L(v) = 2L_2 + 2v + f_1(L_1 - v) + b, \tag{13}$$

where $b = \tau - W_1 - r_2$ if $L_1 + L_2 \leq T_1$ and $v = v_1^U = (N_2 + 1)W_1$, and $b = 0$ otherwise. Moreover, there are at least one policy in Π_2 that can achieve these bounds.

Proof: Here we only prove the lower bounds. One policy that achieves both bounds can be constructed. Since the construction is complicated and skipping it does not affect the following part of the paper, for brevity, we omit the details.

The lower bound $t_2^L(v)$ comes directly from the definition of v_1. Now consider $t_1^L(v)$. By the definition of v_1, there are $L_1 - v$ packets from job 1 left after job 2 is finished, which need at least $f_1(L_1 - v)$ slots to be served. Therefore, $t_1 \geq t_2 + f_1(L_1 - v) \geq t_2^L(v) + f_1(L_1 - v)$. This is one lower bound of t_1. A tighter lower bound can be found when $L_1 + L_2 \leq T_1$ and $v = v_1^U = (N_2 + 1)W_1$, as follows. By the definition of T_1, we have $t_1 \geq T_1$. When $L_1 + L_2 \leq T_1$ and $v = v_1^U = (N_2 + 1)W_1$, by plugging $v = (N_2+1)W_1$ into the equations and after some manipulations, we obtain $T_1 = t_2^L(v) + f_1(L_1 - v) + (\tau - W_1 + r_2)$. We hence have $t_1 \geq t_2^L(v) + f_1(L_1 - v) + (\tau - W_1 + r_2)$ in this case. Combining the above analysis leads to the lower bound $t_1^L(v)$ given in the lemma.

The lower bound $c^L(v)$ for the cost follows from $c = t_1 + t_2 \geq t_1^L(v) + t_2^L(v) = c^L(v)$. ■

Over the policy set Π_2, for a fixed $v_1 = v$, Lemma 4 gives the lowest cost. Lemma 3 further provides upper and lower bounds on v_1. We can thus optimize the lowest cost in Lemma 4 over all possible v_1 and obtain the optimal cost over Π_2. Specifically, for an arbitrary policy π_2 in Π_2, we have $c(\pi_2) \geq c^L(v_1(\pi_2)) \geq \min_{v \in [v_1^L, v_1^U]} c^L(v)$. Let c_2^* denote the minimum and v_1^* denote the v that achieves the minimum. The next lemma develops this idea and gives v_1^* over Π_2.

Lemma 5: When $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_2$, the lowest cost over Π_2, c_2^*, is achieved at the following v_1^*:

1) when $W_1 \geq \tau$: $v_1^* = v_1^L$;
2) when $\tau/2 < W_1 < \tau$ and $W_2 \geq \tau$: $v_1^* = v_1^L = 0$;
3) when $\tau/2 < W_1 < \tau$ and $W_2 < \tau$:

a) if $Q_1 = 0$: $v_1^* = v_1^L = L_1 - d_1$;
b) if $Q_1 \neq 0$ and $d_1 \geq \tau - W_1$: $v_1^* = v_1^L = T_2 - L_2$;
c) if $Q_1 \neq 0$ and $d_1 < \tau - W_1$: $v_1^* = v_1^L + d_1 = (M_1 - Q_1)W_1$;

4) when $W_1 \leq \tau/2$ and $L_1 + L_2 > T_1$:
a) if $W_2 < \tau$ and $Q_1 = 0$: $v_1^* = v_1^L = L_1 - d_1$;
b) if $W_2 \geq \tau$ or $Q_1 \neq 0$: $v_1^* = (M_1 - 1)W_1$;

5) when $W_1 \leq \tau/2$ and $L_1 + L_2 \leq T_1$:
a) if $r_2 \geq W_1$: $v_1^* = v_1^U = (N_2 + 1)W_1$;
b) if $r_2 < W_1$: $v_1^* = v_1^U - W_1 = N_2 W_1$.

