A Schedulability Condition for Deadline-Ordered Service Disciplines

Norival R. Figueira and Joseph Pasquale, Member, IEEE

Abstract—In a deadline-ordered service discipline, packets are assigned transmission deadlines and eligibility times and are transmitted in increasing order of deadlines. Different deadline-ordered service disciplines are distinguished by how they calculate deadlines and eligibility times. One of the more difficult analytical problems one faces when designing a new deadline-ordered service discipline is to prove that one can bound the end of transmission times of packets relative to their assigned deadlines, which we call schedulability. We show that, no matter how one calculates deadlines, there is a simple schedulability condition for deadline-ordered service disciplines. This schedulability condition is necessary and sufficient for preemptive deadline-ordered service disciplines, and for a server that allows the presence of nonreal-time packets (i.e., packets with no deadlines), it is also necessary and sufficient for nonpreemptive deadline-ordered service disciplines. We also address the schedulability problem for service disciplines in general, and show the optimality of deadline-ordered service disciplines. To demonstrate how our results simplify schedulability determination, we use them to prove the known schedulability conditions of VirtualClock, Packet-by-Packet Generalized Processor Sharing (PGPS), Stop-and-Go, and Delay-Earliest-Due-Date (Delay-EDD), and to prove a new result, the necessary schedulability condition of VirtualClock.

Index Terms—Admission control, deadline ordered, packet scheduling, quality of service, schedulability condition, service disciplines.

I. INTRODUCTION

MODERN high-speed networks allow applications to send new traffic types such as voice, video, and other kinds of interactive data. These new traffic types, generally called real time, require stringent performance guarantees in terms of throughput, end-to-end delay, and packet loss rate, which cannot be provided by first-come, first-served service disciplines and conventional window-based flow control schemes. In order to provide quality of service (QoS) guarantees for real-time traffic, several service disciplines have been proposed: Delay Earliest-Due-Date (Delay-EDD) [1], Jitter Earliest-Due-Date (Jitter-EDD) [12], Rate-Controlled Static-Priority Queueing (RCSP) [13], VirtualClock [14], Packet-by-Packet Generalized Processor Sharing (PGPS) [10]–[11], Stop-and-Go [5]–[7], Hierarchical Round Robin (HRR) [8], and more recently, Leave-in-Time [3].

For most of these service disciplines (which we will show in Section V), packets are assigned transmission deadlines and eligibility times, and eligible packets are transmitted according to the deadline-ordered scheduling policy: transmit packets in increasing order of deadlines, where ties are ordered arbitrarily. We will call a server (and its service discipline) deadline-ordered if it uses the deadline-ordered scheduling policy.

The eligibility time of a packet is the earliest time the packet can begin being transmitted. If all packets immediately become eligible for transmission upon arrival, the service discipline is called work-conserving; otherwise, it is called nonwork-conserving. We refer to the eligible packet sequence as the sequence of packets that become eligible for transmission over time in a server. Note that this definition applies to servers in general, i.e., not necessarily deadline-ordered. Fig. 1 shows a deadline-ordered server and its eligible packet sequence.

Different deadline-ordered service disciplines are distinguished by how they calculate deadlines and eligibility times. One of the more difficult analytical problems one faces when designing a new deadline-ordered service discipline is to solve the schedulability problem: Can the server transmit all packets before their deadlines? Since we want to solve the schedulability problem for deadline-ordered servers in general, i.e., without regard to how its service discipline calculates deadlines and eligibility times, we address the above

Fig. 1. Deadline-ordered server. An arriving packet is characterized by its session identifier s, its index i within the session, its length L, and its arrival time t. Upon arrival, the server assigns it an eligibility time e and a deadline F. A packet becomes eligible for transmission at its eligibility time, and the goal is to transmit it by its deadline. The service discipline is characterized by how it calculates deadlines and eligibility times, and by the scheduling policy it uses to decide which packet to transmit next from the set of eligible packets.

1The assignment may be implicit as in Stop-and-Go.
2We prefer the term deadline-ordered over the more common term earliest deadline first (EDF) because EDF is generally associated with service disciplines in which the deadline of each packet of a session is the sum of the arrival time of the packet and the fixed delay bound of the session (as Delay-EDD [1]). We make no assumption about how one calculates deadlines.

Manuscript received November 19, 1995; revised November 15, 1996; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. Zhang. This research was supported in part by NASA Grant NASA-NAGW-4260.

The authors are with the Computer Systems Laboratory, Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093-0114 USA.

Publisher Item Identifier S 1063-6692(97)03037-9.
schedulability problem from the point of view of the eligible packet sequence of the server. We say that the eligible packet sequence of a server is schedulable if the server is able to transmit all packets before their deadlines. Note that this definition applies to servers in general, i.e., not necessarily deadline-ordered.

For the purposes of this paper, the above definition of schedulability is adequate when we discuss preemptive servers. However, for nonpreemptive deadline-ordered servers, we will show later that there exist eligible packet sequences that, according to the above definition, are not schedulable, but for which delay bounds can be provided. Because of this, we propose a more flexible definition of schedulability. We say that the eligible packet sequence of a server is δ-schedulable if the server is able to bound the end of transmission of all packets relative to their deadlines, i.e., $F' \leq F + \delta$ for all packets, where $F'$ is the actual finishing transmission time of the packet, $F$ is the deadline of the packet, $\delta \geq 0$, and $\delta$ is a constant. Thus, by schedulable, we simply mean 0-schedulable.

Schedulability determination generally involves intricate and long proofs (e.g., Delay-EDD [1] and VirtualClock [2]). In this paper, we show that there is a simple schedulability condition for deadline-ordered servers. This schedulability condition is necessary and sufficient for preemptive deadline-ordered servers. For nonpreemptive deadline-ordered servers, it is only sufficient; however, given only that the eligible packet sequence of a nonpreemptive server satisfies this sufficient condition, there is no nonpreemptive scheduling policy with which the server can guarantee (in advance) a better delay bound than what it can provide with the nonpreemptive deadline-ordered scheduling policy. Finally, the condition becomes both necessary and sufficient if the server allows the presence of nonreal-time sessions (i.e., sessions composed of packets that have no deadlines).

The results in this paper apply most directly to deadline-ordered servers. However, they also address the schedulability problem of servers that use arbitrary scheduling policies. We show that if the eligible packet sequence of a server is schedulable under some (preemptive or nonpreemptive) scheduling policy, then it is schedulable under preemptive deadline-ordered scheduling. This optimality result was first shown in [9], but only for sessions with periodic arrivals. For a server that allows the presence of nonreal-time sessions, we show that if the eligible packet sequence of the server is δ-schedulable under some nonpreemptive scheduling policy, then it is δ-schedulable (with the same δ) under nonpreemptive deadline-ordered scheduling. For the general case in which the presence of nonreal-time sessions is undetermined (i.e., it does not matter whether the server allows the presence of nonreal-time sessions or not), we show that if the eligible packet sequence of the server is schedulable under some (preemptive or nonpreemptive) scheduling policy, then it is δ-schedulable under nonpreemptive deadline-ordered scheduling.

In [15], Zheng and Shin showed necessary and sufficient schedulability conditions for preemptive and nonpreemptive deadline-ordered service disciplines in which the deadline of each packet of a session is the sum of the arrival time of the packet and the fixed delay bound of the session. Their result for nonpreemptive service disciplines (which assumes the presence of nonreal-time sessions sharing the server) shows that the schedulability condition presented for Delay-EDD in [1] is restrictive because it is only a sufficient condition. Their result for preemptive service disciplines subsumes the results of Liu and Layland [9] for task scheduling in a real-time environment since the former relaxes some of the suppositions of the latter.

Our results generalize those in [15] because we make no assumptions regarding how one calculates deadlines and eligibility times of packets, and we also present results for both preemptive and nonpreemptive deadline-ordered service disciplines. In fact, we show that our results imply the results found in [15] for both preemptive and nonpreemptive service disciplines. Our general result for preemptive service disciplines also applies for task scheduling in a real-time environment.

To demonstrate how our results simplify the schedulability analysis of deadline-ordered service disciplines, we use them to prove the known schedulability conditions of VirtualClock, PGPS, Stop-and-Go, and Delay-EDD. We also use our results to prove a new result: the necessary schedulability condition of VirtualClock. The proofs are remarkably simple. Our results are not limited to work-conserving service disciplines, and they also apply to Stop-and-Go because, as we will show, it is indeed a deadline-ordered service discipline.

