ON COORDINATION OF CYBER-PHYSICAL SYSTEMS

TAO LI

Ph.D

The Hong Kong Polytechnic University

2014

The Hong Kong Polytechnic University
Department of Computing

On Coordination of Cyber-Physical Systems

Tao Li

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

October 2013

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Tao Li (Name of Student)

Abstract

The physical world will be saturated by networked devices with sensing and actuating capabilities. This trend will generate profound impacts on our way of monitoring and controlling the physical world, and meanwhile bring us towards a cyber-physical convergence era. Along this trend, a new type of systems, named *Cyber-Physical Systems (CPS)*, is developed by taking into consideration the cyber and physical world interactions. CPS refer to the integrated systems with both cyber and physical aspects, and their applications have been envisioned to generate grand societal and economic impacts. In this dissertation, we study the requirements and fundamental problems in CPS design. More specifically, we address the issue of how devices in distributed CPS coordinate their activities in order to achieve different goals in CPS applications. Examples of the coordination goals include cooperation of devices for verifying and maintaining desired system properties, and adapting to external situations, etc. Traditional methods of dealing with coordination have become insufficient in the CPS scenarios due to CPS' special characteristics, such as tight coupling of cyber and physical components, and safety/time critical nature. Moreover, the internal and external environments of CPS are intrinsically associated with diverse *uncertainties*, such as variation of network conditions, unpredictable user interactions, etc. Those uncertainties further complicate the coordination. This dissertation actually has made original

contributions in developing solutions to the above coordination problems.

First, we study how CPS devices coordinate to verify the system's properties, like safety. Our study is conducted in a context of *Medical Cyber-Physical Systems (MCPS)*, which are a typical CPS application in hospital and aim to coordinate networked medical devices for safer delivery of medical care to patients (i.e., a physical component). In medical domain, formal verification is a highly desirable procedure for eliminating design defects and increasing software quality. The combination of control, networking and patient in MCPS requires a joint modelling of the dynamics about the cyber and physical components, resulting in a hybrid system model. However, obtaining a tractable model for patient is difficult due to the *uncertainties* about patient's response to medical treatments (e.g., medication infusion). As a result, the overall verification is hindered. To solve this problem, our approach is to transform the traditional offline verification to an *online* version, in which MCPS devices coordinate at runtime to periodically generate a hybrid system model describing the system's behavior in the future short-term. Then, formal verification, specifically model checking, can be conducted over the online model. The rational under our approach is the fact that patient's physiological dynamics turns to be predictable in the near future and thus can be readily modelled. Furthermore, we propose to build the online verification procedure as a real-time task for continuous fault prediction and risk prevention. Once the online verification results indicate a violation of the desired properties, the devices coordinate to enter a fall-back plan, thus circumventing the potential risks. The feasibility and effectiveness of our approach have been validated in a concrete medical case, named Airway-laser Surgery.

Second, we study how CPS devices coordinate to maintain a crucial system property, i.e., safety. The definition of safety varies in different application contexts. We restrict

our study to MCPS where safety refers to freedom of medical accidents. To maintain this property, medical devices must communicate with each other, and act appropriately and promptly. However, one challenge that hinders the fulfillment of the safety property is network *uncertainties*, such as message loss and varied transmission delay. This problem arises especially when wireless technologies are adopted for inter-connecting medical devices. Existing coordination mechanisms in MCPS for dealing with network uncertainties are time-triggered and consequently can not give prompt response to surgeon's requests. The mechanism we proposed is *event-triggered*, and thus coordination can be initiated whenever needed. Besides, in order to tolerate network uncertainties, we introduce a central entity, called supervisor, to oversee the states of patient and each medical device, and then coordinate the current and future operations for each medical device. We designed the interaction mechanism for the supervisor and medical devices, as well as the coordination algorithm for the supervisor. In the algorithm, we take into account the special requirements from medical domain, and prove that the planned operations are always safe to be adopted by medical devices when network problems occur. Trace-driven simulation has demonstrated the advantages of our scheme in many aspects, such as clinical efficiency and response delay, compared to the existing approaches.

Third, we investigate the problem of coordinating CPS devices to enable adaptation to external situations, like user status and environmental changes. Increasing the system's awareness to the external situations has many benefits, e.g., createing autonomous and intelligent CPS applications, and better serving users. Again, we choose MCPS as the investigation scenario, and propose to make MCPS particularly adaptive to user's status, because the safety and quality of medical care highly depend on user's participation and

operation. Specifically, we treat *user errors*, emerging when users are operating medical devices, as *contexts* to MCPS, and build a context infrastructure to infer the user errors. With recognition of these errors, devices in MCPS then can coordinate appropriately to prevent the potential risks of those errors. However, it has been well acknowledged that the contexts we obtain are subject to *uncertainty*. Existing work to handle context uncertainty mainly focuses on modeling, inference, and mitigation of uncertainty. The adverse consequences of uncertainty, especially on patient safety, are neglected. Our solution to addressing this issue is two-fold. First, it is necessary to improve the quality of the detected contexts by appropriate approaches. Second, we modify the "context-action" adaptation style in the traditional context-aware systems to a *"context-assessment-action"* style, where the "assessment" step evaluates the safety of each context-triggered action based on solid medical knowledge. An action is allowed to be executed only if its execution is ascertained to be safe. Following this principle, we have developed a context-aware MCPS for a particular medial case, named Patient-Controlled Analgesia (PCA). Evaluation results about the system have shown that our system can prevent 95% more adverse events compared to the state-of-the-art context-insensitive PCA systems.

Publications

Journal Papers

1. **Tao Li**, Feng Tan, Qixin Wang, Lei Bu, Jiannong Cao and Xue Liu, "From Offline toward Real-Time: A Hybrid Systems Model Checking and CPS Co-Design Approach for Medical Device Plug-and-Play Collaborations," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 25, no. 3, February, 2014.

2. Lei Yang, Jiannong Cao, Yin Yuan, **Tao Li**, Andy Han and Alvin Chan, "A Framework for Partitioning and Execution of Data Stream Application in Mobile Cloud Computing," *ACM SigMetrics Performance Evaluation Review (PER)*, vol. 40, no. 4, March, 2013.

3. **Tao Li** and Jiannong Cao, "Towards Context-aware Medical Cyber-Physical Systems: Design and A Case Study," *to be submitted to Elsevier Pervasive and Mobile Computing Journal.*

Conference and Workshop Papers

1. **Tao Li** and Jiannong Cao, "Device Collaboration for Stability Assurance in Distributed Cyber-Physical Systems," *accepted by The 33rd IEEE Symposium on Reliable Distributed Systems (SRDS 2014).*

2. **Tao Li** and Jiannong Cao, "Reliable Closed-loop Control in Medical Cyber-Physical Systems over Wireless Ad-hoc Networks," *The 22nd International Conference on Computer Communications and Networks (ICCCN)*, Nassau, Bahamas, July 30-August 2, 2013.

3. Shaojie Tang, Jie Wu, Guihai Chen, Cheng Wang, Xuefeng Liu, **Tao Li** and Xiang-Yang Li, "On minimum delay duty-cycling protocol in sustainable sensor network,"

IEEE International Conference on Network Protocols (ICNP), Austin, USA, 2012.

4. Lei Yang, Jiannong Cao, Shaojie Tang, **Tao Li** and Alvin T.S. Chan, "A Framework for Partitioning and Execution of Data Stream Application in Mobile Cloud Computing," *The 5th IEEE International Conference On Cloud Computing (CLOUD 2012)*, June 24-29 2012, Hawaii, USA.

5. **Tao Li**, Feng Tan, Qixin Wang, Lei Bu, Jiannong Cao and Xue Liu, "From Offline toward Real-Time: A Hybrid Systems Model Checking and CPS Co-Design Approach for Medical Device Plug-and-Play (MDPnP)," *The 3rd International Conference on Cyber-Physical Systems (ICCPS)*, pp.13-22, Beijing, China, April 17-19, 2012.

6. Chao Ma, Jiannong Cao, Lei Yang, Junjun Kong, Jun Ma, **Tao Li** and Wenwen Cheng, "An Approach to Measuring Social Relationship Based on Users' Location Trails," *The 7th Joint Conference on Harmonious Human*, Beijing, China, September 17-18, 2011.

7. **Tao Li**, Qixin Wang, Feng Tan, Lei Bu, Jian-nong Cao, Xue Liu and Rong Zheng, "From Offline Long-Run to Online Short-Run: Exploring A New Approach of Hybrid Systems Model Checking for MDPnPs," *3rd Joint Workshop On High Confidence Medical Devices, Software, and Systems & Medical Device Plug-and-Play Interoperability (HCMDSS/MDPnP)*, Chicago, IL, April 11-14, 2011.

Acknowledgements

I would like to thank a lot of people who have offered kind help during the past few years of my PhD study. My foremost and deepest gratitude must go to my chief supervisor, Prof. Jiannong Cao. He has not only provided continuous guidance and support to me, but also set an ideal example of good researcher. His grand vision, immense knowledge, rigorous thinking, and systematic approach of studying have generated great influence on my way of defining and doing high-quality research. Undoubtedly, what I have leant from Prof. Cao will pave the way for my future research career.

Also, I am indebted to my co-supervisor, Dr. Qixin Wang, who led me into the exciting research field — Cyber-Physical Systems. My research work in this dissertation has benefited a lot from his invaluable comments and suggestions. Another professor that I would like to thank is Prof. Lui Sha. I am grateful for his offering of visit opportunity to me so that I could study in USA for three months.

Besides, I would like to thank a number of present and past members of our research group, such as Dr. Xuefeng Liu, Dr. Peng Guo, Dr. Vaskar Raychoudhury, Dr. Weiping Zhu, Dr. Chao Ma, Dr. Jingjing Li, Dr. Wei Feng, Mr. Lei Yang, Ms. Yin Yuan, Mr. Zongjian He, Ms. Joanna Siebert, Mr. Chao Yang, Mr. Junjun Kong, Ms. Miao Xiong, Mr. Guanqing Liang, Mr. Rui Liu, Mr. Junhao Zheng, Mr. Yang Liu, Ms. Wanyu Lin,

Mr. Gang Yao, Mr. Chisheng Zhang, etc. I am fortunate to having worked with you and learnt a lot from you. Our big group is indeed a warm family to me.

I am also thankful to many friends in Hong Kong: Mr. Feng Tan, Ms. Xuan Liu, Mr. Zhibao Li, and Mr. Dawei Pan. We shared a lot of happy and tough moments when we all were far away from our hometowns. I really appreciate your time and help.

Last but not least, I would like to express my heart-felt gratitude to my family and girl friend, Ms. Jie Zhou. It is your love, understanding, support and patience that gave me strength to walk through the journey of PhD study.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CPS: Cyber-Physical Systems

FAA: Federal Aviation Administration

FDA: Food and Drug Administration

FPL: First-Order Probabilistic Logic

ICE: Integrated Clinical Environment

LHA: Linear Hybrid Automata

MCPS: Medical Cyber-Physical Systems

MDPnP: Medical Device Plug-and-Play

MPC: Model Predictive Control

NCS: Networked Control Systems

PCA: Patient-Controlled Analgesia

SNZ-LHA: Strongly Non-Zeno Linear Hybrid Automata

Chapter 1

Introduction

In this dissertation, we mainly investigate the coordination problem in Cyber-Physical Systems (CPS). This chapter gives an overview about CPS, the coordination problem, as well as the dissertation's contributions. Section 1.1 serves as the background for understanding CPS and their characteristics. In Section 1.2, we describe the CPS coordination problem in detail, and point out why this problem is different from the one studied in traditional distributed systems. Then, a 3D framework is developed in Section 1.3 for identifying the specific research problems to be investigated in this dissertation. Section 1.4 summarizes this dissertation's contributions. Finally, we outline the organization of the dissertation in Section 1.5.

1.1 Cyber-Physical Systems (CPS)

Human live in the physical world, and most of the time can only directly interact with the physical world around them. However, this situation will be changed in the near future. Thanks to the rapid advancement of technologies, the physical environment will be saturated by various types of objects with sensing, actuation, computing and communication capabilities. Those objects can be harnessed to connect human with the physical world, and

this will hold tremendous promise to broaden human's interaction with the physical environment in both time and space dimensions. For example, green buildings will become possible if we can remotely monitor the lighting and temperature conditions by sensors embedded in the building, and correspondingly control the appliances to save energy [SGLW08]. We can even imagine that one day green buildings with zero energy consumption can be achieved by recycling the emitted energy. However, such systems are very complex as they involve both cyber components (e.g., communication and computation) and physical components (e.g., physical variables like temperature). The interplay of cyber and physical components poses a lot of challenges in designing and optimizing such systems.

This emerging type of system is called Cyber-Physical Systems (CPS). As defined in [RLSS10], *Cyber-Physical Systems* are physical and engineered systems whose operations are monitored, coordinated, and controlled and integrated by a computing and communication core. So far, CPS have found a wide range of applications in areas, such as environmental monitoring and control, healthcare, transportation, aerospace, electric power grid, etc. In the future, CPS will play a more and more important role in our lives, as human's interactions with the physical world become pervasive. Just like the Internet changes people's way of communicating with each other and sharing information, CPS have been believed to revolutionize the way that human interact with the physical world, and meanwhile they will bring significant societal and economic impacts.

The uniqueness of CPS lies in that it involves both the cyber and physical components, and more importantly, these components are tightly coupled. An overview of CPS is depicted in Fig. 1.1. Users of the CPS are able to interact with the physical world through the cyber world. In the cyber world, sensors, actuators, controllers and various

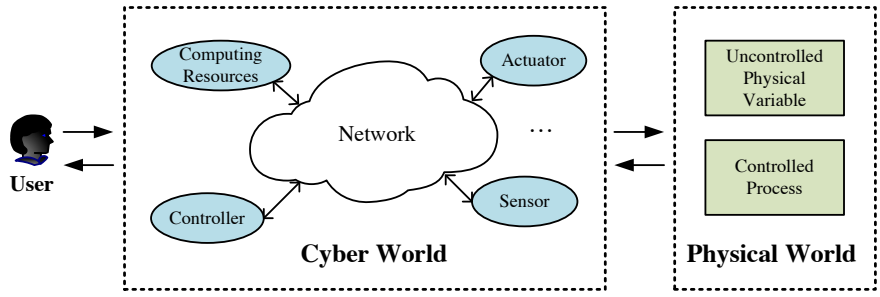


Fig. 1.1: Overview of a Cyber-Physical System.

computing resources are connected through the network. Relying on network allows the devices to be geographically distributed in different places and embedded in the physical environment. It will greatly extend human's awareness and control over remote physical environment. Furthermore, wireless technologies can be adopted for inter-device communication, thus bringing great freedom for device mobility and enabling seamless integration of human-physical world interaction into people's lives.

As shown in Fig. 1.1, the physical world consists of certain *controlled processes* and various uncontrolled physical variables. The former are named according to the conventions in control theory [GK09], and basically refer to the entities that are monitored and controlled by the system, such as the room temperature in the previous green building application. The latter are physical entities not under the system's control; instead, they form an external physical environment of the CPS. For example, in the green building application, the humidity of the room environment is an uncontrolled physical variable which, however, could affect the functioning of the CPS devices, e.g., the measurement of temperature sensors.

1.1.1 Characteristics

In fact, CPS are a natural evolution from traditional distributed systems, control systems, and real-time embedded systems. So CPS actually share a lot of common features with those systems. However, CPS are more than any single of them, because CPS are basically a confluence of these systems [RLSS10]. Listed as follows are some prominent characteristics of CPS.

- **Combination of continuous and discrete dynamics.** In CPS, the controlled process basically follows certain physical laws to change (i.e., continuous dynamics), whereas the cyber components (e.g., computation and networking) exhibit discrete dynamics. In other words, CPS are composed systems with both continuous and discrete natures. Such systems are also referred to as *hybrid systems* [Tab09].

- **Real-time requirements.** Timeliness is a great concern in a wide range of CPS applications. Missing deadline of performing certain tasks is unacceptable. One example is in Medical Cyber-Physical Systems [LS10]: suppose the activation of a lift-supporting medical device (e.g., ventilator that sustains patient's respiration) is delayed, serious medical accidents can be caused.

- **Safety-/Life-critical.** Applications of CPS can be found in many safety-/life-critical environments, such as medical care, transportation, and aerospace. Hence, robustness and reliability are desirable or even indispensable features to those CPS. In addition, the safety-/life- nature also gives rise to a stringent requirement on the quality of those CPS. Examples can be found in areas like avionics and medical care. Prior to being adopted for real use, it is obligatory for a CPS system to go through rigorous

quality examination by the government authorities.

1.1.2 CPS Example: Medical Cyber-Physical Systems (MCPS)

A typical sort of CPS, called Medical Cyber-Physical System (MCPS) [LS10], will be briefly introduced in the following to enhance readers' understanding about CPS. Also, in this dissertation we will extensively use MCPS applications as examples to study various research problems in CPS coordination. A comprehensive review of MCPS will be given in Section 2.1.

Medical procedure are usually very complex, and involve many different medical devices. MCPS have been proposed in the medical domain for integrating and coordinating medical devices to reduce medical accidents. In such systems, patient is a physical process of our central concern. Patient's physiological parameters (e.g., blood pressure) are physical variables to be controlled by the system and may follow sophisticated ways to change. Medical sensors and actuators are connected through network, and their activities are coordinated with the objective of maintaining the physiological variables within safe ranges and meanwhile avoiding various hazardous accidents, such as surgical fire.

MCPS are safety-critical and time-critical. Undoubtedly, patient safety can not be sacrificed in any circumstances. Delayed or improper control of medical devices is likely to cause serious consequences to patient, and thus has to be prevented. Furthermore, MCPS should be robust to many unexpected situations. For example, it has to be resilient to network/device failures and various human operation errors. Even if misoperations are conducted by medical professionals, it is MCPS's responsibility to avert the potential harms to

patient. Also, to become available in the market, MCPS have to go through rigorous examination by regulatory agencies, like Food and Drug Administration (FDA) in US [L+06].

