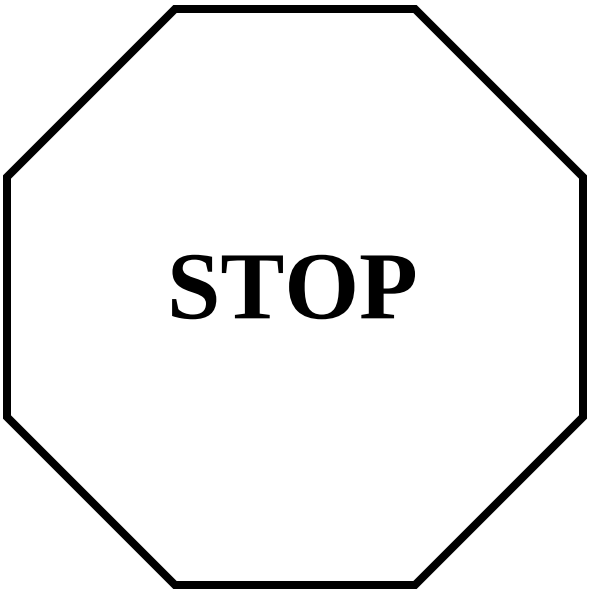
Educational Materials
CMU/SEI-94-EM-11

Rate Monotonic Analysis for Real-Time Systems: Instructor's Guide

Ruth Ravenel
Ray Obenza

April 1994



Please note that this material is intended to be used in conjunction with the SEI Technology Series videotape, *An introduction to Rate Monotonic Analysis*. The information in this document is incomplete and should be supplemented by the videotaped lecture. You can obtain the tape using the order form in the back of this document.

Table of Contents

Preface 1

1 Introduction 2

2 Educational Objectives 2

3 Recommendations for Use 3

- 3.1 Including RMA in a Course 3
- 3.2 Lecture Notes 4
- 3.3 Sample Homework Assignments and Exam 4
- 3.4 Presentation Slides 5
- 3.5 RMA Programs 5

4 Pedagogical Considerations 6

5 Bibliography 7

Lecture Notes

- Background on RMA
- What's Important for Real-Time Systems
- The Mathematical Notation of Theorem 2

Sample Homework Assignments and Exam Question

- Homework Assignment #1
- Homework Assignment #1 Solution and Comments
- Homework Assignment #2
- Homework Assignment #2 Solutions and Comments
- Exam
- Exam Solution and Comments

Preface

This educational materials package has been developed for instructors of software engineering and, more specifically, real-time systems. This package will help instructors teach rate monotonic analysis (RMA) to graduate and undergraduate software, computer, and electrical engineering students. The package can also be used to teach RMA to continuing education students. The presentation materials and exercises included have been used by Ruth Ravenel in both graduate and undergraduate courses.

Since RMA has only recently been applied to real-time designs to analyze whether a system will meet timing requirements, RMA is not yet covered in standard textbooks. A handbook now exists, *A Practitioner's Handbook for Real-Time Analysis* [Klein 93] that steps engineers through the application of RMA to various real-time situations; however, the handbook is an advanced text for people who have already been introduced to RMA. This serves as introductory material to RMA.

These educational materials are intended to be used in conjunction with the videotape, *An Introduction to Rate Monotonic Analysis*, from the SEI Technology Series. (For instructors who have not already obtained the videotape from the SEI, an order form is included in this package.)

This educational materials package consists of the following materials:

- 79-minute instructional videotape, *An Introduction to Rate Monotonic Analysis* (to obtain, use attached order form)
- Instructor's guide (this document), including
  - educational objectives and recommendations for use
  - outline of videotaped lecture
  - bibliography
  - lecture notes
  - background on RMA
  - discussion on the distinction between real-time systems and other systems
  - added detailed explanations on selected concepts (not in videotape)
  - sample homework assignments with answers and explanations
  - sample exam question with answer and explanation
- (Optional) Presentation slides used in videotaped presentation (see section 3.4). These slides can be obtained two ways:
  - diskette containing PowerPoint files [Macintosh version 2.0.1] (to obtain, use attached order form)
  - anonymous FTP of PostScript files
- (Optional) Non-supported C programs (see section 3.5). These slides can be obtained two ways:
  - diskette containing source code in text files (to obtain, use attached order form and memorandum of agreement)
  - SEI world wide web server, via Mosaic

1 Introduction

With the burgeoning use of microprocessors in commonplace products, there is an ever-increasing need for software developers to understand the issues involved in real-time system design. This introduction will not educate a developer sufficiently to be proficient, nor is it intended to do so. But it can introduce the key concepts in sufficient depth to motivate the student to learn more when the need arises. It is packaged compactly so that it will not occupy an undue portion of a course and can therefore be inserted in courses in software engineering, systems design, or real-time systems.

Typically, textbook discussions of real-time system design give heuristics based on experience and intuition without recourse to science or mathematics as an aid. Now that the mathematical foundation has been developed, it is possible to replace intuitive heuristic approaches with genuinely substantive material for use in master's level graduate courses and also undergraduate courses. Those who already had the years of experience and have learned about RMA in a graduate course have reported that they found its application of great benefit upon returning to their employment.

2 Educational Objectives

Upon completion of the RMA unit, students should

- Understand the key issues that make the design of real-time systems different from the design of other systems, including interactive time-sharing systems.
- Understand that guaranteeing the schedulability of a set of real-time tasks is not only possible, but also mathematically simple.
- Understand that the applicability of RMA has been broadened to a wide range of task characteristics and that RMA is therefore useful in the design and evaluation of practical real-time systems.
- Know essential vocabulary necessary for reading other sources for further study of this topic.

More specifically, students should be able to

- Evaluate a set of real-time tasks to determine whether they meet the criteria for application of RMA.
- Apply RMA principles to determine whether a set of tasks is schedulable; that is, to determine whether one can guarantee that all tasks will be able to meet their deadlines, even in worst-case conditions.
- Extend RMA principles to handle certain deviations from basic criteria required for RMA application. In particular, students should be able to extend RMA principles to handle tasks that are aperiodic, that share a common resource (not independent), that have preperiod deadlines, or that are interrupt-driven.

3 Recommendations for Use

3.1 Including RMA in a Course

The RMA unit has already been included in two kinds of graduate-level courses. First, in a master's level course in software engineering, the RMA unit was in the section on design and followed lectures on design fundamentals and data flow-oriented design. It could have been inserted immediately after the design fundamentals lecture. Students in the course were from computer science, electrical and computer engineering, and software engineering graduate programs. There were also individuals from industry taking the course for their continuing education.

Second, the RMA unit has been included in a master's level course in real-time hardware-software system design. It was found to fit well after the topics of synchronization, parallelization, real-time kernels, and basic scheduling issues.

The videotape (to obtain, use attached order form) is 79 minutes and is divided as follows:

Introduction:

- Approximately 9 minutes
- Introduces the concepts of real-time systems development.

Basic theory through examples:

- Approximately 20 minutes
- Reviews the two basic theorems that are used to test the ability of real-time tasks to meet timing requirements.

Some extensions to basic theory:

- Approximately 20 minutes
- Extends basic theory to be useful on actual systems.

Extended case study:

- Approximately 26 minutes
- Reviews an example of an actual system on which RMA was used to troubleshoot and modify in order to meet timing requirements.

A possible 3-class (50 minutes per class) sequence could be

Class 1:

- Discussion of fundamentals that distinguish real-time systems from other types of systems (not included on video; see lecture notes provided in this package).
- Video: *Introduction* and *Basic Theory* sections.

Class 2:

- Review of examples from video or work on some additional problems that are provided in this package.
- Video: section on *Extensions*.

Class 3:

- Review of examples of extensions from the video or work on some additional problems involving extensions.
- Video: section on the case study.

It is of course possible to run the video completely in two class sessions with time for discussion at the end of the second day. However, this approach has not been found to provide the same level of student retention of the material.

3.2 Lecture Notes

The lecture notes, included at the back of this document, are provided to the instructor as additional background on RMA. Although these notes are intended to help instructors, the notes are designed so that they can also be distributed to students.

The *Background on RMA* section reviews some history of RMA and explains its usefulness.

The discussion on *What's Important for Real-Time Systems* is provided as basic information on real-time systems in software engineering. If this unit is going to be inserted in a general software engineering course, students may not be sufficiently familiar with real-time systems to understand how these design issues differ from issues in other systems.

Depending on the mathematical sophistication of the student, the notation in Theorem 2 can be quite intimidating. The section *The Mathematical Notation of Theorem 2* is a step-by-step explanation of what that notation indicates. Theorem 2, shown below, appears in the video as an important theorem for determining schedulability precisely. Its proof is in a paper by Lehoczky, Sha, and Ding [Lehoczky 89]. While the video explains the net effect of this theorem in terms of a timeline, the video does not explain how the timeline approach and the algebra are describing the same process.

3.3 Sample Homework Assignments and Exam Question

The following homework assignments and exam question are included at the back of this document. The homework assignments can be used to enhance and test the understanding of students. Two equivalent homework assignments are provided. One could be given to fall classes, and the second to spring classes. Note that each assignment asks several questions that build in difficulty. Determine and assign the individual questions as appropriate, based on the material covered in each lecture.

Included for each homework assignment are

- Homework handouts
- Answers
- Further comments

The exam question can be used to test the understanding of students. It has previously been provided to students as one of five questions, three of which would be answered by each student, on a final exam.

Included for the exam question are

- The examination question
- Answers
- Further comments

3.4 Presentation Slides

Instructors who choose to present the material live rather than use the videotaped lecture in class can obtain the slides that were used in the video. These slides can be obtained two ways:

- Original slides (as presented on the videotape) were developed using PowerPoint 2.0.1 for the Macintosh. Instructors can order a diskette from the SEI containing both color and black-and-white versions of the slides in PowerPoint files. These PowerPoint files include all slides (both textual and graphic) used in the videotape.
- Those that do not have access to PowerPoint may obtain printed slides via anonymous FTP of a PostScript file called *em11slides.ps*. Printing this file produces only slides that employ graphics. Text slides are not included. Printing this PostScript file produces 32 pages.