The remainder of this paper is structured as follows. Section II introduces the notation that is used in the paper. Sections III and IV develop results for nonpreemptive and preemptive service disciplines, respectively. In Section V, we apply our results on schedulability conditions to VirtualClock, PGPS, Stop-and-Go, and Delay-EDD. Section VI is a summary of the paper's contributions.

II. PRELIMINARY DEFINITIONS

The results in this paper pertain to a single server of capacity $C$ that transmits packets from a set $\Omega$ of sessions, where a $session$ is a sequence of packets. The packets of a session are numbered according to their arrival order at the server. The following terms refer to packets of session $s$ that arrive at a server: $t_{i,s}$ is the arrival time of the $i$th packet, where a packet has arrived only after its last bit has arrived, and $L_{i,s}$ is the length of the $i$th packet. Every packet of the session is assigned a deadline and an eligibility time. The transmission deadline (or simply, deadline) of the $i$th packet $F_{i,s}$ is the latest time that the last bit of the $i$th packet can be transmitted. The eligibility time of the $i$th packet $e_{i,s}$ is the earliest time the packet can begin being transmitted. Between $t_{i,s}$ and $e_{i,s}$, the $i$th packet is not eligible for service, and it will not be transmitted, even if the server becomes idle during the interval of time $[t_{i,s}, e_{i,s})$. Thus, at time $t$, a server has a set of eligible packets, $\xi(t)$, which is defined as

$$\xi(t) = \{p_{i,s} : t \geq e_{i,s} \text{ and the last bit of } p_{i,s} \text{ was not yet transmitted}\}$$

where $p_{i,s}$ is the $i$th packet of session $s$. Thus, the server can be nonwork-conserving since, at some time $t$, the set $\xi(t)$ may be
empty and the server may have packets in its queue. However, the server is never idle when there are eligible packets to transmit.

How a server works depends on its service discipline. A service discipline entails the calculation of deadlines and eligibility times of packets, and a scheduling policy to decide which packet to transmit next from the set of eligible packets. We call a server (and its service discipline) deadline-ordered if it uses the deadline-ordered scheduling policy, i.e., select the packet with the earliest deadline. A server is nonpreemptive if the selection can only occur after it is finished transmitting the current packet. A server is preemptive if the selection can occur whenever a new packet becomes eligible, possibly interrupting the transmission of a current packet and delaying its further transmission until the newly selected packet is transmitted.

In this paper, a real-time session is one that assigns finite deadlines to its packets (called real-time packets), whereas a nonreal-time session is one that assigns infinite deadlines to its packets (called nonreal-time packets). We assume that a server selects eligible nonreal-time packets for transmission only when there are no eligible real-time packets waiting for transmission.

The eligible packet sequence of a server is the sequence of packets that become eligible for transmission over time in the server. The eligible packet sequence of a server is defined by 6-tuples of the form (session identifier, packet number, packet length, arrival time, eligibility time, deadline), and contains packets from all the sessions sharing the server (see Fig. 1). We say that the eligible packet sequence of a server is schedulable if the server is able to transmit all real-time packets before their deadlines. Packets from nonreal-time sessions are not given any guarantees. While this definition of schedulability is adequate for preemptive deadline-ordered servers, we need a more general definition for nonpreemptive deadline-ordered servers. As we show in the next example, there are eligible packet sequences that are not schedulable by nonpreemptive deadline-ordered servers. However, it may still be possible to provide a delay bound, i.e., a guarantee that the amount by which the end of transmission times exceed deadlines is bounded by a constant.

Suppose that a nonpreemptive server of capacity $C$ is serving two sessions $s_a$ and $s_b$. Both sessions generate one packet of length $L$ bits every $3L/C$ seconds. The first packet of session $s_a$ is generated at time 0, just before the first packet of session $s_b$, which is generated at time $0^+$. Suppose that all packets immediately become eligible for transmission upon arrival, that the deadlines of the packets of session $s_a$ are equal to $arrival\ time + 3L/C$, and that the deadlines of the packets of session $s_b$ are equal to $arrival\ time + L/C$. In this case, no matter what nonpreemptive scheduling policy is used by the server, the server is not able to transmit the packets of session $s_b$ before their deadlines (see Fig. 2). This happens because the server cannot preempt the transmission of the packets of session $s_a$ (and, by definition, the server is never idle when there are eligible packets to transmit).

According to our definition of schedulability, the above eligible packet sequence is not schedulable by a nonpreemptive deadline-ordered server. However, this does not mean that a nonpreemptive deadline-ordered server cannot provide a delay bound for these sessions. In fact, a nonpreemptive deadline-ordered server guarantees that $\hat{F}_{i,s_b} < \hat{F}_{i,s_b} + L/C$ for all packets of session $s_b$, where $\hat{F}_{i,s_b}$ is the actual finishing transmission time of the $i$th packet of session $s_b$. Because of this, we propose a more flexible definition of schedulability.

We say that the eligible packet sequence of a server is $\delta$-schedulable if the server is able to bound the end of transmission of all real-time packets relative to their deadlines, i.e., $\hat{F}_{i,s} \leq F_{i,s} + \delta$ for all packets of all real-time sessions $s$, where $\delta \geq 0$ and $\delta$ is a constant.

In this paper, we answer the following questions: 1) What (schedulability) condition must be satisfied so that the eligible packet sequence of a deadline-ordered server is $\delta$-schedulable?, and 2) For which values of $\delta$ is the eligible packet sequence of a deadline-ordered server $\delta$-schedulable? We answer 1) by providing a (general) schedulability condition in which $\delta$ is a parameter, and this condition provides the answer to 2).

We show in this paper that $\delta = -\theta/C$ for preemptive deadline-ordered servers, and that $\delta = L_{MAX}/C - \theta/C$ for nonpreemptive deadline-ordered servers, where $L_{MAX}$ is the maximum packet length allowed in the server and $\theta$ represents an amount in bits ($\theta$ can assume any real value). From this formula, we see that $\theta$ is a necessary parameter for the answers to questions 1) and 2). We will show later (in Section V) that Delay-EDD, which is a nonpreemptive deadline-ordered service discipline, uses $\theta = L_{MAX}$ in order to achieve $\delta = 0$.

The condition established in question 1) comprises the admission control of a deadline-ordered service discipline, i.e., the set of tests (sometimes called schedulability tests) executed by the service discipline to verify that it can accept new sessions and keep the delay guarantees given to all sessions.

In this paper, we provide results for preemptive and nonpreemptive service disciplines. These results use the following measure of a session’s “appetite” for service.

$$Z_s(a,b) = \sum_{j=first_s(a,b)}^{last_s(a,b)} L_{j,s}$$

where $first_s(a,b) = \min\{i: e_{i,s} \geq a$ and $F_{i,s} \leq b\}$ and $last_s(a,b) = \max\{i: e_{i,s} \geq a$ and $F_{i,s} \leq b\}$. $Z_s(a,b)$ is the sum of the lengths of all packets $p_{j,s}$ of session $s$ for which the interval of time $[e_{j,s}, F_{j,s}]$ is contained entirely in $[a, b]$ (see Fig. 3).
**Definition 1 (Bounded Appetites Property):** We say that the eligible packet sequence of a server satisfies the **bounded appetites property** for a given value of $\theta$ if

$$\sum_{s \in \Omega} Z_s(a, b) + \theta \leq C$$

for any interval of time $[a, b]$, $b > a$, in which $\sum_{s \in \Omega} Z_s(a, b) > 0$, where $\Omega$ is the set of sessions sharing the server.

Note that, for $\theta \leq 0$, we can state the bounded appetites property without requiring that $\sum_{s \in \Omega} Z_s(a, b) > 0$ since, in this case, $\sum_{s \in \Omega} Z_s(a, b) = 0$ implies (2).

The following notation refers to packets of a session being transmitted by a server, and is used extensively in this paper. Terms containing subscripts $i$ and $s$ correspond to the $i$th packet of session $s$:

- $p_{i,s}$ denotes the packet itself;
- $F_{i,s}$ packet’s assigned deadline;
- $\hat{F}_{i,s}$ packet’s actual finishing transmission time;
- $t_{i,s}$ packet’s arrival time (when the last bit has arrived);
- $e_{i,s}$ packet’s eligibility time;
- $L_{i,s}$ packet’s length;
- $L_{\text{max},s}$ maximum length of packets of session $s$, i.e., $L_{\text{max},s} = \max\{L_{i,s}; i \geq 1\}$;
- $L_{\text{MAX}}$ maximum packet length allowed in the server; and
- $C$ capacity of the outgoing link of the server.