1.2 Coordination in CPS

Now let us discuss a fundamental problem in CPS, i.e., coordination. A CPS device can be a sensor, actuator, controller, or any cyber world entity participating in the system. In general, those devices are distributed in different places, and communicate with each other through network; they are loosely coupled, may not share a global clock, and can fail independently. To compose as a whole and further perform complex tasks, individual devices need to be coordinated. To be specific, *coordination* in CPS refers to coordinating the activities of individual devices to achieve certain goals. A typical example of coordination goal is to maintain certain desired properties for the system, e.g., in MCPS patient's physiological parameters should always remain within clinically safe ranges.

On one hand, goals of coordination may vary depending on the application contexts. For example, the primary goal of coordinating medical devices in medical environment is to prevent the occurrence of medical accidents. But in train control CPS applications, coordination of railway equipment aims at guaranteeing the freedom of train collision and derailment. On the other hand, in terms of the mode of coordination, CPS devices can be either controlled by a central entity, or able to coordinate their own behaviors and collectively perform complex tasks. For example, in MCPS, a group of medical devices can be coordinated by a central controller (called supervisor) to maintain the stability of patient's physiological parameters and avoid medical accidents [A+10]. Whereas, in the military application, unmanned aerial vehicles (UAVs) can autonomously communicate with

their neighbors and adjust their own coordinates to maintain certain tactical topology.

1.2.1 Comparison to Traditional Distributed Systems

In fact, the term of "coordination" has been widely used in traditional distributed systems [C+11]. In this realm, people are interested in a number of different coordination problems, such as global snapshot recording, termination detection, mutual exclusion, reaching consensus, etc. Actually, each of these coordination problems has received tremendous research efforts from the academia and industry during the past decades.

Coordination problem in CPS faces some common challenges as the one in traditional distributed systems, such as, no perfect system-wide clock synchronization, asynchronous communications, etc [KS11]. However, it also differs in many ways from the latter one, and hence demands a new approach to studying it. The following summarizes some distinctive features of CPS coordination in comparison with the traditional distributed systems.

- **Real-time constraints.** Traditional distributed systems usually do not consider real-time constraints. For example, in the case of mutual exclusion, the primary concern is that at most one process is allowed to access the critical section at any time [Sin93]. But one process can wait for a long time to get access permission. In contrast, CPS are usually real-time systems, which means that missing the deadline of performing certain actions will result in failure of coordination. As a result, lack of global clock among distributed devices becomes a more serious issue in CPS coordination.

- **Special concern on physical components.** Physical processes are an integral part of a CPS; their dynamics are usually governed by physical laws and subject to environmental noises [SGLW08]. Sometimes the central concern of CPS coordination

is on the controlled processes, rather than communication, computation or system users. In other words, the main goal of coordination in CPS is to control the physical processes properly, e.g., maintain patient's physiological parameters within safe ranges in MCPS. However, such goal commonly does not fall in the interests of coordination in traditional distributed systems. That is why the mechanism of closed-loop control has been intensively used in CPS for maintaining the stability of physical processes. Also, it is worthwhile to mention that change of the physical processes (e.g., temperature change) is *continuous* and generally much *slower* than cyber world computations. So, some coordination problems extensively studied in traditional distributed systems, like checkpoint and rollback recovery, become less important in CPS, because the CPS contains continuous aspect and can not instantly jump to the previous checkpoint state. In some extreme cases, a CPS is even unable to rollback to the previous state [T+13]. Consider an example in MCPS scenarios: after opening a patient's chest during a surgery, it is impossible to undo the surgical procedures and make patient return to previous state.

- **Interplay of cyber and physical components.** Coordination of CPS devices (cyber components) can not be isolated from the physical components; the cyber and physical components frequently interact with each other. Hence, separation of the physical components from the coordination mechanism design is likely to cause failure of the system or performance degradation. One example is in a CPS involving a cardiac pacemaker and patient's heart [JPM11]. Coordination of the sensor, actuator and controller inside the pacemaker is triggered by the activity of patient's heart,

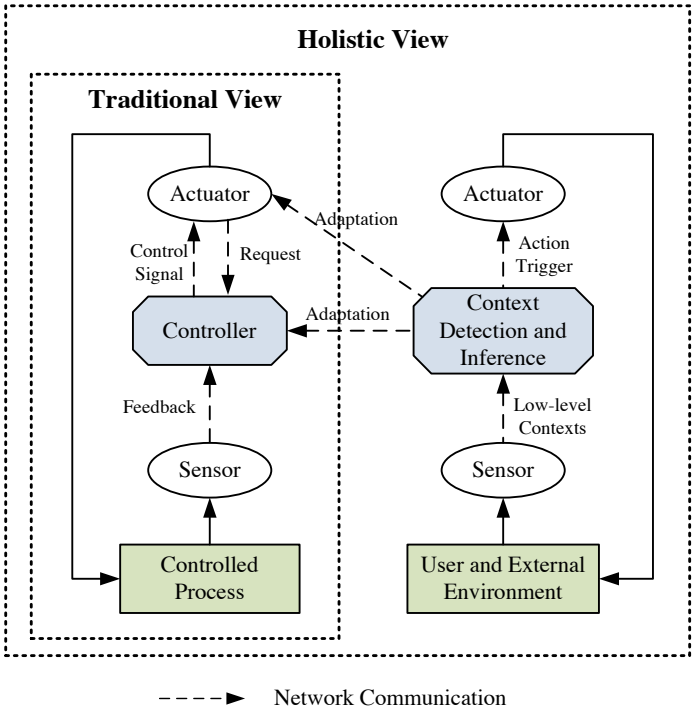


Fig. 1.2: Two views on designing device coordination in CPS.

and in turn tries to adjust the heart's behavior so as to maintain an adequate heart rate for the patient. In this sense, it is necessary to *jointly* consider the cyber and physical components in the design of coordination mechanisms for CPS. Obviously, in traditional distributed systems, generally there is no need to do so.

1.2.2 Two Different Views of Designing CPS Coordination

In the following, we will introduce two different views of designing coordination mechanism: *traditional view* and *holistic view*. As shown in Fig. 1.2, the traditional view of designing coordination mechanism respects the tight coupling of cyber and physical components in CPS. For example, physical processes, sensors, controllers, and actuators can form a distributed and closed-loop system. But human (user) factors and the system's external

physical environment are neglected. Most of existing work on CPS coordination actually falls in this category. However, given the randomness of user's inputs and operations in reality and the complicated impact of external physical environment on the CPS, it is a formidable problem to design robust and safety-assured CPS.

In our opinion, consideration of the human and external environment aspects in the device coordination design would be a prudent choice because of the following reasons. First, users typically rely on manual operations to interact with CPS, which are unfortunately error-prone. Understanding the users' status (e.g., identity and behavior) can be helpful to recognize the erroneous operations and prevent the resulting harms, especially in safety-critical environments. As a consequence, the safety and smartness of the system can be improved. Second, the correct functioning of a CPS device may depend on its knowledge about the contextual information, e.g., the ambient environment situations. Among the numerous examples, listed as follows are two in medical domain. i) Patient's mean arterial pressure (MAP) is an important physiological parameter and should be monitored in many medical cases. The reading of MAP actually relies on the relative position between the MAP sensor and the patient [LS10]. So an accurate MAP reading can be obtained only if this relative position is known to the MAP sensor. ii) Pacemakers are implantable devices for patients with heart diseases. In order to provide sufficient blood supply to patient's body, it is very important for a pacemaker to understand the current activity of the patient. If the patient is doing exercise, obviously higher heart beat rate should be maintained, in contrast with the case of sleeping.

Having recognized the above reasons, we thus propose a holistic approach of designing

device coordination mechanism, where user factors and the external environment are regarded as contextual information to the CPS (see Fig. 1.2). Empowering the CPS with *adaption* capabilities to these contexts has the potential to improve the system's safety and robustness, enable autonomous CPS operation, as well as provide customization for users. For this purpose, device coordination in the holistic view needs to incorporate the devices in the traditional view, as well as the devices for supporting context acquisition and reasoning. However, new problems will arise while enabling context-awareness for CPS. We will elaborate this issue later in Section 1.3.

1.3 Motivation of the Dissertation

Now that we have understood the special characteristics of CPS coordination and the necessity of studying this problem, in this section we will propose a 3D framework (shown in Fig. 1.3) to illustrate the specific research problems to be solved in this dissertation.

The proposed 3D framework has three axes named *Coordination Goal*, *Application*, and *Uncertainty*, respectively. First of all, Coordination Goal axis specifies the goals to be achieved by a coordination task. Some examples of coordination goals include cooperation of CPS devices for verifying and maintaining certain system properties, and adapting to external situations. To be concrete, the following explains the goal of maintaining system properties. In practice, there exist many properties preferred by the CPS, such as safety, liveness, fairness, etc. Violation of those properties may result in serious consequences. For example, electrical power grid CPS application has a stringent safety requirement, called *free of blackout* [RLSS10]. If such safety property can not be preserved, huge amount of economic loss can be caused. As a matter of fact, maintaining certain system properties is

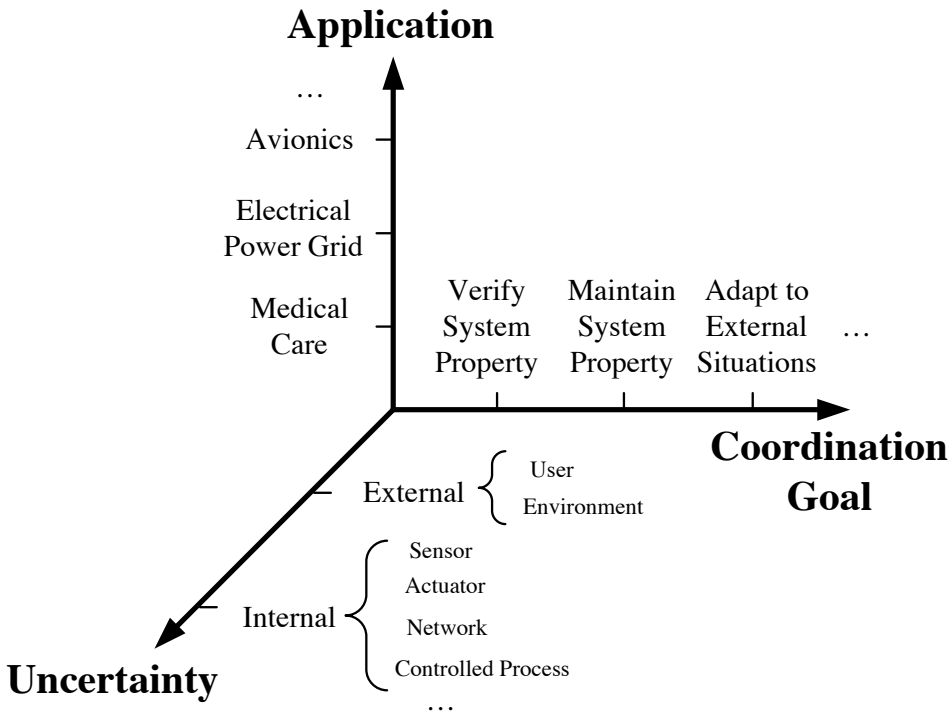


Fig. 1.3: A 3D framework for the research of CPS coordination in the dissertation.

crucial to the success of many CPS applications.

The second axis, i.e., Application, then embodies the specific application domain for a coordination task. So far, CPS has found applications in a wide range of domains, such as medical care, electrical power grid, transportation, aerospace, etc. Certainly, each domain will pose special requirements, and one coordination goal may have distinct interpretations in different application domains. For example, maintaining safety property would refer to free of blackout in the electrical power grid application, whereas its meaning will change to avoidance of medical accidents in the medical care application. In this sense, a coordination goal is constrained by the specific CPS applications. In other words, a coordination goal can be well and unambiguously defined only within a concrete and meaningful application

environment.

The last axis, i.e., Uncertainty, points out various uncertainties which can hinder the fulfillment of the coordination goals. The uncertainties can stem from either the *internal* CPS entities (e.g., CPS device, network and the controlled process), or the *external* entities (e.g., user and uncontrolled physical process). The existence of those uncertainties are so pervasive due to the harsh environment where CPS applications are situated. In real world, there exist many unexpected problems coming out, which actually impede the robustness and reliability of CPS applications. In many cases, uncertainties are recognized as a big challenge to CPS research [SGLW08].

From the 3D framework, we can find that by combining the choices from these three dimensions, it is likely to yield a meaningful research problem in the form of *Coordination Goal-Application-Uncertainty*. This observation actually has guided us to identify the following research problems which serve as the motivation of our research in this dissertation.

- **Problem 1: Verify System Property-Medical Care-Patient Uncertainty.** In the area of medical care, formal verification is a highly recommended procedure by regulatory agencies to examine the correctness of a medical device/system design. For MCPS, it could help to check whether the designed device coordination mechanism satisfies certain desired properties, like safety [L$^+$06]. The tight coupling of cyber and physical components in MCPS requires us to model the dynamics of both computation/communication and the controlled process (i.e., patient) as a whole, and construct a combined model (i.e., hybrid system model) [Tab09]. Then, we should conduct verification over the hybrid system model. However, it is not uncommon that there is

no available model that can characterize the patient's response to medical treatments (e.g., inject a dose of medication), or the model is too complex (e.g., multi-parametric and non-linear) to be supported by the existing hybrid systems verification tools, such as HyTech [HHWT97] and PHAVer [Fre05]. This is mainly because of the complicated bio-chemical reactions happening in patient's body and our insufficient knowledge on pharmacodynamics. The dynamics of patient's body exhibits great *uncertainty* and high variability among different patients, which eventually prohibits the traditional offline verification. However, it has been found that modeling of patient's dynamics turns to be tractable if it is restricted within a short period of time. In fact, patient's physiological variables commonly change smoothly and regularly in a short time span [L$^+$12b]. This thus gives rise to an opportunity for coordinating the MCPS devices to build an online short period model for the patient, and then conducting formal verification based on the online model. But how to coordinate the devices to enable online verification and further respond appropriately to the verification results is a new problem that needs to be studied.

- **Problem 2: Maintain System Property-Medical Care-Network Uncertainty.** In medical environment, maintaining patient safety is a crucial concern. However, this goal is not easy to achieve because the network for connecting the MCPS devices may encounter various *uncertainties*, such as message loss and varied transmission delay. The main reasons for network uncertainty are two-fold. First, wireless technologies have been gaining increasing adoption by medical devices because of the advantages, like easing system deployment and posing less restriction on people's

movement [KSM+10]. However, wireless communication is intrinsically unreliable; there exist a variety of factors, like multi-path effect and self-fading, which can cause message loss. Second, even if wired connection is used by MCPS devices, people can occasionally trip over the wires, resulting in disconnection [Hof07]. Unfortunately, those network uncertainties can have serious impacts on MCPS because of MCPS's stringent safety requirements. Suppose a 'start' signal from the controller gets lost in transit and never activates the ventilator for oxygen supply, patient may suffer irreversible organ damage. In this case, it is highly desirable to design a device coordination mechanism which can preserve patient safety even in the presence of network uncertainties.

- **Problem 3: Adapt to External Situations-Medical Care-Context Uncertainty.** In the holistic view of designing coordination mechanism (see Fig. 1.2), we have explained the importance of adaptation to external situations related to users and environment. However, the contexts obtained by the CPS are likely to be *uncertain*. Such context uncertainty problem has been well recognized by the Pervasive Computing research community [RAMC04a]. In real world, physical sensors for acquiring contexts are subject to noise and failure, and the reasoning techniques for determining the high-level contexts may not be perfect. As a result, a context can be uncertain in many aspects, such as accuracy, confidence level, ambiguity, etc [B+10]. Adaptation to uncertain contexts may result in execution of unnecessary or even risky actions, which is definitely unacceptable in the safety-critical MCPS applications. For example, stopping a life-supporting medical device because of a context which does

not hold in reality (i.e., false positive) may be detrimental to patient's life. Therefore, while we enable context-awareness for MCPS to increase the system's smartness and autonomous capabilities, it is also necessary to overcome the context uncertainty problem and meet the safety requirements.

1.4 Contributions of the Dissertation

The contributions of this dissertation mainly lie in the novel device coordination mechanisms that we have proposed to overcome various uncertainties emerging from the CPS applications. To be specific, we have systematically studied the research problems identified in the previous section. The following summarizes our contributions in solving these research problems, respectively.

First, with respect to Problem 1, we choose *model checking* as the formal verification method, and endeavor to enable hybrid systems model checking for the MCPS even if the patient model is not available or too complicated. Our proposed approach is to transform the traditional offline hybrid systems model checking into *online* version. This is motivated by the observation that patient's physiological state can be very predictable in the near future, and thus can be readily modeled. To be specific, MCPS devices can periodically coordinate to generate the overall system model for the coming short period of time, and then model checking the system's behavior. A MCPS is permitted to continue running under current configuration only if the online model checking result proves the future system execution is safe; otherwise, the system is forced to switch to a fall-back plan. All the MCPS devices follow this rule to coordinate their behaviors, thus avoiding any possible harms to patient. Furthermore, we also propose several system co-design patterns by joint consideration of

the real-time requirement and the decidability of online model checking. These patterns provide useful guidelines for building MCPS that can take online model checking as an effective procedure for continuous fault prediction and risk prevention. In addition, we further discuss the effectiveness of our approach under situations where communications are not reliable and online modeling of patient's dynamics is not accurate. We believe our approach finds a new direction for overcoming the difficulty of patient uncertainty, and model checking MCPS. Experimental results based on a concrete medical case, named Airway-laser Surgery, have validated the feasibility of our approach under diverse settings.