3.5 RMA Programs

Two C programs that apply RMA techniques can be obtained. They are available via Mosaic or by ordering the diskette (which contains both the presentation slides and the programs). U.S. Air Force policy requires that a memorandum of agreement (MOA) be signed before code developed under U.S. government funds can be provided to a requester.

To obtain the code on diskette, complete the order form and the attached MOA, and send them both to the SEI. You will receive the code listed below.

To obtain the code using Mosaic, your version of Mosaic must be able to support forms. Open Mosaic and go to the SEI home page at *http://www.sei.cmu.edu*. Click on *Software Engineering Institute*, then *SEI Publications*, then *Search the Annotated Catalog*. Search using the keyword *em11*. Then click on the title found, *Rate Monotonic Analysis for Real-Time Systems: Instructor's Guide*. Click on *Retrieve accompanying code*. You will see the MOA as

a form that you can fill out on your screen. Complete the MOA and click on the *Sign* button, and the code will be sent to you via email. You will receive the following programs:

- C Program called *Busyperiod test*

  This program calculates response times for a set of events (or tasks). It is based on Technique 5 in *A Practitioner's Handbook for Real-Time Analysis*.

- C Program called *Overrun test*

  This program can be used when an event (or task) is not able to meet a deadline. It will identify areas where execution time can be reduced to eliminate overrun.

- Sample input

  This is an example of the input required to run the supplied programs.

4 Pedagogical Considerations

Those who have taught this material stress the importance of the exercises in sealing the learning process. Because the video is rather fast-paced, it is possible for students to find the logical presentation of the material quite understandable while watching the video. However, professors have found that students do not grasp subtle but important points until they attempt the homework assignments.

Answers and comments accompany the sample homework assignments in Appendix B.

For a course in real-time systems that is accompanied by a semester project, it is highly recommended that *A Practitioners Handbook for Real-Time Analysis* [Klein 93] be available for reference. The video is a substantive introduction to the topic, but there are special cases that are not covered in the video and yet can occur in a student lab project. The RMA handbook is comprehensive in the situations it discusses, and its vocabulary is readily accessible to students who have completed this unit. With some guidance from the handbook, a student can understand how the theory is extended to his or her special case and be able to proceed with the design.

5 Bibliography

[Goodenough 88] Goodenough, John B.; & Sha, Lui. "The Priority Ceiling Protocol: A Method for Minimizing the Blocking of High Priority Ada Tasks." *Proceedings of the 2nd International Workshop on Real-Time Ada Issues, Ada Letters 8*, 7 (Fall 1988): 20-31.

[Klein 93] Klein, M. H.; Ralya, T.; Pollak, B.; Obenza, R.; & Gonzalez Harbour, M. *A Practitioners Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*. Boston, MA: Kluwer Academic Publishers, 1993.

[Lehoczky 89] Lehoczky, J. P.; Sha, L.; & Ding, Y. "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," 166-171. *Proceedings of the IEEE Real-Time Systems Symposium*. Los Alamitos, CA: IEEE Computer Society Press, 1989.

[Liu 73] Liu, C. L.; & Layland, J. W. "Scheduling Algorithms for Multi-Programming in a Hard Real-Time Environment." *Journal of the Association for Computing Machinery (ACM) 20,* 1 (January 1973): 40-61.

[Locke 92] Locke, C. Douglass. "Software Architecture for Hard Real-Time Applications: Cyclic Executives vs. Fixed Priority Executives." *The Journal of Real-Time Systems 4* (1992): 37-53.

[Sha 90a] Sha, Lui; & Goodenough, John B. "Real-Time Scheduling Theory and Ada." *IEEE Computer 23,* 4 (April 1990): 53-62.

[Sha 90b] Sha, Lui; Rajkumar, R.; & Lehoczky, John P. "Priority Inheritance Protocols: An Approach to Real-Time Synchronization." *IEEE Transactions on Computers* 39, 9 (September 1990): 1175-1185.

[Sha 91] Sha, Lui; Rajkumar, R.; & Lehoczky, J.P. "Real-Time Computing with IEEE Futurebus+." *IEEE Micro 11* (June 1991): 30-38 & ff.

[Sprunt 89] Sprunt, Brinkley; Sha, Lui; & Lehoczky, John P. "Aperiodic Task Scheduling for Hard Real-Time Systems." *The Journal of Real-Time Systems 1* (1989): 27-60.

Lecture Notes

Background on RMA

1 What is RMA?

Timing behavior is important to all computing systems. However, real-time systems are distinguished by the fact that strict timing behavior is included in the performance requirements. The “correctness” of a real-time system is based not only on its functionality, but also its timeliness with respect to hard deadlines. Depending on the application, an incorrect system can result in catastrophe including loss of life. Therefore, it is important for real-time system developers to be able to ensure that their designs will meet those timing requirements. Because it stresses the important aspects particular to real-time systems, rate monotonic analysis (RMA) provides the real-time systems practitioner with this capability. For example, unlike time-sharing systems in which fairness among clients is favored, RMA focuses on guaranteeing that critical deadlines will always be met, even in worst-case loading, at the risk of starving noncritical tasks.

RMA is a mathematical approach that can help ensure that a real-time system meets its performance requirements. RMA provides a collection of quantitative methods and algorithms that allow engineers to understand, analyze, and predict the timing behavior of their designs.

In addition to ensuring that critical deadlines will be met, RMA can also help ensure that requirements on soft deadlines will be met. It gives designers a scientific approach to identifying potential timing problems before the system is built, as well as helping to troubleshoot performance problems in existing systems. With the use of RMA, timing problems can be identified more easily and with greater certainty than has previously been possible.

Since the case study in this introductory video covers only the software tasks of an embedded system, it is easy to overlook the fact that RMA can be applied to the entire system—hardware, software, clients, resources, etc. Students who are employed in the development of real-time systems will no doubt want to know that industrial training on more extensive applications of RMA is available from commercial trainers. Interested persons can contact the SEI for up-to-date information about trainers. Also, the RMA handbook, *A Practitioner’s Handbook for Real Time Analysis* [Klein 93], is an invaluable resource for addressing more advanced concepts.

2 Why Study RMA?

With the burgeoning use of microprocessors in commonplace products, there is an ever-increasing need for software developers to understand the issues involved in real-time system design. This introduction will not educate a developer sufficiently to be proficient, nor is it intended to do so. But it can introduce the key concepts in sufficient depth to motivate the student to learn more when the need arises. It is packaged compactly so that it will not occupy

an undue portion of a course and can therefore be inserted in courses in software engineering, systems design, or real-time systems.

Typically, textbook discussions of real-time system design give heuristics based on experience and intuition without recourse to science or mathematics as an aid. Now that the mathematical foundation has been developed, it is possible to replace intuitive heuristic approaches with genuinely substantive material for use in master's level graduate courses and also undergraduate courses. Those who already had the years of experience and have learned about RMA in a graduate course have reported that they found its application of great benefit upon returning to their employment.

3 A Bit of History

Historically, designers of real-time systems have built capacity in individual components to handle high throughput and high speeds. Then, when the components were integrated, the system was tested to see if it met the timing requirements. If not, adjustments were made to individual components and the system was tested again. Even if timing requirements were satisfied in the tests that were attempted, there was no guarantee that the testers had been able to simulate the worst-case situations.

This approach has been very unpredictable and very costly. IBM was interested in applying a formal mathematical approach to system design, long before integration, to ensure that the system would meet timing requirements when built. They identified a scheduling algorithm that guaranteed mathematically that a set of tasks would meet all their deadlines, even in the worst-case situations.

The methods and algorithms of RMA are based on a theory of fixed-priority scheduling first discussed in a 1973 *Journal of the Association for Computing Machinery* paper by Liu and Layland [Liu 73]. They introduced a theorem that proved the schedulability of a set of concurrent tasks. However, the imposed assumptions were much too restrictive for the approach to be practical in today's real-time systems.

In 1982, IBM and Carnegie Mellon University (CMU) initiated a joint research project to extend the work of Liu and Layland. In 1988, the CMU Software Engineering Institute (SEI), sponsored by the U.S. Department of Defense, began extending this theory to address a broader range of systems. Today RMA is useful on all real-time systems and has been applied in several industrial development efforts.

What's Important for Real-Time Systems

	Time-Share Systems	Real-Time Systems
Capacity	High throughput	Schedulability
Responsiveness	Fast average response	Ensured worst-case latency
Overload	Fairness	Stability

In time-sharing systems, the measures of merit are high throughput, fast average response time, and fairness to all the users. For example, in an interactive airline reservation system, high throughput ensures that the system processes as many reservations per hour as possible; fast average response time ensures that, most of the time, the system responds quickly enough to satisfy the users; and fairness ensures that, if the system is overloaded, every user's requests will be processed and each user will suffer roughly the same amount from the system degradation.

On the other hand, a real-time system will have critical tasks that must be completed before given deadlines; otherwise the system is at risk of failure. In these systems, the measures of merit are schedulability, ensured worst-case latency, and stability. For example in a nuclear power plant, the tasks that monitor and regulate core temperature might be critical tasks that must be guaranteed to be schedulable, meaning that they are able to meet their given executing deadlines. We must be able to determine that their worst-case latency (that is, their worst-case response time to events, such as temperature changes) must be acceptable. Finally, in the case of overload, the system must continue to meet these tasks' deadlines, even if other, less critical system tasks must be starved out.

The Mathematical Notation of Theorem 2

Theorem 2: n independent periodic tasks scheduled rate monotonically always meet their deadlines for all task phasings if and only if

$$\forall i, 1 \le i \le \mathrm{n}, \min \sum_{\mathrm{j}=1}^{i} C_j \frac{1}{mT_k} \left\lceil \frac{mT_k}{T_j} \right\rceil \le 1$$

$$(k, m)\, \varepsilon \mathrm{R}_i$$

$$\mathrm{R}_i = \{(k, m) \mid 1 \le k \le i, m = 1, \ldots, \left\lfloor \frac{T_i}{T_k} \right\rfloor\}$$

Note that n refers to the number of tasks; C_j is the execution time of task j; and T_k is the length of task k's period.

