Toward an Automated Patient Care System (APCS)

Ghasem S. Alijani[1] and Raghuram N. S. Tadimalla[2]
Southern University at New Orleans
6400 Press Drive, New Orleans, LA 70126

Abstract

Improvements in technology call for the development of patient care systems that can alleviate problems encountered due to decrease in the workforce of hospitals. The systems developed for patient care and clinical care should be real-time systems that can perform required functions. Criticality of patients' health should be taken into consideration while developing these systems. Simulation modeling helps solve the problem in the design of patient care systems. Automated Patient Care System (APCS) encompasses the requirements in hospitals that specifically address the needs of in-patients. APCS aims at the design and implementation of a highly reliable real-time model that provides the means to investigate the feasibility of an automated patient care system. Systems that perform critical missions in unpredictable environments require a significant consideration in efficient use of the available resources. Further, the underlying system requirements should be taken into consideration. The APCS model will address all the criticalities involved and will result in effective implementation of real-time system. The model should be of interest to medical professionals, hospitals and clinics, and the officials of the Health Department as it focuses on the system design and lays out the groundwork for a complete automated system supported by real-time task scheduling and incremental learning techniques for effective performance in unpredictable environments.

Keywords: critical mission, patient care systems, adaptive learning, hard real-time task-scheduling

1. INTRODUCTION

In recent years, the use of computers in controlling and monitoring of information, industrial, and medical systems has been expected greatly. The two million nurses in the medical field constitute the largest group of health care professionals. Between the years of 1983 and 1998, the average age of registered nurses has increased by 4.5 years (Buerhas, Staiger, and Auerbach 2000). This is due to the decrease in the number of women who show interest in the career of nursing. This decline in the number of women in nursing career is attributed to wide choice of career opportunities that exist for them (Bednash 2000). Improvements in technology call for the development of patient care systems that can alleviate problems encountered due to decrease in the workforce of hospitals. Time-critical applications such as life support systems require real-time response to operations and should be able to function in unpredictable environment. For the last two decades, the issues of real-time responsiveness and unpredictability of environment have been studied in terms of other requirements, such as languages (Krishnan and Volz 1989), task scheduling (Alijani and Chang Su 1990), load balancing (Hac and Jin 1987), and reliability and fault tolerance (Alijani and Wedde 1991; Anderson and Knight 1983). Also, learning by incremental techniques would make time-critical missions more effective. Such an adaptive learning strategy is vital for hard real-time tasks (Alijani, Omar, and Welsh 1997). All these aspects contribute to design of more complex and complete systems; however, it is clear that task scheduling schemes and fault tolerance mechanisms are more essential elements required by any real-time system performing critical missions.

Real-time systems are categorized as soft and hard real-time systems. In a soft real-time system, the time constraints of time-critical tasks might not be so restrained, and a statistical distribution of response time is normally acceptable. However, in hard real-time systems, tasks must be completed before some fixed time has elapsed. Thus, the importance of meeting the task's execution deadline make a reliable task scheduling scheme a central issue for the correctness and reliability of the system (Biyabani, Stankovic, and Ramamithram 1988; Woodside and Craig 1987).

[1] dalijani@ix.netcom.com
[2] raghutadimalla@hotmail.com

Wide ranging issues such as public policy, patient treatment procedures, and capital expenditure requirements in health care delivery can be addressed using simulation (Standridge 1999). Some of the characteristics of simulation are given below.

1) Conformation to both system structure and available system data.
2) Support to experimentation at relatively low cost and at little risk.
3) Conformation of results to unique system requirements for information.

Mistakes in medical field are very expensive and simulation can help in generating correct architectural decisions (Lange 1999) to avoid major losses to lives of human beings. Traditional medical planning and architecture methods are very expensive and hence, are seldom implemented by hospitals and clinics. Simulation can address all the "chaotic states" before actual implementation can be carried out. Simulation also helps in elimination of potential errors and mistakes in the design of any patient care system. Computer-human interface solutions for emergency medical were designed utilizing the technology of simulation (Holzman 1999). In such solutions, simulation was used for the configuration, functionality, and evaluation apart from system architecture of computer systems that assist medics in providing effective emergency medical care.

Before describing the task-scheduling scheme, the physical and logical elements of the system are presented.