Define a **server busy period** to be a period of time in which the server always has packets to transmit. We extend the notation above by dropping the session identifier $s$ such that index $i$ indicates the index of a packet without regard to its session, in the order it ends transmission in a busy period. The following definitions apply to a server busy period (see Fig. 4).

- $F_i$ packet’s deadline (the “packet” being the $i$th packet to end transmission in a server busy period);
- $\hat{F}_i$ packet’s actual finishing transmission time;
- $t_i$ packet’s arrival time (when the last bit has arrived);
- $e_i$ packet’s eligibility time; and
- $L_i$ packet’s length.

Besides this notation, we also use the following:

- $P_{(\text{lower case letter})}$ to denote a packet, i.e., without regard to its session or time of service;
- $L_{(p,s)}$ to denote the length of packet $p_s$;
- $F_{(p,s)}$, $\hat{F}_{(p,s)}$ to denote the deadline and the actual end of transmission time of packet $p_s$, respectively.

### III. NON-PREEMPTIVE SCHEDULING

In this section, we prove a sufficient condition for the schedulability of nonpreemptive deadline-ordered servers (this is Theorem 1) and necessary conditions for the schedulability of nonpreemptive servers in general (these are Theorems 2 and 3). The two necessary conditions refer to distinct traffic scenarios. Some of the results in this section are summarized in Theorems 5 and 6. We also address the schedulability problem of servers that use arbitrary scheduling policies.

**Theorem 1:** Consider a nonpreemptive deadline-ordered server. If the eligible packet sequence of the server satisfies the bounded appetites property for a given value of $\theta$, then

$$\hat{F}_{i,s} < F_{i,s} + \frac{L_{\text{MAX}}}{C} - \frac{\theta}{C}, \quad l \geq 1$$

for all real-time sessions $s$ in $\Omega$, where $\Omega$ is the set of sessions sharing the server.

In Theorem 1, it does not matter whether there are nonreal-time sessions sharing the server or not.

Theorem 1 determines that, if the eligible packet sequence of a nonpreemptive deadline-ordered server satisfies the bounded appetites property for a given value of $\theta$, then this eligible packet sequence is $\delta$-schedulable, where $\delta = L_{\text{MAX}}/C - \theta/C$.

**Definition 2 (Ordering Property):** We say that session $s$ satisfies the **ordering property** if $e_{i,s} \leq e_{i+1,s}$ and $F_{i,s} \leq \hat{F}_{i+1,s}$ for $i \geq 1$.

For a session $s$ that satisfies the ordering property, (3) can be alternatively stated with $L_{\text{max},s}$ in place of $L_{\text{MAX}}$, where $L_{\text{max},s}$ is the maximum length of packets of the sessions in
Fig. 5. Illustration of Case 1 in the proof of Theorem 1. Shown is a server busy period beginning at time \( t \), and an interval of time over which packets are transmitted. The server is idle before time \( t \). For 1 \( \leq j \leq i \), we have that \( F_j \leq F_i \).

\[ \Omega = \{ s \}. \] More precisely, \( L_{\text{max},s} = \max\{L_{\text{max},s'} : s' \neq s \) and \( s' \in \Omega \}. \] This alternative result is important for service disciplines such as VirtualClock [14] where all sessions satisfy the ordering property.

**Proof of Theorem 1:** Consider the \( i \)th transmitted packet of a server busy period beginning at time \( t \). We consider two cases.

**Case 1:** All the packets that were transmitted before the \( i \)th transmitted packet have deadlines no later than \( F_i \), i.e., \( F_j \leq F_i \) for 1 \( \leq j \leq i \) (see Fig. 5).

Since all the packets that were transmitted in \([t, \hat{F}_i]\) have eligibility times and deadlines in \([t, F_i]\), we conclude that \( \sum_{s \in \Omega} Z_s(t, F_i) \) is larger than or equal to the total number of bits transmitted in \([t, \hat{F}_i]\). This is not necessarily an equality because there can be packets with eligibility times in \([t, F_i]\), and deadlines equal to \( F_i \) that were not transmitted in \([t, \hat{F}_i]\).

Thus,

\[
\hat{F}_i \leq t + \frac{\sum_{s \in \Omega} Z_s(t, F_i)}{C}.
\]  

(4)

From (2) with \( a = t \) and \( b = F_i \),

\[
t + \frac{\sum_{s \in \Omega} Z_s(t, F_i)}{C} \leq F_i - \frac{\theta}{C}.
\]  

(5)

From (4) and (5), we have that \( \hat{F}_{i,s} \leq F_i - \theta/C \), which implies that \( \hat{F}_{i,s} \leq F_i - \theta/C \) since the \( i \)th packet to end transmission in the busy period is the \( i \)th packet of some session \( s \). This completes the proof of Case 1.

**Case 2:** There is at least one packet whose deadline is later than \( F_i \) and that was transmitted before the \( i \)th transmitted packet. Therefore, there exists a \( k \) such that \( k = \max\{j : F_j > F_i \) and \( j < i\). Thus, the \( k \)th transmitted packet is the last packet before the \( i \)th transmitted packet such that \( F_k > F_i \). If there is no \( k \) satisfying this definition, the problem reduces to Case 1. This condition may occur because the server is non-preemptive, and a packet may become eligible after the server has begun serving a packet that has a later deadline than this new eligible packet (Fig. 6).

Assume that the \( k \)th transmitted packet begins transmission at time \( \hat{t} \) (see Fig. 7).

This is a server busy period; therefore,

\[
\hat{F}_i = \hat{F}_k + \sum_{j=k+1}^{i} \frac{L_j}{C}.
\]  

(6)

Fig. 6. Example where a non-preemptive deadline-ordered server does not transmit packets in increasing order of deadlines. This figure shows an interval of time over which packets are being transmitted. The server is idle before time \( e_1.s_a \). At time \( e_1.s_a \), the first packet of session \( s_a \) (i.e., \( p_1.s_a \)) becomes eligible. Packet \( p_1.s_a \) has deadline \( F_1.s_a \). The server begins the transmission of packet \( p_1.s_a \) immediately (i.e., at \( e_1.s_a \)), otherwise, it would become idle. At time \( e_5.s_b \), the fifth packet of session \( s_b \) (i.e., \( p_5.s_b \)) becomes eligible. Packet \( p_5.s_b \) has deadline \( F_5.s_b \). Note that packet \( p_1.s_a \) has a later deadline than packet \( p_5.s_b \) (i.e., \( F_1.s_a > F_5.s_b \)), but packet \( p_1.s_a \) is transmitted before packet \( p_5.s_b \).

Fig. 7. Illustration of Case 2 in Theorem 1. This figure shows an interval of time over which packets are being transmitted. The server busy period begins at time \( t \). The \( k \)th transmitted packet has a deadline later than that of the \( i \)th transmitted packet. However, the \( k \)th transmitted packet is transmitted before the \( i \)th transmitted packet because the \( i \)th transmitted packet becomes eligible for service after \( t \).

The definition of \( k \) implies that the \( j \)th transmitted packet, \( k + 1 \leq j \leq i \) since \( F_j \leq F_i < F_k \) and \( \hat{F}_i > \hat{F}_k \), i.e., the \( j \)th transmitted packet becomes eligible for service after \( t \) because it is transmitted after a packet with a later deadline. Note that, if \( e_j = \hat{t} \), the server would transmit the packet with the earlier deadline first. Therefore, the \( j \)th transmitted packet, \( k + 1 \leq j \leq i \), has eligibility time and deadline in \([e_{\text{min}}, F_i]\), where \( e_{\text{min}} = \min\{e_j : k + 1 \leq j \leq i\} \). Thus, \( e_{\text{min}} > \hat{t} \) and

\[
\sum_{s \in \Omega} Z_s(e_{\text{min}}, F_i) \geq \sum_{j=k+1}^{i} L_j,
\]

which is not necessarily an equality because there can be packets of a session \( s \) in \( \Omega \) that are not transmitted in \([\hat{F}_k, \hat{F}_i]\), but must be accounted for in \( Z_s(e_{\text{min}}, F_i) \) (e.g., a packet that becomes eligible after \( e_{\text{min}} \) and that has a deadline equal to \( F_i \)). Thus, from (2) with \( a = e_{\text{min}} \) and \( b = F_i \),

\[
\sum_{j=k+1}^{i} \frac{L_j + \theta}{F_i - e_{\text{min}}} \leq C,
\]

which implies that

\[
\hat{F}_k + \sum_{j=k+1}^{i} \frac{L_j}{C} < F_i + \frac{L_k}{C} - \frac{\theta}{C}
\]

(7)

since \( e_{\text{min}} > \hat{t} = \hat{F}_k - L_k/C \).

