

August 1997

UILU-ENG-97-2220

University of Illinois at Urbana-Champaign

On Causal Scheduling of Multiclass Traffic with Deadlines

Bruce Hajek and Pierre Seri

Coordinated Science Laboratory
1308 West Main Street, Urbana, IL 61801

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comment regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 1997	3. REPORT TYPE AND DATES COVERED Technical Report
4. TITLE AND SUBTITLE ON CAUSAL SCHEDULING OF MULTICLASS TRAFFIC WITH DEADLINES			5. FUNDING NUMBERS
6. AUTHOR(S)			
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES) Coordinated Science Laboratory University of Illinois at Urbana-Champaign 1308 W. Main St. Urbana, IL 61801			8. PERFORMING ORGANIZATION REPORT NUMBER UILU-ENG-97-2220
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211			10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12 b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 words) Causal scheduling-dropping policies for multiclass traffic with deadlines are considered. Packets with deadlines and class labels arrive in discrete time and are either scheduled by their deadlines or dropped on or possibly before their deadlines. Emphasis is on the case that there are two classes of traffic, and on causal policies which at any given time base decisions on arrivals up to that time. First the causal policies that maximize total throughput are identified. Then those policies with the greatest throughput for the high priority class, subject to being causal and maximizing total throughput, are identified.			
14. SUBJECT TERMS priority scheduling, deadlines, multiclass service			15. NUMBER OF PAGES 32
			16. PRICE CODE
17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only (Leave blank)

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	PR - Project
G - Grant	TA - Task
PE - Program Element	WU - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as; prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NORFON, REL, ITAR).

DOD - See DoDD 4230.25, "Distribution Statements on Technical Documents."
DOE - See authorities.
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank
DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports
NASA - Leave blank.
NTIS - Leave blank.

Block 13. Abstract. Include a brief (*Maximum 200 words*) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (*NTIS only*).

Block 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

ON CAUSAL SCHEDULING OF MULTICLASS TRAFFIC WITH DEADLINES

Bruce Hajek and Pierre Seri

University of Illinois at Urbana Champaign

{b-hajek,p-seri}@uiuc.edu

Abstract

Causal scheduling-dropping policies for multiclass traffic with deadlines are considered. Packets with deadlines and class labels arrive in discrete time and are either scheduled by their deadlines or dropped on or possibly before their deadlines. Emphasis is on the case that there are two classes of traffic, and on causal policies which at any given time base decisions on arrivals up to that time. First the causal policies that maximize total throughput are identified. Then those policies with the greatest throughput for the high priority class, subject to being causal and maximizing total throughput, are identified.

1 INTRODUCTION

The basic multiplexing problem, in which several streams of packets are to be merged, arises in many applications, in both high speed networks and wireless communication networks. Multiple classes of traffic may be present, with some having higher priority than others. In some situations, packets arrive with deadlines after which they will expire unless they are already scheduled for the merged stream. The focus of this paper is the scheduling of traffic with multiple priorities and deadlines into a single merged stream. In addition, policies for dropping some packets strictly before their deadline are considered. Two motivations for such early dropping are to reduce the buffer space required, and to enable negative acknowledgments to be issued earlier (but we don't explicitly consider the effects of negative acknowledgments on arrival streams).

A discrete time formulation is adopted. Scheduling policies are desired which maximize the

throughput of each class of traffic, with particular emphasis on the higher priority classes. For the bulk of this paper we assume that there are only two classes of traffic: class 1, typically with high priority, and class 2, typically with low priority. Implications for systems with more than two priority classes are discussed.

Typically it is not possible to schedule packets in order to simultaneously maximize both the throughput for the high priority class and the total throughput. For example, suppose that in a time slot there are two packets available for scheduling, one of class 1 and one of class 2. Suppose the class 1 packet can wait a slot, but the class 2 packet cannot. In order to maximize the throughput of the high priority traffic, the class 1 packet must be scheduled. This is so because if another class 1 packet arrives in the next time slot that must be scheduled immediately, then both class 1 packets can be accommodated. On the other hand, in order to maximize the total throughput, the class 2 packet must be scheduled. This is so to insure that packets are scheduled in both the current and next slot, even if no more packets arrive in the next slot.

To better understand the tradeoff, consider the following simulation example. The simulation covers 100,000 time slots. Packets of class 1 and class 2 each arrive according to Poisson processes with intensity 0.5. The laxity of each packet upon arrival is uniformly distributed over $\{1, 2, 3\}$. The number of packets of each class that were scheduled by their deadline for five different scheduling policies is pictured in Figure 1.

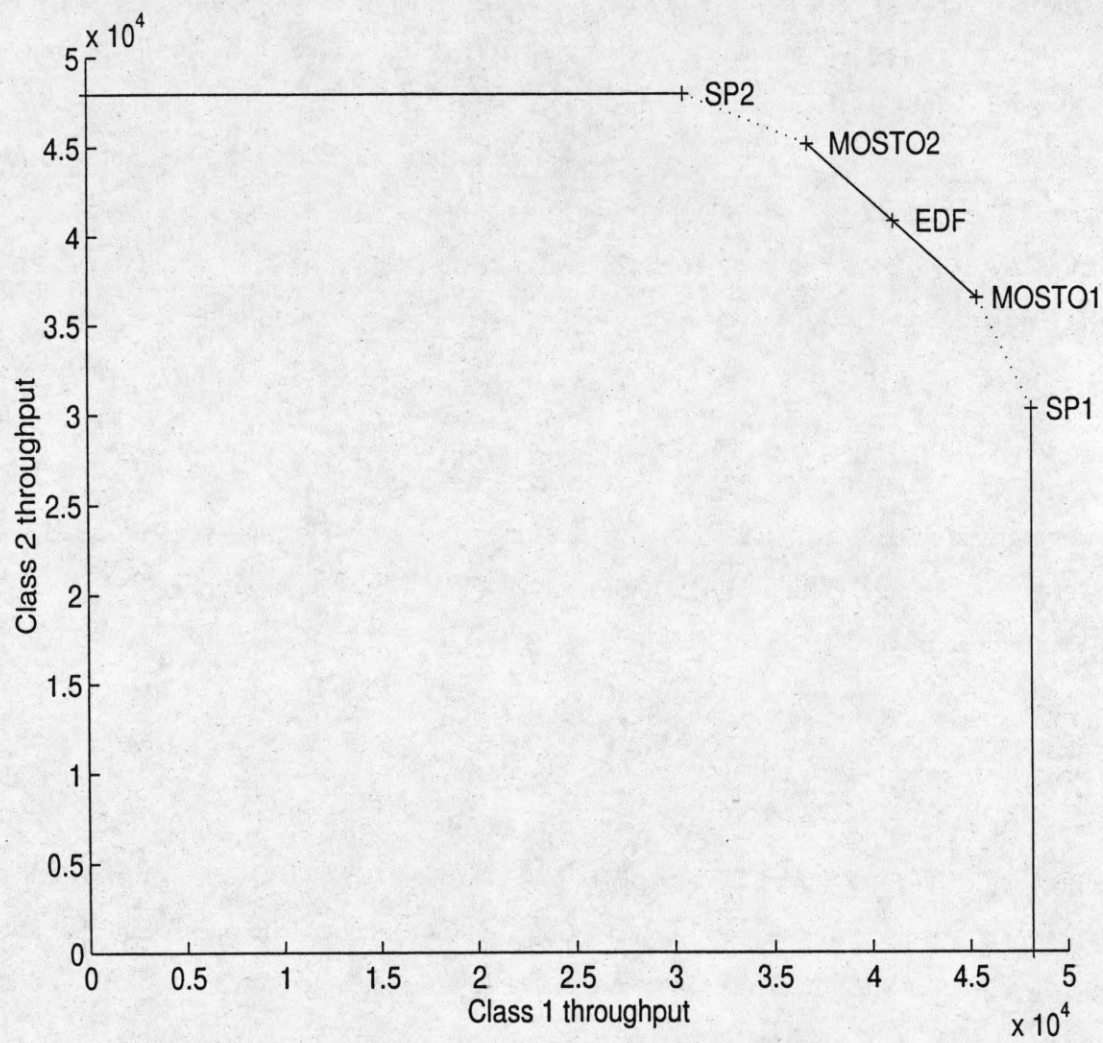


Figure 1: A sample throughput region for two classes of traffic

Point $SP1=(48163,30310)$ in the figure corresponds to the throughput pair achieved by a causal “static priority” scheduling policy that maximizes the throughput of class 1 packets, and, subject to that, maximizes the throughput of class 2 packets. Point $SP2$ is similar, with the priorities swapped. Point $EDF=(41131,40824)$ is the throughput pair achieved by the earliest deadline first policy, which in each slot schedules a minimum laxity packet, (in case of a tie packets are ordered randomly, independently of class). The policy EDF is well-known to maximize the total throughput, so that at most $41131+40824=81955$ packets can make their deadlines under any schedule for the given arrival sequence. In the terminology to be introduced later, EDF is a throughput optimal (TO) policy, and it is causal (or “real-time”) in that scheduling decisions are not based on future arrivals. The EDF policy is not the only one to maximize total throughput. Point $MOSTO1=(45429,36526)$ is the throughput pair achieved by a policy that maximizes the throughput of class one packets, subject to being a causal TO policy. That is, it maximizes throughput for one class subject to throughput optimality (MOSTO). Point $MOSTO2$ is the throughput pair achieved by a MOSTO policy when the priorities are swapped. Note that the solid lines in the figure, namely the line segment from $MOSTO1$ to $MOSTO2$, as well as the vertical line segment ending at $SP1$ and the horizontal line segment ending at $SP2$, are on the boundary of the throughput pairs achieved by all possible policies.