Proof: First consider part 1) when $W_1 \geq \tau$. In this case $f_1(l) = l$ and $L_1 + L_2 > T_1$. Hence from Lemma 4 we have: $c_2^L(v) = 2L_2 + 2v + L_1 - v = L_1 + 2L_2 + v$. Therefore, $c_2^L(v)$ increases monotonically with v and consequently, $v_1^* = v_1^L$. Part 1) thus holds.

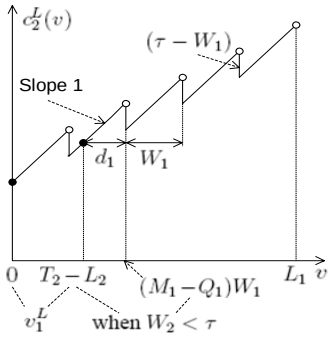

Fig. 7. $c_2^L(v)$ against v when $\tau/2 < W_1 < \tau$ (when $W_2 \geq \tau$, $v_1^L = 0$ and when $W_2 < \tau$, $v_1^L = T_2 - L_2$)

Next consider part 2) and 3) when $\tau/2 < W_1 < \tau$. From Lemma 4, Figure 7 plots $c_2^L(v)$ against v for this case. This figure directly gives v_1^* under different cases in part 2) and 3) of the lemma. Here for brevity, we omit the details.

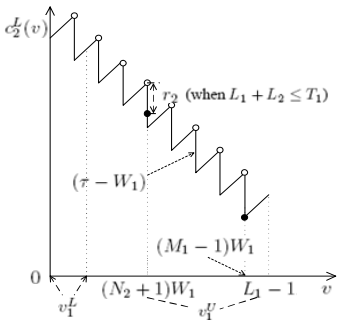

Fig. 8. $c_2^L(v)$ against v when $W_1 \leq \tau/2$ (when $L_1 + L_2 > T_1$, $v_1^U = L_1 - 1$ and when $L_1 + L_2 \leq T_1$, $v_1^U = (N_2 + 1)W_1$)

Now consider part 4) and 5) when $W_1 \leq \tau/2$. From Lemma 4, Figure 8 plots $c_2^L(v)$ against v for this case. The figure shows that when $L_1 + L_2 > T_1$, if $W_2 < \tau$ and $Q_1 = 0$, then v lies in the last segment of the curve. Since along the last segment, $c_2^L(v)$ increases monotonically with v, we have in this case $v_1^* = v_1^L = L_1 - d_1$. This is part 4a). Whereas when $L_1 + L_2 > T_1$ but $W_2 \geq \tau$ or $Q_1 \neq 0$, the figure shows that $c_2^L(v)$ achieves its minimum at $v_1^* = (M_1 - 1)W_1$. This is part 4b).

Furthermore, when $W_1 \leq \tau/2$ and $L_1 + L_2 \leq T_1$, it can be shown that $v_1^U - v_1^L > W_1$. Figure 8 then gives v_1^* as in part 5) of the lemma. Again for brevity, we omit the details. ■

For convenience, for $i, j = 1$ or 2 and $i \neq j$ and an integer $a \leq W_i$, let $\pi_{a,i}$ denote the policy that serves a packets from job j first, then gives full priority to job i. Furthermore, when $W_1 < \tau/2$, $W_1 + W_2 > \tau$ and $L_1 + L_2 > T_1$, let $\underline{\pi_2}$ denote the following policy: if $W_2 \geq \tau$, then first gives priority to job 1 until W_1 packets from job 1 left, then gives priority to job 2; if $W_2 < \tau$, then first serves n packets from job 1, next serves m blocks of W_2 packets from job 2 and m blocks of W_1 packets from job 1 alternatively, then gives priority to job 2. Here m and n are integers that satisfy $L_1 + L_2 - T_2 - W_1 = m(W_1 + W_2 - \tau) + n$ and $n \in [0, W_1 + W_2 - \tau]$. It can be seen that policy $\underline{\pi_2}$ corresponds to case 4b) in Lemma 5 with $v_1(\underline{\pi_2}) = v_1^*$. Policy $\underline{\pi_1}$ is similarly defined.