The theorem tests whether each task within the task set is able to meet its first deadline while accommodating preemption from all higher priority tasks.

Before explaining Theorem 2 in terms of the algebraic notation, let us review what Theorem 2 does by way of the sample problem. To do that, we need to clarify the notion of a scheduling point. We use the term "scheduling point" to mean a point at which new work is scheduled to run. For periodic tasks, the beginning of a task's period would be a scheduling point.

Theorem 2 enumerates each scheduling point and tests to see if all previously scheduled work could have been completed by that point, just before new work is introduced. That is, if we consider only task 1, where its period $T_1 = 50$, and task 2, where its period $T_2 = 120$, one could test task 2's ability to meet its first deadline by checking if all the scheduled work could be completed by elapsed time t = 50 (T1), elapsed time t = 100 (the end of task 1's second period, 2T1), or elapsed time t = 120 (the end of task 2's first period), just before the new work is introduced at those points. Notice that as one progresses from one scheduling point to another, the "scheduled work" increases.

Note that we could check at any point along the timeline (from 0 to the deadline) to see if all the scheduled work has been completed. However, we only check at the scheduling points because the result would change only as new work is introduced.

In the sample problem shown in the video, there are 3 tasks. They have periods of 100 msec, 150 msec, and 350 msec, respectively. The question for Theorem 2 is whether each task is able to complete its execution before its first deadline while accommodating any preemption. Theorem 2 enumerates each scheduling point for a given task and checks if all of the work

already scheduled can be completed before that scheduling point, when new work will be introduced. The proof of Theorem 2 demonstrates that the condition is necessary for schedulability.

In the sample problem, the scheduling points, ordered chronologically, are 100, 150, 200, 300, and 350 — which are T_1, T_2, $2T_1$, $2T_2 = 3T_1$ and T_3, respectively. All 3 tasks are scheduled to run at t = 0. At t =100, task 1 is scheduled to run again. At t = 150, task 2 is scheduled, and so on. If we were to apply Theorem 2 to task i = 3, we would ask whether

$C_1 + C_2 + C_3 \leq T_1$

$2C_1 + C_2 + C_3 \leq T_2$

$2C_1 + 2C_2 + C_3 \leq 2T_1$

$3C_1 + 2C_2 + C_3 \leq 3T_1$ (In this example, $3T_1 = 2T_2$ so the inequality for $2T_2$ is omitted.)

$4C_1 + 3C_2 + C_3 \leq T_3$

If at least one of these is true, we have satisfied the conditions of the theorem for i = 3; that is, for task 3's schedulability.

Let us now take a closer look at Theorem 2 and step through the notation. Each mT_k corresponds to a scheduling point, e.g., $2T_1$ or T_3. If we were applying Theorem 2 to the sample problem of 3 tasks, there would be 3 sets of (k,m) pairs, one set R_i for each task i.

The cases where i = 1 or 2 in the sample problem are less interesting than i = 3, where we are determining the schedulability of the third task. For i = 3, we will see how the conditions generate a set of 6 (k,m) pairs. These provide the subscript and coefficient of each of the scheduling points, as follows. Applying the definition (from Theorem 2) of R_i for *i* = 3, k will take on values from 1 to 3. For each value of k, some number of m's are generated that is equal to the number of times task k is scheduled before task *i* reaches its first deadline. The floor operator finds this number.

- for k =1, m =1,..., $\left\lfloor \frac{T_3}{T_1} \right\rfloor$

  which in this case is 3 because the FLOOR of T_3/T_1

  = FLOOR of 350/100 = FLOOR of 3.5 = 3. So m will range from 1 to 3 while k remains 1 – thus enumerating the (k,m) pairs of (1,1), (1,2), and (1,3). The values for k and m are used for the subscript and coefficient of each scheduling point mT_k. In this case, we have $1T_1$, $2T_1$, and $3T_1$.

- for k = 2: m =1,..., $\left\lfloor \frac{T_3}{T_2} \right\rfloor$

  In our example, FLOOR (350/150) = FLOOR (2.3) = 2. This enumerates the (k,m) pairs (2,1) and (2,2), which yields the scheduling points $1T_2$ and $2T_2$. This is what we would expect, since task 2 is scheduled 2 times while task 3 is attempting to meet its first deadline.

- for k = 3: m =1,..., $\left\lfloor \frac{T_3}{T_3} \right\rfloor$

  which produces the pair (3,1) or $1T_3$, the last scheduling point to check.

Together, the (k,m) pairs generated by Theorem 2 yield all the scheduling points in our sample problem, namely T_1, $2T_1$, $3T_1 = 2T_2$, T_2, $2T_2$, and T_3. It is at these scheduling points that we will check if all the work previously scheduled could have been completed. If there is a scheduling point at which this is true, then we will know that task 3 finished its work before the deadline of t = 350 while accommodating the preemption of tasks 1 and 2.

Concentrating on the inequality expressed in Theorem 2, each (k,m) pair will produce a *sum* of (multiples of) task execution times C_j that should have been completed if all of the required deadlines are to be met by that scheduling point. For example, as we have already seen, the (k,m) pair of (1,2) corresponds to $2T_1$ and produces the sum $(2C_1/2T_1) + (2C_2/2T_1) + (C_3/2T_1)$, which is then checked for being less than or equal to 1. Theorem 2 uses the expression

$$\sum_{j=1}^{i} C_j \frac{1}{mT_k} \left\lceil \frac{mT_k}{T_j} \right\rceil$$

to create that sum. Let's plug in the values and see for ourselves. In this case we have k=1 and m=2. As j goes from 1 to i, which is 3, the expression becomes the following sum:

$$C_1 \frac{1}{2T_1} \left\lceil \frac{2T_1}{T_1} \right\rceil + C_2 \frac{1}{2T_1} \left\lceil \frac{2T_1}{T_2} \right\rceil + C_3 \frac{1}{2T_1} \left\lceil \frac{2T_1}{T_3} \right\rceil$$

and we want to know if that is less than 1. We can multiply through by mT_k, which in this case is $2T_1$, to obtain a simpler form:

$$C_1 \left\lceil \frac{2T_1}{T_1} \right\rceil + C_2 \left\lceil \frac{2T_1}{T_2} \right\rceil + C_3 \left\lceil \frac{2T_1}{T_3} \right\rceil \leq 2T_1$$

Plugging in the numbers for our example in which the period lengths were 100, 150, and 350, we obtain

$$C_1\left\lceil \frac{200}{100} \right\rceil + C_2\left\lceil \frac{200}{150} \right\rceil + C_3\left\lceil \frac{200}{350} \right\rceil \leq 200$$

which reduces to

$$2C_1 + 2C_2 + C_3 \leq 200$$

If we were to perform the above numeric substitutions for the other 5 (k,m) pairs for task 3, i.e., when i = 3, we would produce the 5 inequalities (2 of them are exactly the same) listed earlier in this section. If any of the 5 inequalities is true, we know that there is a scheduling point by which task 3 is able to meet its first deadline, while accommodating preemption from the other tasks. We therefore conclude that task 3 is schedulable.

Thus we see that each (k,m) pair provides the subscript and coefficient of a scheduling point and corresponds to a particular sum involving that scheduling point. Since a given task set will have n sets of (k,m) pairs, there will actually be n sets of sums produced, each involving the scheduling point indicated by the associated (k,m) pair. The “min” in the algebraic expression of Theorem 2 finds the smallest sum for which the inequality holds. It is necessary to be sure only that there is at least one for each task.

Sample Homework Assignments and Exam

Homework Assignment #1

Homework Assignment #1 Solution and Comments

Homework Assignment #2

Homework Assignment #2 Solutions and Comments

Exam

Exam Solutions and Comments

Homework Assignment #1

Consider the following tasks:

Name	Execution time	Period length
X	10	100
P	20	50
S	20	150
G	25	80

1. List them in rate monotonic order with the highest priority first.

In the following problems, when you are asked to "*use Theorem 1*," write the expression algebraically, then write it again plugging in numbers. If you are asked to "*use Theorem 2*," show a timeline (use the blank one provided and label it regarding which numbered problem it is answering) and also state your conclusion specifically. For example, "Theorem 2 is satisfied for task___ at the scheduling point t = ___." If you are asked, "are these tasks *schedulable*?" use Theorem 1 first. If that is sufficient to answer the questions, fine. If not, then use Theorem 2 as described above for each task. If you are asked to use the *extended model* for Theorem 1, use the expression that includes

preemption factors + task + blocking factors ≤ U(n) where you have selected n correctly.

2. Based on this ordering, are the tasks with the two highest priorities schedulable?

3. Are all 4 tasks schedulable? If not, why not?

4. Change the problem as follows: assume that task S executes as an interrupt instead of its rate monotonic priority. Use Theorem 2 to show whether the 4 tasks are schedulable. If not, which task(s) miss(es) the deadline(s)?

5. Based on the change you made in problem #4, change the problem again: assume S is a task that involves 10 units of interrupt processing and 10 units of application processing. Use the extended model for Theorem 1 to determine if this solves the schedulability problem.