Second, we have designed a new device coordination mechanism for MCPS to solve Problem 2. Existing mechanisms with capabilities of tolerating network uncertainties in M-CPS, like [KSM+10], are *time-triggered* and depend on periodic message exchange between MCPS devices. In other words, coordination is initiated periodically, and consequently can not respond to surgeon's request promptly. This issue leads the solution to be inapplicable in many delay-sensitive medical procedures. To deal with this issue, we opt for an *event-triggered* approach, that is, whenever needed, the coordination task will be started immediately. Our mechanism runs based on a centralized architecture defined by the Integrated Clinical Environment (ICE) standard [AST09]. In the architecture, a central controller, called supervisor, is responsible for overseeing the states of each medical device and makes decision to coordinate their behaviors to avoid medical accidents. To deal with network uncertainties, we empower the supervisor with the capabilities of planning the future operations for each medical device, instead of making control decision for the current time point. The planned operations are determined based on the worst-case prediction of patient's state in the near future. Therefore, as long as the supervisor has validated the

safety property for the planned operations, each medical device is able to run safely by consuming the decisions from the planned operations when communication problems occur. We choose this planning-based method because patient's state changes constantly; we can not retransmit the lost messages for many times until success, or restore to the past system states (e.g., by checkpoint) [T+13]. In fact, this planning-based approach works in a manner similar to Model Predictive Control (MPC) in the industry control domain [Cam04], where control signal is periodically generated only for the coming short period of time based on the estimation of plant's future state. Nevertheless, MPC is not designed for tolerating network uncertainties and does not consider user's interactions, i.e., in MCPS, users like surgeon can request to use the medical devices for surgical purposes and change medical devices' states. At last, we further tune the proposed mechanism to allow it workable in the environments without perfect clock synchronization. Through trace-driven simulations, we have shown that our mechanism outperforms the state-of-the-art approach [KSM+10] in many aspects, such as clinical efficiency, response delay, and communication cost.

Third, we have extended the application of context-awareness concept to the safety-critical environment, e.g., MCPS. To date, very few existing works on device coordination in MCPS take user or environment related contexts into consideration. On the contrary, our research not only utilizes those contexts to improve patient safety and user's experience, but also can tame context uncertainty (Problem 3). Context uncertainty is a well acknowledged issue and inevitable in the real world. Existing works on context uncertainty mainly focus on modeling, inference, and mitigation of uncertainty [B+10][DM05][XC05]. They do not consider the impact of context uncertainty, especially on patient in the medical settings. To protect patient in the case of utilizing uncertain contexts, first we allow the MCPS devices

to coordinate to reduce the degree of context's uncertainty by methods, like multi-modal sensing [RMJ+11], depending on the specific aspects of uncertainty associated with the contexts. Then, we carefully design the adaptation rules regarding each context, so that the context-triggered actions permitted to be executed are always free of harm. To be specific, we propose a novel adaptation style, named *"context-assessment-action"*, where an "assessment" step is intentionally added to filter out all the risky context-triggered actions. In this way, we can minimize the adverse impacts of context uncertainty on patient safety. In addition, we have built a prototype context-aware system for a specific medical procedure, named Patient-Controlled Analgesia (PCA), and deployed it in a ward-like environment. Evaluation results about the system demonstrate that our approach can achieve significant safety improvement while capable of suppressing the rate of false context detection under very low level.

1.5 Organization of the Dissertation

The structure of the dissertation is organized as follows. Chapter 1 is the introduction of this dissertation, where we give an overview of CPS and the coordination problem, as well as a summary of our contributions to the community.

Chapter 2 gives some further background knowledge and reviews the existing literature related to our research. As we extensively use MCPS as a CPS application to study various coordination problems, it necessitates a detailed review of MCPS. So we will present a comprehensive description about the Integrated Clinical Environment (ICE) standard for MCPS, and illustrate the standard by two medical examples, i.e., Airway-laser MCPS and Patient-Controlled Analgesia MCPS. Note that these two examples will also serve as case

studies for evaluating our proposed approaches in the following chapters. Besides, this chapter also studies the related work in three specific topics: model checking, tackling network uncertainties, and handling context uncertainties.

In Chapter 3, we endeavor to enable model checking for MCPS in the case of no tractable model for the patient. We investigate how to coordinate medical devices to build the online system model, and furthermore propose several system co-design patterns that jointly consider the decidability of online hybrid systems model checking and the real-time requirement. This co-design approach has been evaluated in a concrete medical case, i.e., Airway-laser MCPS. The content of Chapter 3 is published (or to be published) in the following IEEE/ACM papers:

In Chapter 4, we introduce the planning-based approach to solve the network uncertainty problems in MCPS. Closed-loop control and safety interlock are two standardized mechanisms for coordinating medical devices to protect patient from potential harms. We will show how these two mechanisms still work even in the presence of network uncertainties. Again, we will demonstrate the effectiveness and efficiency our approach in the Airway-laser MCPS. The content of Chapter 4 is published in the following IEEE paper:

- Copyright © 2013 IEEE. Reprinted, with permission, from *Tao Li and Jiannong Cao, "Reliable Closed-loop Control in Medical Cyber-Physical Systems over Wireless Ad-hoc Networks," The 22nd International Conference on Computer Communications and Networks (ICCCN), Nassau, Bahamas, July, 2013.*

Chapter 5 aims at handling the context uncertainty problem when building context-aware MCPS. We analyze the severity of context uncertainty in the medical scenarios, and then propose an approach to taking advantage of contexts while preventing the negative consequences caused by context uncertainty. In addition, we develop a context-aware MCPS for Patient-Controlled Analgesia, and conduct extensive experiments to prove the effectiveness and advantages of our approach.

Finally, Chapter 6 concludes the dissertation with a summary of our research work, and points out some future research directions.

Chapter 2

Background and Literature Review

We start this chapter with some background knowledge about MCPS and two real-world MCPS applications. The background knowledge will assist the readers to gain a deeper understanding about the characteristics of Cyber-Physical Systems (CPS). Also, these MCPS applications will serve as case studies for our research in the following chapters. Following the background, we will review the existing literature related to the three problems (Problem 1, 2 and 3 in Section 1.3) we are going to study in this dissertation, respectively. Specifically, Section 2.2 introduces existing work on model checking and its applications in discrete and hybrid systems; Section 2.3 introduces the existing work on dealing with network uncertainties in the application context of physical process control; Section 2.4 presents the existing work on context uncertainty modeling, reasoning and resolution.

2.1 MCPS Standard and Applications

As a typical type of CPS, MCPS have many unique features. Recall that in Section 1.1.2 we give a concise description about MCPS. In the following, we will look into the details about MCPS, including their emergence, standardization, and some real application examples.

2.1.1 Integrated Clinical Environment (ICE) Standard

During the past years, the large number of medical accidents have attracted tremendous attentions from the society and regulatory departments. As stated in a well-known report, entitled "To Err is Human" [KCD00], it is estimated that at least 44,000 Americans die in hospital each year as a result of medical errors, most of which actually are preventable. This estimated number of preventable death can be even higher than 98,000, as stated in an MIT Technology Review article [Gri08]. Undoubtedly, there is an urgent need for improving healthcare safety. One of the leading roles in this direction is the Medical Device Plug-and-Play (MDPnP) Interoperability program [Med04]. Since its establishment in 2004, the program has been continuously promoting the interoperability among diverse medical devices, thus creating possibilities for building patient-centric *Integrated Clinical Environment (ICE)*. Basically, the concept of ICE is proposed to create medical systems which integrate interoperable medical devices and other equipment for the care of high acuity patients [AST09].

One clinical benefit of creating ICE is that automatic control/coordination of medical devices can be implemented to improve the efficiency of clinical workflow and the system's ability of resisting human errors. Meanwhile, the integration and coordination of medical devices combined with patient physiologic dynamics have generated a new type of system, called Medical Cyber-Physical System (MCPS) [LS10]. Such system is a typical CPS, as it involves both cyber components (e.g., embedded medical device software and network) and physical components (e.g., patient's body). To facilitate medical device integration and coordination, MDPnP Interoperability program has developed an ICE architecture, which

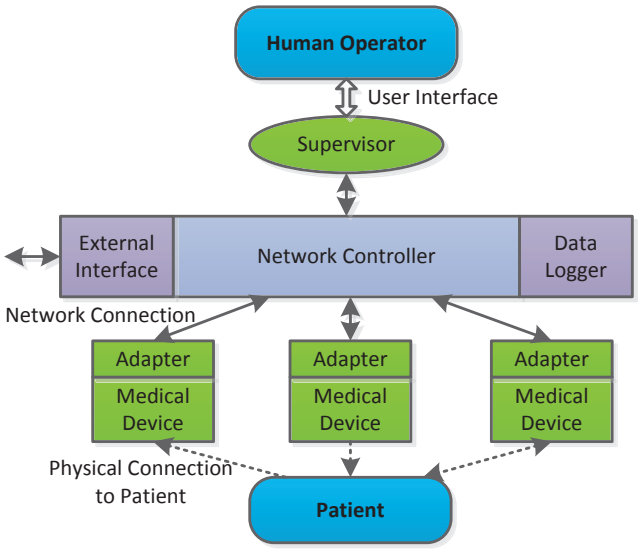


Fig. 2.1: Integrated Clinical Environment (ICE) architecture and some main functional components.

specifies the components in an ICE and the functional role of each component. In year 2009, this architecture was published as an ASTM standard [AST09]. So in the rest of this dissertation, all the MCPS discussed will be compliant with this standard.

Fig. 2.1 shows a diagram about the ICE architecture. Several major functional components in ICE are *medical device*, *supervisor*, and *network controller*. Medical devices are physically connected with patient for diagnosis or therapeutic purposes. Generally, medical devices can be categorized into two groups: *monitoring devices* and *delivery devices* [L+12a]. The first group, such as heart rate and blood pressure sensors, monitors patient's physiological state. The second group, such as infusion pump and laser scalpel, delivers therapy to patients and is capable of changing patient's state. Supervisor is an entity providing application logic for the ICE, such as clinical decision support and mechanisms for coordinating the medical devices. Supervisor and medical devices are connected through network;

the network controller component provides communication support for the aforementioned ICE devices. In addition to various equipment, there are two human roles related to the ICE architecture. The first role is patient who is the recipient of the medical treatment; the second is human operator who can configure and operate the ICE devices.

Another feature of the ICE architecture is its ability to support the implementation of *closed-loop control* and *safety interlock*, which are two effective coordination mechanisms in reducing medical accidents (documented in [AST09]). Even though these two mechanisms are not formally described, the published ICE standard has illustrated them by several real medical cases (see annex of reference [AST09]). In the following, we will introduce two real-world cases to help readers understand how closed-loop control and safety interlock mechanisms work.

2.1.2 Patient-Controlled Analgesia MCPS

The first medical case we introduce is Patient-Controlled Analgesia (PCA). Patients may experience severe pain in many situations, e.g., after receiving a surgery, and thus need pain relief. PCA is designed to enable patients to *self-administer* the delivery of analgesic drugs, thus giving them a sense of self-controlling the pain level. The cornerstone medical device in PCA is *PCA infusion pump* (or *PCA pump* for short), which is a sophisticated microprocessor-controlled infusion pump. Each PCA pump provides a bolus button. By pressing the button once, patients are able to request a dose of medication (interchangeable with drug in this dissertation) called *bolus dose* delivered into the body intravenously for pain relief. However, the term "PCA" is not limited to a single medical device. Rather, it generally refers to a medical process which involves a number of people, including prescriber,

pharmacist, nurse, and patient, etc [D'A08]. Prescriber orders the appropriate drug (often opioids) and PCA pump settings involving a number of pump programmable parameters, e.g., drug concentration, lockout and dose limit, etc. Finally, according to prescriber's instructions, nurse then picks up drug syringes/bags from pharmacist and configures the PCA pump for patient use. The following shows an example of prescriber's order:

1. Drug: morphine
2. Drug concentration: 1.0 mg/mL
3. Background infusion: 1.0 mg/h
4. Bolus dose: 1.0 mg
5. Lockout interval: 6.0 minutes
6. Dose limit: 10.0 mg in 1 hour.

Parameter 1 and 2 indicate the drug type and concentration, respectively. Drugs of standard concentrations are commonly available as prefilled syringes/bags. Nurses are responsible for loading the syringes/bags on the PCA pump. Parameter 3-6 are typical programmable variables in PCA pump. *Background infusion* indicates the continuous drug delivery regardless of patient's request of bolus dose. In addition to background infusion, patients can request more drugs by pressing the bolus button on the PCA pump. *Lockout interval* is a time period within which a patient can receive at most one bolus dose. Therefore, only one dose will be delivered in the lockout interval even if he/she presses the button more than once. *Dose limit* specifies the maximum amount of medication a patient can receive during a specific interval, i.e., *dose limit interval*. In the example above, at

most 10.0 mg can be injected to a patient within 1 hour. The main purpose of the above parameter setting is that only limited amount of drugs are allowed to be injected into a patient's body within a certain time, thus reducing the possibility of overdose [Gra05].

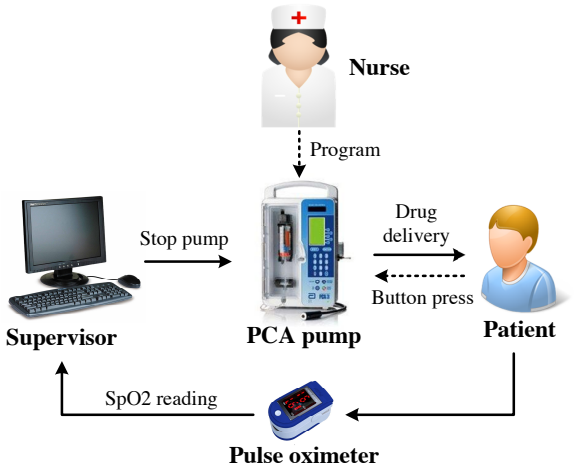


Fig. 2.2: Closed-loop control-based MCPS for Patient-Controlled Analgesia.

Change in respiratory status (e.g., decrease of respiratory rate) is a leading indicator of adverse patient response to PCA drugs. The major PCA safety issue is *overdose* (excessive drug delivery) which can lead to a fatal complication, i.e., respiratory depression. If the PCA pump is set properly, overdose problem can be effectively prevented. However, there exists drastic variability of patient's response to opioids. Patient characteristics, e.g., age, gender, weight, psychological state, opioid consumption history, concurrent diseases, can affect drug effect. To date, there has been no reliable method of determining how much opioid a patient will require for analgesia [Mac01]. Although with lockout and hourly limit configuration, overdose may still happen before the preset limit is reached. Furthermore, there exist many human errors during the PCA process that can induce overdose, such as drug misuse, misprogramming, PCA-by-Proxy, etc [Coh05]. Statistics from MEDMARX

and MAUDE databases have shown that the annual error rates were estimated as 407 PCA-related errors per 10,000 people within US [M$^+$09], and so far, US Food and Drug Administration (FDA) has received numerous adverse reports about PCA [US 10].

To combat the above safety issues, University of Pennsylvania has developed a closed-loop control based PCA system (shown in Fig. 2.2) [A$^+$10][KAL$^+$10]. In this paper, we call such system a *PCA-MCPS*. At present, such system is still in the demonstration phase and far from wide-spread adoption. The system interconnects a PCA pump, a supervisor, and a pulse oximeter which continuously monitors patient's blood oxygen saturation (SpO2) as an indicator of patient's response to analgesic drug. Based on the feedback of patient's SpO2, the supervisor is able to automatically stop the PCA pump when patient's SpO2 falls below certain threshold, thus preventing the risk of further drug delivery. By forming a physiological closed-loop control, there exist a great opportunity of enhancing PCA safety. In addition to PCA, successful closed-loop control based MCPS can be found in other medical applications like pacemaker [JPM11] and insulin delivery [Hov06] [KCMFL13].

2.1.3 Airway-laser MCPS

The second medical case to be introduced is Airway-laser Surgery. An Airway-laser MCPS (see Fig. 2.3) tailored for the surgery involves following entities: surgeon, patient, O_2 sensor, pulse oximeter, ventilator, and supervisor [L$^+$12b]. The application context is as follows. In the surgery, due to general anesthesia, the patient is paralyzed, hence has to depend on the ventilator to breathe. The surgeon requests the laser scalpel to emit laser so as to cut patient's trachea. When the laser scalpel is to emit laser, the oxygen concentration inside the trachea (measured by O_2 sensor) must be lower than a threshold,

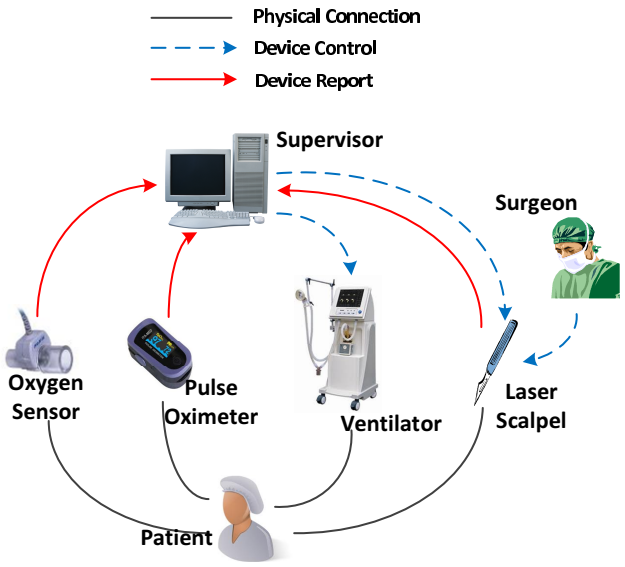


Fig. 2.3: MCPS for Airway-laser Surgery.

i.e., $\Theta_{O_2} = 30\%$. Otherwise, the laser may trigger *surgical fire*, which can seriously burn the patient. Therefore, before the laser scalpel is allowed to emit laser, the ventilator must have blocked the oxygen flow for a while. On the other hand, the ventilator can neither block oxygen flow for too long, or the patient will *suffocate* due to too low blood oxygen level (SpO_2), e.g., below threshold $\Theta_{SpO_2} = 90\%$. To avoid both patient suffocation and surgical fire, basically the MCPS involves following two coordination mechanisms.

