中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-06-003

A Scarce Resource Model for Medication Scheduling

P.H. Tsai, H.C. Yeh, P.C. Hsiu, C.S. Shih, T.W. Kuo and J.W.S. Liu

April 6, 2006 || Technical Report No. TR-IIS-06-003

http://www.iis.sinica.edu.tw/LIB/TechReport/tr2006/tr06.html

Institute of Information Science, Academia Sinica

Technical Report TR-IIS-06-003

A Scarce Resource Model for Medication Scheduling

P. H. Tsai, H. C. Yeh, P. C. Hsiu, C. S. Shih, T. W. Kuo and J. W. S. Liu*

Abstract

This report describes a resource model for medication scheduling. It is called the General Medication Specification-Scarce Resource (GMS-SR) model. The GMS-SR model assumes that the medication dispenser has sufficient power and capacities to send reminders, dispense medications, and monitor/record medication histories on a timely basis. The user is a scarce resource, however. The dispenser must take into account contention for the user when it dispenses medications. The GMS-SR model puts medication scheduling in the rigorous framework of real-time scheduling. Because medication scheduling algorithms based on the model resemble real-time scheduling algorithms, some concepts and guidelines in real-time scheduling are readily applicable.

* P. H. Tsai is affiliated with Department of Computer Science, National Tsing Hua University, Taiwan. H. C. Yeh, P. C. Hsiu, C. S. Shih and T. W. Kuo are affiliated with Department of Computer Science and Information Science, National Taiwan University, Taiwan. J. W. S. Liu is affiliated with Institute of Information Science, Academia Sinica, Taiwan.

Table of Contents

1 Introduction

Pillboxes and medication dispensers such as the ones described in [1, 2] are devices designed to ease the effort and improve the chance in medication compliance. A smart *medication dispenser* reminds its user at times when medications should be taken and controls their dosages as it administers the medications. When configured by the user to do so, the dispenser monitors user actions and records user's medication history. It also can provide the user with warnings or notify a caretaker of the user's choice when it detects instances of non-compliance.

In a recent technical report [3], we proposed the General Medication Schedule (GMS) model as a basis for specifying requirements and constraints derived from directions of medications taken by individual users. A smart medication dispenser computes schedules of dispenser commands based on a GMS specification for its user. By executing the commands according to the schedules, the dispenser deliveries reminders and dispenses medications in conformance with directions. The rules embedded in the specification provide the dispenser with criteria needed for compliance monitoring and enforcement. The GMS specification also captures information on user preferences and daily routines. The dispenser tries to dispense medications in time intervals preferred by the user and fits the medication schedule in the user's lifestyle and daily routines whenever possible.

A basic assumption of the GMS model in [3] is that there is no scarce resource: The dispenser not only has sufficient resources to carry out all of its tasks in time. But also, it assumes that the user is always available. Hence, the controller does not need to be concerned with resource contentions when making scheduling decisions. Hereafter, we call this model the *GMS-RR* (*Rich Resource*) model. A negative aspect is that scheduling based on the RR model is by and large *ad hoc*. There are no simple rules, similar to schedulability conditions used to check feasibility of real-time applications [4-9], to guide the choice of dosage parameters from the given ranges of direction and user preference parameters. Whether there are feasible schedules (i.e., schedules that meet all the constraints given by the GMS specification) can be determined only by exhaustive search. The search can take too much time when the user takes a large number of interacting medications and the direction parameters of the medications have wide ranges. Time consuming searches handicap the dispenser's ability to dynamically adjust the medication schedule in reaction to unpredictable user behavior.

A more natural alternative is to think of the user as a scarce resource. Because some medication and/or food remain in effect within the user, the user is not available to take new doses of some medications for safety reasons. The resource model described in this paper takes this view. We call this variant of the model the *GMS-SR* (*Scarce Resource*) *model* (or *SR model*

for short). Like the RR model, the SR model also assumes that the dispenser has sufficient power and capacities to send reminders, dispense medications, and monitor/record medication histories on a timely basis. In other words, the dispenser remains to be resource rich. The user is a scarce resource, however. The dispenser must take into account contention for the user when it dispenses medications. To a great extent, the SR model enables us to take a more systematic approach to medication scheduling. Scheduling algorithms based on the SR model resemble real-time scheduling algorithms [4-12] in many respects. Some concepts and guidelines in real-time scheduling are readily applicable.

In the remainder of the report, Section 2 describes elements of the SR model and compares and contrasts the model with real-time workload models. Section 3 presents a family of scheduling algorithms to illustrate how the SR model supports decisions in medication scheduling. Section 4 defines performance measures used to evaluate medication schedules and scheduling algorithms and discusses the applicability of well-known schedulability conditions and tests. Section 5 discusses additional families of scheduling algorithms based on the SR model and outlines future work to compare the algorithms in this family with each other and with algorithms based on the GMS-RR model. Appendix A provides additional illustrative examples. Many terms and notations used here were introduced in [3]; they are listed in Appendix B.

2 Scarce Resource Model

In the GMS-SR model, there are two types of virtual resources: processors and resources. There is a processor P_M for every medication M taken by the user. If M interacts with some other medication of the user or with food, then there is also a resource R_M associated with M. The virtual resources are abstractions maintained by the dispenser controller. The controller manages their usage as if they were physical processors and resources; hence, they are named such.

2.1 Dispensing and Constraint-Enforcement Jobs

As in [3], we refer to the following sequence of steps carried out by a smart dispenser to administer one dose of a medication M as a *dispensing job* of M:

1. Send a reminder to tell the user it is time to take the medication,
2. Wait, and adjust the schedule if necessary, until the user comes and retrieves the medication,
3. Mark the time of the retrieval,
4. Check for compliance and take specified action(s) when non-compliance is detected,
5. Record the time of retrieval, (user) promptness and other statistics for later use.

The dispensing job *starts* when Step 1 starts and *completes* when Step 3 completes. The amount of time the user takes to complete Step 2 is called user *promptness*. Key assumptions are that the dispenser always completes Steps 1, 3, 4 and 5 in time small compared with promptness and that subsequent dispensing jobs can start before Steps 4 and 5 of previous jobs complete.

Relationship between Jobs There is a *constraint-enforcement* (*CE*) *job* corresponding to each dispensing job. CE jobs are also abstract entities of the resource model. Unlike dispensing jobs, they have no code, do not require any physical resource, and do no actual work. The dispenser controller uses them to track of the availability of the user to doses of medications and to enforce timing constraints in scheduling dispensing jobs.

Figure 1 illustrates the temporal relationship between dispensing and CE jobs. In the figure and hereafter, we consistently use the term dispensing jobs to mean the actual work done by the dispenser and the user in the five steps listed above. We use the simple term *jobs* to mean the corresponding CE jobs when there is no ambiguity. The notations $J_M(i)$ and $DJ_M(i)$ for $i = 1, 2, \ldots$ refer to the individual CE and dispensing jobs, respectively, of medication M, while the notation $J_M(d, i)$ (or $DJ_M(d, i)$) refers to the ith job (or dispensing job) of M in day d. A dispensing job $DJ_M(i)$ starts when the dispenser sends a reminder to the user to take a dose of M. The dispensing job completes when the user retrieves the dose, and as the figure shows, that is when the corresponding CE job $J_M(i)$ starts.

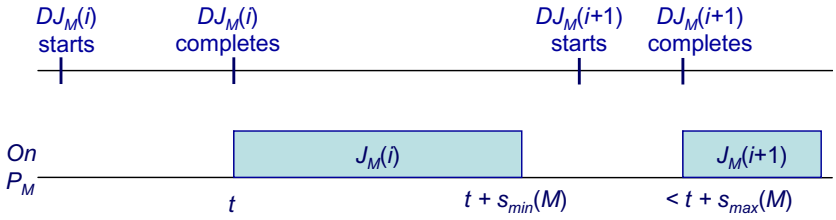

Figure 1 Relation between dispensing and CE jobs

Each CE job of M is scheduled non-preemptively on P_M. The job *completes* at the end of the time interval allocated to it. The *current* job at any time x refers to the job of M that has started and is not yet complete at x, if there is such a job at the time. If there is no such job, the current job of M is the job with the earliest start time among all the jobs of M that start at or after x.

Relative and Absolute Deadlines We say that a CE job is *ready* for scheduling when the controller has the information needed to schedule it (i.e., to compute its start time). The first job of each medication M is scheduled to start at some specified time or on best effort basis.

The start times of subsequent jobs of M are constrained by the following three types of relative and absolute deadlines:

- *Inter-stream separation deadline:* By definition, the relative deadline of a job $J_M(i)$ of M with respect to the previous job $J_M(i\text{-}1)$ of M is equal to the absolute maximum separation $S_{max}(M)$. Therefore, the absolute deadline for the start of $J_M(i)$ in order to maintain correct maximum separation between doses of M is equal to $S_{max}(M)$ plus the start time of the previous job $J_M(i\text{-}1)$.