Next lemma gives the cost functions of policies that will be used later when developing optimal policies and suboptimal policies.

Lemma 6: When $W_1 + W_2 > \tau$, for $i, j = 1$ or 2 and $i \neq j$, the cost functions of policy $\pi_{0,i}$ is

$$c(\pi_{0,i}) = \begin{cases} 2T_i + f_j(L_1 + L_2 - T_i) & \text{when } L_1 + L_2 > T_i \\ T_i + (N_j + 1)\tau - (\tau - W_i - r_j) & \text{when } L_1 + L_2 \leq T_i \end{cases} \tag{14}$$

When $W_1, W_2 < \tau$ and $L_1 + L_2 > T_i$, the cost function of policy $\pi_{d_j,i}$ is

$$c(\pi_{d_j,i}) = 2T_i + 2d_j + \begin{cases} f_j(L_1 + L_2 - T_i - d_j) & \text{when } Q_j \geq 1 \\ -W_i & \text{when } Q_j = 0 \end{cases} \tag{15}$$

Moreover, when $W_i < \tau$ and $L_1 + L_2 \leq T_i$, the cost function of policy $\pi_{r_j,i}$ is $c(\pi_{r_j,i}) = T_i + N_j\tau + 2r_j$. When $W_j < \tau$ and $L_1 + L_2 > \max\{T_1, T_2\}$, the cost function of policy $\underline{\pi_i}$ is $c(\underline{\pi_i}) = 2(L_1 + L_2) - W_j$.

These cost functions can be easily verified from Figure 1 to Figure 5. For brevity, we omit the proof.

When $W_1 + W_2 > \tau$ and $T_2 < L_1 + L_2 \leq T_1$, Lemma 2 says that $\Pi^* \subseteq \Pi_2$. That is, optimal policies over Π_2 are optimal policies over the entire policy space as well. In addition, it can be easily verified that in this case $Q_1 \neq 0$. Furthermore, in this case, it can also be verified that policy $\pi_{0,2}$ has the v_1^* described in Lemma 5 in case 1), 2), 3b) and 4a), policy $\pi_{d_1,2}$ has the v_1^* in case 3c), and policy $\pi_{0,1}$ and $\pi_{r_2,1}$ have the v_1^* in case 5a) and 5b), respectively. We hence have the following theorem:

Theorem 2: When $W_1 + W_2 > \tau$ and $T_2 < L_1 + L_2 \leq T_1$, the following policy is optimal:

1) when $W_1 > \tau/2$: if $W_1, W_2 < \tau$ and $d_1 < \tau - W_1$, policy $\pi_{d_1,2}$ is optimal; otherwise policy $\pi_{0,2}$ is optimal;
2) when $W_1 \leq \tau/2$: if $r_2 \geq W_1$, policy $\pi_{0,1}$ is optimal; otherwise policy $\pi_{r_2,1}$ is optimal.

By symmetry, the next theorem follows:

Theorem 3: When $W_1 + W_2 > \tau$ and $T_1 < L_1 + L_2 \leq T_2$, the following policy is optimal:

1) when $W_2 > \tau/2$: if $W_1, W_2 < \tau$ and $d_2 < \tau - W_2$, policy $\pi_{d_2,1}$ is optimal; otherwise policy $\pi_{0,1}$ is optimal;
2) when $W_2 \leq \tau/2$: if $r_1 \geq W_2$, policy $\pi_{0,2}$ is optimal; otherwise policy $\pi_{r_1,2}$ is optimal.