From (6) and (7),

\[
\hat{F}_i < F_i + \frac{L_k}{C} - \frac{\theta}{C}.
\]
Thus,
\[
\hat{F}_{l,s} < F_{l,s} + \frac{L_{\text{MAX}}}{C} - \frac{\theta}{C}
\]
since the \(i\)th transmitted packet in the busy period is the \(l\)th packet of some session \(s\). This completes the proof of Case 2.

In the proof of Case 2 above, if we assume that the \(i\)th transmitted packet belongs to a session for which the ordering property is satisfied, then the \(k\)th transmitted packet cannot be from the same session as the \(i\)th transmitted packet. Otherwise, given that \(F_i < F_k\), it would imply that \(e_i \leq e_k\), which is not the case. Thus, for a session that satisfies the ordering property, \(\hat{F}_{l,s} < F_{l,s} + \frac{L_{\text{MAX}}}{C} - \frac{\theta}{C}\).

**Theorem 2:** Consider a nonpreemptive server that employs some specific scheduling policy. Given the set of real-time sessions sharing the server, a particular pattern of arrivals for the traffic of these real-time sessions, and \(\theta\), if the server is able to schedule the transmission of packets such that
\[
\hat{F}_{l,s} < F_{l,s} + \frac{L_{\text{MAX}}}{C} - \frac{\theta}{C}, \quad l \geq 1 \tag{8}
\]
for all real-time sessions \(s\) sharing the server in the presence of any possible pattern of arrivals of nonreal-time packets, then the eligible packet sequence of the server satisfies the bounded appetites property for the given value of \(\theta\).

The scheduling policy in Theorem 2 can be any nonpreemptive scheduling policy with which the server is capable of guaranteeing (8).

**Theorem 2 can be generalized by replacing** \(L_{\text{MAX}}\) by a constant \(\gamma\), where \(0 < \gamma \leq L_{\text{MAX}}\). Thus, Theorem 2 can also be stated with \(L_{\text{MAX},s}\) in place of \(L_{\text{MAX}}\).

**Proof of Theorem 2:** The proof is by contradiction. We assume that (8) is true for a particular pattern of arrivals of nonreal-time packets, and that
\[
\sum_{s \in \Omega} Z_s(f,g) + \theta \geq C + \alpha, \quad \alpha > 0 \tag{9}
\]
for some interval of time \([f,g]\), \(g > f\), in which \(\sum_{s \in \Omega} Z_s(f,g) > 0\). We show that we can modify that pattern of arrivals of nonreal-time packets to make (8) false for a real-time packet.

Call \(A\) the set of packets accounted for in \(\sum_{s \in \Omega} Z_s(f,g)\). Define an interval of time \([a,b]\) contained in \([f,g]\) such that \(a\) is equal to the earliest eligibility time among the eligibility times of the packets in \(A\), and \(b\) is equal to the latest deadline among the deadlines of the packets in \(A\). Thus,
\[
\sum_{s \in \Omega} Z_s(f,g) = \sum_{s \in \Omega} Z_s(a,b)
\]
which from (9) implies that (since \(g - f \geq b - a\))
\[
\sum_{s \in \Omega} Z_s(a,b) + \theta \geq C + \alpha. \tag{10}
\]

By the definition of set \(A\), the server has at least one packet becoming eligible for transmission at time \(a\). Thus, at time \(a\), the server is necessarily in a busy period. Consider a particular pattern of arrivals of nonreal-time packets for which (8) is satisfied. For this arrival pattern, call \(a'\) the time at which the server busy period that contains time \(a\) begins.

Now, suppose that we change that pattern of arrivals of nonreal-time packets by introducing a new nonreal-time packet \(p_y\) such that its eligibility time is \(a' - \varepsilon\), where \(\varepsilon > 0\). Assume that the server is idle at time \(a' - \varepsilon\), and that \(L(p_y)/C > \varepsilon\), where \(L(p_y)\) is the length of packet \(p_y\). This means that the server will begin the transmission of packet \(p_y\) at time \(a' - \varepsilon\), and that the transmission of packet \(p_y\) will extend beyond time \(a'\). This will delay the transmission of the packets in the busy period that begins at time \(a'\) with the original arrival pattern (see Fig. 8). In the new scenario, call \(p_x\) the packet from set \(A\) that is transmitted last by the server. Suppose that packet \(p_x\) is from session \(s_x\). Thus, the end of transmission time of packet \(p_x\), i.e., \(\hat{F}_{(p_x)}\), is
\[
\hat{F}_{(p_x)} \geq \sum_{s \in \Omega} Z_s(a,b) + \frac{L(p_x)}{C} + a - \varepsilon \tag{11}
\]

since the length of time the server takes to transmit all the packets in \(A\) is \(\sum_{s \in \Omega} Z_s(a,b)/C\) and the server was originally busy in the interval of time \([a',a]\). We state (11) as an inequality because we make no assumption about the server always being busy in \([a,\hat{F}_{(p_x)}]\).

Since packet \(p_x\) is in \(A\), \(\hat{F}_{(p_x)} \leq b\), where \(F_{(p_x)}\) is the deadline of packet \(p_x\), which from (10) implies that
\[
F_{(p_x)} \leq \sum_{s \in \Omega} Z_s(a,b) + \theta \geq C + \alpha + a. \tag{12}
\]

From (11) and (12),
\[
\hat{F}_{(p_x)} \geq F_{(p_x)} + \frac{L(p_x)}{C} - \frac{\theta}{C} + \frac{\alpha}{C(C + \alpha)} \left(\sum_{s \in \Omega} Z_s(a,b) + \theta\right) - \varepsilon. \tag{13}
\]

If
\[
\varepsilon \leq \frac{\alpha}{C(C + \alpha)} \left(\sum_{s \in \Omega} Z_s(a,b) + \theta\right)
\]
equation (13) implies that
\[
\hat{F}_{l,s_x} \geq F_{l,s_x} + \frac{L(p_x)}{C} - \frac{\theta}{C},
\]
since packet \(p_x\) is the \(l\)th packet of session \(s_x\) (for some \(l \geq 1\)). If \(L(p_x) = L_{\text{MAX}}\), this contradicts our assumption that (8) is true for all real-time sessions.

\[\square\]
**Theorem 3:** Consider a nonpreemptive server that employs some specific scheduling policy. Given the set of real-time sessions sharing the server, a particular pattern of arrivals for the traffic of these real-time sessions, and \( \theta \), if the server is able to schedule the transmission of packets such that

\[
\hat{F}_{l,s} \leq F_{l,s} - \frac{\theta}{C}, \quad l \geq 1
\]

for all real-time sessions \( s \) sharing the server, then the eligible packet sequence of the server satisfies the bounded appetites property for the given value of \( \theta \).

In this theorem, it does not matter whether there are nonreal-time sessions sharing the server or not.

**Proof of Theorem 3:** The proof is by contradiction. We assume (14), and suppose that

\[
\sum_{s \in \Omega} Z_s(a, b) + \theta \cdot b - a > C
\]

for an interval of time \([a, b], b > a\), in which \( \sum_{s \in \Omega} Z_s(a, b) > 0 \).

Among the packets accounted for in \( \sum_{s \in \Omega} Z_s(a, b) \), packet \( p_x \) is the one that finishes transmission last. If \( \hat{F}(p_x) \) is the end of transmission time of packet \( p_x \),

\[
\hat{F}(p_x) \geq \sum_{s \in \Omega} Z_s(a, b) + a \cdot \frac{C}{C} + a > F(p_x) - \frac{\theta}{C}
\]

since packet \( p_x \) ends transmission after all of the packets that are accounted for in \( \sum_{s \in \Omega} Z_s(a, b) \) are transmitted, and the transmission of these packets cannot begin before time \( a \). We state (16) as an inequality because we make no assumption about the server always being busy in \([a, \hat{F}(p_x)]\).