The emphasis of this paper is on *causal* policies, but we comment here briefly on non-causal policies, which have been studied previously by several authors. If an arrival sequence is known in advance for slots $1, \dots, N$, then a single schedule can be found to maximize both the total throughput for the N slots and the number of class 1 packets scheduled in the N slots [1]. That is, in the context of Figure 1, the throughput region for non-causal scheduling policies is obtained by extending the solid lines until they intersect. Indeed, any schedule corresponds to a matching on the bipartite graph with packets as one set of nodes and the slots $1, \dots, N$ as the other set of nodes. There is an edge between a packet and a slot k if k is included in the interval beginning with the arrival slot and ending with the deadline, of the packet. The well known labeling algorithm (see [2]) for finding bipartite matchings can be used to first schedule a set of class 1 packets with maximum possibly cardinality, and then augment that set with class 2 packets to obtain a scheduleable subset with the maximum cardinality. The paper [3] provides an algorithm which exploits the special

structure of the matching problem to provide the (non-causal) maximum weighted throughput schedule for n packets using $O(n^2)$ computations.

The basic goal of this paper is to identify the set of all causal TO scheduling policies, and the set of all MOSTO policies. In addition to scheduling policies, we also consider scheduling-dropping policies which drop packets early. To see why it is possible to drop a packet strictly before its deadline without loss of throughput, simply think of a case that five new packets with the same class arrive with common laxity three, so at most three of the packets can ever be scheduled. Then clearly two of the packets can be dropped immediately. In this paper we characterize the causal TO and the MOSTO scheduling-dropping policies. As an application of our results, we identify those causal TO and the MOSTO scheduling-dropping policies which minimize (in a sample path sense) the number of packets that are carried over from slot to slot.

The exploration of *all* the causal TO policies is a natural step in identifying a MOSTO policy. However, another motivation for exploring *all* the causal TO policies, and also for exploring *all* the MOSTO policies, is that in different settings, some of the policies may be easier to implement than others, or may have additional properties that are desirable in different situations. For example, the policy S-OPT of [4] is a causal TO scheduling-dropping policy which either immediately drops, or eventually schedules, each packet. That is, the fate of each packet is determined upon arrival. We explore this concept for multiclass scheduling as well.

The optimization philosophy underlying the definition of the MOSTO points and the static priority points discussed above is *optimization with hierarchical objectives*. That is, a second quantity is maximized, subject to the condition that a first quantity has to be maximized. A standard alternative philosophy for dealing with multiple objectives is to use weights. For example, suppose class i packets have weight w_i , where $w_1 > w_2$. The objective for the finite simulation could be to maximize the weighted throughput $n_1w_1 + n_2w_2$, where n_i is the number of class i packets scheduled. For example, the value for point SP1 is $48163w_1 + 30310w_2$ while the value for point MOSTO1 is $45429w_1 + 36526w_2$. Which of these points has higher weighted throughput depends on the numerical values of the weights. Clearly, in order for causal policies to maximize a weighted throughput, the numerical values of the weights (not just the order of the weights) and possible side information about the arrival sequence (such as statistical predictions) must be taken into account.

In contrast, if noncausal policies are considered, then as mentioned above there exist policies that maximize both the total throughput and the throughput for a specified class of traffic. Such a policy maximizes the weighted throughput for any weight pair (w_1, w_2) such that the specified class has weight at least as large as the weight for the other class.

If more than two classes of traffic are considered then there is a large variety of heirarchical scheduling objectives. For example, for three classes one could maximize the class 1 throughput subject to maximizing the class 1 plus class 2 throughput, all subject to maximizing the total throughput. That is perhaps the most natural extension of the MOSTO philosophy. Alternatively, one could contemplate maximizing the class 2 throughput subject to maximizing the type 1 throughput, all subject to maximizing the total throughput. This represents a mixture of the MOSTO and static priority criteria.

There is an extensive literature on scheduling packets with deadlines within the computer science literature. However, almost all of the literature concerns situations in which suitable admission control is in place so that all deadlines can be met.

The results in this paper involve arbitrary arrival sequences, rather than random sequences. It is a challenging and still largely open problem to compute throughput regions and average delays under particular random models for arrivals, though many interesting examples exist (see for example [6]).

The terminology and precise results of the paper are presented in the next section, along with simulation results which illustrate the points. Proofs of the results are not included here, but may be found in the full version of the paper posted on the web (URL <http://tesla.csl.uiuc.edu/~hajek/>).

2 STATEMENT OF THE RESULTS

2.1 Notation

Each packet takes one slot to be scheduled, and the deadline $d(p)$ of a packet p is the last slot in which it can be scheduled. An arrival sequence \mathcal{A} is a sequence $\mathcal{A} = (A_n : n \geq 1)$ such that A_n is

the set of packets arriving in slot n . The sets A_n are assumed to be mutually disjoint, and $d(p) \geq n$ for all $p \in A_n$.

A *schedule* for an arrival sequence \mathcal{A} is a sequence $(p_n : n \geq 1)$ such that for each n either $p_n = \Delta$ (indicating that no packet is scheduled in slot n), or $p_n \in A_1 \cup \dots \cup A_n$ with $n \leq d(p_n)$. Also, no packet appears in a schedule more than once. A *scheduling policy* π gives a schedule $(\pi(\mathcal{A}, n) : n \geq 1)$ for each arrival sequence \mathcal{A} . A policy π is *causal* if $\pi(\mathcal{A}; n)$ depends only on A_1, \dots, A_n .

The following sets can be defined, given an arrival sequence \mathcal{A} and schedule $(p_n : n \geq 1)$:

R_n , for $n \geq 0$, is the set of packets remaining in the system at the end of slot n .

S_n , for $n \geq 1$, is the set of packets available for scheduling in slot n .

E_n , for $n \geq 1$, is the set of packets that expire at the end of slot n without being scheduled.

The following evolution equations hold. It is assumed that $R_0 = \emptyset$. For $n \geq 1$,

$$\begin{aligned} S_n &= R_{n-1} \cup A_n \\ E_n &= \{p \in S_n - p_n : d(p) = n\} \\ R_n &= S_n - E_n - p_n \end{aligned}$$

A scheduling policy π and an arrival sequence \mathcal{A} determine the sets $(S_n \geq 1)$. Sometimes we explicitly denote the dependence of S_n on π by writing $S_n(\pi)$ for S_n .

The *laxity* of a packet p , given the "current slot" is k , is $d(p) - k + 1$. Typically the laxity of a packet under consideration is greater than or equal to one, since a packet is expired before the current slot if its laxity is 0 or smaller. Note that a packet with laxity $l \geq 1$ which has not been scheduled before the current slot k can be scheduled in any of the l slots $k, \dots, k + l - 1$. For example, if the current slot is 9, then a packet p with deadline $d(p) = 12$ has laxity 4 and can be scheduled in any of the four slots 9, 10, 11, 12. For convenience we sometimes discuss a set of packets with laxities, without explicit mention of the current slot.

2.2 Throughput optimal scheduling policies

A policy π is said to be *throughput optimal* (TO) if, for any arrival sequence \mathcal{A} and any $n \geq 1$, policy π schedules at least as many packets in slots $\{1, \dots, n\}$ as any other policy. As stated in the introduction, we seek to characterize the set of *all* causal TO policies.

Let S be a set of packets with laxities. Define $\Phi_s(S)$ to be a subset of S defined as follows. (The subscript "s" denotes "single class.") Write $S = \{p_1, \dots, p_k\}$, ordered so that $l(p_1) \leq l(p_2) \leq \dots \leq l(p_k)$. Define the following quantities in turn:

$$\delta(i) = l(p_i) - i \quad \text{for } 1 \leq i \leq k \quad (2.1)$$

$$\delta_{\min} = \min\{\delta(i) : 1 \leq i \leq k\} \quad (2.2)$$

$$i^* = \min\{i : \delta(i) = \delta_{\min}\} \quad (2.3)$$

$$\Phi_s(S) = \{p_1, \dots, p_{i^*}\} \quad (2.4)$$

$$l^* = l(p_{i^*}) \quad (2.5)$$

Some examples of this definition are illustrated in Figure 2.

	Example 1	Example 2	Example 3
S	1 2 5 5 5	3 3 6	1 2 2 4 4 4 6
Positions i	1 2 3 4 5	1 2 3	1 2 3 4 5 6 7
$\delta(i)$	0 0 2 1 0	2 1 3	0 0 -1 0 -1 -2 -1
$\Phi_s(S)$	1	3 3	1 2 2 4 4 4
i^*, l^*	$i^* = 1, l^* = 1$	$i^* = 2, l^* = 3$	$i^* = 6, l^* = 4$

Figure 2: Illustration of the computation of $\Phi_s(S)$.

Note that $|\Phi_s(S)| = i^*$, that l^* is the largest laxity of packets in $\Phi_s(S)$, and that all packets in $S - \Phi_s(S)$ have laxities strictly greater than l^* . Even though the ordering of packets in S is not unique (since some packets may have the same laxity) the set $\Phi_s(S)$ does not depend on the ordering and is thus a well defined subset of S .

Theorem 2.1 *A causal policy π is TO if and only if for any arrival sequence A and all k such that $S_k \neq \emptyset$, $\pi(A; k) \in \Phi_s(S_k(\pi))$.*

We call $\Phi_s(S_k)$ the *no regret scheduling set* of S_k , because given S_k , a packet p can be scheduled in slot k without the possible loss of future throughput if and only if $p \in \Phi_s(S_k)$.

2.3 Greedy algorithms for schedulable subsets

This section describes some basic results that are required to state and appreciate the results for scheduling-dropping policies. Let A be a set of packets with laxities and let τ be a finite subset of $\{1, 2, \dots\}$. (In this paper, the symbol τ will always denote a subset of $\{1, 2, \dots\}$.) Think of τ as indexing a set of slots, relative to the “current” slot: $t \in \tau$ corresponds to slot $k + t - 1$, where k is the “current” slot.) Say that A *can cover* τ if there is an assignment of a subset (possibly all) packets in A to elements in τ such that all elements in τ are covered and such that if a packet p is assigned to t then the laxity of p is at least t .