- *Effective minimum intake deadline*: The minimum total intake constraint of the medication M may impose a deadline on a job $J_M(i)$ relative the beginning of an intake interval [3]: To avoid violation of this constraint, $J_M(i)$ may be required to start by the end of the current minimum intake interval[1]. In general, the effective absolute minimum intake deadline of the job may be sooner than the end of the current intake interval. Let n (≥ 0) be the number of jobs of M that start after $J_M(i)$ and must also start by the end of the current intake interval in order to meet the minimum intake constraint. The effective absolute minimum intake deadline of $J_M(i)$ is the end of the current intake interval minus n times the nominal minimum separation of M.

- *Inter-stream separation deadline:* Some medications are constrained to be taken sufficiently close together or taken together with food. Such a constraint between two medications M and N (or food) is given in a GMS specification in terms of maximum separation between their jobs: The current job of medication M (or N) must start within $\sigma_{max}(N, M)$ (or $\sigma_{max}(M, N)$) time units from the latest job of N (or M) that has an earlier start time. (The precise definition of maximum separation between jobs of different medications can be found in [3].) This requirement can be treated also as a relative deadline. Because a medication M may interact with multiple medications (and food), a job $J_M(i)$ may have multiple inter-stream relative deadlines, each $\sigma_{max}(N, M)$ of which is relative to the start time of a job $J_N(j)$ of some interacting medication $N \neq M$. The resulting absolute deadline for $J_M(i)$ is the earliest of the absolute deadlines, each of which is the sum of $\sigma_{max}(N, M)$ and the start time of $J_N(j)$, for every N in the interaction pairs given by the user's GMS specification.

The *absolute start time deadline* of the job $J_M(i)$ is the earliest amongst its intra-stream separation deadline, effective minimum intake deadline, and inter-stream separation deadline. The *absolute completion time deadline* of $J_M(i)$ is the absolute start time deadline of the next job $J_M(i+1)$ if $J_M(i+1)$ exists; otherwise, it is equal to infinity. The example in Section A1 of Appendix A illustrates these deadlines.

[1] From the perspective a job $J_M(i)$, the current intake interval is a constraint interval which includes the start time of the job.

Periodic and Sporadic Tasks Some medication scheduling algorithms may treat the sequence of CE jobs of a medication M as a periodic task or sporadic task [4-7] on P_M. The inter-arrival interval π_M between consecutive jobs of the task is in the range $[s_{min}(M), s_{max}(M)]$. The task is periodic if $s_{min}(M)$ is larger than zero and $s_{max}(M)$ is finite. Otherwise, the task is sporadic. The separation in start times of the jobs in consecutive periods of a periodic task must be no greater than the relative deadline $S_{max}(M)$. This requirement is hard and is equivalent to a jitter constraint of no greater than $S_{max}(M) - \pi_M$.

Our previous report on GMS model suggests the use of the well-known sporadic server algorithm [7] to maintain the maximum total intake constraint of a taken-as-needed medication. The dispenser treats CE jobs of such a medication as a sporadic task.

Scheduling Dispensing Jobs This report focuses on algorithms for scheduling CE jobs. Different algorithms differ in when the next job is scheduled and the order in which the next job is chosen from all the ready jobs. The controller first computes a schedule of CE jobs following the resource allocation rules discussed in the remainder of this section. It then uses the schedule to determine when the corresponding dispensing jobs should start.

For the sake of concreteness, we assume that the controller uses the following strategy to schedule dispensing jobs:

1. Compute an estimate, θ, of user promptness based on user input and recorded user behavior.
2. Depending on whether the controller schedules one ready CE job at a time or all the CE jobs of each medication M at once, compute the start time t of the next CE job or every CE job of M according to the chosen algorithm(s) for scheduling CE jobs.
3. Assume that t is the completion time of the corresponding dispensing job and set the start time of the corresponding dispensing job at $t - \theta$.
4. If $t - \theta$ is later than the current time, wait until $t - \theta$ and then start the dispensing job. Else, start the dispensing job immediately.

After computing the start and completion times of each dispensing job in this manner, the controller then chooses the dose size used at each dispensing based on the dose size and intake constraints. The feasibility of dose-size selection will be treated in a separate report.

2.2 Processor Time Requirements

Specifically, the controller uses the schedule of CE jobs of each medication M on the processor P_M of the medication to determine when to administer doses of M. As stated earlier, each CE job of M is scheduled non-preemptively on the processor P_M.

Precise Schedules When the user is sufficiently predictable, the error in the estimate θ of promptness is negligibly small. CE jobs actually start close to their schedule times. In this case, each job is allocated e_M units of time, where e_M is equal to the nominal minimum separation $s_{min}(M)$ of the medication. In other words, the processor P_M is no longer available for e_M units of time to subsequent CE jobs of M at the instant when a dose of M is dispensed to the user.

Borrowing the terms used in real-time systems literature [4, 5], we call e_M the *execution time* of CE jobs of M. We say that a job is scheduled *precisely* when it is given e_M units of time. A schedule according to which all jobs start by their respective deadlines and are precisely scheduled is called a *feasible precise schedule.*

Imprecise Schedules A typical user is not always prompt, and promptness varies from time to time, often unpredictably. Dispensing jobs may not complete, and their corresponding CE jobs may not start, according to schedule. In particular, larger-than-expected promptness may cause some CE jobs to start later than their scheduled times. As a consequence, there may not be sufficient time to schedule some job for e_M units of time before the subsequent job of M is required to start. When this occurs, the controller treats the job as an *imprecise job* [8]. An imprecise job is divided into a *mandatory part* and an *optional part*. The *mandatory part* of every job must be scheduled precisely: The mandatory part of a job of M occupies the processor P_M non-preemptively for $S_{min}(M)$ units of time, where $S_{min}(M)$ is the absolute minimum separation of M. The required processor time $S_{min}(M)$ of the mandatory part is called the *mandatory execution time* of M in the imprecise computation model.

The optional part of each CE job takes $e_M - S_{min}(M)$ units of time; $e_M - S_{min}(M)$ is called the *optional execution time* of M. When there is insufficient time to schedule the optional part for this amount of time immediately following the mandatory part, the controller tries to schedule as much of the optional part as possible. It may terminate the optional part of a job prematurely in order to start the mandatory part of a subsequent job on time. In this case, the prematurely terminated job is said to be scheduled *imprecisely*.

The timing requirement of the mandatory part is hard: A timing fault of M occurs and the controller must take appropriate corrective action when there is insufficient time on P_M for the mandatory part of any job of M by the absolute completion deadline of the job. On the other hand, as long as every job gets its mandatory execution time in time, the schedule remains feasible. A feasible schedule according to which some optional parts are not scheduled in their entirety is a *feasible imprecise schedule*.

Choices of Schedules A reasonable design choice is that the controller uses only feasible precise

schedules as long as the error in its estimate θ of user promptness is negligible. If the absolute minimum separation of a medication is equal to its nominal minimum separation, the controller has not other choice anyway. If the absolute minimum separation is larger, the controller will gain some tolerance to user tardiness by degrading the schedule to an imprecise one.

The observations below follow straightforwardly from the definitions of CE jobs, their processor time requirements, timing constraints and scheduling rules.

Observation 1 *All separation constraints between doses of M are met when every CE job starts within its deadline $S_{max}(M)$ relative to the start time of the previous job of M and is allocated at least its mandatory execution time on P_M.*

Observation 2 *Absolute maximum separation constraints specified by all the interaction pairs are met when every CE job starts by its absolute deadline and is allocated at least its mandatory execution time on P_M.*

2.3 Resource Requirements

Let M be a medication taken by the user, and M interact with some other medication or food. (That is, the GMS specification of the user contains at least one interaction pair involving M.) For each such medication, the controller maintains a resource R_M. It uses the resource R_M to maintain the required minimum separation between doses of the medication and doses of other medications and food.

Shared and Exclusive Uses The resource requirements and allocation rules defined below are such that no new job of M can start when it is unsafe for the user to take the new dose because of possible undesirable interaction with earlier doses of other medications:

- Rule 1: Every job of M must have R_M exclusively for an infinitesimally small amount of time in order to start.
- Rule 2: Every job of medication N ($\neq M$) in each interaction pair $I(M, N)$ of M requires the resource R_M on a shared basis for $\sigma_{min}(N, M)$ units of time beginning from when the job starts, where $\sigma_{min}(N, M)$ is the minimum separation between a dispensing job of N and a dispensing job of M when the former completes earlier.

Because of Rule 1, the resource R_M serves as a permit for jobs of M. A job of N can block a job of M for $\sigma_{min}(N, M)$ units of time. We call this time *blocking time* of M by N. The *worst case blocking time* of a medication M is equal to the maximum over all $N \neq M$ the blocking time of M due to N.

In contrast, an arbitrary number of jobs of medications other than M can share the resource $R_{M.}$ Hence, their jobs are never blocked due to contention for $R_{M.}$ The observation below follows as a direct consequence.