From (15) and the fact that \( F(p_x) \leq b \) (since packet \( p_x \) is accounted for in \( \sum_{s \in \Omega} Z_s(a, b) \)), where \( F(p_x) \) is the deadline of packet \( p_x \),

\[
\sum_{s \in \Omega} Z_s(a, b) + a > F(p_x) - \frac{\theta}{C},
\]

From (16) and (17), we have that \( \hat{F}(p_x) > F(p_x) - \frac{\theta}{C} \), which implies that \( \hat{F}_{l,s} > F_{l,s} - \frac{\theta}{C} \) since packet \( p_x \) is the \( l \)th packet of some session \( s \). This is a contradiction. \( \square \)

In the presence of nonreal-time sessions, the necessary condition for schedulability of any nonpreemptive server (see Theorem 2) is equal to the sufficient schedulability condition of nonpreemptive deadline-ordered servers (see Theorem 1). Thus, in the presence of nonreal-time sessions, if the eligible packet sequence of a server is \( \delta \)-schedulable under some nonpreemptive scheduling policy, then it is also \( \delta \)-schedulable (with the same \( \delta \)) under nonpreemptive deadline-ordered scheduling.

Theorems 1 and 3 lead to another optimality result: if the eligible packet sequence of a server is schedulable under some (preemptive or nonpreemptive) scheduling policy, then it is \( \delta \)-schedulable under nonpreemptive deadline-ordered scheduling. This result is valid regardless of whether nonreal-time sessions are present or not. We include preemptive scheduling policies in the above result because we show in Section IV that the result of Theorem 3 is also valid for preemptive servers (this is Theorem 8).

Given a server with a particular eligible packet sequence that does not contain packets from nonreal-time sessions, the delay bound provided under nonpreemptive deadline-ordered scheduling may be larger by one packet’s transmission time than the delay bound provided under some other nonpreemptive scheduling policy. The following example shows how this may happen.

Suppose that before time \( t \), no packets arrived at a server of capacity \( C \) (see Fig. 9). At time \( t \), packets \( p_x \) from session \( s_x \) and \( p_y \) from session \( s_y \) arrive and immediately become eligible for transmission. At time \( t + L/(2C) \), packet \( p_z \) from session \( s_z \) arrives and immediately becomes eligible for transmission. Packets \( p_x, p_y \), and \( p_z \) have lengths \( L, L/2 \), and \( L \), respectively. The deadlines of packets \( p_x, p_y \), and \( p_z \) are \( F(p_x), F(p_y), \) and \( F(p_z), \) respectively, where \( F(p_x) = t + L/(2C) + 2L/C, F(p_y) = t + 3L/C, \) and \( F(p_z) = t + L/(2C) + L/C \). There are no other packet arrivals in the interval of time \([t, F(p_y)]\). In this case, the eligible packet sequence of the server satisfies the bounded appetites property for \( \theta = 0 \). Suppose that the server employs a nonpreemptive scheduling policy \( P \) (e.g., the shortest packet first scheduling policy) that selects packets \( p_y, p_z, \) and \( p_x \) in this order. In this case, the deadlines of these packets are satisfied since \( F(p_x) = F(p_z), \hat{F}(p_y) < F(p_y), \) and \( \hat{F}(p_z) = F(p_z) \). If the server employed the nonpreemptive deadline-ordered scheduling policy, it would select packet \( p_x \) first since it has an earlier deadline than packet \( p_y \). In this case, the transmission order would be \( p_x, p_z, \) and \( p_y \). This would result in the end of transmission time of packet \( p_x \) being \( \hat{F}(p_x) = F(p_x) + L/(2C) \), which violates the deadline. However, note that \( \hat{F}(p_x) < F(p_x) + L_{MAX}/C \) for \( L_{MAX} = L \), which complies with the delay bound of nonpreemptive deadline-ordered servers, i.e., (3).
This problem is not an intrinsic one of deadline-ordered servers. It may happen with any nonpreemptive server. In this example, when the server employs the nonpreemptive scheduling policy \( P \), it takes advantage of the length of packet \( p_y \) being \( L/2 \). If packet \( p_y \) had length \( L \), there is no nonpreemptive scheduling policy with which the server can transmit these packets such that \( F_{(p_y)} \leq F_{(p_y)} \). In other words, simply satisfying the bounded appetites property for some value of \( \theta \) is not sufficient to guarantee (14). In summary, given the eligible packet sequence of a nonpreemptive server (i.e., we know in advance the eligible packet sequence), and the fact that this eligible packet sequence satisfies the bounded appetites property, there can be a nonpreemptive scheduling policy with which the server can guarantee all deadlines as in (14). However, given only that the eligible packet sequence of a nonpreemptive server satisfies the bounded appetites property (i.e., the only information we have in advance about the eligible packet sequence of the server is that it satisfies the bounded appetites property), there is no nonpreemptive scheduling policy with which the server is capable of guaranteeing a better delay bound than what it can provide with the nonpreemptive deadline-ordered scheduling policy. We formalize this result in Theorem 4 below.

In order to compensate for eligible packet sequences similar to the one in Fig. 9, a service discipline may set \( \theta = L_{\text{MAX}} \) in its schedulability test, i.e., (2). In fact, we will show later that this is the case with Delay-EDD.

Theorem 4: Given only that the eligible packet sequence of a nonpreemptive server satisfies the bounded appetites property for some value of \( \theta \), there is no nonpreemptive scheduling policy with which the server is capable of guaranteeing a better delay bound than what it can provide with the nonpreemptive deadline-ordered scheduling policy.

Proof of Theorem 4: The proof is by contradiction. Suppose that the eligible packet sequence of a nonpreemptive server satisfies the bounded appetites property for some value of \( \theta \), and that this server with a nonpreemptive scheduling policy \( P \) guarantees a delay bound of what it can provide with the nonpreemptive deadline-ordered scheduling policy less \( \alpha \), where \( \alpha > 0 \). We show a scenario that contradicts these assumptions.

Consider an eligible packet sequence similar to the one in Fig. 9 with the following exceptions: all packets have length \( L, L = L_{\text{MAX}} \), packet \( p_z \) arrives at time \( t+\epsilon, 0 < \epsilon \leq L/(2C) \), and the deadline of packet \( p_z \) is \( F_{(p_z)} = t + \epsilon + L/C \). This eligible packet sequence satisfies the bounded appetites property for \( \theta = 0 \). Thus, from Theorem 1, a nonpreemptive deadline-ordered server guarantees that \( F_{(p_z)} < F_{(p_z)} + L/C \). Under the nonpreemptive scheduling policy \( P \) (in fact, under any nonpreemptive scheduling policy), packet \( p_z \) cannot finish transmission before time \( F_{(p_z)} + L/C - \epsilon \). This time is larger than \( F_{(p_z)} + L/C - \alpha \) for \( \epsilon < \alpha \). This is a contradiction.

Theorems 5 and 6 below summarize some important consequences of these results.

Theorem 5: Consider a nonpreemptive deadline-ordered server that allows the presence of nonreal-time sessions. For a given value of \( \theta \), the server can guarantee that

\[
\hat{F}_{i,s} < F_{i,s} + \frac{L_{\text{MAX}}}{C} - \frac{\theta}{C}, \quad l \geq 1
\]

(18)

for all real-time sessions \( s \) sharing the server, if and only if the eligible packet sequence of the server satisfies the bounded appetites property for the given value of \( \theta \).

Note that, if all sessions satisfy the ordering property, then (18) can be alternatively stated with \( L_{\text{MAX}} \) in place of \( L_{\text{MAX}} \).

Proof of Theorem 5: With the stated constraints, satisfying the bounded appetites property is a sufficient condition for (18) according to Theorem 1, and satisfying the bounded appetites property is a necessary condition for (18) according to Theorem 2.

Theorem 6: Consider a nonpreemptive deadline-ordered server for which it is not possible for a packet to become eligible at a time \( t \) such that \( u < t < v \), where \( u \) and \( v \) are the beginning and the end of transmission times of another packet, respectively.\(^3\) For a given value of \( \theta \), the server can guarantee that

\[
\hat{F}_{i,s} \leq F_{i,s} - \frac{\theta}{C}, \quad l \geq 1
\]

(19)

for all real-time sessions \( s \) sharing the server, if and only if the eligible packet sequence of the server satisfies the bounded appetites property for the given value of \( \theta \).