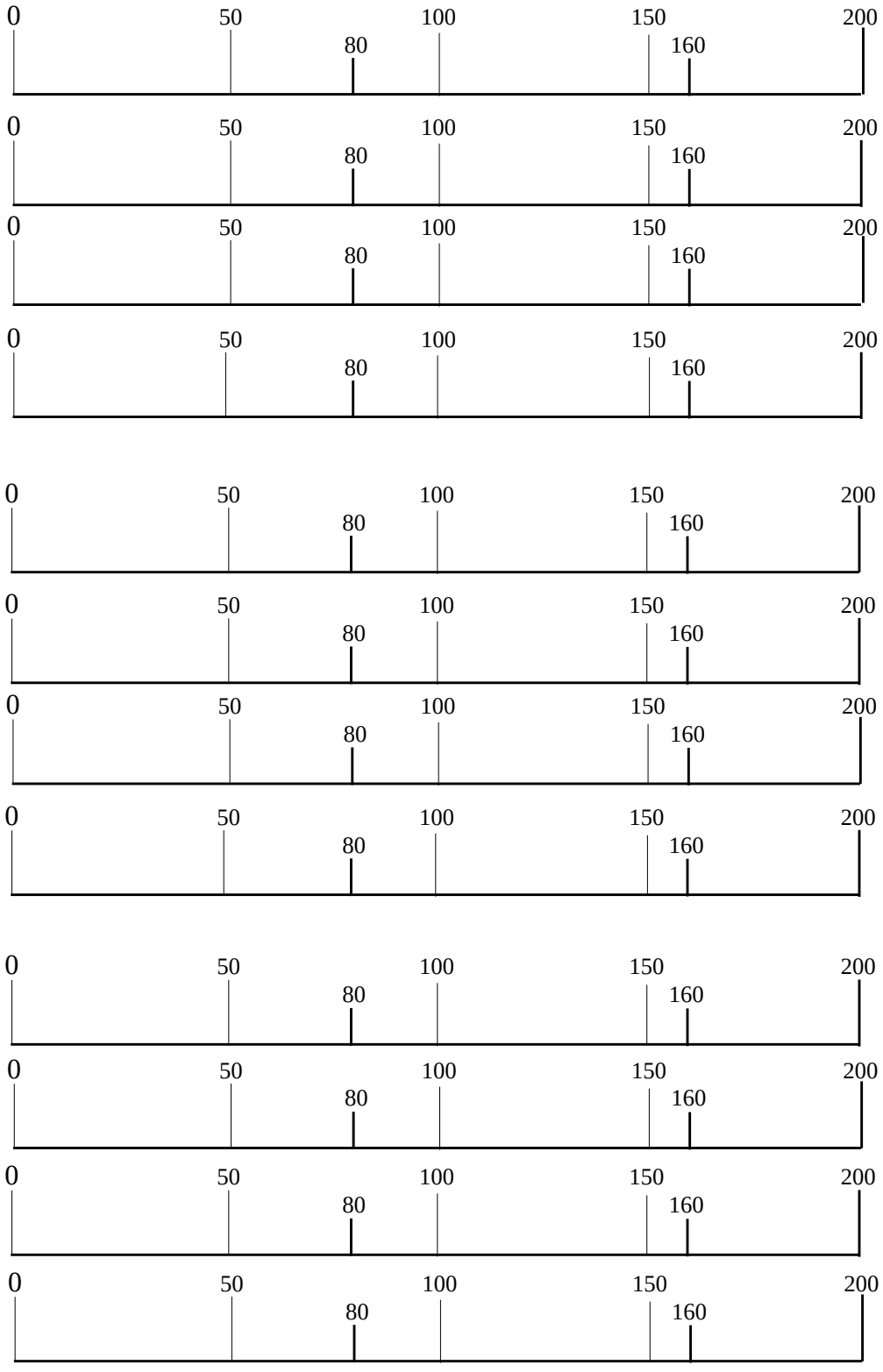
6. Suppose the conditions of #5 hold for S. Where would you first direct your attention in an attempt to make this task set schedulable? That is, which task contributes the most to the fact that this set is not schedulable?

7. What does it mean if a task set of n tasks fails both Theorem 1 and Theorem 2 (assuming there is no room for improvement of execution times) but

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq = 1$$

8. What does it mean if $\sum_{i=1}^{n} \frac{C_i}{T_i} > 1$?

Use the following timelines as needed.



Solutions and Comments
Homework Assignment #1

1. Rate monotonic order is P, G, X, S

2. Task P: Cp/Tp < U(1)
   20/50 = .4 ≤ 1.0 schedulable

   Task G: Cp/Tp + Cg/Tg < U(2)
   20/50 + 25/80 = .713 ≤ .828 schedulable

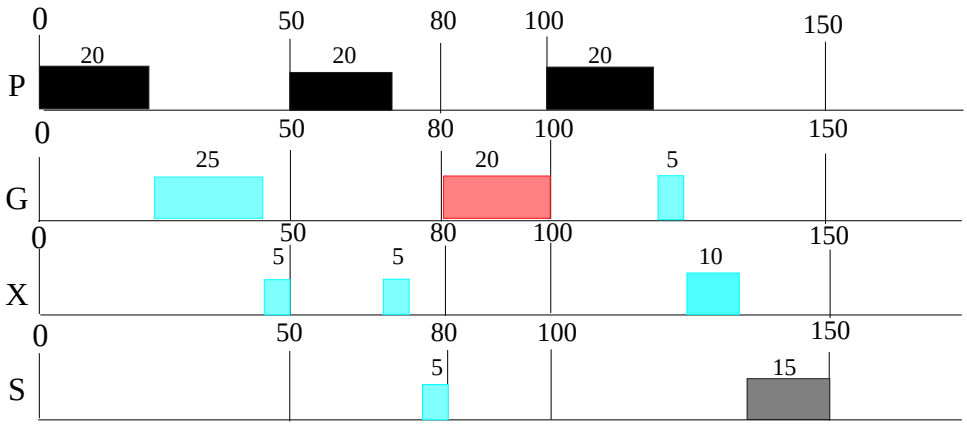
3. Task P: Cp/Tp < U(1)
   20/50 = .4 ≤ 1.0 schedulable

   Task G: Cp/Tp + Cg/Tg < U(2)
   20/50 + 25/80 = .713 ≤ .828 schedulable

   Task X: Cp/Tp + Cg/Tg + Cx/Tx < U(3)
   20/50 + 25/80 + 10/100 = .813 > .779 not schedulable

   Task S: Cp/Tp + Cg/Tg + Cx/Tx + Cs/Ts < U(4)
   20/50 + 25/80 + 10/100 + 20/150 = .947 > .756 not schedulable

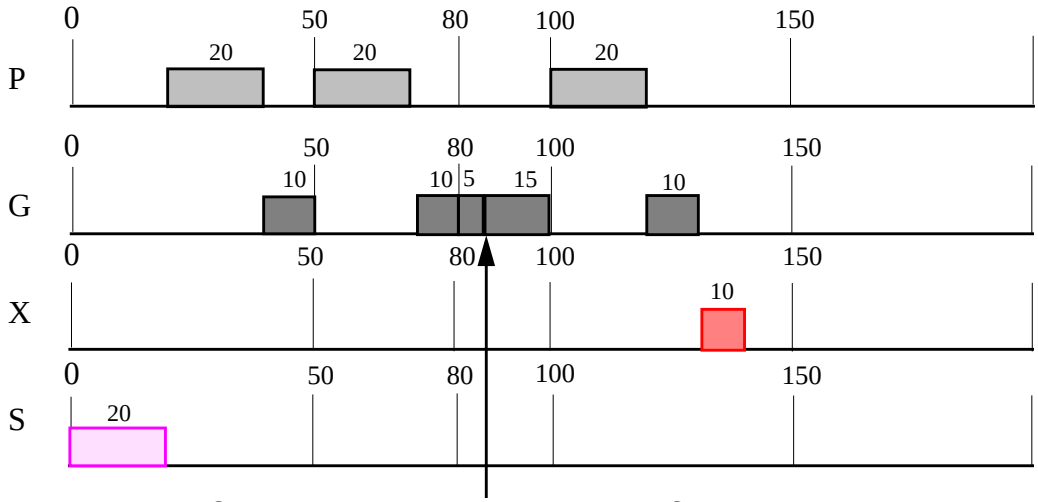
.



Theorem 2 is satisfied at scheduling point T=150
Task X completes execution at t = 135
Task S completes execution at t = 150

Therefore, all 4 tasks are, in fact, schedulable.

4.



Task G completes its first execution at t = 85, missing its first deadline at t = 80
Task X completes its first execution at t = 140, missing its first deadline at t = 100
Therefore, tasks G and X are not schedulable.

5. (Task P) $Cp/Tp + Csi/Tp \leq U(1)$
$20/50 + 10/50 = .6 \leq U(1)$ OK

(Task G) $Cp/Tp + Cg/Tg + Csi/Tg \leq U(2)$
$20/50 + 25/80 + 10/80 = .838 > U(2)$ not OK

(Task X) $Cp/Tp + Cg/Tg + Cx/Tx + Csi/Tx \leq U(3)$
$20/50 + 25/80 + 10/100 + 10/100 = .913 > U(3)$ not OK

(Task S) $Cp/Tp + Cg/Tg + Cx/Tx + Cs/Ts \leq U(4)$ not OK
$20/50 + 25/80 + 10/100 + 20/150 = .946 > U(4)$