Observation 3 *Minimum separation constraints specified by all interaction pairs are met when resources are allocated to CE jobs according to Rules 1 and 2.*

Food Jobs and Their Resource Requirements The controller uses virtual processor P_{Food} and resource R_{Food} to keep track of user's intake of food when some medication taken by the user interacts with food. Let M be such a medication. That is, the user's GMS specification contains the interaction pair $I(M, Food)$. Each job of M holds the resource R_{Food} for $\sigma_{min}(M, Food)$ units of time. Every CE job corresponding to a meal or snack must have the exclusive use of the resource R_{Food} in order to start. When it starts, it holds the resource R_M for $\sigma_{min}(Food, M)$ units of time.

Each constraint enforcement job of food is scheduled on processor P_{Food} for ζ units of time. ζ denotes user's *eating time*: the amount of time the user takes to consume a meal or snack. Unlike CE jobs of medications, which are scheduled non-preemptively on their respective processors, a new CE job of food can preempt an earlier food job. Hence a new food job can start at any time provided that the resource R_{Food} is free at the time. A preempted food job is terminated. This models the reality that the user does not eat continuously.

In all the examples presented in this report, we let ζ equal to 0.5 hour. In general, the amount of time the user takes to eat a meal or snack is user and time dependent. It may be a random variable or a deterministic variable with a range of values. The assumption of ζ being a short 0.5 hour leads to no loss of generality, however. Because food jobs are preemptively scheduled, we can model a long banquet meal by a sequence containing a known or random number of snacks.

2.4 Comparison with Real-Time Workload Models

The SR model can be thought of as a special case of well-known workload models [4-7] of real-time applications. In terms from real-time systems literature, the sequence of constraint enforcement (CE) jobs of each medication is a *task.* – Hereafter, we use the term task and medication interchangeably: M denotes a medication sometimes and a task at other times, depending on the context of our discussion.

An important difference between a CE task and real-time task is that the former does not represent actual workload. In contrast, the latter represents some work done by an application, runs on a physical processor, and uses system resources. If we were to ignore this difference, a system of CE tasks could easily be modeling the workload from a hard real-time, mission-critical

application on a multiprocessor platform. The system is statically configured: Each task is statically assigned to a processor, and there is only task per processor.

In general, CE tasks are imprecise [8] and distance constrained [9]. As stated in Section 2.1, some algorithms used to schedule them treat some of the tasks as periodic tasks. With rare exceptions, periodic CE tasks do not all have constant periods and infinite numbers of periods. Hence, they do not fit the traditional definition [4] of periodic task. Rather, they behave according to the more general definition given in [5]: The period of a task may vary but is always bounded from below. Furthermore, the task may have only a finite number of periods.

Even in a supposedly deterministic hard real-time application, the execution times of jobs can vary, causing overruns and timing faults. The imprecise computation [8] is one of the approaches to overrun handling. In contrast, the execution times of CE jobs are deterministic. Overruns and timing faults can occur nevertheless. As stated earlier, the amount time available for a CE job is non-deterministic. The non-determinism is due to variations in user promptness and the resulting variations in when the job actually starts. Treating CE jobs as imprecise jobs is a nature way to avoid timing faults (i.e., violation of timing constraints).

Similar to real-time tasks, CE tasks also require resources. Because of Rule 1 and Rule 2 in Section 2.3, a job in a task M may be blocked from starting while waiting for resource $R_{M.}$ We will return to discuss the exact length of the worst-case blocking time, the effect of blocking and approach to deal with the effect shortly.

Like audio and video tasks, CE tasks also have inter-stream synchronization requirements. Lip synch between audio stream and video stream is a maximum separation requirement. CE tasks may be required to meet both maximum and minimum separation constraints.

3 An Illustrative Example

To illustrate how the dispenser controller uses the virtual processors and resources in its decision making process, we consider as an example a user who takes vitamin and Fosamax on a long term basis. (The generic Fosamax name is alendronate. It is used for prevention of brittle bone.) At the time, the user also takes an antibiotic for some finite duration. The separation graph in Figure 2 summarizes the interactions between the medications and their interaction with food. The numbers in the square bracket underneath the name of each medication give the range of the nominal separation of the medication. Each edge (M, N) is labeled by the minimum separation between doses of M and N when M is taken earlier. The maximum separations between doses of different medications are infinite for all medications.

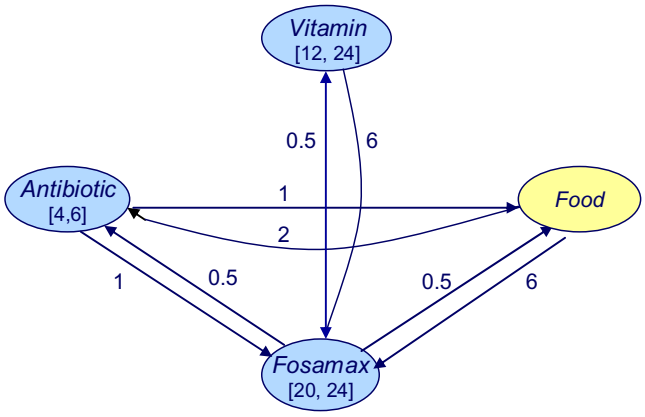

Figure 2 A Separation Graph

3.1 Simplifying Assumptions and Scheduling Rules

For the sake of simplicity, we let dose sizes of all the medications in the example be fixed. The freedom from concerns with choices of dose size here allows us to focus on choices of dispensing times.

Another simplifying assumption of this example is that that the user is perfectly predictable. That is, variations in user promptness are negligible. Consequently, the dispenser controller can accurately plan when to start each dispensing job so that the corresponding CE job starts at the scheduled time (i.e., every dose is dispensed at the scheduled time). In this case, every job is precisely scheduled according to a feasible schedule of the medications. We will return in Section 4 to discuss the effect of significant variations in user promptness.

All algorithms used in the example assign priorities to jobs. We consider here two variants: non-greedy and greedy. A *non-greedy algorithm* may choose to let jobs wait even when the jobs are ready and processor and resource are free. According to the family of non-greedy algorithms used here, the controller considers one medication at a time in priority order and generates a complete schedule of the medication. After the schedule of CE jobs (and consequently, the schedule of corresponding dispensing jobs) of all higher priority medications have been completed, the controller then computes a schedule of CE jobs of the medication with the highest priority among the yet-to-be scheduled medications taken by the user. This class of algorithms is sometimes said to be *time-driven* [5] or *static* because the start times of all future jobs within the durations of all medications have been computed.

The second variant of algorithms is greedy. A *greedy algorithm* never let jobs wait intentionally. According to a greedy algorithm, the controller places each job in the ready queue of each processor or resource in priority order when the job is ready. It allocates the processor or

resource to the job at the head of the queue whenever the processor or resource becomes free. Greedy algorithms are also called priority-driven algorithms. A *priority-driven algorithm* schedules one job at a time without looking ahead. When using a priority-driven algorithm, the controller views the sequence of jobs of each medication as a periodic task. The job in each period is ready at the beginning of the period.

3.2 Most-Victimized-First Algorithms

All the algorithms illustrated in this section use the Most-Victimized-First (MVF) priority scheme. The MVF scheme gives priorities to jobs based on their worst case blocking times; the longer the worst case blocking time, the higher the priority: In other words, jobs of medication M have an equal or higher priority than jobs of medication M' if

$$\max_N \sigma_{min}(N, M) \geq \max_N \sigma_{min}(N, M')$$

Jobs of the same medications are scheduled in FIFO order. MVF is a *fixed priority scheme* because it gives jobs in each medication the same priority relative to jobs in other medications.

The worst case blocking times of Fosamax, antibiotic, food and vitamin are 6, 2, 0.5 and 0, respectively, according to Figure 2. Hence, the MVF scheme gives jobs of Fosamax the highest priority, followed by jobs of antibiotic, meals and snacks and then by jobs of vitamin.

MVF-NG Schedule Figure 3 shows a schedule of CE jobs of the medications and food produced by the non-greedy MVF (MVF-NG) algorithm. Since all the medications and food interact, the controller maintains a resource for each. In this and later figures, processors for Fosamax, antibiotic, food and vitamin are named P_{FX}, P_A, P_{FD}, *and* P_V, respectively. Their resources are named R_{FX}, R_A, R_{FD}, *and* R_V, respectively. Following the MVF-NG algorithm, the controller treats food as if it were a medication. It begins by scheduling CE jobs of Fosamax because they have the highest priority. It then schedules jobs of antibiotic, meals/snacks and vitamin in order. The resultant schedule is periodic with a 24-hour period.

Specifically, Figure 3 shows a 24-hour segment of the schedule; the time origin is the start of a day. The boxes on timelines labeled by the names of processors and resources indicate the time intervals during which the resources are in use.

- The Fosamax CE job of day begins at time 0. (This means that the controller wakes up the user by sending an alarm/reminder at θ units of time before 0, where θ is an estimate of the user promptness.) The job occupies the processor P_{FX} for 20 hours. The controller also allocates the job resources R_A, R_{FD}, and R_V, each for 0.5 hour.
- Since R_A is not available until time 0.5, the first antibiotic job cannot start until that time.

The controller decides to use the maximum nominal separation of antibiotic and schedules the 4 jobs of the day evenly space in time at 0.5, 6.5, 12.5 and 18.5. Each job is allocated the resources R_{FX} of Fosamax and R_{FD} of food for one hour.