TABLE I

OPTIMAL POLICIES OVER Π_1 OR Π_2 WHEN $W_1 + W_2 > \tau$ AND $L_1 + L_2 > \max\{T_1, T_2\}$

W_1	W_2		
	$\geq \tau$	$[\tau/2, \tau]$	$\leq \tau/2$
$\geq \tau$	$\pi_{0,2}$,$\pi_{0,1}$	$\pi_{0,2}$,$\pi_{0,1}$	$\pi_{0,2}$,$\underline{\pi_1}$
$[\tau/2, \tau]$	$\pi_{0,2}$,$\pi_{0,1}$	$\pi_{0,2}$,$\pi_{d_1,2}$,$\pi_{0,1}$,$\pi_{d_2,1}$	$\pi_{0,2}$,$\pi_{d_1,2}$,$\pi_{0,1}$,$\underline{\pi_1}$
$\leq \tau/2$	$\underline{\pi_2}$,$\pi_{0,1}$	$\pi_{0,2}$,$\underline{\pi_2}$,$\pi_{0,1}$,$\pi_{d_2,1}$	N/A

Theorem 2 and 3 provide optimal policies under case $T_2 < L_1 + L_2 \leq T_1$ and $T_1 < L_1 + L_2 \leq T_2$. Now consider optimal policies under case $L_1 + L_2 > \max\{T_1, T_2\}$.

Lemma 5 shows that when $L_1 + L_2 > \max\{T_1, T_2\}$, over policy set Π_2, $v_1(\pi_{0,2})$, $v_1(\pi_{d_1,2})$ or $v_1(\underline{\pi_2})$ equals to v_1^* for different cases. Therefore, policy $\pi_{0,2}$, $\pi_{d_1,2}$ or $\underline{\pi_2}$ is optimal for the corresponding cases over Π_2. By symmetry, policy $\pi_{0,1}$, $\pi_{d_2,1}$ or $\underline{\pi_1}$ is optimal for the corresponding cases over Π_1. Table I lists the optimal policies over Π_1 or Π_2 for different cases. For example, when $W_1, W_2 \geq \tau$, $\pi_{0,2}$ is optimal over Π_2 and $\pi_{0,1}$ is optimal over Π_1. We therefore have policy $\pi_{0,2}$ and $\pi_{0,1}$ in the cell for $W_1 \geq \tau$ and $W_2 \geq \tau$.

Furthermore, since $\Pi^* \subseteq \Pi_1 \cup \Pi_2$, one of the optimal policies over Π_1 and Π_2 is also optimal over the entire policy space. Although we can further classify different cases and derive their corresponding optimal policies, the number of categories is significant, and the description of optimal policies is tedious and provides little insights. We therefore do not further classify the cases, but rather give the following theorem:

Theorem 4: When $W_1 + W_2 > \tau$ and $L_1 + L_2 > \max\{T_1, T_2\}$, for different cases in Table I, the policy that has the lowest cost among policies in the corresponding cell is optimal.

Notice that the cost functions for policies listed are given in Lemma 6.

Theorem 2 to 4 provide optimal policies under different cases. They show that the optimal policy is a function of job lengths, window sizes as well as the time interval τ. Furthermore, in most cases, the optimal policy does not change the priority of jobs but rather gives full priority to one job.

Corollary 1: If $W_1, W_2 \geq \tau$, the policy that gives priority to the shorter job is optimal.

Proof: When $W_i \geq \tau$ for some i, we have $T_i = L_i$, which leads to $L_1 + L_2 > T_i$. Therefore, when $W_1, W_2 \geq \tau$, $L_1 + L_2 > \max\{T_1, T_2\}$. The case thus falls into the first cell in Table I, which lists policy $\pi_{0,2}$ and $\pi_{0,1}$. From Lemma 6, their costs are: $c(\pi_{0,2}) = 2T_2 + f_1 L_1 + L_2 - T_2 = 2L_2 + (L_1 + L_2 - L_2) = L_1 + 2L_2$, where the second equality comes from the fact $f_1(l) = l$ and $T_2 = L_2$. Similarly, $c(\pi_{0,1}) = 2L_1 + L_2$. Hence, if $L_1 \leq L_2$, then policy $\pi_{0,1}$ is optimal, and if $L_2 \leq L_1$, then policy $\pi_{0,2}$ is optimal. The corollary thus holds. ■

As mentioned before, when $W_1, W_2 \geq \tau$, the window constraints take no effect on the service of both jobs. The problem thus becomes the traditional problem that minimizes average response time. The optimal SRPT policy also gives priority to the shorter file in our case. This is consistent with the above corollary.