6. P due to high utilization, or G if justified based on timeline.

7. There may be an ad hoc solution, but there is no algorithmic solution.

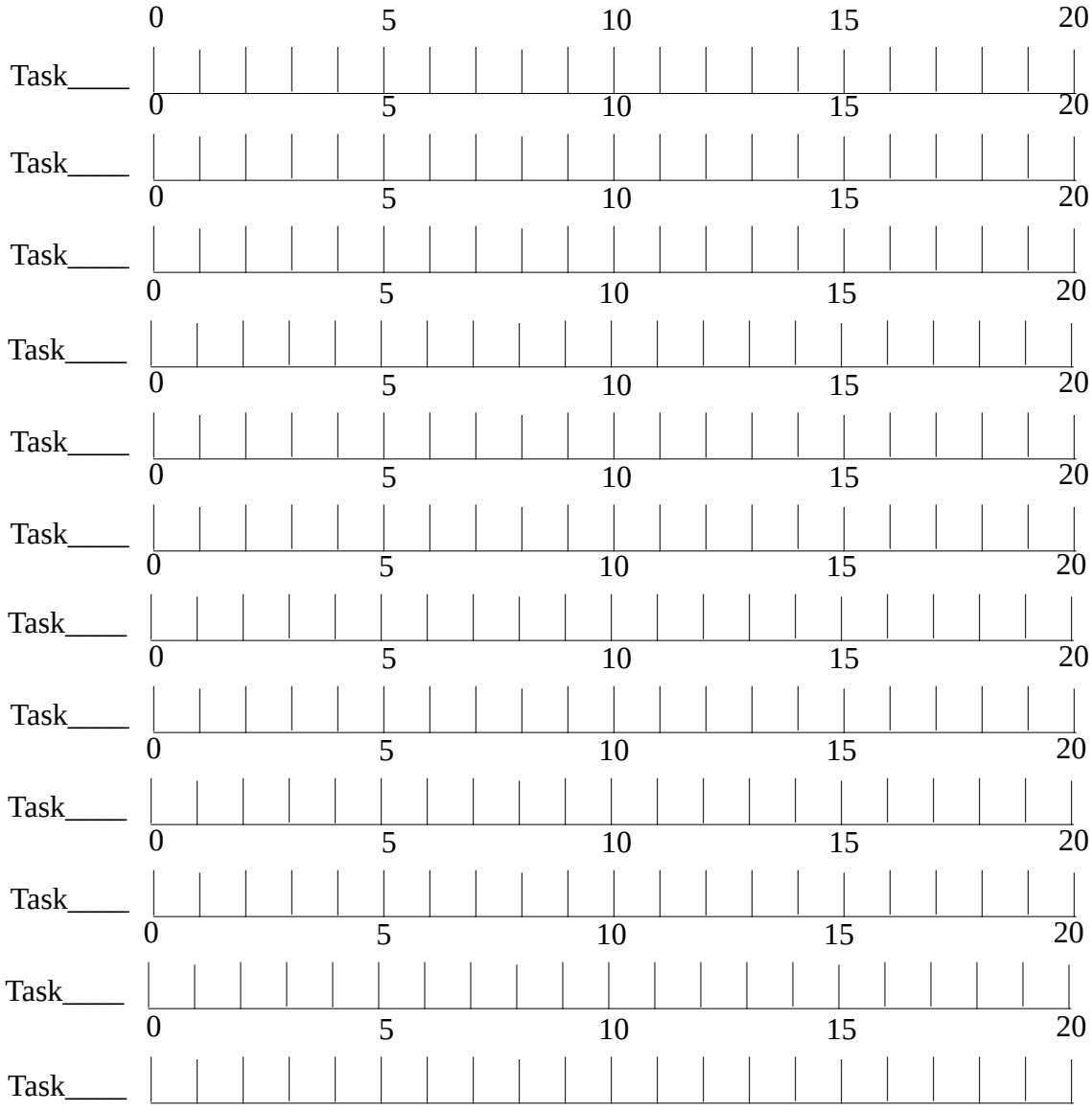
8. The task set is not schedulable.

Comments on Homework Assignment #1

Question #	Comment
1	No comments
2	No comments
3	Although Theorem 1 does not find task X and task S to be schedulable, applying Theorem 2 finds all 4 tasks to be schedulable.
4	When task S runs as an interrupt, it acts as blocking to the other three tasks, and will now appear as a blocking term in the other 3 tasks' schedulability tests. Task P can still meet its deadline, in spite of the blocking. However, task G and task X now fail to meet their deadlines. Note that the Theorem 1 test for task S is now $Cs/Ts \leq U(1)$.
5	When task S is broken up into 10 msec of interrupt processing and 10 msec of application processing, the 10 msec of interrupt processing continue to block the other 3 tasks. However, the 10 msec of application will be preempted by the other 3 tasks. The resulting change is that task G still fails Theorem 1, but now passes Theorem 2. Task X still fails both tests. Task S now fails Theorem 1, but passes Theorem 2.
6	One approach to improve schedulability is to find the tasks that have the highest utilizations, particularly those that impact the rest, and reduce their computation times to reduce total utilization.
7	If Theorem 1 results in a utilization above the bound but less than 100%, we know that there is actually enough CPU capacity to perform all the required computation. However, worst-case task phasings may cause tasks to miss deadlines intermittently.
8	There is too much computation; the CPU does not have the capacity to perform all the work, no matter what scheduling approach is used.

Homework Assignment #2

Please use the timeline templates below for answering timeline questions. Label each timeline carefully so that it is clear which timeline pertains to which question. Cutting up the templates or other modifications are encouraged as long as you present the information clearly.



1. Given the following information about 3 periodic tasks:

Task	Execution Time	Period	Utilization
a	1	5	
b	2	7	
c	1	4	

a) What is the total utilization of the set?

b) Is the set of tasks schedulable?

c) Identify the highest and lowest priority task when using rate monotonic priorities.

d) Assuming that the tasks are assigned rate monotonic priorities, draw the timeline (starting all 3 tasks at time 0).

e) Add 1 unit to the execution time of task b. Calculate the total utilization of the resulting task set. Is the new task set schedulable according to Theorem 1? Draw the timeline for this new task set.

2. Assess the schedulability of the following task set.

Task a:	$C_1 = 1$	$T_1 = 5$
Task b:	$C_2 = 3$	$T_2 = 6$
Task c:	$C_3 = 3$	$T_3 = 14$

3. Given the following tasks:

	C	T	Priority
Task τ_1	10	50	high
Task τ_2	10	75	medium
Task τ_3	40	100	low
Interrupt	15	200	

a) Build a schedulability model for the task set.

b) Given (a), now assume that τ_3 is composed of 2 sections, τ_{3a} and τ_{3b}. Execution times are 30 and 10, respectively. Section τ_{3a} is non-preemptible and non-interruptible. Set up the new schedulability model.

c) Perform the numerical test, Theorem 2, on task τ_1.

d) Consider breaking up the non-preemptible section of τ_3 into 2 or more smaller, non-preemptible blocks of time (for example: $C_{3a\text{-}1}$=10 and $C_{3a\text{-}2}$=20). What is the new schedulability inequality for taskτ_1? What is the impact on the schedulability of taskτ_1?

Solutions and Comments
Homework Assignment #2

1a)

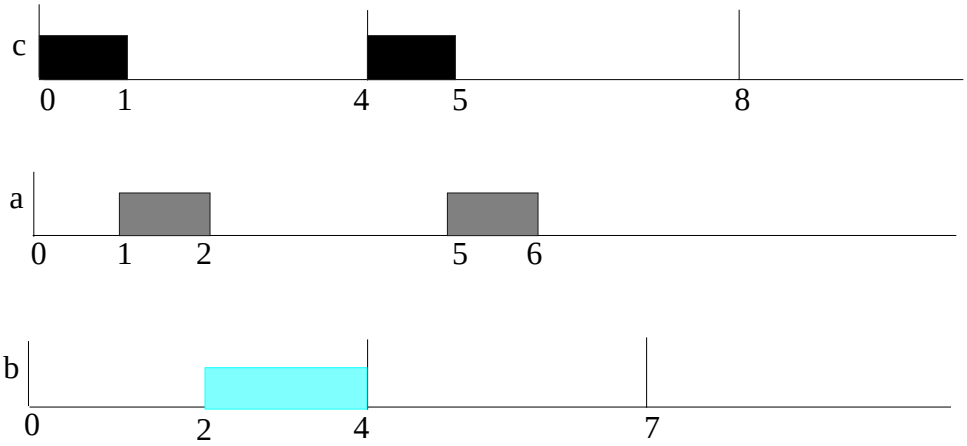
Task a:	C_1=1	T_1=5	U_1=0.2
Task b:	C_2=2	T_2=7	U_2=0.286
Task c:	C_3=1	T_3=4	U_3=0.25

Total utilization: 73.6%

1b) Yes. 73.6% < U(3) = 77.9%

1c) Priority order is c, a, b.

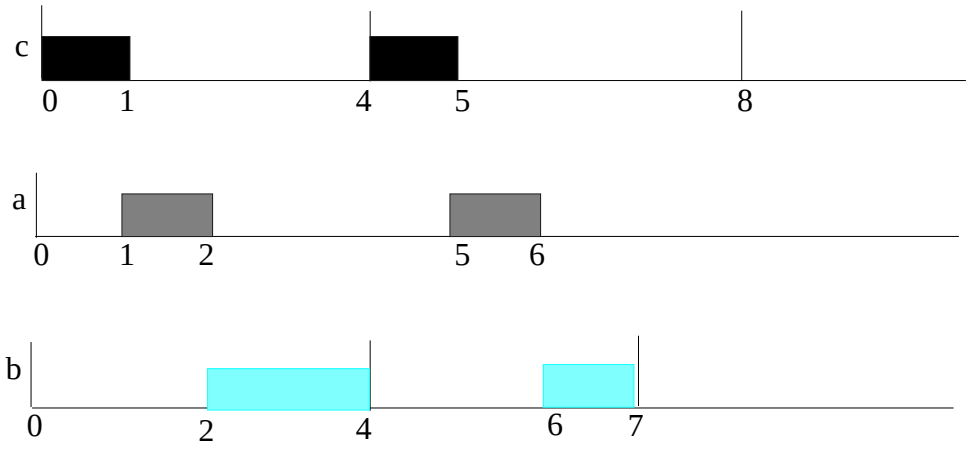
1d)



1e)

Task a:	C_1=1	T_1=5	U_1=0.2
Task b:	C_2=3	T_2=7	U_2=0.429
Task c:	C_3=1	T_3=4	U_3=0.25

Total utilization: 87.9 > U(3) = 77.9%. It is not schedulable by Theorem 1.