- The user cannot eat the first meal until time 1.5 since R_{FD} is not available until then. Given the schedule of subsequent antibiotic jobs, the controller finds that a good choice of time for the second meal/snack is either at 4.5, allowing the required 2-hour separation before the next dose of antibiotic, or at 7.5, leaving the required 1-hour separation after the second dose of antibiotic. The controller chooses the latter option, as indicated by the solid shaded box beginning from 7.5. It chooses to schedule the third meal at 13.5, one hour after the third dose of antibiotic when the resource R_{FD} becomes free again.
- Finally, since only Fosamax blocks the vitamin job of the day, the job can start at time 0.5, together with the first antibiotic job of the day.

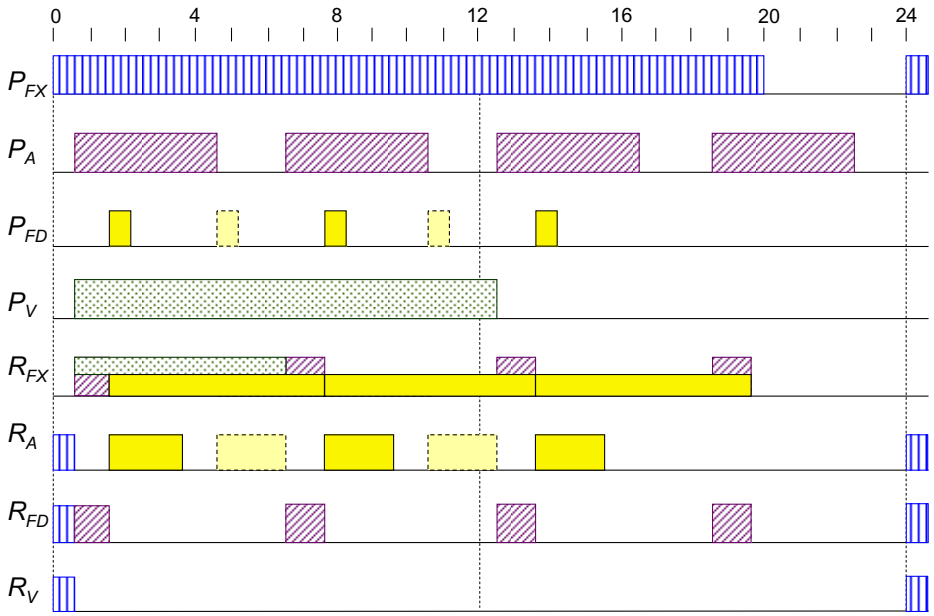

Figure 3 A MVF-NG Schedule of Medications in Figure 2

We note that the schedule in Figure 3 is not user friendly: Suppose that the start of the day is 6 AM. The last dose of antibiotic is dispensed at 12:30 AM, leaving the user with at most 5 and half hours of uninterrupted sleep. The schedule can be improved in this respect by scheduling the last dose at 16.5 (i.e., 10:30 PM). The resultant schedule does not meet the maximum nominal separation constraint, however, since the separation between the last dose of a day and first one of the next day is 8. The schedule is nevertheless acceptable if the absolute maximum separation is 8 or more. Meal times are not ideal also. After having breakfast at 7:30 AM, the user must wait until 1:30 PM to have lunch. The large separation between the meals is not a serious problem,

however, since there is time for snacks in between meals. – In Figure 3, dotted boxes starting from 4.5 and 10.5 indicate the latest times when the user can take snacks between meals without blocking any antibiotic job.

MVF-NG-FL and MVF-NG-FF Schedules The MVF-NG algorithm treats food as it were a medication and schedule meals and snacks as dispensing jobs. In contrast, the MVF-NG-FL and MVF-NG-FF algorithms plan meals and snacks separately from medications.

The FL (Food Last) variant first schedules jobs of medications. It then tries to schedule meals and snacks in time intervals when food intakes are allowed. In our example here, the MVF rule assigns to food a low priority. Consequently the MVF-NG-FL algorithm happens to produce the same schedule (namely, the one in Figure 3) as the MVF-NG algorithm.

The FF (Food First) first schedules meals and snacks times based on user preference and then schedules jobs of medications in time intervals allowed. This is the strategy advocated in [3]. In essence, it always tries to satisfy user preference whenever possible. Meal times specified by the user are treated as forbidden intervals [10] for medications that interact with food. It alters the forbidden intervals with the help from the user only when the changes are necessary. If the user prefers to eat meals at 1.5, 7.5 and 13.5 time units from the start of the day, the MVF-NG-FF algorithm also produces the schedule in Figure 3. Figure 4 shows the schedule the MVF-NG-FF algorithm produces if the user rises at 6:00 AM, and wants to eat breakfast at 0.5 (i.e., 6:30 AM), lunch at 6 (i.e., 12 noon) and dinner at 12 (i.e., 6:00 PM).

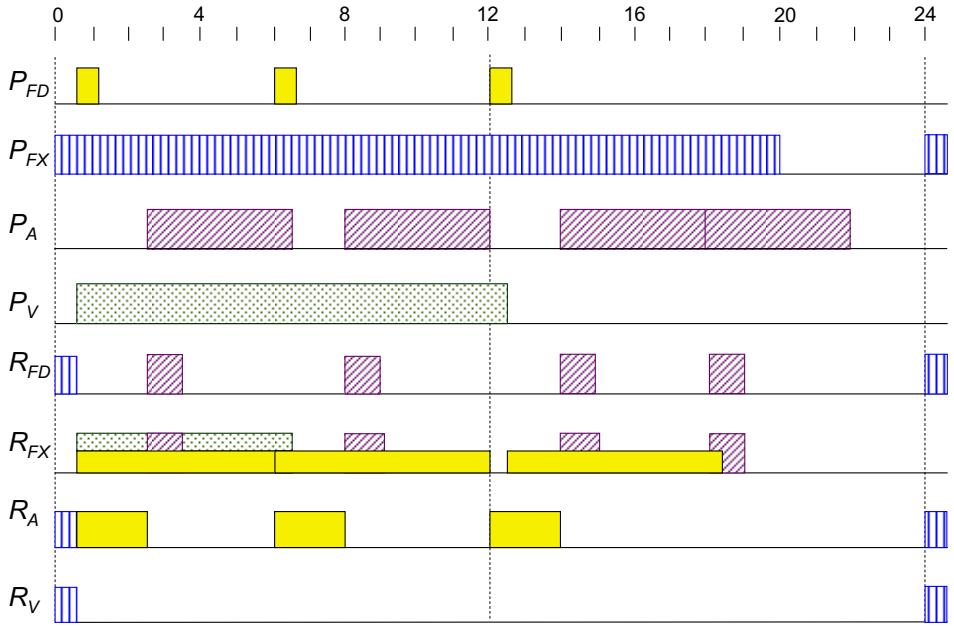

Figure 4 A MVF-NG-FF Schedule

As the figure shows, the user's willingness to wait for 0.5 hour before breakfast allows a dose of Fosamax to be scheduled at 0. The doses of antibiotic are scheduled as soon as R_A is free at 2.5. Depending on whether the absolute maximum separation of the antibiotic is at least 8.5 or 6.5, the controller schedules the last dose at 18 or 20, respectively. The algorithm would fail to find a feasible schedule if the absolute maximum separation is equal to 6.

MVF-PD Schedule Figure 5 shows the schedule produced by the priority-driven variant of the MVF (MVF-PD) algorithm. According to this algorithm, food is scheduled together with medication. The algorithm views the sequence of jobs of each medication as a periodic task. The periods of the Fosamax, antibiotic, and vitamin are 24, 6, and 24, respectively. At time 0 (the start of the day), the tasks are in-phase (i.e., a period of every task starts at 0.) Suppose that the user prefers to have breakfast at time 0, lunch around 6 (noon) and dinner around 12 (6 PM). So the food jobs of the day become ready at 0, 6 and 12.