- Closed-loop control: Supervisor, ventilator, and pulse oximeter form a closed control loop so as to maintain patient's SpO_2 within a safe range. Once patient's SpO_2 measured by pulse oximeter is detected to be low (i.e., less than 90%), supervisor should activate ventilator to resume oxygen supply.

- Safety interlock: There are interlocks between laser scalpel and ventilator, as well as between laser scalpel and O_2 sensor. It is forbidden that laser scalpel is emitting and ventilator is supplying oxygen simultaneously. Meanwhile, it is also forbidden that

laser is emitting and O_2 concentration is above 30% at the same time. Otherwise, surgical fire is likely to be triggered.

The challenge in such system lies in that closed-loop control and safety interlock are not independent. For example, if supervisor intends to activate ventilator because of the feedback about low SpO_2, it has to guarantee laser emitting has already been stopped in order to respect the interlock requirement between laser scalpel and ventilator. The interleaving between closed-loop control and safety interlock obviously complicates the device coordination mechanism design.

2.2 Existing Work on Model Checking and Its Applications

The safety-critical nature of many CPS applications (e.g., MCPS) necessitates formal verification to uphold our confidence on the designed coordination mechanisms. As a mainstream approach of formal verification, *model checking* will be introduced in this section. Moreover, we will review the existing approaches of carrying out model checking.

Let me start with the introduction about model checking. Model checking is a formal verification technique which can automatically (i.e., without user supervision) check the correctness of a system's design. It allows early detection of defects existing in the software/hardware system design, and thus avoids the high cost incurred when the detection of those defects are delayed to the testing or deployment phases. Basically, model checking relies on a model that describes the system's behavior in a mathematical manner, and formalizes the property to be checked by the specification languages (e.g., temporal logics). Then, it explores all states of the system model to check whether the given property is

satisfied or not [BK08]. This exhaustive search is necessary to rigorously analyze the system's conformance to the defined property. Some common properties of people's interest include safety, functional correctness, liveness, reachability, etc. In the past decades, due to the improvement of the underlying algorithms and hardware, model checking nowadays has been applicable in a wide range of realistic applications and adopted by many organizations, such as FAA (Federal Aviation Authority) and NASA (National Aeronautics and Space Administration) [HP00].

In principle, model checking is just a technique, and thus can be implemented by different algorithms and applied in diverse scenarios. In this section, our literature review will only discuss the existing approaches on the usage of model checking. Since we have proposed a novel online hybrid systems model checking solution to solve Problem 1 (see Section 1.3), we categorize those existing approaches by two dimensions (i.e., offline/online, and discrete system/hybrid system), and study them in comparison with our approach in [L$^+$12b].

Traditionally, model checking is applied in an *offline* manner, that is, construction of the system model and the verification are conducted during the system design phase. On the contrary, *online* model checking refers to the model checking process that is conducted at runtime when the system is running. In addition, depending on the dynamics of the system to be checked, model checking can be applied for *discrete systems* or *hybrid systems*. The former refer to the systems with a countable number of states, whereas the latter can involve both discrete and continuous dynamics, and thus have infinite number of states [Lyg04]. Examples of discrete systems include circuit designs and communication protocols, which were the applications of model checking in the first place [CGP00]. Hybrid

Table 2.1: Categorization of existing approaches in applying model checking

	Offline	**Online**
Discrete System	[BCL+94] [Hol90]	[SRA04] [EKS06] [Q+09] [HKMP02]
Hybrid System	[LPY97] [B+96] [Alu99] [A+10] [PMS+12]	[S+09] [LNKM11] [LMN07] [L+12b]

system examples can be found in many scenarios, like medical and manufacturing environments, where continuous physical variables are controlled by computerized systems. The interaction between discrete and continuous dynamics makes hybrid systems more difficult to be formally verified. In the following, we will describe the related work one by one from a comparative perspective. The categorization for each work can be found in Tab. 2.1.

2.2.1 Offline Model Checking

First of all, since the emergence of model checking, there have been numerous cases of successful applications. However, most of the applications have been conducted in an offline manner. An exhaustive enumeration of them is difficult and not necessary in this dissertation. We refer the interested readers to some notable examples listed in [BK08]. Here, in Tab. 2.1, we list several typical applications of model checking for discrete systems, e.g., circuit designs [BCL+94], communication protocols [Hol90], as well as for hybrid systems, e.g., real-time systems [B+96] [LPY97] [Alu99]. Our special attention has been given to the applications in MCPS which is the focus of our research. MCPS is a typical hybrid system involving continuous patient's physiologic dynamics and discrete dynamics about network and embedded medical device software. In [A+10], the authors have proposed a method for model checking MCPS with patient in the control loop. They construct a timed-automata based model for the MCPS based on UPPAAL tool [LPY97] and verify the system's safety

properties. As a follow-up, Miroslav Pajic *et al.* [PMS+12] improve the work in [A+10] with additional consideration of parametric uncertainty in patient's model, as well as derive some theoretical results for obtaining the correct timing parameters in UPPAAL verification.

2.2.2 Online Model Checking

Now let us switch our focus to the online model checking approaches. Our proposed method in [L+12b] is different from the well-known runtime verification [F+02]. Runtime verification aims to discover latent bugs of programs by logging and analyzing the programs' execution traces under varied inputs/configurations. It is not for predicting/preventing faults before they ever happen; whilst our approach is. For many medical CPS systems, the cost/consequence of possible faults in test runs is high or even unbearable. This necessitates our approach of predicting and preventing faults before they ever happen.

Sen *et al.* [SRA04] propose an online safety analysis method for multithreaded programs. However, this work only focuses on how to infer other potential executions that can take place in the past. Our work tries to predict the future state of patient based on recent observations. Easwaran *et al.* [EKS06], Qi *et al.* [Q+09], and Harel *et al.* [HKMP02] also propose to bring model checking online. But they are still focusing on discrete systems model checking, rather than hybrid systems model checking that our work is about.

Sauter *et al.* [S+09] propose a lightweight hybrid system model checking method, which uses ordinary differential equations (ODE) to predict temporal logic properties. However, in MCPS it is not uncommon to be lack of differential equations governing patient's dynamics, i.e., patient's model. Li *et al.* [LNKM11] propose an online model checking approach aiming at automatically estimating parameters in simulation models, which are often used

Table 2.2: Categorization of related work on handling network uncertainties

	Message Delay	**Message Delay & Message Loss**
NCS	[BPZ00] [MA04] [ZSCH05] [LY05] [L+07]	[X+06] [AS03] [ZY07] [YHL05]
CPS	[KKH+08] [GSC11]	[A+10] [KSYS10a] [KSM+10] [PMS+12] [LC13] [L+13]

for biological purpose to understand complex regulatory mechanisms in cell. Larsen *et al.* [LMN07] propose an online model-based testing tool for real-time systems, UPPAAL TRON. The tool is based on UPPAAL engine [LPY97] and models real-time systems as timed automata, whereas our online model checking of MCPS focuses on more general hybrid systems.

2.3 Existing Work on Process Control with Network Uncertainties

Communication infrastructure has been becoming ubiquitous, and there have been more and more devices emerging with networking capabilities. From spatial point of view, the communication network has greatly broadened our interaction with people and the physical world. However, the network itself has intrinsic uncertainties. For example, wireless communications are not stable; network congestion is common in the Internet; messages in the network can be tampered by attackers. Faced with these problems, design of advanced mechanisms that can tolerate the problems is important, especially in the safety-critical environments.

In this section, we will review the existing work related to handling network uncertainties in physical process control. Here, network uncertainties include various problems, e.g.,

jitter, transmission delay and message loss, which a communication network can exhibit during message transmission. Since our work on handling network uncertainty focuses on MCPS (see Problem 2 in Section 1.3), in this section we will pay special attention to the existing work in the following two areas, i.e., Networked Control Systems (NCS) and Cyber-Physical Systems (CPS), because of the closeness to MCPS. The existing work discussed in the following has been classified and shown in Tab. 2.2.

2.3.1 Literature in Networked Control Systems

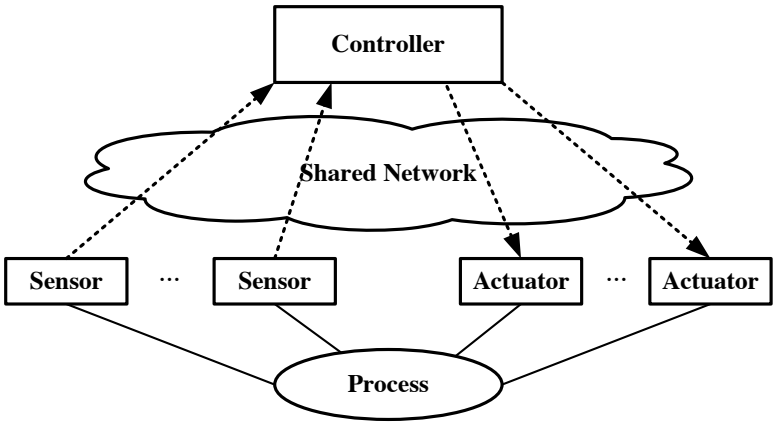


Fig. 2.4: Network Controlled Systems.

NCS are control systems where sensors, actuators and controllers are spatially distributed and communicate through a shared bandwidth-limited network (see Fig. 2.4) [H+07]. In a NCS, control loops are closed through a network which is also shared by other nodes outside the control system [GC10]. Message delay and loss are the most common problems when devices communicate over a shared bandwidth-limited network. They can result in system instability or performance degradation. So far, most of existing literature on NCS have been focused on the impacts of message delay and loss on the control system, or endeavored to countermeasure the adverse impacts by sophisticated controller design [Yan06].

As shown in Tab. 2.2, we have categorized the existing work based on their assumptions about the network conditions, i.e., some only consider message delay, while others consider both message delay and loss.

First of all, many authors assume that the varying message delay is bounded (e.g., less than a sampling period) [BPZ00] [MA04], or driven by a stochastic process (e.g., Markov chain) [ZSCH05] [LY05] [L$^+$07], and then study how to compensate for the delay in the controller design. In addition to message delay, some studies also take message loss into account, such as [YHL05] [X$^+$06] [AS03] [ZY07], to investigate the system's stability. Similarly to message delay, packet loss can also be characterized by different models, e.g., stochastic model or deterministic model. A more comprehensive survey of the state-of-the-art on NCS can be found in [H$^+$07] and [GC10].

From the above literature review on NCS, it is easy to be concluded that in the NCS community, people still investigate the same problems in the traditional control systems, like stability. Moreover, most of the existing results have been obtained based on linear NCS, whereas non-linear NCS and physical plant with uncertainty have received little attention [Yan06]. However, in the problem that our dissertation is going to solve (i.e., Problem 2), it is likely that there is no accurate or linear model for the patient. What's more, in MCPS we are concerned about safety, which is not equivalent to stability. Stability often concerns about the boundedness of the system's output when given arbitrary inputs, whereas safety in MCPS has clinical implications, and commonly refers to the avoidance of any harms to the patient.

2.3.2 Literature in Cyber-Physical Systems

Even though Cyber-Physical System (CPS) shares many similarities with NCS, like control over the network, it differs from NCS in that the system components, including physical process, controller, actuators, and sensors, are more consolidated. In addition, there exist more complicated interactions between CPS devices, e.g., there are uplink communications from actuators to the controller, whereas in NCS actuators are commonly passive and only accept control inputs from the controller. This phenomenon also implies another difference, that is, in CPS we have to take system's user into account; the system user can operate the CPS devices and change device states from time to time.

In this subsection, we will review the existing approaches in CPS that can handle message delay, as well as those that can handle both message delay and loss. Actually, research on CPS is still in its infancy, and there exist limited work on this topic. Regarding the classification of the existing work, we refer the readers to Tab. 2.2.

In [KKH$^+$08], the authors propose a passive control architecture which accounts for time-varying message delays in the network. An advantage of the architecture is that it decouples the controller design from the network uncertainties, thus enabling model-driven development. Dip Goswami *et al.* [GSC11] propose an approach, called control/communication architecture co-design, that jointly designs the controller and the communication schedule so that the closed loop system is stable. In other words, this approach manipulates the message delays in the network along with the controller to achieve system stability.

In contrast with [KKH$^+$08] and [GSC11] which only deal with message delay, the following work further considers message loss. In [L$^+$13], the authors investigate a type of CPS

that can control civil structures (physical plants) according to the feedback of structure status monitored by a wireless sensor network. They first build a simulation environment which can model civil structure dynamics, sensing, and realistic wireless communication with message delay and loss. Then, a cyber-physical co-design approach, which integrates control algorithms and the scheduling strategies for data collection and communication, is proposed to improve structural control performance.

Now let us move to the medical domain. The existing work most relevant to ours are [A+10], [PMS+12], and [KSM+10]. In [A+10] and its subsequent improvement [PMS+12], the authors design a mechanism to tolerate message loss in PCA-MCPS. However, the limitation of their approach is that it only deals with a specific medical case (i.e., PCA), and does not consider the interlock requirements. In this sense, this approach is not applicable to general MCPS cases. In [KSM+10], the authors propose a framework named NASS for medical device coordination, which can tolerate network failures. Its drawback, however, is that it relies on periodic supervisor-medical device interaction. In other words, the NASS framework is time-driven rather than event-driven, thus resulting in frequent message exchange and long response delay to surgeon's operation request.

2.4 Existing Work on Handling Context Uncertainty

As background knowledge, we start this section with explanations of the existence of context uncertainty, followed by a study of existing work. Specifically, we will review the literature on how to handle context uncertainty, including uncertain context modeling and inference, as well as resolution of uncertainty.

Pervasive Computing has been regarded as a computing paradigm for the 21st century [SM03]. Context-awareness is one of the enabling technologies for achieving the vision of Pervasive Computing. Increasing the awareness of the implicit contexts information about the user and physical world helps to build smart environments which can proactively provide services for the user. However, one challenge in context-awareness is that the contexts obtained by the system are uncertain. In fact, such phenomenon has been well acknowledged by the pervasive computing community. The main reasons for context uncertainty are two-fold. First, in real world, physical sensors for capturing contexts are subject to noise and failure. Besides, people are faced with limited choices of sensors for monitoring the physical world [DM05]. As a result, the inaccurate and limited data may not be able to reflect the real world situations. Second, context processing and reasoning techniques are not perfect. For example, a biometric based algorithm (e.g., recognize user identity by fingerprint) to determine people's identity is likely to generate false positive and false negative results.

As such, context uncertainty is an inevitable problem that context-aware applications have to face. To handle context uncertainty, existing research mainly endeavors to answer two questions: i) how to provide a unified model for uncertainty to ease context reasoning and application development? ii) how to resolve or mitigate uncertainty to increase the quality of contexts? The following two subsections will discuss the existing work on these two questions, respectively.

2.4.1 Uncertain Context Modeling and Inference

In real world, context uncertainty can stem from different factors, e.g., lack of precision about the sensor measurements or lack of timeliness. Intuitively, people thus have proposed corresponding quality metrics to describe those factors, and then organized them in a unified way to represent the overall uncertainty. In [BKS03], the authors describe five quality metrics that are most important ones from the authors' viewpoint; they are precision, probability of correctness, trust-worthiness, resolution, and up-to-dateness. However, this work does not discuss how to model those metrics. In [GS01], the authors consider six quality metrics, e.g., coverage, resolution, accuracy, repeatability, frequency, and timeliness, which are all modeled as meta-information associated with the contexts. Hui Lei *et al.* [LSD+02] develop a context collection and dissemination middleware, named Context Service, which allows context information to be associated with quality metrics, like freshness, confidence, error, etc. In addition to those general quality metrics, there have been many context-aware systems which only consider the metrics related to certain particular types of contexts, such as location [HBB02][RAMC+04b].

The above work has made early contributions to the research of handling context uncertainty, and provided many insights about context uncertainty. However, a major drawback of them is the *lack of formality* for representing the quality metrics. As a result, they can not support advanced context reasoning and uncertainty resolution. To overcome this drawback, there have been a wealth of follow-up work incorporating formal techniques, such as probabilistic logic, fuzzy logic [RAMC04a], Bayesian networks [CCKM01], hidden Markov model [OHG02] [KHC10], Dempster-Shafer theory [Wu03], ontology [BGT+06], etc.

In [RAMC04a], the authors first propose an approach of modeling contexts by predicates. Then, based on such model, a variety of techniques for uncertain context reasoning, e.g., probabilistic logic, fuzzy logic and Bayesian networks, can be developed. In [OHG02] [KHC10], the authors are focused on one type of context, i.e., human activity, and use hidden Markov model to support representation and inference of human activity. Huadong Wu *et al.* [Wu03] apply Dempster-Shafer theory to handle context uncertainty during data fusion process. As shown in the paper, Dempster-Shafer theory determines the probability of an event by combining separate pieces of information, and thus can be easily mapped to the data fusion process. In [BGT$^+$06], the authors propose an ER-ontology based model to manage various context quality metrics, e.g., delay time, context correctness probability, and context consistency probability. The strong expressiveness and reasoning power of ER-ontology allow the implementation of advanced algorithms to resolve context inconsistency.

2.4.2 Context Uncertainty Resolution

Even though uncertainty is an inevitable problem with contexts, there exist many approaches which can be utilized to resolve the uncertainty. As there exist many quality metrics that can affect context uncertainty, the uncertainty resolution approaches differ in their abilities in solving the specific uncertainty aspects. In this subsection, we will focus on three aspects: *confidence level*, *inconsistency* and *ambiguity*. Related work on dealing with those uncertainty problems are described as follows.

First, confidence level refers to the degree of our confidence about a context that takes place in reality. In other words, it characterizes the context's ability of reflecting the reality. To improve confidence level, an effective approach is data fusion; it can integrate data from

multiple sources for obtaining a better context detection result. Moreover, data fusion can be based on single type of sensors, or multiple types of sensors. The latter one is also called multi-modal sensing. In [THM05], the authors design a system with camera arrays to understand the events and human activities in the smart spaces. Amir Padovitz *et al.* [P+05] propose a data fusion approach based on multi-attribute utility theory, which can integrate data from both physical and logical sensors, to improve the detection of user's situations. Nirmalya Roy *et al.* [RMJ+11] study how to dynamically adjust the selection of sensor modalities, i.e., choosing different sensors for fusion, to reduce energy consumption while satisfying the desired quality level for the context to be detected.