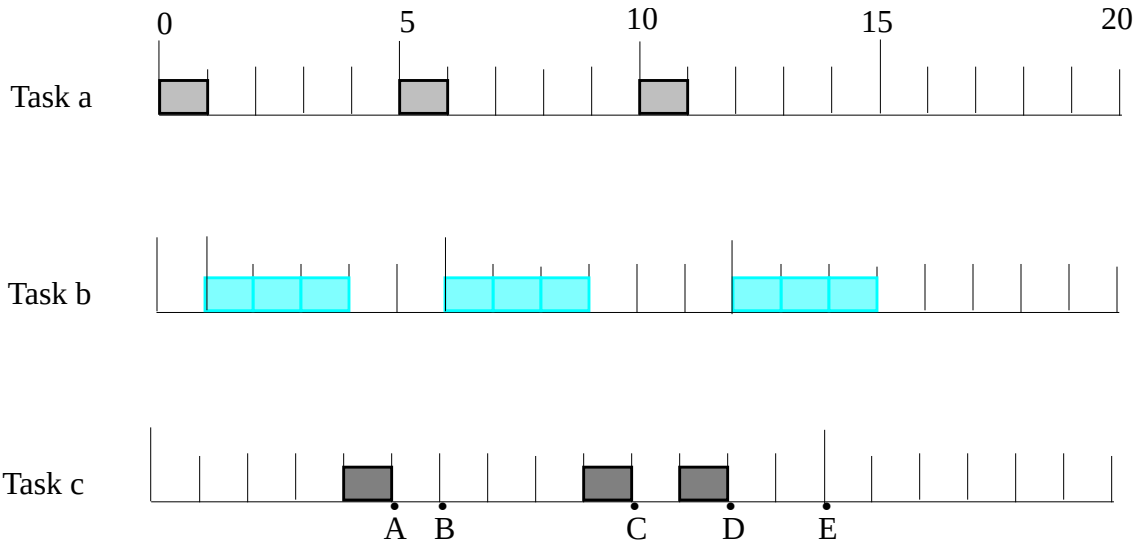


COMMENTS: Note that, although the total utilization factor of the new task set is greater than the bound in Theorem 1, it is still schedulable. The bound in Theorem 1 is sufficient but it is not necessary.

2.

Task a:	C_1=1	T_1=5
Task b:	C_2=3	T_2=6
Task c:	C_3=3	T_3=14

Utilization for tasks a and b is .700 < U(2) = .828
Utilization for tasks a, b, and c is .914 > U(3) = .779
COMMENT: The numerical test is necessary for task c.



COMMENT:

(A)	$1C_1 + 1C_2 + 1C_3 \leq 1T_1$	7 > 5
(B)	$2C_1 + 1C_2 + 1C_3 \leq 1T_2$	8 > 6
(C)	$2C_1 + 2C_2 + 1C_3 \leq 2T_2$	11 > 10
(D)	$3C_1 + 2C_2 + 1C_3 \leq 2T_2$	12 = 12
(E)	$3C_1 + 3C_2 + 1C_3 \leq 1T_3$	15 > 14

If any of the above formulas is true, we can conclude that task c will succeed.
“D” is true, therefore task c is guaranteed to meet its time requirements.

3a) KEY: (P) = Preemption; (B) = Blocking

$$\tau_1 = \frac{C_1}{T_1} + \overset{(B)}{\frac{C_i}{T_1}} \leq 1$$

$$\tau_2 = \overset{(P)}{\frac{C_1}{T_1}} + \frac{C_2}{T_2} + \overset{(B)}{\frac{C_i}{T_2}} \leq 2\left(2^{\frac{1}{2}} - 1\right)$$

$$\tau_3 = \overset{(P)}{\frac{C_1}{T_1}} + \overset{(P)}{\frac{C_2}{T_2}} + \frac{C_3}{T_3} + \overset{(B)}{\frac{C_i}{T_3}} \leq 3\left(2^{\frac{1}{3}} - 1\right)$$

$$\tau_i = \frac{C_i}{T_i} \leq 1$$

COMMENTS:

Under Theorem 1, task τ_3 is not schedulable:

τ_1: .2 + .3 ≤ 1

τ_2: .2 + .133 + .2 ≤ .828

τ_3: .2 + .133 + .4 + .15 ≤ .799 FAILS TEST

τ_4: .075 ≤ 1

However, under Theorem 2, τ_3 is schedulable:

τ_3: $C_1 + C_2 + C_3 + C_i \leq 50$ 10 + 10 + 40 + 15 > 50

$2C_1 + C_2 + C_3 + C_i \leq 75$ 20 + 10 + 40 + 15 > 75

$2C_1 + 2C_2 + C_3 + C_i \leq 100$ 20 + 20 + 40 + 15 < 100 True

Task τ_3 completes its work before the end of its first period and thus is schedulable.

3b)

$$\tau_1: \ \overset{(B)}{\frac{C_1}{T_1}} + \overset{(B)}{\frac{C_i}{T_1}} + \frac{C_{3a}}{T_1} \ \leq \ 1$$

$$\tau_2: \ \overset{(P)}{\frac{C_1}{T_1}} + \frac{C_2}{T_2} + \overset{(B)}{\frac{C_i}{T_2}} + \overset{(B)}{\frac{C_{3a}}{T_2}} \ \leq \ 2\,(2^{\frac{1}{2}} - 1\,)$$

$$\tau_3: \ \overset{(P)}{\frac{C_1}{T_1}} + \overset{(P)}{\frac{C_2}{T_2}} + \frac{C_3}{T_3} + \overset{(B)}{\frac{C_i}{T_3}} \leq 3\,(2^{\frac{1}{3}} - 1\,)$$

$$\tau_i: \ \overset{(B)}{\frac{C_i}{T_i}} + \frac{C_{3a}}{T_i} \ \leq \ 1$$

COMMENTS:

The task set is not schedulable under Theorem 1:

τ_1: .2 + .3 + .6 > 1	FAILS TEST
τ_2: .2 + .133 + .2 + .4 > .828	FAILS TEST
τ_3: .2 + .133 + .4 + .15 > .779	FAILS TEST
τ_i: .075 + .15 ≤ 1.0	

Notice that, although the period of τ_3 is shorter than that of τ_i, τ_{3a} is modeled as blocking rather than preemption. This is because τ_{3a} can block τ_i once at most during τ_i's period.

3c) τ_1: $C_1 + C_1 + C_{3a} \leq 50$ FAILS TEST

COMMENTS:

Task τ_1 also fails the numerical test:

$$10 + 15 + 30 > 50$$

τ_1 is unable to meet its first deadline under worst-case conditions and is, therefore, still not guaranteed to be schedulable.

Tasks τ_2 and τ_3, however, do pass the numerical test, completing their work under worst-case conditions at or before their first deadlines:

τ_2: $C_1 + C_2 + C_i + C_{3a} \leq 50$	$10 + 10 + 15 + 30 > 50$
$2C_1 + C_2 + C_i + C_{3a} \leq 75$	$20 + 10 + 15 + 30 = 75$
τ_3: $C_1 + C_2 + C_3 + C_i \leq 50$	$10 + 10 + 40 + 15 > 50$
$2C_1 + C_2 + C_3 + C_i \leq 75$	$20 + 10 + 40 + 15 > 75$
$2C_1 + 2C_2 + C_3 + C_i \leq 100$	$20 + 20 + 40 + 15 < 100$

3d)

$$\tau_1: \frac{C_1}{T_1} + \frac{C_1^{(B)}}{T_1} + \frac{\text{MAX}\,(C_{3a-1}^{(B)} C_{3a-2})}{T_1} \leq 1$$

COMMENTS:

Although perfect preemption may not be possible, “partial” preemption can be achieved by breaking up non-preemptible code into several smaller segments of non-preemptible code.

Using this approach in this problem, if τ_1 is blocked by $\tau_{3a\text{-}1}$, it will still have the opportunity to execute before $\tau_{3a\text{-}2}$ since it has higher priority.

At worst, τ_1 will be blocked by the longest of the smaller non-preemptible segments. Therefore, the blocking term in the schedulability test for task τ_1, caused by task τ_3, is the MAX (or the longest) of τ_3’s non-preemptible segments.

Now, task τ_1 passes the schedulability test and is guaranteed to meet its deadlines:

$$.2 + .3 + .4 < 1$$

Exam

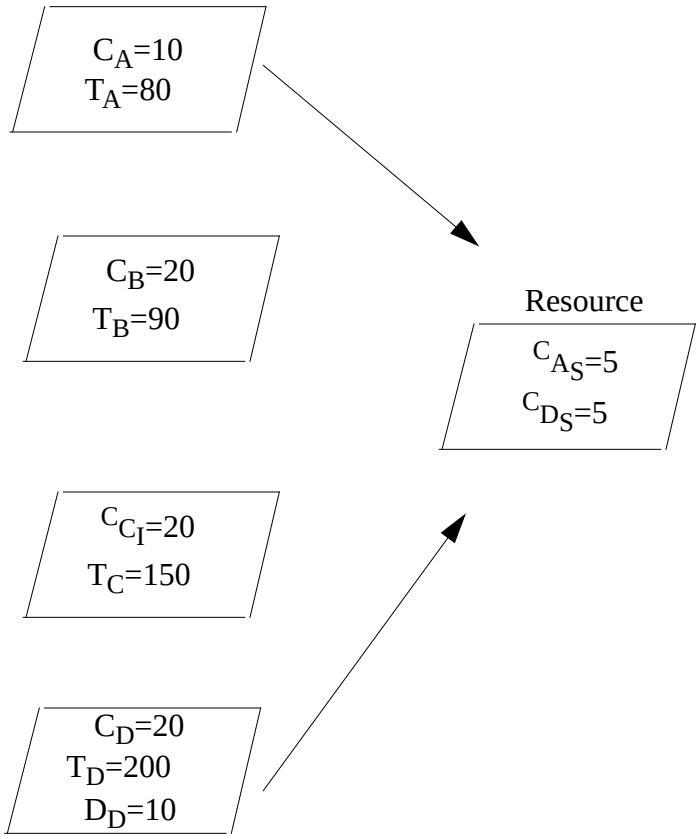
Consider the following set of tasks with the attributes shown. Use the expanded (extended) schedulability model to determine if each task is schedulable. Use a clear algebraic notation to represent what you are interpreting as preemption factors, the task itself, and blocking factors before "plugging in numbers." Use a timeline to see if Theorem 2 gives you additional information. Label it clearly.