Although the policies described in Theorem 2 to 4 are optimal, they are complex and it is not clear which parameter is essential. We hence gives a suboptimal policy in the following corollary, which is more insightful.

Corollary 2: When $W_1 + W_2 > \tau$, the following policy is suboptimal: if $W_i \leq \tau/2$ for $i = 1$ or 2, give full priority to job i; if $W_i > \tau/2$ for both $i = 1$ and 2, give priority to the shorter job. The difference between the optimal cost and the cost of this suboptimal policy is less than $\max\{W_1, W_2\}$.

This corollary can be proven by comparing the cost of the suboptimal policy described with the cost of the optimal policy described in Theorem 2 to 4, where the cost functions are given in Lemma 6. The comparison is tedious. Due to limited space, here we omit the details.

Notice that when $W_1 + W_2 > \tau$, $W_i \leq \tau/2$ for $i = 1$ or 2 means that job i has smaller window size. We can thus combine Theorem 1 and Corollary 2 and obtain the following suboptimal policy for all cases:

Theorem 5: The following policy is suboptimal for minimizing average response time of two jobs subject to window constraints: if both window sizes are greater than one half of the time interval, give full priority to the shorter job. Otherwise, give full priority to the job with the smaller window size. The difference between the optimal cost and the cost of this suboptimal policy is less than the maximum of the two window sizes.

Now it's clear that when $W_i > \tau/2$ for both $i = 1$ and 2, the job lengths determine the optimal policy. Otherwise, the window sizes are essential.

IV. CONCLUSION

This paper considers the scheduling problem that minimizes the average response time of two jobs subject to window constraints. The effects of the window constraints are presented, and the optimal policy is derived. In most cases, instead of changing priority of jobs at different times, the optimal policy gives full priority to one job. In traditional optimization problems without window constraints, the remaining processing times are the only parameters that affects the optimal policy (SRPT policy). While the suboptimal policy derived here shows that under window constraints, not only the job lengths (which correspond to the remaining processing times in traditional optimization problems) determine the optimal scheduling policy, but the relative magnitude of the window sizes are essential as well. One direction of future work is to compare the performance of the optimal policy and suboptimal policy with the traditional round-robin policy and SRPT policy.

REFERENCES

[1] N. Bansal and M. Harchol-Balter, "Analysis of SRPT Scheduling: Investigating Unfairness," ACM SIGMETRICS Performance Evaluation Review vol. 29, Issue 1, 2001.

[2] S. Muthukrishnan, Rajmohan Rajaraman, Anthony Shaheen and Johannes E. Gehrke, "Online Scheduling to Minimize Average Stretch," Proc. 40th Annual IEEE Symp. on Foundations of Computer Science, pp. 433-443, 1999.

[3] L. Schrage, "A Proof of the Optimality of the Shortest Remaining Processing Time Discipline," Operation Research, vol. 16, no.3, pp. 687-690, 1968.

[4] L. Schrage and L. W. Miller, "The Queue M/G/1 with the Shortest Remaining Processing Time Discipline," Operation Research, vol. 14, no.4, pp. 670-684, 1966.

[5] D. R. Smith, "A New Proof of the Optimality of the Shortest Remaining Processing Time Discipline," Operation Research, vol. 26, no.1, pp. 197-199, 1978.

[6] A. Wierman and M. Harchol-Balter, "Classifying Scheduling Policies with respect to Unfairness in an M/GI/1," ACM SIGMETRICS Performance Evaluation Review, vol. 31, Issue 1, 2003.