The second aspect of context uncertainty studied by many researchers is inconsistency. Context inconsistency occurs when different entities for detecting the same type of context obtain conflicting results. Chang Xu and S.C. Cheung [XC05] develop a context inconsistency detection and resolution approach using software engineering methodology. Inconsistent contexts are automatically captured based on formal semantic matching among the contexts. Another work on context inconsistency resolution is [BGT+06] where inconsistency can be easily detected by assertions of the ER-ontology. Then, the conflicted contexts with lower occurrence frequencies will be discarded to restore the consistent state. In contrast with the previous work relying on a central entity for context inconsistency detection, in [ZCHG13] the authors describe a decentralized approach.

Context ambiguity is a little similar to context inconsistency; to be precise, it means that one context has more than one interpretations. Resolving ambiguity is to determine an appropriate interpretation for the context from the multiple choices. In reality, automatic ambiguity resolution is very difficult due to the limited information we can acquire. So in

[DM05], the authors design a mediation process involving user's knowledge and judgement to help the system address ambiguity.

Chapter 3

Coordination of MCPS towards Online Hybrid Systems Model Checking

In this chapter, we will investigate the coordination problem with the purpose of enabling formal verification (e.g., model checking) for MCPS. We will propose a new approach to tackling the uncertainties regarding the patient's dynamics, and eventually develop an online method to carry out model checking. The online model checking procedure then runs as a real-time task to predict faults and prevent the resulting harms. Section 3.1 gives an overview about this work. In Section 3.2, we introduce some preliminary knowledge about model checking hybrid systems that MCPS belong to. Section 3.3 presents the drawbacks of existing offline hybrid systems model checking approach, as well as our online modeling approach by using a case study on Airway-laser Surgery. In Section 3.4, we will propose several system co-design patterns for enabling decidable and real-time online model checking. Our co-design approach is evaluated in Section 3.5. Afterwards, Section 3.6 points out several aspects deserving further discussion. We leave some theoretical analysis and results to Section 3.7 for easy reading, and finally we conclude this chapter in Section 3.8.

3.1 Overview

Medical Cyber-Physical System (MCPS) are complex systems having characteristics from both cyber and physical worlds. On the one hand, it involves cyber-world discrete computer logic of various embedded medical devices. On the other hand, it involves physical-world patient-in-the-loop, which is a continuous complex biochemical system.

The top concern of any MCPS is safety. In the cyber-world, for a safety-critical system, people often carry out model checking [BK08] *before* the system is put online. In such case, model checking builds an *offline* model of the system, and checks the system's possible behaviors in the *time-unbounded future* (i.e., *infinite-horizon*). Only after passing model checking may the system be allowed to run.

This practice is a great success. For CPS verification, the state-of-the-art model checking tools are the *hybrid systems* model checking tools [Tab09][AK03], which integrate the discrete automata models with the continuous differential equation (and other control theory) models. Today, hybrid systems model checking can already analyze many computerized control systems, i.e., control CPS.

The success of hybrid model checking in control CPS inspires the interest to apply it in MCPS. However, this faces a major challenge: in most MCPS applications, there are *no* good *offline* models to describe the complex biochemical system of the patient [LS10]. Even if some vital signs can be modeled offline, the models may not (with some exceptions [KSM+10]) fit into existing hybrid systems model checking tools, e.g., HyTech [HHWT97] and PHAVer [Fre05], which mainly use linear differential equations to describe the physical world entities.

To deal with the above challenges, we propose to alter the traditional practice of *offline* model checking of hybrid system's behavior in the *infinite-horizon*. Instead, we carry out periodical *online* model checking. In every period, we only model check the hybrid system's behavior in the next (few) period(s); i.e., we only model check the hybrid system's behavior in *time-bounded future* (i.e., *finite-horizon*). Based on this idea, MCPS devices then need to coordinate their activities appropriately to facilitate the online model checking.

The merits of the proposed approach are as follows. First, though many human body parameters are hard to model offline, their online behaviors in finite-horizon are quite predictable. For example, after injecting 1ml of morphine, it is hard to accurately predict the blood oxygen level curve in the next 40 minutes, as it depends on too many factors, even including the patient's emotion [Gra05][MBS07]. However, it is easy to predict the blood oxygen level curve in the next 4 seconds: it cannot plunge from 100% to 10%, nor show a saw-toothed wave form; instead, it has to be smooth, which can be effectively described with existing tools, such as linear regression. Also, within short finite-horizon, we can approximate many variables as constants, and/or approximate nonlinear behaviors as linear behaviors. This would further simplify our model and computation.

The proposed approach can be formalized as follows. Given an MCPS $\mathcal{S}$, we periodically sample the observable state parameters every T seconds. At time instance kT ($k = 0, 1, 2, \ldots$), MCPS devices coordinate to build a hybrid system model (i.e., the "online model") of $\mathcal{S}$ with the observed numerical values of state parameters, and verify its safety in the time interval $[kT, (k+1)T]$. We hence call T the *finite-horizon* of our online model checking. If the online model is proven safe, the system can run for another T seconds. Otherwise, the system immediately switches to an application dependant fall-back plan.

Each MCPS device thus follows the fall-back plan to operate in order to circumvent the potential risks.

Such model checking must finish within bounded and short time, i.e. *real-time*, to allow decision making (on whether to run the system for another T seconds or switch to fall-back plan) *before* any fault happens. To support real-time, the MCPS design must follow certain patterns, which brings up the issue of hybrid systems model checking and CPS *co-design*. We propose several special design patterns for MCPS so that online model checking of the MCPS is decidable. In other words, the task of online model checking procedure can terminate within a time bound. Therefore, such online model checking procedure can become a *hard* real-time task. However, our theoretical analysis has shown that such online model checking problem is NP-Hard. It implies that the time bound of online model checking execution may be too large to be practical. In this case, we can specify a reasonable deadline for the online model checking procedure, and make it a *soft* real-time task. Finally, we have developed algorithms for both the hard and soft real-time system designs, and evaluated the feasibility of our approach in a real-world medical case, i.e., Airway-laser MCPS.

3.2 Preliminaries

Hybrid systems model checking is first proposed by Alur, Henzinger, *et al.* [A+92] [AHH96] [HHWT97] and has since evolved into a family of state-of-the-art tools in CPS. The main idea is to combine the discrete automata models of computer logic with continuous differential equation models of control systems, which leads to the modeling tool of *hybrid automata*.

3.2.1 Syntax

Following [AHH96]'s conventions on symbols, a hybrid automaton A is syntactically a tuple of $A = (\vec{x}, \vec{x}^0, V, v^0, inv, dif, E, act, L, syn)$, where

- $\vec{x}$ is a vector of n *data variables* $\vec{x} = (x_1, x_2, \ldots, x_n)$. $\vec{x}$ is regarded as a function of time, and we use $\dot{\vec{x}}$ to denote the first order derivative of $\vec{x}$. We also use $\vec{x}' = (x'_1, x'_2, \ldots, x'_n)$ to denote the new values of $\vec{x}$ after an event (see the definitions for E and act). A specific evaluation of $\vec{x}$, denoted as $\vec{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{R}^n$ is called a *data state* of A. In addition, Boolean values of **true** and **false** can be denoted with real number 1 and 0 respectively; hence a data variable can also serve as a Boolean variable.

- $\vec{x}^0$ is the initial data state.

- V is a set of *locations*, a.k.a., *control locations*, where different control laws apply. Each location corresponds to a vertex in the graphical representation of hybrid automaton A. A *state* of hybrid automaton A is denoted as $(v, \vec{s})$, where $v \in V$ and $\vec{s} \in \mathbb{R}^n$ is a data state.

- v^0 is the initial location.

- inv is the *location invariants*, a function that assigns each location $v \in V$ a set of inequalities over data variables $\vec{x}$. That is, when in location v, the value of $\vec{x}$ must satisfy $inv(v)$.

- dif is the *continuous activities*, a function that assigns to each location $v \in V$ a set of inequalities over $\dot{\vec{x}}$ and $\vec{x}$. That is, when in location v, the values of $\dot{\vec{x}}$ and $\vec{x}$ must

satisfy $dif(v)$.

- E is the set of *events*, a.k.a. *transitions*: edges between locations. Formally, $E \subseteq V \times V$. For an event $e = (v, v') \in E$, v is the *source location* and v' is the *target location*.

- act is the *discrete actions*, a function assigns to each event $e = (v, v') \in E$ a set of inequalities over $\vec{x}$ and $\vec{x}'$, where $\vec{x}' = (x'_1, x'_2, \ldots, x'_n)$ refers to the new value of $\vec{x}$ after event e. The event $e = (v, v')$ is enabled only when the value of $\vec{x}$ in v satisfies $act(e)$, and the new value of $\vec{x}'$ after the event is chosen nondeterministically such that $act(e)$ is satisfied. For example, suppose $\vec{x} = (x_1)$, then for $act(e) = (x_1 \leq 3 \wedge x'_1 \leq 5 \wedge x'_1 \geq 5)$, event e is only enabled when $x_1 \leq 3$; and after the event, x_1 is assigned the new value of 5. Like this example, if $\vec{x}$ and $\vec{x}'$ do not mix in any inequalities in $act(e)$, and $\vec{x}'$ has a deterministic value $\vec{s}'$, then we can call the subset of inequalities involving only $\vec{x}$ to be the *guard* of event e, and event e *updates* $\vec{x}$ to $\vec{s}'$, denoted as $\vec{x} := \vec{s}'$.

- L is a set of *synchronization labels*.

- syn is the *synchronization function* that assigns each event $e \in E$ an $l \in L$. L and syn are for composition of multiple hybrid automata. Suppose we have two hybrid automata $A_1 = (\vec{x}_1, \vec{x}_1^0, V_1, v_1^0, inv_1, dif_1, E_1, act_1, L_1, syn_1)$ and $A_2 = (\vec{x}_2, \vec{x}_2^0, V_2, v_2^0, inv_2, dif_2, E_2, act_2, L_2, syn_2)$, if $e_1 \in E_1$, $e_2 \in E_2$ and $syn_1(e_1) = syn_2(e_2)$, then event e_1 and e_2 must always take place together.

Furthermore, when inv, dif, and act only involve linear inequalities, and dif does not involve $\vec{x}$, hybrid automaton A is called *linear hybrid automaton* (LHA) [A+92]. Reference

[AHH96] also describes how to combine several hybrid automata into one hybrid automaton. Particularly, the location set of the combined hybrid automaton $V_{comb} = V_1 \times V_2 \times \ldots \times V_n$, where V_i $(i = 1, \ldots, n)$ is the location set of the ith component hybrid automaton; and "$\times$" is Cartesian product. For $v \in V_{comb}$, we use $v|_i$ to denote the projection of v on V_i.

3.2.2 Semantics

This paper adopts the semantic concepts and the corresponding symbol definitions of [AHH96]. Due to page limit, interested readers shall refer to [AHH96] for these definitions. Of particular importance are the concepts of *state predicate*, *trajectory*, *number of hops (of a trajectory)*, *non-blocking*, *non-zeno*.

We, however, want to emphasize that to simplify narration, in the following, unless explicitly denoted, "model checking" refers to "*model checking of finite-horizon reachability semantics*", i.e., whether a state σ of hybrid automaton A satisfies $\varphi_1 \exists \mathcal{U}_{\leq T} \varphi_2$, where φ_1 and φ_2 are state predicates of A, and T is the *finite-horizon*. Also, unless explicitly denoted, *we only discuss non-blocking hybrid automata*.

3.3 Hybrid Systems Modeling

In this section, we shall use Airway-laser Surgery, a representative MCPS application [KSM+10][AST09], as the context to discuss the proper hybrid systems modeling approach for MCPS. We shall see through this case study why offline model checking must be replaced by online model checking.

Airway-laser MCPS interlocks various medical devices to increase safety. It has the following entities (see Fig. 3.1):

- **Patient:** the patient that receives the surgery;

- O_2 **Sensor:** the patient's trachea oxygen level sensor;
- SpO_2 **Sensor:** the patient's blood oxygen level sensor;
- **Ventilator:** the medical device that administrates the patient's respirations;
- **Surgeon:** the doctor that conducts the surgery;
- **Laser Scalpel:** the medical device for the surgeon to cut the patient's trachea;
- **Supervisor:** the central computer that connects all medical devices and makes decisions to guarantee safety.

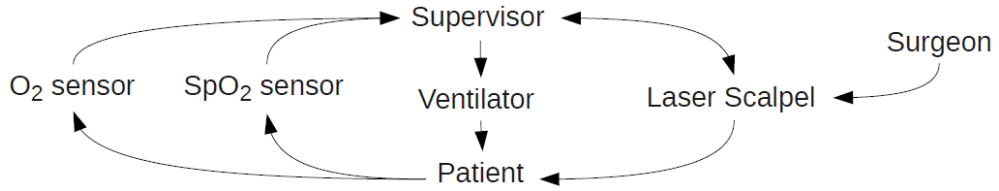


Fig. 3.1: Layout of Airway-laser MCPS

The application context is described briefly as follows (detailed description can be found in Section 2.1.3). In the surgery, due to general anesthesia, the patient is paralyzed, hence has to depend on the ventilator to breathe. The ventilator has three modes: pumping out (the patient inhales oxygen), pumping in (the patient exhales), and hold (the patient exhales naturally due to chest weight). However, when the laser scalpel is to cut the patient's trachea, the oxygen level inside the trachea must be lower than a threshold. Otherwise, the laser may trigger fire. Therefore, before the laser scalpel is allowed to emit laser, the ventilator must have stopped pumping out (oxygen) for a while. On the other hand, the ventilator can neither stop pumping out for too long, or the patient will suffocate due to

too low blood oxygen level. In summary, the Airway-laser MCPS must avoid the following safety hazards:

- **Safety Hazard 1:** when the laser scalpel emits laser, the patient's trachea oxygen level exceeds a threshold Θ_{O_2};
- **Safety Hazard 2:** the patient's blood oxygen level reaches below a threshold Θ_{SpO_2}.

Note that the setting of constant thresholds Θ_{O_2} and Θ_{SpO_2} are medical experts' responsibility and are beyond the coverage of this paper. The formal expressions of safety hazards will become clear by the end of Section 3.3.2, when the corresponding hybrid automata are defined.

3.3.1 Traditional Approach: Offline Modeling

Because the Airway-laser MCPS involves both discrete medical device logic and physical world patient, it is a hybrid system. Therefore we try to model Airway-laser MCPS with hybrid automata.

The traditional approach of model checking, including hybrid systems model checking, is carried out offline. That is, the model is built and its infinite-horizon behavior is verified *before* the system runs. We choose to start with this approach. As a common practice, our offline modeling of Airway-laser MCPS assumes a global time t: t is initialized to 0 second, and $\dot{t} \equiv 1$.

Intuitively, we intend to start with modeling the patient, the core entity of the Airway-laser MCPS. However, the patient's behavior is directly administrated by the ventilator, which has to be understood first.

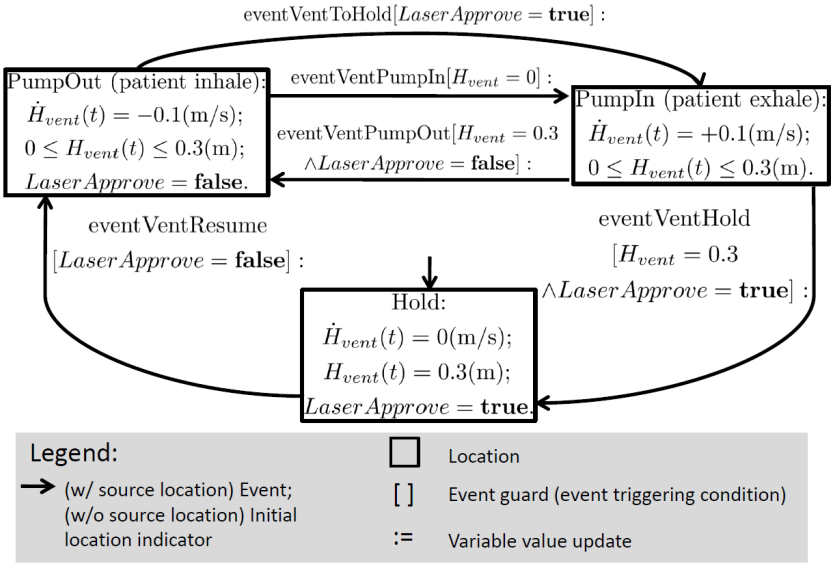


Fig. 3.2: Offline hybrid automaton of Ventilator

The ventilator is basically a compressible air reservoir [DD07]: a cylinder of height $H_{vent}(t)$ $(0 \leq H_{vent}(t) \leq 0.3(\text{m}))$. The movement of the ventilator cylinder (indicated by $\dot{H}_{vent}(t)$) pumps out/in oxygen/air to/from patient, thus helping the patient to inhale/exhale. The ventilator behavior is defined by the hybrid automaton in Fig. 3.2. The automaton has three locations: PumpOut, PumpIn, and Hold. When the supervisor (will be discussed later in Fig. 3.8) allows the ventilator to work (i.e., when data variable *LaserApprove* is set to **false**), the ventilator switches between pumping out (where $\dot{H}_{vent} = -0.1\text{m/s}$) and pumping in (where $\dot{H}_{vent} = +0.1\text{m/s}$). This causes the patient to inhale oxygen and exhale respectively. When the supervisor pauses the ventilator (i.e., when *LaserApprove* is set to **true**), the ventilator cylinder will try to restore to its maximum height (0.3m) and holds there until the ventilator is allowed again (*LaserApprove* set to **false**).