2. PHYSICAL SPECIFICATION

As figure 1 shows, the entire system is composed of experts, monitoring devices (M_1, M_2, ..., M_k), mechanical devices such as intravenous fluid pumps and ventilator denoted as I_1, I_2, ..., I_n, and a control feedback system. The control feedback system consists of a set of subsystems (SS_1, SS_2, ..., SS_k) that are capable of communicating with other subsystems. Physically, each subsystem consists of a set of processors, I/O devices, local and external memories, and communication facilities.

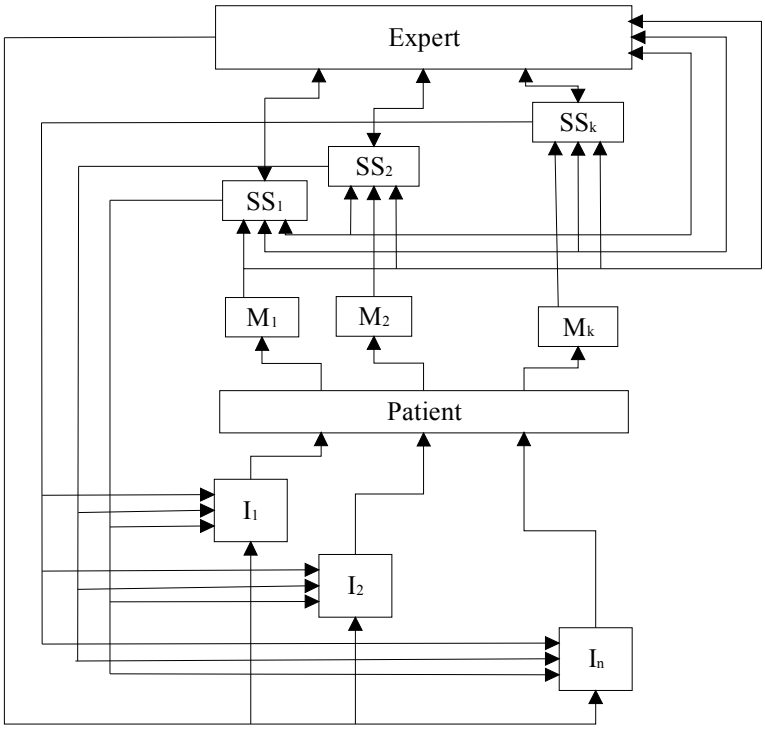

Figure 1. System Overview

Under normal conditions, each subsystem is responsible to receive the information on the status of the patient through a corresponding monitor and controls its associated mechanical device. However, due to the criticalness of the operations, each subsystem is able to receive signals from all the monitors directly or through other subsystems. Further, each subsystem is capable of sending signals to all mechanical devices. Inter-connection of all the monitors and mechanical devices to each individual subsystem will achieve the targeted response. Such a fully connected network gives the subsystem a built-in fault tolerance capability that is essential to every life support system.

The major tasks of each subsystem are collecting the information, scheduling the next event, and coordinating its action with other subsystems. What follows is a description of how these tasks are achieved.

3. LOGICAL SPECIFICATION

The expert is the focal point of this system. He/she will initiate the monitoring of a patient and input recommended treatment. However, to automate the system to the point that it would be capable of making decisions, each subsystem should continuously collect and store information on the status of the patient, schedule the next event, and coordinate its actions with other subsystems. Thus, the logical elements of each subsystem can be specified in terms of a task scheduler, a database management, and a coordinator.

Within each subsystem, a processor is designated to perform a logical element. These processors are referred to as task scheduler, coordinator, and database manager. In other words, each subsystem can be viewed as a multiprocessor that is capable of executing its tasks in real-time and concurrently. The following is a brief description of the logical elements.

Database Manager
The database manager is responsible to collect all the necessary data provided by the monitor and organize them based on their criticalness. It also stores the history of decisions made by the expert so far, and the corresponding data. This information and decisions help the expert to plan his treatment strategy. They can also be used to automate the decision making process used by the feedback control system (the detail of mapping a data set into a decision is beyond the scope of this paper). For the database manager to achieve its tasks, it must receive information from the monitor, communicate with the expert through the interface subsystem, and present the results in a simple form to the expert.

The database manager must communicate with the task scheduler to provide the required information for scheduling the next event. It also must communicate with the coordinator that provides access to other subsystems. This facilitates the information sharing required for coordinating the subsystems actions.