A set of packets S with laxities is said to be *schedulable* if it covers $\{1, \dots, |S|\}$. Equivalently, if the packets in S are ordered according to increasing laxities, then S is schedulable if the laxity of the i th packet is at least i , for $1 \leq i \leq |S|$. Of course if S can cover some τ with $|\tau| = |S|$, then S is schedulable. Given a set S of packets with laxities, define $rank(S)$ to be the largest cardinality of schedulable subsets of S . Given a set of packets with laxities, a simple greedy algorithm, (Algorithm G1 in Figure 3) can be used to compute a maximum cardinality schedulable subset of S , and hence also $rank(S)$. A key feature of the algorithm is that the packets can be considered in arbitrary order. The correctness of Algorithm G1, verified in Section 4, implies that the set of sets of packets together with the rank function constitutes a matroid [2]. We next present Algorithm G2, which essentially includes Algorithm G1 within it, while having comparable complexity.

Of all the sets that S can cover, there is a “largest and latest” one, which we denote by $\mathcal{L}(S)$. Formally, if S can cover a set τ , then $\mathcal{L}(S)$ dominates τ in the following sense: $|\tau| \leq |\mathcal{L}(S)|$, and the i th largest element of \mathcal{L} is greater than or equal to the i th largest element of τ , for $1 \leq i \leq |\tau|$. Algorithm G2, given in Figure 3, simultaneously finds $\mathcal{L}(S)$ and a maximum cardinality schedulable

subset of S . A proof of correctness of Algorithm G2 is given in Section 4.

Algorithm G1

Input: $S = \{p_1, \dots, p_{|S|}\}$ (arbitrary order) with deadlines $l(p_1), \dots, l(p_{|S|})$.

Output: A maximum cardinality scheduleable subset of S

Set $A_0 = \emptyset$;

do for $j = 1$ to $|S|$:

 if $A_{j-1} + p_j$ is scheduleable let $A_j = A_{j-1} + p_j$;

 else let $A_j = A_{j-1}$;

end

return ($A_{|S|}$);

Algorithm G2

Input: $S = \{p_1, \dots, p_{|S|}\}$ (arbitrary order) with deadlines $l(p_1), \dots, l(p_{|S|})$.

Output: A maximum cardinality scheduleable subset of S , and $\mathcal{L}(S)$.

Set $A_0 = \emptyset$ and $\tau_0 = \emptyset$;

do for $j = 1$ to $|S|$:

 if $\{1, \dots, l(p_j)\} \subset \tau_{j-1}$ let $A_j = A_{j-1}$ and $\tau_j = \tau_{j-1}$;

 else let $A_j = A_{j-1} + p$ and $\tau_j = \tau_{j-1} + a_j$, where $a_j = \max(\{1, \dots, l(p_j)\} - \tau_{j-1})$;

end

return ($A_{|S|}, \tau_{|S|}$);

Figure 3: Algorithms G1 and G2.

Let A and B be two sets of packets with laxities. Write $A \succeq B$ (or $B \preceq A$) if A can cover every set τ that B can cover, and write $A \equiv B$ if $A \preceq B$ and $B \preceq A$.

Lemma 2.1 *Let S be a set of packets with laxities and let $Q \subset S$. Then $Q \equiv S$ if and only if $r(Q) = r(S)$. In particular, Q minimizes $|Q|$ subject to $[Q \subset S \text{ and } Q \equiv S]$ if and only if Q is a maximum cardinality scheduleable subset of S .*

Proof. If $r(Q) < r(S)$ then Q can't, but S can, cover $\{1, \dots, |S|\}$, so $Q \not\equiv S$. If $r(Q) = r(S)$,

order the packets of S so that all packets of Q precede the other packets of S and apply algorithm B. The set $A_{|S|}$ produced is also equal to $A_{|Q|}$. It is a subset of both Q and S and $\mathcal{L}(Q) = \mathcal{L}(A) = \mathcal{L}(S)$. Therefore $Q \equiv S$. The second statement of the lemma is an immediate consequence of the first. \square

2.4 Throughput optimal scheduling-dropping policies

Policies which not only schedule packets, but which also drop some packets without scheduling, are considered next. Such policies are called *scheduling-dropping* policies. Given the set of packets S_n available for scheduling in a slot n , a scheduling-dropping policy specifies a set $Q_n \subset S_n$ of packets to be retained, as well as a packet (if any) $\pi(\mathcal{A}; n)$ to be scheduled in slot n . Packets in $S_n - Q_n$ are simply dropped from the system without ever being scheduled. By convention, we agree that the set Q_n includes the packet to be scheduled in slot n , if any. A scheduling-dropping policy π is *causal* if both $\pi(\mathcal{A}, n)$ and Q_n depend only on A_1, \dots, A_n . Given an arrival sequence \mathcal{A} and a scheduling-dropping policy π , the following evolution equations hold. The interpretations of the sets R_n , S_n and E_n are as before. Assume that $R_0 = \emptyset$ and write p_n for $\pi(\mathcal{A}; n)$. Then for $n \geq 1$,

$$\begin{aligned} S_n &= R_{n-1} \cup A_n \\ Q_n &\subset S_n \quad (Q_n \text{ is specified by the policy}) \\ E_n &= \{p \in Q_n - p_n : d(p) = n\} \\ R_n &= S_n - Q_n - p_n \end{aligned}$$

We sometimes write $Q_n(\pi)$ for Q_n to make the dependence of Q_n on π explicit. A scheduling-dropping policy π is said to be *throughput optimal* (TO) if, for any arrival sequence \mathcal{A} and any $n \geq 1$, policy π schedules at least as many packets in slots $\{1, \dots, n\}$ as any other policy. Of course, any TO scheduling policy (with no dropping) can be viewed as a TO scheduling-dropping policy (with $Q_n = S_n$). However, for many arrival sequences it is possible to drop packets without loss of throughput. The theorem below characterizes the set of *all* causal TO scheduling-dropping policies.

Theorem 2.2 *A causal scheduling-dropping policy π is TO if and only if for any arrival sequence \mathcal{A} and all k , $Q_k \equiv S_k$, and if $S_k \neq \emptyset$ then $\pi(\mathcal{A}, k) \in \Phi_s(Q_k)$. Furthermore, if π and π' are both causal TO scheduling-dropping policies, $Q_k(\pi) \equiv Q_k(\pi')$ for all k .*

Theorem 2.2 allows us to identify those causal TO scheduling-dropping policies which minimize, for all slots, the number of packets carried over from one slot to the next. Indeed, by the second

statement of the theorem, $r(Q_k)$ is the same for all TO scheduling-dropping policies. That is, $r(Q_k) = r^*(\mathcal{A}, k)$, where $r^*(\mathcal{A}, k)$ depends only on \mathcal{A} and k . Therefore, $|Q_k| \geq r^*(\mathcal{A}, k)$ for any causal TO policy π , and by selecting Q_k to be a maximum cardinality scheduleable subset of S_k (for example by applying Algorithm G1 to S_k), one achieves $Q_k = r^*(\mathcal{A}, k)$. We thus have the following corollary to Theorem 2.2.

Corollary 2.1 *For any arrival sequence \mathcal{A} and any $k \geq 1$, a causal TO scheduling-dropping policy π minimizes $|Q_k|$ over all such policies if and only if Q_k is scheduleable.*

Figure 4 pictures the average queue size (time average of Q_k) for various mean arrival rates, assuming laxities on arrival are uniformly distributed on $\{1, \dots, 9\}$ (so the mean laxity on arrival is 5), observed for simulation over 10,000 time slots. The upper curve is generated by the Earliest Deadline First policy (without early dropping) and the lower curve is for an arbitrary causal TO policy for which Q_k is scheduleable for all k , such as policy S-opt of [4, 5]. For small values of λ , few packets are dropped and both curves follow the mean queue size for an $M/D/1$ queueing system. For large values of λ the upper curve grows roughly linearly with slope 5 while the lower curve asymptotically approaches 10.

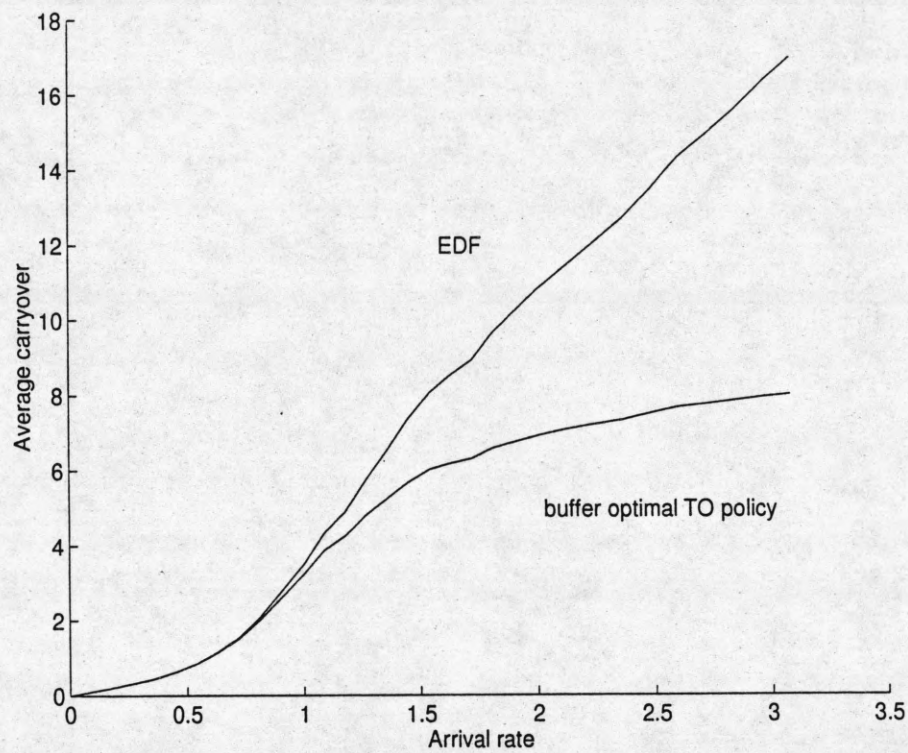


Figure 4: Mean queue size for EDF without early dropping and for an arbitrary buffer optimal TO scheduling-dropping policy.