With the ventilator hybrid automaton at hand, we can now start modeling the patient.

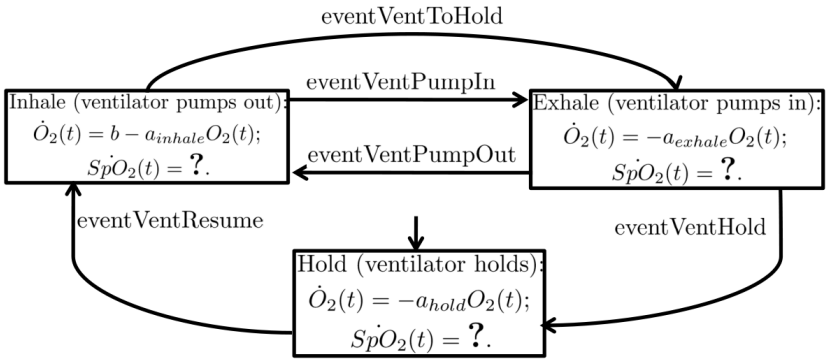


Fig. 3.3: Offline hybrid automaton of Patient. Though good offline models for $\dot{O}_2$ exists [KSM+10], the offline model for $\dot{SpO}_2$ is still an open problem. Also note that in location Hold (which corresponds to ventilator Hold), the patient still exhale due to chest weight.

The patient hybrid automaton (see Fig. 3.3) is tightly coupled with the ventilator hybrid automaton (see Fig. 3.2). It also has three locations: Inhale, Exhale, and Hold, which respectively correspond to the ventilator hybrid automaton's locations of PumpOut, PumpIn, and Hold. The events between the three locations are also triggered by corresponding events from the ventilator hybrid automaton.

Inside of each location are the offline continuous time models for trachea oxygen level $O_2(t)$ and blood oxygen level $SpO_2(t)$. Unfortunately, though there are good offline models for $\dot{O}_2(t)$ [KSM+10], the offline model for $\dot{SpO}_2(t)$ is still an open problem [Gra05][MBS07]. This is because blood oxygen level are strongly affected by complex human body biochemical reactions, even emotions.

Therefore, we fail to model $SpO_2(t)$ offline, and hence fail to model the patient offline. What is worse, as the patient model is an indispensable component of the holistic offline model, *the offline model checking of Airway-laser MCPS fails.*

3.3.2 Proposed Approach: Online Modeling

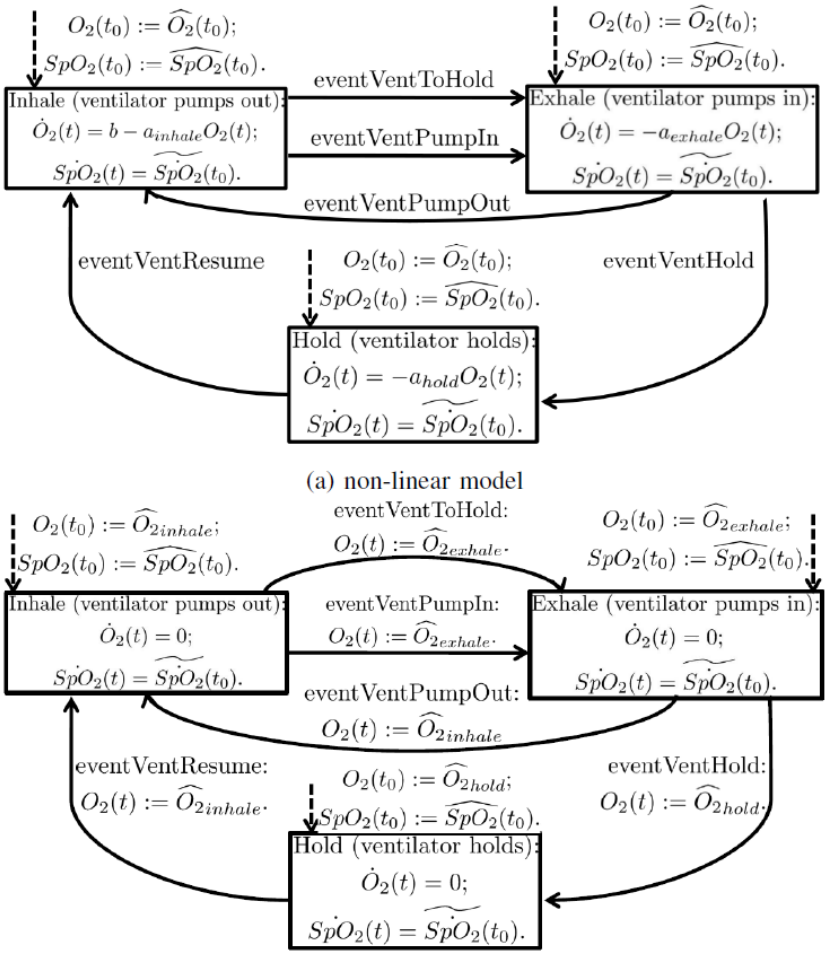
The failure of offline approach forces us to consider the proposed online approach (see Section 3.1) instead. Specifically, the supervisor asks for samples of the patient's trachea/blood oxygen level every T seconds. Suppose at $t_0 = kT$ ($k \in \mathbb{Z}_{\geq 0}$), the supervisor gets the most up-to-date trachea/blood oxygen level sensor reading $\widehat{O_2}(t_0)$ and $\widehat{SpO_2}(t_0)$. Then, it can build the hybrid systems model for interval $[t_0, t_0 + T]$, where T is therefore the *finite-horizon*. This model is built as follows.

First, same as the offline model checking, we use global variable t to represent the global clock, except that now t is initialized to t_0 and stops at $(t_0 + T)$ as we only care about the system's finite-horizon safety until $(t_0 + T)$.

The patient hybrid automaton now looks like Fig. 3.4(a). The biggest change is the continuous time model for the blood oxygen level $SpO_2(t)$. In offline model checking, we have to describe the infinite-horizon behavior of $SpO_2(t)$, which is an open problem. However, in online model checking, we only have to describe $SpO_2(t)$'s behavior in interval $[t_0, t_0 + T]$, where the finite-horizon T is just a few seconds. If we only look into such short-run future, blood oxygen level curve $SpO_2(t)$ is very describable and predictable. For example, it cannot plunge from 100% to 10% within just 4 seconds, neither can it show a saw-toothed wave form. Instead, it must be smooth; in fact smooth enough to be safely predicted with standard tools (such as linear regression) based on its past history.

In Fig. 3.4(a), we propose a simple way for the supervisor to predict/describe $SpO_2(t)$ in $t \in [t_0, t_0 + T]$:

$$\dot{SpO_2}(t) \equiv \widetilde{\dot{SpO_2}}(t_0), \qquad \forall t \in [t_0, t_0 + T],$$



(b) *linear hybrid automaton* (LHA) model (see Section 2.1 for definition of LHA), where $\widehat{O_{2inhale}}$, $\widehat{O_{2exhale}}$, and $\widehat{O_{2hold}}$ are constants, which can be estimated from historical data.

Fig. 3.4: Online hybrid automaton of Patient.

where $\dot{SpO_2}(t)$ is the derivative of $SpO_2(t)$ at time t; and $\widetilde{\dot{SpO_2}}(t_0)$ is the estimation (e.g., via linear regression) of $\dot{SpO_2}(t_0)$ based on $SpO_2(t)$'s history recorded during $(t_0 - T_{past}, t_0)$. T_{past} is a configuration constant picked empirically offline. In our case study, we pick $T_{past} = 6$ seconds.

Also, depending on the patient's state at time t_0, the initial location can be Inhale, Exhale, or Hold. Whichever location it is, the initial value of trachea/blood oxygen value

should be $\widehat{O_2}(t_0)$ and $\widehat{SpO}_2(t_0)$ respectively.

The patient model of Fig. 3.4(a) can be further simplified. Human subject respiration traces (see Fig. 3.5) show that the values of a_{inhale}, a_{exhale}, and a_{hold} in Fig. 3.4(a) are large: so large that $O_2(t)$ almost behaves as rectangular waves when the patient hybrid automaton changes locations. Therefore, we can simplify Fig. 3.4(a) into Fig. 3.4(b), where $O_2(t)$ remains constant within every location, and its value is only updated on the corresponding transitions. This simplification turns the patient hybrid automaton (in fact the whole system) into an *linear hybrid automaton* (LHA) (see Section 3.2 for definition of LHA), which is much easier to verify [Fre05].

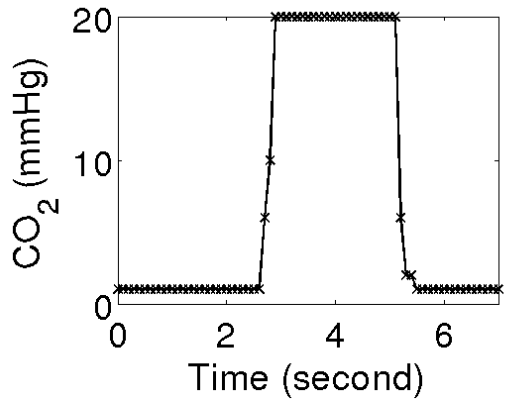


Fig. 3.5: A typical example excerpt of trachea CO_2 level trace (measured on human subjects with Nonin 9843 [Non]); note $O_2(t) = C_1 - C_2 \cdot CO_2(t)$, where C_1 and C_2 are two constants, whose derivation can be found in classic physics textbooks [M+03].

We now check other Airway-laser MCPS entities. First, since the online model only looks into the finite-horizon of $[t_0, t_0 + T]$, where T is also the sensor sampling period, there are no interactions with sensors throughout the interval of $(t_0, t_0 + T)$. Therefore, in online model checking, the hybrid automata of O_2 sensor and SpO_2 sensor are unnecessary.

Next, the ventilator hybrid automaton in online model (see Fig. 3.6) is almost the same

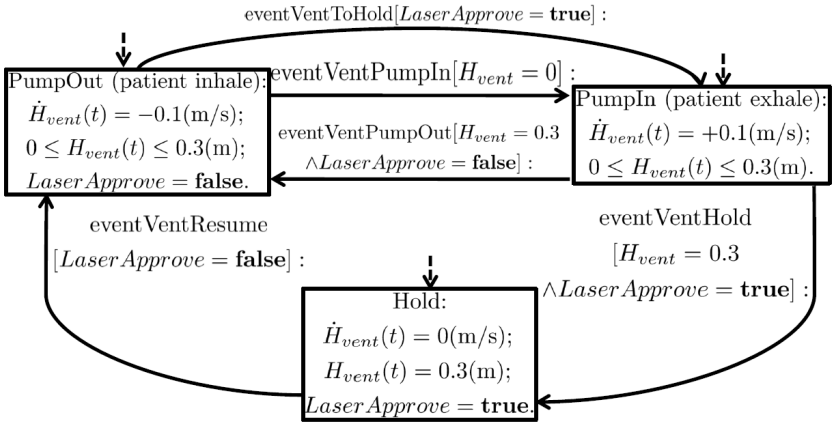


Fig. 3.6: Online hybrid automaton of Ventilator.

as its offline counterpart (see Fig. 3.2) A main difference is that the online model's initial location can be any location depending on the ventilator's state at t_0.

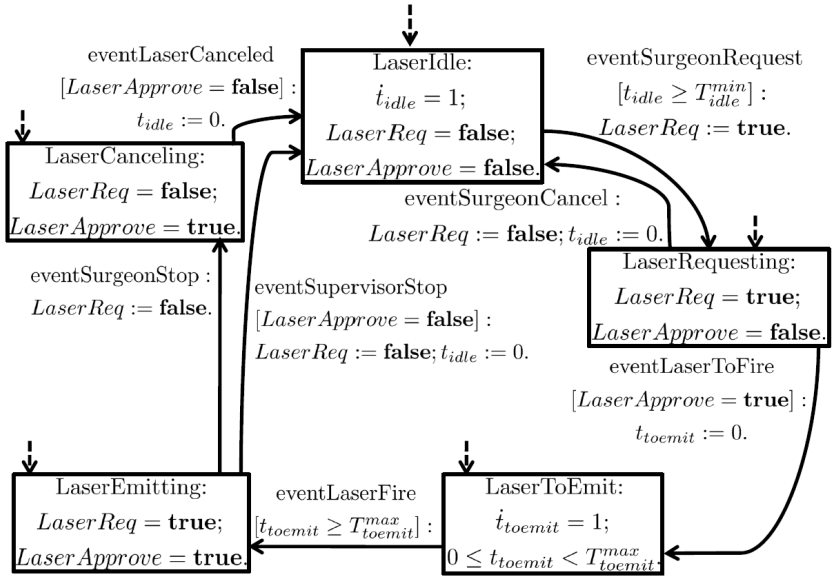


Fig. 3.7: Online hybrid automaton of Laser Scalpel. This is the only automaton that sets the value of state variable $LaserReq$.

The last entity that directly interacts with the patient is the laser scalpel. We can actually model the laser scalpel and the surgeon with one hybrid automaton: the laser scalpel hybrid automaton (see Fig. 3.7).

The automaton's key elements are the two Boolean variables: $LaserApprove$ and $LaserReq$. $LaserApprove$ indicates whether the supervisor (see Fig. 3.1) allows the laser scalpel to emit laser (**true** for yes and **false** for no). Its value can only be set by the supervisor hybrid automaton (see Fig. 3.8), which is to be explained later.

$LaserReq$ indicates whether the laser scalpel wants to emit laser (**true** for yes and **false** for no). Its value can only be set by the laser scalpel hybrid automaton. The value setting is triggered by following events: *i*) when in LaserIdle, the surgeon can request emitting laser through eventSurgeonRequest, which sets $LaserReq$ to **true**; *ii*) when in LaserRequesting or LaserEmitting, the surgeon can request stopping laser emission through eventSurgeonCancel and eventSurgeonStop respectively, which both set $LaserReq$ to **false**; *iii*) when in LaserEmitting, the supervisor can stop the laser emission at any time by setting $LaserApprove$ to **false**, which triggers eventSupervisorStop and sets $LaserReq$ to **false**.

The four possible combinations of $LaserApprove$ and $LaserReq$'s values define the major locations in the laser scalpel hybrid automaton: LaserIdle, LaserRequesting, LaserEmitting, and LaserCanceling. Particularly, laser scalpel emits laser in and only in LaserEmitting. There is an additional location, LaserToEmit, which models the additional delay T_{toemit}^{max} between LaserRequesting and LaserEmitting. This delay is to further ensure oxygen level in trachea falls below threshold before the actual laser emission.

The laser scalpel hybrid automaton's initial location can be anywhere depending on the laser scalpel's state at t_0. One thing to note is that all variables should be initialized to their actual value at t_0. For example, if initial location is LaserIdle, and Laser Scalpel has been idling for 10 seconds by t_0, then t_{idle} shall be initialized to 10 seconds instead of 0.

Finally, all medical device entities are interlocked by the supervisor, the central decision

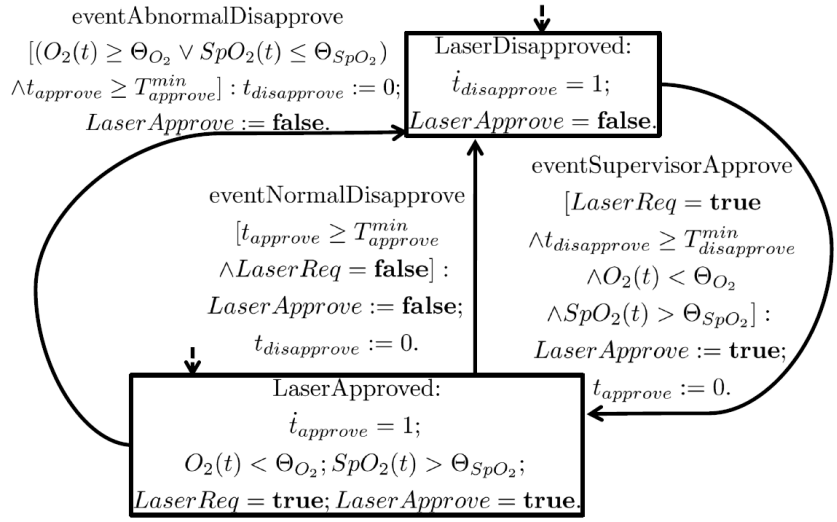


Fig. 3.8: Online hybrid automaton of Supervisor. This is the only automaton that sets the value of data variable *LaserApprove*. Note $t_{approve}$ can be totally removed from the model in *soft* real-time online model checking.

making computer (see Fig. 3.1). The supervisor maneuvers data variable *LaserApprove*. Setting *LaserApprove* to **true**/**false** determines the off/on of the ventilator and the permission/denial of emitting laser respectively.

The value setting decisions are made dependent on the most up-to-date information on the patient's trachea oxygen level $O_2(t)$ and blood oxygen level $SpO_2(t)$. Based on the models given in the patient hybrid automaton (see Fig. 3.4), we can predict $O_2(t)$ and $SpO_2(t)$ for any $t \in [t_0, t_0 + T]$. Therefore, we can construct the supervisor hybrid automaton as Fig. 3.8, which directly uses $O_2(t)$ and $SpO_2(t)$ predicted by the patient hybrid automaton for decision making.

The supervisor hybrid automaton has two locations: LaserDisapproved and LaserApproved. When in LaserDisapproved, the supervisor needs eventSupervisorApprove to move to LaserApproved. This event is triggered when the following prerequisites all hold:

- **Prerequisite 1:** the laser scalpel is requesting emitting laser (i.e., $LaserReq = \textbf{true}$);
- **Prerequisite 2:** $O_2(t)$ is less than threshold Θ_{O_2};
- **Prerequisite 3:** $SpO_2(t)$ is greater than threshold Θ_{SpO_2}.
- **Prerequisite 4:** $t_{disapprove} \geq T^{min}_{disapprove}$. This is a minimal dwelling time requirement to guarantee the automaton's non-zeno property. The purpose will become clear in a latter example (Example 1 of Section 3.7.2). This requirement also models the time cost in switching between LaserDisapproved and LaserApproved modes in the supervisor.