With the traffic constraints of Theorem 6, the server never needs to preempt a packet, and the presence of nonreal-time sessions is irrelevant.

Proof of Theorem 6: Satisfying the bounded appetites property is a necessary condition for (19) according to Theorem 3 (since the additional constraints of Theorem 6 do not change the validity of Theorem 3). The proof of the sufficient condition is based on the proof of Theorem 1. Case 1 of Theorem 1 applies directly in this case, and Case 2 of Theorem 1 applies with a small adaptation. Note that \( \epsilon_{\text{min}} = \hat{F}_k \) because of the traffic constraints of Theorem 6. Thus, (7) becomes

\[
\hat{F}_k + \sum_{j=k+1}^{i} \frac{L_j}{C} \leq F_i - \frac{\theta}{C}
\]

which, following the rest of the proof of Case 2, implies (19).

IV. PREEMPTIVE SCHEDULING

In this section, we prove the sufficient (Theorem 7) and necessary (Theorem 8) condition for the schedulability of preemptive deadline-ordered servers. For all of the results in this section, it does not matter whether there are nonreal-time sessions sharing the server or not. The results in this section are summarized in Theorem 9.

\(^3\) We will show later that this is the case with Stop-and-Go.
Theorem 7: Consider a preemptive deadline-ordered server. If the eligible packet sequence of the server satisfies the bounded appetites property for a given value of \( \theta \), then

\[
\hat{F}_{l,s} \leq F_{l,s} - \frac{\theta}{C}, \quad l \geq 1
\]

for all real-time sessions \( s \) in \( \Omega \), where \( \Omega \) is the set of sessions sharing the server.

Theorem 7 determines that, if the eligible packet sequence of a preemptive deadline-ordered server satisfies the bounded appetites property for a given value of \( \theta \), then this eligible packet sequence is \( \delta \)-schedulable, where \( \delta = -\theta/C \).

Proof of Theorem 7: Consider the \( l \)th packet to end transmission in a busy period that begins at time \( t \). We have two cases to consider.

Case 1: All packets that have some bits transmitted in \([t, \hat{F}_i]\) have deadlines less than or equal to \( F_i \). The proof is similar to the proof of Case 1 of Theorem 1.

Case 2: There is at least one packet whose deadline is later than \( F_i \) that has some bits transmitted in \([t, \hat{F}_i]\). If there is none, the problem reduces to Case 1. This condition may occur because the server is preemptive, and a packet may become eligible after the server has already begun serving a packet that has a later deadline than this new eligible packet (Fig. 10).

Define \( A = \{p_y : \text{packet } p_y \text{ had some bits transmitted in } [t, \hat{F}_i] \text{ and } F_i < F_{l(p_y)}\} \). Define Latest.time \( (p_z) \) to be the latest time in \([t, \hat{F}_i]\) packet \( p_z \) had a bit being transmitted. Call \( p_z \) the packet in \( A \) for which Latest.time \( (p_z) \) is greater than or equal to Latest.time \( (p_y) \), for any packet \( p_y \) in \( A \). Assume that \( g = \text{Latest.time}(p_z) \). Therefore, all packets with some bits transmitted in \([g, \hat{F}_i]\) have deadlines no later than \( F_i \). These packets have eligibility times no earlier than \( g \); otherwise, packet \( p_z \) would have been preempted before time \( g \).

We conclude that all packets with some bits transmitted in \([g, \hat{F}_i]\) have eligibility times and deadlines in \([g, F_i]\). Thus, \( \sum_{s \in \Omega} Z_s(g, F_i) \) is larger than or equal to the total number of bits transmitted in \([g, \hat{F}_i]\). This is not necessarily an equality because there can be packets with eligibility times in \([g, F_i]\) and deadlines equal to \( F_i \) that did not have any bits transmitted in \([g, \hat{F}_i]\). Thus,

\[
\hat{F}_i \leq g + \frac{\sum_{s \in \Omega} Z_s(g, F_i)}{C}.
\]

From (2) with \( a = g \) and \( b = F_i \),

\[
g + \frac{\sum_{s \in \Omega} Z_s(g, F_i)}{C} \leq F_i - \frac{\theta}{C}.
\]

From (20) and (21), we have that \( \hat{F}_i \leq F_i - \frac{\theta}{C} \), which implies that \( \hat{F}_{l,s} \leq F_{l,s} - \frac{\theta}{C} \) since the \( l \)th packet to end transmission in the busy period is the \( l \)th packet of some session \( s \). This completes the proof of Case 2.

Theorem 8: Consider a preemptive server that employs some specific scheduling policy. Given the set of real-time sessions sharing the server, a particular pattern of arrivals for the traffic of these real-time sessions, and \( \theta \), if the server is able to schedule the transmission of packets such that

\[
\hat{F}_{l,s} \leq F_{l,s} - \frac{\theta}{C}, \quad l \geq 1
\]

for all real-time sessions \( s \) sharing the server, then the eligible packet sequence of the server satisfies the bounded appetites property for the given value of \( \theta \).

Proof of Theorem 8: The proof is identical to the proof of Theorem 3.

Note that the scheduling policy in Theorem 8 can be any preemptive scheduling policy with which the server is capable of providing the aforementioned delay guarantee (i.e., (22)). Since satisfying the bounded appetites property is a sufficient condition for preemptive deadline-ordered servers to guarantee (22) (see Theorem 7), and also a necessary condition for nonpreemptive and preemptive servers to guarantee (22) (see Theorems 3 and 8, respectively), we conclude that if the eligible packet sequence of a preemptive server is schedulable under some (preemptive or nonpreemptive) scheduling policy, then it is schedulable under preemptive deadline-ordered scheduling.

Theorem 9: Consider a preemptive deadline-ordered server. For a given value of \( \theta \), the server can guarantee that

\[
\hat{F}_{l,s} \leq F_{l,s} - \frac{\theta}{C}, \quad l \geq 1
\]

for all real-time sessions \( s \) sharing the server, if and only if the eligible packet sequence of the server satisfies the bounded appetites property for the given value of \( \theta \).

Proof of Theorem 9: By Theorem 7, satisfying the bounded appetites property is a sufficient condition for (23), and by Theorem 8, satisfying the bounded appetites property is a necessary condition for (23).

V. APPLICATIONS

In this section, we use our results to show how they simplify the determination of the schedulability condition of some well-known service disciplines such as VirtualClock, PGPS, Stop-and-Go, and Delay-EDD. We also discuss the applicability of our results in the analysis of scheduling algorithms for multiprogramming in a real-time environment.
A. VirtualClock

In [2], it was proved that VirtualClock [14] is able to provide an upper bound on delay for some types of sessions if

$$\sum_{s \in \Omega} r_s \leq C$$  \hfill (24)

where $C$ is the capacity of the outgoing link of the server, $r_s$ is the reserved rate of session $s$ (the reserved rate of a nonreal-time session is equal to zero), and $\Omega$ is the set of sessions sharing the outgoing link of the server. Given this constraint, it was proved in [2] that

$$\hat{F}_{l,s} < F_{l,s} + \frac{L_{\text{max}}}{C}, \quad l \geq 1$$  \hfill (25)

for all real-time sessions $s$ sharing the server.

The proof of this result in [2] is long and intricate. We now show how the same result can be easily proved with the result of Theorem 1. We also prove a new result, that (24) is also a necessary condition for (25).

**Theorem 10:** For a VirtualClock server in the presence of nonreal-time sessions, (24) is the sufficient and necessary condition for (25).

**Proof of Theorem 10:** In VirtualClock, the eligibility time of a packet is equal to the arrival time of the packet. We divide the proof in two cases.