2.5 MOSTO scheduling policies

Suppose now that each packet is either of class 1 (high priority) or class 2 (low priority). A causal policy π is said to be *maximum class one subject to throughput optimal* (MOSTO) if π is TO and if for any arrival sequence \mathcal{A} and any $n \geq 1$, policy π schedules at least as many class 1 packets in slots $\{1, \dots, n\}$ as any other causal TO policy. It is not immediately clear from the definition that MOSTO policies actually exist, but they do and they are characterized by the theorem below.

Given a set of packets S , let \overline{S} denote the set of class 1 packets in S , and let \underline{S} denote the set of class 2 packets in S .

Let $\Phi_m(S)$ be the subset of S defined as follows:

$$\Phi_m(S) = \begin{cases} \Phi_s(S) \cap \Phi_s(\overline{S}) & \text{if } \overline{\Phi_s(S)} \neq \emptyset, \\ \Phi_s(S) = \Phi_s(\underline{S}) & \text{if } \overline{\Phi_s(S)} = \emptyset \text{ and } S \neq \emptyset, \\ \emptyset & \text{if } S = \emptyset. \end{cases} \quad (2.6)$$

Φ_m is the multiclass equivalent of Φ_s . It can be shown that the above definition of $\Phi_m(S)$ is equivalent to the definition obtained if $\Phi_s(S) \cap \Phi_s(\overline{S})$ is replaced in the first line of (2.6) by $\Phi_s(\overline{S})$.

Theorem 2.3 *A causal scheduling policy π is MOSTO if and only if for any arrival sequence \mathcal{A} and all k such that $S_k \neq \emptyset$, $\pi(\mathcal{A}; k) \in \Phi_m(S_k(\pi))$.*

Examples of the computation of $\Phi_m(S)$ are given in Figure 5. Overlines indicate class 1 packets and underlines class 2 packets.

2.6 MOSTO scheduling-dropping policies

A causal scheduling-dropping policy π is said to be *maximum class one subject to throughput optimal* (MOSTO) if, for any arrival sequence \mathcal{A} and any $n \geq 1$, policy π schedules at least as many class one packets in slots $\{1, \dots, n\}$ as any other TO policy.

Given a set of packets S , define $\kappa(S)$ by $\kappa(S) = \mathcal{L}(C)$, where C is the set of $\text{rank}(S) - \text{rank}(\overline{S})$ packets in \underline{S} with largest laxity. Given two sets of packets S and T , write $S \succ T$ if $S \equiv T$, $\overline{S} \equiv \overline{T}$

	Example 1	Example 2	Example 3
S	$\underline{1} \ \bar{2} \ \underline{5} \ \bar{5} \ \bar{5}$	$\bar{3} \ \underline{3} \ \underline{6}$	$\underline{1} \ \bar{2} \ \underline{2} \ \bar{4} \ \underline{4} \ \underline{4} \ \bar{6}$
$\Phi_s(S)$	$\underline{1}$	$\bar{3} \ \underline{3}$	$\underline{1} \ \bar{2} \ \underline{2} \ \bar{4} \ \underline{4} \ \underline{4}$
$\Phi_s(\bar{S})$	irrelevant	$\bar{3}$	$\bar{2}$
$\Phi_m(S)$	$\underline{1}$	$\bar{3}$	$\bar{2}$

Figure 5: Illustration of the computation of $\Phi_m(S)$.

and $\kappa(S) = \kappa(T)$. It can be shown that the first condition, $S \equiv T$, is acutally implied by the other two conditions in the definition of $S \asymp T$.

Theorem 2.4 *A causal scheduling-dropping policy π is MOSTO if and only if for any arrival sequence A and all k , $Q_k \asymp S_k$, and if $Q_k \neq \emptyset$ then $\pi(A, k) \in \Phi_m(Q_k)$. Furthermore, if π and π' are both MOSTO scheduling-dropping policies, then $Q_k(\pi) \asymp Q_k(\pi')$ for all k .*

For example, if $S_k \sim (\underline{1}, \bar{2}, \bar{3}, \underline{3})$, then the packet corresponding to $\underline{1}$ can be dropped early, leading to $Q_k \sim (\bar{2}, \bar{3}, \underline{3})$. The packet $\bar{2}$ would be scheduled leading to the carryover set $R_k \sim (\bar{3}, \underline{3})$. Note that if instead the packet $\underline{3}$ is dropped early, then $Q_k \sim (\underline{1}, \bar{2}, \bar{3})$. This would force the low priority packet with laxity one (corresponding to $\underline{1}$) to be scheduled in the current slot k , and $R_k \sim (\bar{2}, \bar{3})$. This alternative is not MOSTO since it schedules a class 2 packet when a class 1 packet could have been scheduled without loss of throughput optimality.

Given a set S of packets with laxities, a scheduleable subset of S with $Q \asymp S$ is produced by Algorithm G1 applied to S with the following order of packets: the packets in \bar{S} are considered first (in any order) followed by packets in \underline{S} , in order of decreasing laxity. After all packets in \bar{S} have been considered, a maximum cardinality subset of \bar{S} is obtained. The final output of Algorithm G1 is that set together with the first $\text{rank}(S) - \text{rank}(\bar{S})$ packets of \underline{S} considered.

Therefore, Corollary 2.1 has the following extension for MOSTO policies. Note that suitable scheduling-dropping MOSTO policies yield the same queue size as the minimum possible for any causal TO policy, uniformly over all time slots.

Corollary 2.2 *For any arrival sequence A (with two classes of packets) and any $k \geq 1$, a MOSTO scheduling-dropping policy π minimizes $|Q_k|$ over all such policies if and only if Q_k is scheduleable.*

2.7 Algorithms for MOSTO Scheduling-Dropping

A conceptually very simple MOSTO scheduling-dropping algorithm is as follows. There are two phases per time step: the dropping phase and the scheduling phase. In the dropping phase, given a set S_k , a set Q_k with $Q_k \preceq S_k$ is computed using Algorithm G2 as described in the previous section. Packets in $S_k - Q_k$ are then dropped. In the scheduling phase, the set $\Phi_s(Q_k)$ is computed. If this set contains a class 1 packet, then an earliest deadline class 1 packet is scheduled. Otherwise, an earliest deadline class 2 packet is scheduled.

If when Algorithm G2 is run, the new class one packets are considered after the class one packets carried over from previous slot, then each class one packet is either dropped in the slot in which it arrives, or it is eventually scheduled. In general such determination of fate is not possible for class 2 packets for any MOSTO policy.

The computational complexity of the algorithm is as follows, assuming that packets have a maximum laxity of B , and at most n_{max} new packets can arrive in a slot. The dropping phase of the algorithm requires at most $O((B + n_{max}) \log(B + n_{max}) + B^2)$ computations per slot, where the first term is for sorting the packets and the second for executing Algorithm B. The scheduling phase of the algorithm requires complexity at most B .

Other algorithms are possible, and are based on maintaining particular data structures from slot to slot. For example, we believe that when restricted to two classes of traffic, the algorithm M-Opt of Ling and Shroff [4, 5] is a MOSTO algorithm. It requires at most $O(B^2)$ steps to incorporate each one new arrival into the data structure (the new arrival or some other packet may be dropped in the process). Updating the data structure for the passage of time can be done with $O(B)$ computations. This includes $O(1)$ time for deciding which packet to schedule.

There is much left to be understood about algorithms, especially for generalizations to more

than two classes, in which case, as explained earlier, there are many distinct possible optimization criteria. Our belief is that the definitions and characterizations of optimal causal policies given in this paper will be useful for further innovations in this area.

3 PROOFS FOR THROUGHPUT OPTIMAL POLICIES

Theorems 2.1 and 2.2 are proved in this section.

Lemma 3.1 *Suppose that $A \succeq B$, that $C \succeq D$, and that $A \cap C = B \cap D = \emptyset$. Then $A \cup C \succeq B \cup D$.*

Proof. Let τ be an arbitrary set that can be covered by $B \cup D$. Fix an assignment of packets from a subset of $B \cup D$ to τ which covers τ , and let τ_B be the subset of τ assigned to packets in B . Then τ_B can be covered by packets in A , and $\tau - \tau_B$ can be covered by packets in D , so τ can be covered by packets in $A \cup C$. \square

Lemma 3.2 *Let S be a nonempty set of packets with laxities, let p_1 denote an element of S of minimum laxity, and let $p \in \Phi_s(S)$. Then $S - p \equiv S - p_1$.*

Proof. We first establish that $\Phi_s(S) - p \equiv \Phi_s(S) - p_1$. Suppose $\Phi_s(S) - p_1$ can cover a set τ . Then τ is a subset of $\{1, \dots, l^*\}$ with cardinality at most $i^* - 1$. Since the packets $p_{i^*}, p_{i^*-1}, \dots, p_1$ in $\Phi_s(S)$ satisfy $l(p_{i^*}) = l^*$ and $l(p_{i^*-j}) \geq l^* - j + 1$ for $0 \leq j \leq i^* - 1$, it follows that $\Phi_s(S) - p$ can cover any subset of $\{1, \dots, l^*\}$ with cardinality at most $i^* - 1$. In particular, $\Phi_s(S) - p$ can cover τ . This proves that $\Phi_s(S) - p \equiv \Phi_s(S) - p_1$. The lemma then follows from Lemma 3.1 with C in the lemma taken to be $S - \Phi_s(S)$. \square

Given a set A of packets with laxities, let \tilde{A} denote the result on A of the passing of one time increment: \tilde{A} is obtained from A by decreasing the laxity of each packet in A by one, and then dropping any packet with laxity less than or equal to zero.