Through eventSupervisorApprove, the supervisor approves the emission of laser by setting $LaserApprove$ to **true**. This event also resets a clock $t_{approve}$, and moves the location to LaserApproved.

Like $t_{disapprove}$, clock $t_{approve}$ is for guaranteeing a minimal dwelling time of $T^{min}_{approve}$ in LaserApproved. After that, if Prerequisite 1 no longer holds (i.e., when $LaserReq$ becomes **false**), the eventNormalDisapprove is triggered. This event moves the supervisor back to location LaserDisapproved and resets $LaserApprove$ to **false**, and $t_{disapprove}$ to 0.

In contrast to eventNormalDisapprove, event$Abnormal$Disapprove is triggered when the supervisor is in LaserApproved while Prerequisite 2 or 3 stops to hold. This event also moves the supervisor back to location LaserDisapproved and resets $LaserApprove/t_{disapprove}$ to **false**/0 respectively.

Finally, same as the other online hybrid automata, the initial location for the online supervisor automaton can be either LaserDisapproved or LaserApproved, depending on the

state of the supervisor at time t_0; and the variables should be initialized to the actual values at t_0.

With the above hybrid automata model of the Airway-laser MCPS, we can formally express Safety Hazard 1 and 2 (see the beginning of Section 3.3) as follows.

- **Safety Hazard 1:** For any given initial state σ_0, $\sigma_0 \models \mathbf{true}\exists\mathcal{U}_{\leq T} \bigcup_{v\in V_{comb}\wedge v|_{ls}=\text{LaserEmitting}} (v, O_2(t) \geq \Theta_{O_2})$;

- **Safety Hazard 2:** For any given initial state σ_0, $\sigma_0 \models \mathbf{true}\exists\mathcal{U}_{\leq T} \bigcup_{v\in V_{comb}} (v, SpO_2(t) \leq \Theta_{SpO_2})$;

where V_{comb} is the location set of the combined automaton of the Ventilator, Patient, Laser Scalpel, and Supervisor; $v|_{ls}$ is v's projection on the Laser Scalpel automaton location set.

When model checking any one of the above safety hazards, a "yes" answer means the system is unsafe; while a "no" answer means this system is safe.

3.4 System Co-design Pattern

The evolution from offline model checking to online model checking must also be matched with system design changes.

3.4.1 Hard Real-time System Design

First, the overall system architecture shall integrate online model checking as a runtime fault prediction and prevention mechanism.

A straightforward thought is to coordinate the MCPS devices to run online model checking periodically. So far, we have assumed the period to be the same as the online model

checking's finite-horizon T. That is, at the beginning of each period T, online model checking predicts whether unsafe states are reachable within the coming T seconds. If so, the MCPS devices collaboratively switch the system to a fall-back plan for the current period. The fall-back plan is application dependent. For Airway-laser MCPS, a simple fall-back plan is that the supervisor locks *LaserApprove* at **false**, hence forbidding laser emission and keeping the ventilator active.

The above overall architecture works if online model checking costs 0 time. In practice, this is an over simplification. However, if the online model checking has a *worst case execution time bound* $D < T$ (where T is the online model checking's finite-horizon), then we can run the online model checking as a *hard real-time* task and use pipelining to carry out fault prediction and prevention. This is formally described by the algorithm in Fig. 3.9, which, without loss of generality, runs a pipeline with $T = 2D$; and D replaces T to be the new sampling period.

//This code assumes online model checking (see line 4, 5) can always
//finish within hard real-time deadline $D = \frac{T}{2}$.
1. main(){
2. wait till current time t satisfies $(t \bmod \frac{T}{2} = 0)$;
3. $t_0 := t$;
4. read sensors and build online model A;
5. **if** (A may reach unsafe states in $[t_0, t_0 + T]$){
6. /**non-blocking call:**/ switch the hybrid system to fall-back plan;
7. }**else**
/**non-blocking call:**/ allow the hybrid system to run normally;
8. **goto** line 2;
9. }

Fig. 3.9: Overall system architecture for *hard real-time* online model checking, with worst case execution time bound of D (for line 4, 5). Without loss of generality, the code runs a pipeline with $T = 2D$ (see line 2, 5). To "run normally" means that the hybrid system runs according to online model A's (see line 4) descriptions.

To run the hard real-time algorithm of Fig. 3.9, the online model checking problem must be *decidable*. That is, a time cost upper bound must exist. In the following, we show a large family of hybrid automata systems, *strongly non-zeno LHA systems* (SNZ-LHA-Systems) [RR96] to be exact, satisfy the decidability requirement.

Definition 1 (SNZ-LHA-System)**.** *Let $\mathcal{S}$ be a set of* linear hybrid automata *(LHA). For each LHA $A \in \mathcal{S}$, let $\mathcal{T}_A \stackrel{def}{=} \{\tau | \tau$ is a trajectory (see [AHH96] for the definition of "trajectory") of A and τ passes a transition of A twice$\}$. If $\exists \varepsilon > 0$, such that $\forall A \in \mathcal{S}, \inf_{\forall \tau \in \mathcal{T}_A}\{\delta_\tau\} \geq \varepsilon$ (where δ_τ is τ's duration; $\inf \varnothing \stackrel{def}{=} \infty$), then $\mathcal{S}$ is called a* strongly non-zeno LHA system *(SNZ-LHA-System).*

For an SNZ-LHA-System, we have the following:

Theorem 1 (Decidability)**.** *Finite-horizon reachability model checking of an SNZ-LHA-System is decidable.*

Detailed proof for Theorem 1 can be found in Section 3.7.1. What is more, the proof also shows a time cost upper bound for finite-horizon reachability model checking of an SNZ-LHA-system exists. In fact, interested readers can refer to Section 3.7.1 for a loose time cost upper bound, though a tight time cost upper bound is still an open problem.

Therefore, if we ensure an online hybrid systems model to be an SNZ-LHA-System, *real-time* worst case execution time (i.e., deadline) exists. Given a set $\mathcal{S}$ of LHAs, we claim in the following that $\mathcal{S}$ is ensured to be an SNZ-LHA-System if it complies with certain design patterns stated in Theorem 2.

Theorem 2 (Decidable Design Pattern)**.** *If every cycle of transitions in $\mathcal{S}$ complies with one of the following design patterns: ε-Minimal Dwelling Time, ε-Alternating Cyber-Value,*

or ε-Alternating Physical-Value, then finite-horizon reachability model checking on the LHA set $\mathcal{S}$ is decidable.

To describe these patterns, however, we need the following two concepts.

Definition 2 (Cycle of Transitions)**.** *Given a hybrid automaton $A = (\vec{x}, \vec{x}^0, V, v^0, inv, dif, E, act, L, syn)$, a* cycle of transitions *is a sequence of $e_0 v_0 e_1 v_1 \dots e_{k-1} v_{k-1} e_0$, where $e_i \in E$, $v_i \in V$, and $e_i = (v_{((i-1) \mod k)}, v_i)$ $(i = 0, 1, \dots, k-1)$.*

Definition 3 (Minimal Dwelling Time)**.** *Given a hybrid automaton $A = (\vec{x}, \vec{x}^0, V, v^0, inv, dif, E, act, L, syn)$, a location $v \in V$ has* minimal dwelling time *of ε iff for any trajectory τ that enters v via a transition, τ must stay in v for at least ε time before being able to leave v via a transition.*

With the above concepts, we can describe the following design patterns, assuming $\mathcal{S}$ denotes a set of LHAs.

Definition 4 (ε-Minimal Dwelling Time Pattern)**.** *Given a cycle of transitions $\mathcal{C}$, if there is at least one location v in $\mathcal{C}$, s.t. v has a minimal dwelling time of ε, then $\mathcal{C}$ complies with ε-Minimal Dwelling Time pattern.*

Example 1: The supervisor hybrid automaton (see Fig. 3.8) has a cycle of transitions $\mathcal{C}_1$ = "eventSupervisorApprove LaserApproved eventNormalDisapprove LaserDisapproved eventSupervisorApprove". Since all transitions entering location LaserDisapproved sets $t_{disapprove}$ to 0; while all transitions that leaves LaserDisapproved requires $t_{disapprove} \geq T^{min}_{disapprove}$. We assume $T^{min}_{disapprove}$ is a positive constant, then LaserDisapproved has minimal dwelling time of $\varepsilon = T^{min}_{disapprove} > 0$. As LaserDisapproved is in $\mathcal{C}_1$, $\mathcal{C}_1$ hence complies with ε-Minimal Dwelling Time pattern. ■

Definition 5 (ε-Alternating Cyber-Value Pattern)**.** *Given a set $\mathcal{S}$ of LHAs, and suppose LHA $A \in \mathcal{S}$ has a cycle of transitions $\mathcal{C} = e_0 v_0 e_1 v_1 \dots e_{k-1} v_{k-1} e_0$. If there are two transitions e_i, e_j $(i, j \in \{0, 1, \dots, k-1\})$ in $\mathcal{C}$, s.t.*

1. *$(i < j) \vee ((i \neq 0) \wedge (j = 0))$;*
2. *to trigger e_i, a state variable $x_l \in \vec{x}$ must* first *perform a discrete value switch from s to s' (possibly by another automaton in $\mathcal{S}$);*
3. *to trigger e_j, the same x_l must equal s;*
4. *x_l does not change value within any locations (i.e., it only changes during transitions).*
5. *$s \neq s'$, and all transitions in $\mathcal{S}$ that can switch x_l from s to s' enter target locations with a minimal dwelling time of ε.*

then $\mathcal{C}$ complies with ε-Alternating Cyber-Value pattern.

Example 2: In the ventilator hybrid automaton (see Fig. 3.6), there is a cycle of transitions $\mathcal{C}_2$ = "eventVentResume PumpOut eventVentToHold PumpIn eventVentHold Hold eventVentResume". Note that to trigger eventVentResume, *LaserApprove* must be first switched from **true** to **false**; and to trigger eventVentHold, *LaserApprove* must equal **true**. *LaserApprove* is a computer logic (i.e., cyber-) variable that does not change in any locations. Plus, all transitions that set *LaserApprove* from **true** to **false** enter the LaserDisapproved location (see Fig. 3.8), which has a minimal dwelling time of $\varepsilon = T^{min}_{disapprove}$, where $T^{min}_{disapprove}$ is a positive constant. This implies $\mathcal{C}_2$ complies with ε-Alternating Cyber-Value pattern. ■

Definition 6 (ε-Alternating Physical-Value Pattern)**.** *Given a cycle of transitions $\mathcal{C}$, if there are two transitions e_i, e_j in $\mathcal{C}$, s.t.*

1. *to trigger e_i, a state variable $x_l \in \vec{x}$ must equal s;*
2. *to trigger e_j, the same x_l must equal s';*
3. *$s \neq s'$, and x_l represent a physical world parameter, whose value can only change continuously, and there is an upper bound $R > 0$ on $|\dot{x}_l|$, s.t., $\frac{2|s-s'|}{R} \geq \varepsilon$.*

then $\mathcal{C}$ complies with ε-Alternating Cyber-Value pattern.

Example 3: In the ventilator hybrid automaton (see Fig. 3.6), there is a cycle of transitions $\mathcal{C}_3$ = "eventVentPumpOut PumpOut eventVentPumpIn PumpIn eventVentPumpOut". To trigger eventVentPumpOut, state variable H_{vent} must equal 0.3(m); while to trigger eventVentPumpIn, H_{vent} must equal 0(m). Meanwhile, as H_{vent} represents a physical world parameter: the current height of ventilator cynlinder. Its value can only change continuously, and the change rate is bounded by $|\dot{H}_{vent}| = 0.1$(m/sec). Therefore, to change from 0.3(m) to 0(m) and back to 0.3(m), it takes at least $2|0.3(\text{m}) - 0(\text{m})|/0.1(\text{m/sec}) = 6$(sec). Therefore $\mathcal{C}_3$ complies with ε-Alternating Physical-Value pattern, where we can pick $\varepsilon =$ 6(sec). ■

The detailed proof for Theorem 2 can be found in Section 3.7.2. If we review the Airway-laser MCPS online LHA model (see Fig. 3.4(b) $\sim$ 3.8), we find its design pattern complies with Theorem 2. Hence *online finite-horizon reachability hybrid systems model checking* (simplified as "*online model checking*" in the following, unless explicitly denoted) on Airway-laser MCPS is decidable. That is, theoretically, a worst case execution time bound for hard real-time exists.

3.4.2 Soft Real-time System Design

Though online *hard* real-time model checking of SNZ-LHA-Systems is theoretically possible due to Theorem 1, a tight bound on worst case execution time is still an open problem. A very loose bound is known (see Lemma 6 in Section 3.7.1), but it is often too large to be practical. In fact, we know the following:

Theorem 3. *Finite-horizon reachability model checking of an SNZ-LHA-System is NP-Hard.*

The detailed proof of Theorem 3 can be found in Section 3.7.3. This theorem implies online hard real-time model checking of SNZ-LHA-Systems is only practical for very small scale cases; *soft* real-time online model checking instead has more practical value.

In soft real-time online model checking, we directly specify a desired deadline D, without requiring *hard* real-time guarantee. The selection method of D is empirical: as long as D makes deadline misses satisfactorily rare and the online modeling satisfactorily accurate. For example, we can use standard benchmarks to find a desirable D (see Section 3.5.2).

Even though deadline D may be missed, soft real-time online model checking can still serve the MCPS hybrid system in at least two ways: one conservative and the other aggressive, as described by the pseudo code in Fig. 3.10.

In the conservative way, if online model checking misses deadline D, the MCPS hybrid system always switches to the (application dependent) fall-back plan. Assuming the modeling is accurate, the conservative way can prevent all accidents. However, if deadline misses are too often, the system will frequently switch to fall-back plan, annoying the users. In other words, the conservative way can raise a lots of false alarms, but can prevent all

```
//Online model checking deadline is D = T/2 (see line 4, 6, 7, 11, 12).
1. main(mode){
2.    wait till current time t satisfies (t mod T/2 = 0);
3.    t_0 := t;
4.    read sensors and build online model A;
5.    if (mode ="conservative way"){
6.        if ((A may reach unsafe states in [t_0, t_0 + T])
7.            or (current time t ≥ t_0 + T/2)){
8.            /*non-blocking call:*/ switch the hybrid system to fall-back plan;
9.        }else
              /*non-blocking call:*/ allow the hybrid system to run normally;
10.   } else {//mode ="aggressive way"
11.       if ((not (A may reach unsafe states in [t_0, t_0 + T]))
12.           or (current time t ≥ t_0 + T/2)){
13.           /*non-blocking call:*/ allow the hybrid system to run normally;
14.       }else
              /*non-blocking call:*/ switch the hybrid system to fall-back plan;
15.   }
16.   goto line 2;
17. }
```

Fig. 3.10: Revised overall system architecture that allows *soft real-time* online model checking. Without loss of generality, the code runs a pipeline with $T = 2D$ (see line 2, 6, 11), where $D = \frac{T}{2}$ is the real-time online model checking deadline. To "run normally" means that the hybrid system runs according to online model A's (see line 4) descriptions.

accidents.

Take our Airway-laser MCPS for example. Every time the online model checking misses the D seconds deadline on safety check, the supervisor will disapprove any laser emission request for the next D seconds (i.e., the "fall-back plan"). Instead, only when the online model checking confirms safety within the D seconds deadline will the supervisor follow Fig. 3.8's descriptions in the next D seconds.

In the aggressive way, if online model checking misses deadline D, the MCPS system does not switch to fall-back plan. The aggressive way only invokes fall-back plan when it is certain the system is facing risks. In other words, the aim of aggressive way is not to prevent all accidents, but to *reduce* accidents. In medical practice, a method that can

significantly reduce accidents is still a useful method; in fact, most medical routines are of such nature [B+09].

Again take our Airway-laser MCPS for example. Every time the online model checking misses the D seconds deadline on safety check, the supervisor will nevertheless follow Fig. 3.8's descriptions in the next D seconds. The fall-back plan (that the supervisor disapproves any laser emission requests) only kicks in when online model checking is certain that unsafe state is reachable within the D seconds deadline. Therefore, the online model checking is not to *eliminate* all possible accidents that a human surgeon may make, but to *reduce* such accidents as an additional protection.

To summarize, each deadline miss means the online model checking is uncertain about the safety of the MCPS hybrid system in the next D seconds. In the conservative way, the system always switches to the fall-back plan when the online model checking ends up uncertain (of course it also switches to the fall-back plan when the online model checking is certain of pending risks). In the aggressive way, the system only switches to fall-back plan when the online model checking is certain of pending risks.

3.5 Evaluations

To validate our proposed approach, especially the effectiveness (usefulness) of soft real-time online model checking for MCPS (the "conservative way" and the "aggressive way", see Section 3.4.2), we carry out evaluations using real-world trachea/blood oxygen level traces.

3.5.1 Effectiveness

We run soft real-time online model checking program $\mathcal{P}$ (see Fig. 3.10) upon emulated trachea/blood oxygen level sensors for 1200 seconds. We choose soft real-time deadline to be $D = 2$ seconds (see Section 3.5.2 for why). According to the soft real-time pseudo code of Fig. 3.10, this means every $D = \frac{T}{2} = 2$ seconds, $\mathcal{P}$ queries the emulated sensors for trachea/blood oxygen level readings, then builds and verifies an online model with finite-horizon of $T = 2D = 4$ seconds.