Task A: C_A =10 T_A=80	The last 5 msec of execution requires exclusive access to a resource that is shared with task D.
Task B: C_B=20 T_B=90	
Task C: C_C=20 T_C=150	Interrupt-driven, all processing done by interrupt service routine (not preemptible).
Task D: C_D=20 T_D=200	Executes for 10 msec; then requires 5 msec of exclusive access to resource that is shared with task A; then frees resource and continues for 5 msec more. Task D has a preperiod deadline of 10 msec.
Task E: $^{C}E_I$=5 $^{C}E_A$=15 T_E=250	Interrupt-driven. Interrupt processing is 5 msec and application processing is 15 msec.
Task F: C_F=30 aperiodic	The aperiodic events that take 30 msec to process can occur every 300 to 500 msec.

Note: Unless otherwise stated, assume deadlines are at the end of the period. Also assume that all application-level processing is assigned priorities rate monotonically.

Solution to Exam Question

Diagram of Tasks and Interaction with Shared Resources



C_{E_I}=5
C_{E_A}=15
T_E=250

C_F=30
T_F=300
(worst case)

Note:

C_X is computation time for task X

T_X is period of task X

C_{X_I} is computation time of task X that runs at an interrupt priority

C_{X_A} is computation time of task X that runs at an application-level priority

C_{X_S} is computation time during synchronization with a shared resource (i.e., critical section of task X)

D_X is task X's preperiod deadline

Assuming that all application-level processing is prioritized rate monotonically:

E = Execution
B = Blocking
P = Preemption

TASK A

$$\overbrace{\frac{C_A}{T_A}}^{E} + \overbrace{\frac{C_B}{T_A} + \frac{C_{C_I}}{T_A} + \frac{C_{D_S}}{T_A} + \frac{C_{E_I}}{T_A}}^{B} \leq 1$$

$$\frac{10}{80} + \frac{20}{80} + \frac{5}{80} + \frac{5}{80} = \frac{40}{80} = 0.5 \qquad \text{OK}$$

TASK B

$$\overbrace{\frac{C_A}{T_A}}^{P} + \overbrace{\frac{C_B}{T_B}}^{E} + \overbrace{\frac{C_{C_I}}{T_B} + \frac{C_{E_I}}{T_B}}^{B} \leq 0.828$$

$$\frac{10}{80} + \frac{20}{90} + \frac{20}{90} + \frac{5}{90} = \frac{10}{80} + \frac{45}{90} = 0.125 + 0.5 = 0.625 \qquad \text{OK}$$

TASK C

$$\overbrace{\frac{C_{C_I}}{T_C}}^{E} + \overbrace{\frac{C_{E_I}}{T_C}}^{B} \leq 0.779$$

$$\frac{20}{150} + \frac{5}{150} = \frac{25}{150} = 0.167 \qquad \text{OK}$$

TASK D

$$\overbrace{\frac{C_A}{T_A}+\frac{C_B}{T_B}+\frac{C_{C_I}}{T_C}}^{P}+\overbrace{\frac{C_D}{T_D}}^{E}+\overbrace{\frac{D_D}{T_D}+\frac{C_{E_I}}{T_D}}^{B}\le 0.756$$

$$\frac{10}{80}+\frac{20}{90}+\frac{20}{150}+\frac{20}{200}+\frac{10+5}{200} = 0.125+0.223+0.134+0.10+0.075 = 0.657$$ OK

TASK E

$$\overbrace{\frac{10}{80}+\frac{20}{90}+\frac{20}{150}+\frac{20}{200}}^{P}+\overbrace{\frac{20}{250}}^{E} = 0.125+0.223+0.134+0.1+0.08 =$$

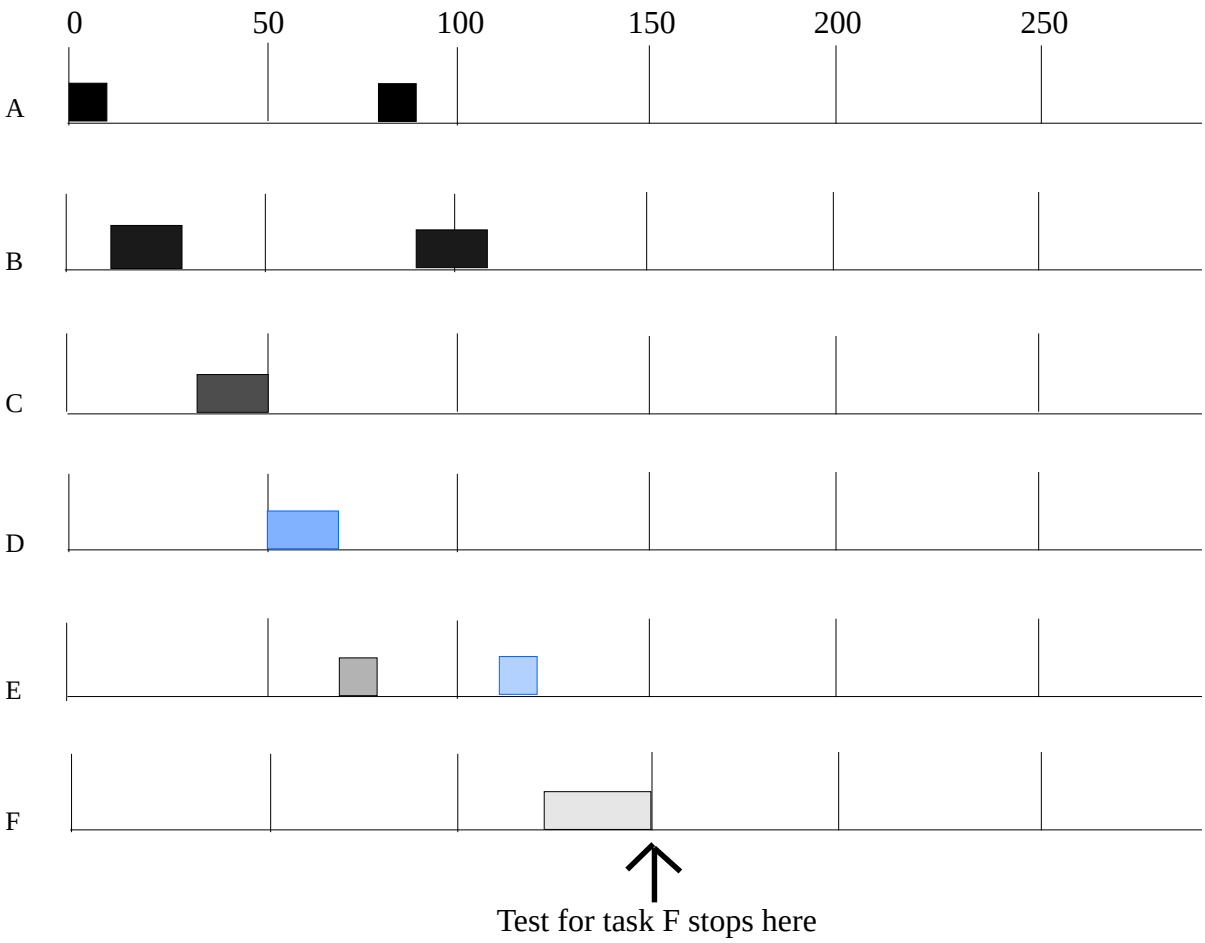
$0.662 \le 0.743$ OK

TASK F

$$\overbrace{\frac{10}{80}+\frac{20}{90}+\frac{20}{150}+\frac{20}{200}+\frac{20}{250}}^{P}+\overbrace{\frac{30}{300}}^{E} =$$

$0.125+0.223+0.134+0.1+0.08+0.1 = 0.762 > 0.734$ NOT OK

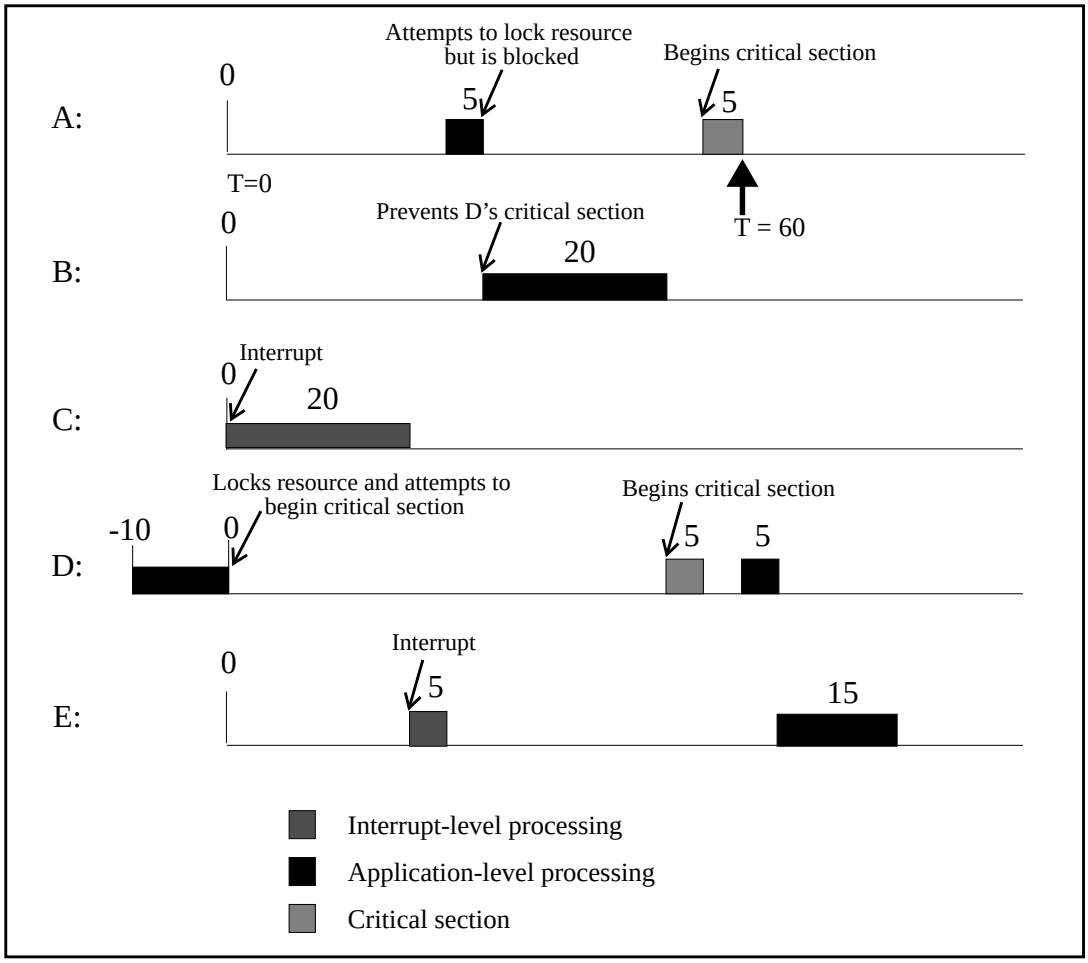
To F, all tasks look like preemption



F completes its 30 msec of execution at t = 150, which is before its first deadline of 300 msec. Therefore, it is schedulable.