Coordinator
The coordinator processor works as media between the database manager, task scheduler, and other subsystems. In a normal condition, the coordinator gets the data from the database manager and the corresponding decision made by the expert or the task scheduler and sends them to other subsystems. Conversely, it receives the information and decisions that have been made by other subsystems and passes them to the database manager to be stored as a history. To tolerate subsystem failures, the functional capability of each subsystem has been replicated in others. Therefore, each of them is capable of performing the affected subsystem tasks. A suitable technique for determining the availability of subsystems is described elsewhere (Alijani and Chang Su 1990).

Task Scheduler
Generally, in a real-time computing system, tasks are classified as periodic and sporadic. In our system, under a normal condition, the scheduler is responsible to schedule the events (periodic tasks) related to a specific mechanical device after receiving information from the database manager and coordinator. However, in case of a subsystem failure, the scheduler should manage more than one device, and since the arrival times of these events are not predictable, the scheduler should be able to schedule sporadic tasks. While a task's deadline is usually known at the time of scheduling, its computation time is dependent on the data and availability of local and global resources. In the following paragraphs, we briefly describe our task scheduler scheme that has been tested in a multiprocessor and a distributed system. More details of algorithms and implementations of the task scheduler can be found elsewhere (Alijani and Wedde 1991).

To alleviate the problem, we make the use of the idea of 'safety time', a factor that is added to the worst-case computation time of each task whenever the deadline of the new task and the status of the scheduled tasks follow. Our scheduler consists of three phases and it is able to take advantage of accumulated safety times and locally available resources provided through replication and relocation techniques (Wedde and Alijani 1990). These factors provide the scheduler an extensive flexibility to handle a maximum number of critical tasks locally.

The success of scheduling a new task depends on its deadline, its worst-case computation time, and the current status of scheduled tasks. In our model, worst-case computation time, WT, consists of retrieving data, making a decision, and sending a signal to an appropriate mechanical device. The current status of each scheduled task is stored in a schedule list defined as S-List. To find an available time slot for a new task a search must be conducted within a 'window frame'. The size of the window, ws, can be defined in terms of the present time, t, and the deadline, dt, of the new task, ws = dt – t. The local scheduler maintains the S-List in which the tasks are ordered according to their start times. Therefore, the task with the Earliest-Start-Time will be executed next. The three phases of the task scheduler are described below.

First Phase: In this phase, all generated tasks (local or remote) are to be examined and possibly scheduled without using the STF (safety time factor) associated with each scheduled task. This phase will activate the second phase of the local scheduler if a new critical task meets a preset threshold (to be explained in

the second phase), and thus cannot be scheduled without further manipulation of the already scheduled tasks.

Second Phase: In this phase, the scheduler tries to guarantee timely execution of a new task by manipulating the already scheduled tasks within a window frame and utilizing the accumulated safety times. A metric will be defined in order to decide whether a task is "critical enough" to activate the second and possibly third phase of the scheduler. The scheduler compares a new task criticalness, cr, with a flexible threshold, TH, that will be set by the system as a parameter. If cr ≥ TH, then the new task is considered as a critical task. Otherwise, it will be classified as a non-critical task. Within the window frame, a forward shifting is performed to utilize the STFs and available times for scheduling the new critical task. If the provided time frame is less than the WT, then the third phase of the scheduler will be activated.

Third Phase: The critical tasks that are not processed by the second phase will be passed to the third phase for further consideration. Removal of one or more less critical tasks from the list is performed in order to provide a time slot for a new highly critical task. These tasks will be sent to other subsystems through the coordinator. Since searching for a victim low priority or non-critical task adds overhead to the subsystem, a provision must be made to reduce the search domain. In this model, a combination of the threshold and window frame will provide the system with an efficient search scheme. It should be noted that the non-critical tasks are guaranteed to be executed in the absence of critical tasks or when the schedule has enough accumulated time due to the safety times. The critical tasks are guaranteed to be executed once they are in the scheduler list.

4. CONCLUSIONS

In this paper, we have described the layout for design requirements of a real-time automated patient care system that is capable of performing in an unpredictable environment and tolerates failures. The advantage of the system is it provides the means for continuous monitoring of a patient, and it releases the staff from performing the trivial tasks and leaves them to more important tasks of supervision and giving the patient the emotional support and human interaction.

5. REFERENCES

Alijani, G. S. and S. Chang Su, 1990, "A Real-Time Task Scheduling Scheme Using Loosely Couple Systems." Proceedings of the Third Conference of the North America User Group, pp. 128-137.