**Proof of the Sufficient Condition:** In VirtualClock, the deadline of the $i$th packet of a session $s$ is (see [2])

$$F_{i,s} = \max\{t_{i,s}, F_{i-1,s}\} + \frac{L_{i,s}}{r_s}, \quad i \geq 1$$

$$F_{0,s} = 0.$$  \hfill (26)

Consider an interval of time $[a, b], b > a$, where $Z_s(a, b) > 0$. From (26),

$$F_{\text{last}_s(a, b), s} \geq t_{\text{first}_s(a, b), s} + \sum_{j = \text{first}_s(a, b)}^{\text{last}_s(a, b)} \frac{L_{j,s}}{r_s}.$$  \hfill (27)

From (1), (27), and the fact that $t_{\text{first}_s(a, b), s} \geq a$ and $F_{\text{last}_s(a, b), s} \leq b$,

$$Z_s(a, b) \leq r_s(b - a)$$

which implies that

$$\sum_{s \in \Omega} Z_s(a, b) \leq \sum_{s \in \Omega} r_s.$$  \hfill (31)

This inequality and (24) imply that the eligible packet sequence of a VirtualClock server satisfies the bounded appetites property for $\theta = 0$. This result and Theorem 1 complete the proof of the sufficient condition.

**Proof of the Necessary Condition:** The proof is by contradiction. Assume that (25) is true and that (24) is false, i.e.,

$$\sum_{s \in \Omega} r_s > C.$$  \hfill (28)

Consider the following scenario. In the interval of time $[a, b]$, the real-time sessions in $\Omega$ generate traffic such that

$$t_{f_s, s} = a$$

$$F_{t_{f_s, s}} = b$$

$$t_{i,s} \leq F_{i-1,s}, \quad f_s < i \leq l_s$$

and

$$t_{f_s, s} \geq F_{f_s - 1,s} \quad \text{if } f_s > 1$$  \hfill (32)

where $f_s$ and $l_s$ are the indexes of the first and the last packets of session $s$ to arrive in the interval of time $[a, b]$, respectively.

From (26), (31), and (32), $F_{i,s} = F_{i-1,s} + L_{i,s}/r_s$ for $f_s < i \leq l_s$ and $F_{f_s,s} = t_{f_s,s} + L_{f_s,s}/r_s$, which imply that

$$F_{l,s} = t_{f_s,s} + \sum_{j = f_s}^{l_s} \frac{L_{j,s}}{r_s}.$$  \hfill (33)

From (29), (30), (33), (1), and the fact that $\text{first}_s(a, b) = f_s$ and $\text{last}_s(a, b) = l_s$,

$$r_s(b - a) = Z_s(a, b).$$  \hfill (34)

From (34) and (28),

$$\sum_{s \in \Omega} \frac{Z_s(a, b)}{b - a} > C.$$

The previous inequality implies that the eligible packet sequence of a VirtualClock server does not satisfy the bounded appetites property for $\theta = 0$. This result and Theorem 2 for $\theta = 0$ contradict the assumption that (25) is true.

B. PGPS and Tracking Service Disciplines

Packet-by-packet generalized processor sharing (PGPS) [10]–[11] is a service discipline that assigns as the deadline of a packet the end of transmission time the packet would have if the server were doing a bit-by-bit round robin ([10] calls GPS a bit-by-bit round robin server). PGPS is an example of a tracking service discipline, i.e., a service discipline that assigns as deadlines the end of transmission times obtained in a simulation of a real server. Lemma 1 below implies that a server that employs a (preemptive or nonpreemptive) tracking service discipline has an eligible packet sequence that satisfies the bounded appetites property for $\theta = 0$. Thus, using either Theorem 1 or Theorem 7 (the appropriate theorem depends on the kind of service discipline, i.e., preemptive or nonpreemptive), we obtain bounds on the end of transmission times of packets in the tracking service discipline. These results for tracking service disciplines agree with the results developed in [4]. For PGPS, which is a nonpreemptive tracking service discipline, Theorem
1 implies that \( \hat{F}_{\text{PGPS}} \leq F_{\text{GPS}} + L_{\text{max}} + \bar{s}/C \) (since sessions satisfy the ordering property), where \( F_{\text{PGPS}} \) is the actual end of transmission time of a packet in the PGPS server and \( F_{\text{GPS}} \) is the deadline of the packet (which is equal to the end of transmission time of the packet in the simulated GPS server).

In a GPS server, the end of transmission time of a packet that arrives at a time \( t \) is only known at time \( t \) if no additional packets were to arrive until the presumed end of transmission time of that packet. This is a problem for the implementation of PGPS since, by its definition, PGPS needs to transmit packets in increasing order of their end of transmission times in the simulated GPS server. In [10], an implementation was proposed for PGPS in which packets are stamped with virtual finish times instead of the deadlines in units of actual time. That implementation works because the ordering of the proposed virtual finish time values is the same as the ordering of the end of transmission time values of the packets in the simulated GPS server. Therefore, although PGPS is implemented using virtual finish time, PGPS can be analyzed using its original definition (as we do here). In fact, the analysis of PGPS that is presented in [10]–[11] is also apart from its proposed implementation.

**Lemma 1:** In a server of capacity \( C \),

\[
\sum_{s \in \Omega} \left( \max_{j} \left\{ e_{i,s} \geq a \text{ and } F_{i,s} \leq b \right\} \frac{L_{j,s}}{b-a} \right) \leq C
\]  

(35)

for any interval of time \([a,b], b > a\), where \( e_{i,s} \) and \( F_{i,s} \) are the eligibility time and the end of transmission time of the \( i \)th packet of session \( s \), respectively, and \( \Omega \) is the set of sessions sharing the server.

The server in Lemma 1 is the one that a tracking service discipline simulates in order to calculate the deadlines of packets. We use the same notation (i.e., \( F_{i,s} \)) for the end of transmission times of packets in the simulated server and for the deadlines of packets in the tracking service discipline since the former assumes the same values as the latter.

Lemma 1 is true for a server in general, i.e., the server can do bit-by-bit round robin, packet-by-packet round robin, etc. Besides, the server may be nonwork-conserving. Note that (35) implies (2) for \( \theta = 0 \).

**Proof of Lemma 1:**

\[
\sum_{s \in \Omega} \left( \max_{j} \left\{ e_{i,s} \geq a \text{ and } F_{i,s} \leq b \right\} \frac{L_{j,s}}{b-a} \right)
\]

is the sum of the lengths of all packets that become eligible for transmission in \([a,b]\) and that are entirely transmitted in \([a,b]\). Since the server has capacity \( C \), it can transmit at most \( C(b-a) \) bits in \([a,b]\). Thus,

\[
\sum_{s \in \Omega} \left( \max_{j} \left\{ e_{i,s} \geq a \text{ and } F_{i,s} \leq b \right\} \frac{L_{j,s}}{b-a} \right) \leq C(b-a).
\]

\[\square\]

### C. Stop-and-Go

We now show that our results apply even to Stop-and-Go [5]–[7] since we show that Stop-and-Go is indeed a nonpreemptive deadline-ordered service discipline. Starting at a time origin, the time axis is divided into periods of some constant length \( T \), called a frame. Over one link, the time frames are viewed as traveling with the packets from one end of the link to the other end. Stop-and-Go is based on the notion of departing and arriving frames. The transmission of a packet that has arrived at any link during an arriving frame \( f \) should always be postponed until the beginning of the next departing frame that starts after the arriving frame \( f \) expires. Thus, for the packets that will be transmitted in a departing frame, the beginning of the frame is their eligibility time, and the end of the frame is their deadline. Stop-and-Go satisfies the assumptions in Theorem 6 because all the packets in a departing frame have as their eligibility times the beginning of the frame. Thus, we must have that \( \sum_{s \in \Omega} Z_{s} \text{ (frame)} \leq C \), with Stop-and-Go guarantees by imposing that sessions be \((r_{s},T)\)-smooth (i.e., during each frame of length \( T \), the arrived packets collectively have no more than \( r_{s}T \) bits) and \( \sum s r_{s} \leq C \), where \( s \) ranges over all sessions sharing the server. In this case, Theorem 6 guarantees that \( \hat{F}_{i,s} \leq F_{i,s} \), where \( F_{i,s} \) is equal to the end time of the frame.

### D. Delay-EDD

In [15], Zheng and Shin determined that the necessary and sufficient schedulability condition for deadline-ordered service disciplines that assign as the deadline of a packet of a session \( s \) the sum of the arrival time of the packet and the fixed delay bound of session \( s \), i.e., for the \( i \)th packet of session \( s \), \( F_{i,s} = t_{i,s} + d_{s} \), where \( d_{s} \) is a fixed delay bound for the packets of session \( s \). The schedulability condition that is proved in [15] for nonpreemptive servers applies to Delay-EDD [1] and assumes the presence of nonreal-time traffic. Our results subsume those in [15] because we do not assume that deadlines are calculated by a fixed delay from the arrival time of packets. In fact, we make no assumptions about the calculation of deadlines and eligibility times. We now show that the results presented in [15] can be derived from our general results. In order to do that, we need the following preliminary result.