Lemma 3.3 *If $A \succeq B$, then $\tilde{A} \succeq \tilde{B}$.*

Proof. If \tilde{B} can cover a set τ , then B can cover the set $\tau' = \{t+1 : t \in \tau\}$. But then A can cover τ' so that \tilde{A} can cover τ . \square

Lemma 3.4 *If $A \succeq B$, if $p \in \Phi_s(A)$, and if $q \in B$ then $\widetilde{A-p} \succeq \widetilde{B-q}$.*

Proof. By Lemma 3.2, $A - p \equiv A - p_1$, where p_1 is a minimum laxity packet in A . Therefore, by Lemma 3.3, $\widetilde{A-p} \equiv \widetilde{A-p_1}$. To complete the proof it will be shown that $\widetilde{A-p_1} \succeq \widetilde{B-q}$. To that end, suppose $\widetilde{B-q}$ can cover some set $\tau = \{t_1, \dots, t_r\}$. Then, by assigning q to 1, B can cover $\{1, t_1+1, \dots, t_r+1\}$. Thus A can also cover this set, and do so such that p_1 is assigned to 1. Thus, $\widetilde{A-p_1}$ can cover τ . \square

Proof of Theorem 2.1: First the “if” half of the theorem will be proved. A policy π is said to be *non-idling* if for any k it schedules a packet in slot k whenever $S_k(\pi) \neq \emptyset$. The search for TO policies can clearly be restricted to non-idling policies. Moreover, a policy π is TO if, for any arrival sequence \mathcal{A} , policy π schedules at least as many packets in slots $\{1, \dots, n\}$ as any non-idling policy, for any n .

Assume that π is a causal policy such that for any arrival sequence \mathcal{A} and all k such that $S_k \neq \emptyset$, $\pi(\mathcal{A}; k) \in \Phi_s(S_k(\pi))$ and let π' be any non-idling policy. Lemmas 3.1, 3.4 and argument by induction on k imply that $S_k(\pi) \preceq S_k(\pi')$ for all k . In particular, π schedules a packet in any slot in which π' schedules a packet. Therefore π schedules at least as many packets in slots $\{1, \dots, n\}$ as π' , for any n . Since this is true for an arbitrary non-idling policy π' and arbitrary arrival sequence, policy π is TO, and the “if” portion of the theorem is proved.

To prove the “only if” portion of the theorem, suppose π is a causal policy such that for some arrival sequence and some slot k , it holds that $S_k \neq \emptyset$ but π does not schedule a packet in $\Phi_s(S_k)$ in slot k . Assume that π schedules some packet in slot k for otherwise π is clearly not TO. The packet scheduled must be in $B = S_k - \Phi_s(S_k)$. Let i^* and l^* be defined as in (2.4) taking $S = S_k$ with the laxities determined for current slot k . Let the future arrival sequence be defined as follows. The set A_{k+1} contains $l^* - 1$ packets with deadline $k + l^* - 1$, and $A_j = \emptyset$ for $j \geq k + 2$.

Let m denote the number of packets in B . We claim that the packets in B can be scheduled in the m slots $k+l^*, \dots, k+l^*+m-1$. Indeed, write $B = \{p_{i^*+1}, \dots, p_{i^*+m}\}$ where the packets are ordered as in the definition of $\Phi_s(S)$ (with nondecreasing laxities). Then $l(p_{i^*+j}) \geq l^* + j$ for $1 \leq j \leq m$ (where the laxities are computed relative to slot k) or equivalently, $d(p_{i^*+j}) \geq k+j+l^*-1$, so that packet p_{i^*+j} can be scheduled in slot $k+j+l^*-1$ for $1 \leq j \leq m$. This proves the claim about B .

Let π' be a policy such that

- π' makes the same scheduling decisions as π for all slots before slot k
- π' schedules a packet in $\Phi_s(S)$ in slot k
- If $l^* \geq 2$ then π' schedules one of the packets that arrived in slot $k+1$, for each slot j with $k+1 \leq j \leq k+l^*-1$.
- π' schedules the packets of B in the m slots $k+l^*, \dots, k+l^*+m-1$

Then policy π' schedules a packet in every slot j with $k \leq j \leq k+l^*+m-1$. On the other hand, since π schedules a packet of B in slot k , and since the m packets in B are the only packets that can be used to cover any of the m slots $k+l^*, \dots, k+l^*+m-1$, it follows that π must fail to schedule a packet in one of these slots. Therefore, π is not TO. The theorem is proved. \square

Proof of Theorem 2.2. First the “if” half of the theorem will be proved. Assume that π is a causal scheduling-dropping policy such that for any arrival sequence \mathcal{A} and all k , $Q_k \equiv S_k$, and if $S_k \neq \emptyset$ then $\pi(\mathcal{A}, k) \in \Phi_s(Q_k)$. and let π' be any non-idling policy that does not drop packets. Lemmas 3.1, 3.4 and argument by induction on k imply that $S_k(\pi) \preceq S_k(\pi')$ for all k . In particular, π schedules a packet in any slot in which π' schedules a packet. Therefore π schedules at least as many packets in slots $\{1, \dots, n\}$ as π' , for any n . Since this is true for an arbitrary non-idling policy π' and arbitrary arrival sequence, policy π is TO, and the “if” portion of the theorem is proved.

For the “only if” portion of the theorem, we must show that a casual scheduling-dropping policy π is not optimal if there is an arrival sequence \mathcal{A} and slot k such that either (i) $Q_k \neq S_k$ or (ii) $\pi(\mathcal{A}; k) \notin \Phi_s(Q_k) \neq \emptyset$. If (i) is true, and if there are no new arrivals after slot k , then π will

schedule strictly fewer packets than a scheduling-dropping policy π' which agrees with π for slots $1, \dots, k-1$, but which for $j \geq k$ takes $Q_j = S_j$ and schedules a packet in $\Phi_s(S_j)$, if any. If (ii) is true, then, by the same reasoning as in the proof of Theorem 2.1 π is again not TO.

It remains to prove the last statement of the Theorem. Both π and π' are non-idling and, by the first part of the theorem, π satisfies $Q_k \equiv S_k$, and if $S_k \neq \emptyset$ then $\pi(\mathcal{A}, k) \in \Phi_s(Q_k)$. Therefore by the first part of the proof, $S_k(\pi) \preceq S_k(\pi')$ for all k . The reverse inequality holds by symmetry, so $S_k(\pi) \equiv S_k(\pi')$ for all k . Therefore, $Q_k(\pi) \equiv S_k(\pi) \equiv S_k(\pi') \equiv Q_k(\pi')$ for all k , and the theorem is proved. \square

4 VERIFICATION OF GREEDY ALGORITHMS G1 and G2

The correctness of Algorithm G2, and hence also of Algorithm G1 which produces the same sets A_j , is verified as follows. The following induction hypothesis is used:

$$\mathcal{P}(j) : A_j \text{ is a maximum cardinality subset of } \{p_1, \dots, p_j\} \text{ and } \tau_j = \mathcal{L}(\{\tau_1, \dots, \tau_j\})$$

Clearly $\mathcal{P}(1)$ is true so suppose that $\mathcal{P}(j-1)$ is true for some j with $2 \leq j \leq |S|$. We prove $\mathcal{P}(j)$. Let B be any scheduleable subset of $\{p_1, \dots, p_j\}$ and let $\sigma \subset \{1, 2, \dots\}$ with $|\sigma| = |B|$ be a set B can cover. It is to be shown that τ_j dominates σ .

If $\{1, \dots, l(p_j)\} \subset \tau_{j-1}$ (so $\tau_j = \tau_{j-1}$) let $\sigma^- = \sigma \cap \{1, \dots, l(p_j)\}$ and $\sigma^+ = \sigma - \sigma^-$. Similarly, let $\tau_j^- = \tau_j \cap \{1, \dots, l(p_j)\}$ and $\tau_j^+ = \tau_j - \tau_j^-$. Since σ^+ can be covered by $B - p_j$, which is a subset of $\{1, \dots, p_{j-1}\}$, τ_j^+ dominates σ^+ . Also, τ_j^- dominates σ^- . Therefore τ_j dominates σ .

If, on the other hand, $\{1, \dots, l(p_j)\} \not\subset \tau_{j-1}$ then fix an assignment of elements of σ to elements of B . If $p_j \in B$ let \hat{a} denote the element assigned to p_j and set $\sigma_o = \sigma - \hat{a}$. Otherwise, set $\sigma_o = \sigma$. By the induction hypothesis, τ_{j-1} dominates σ_o . By the choice of a_j it follows that τ_j dominates σ .

Therefore, in either case, τ_j dominates σ , and the correctness of Algorithm G2 is verified.

5 ODDS AND ENDS

The following lemma shows that a definition of $\Phi_m(S)$ equivalent to that in (2.6) is obtained if $\Phi_s(S) \cap \Phi_s(\bar{S})$ is replaced in the first line of (2.6) by $\Phi_s(\bar{S})$.

Lemma 5.1 *If $A \subset S$ and $A \cap \Phi_s(S) \neq \emptyset$, then $\Phi_s(A) \subset \Phi_s(S)$.*

Proof. Write $S = \{p_1, \dots, p_{|S|}\}$, where the packets are ordered so that $l(p_1) \leq \dots \leq l(p_{|S|})$. Also suppose that if a packet in A has the same laxity as a packet in $S - A$, then the packet in $S - A$ precedes the packet in A in the ordering. Let $\Phi_s(S)$ be defined using (2.1)-(2.5) with this ordering of packets, and let $\Phi_s(A)$ be similarly defined, where packets in A are ordered among themselves as they are ordered in S . Let $\delta(p)$ for any $p \in S$ be defined as in (2.1) relative to the set of packets S , and let $\delta'(p)$ for any $p \in A$ be defined as in (2.1) relative to the set of packets A . It suffices to show that $p^* \in \Phi_s(S)$, where p^* is the last packet in $\Phi_s(A)$. Let j be the index so that p_j is the first packet in A and let k be the index such that $p^* = p_k$. We need to show $i^* \geq k$. Since $A \cap \Phi_s(S) \neq \emptyset$, it follows that $i^* \geq j$. Consider any i with $j \leq i \leq k$. It must be shown that $i \neq i^*$. Suppose that p_a is the last packet of A before or equal to p_i , and let $d = i - a$. Then $\delta(p_i) \geq \delta(p_a) - d$. In turn, $\delta(p_k) - \delta(p_a) \leq \delta'(p_k) - \delta'(p_a) - d < -d$ since there are at least d packets of $S - A$ in between p_a and p_i , and by the choice of k . Hence $\delta(p_a) - d > \delta(p_k)$. Therefore, $\delta(p_i) > \delta(p_k)$. Thus, $i \neq i^*$, and the lemma is proved. \square

The following lemma shows that the first condition in the definition of \asymp is redundant.