Notice if you did Theorem 2 for A:

Worst case is not when they all line up at T=0. The worst case is when D has just started its critical section before A starts running. Then C and E are able to interrupt task A's processing, and B blocks while A is in its critical section.



Worst-case completion for A equals

$$\underbrace{C_A}_{E} + \underbrace{C_B + C_{CI} + C_{DS} + C_{EI}}_{B} = 10 + 20 + 20 + 5 + 5 = 60$$

Comments on Exam Question

To determine the schedulability of the entire task set, because there are sources of blocking, we will have to test each task individually. We will produce a schedulability model (or, a schedulability inequality) for each task. All interrupt-level processing will be included as blocking to higher priority tasks. But remember that, to lower priority tasks, interrupt processing is simply preemption. For example, task C's processing is blocking to task A, but preemption to task D.

The diagram gives the task characteristics and shows the synchronization of task A with task D in accessing the resource. If we assume that no special synchronization protocol is used, we can say the following rules apply:

- The task with the highest priority in the ready queue takes the CPU.
- A task that has grabbed a resource and is in its critical section may be preempted by a higher priority task that does not need the resource.
- If a task gets blocked when trying to grab a resource that is already locked, it is placed on the wait queue until the scheduler is signaled that the resource is free (at which time it is again placed on the ready queue).
- While that task is in the wait queue, the task with the next highest priority in the ready queue gets the CPU.

Following these assumptions, it is possible for task D to enter its critical section, then get preempted by task A until task A tries to use the resource, at which time it is blocked. Task A will be put on the wait queue until task D finishes its critical section. If in the meantime task B or task C is ready to run; it will preempt task D's execution in the critical section and take away the CPU. With this consideration, task B and task C will not experience blocking from task D. Only task A can experience blocking caused by the synchronization.

Note that several operating systems and runtimes follow a no-preemption protocol for synchronization. That is, when a task uses a resource and enters a critical section, it cannot be preempted by another task, even of higher priority. That execution time in the critical section would apply to all tasks of higher priority, even those that do not use the resource. The solution given assumes that the operating system allows preemption, as well as interrupts, during synchronization (that is, while in a task's critical sections).

However, several synchronization protocols exist, such as the priority inheritance protocol and the priority ceiling protocol, which can reduce the blocking effect on higher priority tasks. For more information on these protocols, see the *A Practitioner's Handbook for Real-Time Analysis* or the papers listed in the bibliography that cover them.

Let us now set up the schedulability model for each task.

TASK A:

Task A must accommodate its own execution, plus the following sources of blocking:

- Interrupt-driven processing: all of task C, and interrupt-driven portion of task E.
- Critical sections of tasks that share a resource: critical section of task D
- Processing of medium-priority tasks that do not share the resource: all of task B.

Notice that the same situation exists here that was in the airline tracking system described in the RMA video. While task A is blocked out by task D's execution in the shared resource, it could be further delayed by preemption from task B. Consider a hypothetical case where there are several medium-priority tasks like task B that can take turns delaying task D's critical section. This would be a potential unbounded priority inversion.

TASK B:

Task B must accommodate preemption from task A, its own execution, plus blocking from tasks C and E.

TASK C:

Because task C runs at interrupt level, it essentially runs at a higher runtime priority than any other task. Only task E's interrupt-level processing could delay task C, if task E's interrupt-level processing started before task C started. Therefore task C must accommodate, in the worst case, its own execution plus the blocking from task E.

TASK D:

To task D, task C looks like a preemptor. Therefore task D accommodates preemption from tasks A, B, and C. Then one must add in its own execution, plus blocking from task E's interrupt processing. Remember that task D's preperiod deadline is tacked on as additional blocking. Also notice that the critical section of task A is already accounted for in all of task A's execution time, which can preempt task D.

TASK E:

Task E's schedulability model is straightforward. It must accommodate preemption from all of the higher priority tasks and its own execution. It experiences no blocking.

TASK F:

No requirements for response time for aperiodic events is given, so we assume that computation must be completed by the occurrence of another aperiodic event. If we assume that events occur at worst-case intervals (that is, every 300 msec), we can analyze the task as a periodic task with a period of 300. Therefore it is scheduled as the lowest priority task, and it must accommodate preemption from all other tasks plus its own execution time. This is the only task that fails Theorem 1. However applying Theorem 2 shows it to be schedulable.

Therefore all tasks in this task set are schedulable.

One final point to notice is on the issue of worst-case line-up of task executions. We find the worst case by the set of circumstances that would allow all the preemption and blocking included in a task's schedulability formula. In this particular example, the worst case for task A is not when all tasks are ready to execute at t=0. Worst case for task A is when D has completed the first part of its processing and has just started its critical section; then task A is ready to run, but gets blocked by task C and task E's interrupt processing. After those are completed, task A will still have to wait for task D to finish its critical section, which in the meantime can be preempted by task B. This situation can be seen in the timeline at the end of the solution.

Order Form for Supporting Materials to: Rate Monotonic Analysis for Real-Time Systems

Use this form to order the teaching materials that support this EM: the required videotape, *An Introduction to Rate Monotonic Analysis*; the optional diskette with presentation slides and non-supported RMA programs

To receive the videotape, as well as the optional diskette, complete this form and return it with payment to

Education Program
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Checks should be made payable to **Carnegie Mellon University**.

☐ Videotape *An Introduction to Rate Monotonic Analysis*	$50
☐ Macintosh Diskette with slides and RMA programs (MOA required)	$10
Total enclosed	____

Send to

Name _____
Title _____
School/Company _____
_____
Address _____
_____
City/State/Zip _____
Telephone _____
e-mail _____

MEMORANDUM OF AGREEMENT

1. I/ we the undersigned, on behalf of the Requesting Organization listed below (hereafter referred to as the "Requester"), request release of USAF software and understand and agree to the following:

a. NON-DISCLOSURE AGREEMENT (Applies to commercial components with limited or restricted rights accessed through Government reuse libraries). The Requester requests some or all of the following from the Software Engineering Institute: data, technical data, computer software computer software documentation, computer programs, source code, firmware, and other information of like kind, type or quality, either commercial or non-commercial, all of which may be subject to limited rights, restricted rights, Government purpose license rights, patents, copy rights, trade secret rights, or other confidential or proprietary constraints (collectively, the "Data"). In consideration therefore, the Requester agrees:

1) That the Data shall be used only for Government, non-commercial or non-profit purposes;

2) To strictly abide by and adhere to any and all restrictive markings placed on the Data, and the Requester shall not knowingly disclose or release the Data to third partied who are not engaged in work related to Government, non-commercial, or non-profit purposes;

3) That any restrictive markings on the Data shall be included on all copies, modifications, and derivative works, or any parts or portions thereof, in any form, manner or substance, which are produced by the Requester including but not limited to incorporation of the Data into any other data, technical data, computer software, computer software documentation, computer programs, source code, or firmware, or other information of like kind, type or quality. In all such events, Requester shall clearly denote where such Data initiates and concludes by use of annotations or other standard markings.

b. WAVER OF WARRANTIES AND LIMITATIONS OF DAMAGES AGREEMENT. The Requester and the Approving Authority agree that:

1) No guarantees, representations, or warranties either expresses or implied shall be construed to exist in any language, provision, or term contained in these materials or in any other documentation provided herewith (all such items are collectively referred to as the "Agreement"), and furthermore, the releasing organization disclaims and the Requester waives and excludes any and all warranties of merchant ability and any and all warranties of fitness for any particular purpose;

2) The Requester shall obtain from the releasing organization all of the "Data" (Defined in the Non-Disclosure Agreement above), or any other products or services contemplated by the Agreement, in an "as is" condition.

c. The Requester's use of the Data shall not prevent the Government from releasing the Data at any point in the future.

d. The Requester shall not offer the released Data or any modified version thereof for resale to the Government, in whole or as part or subpart of a Government deliverable, without explicitly stating that he is doing so by providing certification documentation (e.g., Section K of the Government Solicitation) to the contracting officer before contract award.

e. The Requester may use the released Data in a contract with the Government, but understands that the Government shall not pay the Requester for rights of use of such Data in performance of Government contracts or for the later delivery to the Government of such Data. The Requester may be entitled to compensation for converting, modifying, or enhancing the Data into another form for reproduction and delivery to the Government, if authorized under a contract with the Government.

f. The Requester is not entitled to any released Data that are subject to national defense security classification or the proprietary rights of others. The Requester shall report promptly the discovery of any such restricted Data to the USAF release approving authority below, and will follow all instructions concerning the use, safeguarding, or return of such Data. The Requester shall not copy, or make further study or use of, any released Data later found to be subject to such restrictions.

g. As required, the Requester shall be responsible for compliance with any proscriptions on foreign disclosure of the released Data (contained, for example, in the Department of State International Traffic in Arms Regulations or the Department of Commerce Export Administration Regulations).

h. There may be a fee to cover the copying and shipping of the Data and any documentation.

i. The Requester and the Approving Authority intend that all agreements under this Memorandum of Agreement shall be governed by the laws of the United States of America.

Name of Requester	Name / Title of USAF Approving Authority
Requesting Organization / Address	Air Force Organization / Address
City, State, Zip Code	City, State, Zip Code
Signature of Requester and Date	Signature of USAF Approving Authority and Date