Alijani, G. S., A. Omar, and J. S. Welsh, 1997, "An Adaptive Learning Strategy for Autonomous Machines." Proceedings of the URC-TC '97 Conference, pp. 43-48.

Alijani, G. S. and H. F. Wedde, 1991, "Enhanced Reliability In Scheduling Critical Tasks for Hard Real-Time Distributed Systems." Proceedings of ICCI '91.

Anderson, T. and J. C. Knight, 1983, "A Framework for Software Fault Tolerance in Real-Time Systems." IEEE Transactions on Software Engineering, SE-9 (3).

Bednash, G., 2000, "The Decreasing Supply of Registered Nurses – Inevitable Future or Call for Action?" Journal of American Medical Association, 283 (22), pp. 2985-2987.

Biyabani, S. R., J. A. Stankovic, and K. Ramamithram, 1988, "The Integration of Deadline and Criticalness in Hard Real-Time Scheduling." Proceedings of Real-Time Systems Symposium.

Buerhas, P. I., D. O. Staiger, and D. I. Auerbach, 2000, "Implications of an Aging Registered Nurse Workforce." Journal of American Medical Association, 283 (22), 2948-2954.

Hac, A. and X. Jin, 1987, "Dynamic Load Balancing in a Distributed System Using a Decentralized Algorithm." International Conference on Distributed Computing Systems.

Holzman, T. G., 1999, "Computer-Human Interface Solutions for Emergency Medical Care." Interactions, pp. 13-24.

Krishnan, P. and R. Volz, 1989, "A Distributed Real-Time Language and its Operational Semantics." IEEE Real-Time System Symposium, pp. 41-50.

Lange, V. E., 1999, "The benefits of simulation modeling in medical planning and medical design." Proceedings of the 1999 Winter Simulation Conference, pp. 1564-1567.

Standridge, C. R., 1999, "A tutorial on simulation in health care: Applications and issues." Proceedings of the 1999 Winter Simulation Conference, pp. 49-55.

Wedde, H. F. and G. S. Alijani, 1990, "MELODY: A Distributed Adaptive File System for Handling Real-Time Tasks in Unpredictable Environments." Journal of Real-Time Systems, 2 (4).

Woodside, C. M. and D. W. Craig, 1987, "Local Non-Preemptive Scheduling Policies for Hard Real-Time Distributed Systems." Proceedings of Real-Time Systems Symposium.

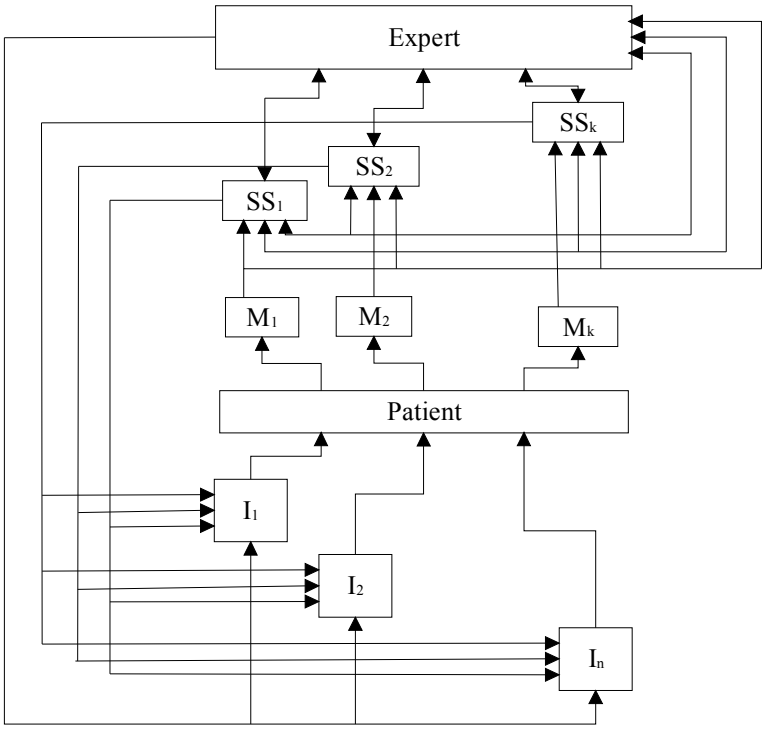

Figure 1. System Overview