Lemma 5.2 *If $\bar{A} \equiv \bar{B}$ and $\kappa(A) = \kappa(B)$ then $A \equiv B$.*

Proof. Let l denote the number of elements in either $\kappa(A)$ or $\kappa(B)$. Let C denote the l largest laxity packets in \underline{A} and D denote the set of l largest laxity packets in \underline{B} . Then, using Lemma 3.1, we obtain $A \equiv \bar{A} \cup C \equiv \bar{B} \cup D \equiv B$. \square

6 PROOF FOR MOSTO SCHEDULING

Theorem 2.3 is proved in this section. We first define a subclass of TO policies, which are easier to work with, and then we exhibit a particular MOSTO scheduling policy, $\pi^{min-max}$.

Throughout this section we assume that there is a total order on the set of all packets such that if p and q are packets with $d(p) < d(q)$, then p precedes q in the total order. Such an order essentially amounts to ordering those packets with the same deadline in an arbitrary fashion. The following definition is based on the given total ordering of packets. A policy π is a (MGC) *min-given-class* scheduling policy, if in each slot, for each class of packets i , if π schedules a packet of class i , that packet must be the minimum packet in the set of class i packets available to be scheduled in the slot.

Lemma 6.1 *Given a scheduling policy π , there is an MGC scheduling policy π' such that in each slot, either π and π' schedule a packet of the same class, or neither schedules a packet.*

Proof. Let π' be the scheduling policy such that, in any time slot n , $n \geq 1$, if π schedules a packet in that slot, π' schedules the minimum packet of the same class in $S_n(\pi)$, if any, and does not schedule any packet otherwise. Clearly π' is an MGC policy. To complete the proof, it must be shown that π' schedules a packet in any slot in which π does. Since the following statement can be easily proved by induction for all $n \geq 1$, the proof is complete: For each class of packets i , if $l_1 \leq \dots \leq l_p$ denote the laxities of the class i packets in $S_n(\pi)$ and $l'_1 \leq \dots \leq l'_p$ denote the laxities of the class i packets in $S_n(\pi')$, then $p \leq p'$ and $l_{p-j} \geq l'_{p'-j}$ for $0 \leq j \leq p-1$. \square

A conclusion that can be drawn from the lemma above is the following one: if there exists a MOSTO scheduling policy, then there exists a MOSTO MGC scheduling policy. Also, if there exists a TO MGC scheduling policy π such that, for any other TO MGC policy $\tilde{\pi}$, any $n \geq 1$, and any arrival sequence, π schedules at least as many class one packets in $\{1, \dots, n\}$ as $\tilde{\pi}$, then π is a MOSTO scheduling policy. We can therefore restrict our focus to MGC scheduling policies.

One advantage of working with MGC policies is that they are easy to compare to each other, as indicated by the following lemma. Given two sets of packets A and B , we say that A is a suffix

of B if $A \subset B$ and any packet in $B - A$ precedes any packet in A in the total ordering of packets. The following lemma easily follows by proof by induction on the time slot index k , so the proof is omitted:

Lemma 6.2 *Given any two MGC policies π and $\tilde{\pi}$, any arrival sequence A , any class $i \in \{1, 2\}$ and any time slot k , either the set of class i packets in $S_k(\pi)$ is a suffix of the set of class i packets in $S_k(\tilde{\pi})$, or the set of class i packets in $S_k(\tilde{\pi})$ is a suffix of the set of class i packets in $S_k(\pi)$.*

Let $\pi^{\min-\max}$ be the scheduling policy such that, in each time slot k such that $S_k(\pi^{\min-\max}) \neq \emptyset$, $\pi^{\min-\max}$ schedules the first (in the total ordering) packet in $\Phi_m(S_k(\pi^{\min-\max}))$.

Lemma 6.3 *If S is a set of packets with laxities and $p \notin S$ with $l(p) \leq l^*(S)$ (where l^* is defined in (2.5) then $l^*(p + S) \leq l^*(S)$ (and hence also $\Phi_s(p + S) \subset p + \Phi_s(S)$).*

Proof. Without loss of generality we can assume that p lies before p_{i^*} in the total order of packets. The index (laxity minus rank) of p_{i^*} relative to $p \cup S$ is smaller by one than the index of the same packet relative to S . If q is any other packet in S then its index in $p \cup S$ is at least its index, relative to S , minus one. It may be that p itself is the last packet of $|\Phi_s(p + S)|$, but in that case $l^*(p + S) = l(p) \leq l^*(S)$. The result follows. \square

We say A is unscheduleable if it is not scheduleable. We say A is *subcritical* if it can cover $\{2, \dots, |A| + 1\}$. Clearly a subcritical set is scheduleable. A *critical* set is a scheduleable set which is not subcritical. Note that if δ_{\min} is defined for A as in (2.2), then A is subcritical (critical or unscheduleable, respectively) if δ_{\min} is positive, (zero or negative, respectively).

Lemma 6.4 *Let A be a set of packets with laxities, and suppose A is either critical or unscheduleable. Then for any $p \in \Phi_s(A)$, $\widetilde{A - p} \equiv \tilde{A}$*

Proof. Since $\widetilde{A - p} \subset \tilde{A}$, by Lemma 2.1 it suffices to show that $\widetilde{A - p}$ contains a maximum cardinality scheduleable subset of \tilde{A} . But $\text{rank}(\tilde{A}) \leq l^* - 1 + |A - \Phi_s(A)|$ since only $|A - \Phi_s(A)|$ packets in \tilde{A} have rank greater than $l^* - 1$. On the other hand, $\widetilde{A - p}$ contains a scheduleable

subset of cardinality $l^* - 1 + |A - \Phi_s(A)|$, namely the $l^* - 1$ packets of $\Phi_s(A) - p$ of largest laxity, together with the packets of $A - \Phi_s(A)$. \square

Lemma 6.5 *Policy $\pi^{\min-max}$ is a MOSTO policy.*

Proof. Let π be an MGC TO policy. To prove the lemma we will establish by induction that the following is true for any $n \geq 1$:

$$\mathcal{P}(n) : \begin{cases} (i) \overline{S_n(\pi^{\min-max})} \text{ is a suffix of } \overline{S_n(\pi)}, \\ (ii) \overline{S_n(\pi^{\min-max})} \succeq \overline{S_n(\pi)}, \\ (iii) \text{ any class one packet scheduled by } \pi \text{ before slot } n \\ \text{ is scheduled by } \pi^{\min-max} \text{ before slot } n. \end{cases}$$

The base case, $n = 1$, is trivially true, so suppose that $\mathcal{P}(n)$ is true. It is easy to establish $\mathcal{P}(n+1)$ in the case that $\pi^{\min-max}$ schedules a class one packet in slot n , or does not schedule any packet in slot n . Thus, suppose that $\pi^{\min-max}$ schedules a class two packet in slot n . We will establish parts (i), (ii) and (iii) for $\mathcal{P}(n+1)$ in order.

Part (i) of $\mathcal{P}(n+1)$ is easy to establish if $\overline{S(\pi^{\min-max})}$ is a strict subset of $\overline{S(\pi)}$, so we assume that $\overline{S(\pi^{\min-max})} = \overline{S(\pi)}$. By Lemma 6.2 there are two cases to consider:

Case 1: $\overline{S_n(\pi^{\min-max})}$ is a suffix of $\overline{S_n(\pi)}$. Since $\pi^{\min-max}$ schedules a class 2 packet in slot n , the minimum laxity packet in $S_n(\pi^{\min-max})$ is class 2. Thus, there is a class 2 packet in $S_n(\pi^{\min-max})$ with laxity less than or equal to $l^*(S_n(\pi^{\min-max}))$. Therefore, $l(p) \leq l^*(S_n(\pi^{\min-max}))$ for any packet in $\overline{S_n(\pi)} - \overline{S_n(\pi^{\min-max})}$. Hence, Lemma 6.3 implies that $\overline{\Phi_s(S_n(\pi))} = \overline{\Phi_s(S_n(\pi^{\min-max}))} = \emptyset$, so that π must also schedule a class 2 packet in slot $n+1$. Part (i) of $\mathcal{P}(n+1)$ thus follows in this case.

Case 2: $\overline{S_n(\pi)}$ is a suffix of $\overline{S_n(\pi^{\min-max})}$. Let $m = |S_n(\pi^{\min-max}) - S_n(\pi)|$. Since $\pi^{\min-max}$ and π are both throughput optimal, $S_n(\pi^{\min-max}) \equiv S_n(\pi)$. Thus, the minimum index value δ_{\min} for packets in $S(\pi^{\min-max})$ is less than or equal to $-m$. Therefore, $\Phi_s(S_n(\pi^{\min-max})) \geq m+1$, and by assumption all the packets in this set are class 2 packets. Thus, letting p^* denote the last

packet in $\Phi_s(S_n(\pi^{\min-\max}))$, it follows that p^* and all packets in $S_n(\pi^{\min-\max})$ after it (in the total packet ordering) are also in $S_n(\pi)$. The index of any of these packets relative to $S_n(\pi)$ is equal to m plus the index of the same packet relative to $S_n(\pi^{\min-\max})$. Therefore, the index of packet p^* is less than or equal to the index of all subsequent packets of $S_n(\pi)$, whether indices are computed relative to $S_n(\pi)$ or relative to $S_n(\pi^{\min-\max})$. Therefore, $\Phi_s(S_n(\pi)) \subset \Phi_s(S_n(\pi^{\min-\max}))$, so that policy π also schedules a class two packet in slot n . Part (i) of $\mathcal{P}(n+1)$ thus follows in this case.