**Theorem 11:** Consider a server in which the deadline of the \( i \)th packet of a real-time session \( s \) is calculated as \( F_{i,s} = t_{i,s} + d_{s}, i \geq 1 \), where \( d_{s} \) is a delay bound for the packets of session \( s \), and packets are eligible for transmission upon arrival. The eligible packet sequence of the server satisfies the bounded appetites property for a given value of \( \theta \) if and only if

\[
\sum_{s \in \Omega} \left( \left\lceil \frac{\tau - d_{s}}{T_{s}} \right\rceil L_{\text{max},s} \right) + \theta \leq C \tau, \quad \forall \tau \geq d_{\min}
\]

(36)

where \( T_{s} \) is the minimum packet interarrival time of session \( s, d_{\min} = \min \{ d_{s} : s \text{ is a real-time session in } \Omega \} \), and \( [x]^{+} = n \) for \( n - 1 \leq x < n, n = 1,2, \ldots \), and \( [x]^{+} = 0 \) for \( x < 0 \).
For $\theta \leq 0$, we can state (36) for $\forall \tau \geq 0$ since
\[
[\frac{\tau - d_s}{T_s}]^+ = 0, \quad \text{for } 0 \leq \tau < d_{\min}.
\]

**Proof of Theorem 11:** We first prove that (36) implies that the eligible packet sequence of the server satisfies the bounded appetites property for the given value of $\theta$. Suppose that at least one packet arrived, and its deadline is in the interval of time $[a, b]$. Thus, $b - a \geq d_{\min}$. As pointed out in [15], in any interval of time $[a, b]$, at most $[\frac{b - a - d_s}{T_s}]^+$ packets from session $s$ can arrive and have deadlines in $[a, b]$. Thus,
\[
\sum_{s \in \Omega} Z_{\max,s}(a, b) = \sum_{s \in \Omega} \left( \frac{b - a - d_s}{T_s} \right)^+ L_{\max,s}
\]
where $Z_{\max,s}(a, b)$ is the maximum value that $Z_s(a, b)$ can achieve in the interval of time $[a, b]$. This implies that
\[
\sum_{s \in \Omega} Z_s(a, b) + \theta \leq \sum_{s \in \Omega} \left( \frac{b - a - d_s}{T_s} \right)^+ L_{\max,s} + \theta \tag{37}
\]

Inequalities (37) and (36) with $\tau = b - a \geq d_{\min}$ imply that the eligible packet sequence of the server satisfies the bounded appetites property for the given value of $\theta$ (note that $\sum_{s \in \Omega} Z_s(a, b) > 0$ since, by definition, at least one packet arrived, and its deadline is in the interval of time $[a, b]$).

We now prove that if the eligible packet sequence of the server satisfies the bounded appetites property for a given value of $\theta$, then (36) is satisfied. Suppose that at least one packet arrived, and its deadline is in the interval of time $[0, \tau]$. Thus, $\tau \geq d_{\min}$ and $\sum_{s \in \Omega} Z_s(0, \tau) > 0$. From (2), with $a = 0$ and $b = \tau$,
\[
\sum_{s \in \Omega} Z_s(0, \tau) + \theta \leq C\tau, \quad \forall \tau \geq d_{\min}.
\]
But this result must be valid even in the worst case. Thus,
\[
\sum_{s \in \Omega} \left( \frac{\tau - d_s}{T_s} \right)^+ L_{\max,s} + \theta \leq C\tau, \quad \forall \tau \geq d_{\min}
\]
since at most $[\frac{\tau - d_s}{T_s}]^+$ packets from session $s$ can arrive and have deadlines in $[0, \tau]$. This completes the proof of Theorem 11.

With the result of Theorem 11, we can derive the results developed in [15]. The result for a preemptive server ([15, Theorem 1], which is equivalent to inequality (36) with $\theta = 0$) is obtained from Theorem 11 with $\theta = 0$ and Theorem 9 with $\theta = 0$. The result for a non-preemptive server ([15, Theorem 6], which is equivalent to inequality (36) with $\theta = L_{\max,\Omega}$, where $L_{\max,\Omega}$ is the maximum packet length among the sessions in $\Omega$) is obtained from Theorem 11 with $\theta = L_{\max,\Omega}$ and Theorem 5 with $\theta = L_{\max,\Omega}$. Note that, in [15], $C_i = L_{\max,s,i}/C$ and $C_p = L_{\max,\Omega}/C$.

**E. Real-Time Task Scheduling**

Real-time task scheduling was studied in [9] for the case of tasks with constant interrequest time and execution deadline equal to the next request time. The results that we present here for preemptive servers can be used for the general task scheduling problem, i.e., not restricted to the assumptions in [9]. We demonstrated that our results subsume those in [15], which in turn showed that its results for preemptive servers subsume the ones in [9]. We now translate our results to meaningful terms in the task scheduling realm. The server and its capacity represent the processor and its rate, respectively. A packet represents a task. A session represents a process, which generates tasks. The eligibility time of a packet represents the request time of a task. The length of a packet divided by the capacity of the server represents the run time of a task, i.e., the time taken by the processor to execute the task without interruption. The deadline of a packet represents the execution deadline of a task. Thus, from Theorem 9 and (2) with $\theta = 0$, the necessary and sufficient schedulability condition for task scheduling is
\[
\sum_{s \in \Omega} \frac{Z_s(a, b)}{b - a} \leq 1
\]
for any interval of time $[a, b]$, $b > a$, where $\Omega$ is the set of processes sharing the processor and $Z_s(a, b)$ is the sum of the run times of all the individual tasks of a process $s$ for which both the request time of the task and the execution deadline of the task are in $[a, b]$. Note that a process is allowed to exhibit a different run time for each new task. Furthermore, the execution deadlines of the tasks of a process do not need to be a constant delay from the tasks’ request times.

**VI. Conclusions**

We determined a simple schedulability condition for preemptive and non-preemptive deadline-ordered service disciplines. Our results do not impose any constraints on how the deadlines and the eligibility times of packets are calculated. This schedulability condition is necessary and sufficient for preemptive deadline-ordered service disciplines and, for a server that allows the presence of non-real-time packets, it is also necessary and sufficient for non-preemptive deadline-ordered service disciplines.

We also addressed the schedulability problem for service disciplines in general (i.e., not necessarily deadline-ordered), and showed the optimality of deadline-ordered service disciplines.

We showed that our results can greatly simplify the analysis of schedulability conditions of deadline-ordered service disciplines. This was demonstrated by proving the known schedulability conditions of VirtualClock, PGPS, Stop-and-Go, and Delay-EDD, and by proving the previously unknown necessary schedulability condition of VirtualClock. We also show that our results for preemptive servers are applicable to the problem of task scheduling in a real-time environment.
REFERENCES


Norival R. Figueira received the electronics engineering degree and the M.Sc. degree in systems engineering and computation from the Federal University of Rio de Janeiro, Brazil, in 1984 and 1988, respectively, and the M.Sc. and the Ph.D. degrees in computer science from the University of California at San Diego, La Jolla, in 1993 and 1995, respectively.

He was involved in research as a member of the Technical Staff of NCE/UFRJ (Computing Center of the Federal University of Rio de Janeiro) from 1982 to 1991. Since 1995, he has been a Postgraduate Researcher in the Department of Computer Science and Engineering, University of California at San Diego. His current research interests are in the area of real-time communications in high-speed (wired and wireless) networks.

Joseph Pasquale (S’83–M’87) received the B.S. and M.S. degrees from the Massachusetts Institute of Technology, Cambridge, in 1982, and the Ph.D. degree from the University of California, Berkeley, in 1988, all in computer science.

He is a Professor in the Department of Computer Science and Engineering at the University of California at San Diego, La Jolla. His research interests are in network and operating system software design. He is also interested in issues of coordination and decentralized control in large distributed systems.

Dr. Pasquale was an NSF Presidential Young Investigator during 1989–1994. In 1991, he received the IBM Faculty Development Award and the TRW Young Investigator Award.