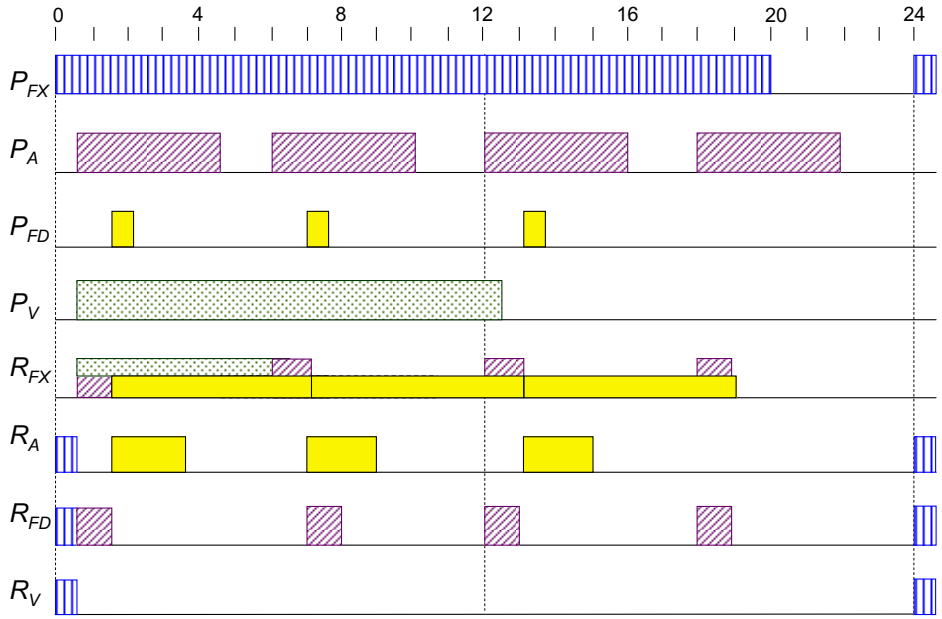

Figure 5 A MVF-PD Schedule of Medications in Figure 2

- At the start of the day, each of the medications has a ready job. The Fosamax job has the highest priority. It starts at time 0. All other jobs are prevented from starting because the resources they require are not available.
- At time 0.5, resources R_A, R_{FD}, and R_V are free. The antibiotic job has the highest priority. It starts and takes resource $R_{FD,}$ blocking the food job from starting. Since the resource R_V is free, the vitamin job also starts at time 0.5 and gets R_{FX} for 6 units of time.
- At time 1.5, R_{FD} becomes free, allowing the first meal to be scheduled.
- At time 6, both the second antibiotic job and the second food job are ready. Since the

former has a higher priority, it starts at 6, gets R_{FD}, and blocks the food job from starting.

- At time 7, R_{FD} becomes free and the second food job is ready and is scheduled.
- Similarly, when the third antibiotic job and food job are ready at 12, the antibiotic job is scheduled. The start of the food job is delayed until time 13.
- The fourth antibiotic job starts at time 18 when it becomes ready.

We note that the MVF-PD schedule is similar to the MVF-NG schedule except for the large jitter in the start times of the antibiotic job. It is acceptable only when the absolute maximum separation is 6.5 or more.

4 Performance Measures

We measure the merits of an algorithm for medication scheduling from both the user perspective and the dispenser perspective. For the user, an algorithm is only as good as the quality of the schedule it produces. For the dispenser, an algorithm is useless unless it can produce schedules of a sufficiently good quality when given a GMS specification. This section defines several measures that we can use to quantify these aspects of quality.

4.1 Medication Schedule Quality

Quality of medication schedules has two dimensions: adherence to medication directions and user friendliness. Often, these dimensions are in conflict, meaning it is not possible to optimize a schedule along both dimensions. The schedule in Figure 3 is an example. It is almost ideal from the perspective of adherence to medication directions: If the user is prompt, no separation between consecutive doses deviates from nominal ranges, and consecutive doses are equally spaced for every medication. This merit is especially important for medications such as antibiotic where maintaining an almost constant dosage level within the user is highly desirable. However, keeping a constant separation between doses is also why the schedule is not user friendly.

Adherence to Medication Directions Several figures of merit measure how close the schedule adheres to medication directions. They include dose-size variation, separation jitter, and deviations from nominal parameter ranges.

Given a schedule Ψ, the *separation jitter* $\zeta(\Psi, M)$ of M is equal to the maximum separation minus the minimum separation between consecutive doses of M according to the schedule Ψ, normalized with respect to the nominal separation range ($s_{max} - s_{min}$). The separation jitter $\zeta(\Psi)$ of the schedule Ψ is the weighted sum $\sum_M \omega_M \zeta(\Psi, M)$ of separation jitters of individual medications. The weight ω_M of every medication M is greater than or equal to 0 and the sum of the weights over all medications taken by the user is equal to 1. As stated above, separation

jitters of all medications are 0 according to the MVF-NG schedule in Figure 3. The separation jitters of antibiotic are 2.25 and 0.5, respectively, for the MVF-NG-FF schedule in Figure 4 and the MVF-PD schedule in Figure 5. By this criterion, one may say that MVF-NG-FF performs poorly if the user's condition treated by the antibiotic is serious enough to warrant a relative constant level of the medication.

Deviation from nominal separation range (*DSR*) of a medication *M* according to a schedule Ψ is the percentage of doses delivered with separations outside the nominal separation ranges. The *DSR* of a schedule is the weighted sum of *DSR* of all medications. – To simplify our discussion, we use the same set of weights $\{\omega_M\}$ to compute all figures of merits here. In general, different sets of weight may be warranted. – Implicit in this definition is that the deviation is a characteristic of the schedule: Some doses are scheduled too close together and some are too far apart by the scheduling algorithm; it is not a result of unpredictable user promptness. The MVF-NG schedule in Figure 3 has zero *DSR*. On the other hand, both the MVF-NG-FF and MVF-PD schedules in Figures 4 and 5 are more inferior because their *DSR* is 25%. As we will see shortly that a schedule with a non-zero *DSR* is not only undesirable from the perspective of adherence to directions, it may also be less user friendly.

Similarly, the *dose-size variation* $\delta(\Psi, M)$ of a medication *M* is equal to the difference between the maximum size and the minimum size of doses of *M* dispensed according to the schedule, normalized with respect to the nominal dose size range ($d_{max} - d_{min}$) of *M*. The dose-size variation of the schedule $\delta(\Psi)$ is the weighted sum $\Sigma_M \omega_M \delta(\Psi, M)$ over all medications taken by the user. *Deviation from nominal dose-size range* (*DDR*) of a medication is the percentage of doses with sizes outside the nominal dose-size range. The issues on selection of dose sizes and the relation between selections of dose-sizes and separations will be treated in a separate report.

User Friendliness We may quantify user friendliness of a schedule using subjective criteria and objective criteria. Subjective criteria are typically user dependent. They include the number of intakes per day and desirability of meal and dispensing times. We used the latter in the previous section as a criterion to judge the user friendliness of the MVF-NG schedule. The MVF-NG and MVF-PD schedules have the minimum number of intakes per day. The MVF-NG-FF schedule lets the user take meals at the preferred times. As an expense, the user has to take medication one time per day more than necessary.

Objective criteria are user-independent. An important quality of a medication schedule is its robustness: In essence, robustness of a schedule measures how tolerant the schedule is to unpredictability of user behavior. Every schedule is computed based on some estimated value(s)

of user promptness, which was defined in Section 2 as the length of time between the instant when the user receives a reminder (to take some medication) and the instant the user comes to retrieve the medication. A user friendlier schedule tolerates longer promptness; more importantly, it tolerates larger variation in promptness. The larger variation in promptness it allows, the more robust the schedule. Robustness can be quantified by maximum allowed tardiness and the rate of non-compliance.

Maximum Allowed Tardiness *Tardiness* is the difference between the actual promptness of the user when responding to a reminder and the estimated promptness, if the difference is positive. It is equal to 0 if the difference is non-postive[2]. The *maximum allowed tardiness* (*MAT*) $\Lambda(\Psi, M, k)$ of a dose k of medication M according to a schedule Ψ is the maximum length of time the corresponding CE job can be delayed without causing an instance of non-compliance. The *MAT* $\Lambda(\Psi, M)$ of the medication M is $min_k (\Lambda(\Psi, M, k))$ (i.e., the minimum *MAT* over all doses of M).

Take the MVF-NG-FF schedule in Figure 4 for example. Suppose that the absolute maximum separation of antibiotic is 9 and the minimum separation of antibiotic is 3. Then the values of *MAT* of the 4 daily doses of antibiotic are 0.5, 3, 1 and 5. (The following constraints would be violated if the user were tardy by more than these amounts: absolute maximum separation, absolute minimum separation, absolute minimum separation and minimum separation between antibiotic and Fosamax, respectively.) Therefore, the *MAT* of antibiotic is 0.5. According to the MVF-NG schedule, the *MAT* of antibiotic is 0: A delay in any of the first three doses of antibiotic would violate the minimum separation constraint between the medication and food.

The dispenser can sometimes improve robustness by re-computing the start times of subsequent jobs whenever the user is tardy. This would be case with the MVF-NG schedule. Suppose that the user is late coming to retrieve the first dose of antibiotic by 1 (an hour). The dispenser can give the user the choice between postponing breakfast by an hour, or eat breakfast at that time and postpone the first dose of antibiotic. Figure A3 in Appendix A shows the modified schedule if the user chooses to eat first. Indeed, the start time of each of the first three doses of antibiotic can be thus modified if necessary. Therefore, in the case where the dispenser does re-compute the medication schedule in response to user being tardy, the *MAT* of the MVF-NG schedule is 1.

In contrast, the *MAT* of antibiotic according to the MVF-NG-FF schedule cannot be

[2] We are concerned with user promptness only when it is longer than the estimate θ used to compute the schedule. If the user comes to retrieve a medication sooner than the scheduled time by a non-negligible amount of time, the dispenser can ask the user to wait, withhold the medication, and so on. The dispenser should be designed to handle this situation similar to how it safeguards against abuse and overdose of the medications under its care.

lengthened. The best the dispenser can do is 0.5, which is the *MAT* of the first dose. A tardiness of more than 0.5 would cause the separation between the last dose of previous day and this dose to exceed the absolute maximum separation of 9. Clearly, it is impossible for dispenser to delay the last dose of the previous day to prevent this instance of non-compliance. This example points out another reason why schedules with large separation jitter and non-zero *DSR* are undesirable. Such schedules tend to provide the dispenser with fewer options in handling user tardiness.