Thus, part (i) of $\mathcal{P}(n+1)$ is proved. Part (ii) will be proved by considering two cases:

Case 1: $S_n(\pi^{\min-\max})$ is critical or unscheduleable. In this case, Lemmas 6.4 and 3.1 imply part (ii) of $\mathcal{P}(n+1)$.

Case 2: $S_n(\pi^{\min-\max})$ is subcritical. Since it is assumed that $\overline{\Phi_s(S_n)} = \emptyset$, it follows that $S_n(\pi^{\min-\max})$ is also subcritical. Since $S_n(\pi^{\min-\max}) \equiv S_n(\pi)$ (because both policies are TO) it follows that $S_n(\pi)$ is also subcritical and it has the same cardinality as $S_n(\pi^{\min-\max})$. Since $\overline{S_n(\pi^{\min-\max})}$ is a suffix of $\overline{S_n(\pi)}$, it follows that $\overline{S_n(\pi)}$ is a suffix of $\overline{S_n(\pi^{\min-\max})}$. If $\overline{S_n(\pi)} = \overline{S_n(\pi^{\min-\max})}$, then $S_n(\pi) = S_n(\pi^{\min-\max})$, and π must also schedule a class 2 packet in slot n . On the other hand, if $\overline{S_n(\pi)}$ is a strict suffix of $\overline{S_n(\pi^{\min-\max})}$, then no matter the packet scheduled by π in slot n , $\overline{S_{n+1}(\pi)} \subset \overline{S_n(\pi^{\min-\max})}$. Either way, part (ii) of $\mathcal{P}(n+1)$ is implied.

It remains to establish part (iii) of $\mathcal{P}(n+1)$. In view of the induction hypothesis, we assume that π schedules a class 1 packet p in slot n , and only need to show that p is scheduled by $\pi^{\min-\max}$ in slot n or sooner. If $\pi^{\min-\max}$ also schedules a class 1 packet in slot n , then by part (i) already proved, either $\overline{S_n(\pi)} = \overline{S_n(\pi^{\min-\max})}$ (so $\pi^{\min-\max}$ schedules p in slot n) or $\overline{S_n(\pi^{\min-\max})}$ is a strict prefix of $\overline{S_n(\pi)}$ (so $\pi^{\min-\max}$ schedules p before slot n). If $\pi^{\min-\max}$ schedules a class 2 packet in slot n , then by the proof of part (i) of $\mathcal{P}(n+1)$ above, $\overline{S_n(\pi^{\min-\max})}$ is a strict prefix of $\overline{S_n(\pi)}$ (so $\pi^{\min-\max}$ schedules p before slot n). The proof of part (iii) of $\mathcal{P}(n+1)$, and hence the proof of the lemma, is complete. \square

Proof of Theorem 2.3 Suppose π is a policy such that, for any arrival sequence \mathcal{A} and any $k \geq 1$ such that $S_k(\pi) \neq \emptyset$, $\pi(\mathcal{A}; k) \in \Phi_m(S_k(\pi))$. We show that the following induction hypothesis holds

for any $n \geq 0$:

$$\mathcal{P}(n) : \begin{cases} \overline{S_n(\pi)} \equiv \overline{S_n(\pi^{min-max})} \\ S_n(\pi) \equiv S_n(\pi^{min-max}) \end{cases} \quad (6.1)$$

Of course $\mathcal{P}(0)$ is true. We assume that the $\mathcal{P}(n)$ is true for some $n \geq 0$. Then obviously either both π and $\pi^{min-max}$ schedule a packet of the same class or they do not schedule a packet at all. Moreover, if π schedules a class one packet in slot n then that packet is in $\Phi_s(\overline{S_k(\pi)})$ and if π schedules a class two packet in slot n then that packet is in $\Phi_s(\underline{S_k(\pi)})$. Policy $\pi^{min-max}$ also has these properties. Therefore $\mathcal{P}(n+1)$ is simply established by applying Lemmas 3.1 and 3.4 to each class of packets separately. Thus the two scheduling policies schedule a packet of the same class or do not schedule a packet in each slot. Since $\pi^{min-max}$ is a MOSTO policy, so is π . Thus the “if” part of the theorem is proved.

Turning to the “only if” part of the Theorem, suppose that π is a MOSTO policy. Since π is TO, Theorem 2.1 implies that for all k such that $S_k \neq \emptyset$, $\pi(\mathcal{A}; k) \in \Phi_s(S_k(\pi))$. Also, if there is a class one packet in $\Phi_s(S_k(\pi))$ then π must schedule a class one packet, and moreover, that packet has to be in $\Phi_s(\overline{S_k(\pi)})$. Otherwise, we could construct a TO scheduling policy scheduling more class one packets in an interval starting from time slot one using a subsequent arrival sequence of class one packets similar to the one constructed in the proof of the “only if” part of Theorem 2.1.

□

7 PROOF FOR MOSTO SCHEDULING-DROPPING

Theorem 2.4 is proved in this section.

Lemma 7.1 *If $S \asymp T$ and R is disjoint from $S \cup T$, then $S \cup R \asymp T \cup R$.*

Proof. It suffices to consider the case that R contains only a single packet p . Note that $rank(S) = rank(T)$, $(S + p) = rank(S + p)$, and $rank(\overline{S} + p) = rank(\overline{T} + p)$. First suppose p is a class one packet. There are two cases. If adding p either increases the rank of both S and \overline{S} , or leaves unchanged the rank of both S and \overline{S} , then the same is true for S replaced by T , and

$\kappa(S + p) = \kappa(S) = \kappa(T) = \kappa(S + p)$. On the other hand, if adding p increases the rank of \bar{S} but leaves unchanged the rank of S , then the same is true for S replaced by T , and $\kappa(S + p)$ and $\kappa(T + p)$ are both equal to $\kappa(S)$ with the smallest element removed.

Suppose instead that p is a class two packet. Again there are two cases. If $\text{rank}(S + p) = \text{rank}(S)$, then also $\text{rank}(T + p) = \text{rank}(T)$, and $\kappa(S + p) = \kappa(S) = \kappa(T) = \kappa(S + p)$. If on the other hand $\text{rank}(S + p) = \text{rank}(S) + 1$, then $\kappa(S + p)$ and $\kappa(T + p)$ are both equal to the result of adding to them the largest number in $\{1, \dots, l(p)\}$ not already in them. \square

The following lemma gives a useful characterization of $\Phi_s(S)$ in terms of a covering property.

Lemma 7.2 *Let S be a set of packets with laxities and let $p \in S$. Then $p \in \Phi_s(S)$ if and only if $\widetilde{S - p} \equiv \widetilde{S - p_1}$, where p_1 is a minimum laxity packet in S .*

Proof. The “only if” part of the lemma is immediate from Lemma 3.4. For the “if” part, suppose that $p \notin \Phi_s(S)$. Let l^* be defined as in (2.5) and let $m = |S - \Phi_s(S)|$. All packets in $\Phi_s(S)$ have laxity less than or equal to l^* , and the packets of $S - \Phi_s(S)$ can cover the m slots $\{l^* + 1, \dots, l^* + m\}$. Therefore, if $\tau = \{l^*, \dots, l^* + m - 1\}$, then $\widetilde{S - p_1}$ can cover τ , but $\widetilde{S - p}$ cannot. Thus $\widetilde{S - p} \not\equiv \widetilde{S - p_1}$, and the lemma is proved. \square

Lemma 7.3 *Let S and T be two sets of packets such that $S \asymp T$. Let $p \in \Phi_m(S)$ and $q \in \Phi_m(T)$, then*

- (i) p and q are the same class,
- (ii) $\widetilde{S - p} \asymp \widetilde{T - q}$.

Proof. We first prove (i) by showing that if p is a class one packet then there is a class one packet in $\Phi_m(T)$. That is sufficient since all packets in $\Phi_m(T)$ are the same class. Equivalently, we need show that there is a class one packet in $\Phi_s(T)$. We use Lemma 7.2. Suppose p is a class one packet and let $1 \in \tau \subset \{1, 2, \dots\}$ such that T can cover τ . It suffices to show that T can cover τ with a class one packet assigned to 1. The set of packets S can cover τ (since $S \equiv T$) and can do so with packet p assigned to 1 (since $p \in \Phi_s(S)$). Fix a corresponding assignment of packets in S covering

τ and let τ_1 denote the subset of τ which is assigned to packets in \bar{S} . Note that $1 \in \tau_1$. Since $\bar{T} \equiv \bar{S}$, the set τ_1 can also be covered by T (in particular a class one packet is assigned to 1). The remaining slots, i.e. those in $\tau - \tau_1$, can be covered by \underline{S} , and hence by \underline{T} , by the third condition in the definition of $S \asymp T$. Thus, T can indeed cover τ with a class one packet assigned to 1, and part (i) is proved.

We next show (ii). We know from Lemma 3.4 and part (i) proved above that

$$\widetilde{S - p} \equiv \widetilde{T - q} \text{ and } \widetilde{\bar{S} - p} \equiv \widetilde{\bar{T} - q} \quad (7.1)$$

It remains to prove $\kappa(\widetilde{S - p}) = \kappa(\widetilde{T - q})$. Let $l = \text{rank}(\widetilde{S - p}) - \text{rank}(\widetilde{\bar{S} - p})$. All we need is to show that the set of the l largest packets in $\widetilde{S - p}$ is equivalent to the set of the l largest packets in $\widetilde{T - q}$. Let $l' = \text{rank}(S) - \text{rank}(\bar{S}) = \text{rank}(T) - \text{rank}(\bar{T})$. Then $l' \geq l$ since

$$\begin{aligned} l' - l &= (\text{rank}(S) - \text{rank}(\bar{S})) - (\text{rank}(\widetilde{S - p}) - \text{rank}(\widetilde{\bar{S} - p})) \\ &= 1 - (\text{rank}(\bar{S}) - \text{rank}(\widetilde{\bar{S} - p})) \geq 0 \end{aligned}$$

Thus let us consider two cases:

Case 1: p and q are class one packets. In this case

$$\widetilde{S - p} = \tilde{S} \equiv \widetilde{T - q} = \tilde{T} \quad (7.2)$$

Thus the l largest packets of $\widetilde{S - p}$ and $\widetilde{T - q}$ form equivalent sets, since the l' largest packets of those sets do so.

Case 2: p and q are class two packets. Since there are no class one packets in $\Phi_s(S)$ and $\Phi_s(T)$, \bar{S} and \bar{T} are subcritical, so

$$\text{rank}(\bar{S}) = \text{rank}(\tilde{\bar{S}}) = \text{rank}(\bar{T}) = \text{rank}(\tilde{\bar{T}}) \quad (7.3)$$

Thus $l = l' - 1$. Since $p \in \Phi_s(\underline{S})$, the set of the $l - 1$ largest packets of $\widetilde{S - p}$ is equivalent to the set of the $l - 1$ largest packets of \tilde{S} (by Lemma 7.2). We have a similar result for \underline{T} , and the result is established. \square

Proof of Theorem 2.4 Suppose π is a scheduling-dropping policy such that for any arrival sequence \mathcal{A} and all k , $Q_k \asymp S_k$, and if $S_k(\pi) \neq \emptyset$ then $\pi(\mathcal{A}, k) \in \Phi_m(Q_k)$. Lemmas 7.1 and 7.3

and induction on k imply that $S_k(\pi) \asymp S_k(\pi^{\min-\max})$ for all k . Therefore also by Lemma 7.3, in each slot, either π and $\pi^{\min-\max}$ schedule a packet of the same class, or neither schedules a packet. Since $\pi^{\min-\max}$ is MOSTO, so is π . The “if” half of Theorem 2.4 is proved.

Turning to the “only if” portion of the theorem, suppose π is a causal policy such that for some arrival sequence and some slot k , it holds that either $S_k \not\equiv Q_k$ or $[Q_k \neq \emptyset$ but π does not schedule a packet in $\Phi_m(Q_k)$ in slot $k]$. It must be shown that π is not MOSTO. If $[Q_k \neq \emptyset$ but π does not schedule a packet in $\Phi_m(Q_k)$ in slot $k]$ then π is not MOSTO by the “only if” half of Theorem 2.3. Thus, we can restrict attention to the situation that for some arrival sequence and some slot k , it holds that $S_k \not\equiv Q_k$. If $S_k \not\equiv Q_k$ then π is not even TO by the “only if” part of Theorem 2.1. If $\overline{S}_k \neq \overline{Q}_k$ then, by Lemma 2.1, and the fact $\overline{Q}_k \subset \overline{S}_k$, $\text{rank}(\overline{Q}_k) < \text{rank}(\overline{S}_k)$. Therefore, π is not MOSTO because if there are no future arrivals, π can schedule at most $\text{rank}(\overline{Q}_k)$ class one packets in slots $\{k, k+1, \dots\}$, whereas a MOSTO scheduling policy starting with S_k in slot k will schedule a total of $\text{rank}(\overline{S}_k)$ class one packets in those slots.

It remains to consider the possibility that $Q_k \equiv S_k$ and $\overline{Q}_k \equiv \overline{S}_k$ but $\kappa(Q_k) \neq \kappa(S_k)$. For simplicity of notation, suppose that $k = 1$ and let $S = S_1$ and $Q = Q_1$. Let π be a TO scheduling policy with initial state Q in slot 1. We are to construct a TO scheduling policy $\hat{\pi}$ and arrival sequence for slots 2, 3, \dots such that, using the arrival sequence, the following holds for some $n \geq 1$: Policy $\hat{\pi}$ starting with set S in slot 1, schedules more class one packets during $\{1, \dots, n\}$ than does π starting with set Q in slot 1.

Recall that $\mathcal{L}(S)$ can be computed by Algorithm G2, given an arbitrary ordering of packets. Let $l = \text{rank}(S) - \text{rank}(\overline{S})$ (which is also equal to $\text{rank}(Q) - \text{rank}(\overline{Q})$ since $Q \equiv S$ and $\overline{Q} \equiv \overline{S}$) and suppose the algorithm is run on the packets of S in the following order. Packets in the set C , consisting of the l largest laxity packets in \underline{S} , are considered first, followed by the packets of \overline{S} , and finally by packets in $\underline{S} - C$. Write $\mathcal{L}(S) = \{t_1, \dots, t_{\text{rank}(S)}\}$, and let p_i denote the packet to which t_i is assigned by the algorithm (note, this is not the same as p_i as described in the algorithm), for $1 \leq i \leq \text{rank}(S)$. Note that $\kappa(S) = \mathcal{L}(C) \subset \mathcal{L}(S)$ and that the l packets in C , which are among $p_1, \dots, p_{\text{rank}(S)}$, are assigned elements of the set $\mathcal{L}(C)$. Also note that, since $\kappa(Q) \neq \kappa(S)$, the set $\mathcal{L}(C)$ strictly dominates any set τ that can be covered by any set of l packets in \underline{Q} .

Let the future arrival sequence A_2, A_3, \dots be as follows: In every time slot *except* slots $1, t_2, t_3, \dots, t_{\text{rank}(S)}$, a class one packet arrives with laxity one. No other packets arrive in any slot.

Let $\hat{\pi}$ be defined as follows. We first describe $\hat{\pi}$ for the particular arrival sequence just specified. Packet p_1 is scheduled in slot 1. Packet p_i is scheduled in slot t_i for $2 \leq i \leq \text{rank}(S)$. Finally, in any other slot, the class one packet just arriving in the slot is scheduled. Check that the packet $\hat{\pi}$ schedules in any given slot k is in $\Phi_s(S(k))$. This is readily verified using the characterization of Lemma 7.2, because after each slot, the set of packets that $\hat{\pi}$ has not yet scheduled can cover any set that would be coverable had π made a different decision. So far $\hat{\pi}$ has been specified for only a single arrival sequence. However, since (for that arrival sequence) any packet π schedules in any given slot k is in $\Phi_s(S(k))$, the policy $\hat{\pi}$ can be extended to a causal TO policy defined for arbitrary arrival sequences.

A particular arrival sequence \mathcal{A} and a TO policy $\hat{\pi}$ have now been specified. Note that there is a departure for $\hat{\pi}$ in every slot. Since π is TO, it must also have departures in every slot. Since the new arrivals cannot be used for any of the slots $1, t_2, t_3, \dots, t_{\text{rank}(S)}$, it follows that π must schedule packets of Q in these slots, and the new (class 1) arrivals in all other slots. If π schedules fewer class one packets in slots $1, t_2, t_3, \dots, t_{\text{rank}(S)}$ than does $\hat{\pi}$, then π is not MOSTO. On the other hand, if π schedules as many class one packets in those slots as $\hat{\pi}$, then it also schedules exactly l packets of Q in those slots.

Let $s_1 < \dots < s_l$ denote the times that π schedules class two packets and let $\hat{s}_1 < \dots < \hat{s}_l$ denote the times that $\hat{\pi}$ schedules class two packets. Both sequences are subsequences of $1, t_2, \dots, t_{\text{rank}(S)}$. We consider two cases:

Case 1: Policy $\hat{\pi}$ schedules a class one packet in slot 1. Then $\mathcal{L}(C) = \{s_1, \dots, s_l\}$, so that $s_i \leq \hat{s}_i$ for $1 \leq i \leq l$, and $s_i < \hat{s}_i$ for at least one value of i . Therefore π is not MOSTO in this case.

Case 2: Policy $\hat{\pi}$ schedules a class two packet in slot 1. Then t_1 is the smallest element of $\mathcal{L}(C)$. Policy π schedules $l = |\mathcal{L}(C)|$ packets of Q , and the set of slots used is dominated by $\mathcal{L}(C)$ —, so π must schedule a packet of Q in or before slot t_1 . Since slot 1 is the only slot available to π in or before slot t_1 for scheduling packets of Q , policy π must also schedule a class 2 packet in slot 1. The laxity of that packet, indeed all packets in S , is at least equal to t_1 . Thus, if $t_1 > 1$, then

π schedules a (new) class one packet in slot t_1 , so if the packet scheduled in slot 1 is delayed to slot t_1 , then all deadlines are still met by the class two packets. (If $t_1 = 1$ such a delay is not necessary.) Therefore, \underline{Q} can cover $t_1, s_2, s_3, \dots, s_l$. Each number in this sequence is less than or equal to the corresponding number in $C = \{t_1, \hat{s}_2, \hat{s}_3, \dots, \hat{s}_l\}$, and, moreover, strict inequality holds in at least one instance. Thus, $s_i \leq \hat{s}_i$ for $2 \leq i \leq l$, with strict inequality for at least one value of i . Therefore, π is not MOSTO, and the first statement in Theorem 2.4 is proved.

It remains to prove the second statement of the Theorem. By the first statement of the Theorem and its proof, $S_k(\pi) \asymp S_k(\pi^{\min-max}) \asymp S_k(\pi')$ for all k , so that $Q_k(\pi) \asymp S_k(\pi) \asymp S_k(\pi') \asymp Q_k(\pi')$ for all k . The theorem is proved. □

References

- [1] E. L. Lawler *Combinatorial Optimization: Networks and Matriods* Holt, Rinehart and Winston, New York, 1976.
- [2] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* Prentice Hall, Englewood Cliffs, NJ, 1982.
- [3] J. M. Peha and F. A. Tobagi, Evaluating Scheduling Algorithms for Traffic with Heterogeneous Performance Objectives *Proceedings IEEE GLOBCOM*, pp. 21-27, December 1990.
- [4] T.-L. Ling and N. Shroff "Scheduling Real-Time Traffic in ATM Networks", *Proceedings IEEE INFOCOM '96*, pp. 198-205, March 1996.
- [5] T.-L. Ling and N. Shroff "Scheduling Real-Time Traffic in ATM Networks", Preprint submitted to *IEEE/ACM Transactions on Networking*, 1996.
- [6] R. Chiopalkatti, J. F. Kurose, and D. Towsley, Scheduling Policies for Real-Time and Non-Real-Time Traffic in a Statistical Multiplexer *Proceedings IEEE INFOCOM 89*, pp. 774-783, Ottawa, April 1989.