In general, the *MAT* of a medication may vary with time, as a function of user behavior. Take Fosamax as an example. All three MVF schedules start the day with a dose of this medication. A way to handle user tardiness by a non-negligible amount is simply to shift the entire schedule later, without regard to meal and bed time preferences. This is not user friendly, of course. Another possibility is to use an imprecise schedule similar to the one in Figure A2. Yet another option for medications such as Fosamax is to let the user skip the dose of the day and make up the missing dose the next day. Fosamax, like many modern medications, offers the choices between taking the minimum dose of 10 mg daily and larger doses up to 70 mg totaling 70 mg per week. The option of skipping some smaller doses is possible if the user can tolerate 70 mg but prefers to take 10 mg per day while taking antibiotics in order to minimize the chance of stomach upset. With this option, the *MAT* of Fosamax according to the MVF schedules is 24 hours for all but the last day in every seven days. The *MAT* for that day is zero. If the user were late again on that day, the dispenser would have to re-compute the schedule of subsequent doses of medications in order to accommodate Fosamax. Fosamax resembles tasks with (*m, k*)-firm deadlines [12] ($m = 7$, $k = 1$) as well as imprecise tasks with cumulative errors [8].

Non-Compliance Rate Schedules with the same *MAT* are not equally tolerant to tardiness. The rate of non-compliance as a function of user tardiness further distinguishes them. For a given schedule Ψ, the *rate of non-compliance* (*RNC*) $\rho(\Psi, M, x)$ of medication M is the percentage of non-compliant doses when the user is tardy by x units of time. For a dispenser that is incapable of dynamically re-compute the schedule, the MVF-NG schedule is worse than the MVF-NG-FF schedule. The rates of non-compliance of the schedules are 75 % and 25 %, respectively, for tardiness equal to their respective *MAT*. – By this criterion, the MVF-NG is again a poor schedule in terms of user friendliness. It is only useable when the dispenser can dynamically reschedule whenever the user is late.

4.2 Schedulability

Schedulability conditions are commonly used to compare algorithms for scheduling jobs with timing constraints. Literatures on real-time scheduling offer numerous schedulability conditions and time-demand analysis methods (e.g., [7, 8]) and tools (e.g., [9]) with which one can

determine whether a given set of tasks can be feasibly scheduled according to some algorithms. Unfortunately, most of them are not directly applicable to medication scheduling, and what are applicable are simple. A reason is that CE jobs in different tasks never compete for processor time, since each task has a dedicated processor in the SR model. Contention for processor time comes only from jobs in the same task: The delayed start of a previous job may in turn delay the start of the current job.

A job in a task M may be blocked from starting while waiting for resource $R_{M.}$ Because CE jobs of medications are scheduled non-preemptively, we are sure that the delay due to this blocking is always bounded, i.e., uncontrolled priority inversion can never happen. Furthermore, every job of M is blocked by at most a lower priority job for the length of time the lower priority job uses the resource usage time of R_M.

The only exception is food. In principle, a user can eat continuously, delaying indefinitely the dispensing of all medications that cannot be taken with food. The controller must anticipate the occurrence of this situation and have an effective approach to overrun handling of CE jobs.

In our example here, the maximum blocking times suffer by jobs in Fosamax, antibiotic, food and vitamin are 6, 2, 0.5 and 0 when priority are assigned on the MVF basis. This means that it may not be possible to schedule some Fosamax jobs precisely because the sum of their execution time (20) and worst case blocking time (6) exceeds their relative deadline (24). The imprecise schedule in Figure A2 in Appendix A illustrates a way to deal with the effect of blocking.

5 Future Work

This report presents the GMS-SR (General Medication Specification – Scarce Resource) model for medication scheduling. The model enables us to cast medication scheduling in the well-established framework of real-time scheduling. By doing so, we hope to be able to apply the rigorous real-time scheduling methods and theories as a foundation for medication scheduling.

Our preliminary investigation shows that medication scheduling indeed resembles real-time scheduling. Similar to real-time scheduling, most of the problems in medication scheduling are hard. Similar to schedulers in a hard real-time, mission-critical system, the dispenser controller wants are to meet all the timing constraints if at all possible and degrade gracefully when it cannot meet all constraints.

However, there are numerous differences between the problems. In particular, differences in resource demands and usage are fundamental: Tasks in SR model are non-preemptable. They do not share processors but can share resources. In contrast, most works on real-time scheduling

assume that tasks content for processor, are preemptively scheduled, and use resources exclusively. These differences, as well as differences in the types of timing constraints and performance goals, prevent the straightforward application of classical real-time scheduling algorithms and schedulability analysis theories and methods to medication scheduling.

To illustrate that classical algorithms known for their good performance and optimality do not perform well, Figures A4 and A5 show the schedules generated by the RM-PD (Rate Monotonic, Priority Driven) algorithm and the EDF-PD (Earliest Deadline First, Priority Driven) algorithm, respectively. Because the execution rate of Fosamax is the lower than the rates of antibiotics and food, it is scheduled in the background of those tasks by the RM-PD algorithm. The resultant schedule in Figure A4 is less user friendly than the MVF-PD schedule in Figure 5. The schedule produced by the EDF-PD algorithm is similarly flawed. Interestingly, if the minimum separation between antibiotic and Fosamax were 6 instead of 1, RM-PD algorithm would fail altogether to schedule any dose of Fosamax, while the MVF-PD algorithm would produce the same MVF-PD schedule. The EDF-PD algorithm would also fail to schedule the dose of Fosamax on the first day. It would, however, succeed on each of the subsequent days to schedule the dose for the previous day first thing in the morning and produces a schedule similar to the MVF-NG-FF schedule. The reason is that by the start of each subsequent days, the dose of Fosamax of the previous day is already late and, hence, has the earliest deadline. On the other hand, if the dispenser controller were to discard the late Fosamax job each day, as EDF scheduler sometimes does to late jobs, the algorithm would also fail to schedule any Fosamax dose.

We are currently evaluating the relative performance of algorithms based on the SR model and the RR model, both in terms of their ability to produce feasible schedules and their other quality measures, on a large set of real-life and synthetic GMS specifications. An important goal of our work is to prove rigorously the correctness and to predict reliably the behavior of efficient scheduling algorithms so that they can be used safely for medication scheduling and compliance enforcement. Much work in this direction remains to be done. Compared with what we know about algorithms for scheduling real-time tasks, we are still far from having necessary insight and thorough understanding of the behavior of medication scheduling algorithms. We also need to have sound graceful degradation mechanisms. The report discussed earlier the applicability of imprecise computation and (*m, k*)-firm deadline mechanisms. We will explore in depth their use for prevention of non-compliance in the future.

References

[1] S. C. Dursco, “Technological Advances in Improving Medication Adherence in the Elderly,”

Annals of Long-Term Care: Clinical Care and Aging, Vol. 9, No. 4, 2001.

[2] Pill boxes and medication scheduling http://www.epill.com/ and http://www.dynamic-living.com/automated_medication_dispenser.htm

[3] P. C. Hsiu, H. C. Yeh, P. H. Tsai, C. S. Shih, D. H. Burkhardt, T. W. Kuo, J. W. S. Liu and T. Y. Huang, "A General Model for Medication Scheduling," Institute of Information Sciences, Academia Sinica, Taiwan, Technical Report No. TR-IIS-05-008, October 2005.

[4] C. L. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of ACM*, Vol. 20, No. 1, 1973.

[5] J. W. S. Liu, *Real-Time Systems*, Chapters 6-9, Prentice Hall, 2000.

[6] M. Klien, L. Sha, J. Lohoczky, R. Rajkumar, *et al*. Rate-Monotonic Analysis: Software Technology Roadmap, http://www.sei.cmu.edu/str/descriptions/rma_body.html.

[7] B. Spruri, L. Sha, J. P. Lehoczky, "Aperiodic task Scheduling for Hard Real-Time Systems," *Real-Time Systems Journal*, Vol. 1, No. 1, 1989.

[8] J. W. S. Liu, K. J. Lin, W. K. Shih, R. Bettati and J.Y. Chung, "Imprecise Computations," *IEEE Proceedings*, Vol. 82, pp. 1-12, January 1994.

[9] C.C. Han, K.J. Lin and J.W.S. Liu, "Scheduling jobs with temporal distance constraints," *SIAM Journals on Computing,* Vol. 24, No. 5, 1995.

[10] C. S. Shih, J. W. S. Liu and I. Cheong, "Scheduling Jobs with Multiple Feasible Intervals," *Proceedings of 2003 RTCSA,* pp. 213-231, February 2003.

[11] J. W. S. Liu, C. L. Liu, L. Redondo, Z. Deng, T.S. Tia, R. Bettati, J. Sun, A. Silberman, M. Storch and D. Hull, "PERTS: A Prototyping Environment for Real-Time Systems," *International Journal of Software Engineering and Knowledge Engineering*, Vol.6, No.2, pp. 161-177, June 1996.

[12] M. Hamdaoui and P. Ramanathan, "A Dynamic Priority Assignment Technique for Streams with (m, k)-firm deadlines," *IEEE Transactions on Computers*, Vol. 44, No. 12, December 1995.

Appendix A More Illustrative Examples

This appendix provides examples to clarify definitions and algorithms presented in this report.

A1 Relative and Absolute Deadlines

The example in Figure A1 intends to illustrate the computation of relative and absolute deadlines defined in Section 2.1. The relevant direction parameters of medication *M* are listed below. To keep the example simple, we suppose that *M* has a fixed dose size, and the user is to take only one dose of each of the interacting medications *X* and *Y*.

- Nominal and absolute dose sizes = [1, 1];
- Nominal separation [s_{min}, s_{max}] = [5, 8];
- Absolute separation [S_{min}, S_{max}] = [3, 10];
- Minimum total intake = (5, 33)-*Periodic;*
- There are two interaction pairs: (*M, X*) and (*M, Y*) with $\sigma_{max}(X, M) = 6$; $\sigma_{max}(Y, M) = 8$.

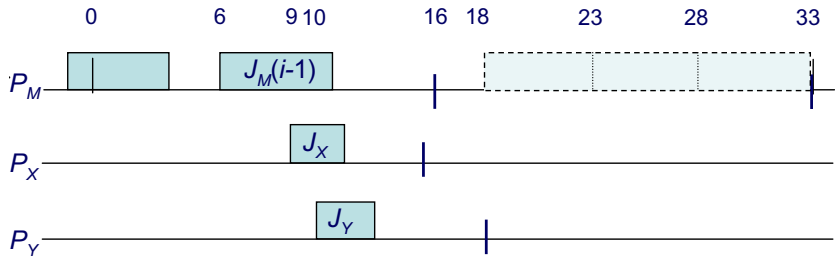

Figure A1 Relative and Absolute Deadlines of a Job

Suppose that at time *t*, the controller computes the absolute start deadlines of $J_M(i)$. The time origin used in the figure is the start of the current minimum intake interval at the time. Jobs $J_M(i\text{-}1)$, J_X, and J_Y have already been scheduled to start at time instants 6, 9 and 10, respectively, as indicated by solid boxes on the timelines labeled by the processor names. The absolute start time deadline of $J_M(i)$ is the earliest of the following three factors:

- The inter-stream relative deadline of $J_M(i)$ is S_{max} = 10. Hence the absolute deadline from this requirement is 6+10 = 16.
- The end of the current minimum intake period is 33. Clearly, $J_M(i)$ must start by this time or sooner. In addition to $J_M(i\text{-}1)$ and $J_M(i)$, three more jobs of *M* must start by the end of the current minimum intake interval in order to meet the (5, 33)-*Periodic* constraint. Suppose that $J_M(i)$ and the subsequent jobs within this period are separated by the nominal minimum separation s_{min} = 5. Effectively, $J_M(i)$ must start by 18 in order to leave sufficient time for the $J_M(i+1)$, $J_M(i+2)$ and $J_M(i+3)$ to start by time 33. This is illustrated

by the dashed boxes on the timeline labeled P_M. It follows that the effective absolute minimum intake deadline of $J_M(i)$ is 18.

- $J_M(i)$ has two inter-stream deadlines: $\sigma_{max}(X, M) = 6$ relative to the start time of J_X and $\sigma_{max}(Y, M) = 8$ relative to the start time of J_Y. The respective absolute deadlines are 9+6 = 15 and 10+8 = 18. It follows that the absolute inter-stream deadline of $J_M(i)$ is 15.

The absolute start time deadline of $J_M(i)$, calculated from these factors, is 15.

The absolute completion time deadline of $J_M(i)$ is equal to the absolute start time deadline of $J_M(i+1)$. $J_M(i+1)$ does not have inter-stream maximum separation requirement. Suppose that the controller schedules $J_M(i)$ at 14. Then the absolute start time deadline of $J_M(i+1)$ is 23, which is the minimum of 14+10 = 24 and 33–2*5 = 23.

A2 Effect of Blocking

The example below illustrates how contention for resource may lead to a poorer schedule. The schedule in Figure A2 is a MVF-NG schedule. Rather than taking Fosamax first thing in the morning, the user begins that day with breakfast. As a result, the Fosamax job is blocked from starting until time 6. By dispensing the day's dose of Fosamax at time 6, the remaining schedule of the day is left unchanged. This choice is acceptable if the absolute minimum separation of Fosamax is no less than 18. An alternative is to cancel the Fosamax job of the day and give the user a double-sized dose the next day according to the schedule in Figure 3.

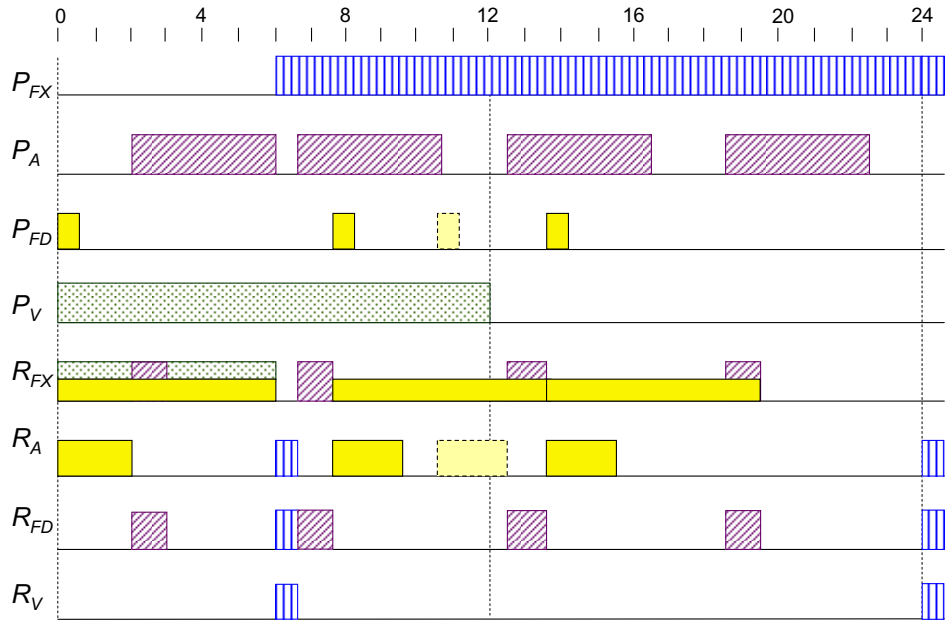

Figure A2 Another MVF-NG Schedule of Medications in Figure 2

A3 Miscellaneous Examples

The dispenser can improved robustness and, hence, be more user friendly by re-computing the medication schedule when necessary in order to avoid non-compliance. The modified MVF-NG schedule in Figure A3 is an example. The first dose of antibiotic is postponed because the user was late by one hour in responding to the reminder sent at time 0.5.

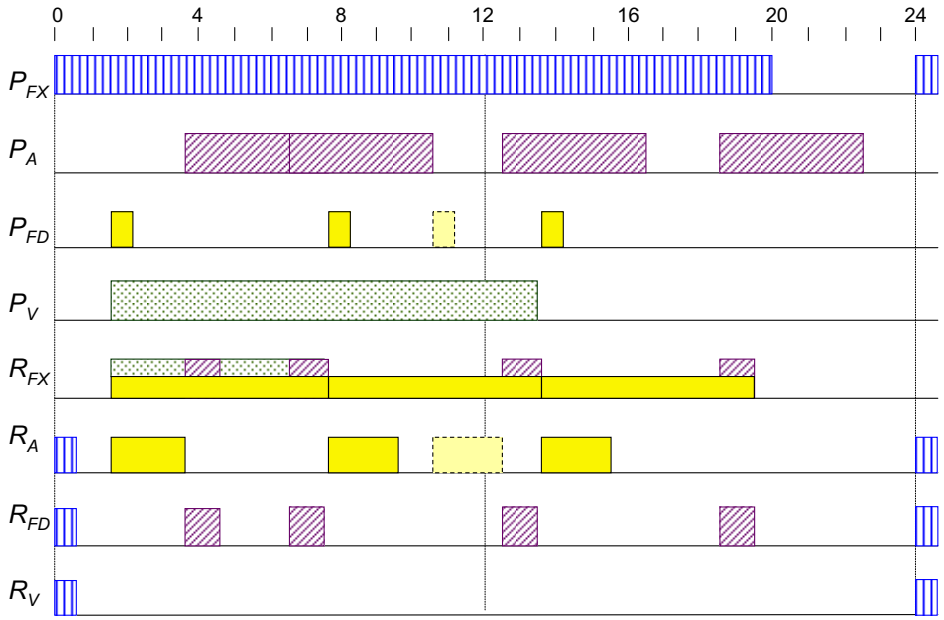

Figure A3 A Re-Computed MVF-NG Schedule

The schedules in Figure A4 and Figure A5 are produced by applying RM-PD (Rate Monotone Priority Driven) algorithm and EDF-PD (Earliest Deadline First Priority Driven) algorithm, respectively, on the medications in Figure 2. Similar to the MVF-PD algorithm, these algorithms treat the medication and food as periodic tasks. The periods of Fosamax and vitamin are 24 hours, the period of antibiotic is 6 hours, and the relative deadline of food is 6 hours. The tasks are in-phase at 0 when a day starts. The medications do not have inter-medication maximum separation constraints.

The RM-PD algorithm gives priorities to tasks according to their rates of execution. So, antibiotic has the highest priority, food the next highest priority, followed by Fosamax and vitamin. The tie between Fosamax and vitamin is broken by giving vitamin the lowest priority. As Figure A4 shows that while the schedule produced by the RM-PD algorithm adheres to directions, it is not user friendly. Because the dose of Fosamax is scheduled in the background of antibiotic and meals, it cannot be scheduled until the end of day.

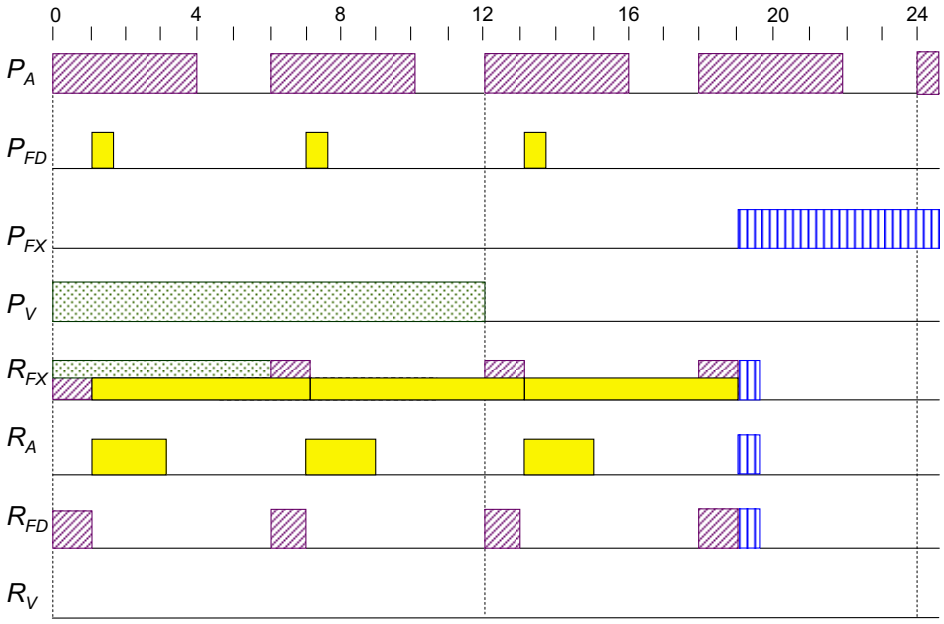

Figure A4 A RM-PD Schedule of Medications in Figure 2

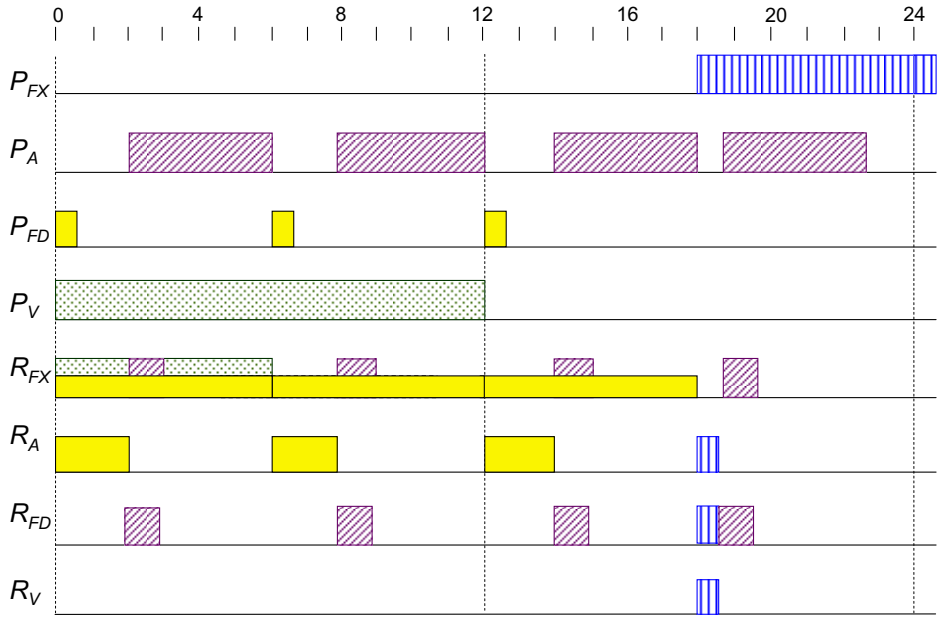

Figure A5 An EDF-PD Schedule of Medications in Figure 2

We also assume the maximum and minimum intake constraints are automatically satisfied when the absolute separation constraints are met. Hence, the absolute deadlines of the first jobs of Fosamax, antibiotic, food and vitamin are 6, 9, 24 and 24 respectively. The jobs become ready at time 0. The EDF-PD algorithm schedules the food job first because the job has the earliest deadline. It schedules the first antibiotic job as soon as R_A becomes free. It repeats this pattern at

time 6 and 12 when the second and third jobs of food and antibiotic are ready. (Because of the shorter relative deadline of food, the EDF-PD schedule in Figure A5 is identical to a EDF-PD-FF schedule.) The fourth antibiotic job has the same deadline as the Fosamax job. Because the Fosamax job became ready earlier, it is scheduled ahead of the antibiotic job.

Appendix B Terms and Notations

The GMS terms used in the report were defined in [3]. The appendix lists their definitions and notations in order to make the report more self-contained. The parameters listed are that of a medication *M.*

Absolute minimum separation $S_{min}(M)$ and absolute maximum separation $S_{max}(M)$: Values delimiting the compliant range of temporal distance between consecutive doses of *M.*

Absolute minimum dose D_{min} and absolute maximum dose D_{max}: Values delimiting the range of compliant dosage.

Change list $C(M)$: A list containing changes in direction of medication *M* necessitated by interaction of *M* with other mediations and food.

Completion time of a job: The time instant when the user retrieves a dose of the medication that is dispensed by the job.

Concurrency event: A violation of the instruction that two or more actions must be taken (or must not be taken) at the same time.

Direction parameters: GMS parameters that are derived from directions of medications.

Dispensing job (or **job**) **for a medication:** A sequence of actions by the dispenser to administer a dose of the medication:

Duration: The length of time or total number of doses (delimited by the minimum duration T_{min} and the maximum duration T_{max}) over which direction parameters of a medication are to be followed.

Elapse time of a job: The length of time between the start time and completion time of the job.

Feasible interval: A preferred interval time for dispensing a medication.

Forbidden interval: A time interval in which no dispensing job is allowed to start.

GMS graph: A graph that represents a GMS schedule.

GMS specification: A specification of medication schedules based on the GMS model described in the report.

Granularity: The size of unit in term which dosage of a medication is specified.

Interaction event: A violation of a constraint arisen from interaction of a medication with other medications and food.

Interaction pair *I* (*M, M'*): A part of the GMS specification that defines changes in directions of medication *M* and *M'* due to interactions between the medications.

Maximum total intake (*B, R*): A limit (defined by a budget *B* and replenishment time *R*) to total dosage *B* over any time interval of length *R*.

Minimum total intake (*L, P*)-Periodic: Time being segmented into periods of length *P,* a limit *L* to the minimum total dosage taken in every period.

Minimum total intake (*L, P*)-Uniform: A limit *L* to the minimum total dosage taken within any time interval of length *P.*

Nominal minimum dose size d_{min} and maximum dose size d_{max}: Dose sizes delimiting the range of dosage dispensed each time.

Nominal minimum separation s_{min} and maximum separation s_{max}: Values delimiting the range of temporal distance between consecutive dose.

Over-dose event: An event of the user taking a dose of size larger than the absolute maximum dose size, or two doses separated in time less than the absolute minimum separation, or both.

Precedence constraint: A constraint in the order of dispensing jobs.

Predecessor (or successor): A dispensing job that must complete before (or after) another job.

Over-medicate event: A violation of the maximum intake constraint or the maximum duration constraint.

Promptness: The amount of time the user takes to come and retrieve a medication after being reminded to take a dose.

Smart (medication) dispenser: A programmable device designed to help its user to compliant to medication directions and to provide the user or caretaker with warnings when it detects non-compliance.

Start time of job: The time instant when the dispenser reminds the user to come a take a dose of medication.

Static direction: A medication direction defined by constant parameters.

Time instant when a medication is administered or dispensed: The completion time of an associated dispensing job.

Time varying direction: A sequenced set of static medication directions.

Under-dose event: An event of the user taking a dose of size less than the absolute minimum dose size D_{min}, or two consecutive doses further apart than the absolute maximum separation. S_{max}, or both.

Under-medicate event: A violation of the minimum intake constraint or the minimum duration constraint.