We have two sets of 1200-second traces for the emulated sensors. The first set of 1200-second traces comes from PhysioNet [G$^+$00], a comprehensive online public database (set up by NIH, NIBIB, and NIGMS) of real-world medical traces logged by hospitals. For simplicity, we call it "PhysioNet Traces". The other set of 1200-second traces comes from our own experiments on two human subjects. Human Subject 1 (HS1) mimics the combined behavior of the supervisor, laser scalpel, and surgeon in Airway-laser MCPS. As shown by Fig. 3.11(a), HS1 randomly swaps between holding the flag of "Laser Disapproved" and "Laser Approved". Human Subject 2 (HS2) mimics the combined behavior of the ventilator and the patient in the Airway-laser MCPS. When HS1 holds the "Laser Disapproved" flag, HS2 breathes smoothly at the rate of 6 seconds per respiration-cycle. When HS1 holds the "Laser Approved" flag, HS2 first tries to exhale (to his very best) and then holds his breath until HS1 raises the "Laser Disapproved" flag again (in case HS1 holds the "Laser Approved" flag for too long, HS2 is free to abort the experiment by resuming normal breath). Meanwhile, HS2's trachea and blood oxygen level are recorded by Nonin 9843 [Non]. We call the derived traces the "HKPolyU Traces".

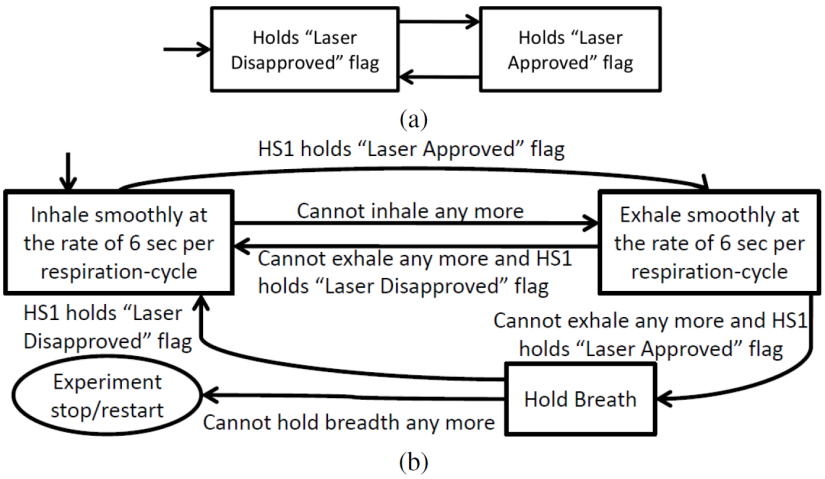


Fig. 3.11: Human subjects roles and behaviors. (a) HS1; (b) HS2.

The two emulated sensors read corresponding real-world traces (PhysioNet or HKPolyU) respectively. Based on the readings, $\mathcal{P}$ builds online hybrid systems models as described in Section 3.3.2, and verifies it. The specific modeling and verification software used is PHAVer [Fre05], a well-known hybrid systems model checking tool. Our computation platform is a Lenovo Thinkpad X201 with Intel Core i5 and 2.9G memory; the OS is 32-bit Ubuntu 10.10.

For each trace, throughout its 1200-second emulation period, program $\mathcal{P}$ carries out $1200/D = 1200/2 = 600$ trials of online modeling and verifications. The statistics of execution time cost is depicted by Table 3.1.

Table 3.1: Statistics of execution time cost of online model checking (unit: second; deadline $D = 2$ seconds)

	% of trials missed deadline	Execution time of those caught deadline (secs)			
		Min	Max	Mean	Std
PhysioNet Trace	2.2%	0.817	1.720	0.932	0.126
HKPolyU Trace	1.7%	0.818	1.940	0.965	0.146

The statistics show that more than 97.8% of the online model checking trials finished within the $D = 2$ (sec) deadline. In other words, only no more than 2.2% of the online model checking trials missed deadline.

Assume the modeling is accurate (which is going to be validated soon), in case $\mathcal{P}$ runs the "conservative way" (see Fig. 3.10), the above result means not only all accidents are prevented, the false alarm probability is no more than 2.2%. In case $\mathcal{P}$ runs the "aggressive way", the above result means more than 97.8% of accidents can be reduced (every time the system can reach unsafe states in the next D seconds, there is a $\geq 97.8\%$ chance that online model checking finishes within deadline, hence triggering the fall-back plan). Such reduction of accidents is significant according to the standards of medical practice [B$^+$09]. In either case, the results provide strong evidence that (soft) real-time online model checking is effective (i.e., feasible and useful).

To validate the assumption that the online modeling is accurate, we carry out statistics on the prediction error of blood oxygen level curve. During the online model checking, at every time instance $t_0 = kD$ ($k \in \{0, 1, \ldots, 599\}$, and $D = 2$ seconds), we sample the blood oxygen level and predict (see Fig. 3.4) the blood oxygen level curve in $[t_0, t_0+T]$ ($T = 2D = 4$ seconds). Let the predicted blood oxygen level at time $(t_0 + T)$ be $\widetilde{SpO}_2(t_0 + T)$. Let the PhysioNet/HKPolyU trace reading of blood oxygen level at time $(t_0 + T)$ be $\widehat{SpO}_2(t_0 + T)$. We define the relative prediction error at time $(t_0 + T)$ to be

$$ERR_{SpO_2}(t_0 + T) = \frac{|\widehat{SpO}_2(t_0 + T) - \widetilde{SpO}_2(t_0 + T)|}{\widehat{SpO}_2(t_0 + T)}.$$

The statistics of the relative prediction errors throughout the 600 trials for each trace are depicted by Table 3.2. The statistics show that our online model checking's predictions

on the finite-horizon behavior of blood oxygen level curve match the real-world traces quite accurately (with maximum relative error of 3.92%).

Table 3.2: Statistics of blood oxygen level online modeling relative errors (%)

	Min	Max	Mean	Std
PhysioNet Trace	0.03	2.53	0.51	0.52
HKPolyU Trace	< 0.01	3.92	0.61	0.60

3.5.2 Selection of Online Modeling Checking Deadline

Now we show why $D = 2$ seconds is an empirically desirable soft real-time online model checking deadline for the pseudo code of Fig. 3.10. We use both the 1200-second PhysioNet Trace and the 1200-second HKPolyU Trace as benchmark, and try out different values of D.

Table 3.3 shows the statistics on online modeling relative errors under different Ds. The statistics show that $D = 2$ seconds incurs least maximum relative error compared to other candidates. Note $D = 2$ seconds might not be the optimal choice, but based on the evaluations on the 2400-second medical traces, it turns out to be an empirically effective choice. A lot of parameters used in medicine are derived from such empirical studies.

Table 3.3: Online model checking relative error statistics under different Ds

Trace	D(sec)	Relative Error (%)			
		Min	Max	Mean	Std
PhysioNet	2	0.03	2.53	0.51	0.52
	3	0.04	4.52	0.76	0.74
	4	< 0.01	5.98	0.96	0.94
HKPolyU	2	< 0.01	3.92	0.61	0.60
	3	< 0.01	4.81	0.90	0.90
	4	< 0.01	6.29	1.18	1.12

3.6 Discussions

So far, our approach has been developed based on the assumption that inter-device communications are reliable and the online models are accurate (e.g., online predication based model of patient's state is accurate). In this section, we will relax those assumptions, and give in-depth discussions.

3.6.1 Discussions on Unreliable Communications

While building the online models, we have assumed reliable communications links between entities. Though this assumption is empirically valid for wired communications links, it is not for wireless.

Handling Unreliable Communications

How to adopt unreliable wireless communications links in life/safety critical medical settings is a nontrivial and active research area [W+11][H+10][W+07][B+08]. A comprehensive solution is beyond the scope of this paper. However, we can still provide a simple hybrid solution to allow wireless links *between the sensors and the supervisor*. Our solution is as follows.

According to the pseudo code of Fig. 3.10, every D seconds, the sensors are supposed to update the supervisor with the new readings of the patient's vital sign(s). Suppose at time instance iD ($i \in \mathbb{Z}_{\geq 0}$), the corresponding reading is X_i. Suppose at time instance iD, the supervisor needs to look at X_{i-k}, X_{i-k+1}, ..., X_i to build the online model. If any reading(s) of $X_{i-k} \sim X_i$ is(are) lost due to wireless communications failures, then for the period of $[iD, (i+1)D]$, the supervisor shall refuse to carry out online model checking, to cause a deliberate "deadline miss". This deliberately created deadline miss shall then be

treated as a usual deadline miss.

In this way, any wireless communications failures will only result in more deadline misses. The designs and analysis described in the previous sections (and subsections) still sustain.

Further Evaluations with Packet Loss

To evaluate the method proposed to handle communication failures, we redo the evaluations of Section 3.5 to redraw its Table 3.1 and 3.2. All other settings are the same except that we replace the link between the oximeter and supervisor with a wireless link.

The wireless link is unreliable, therefore packets carrying blood oxygen level readings may be lost. The packet losses are treated with the method proposed in the above. The results are summarized in Table 3.4 and 3.5, which respectively replace Table 3.1 and 3.2 of Section 3.5.

The results show that with packet loss rate of $\leq 3\%$, both the deadline miss rates and the relative errors are moderately low. Hence even with the unreliable wireless link, we can still carry out online hybrid systems model checking.

Note with the new advancements in medical grade wireless communications technology, it is possible to control wireless link packet loss rate to below 1%, or even 0.1% [H$^+$10][W$^+$07].

3.6.2 Discussions on False Negatives and False Positives

If the online model is absolutely accurate, the online model checking either misses deadline, or produces true-positive/true-negative conclusions. Interestingly, even if the online model is inaccurate, i.e., if the online model checking *can* produce false-positive/false-negative conclusions, our proposed method can still be useful for medical practices.

Table 3.4: Statistics of execution time cost of online model checking (unit: second; deadline $D = 2$ seconds)

	Packet loss rate	% of trials missed deadline	Execution time of those caught deadline (secs)			
			Min	Max	Mean	Std
PhysioNet Trace	1%	5.034%	0.817	1.720	0.932	0.124
	2%	6.387%	0.817	1.720	0.932	0.125
	3%	11.429%	0.817	1.720	0.929	0.124
HKPolyU Trace	1%	4.690%	0.818	1.940	0.964	0.147
	2%	6.030%	0.818	1.940	0.964	0.146
	3%	11.725%	0.818	1.940	0.966	0.149

Table 3.5: Statistics of blood oxygen level online modeling relative errors (%)

	Packet loss rate	Min	Max	Mean	Std
PhysioNet Trace	1%	< 0.01	2.529	0.512	0.526
	2%	< 0.01	2.529	0.510	0.523
	3%	< 0.01	2.529	0.503	0.526
HKPolyU Trace	1%	< 0.01	3.918	0.602	0.602
	2%	< 0.01	3.918	0.600	0.606
	3%	< 0.01	3.918	0.591	0.608

For ease of narration, we call our proposed online model checking based MDPnP practice as "*MDPnP-practice*"; call the corresponding online modeling and online model checking as "*MDPnP-online-modeling*" and "*MDPnP-online-model-checking*" respectively.

Still take the laser tracheotomy for example, Table 3.6 lists all possible cases for "MDPnP-practice". We see that the upper bounds of accident probability are

$$P_m^{cons} = p(+)p_m(-|+) \tag{3.1}$$

$$\text{and } P_m^{aggr} = p(+)\left[p_m(-|+) + p_m(?|+)\right] \tag{3.2}$$

respectively for "conservative mode" and "aggressive mode", where $p(+)$ is the probability that unsafe states are reachable if Fig. 3.4's online patient model is replaced with the absolutely accurate model (the model in "God's view"); $p_m(-|+)$, $p_m(?|+)$ are respectively the

conditional probability that MDPnP-online-model-checking gives "negative" answer (i.e., a false-negative answer), or misses deadline (i.e., cannot give a deterministic answer). Note false-positive is not a big concern as it will trigger fall-back plan, leaving no chances for accidents (though may be a nuisance to the surgeon).

Table 3.6: All possible cases for MDPnP practice

Reality	Online-Model-Checking Result	What Happens	Accident Possible?
positive	positive	scenario 1	No
	negative	scenario 2	Yes
	deadline	scenario 1 (cons)	No
	miss	scenario 2 (aggr)	Yes
negative	no need to care	no need to care	No

positive: unsafe states are reachable.
negative: unsafe states are not reachable.
scenario 1: fall-back plan kicks in, which forbids use of laser and keeps ventilator on; the worst case is that the surgeon may be annoyed.
scenario 2: the system run as what Fig. 3.8, 3.6, 3.4, and 3.7 describe.

In comparison to MDPnP-practice, now let us study the *current-practice* (i.e., the actual practice in nowadays hospitals) of Airway-laser Surgery.

First, the role of supervisor (i.e. the procedure described in Fig. 3.8) is taken over by a human-supervisor. Usually, the human-supervisor is the surgeon himself/herself; but for clarity, let us differentiate the two persons.

Second, as for line 4 of the algorithm described in Fig. 3.10, instead of MDPnP-online-modeling, the human-supervisor uses his/her subjective judgement to model the patient in the near future (e.g., replace $\widetilde{\dot{SpO}_2}(t_0)$ in Fig. 3.4 with his/her subjective prediction). We call this "*subjective-online-modeling*".

Third, as for line 6 and 11 of the algorithm described in Fig. 3.10, instead of MPnP-online-model-checking, the human-supervisor uses his subjective judgement to decide whether

unsafe states are reachable. We call this "*subjective-online-model-checking*".

Therefore, reusing the same analysis on the MDPnP-practice, we can derive the upper bounds of accident probability for the current-practice:

$$P_c^{cons} = p(+)p_s(-|+) \tag{3.3}$$

$$\text{and } P_c^{aggr} = p(+)\left[p_s(-|+) + p_s(?|+)\right] \tag{3.4}$$

respectively for "conservative mode" and "aggressive mode", where $p_s(-|+)$, $p_s(?|+)$ are respectively the conditional probability that subjective-online-model-checking gives "negative" answer (i.e., a false-negative answer), or misses deadline (i.e., cannot give a deterministic answer).

Suppose we adopt the "conservative mode". By comparing Equation (3.1) and (3.3), we see the MDPnP-practice is safer than the current-practice when

$$p_m(-|+) \leq p_s(-|+). \tag{3.5}$$

How to mathematically verify Inequality (3.5) is beyond the scope of this paper. However, we can still verify empirically. For example, if some well-established math model for predicting patient near-future behavior exists [MBS07], then we'd better use MDPnP-online-modeling rather than relying on subjective-online-modeling. Or, we can carry out comparison using well-known benchmark patient traces, to see which online-modeling is more trustworthy.

The same thing is for "aggressive mode", except that Inequality (3.5) now becomes

$$p_m(-|+) + p_m(?|+) \leq p_s(-|+) + p_s(?|+). \tag{3.6}$$

3.7 Theoretical Support for System Co-design Pattern

In this section, we provide the theoretical evidence for the theorems shown in the previous sections. We leave those theoretical analysis until this section for the sake of easy reading.

3.7.1 Decidability of SNZ-LHA System

This subsection gives a proof for the decidability of SNZ-LHA system. Though [RR96] gives a guideline for proof, to our best knowledge, a formal proof is still missing in the literature. We therefore give a formal proof in the following.

The proof involves heavy usage of symbols. Due to space limit, we are not going to explicitly re-define each symbol, instead, readers shall refer to [AHH96] for the definitions of the corresponding symbols.

We first define the following concepts:

Definition 7. *We say trajectory τ is from hybrid automaton state (v, σ) to a state space χ, denoted as $(v, \sigma) \rightsquigarrow_\tau \chi$, iff $\tau(0, 0) = (v, \sigma)$ and $\tau(h, \delta_h) \in \chi$, where $h = \|\tau\|$ is the number of hops of τ.*

For a finite-horizon model checking problem on whether $\sigma \models \varphi_1 \exists \mathcal{U}_{\leq T} \varphi_2$, we can use the well-known SMC-procedure proposed in [AHH96]. To prove the decidability of SNZ-LHA system, we only need to prove SMC-procedure has limited iterations for our case.

Let us first prove the following lemmas.

Lemma 4. *In SMC-procedure reachability model checking, if automaton state $(v, \sigma) \in \chi_i \backslash \chi_{i-1}$ (see Section 5.1 of [AHH96] for the definition of χ_i), then $\forall \tau$, if $(v, \sigma) \rightsquigarrow_\tau \chi_0$, then $\|\tau\| \geq \lfloor \frac{i}{2} \rfloor$.*

Proof: Suppose there is a trajectory τ, s.t. $(v,\sigma) \rightsquigarrow_\tau \chi_0$ and $h = \|\tau\| < \lfloor \frac{i}{2} \rfloor$.

Note τ can be denoted as $\tau = (v_0,\delta_0,\rho_0) \rightarrow (v_1,\delta_1,\rho_1) \rightarrow \ldots \rightarrow (v_h,\delta_h,\rho_h)$, which consists of h transitions between $(h{+}1)$ v-trajectories, and $(v_0,\rho_0(0)) = (v,\sigma)$ and $(v_h,\rho_h(\delta_h)) \in \chi_0$.

Since in each iteration of SMC-procedure, all predecessor regions within one v-trajectory or within one transition is included, it hence takes no more than $(2h+1)$ iterations for SMC-procedure to include state (v,σ). Meanwhile, $h < \lfloor \frac{i}{2} \rfloor \Rightarrow h \leq \lfloor \frac{i}{2} \rfloor - 1 \Rightarrow 2h+1 \leq 2(\lfloor \frac{i}{2} \rfloor - 1) + 1 \leq 2(\frac{i}{2} - 1) + 1 = i - 1$. This means $(v,\sigma) \in \chi_{i-1}$, which contradicts $(v,\sigma) \in \chi_i \backslash \chi_{i-1}$. □

Lemma 5. *Let A be an SNZ-LHA system. Then for any trajectory τ in A, the trajectory duration $\delta_\tau \geq \left\lfloor \frac{\|\tau\|}{\|E\|+1} \right\rfloor \varepsilon$, where $\|\tau\|$ is the hop length of τ, $\|E\|$ is the number of transitions of A, and ε is defined in the definition of SNZ-LHA system.*

Proof: Due to the well-known pigeonhole principle, for every sub-trajectory τ' of τ, if $\|\tau'\| \geq \|E\| + 1$, then τ' must have passed at least one cycle of transitions in A. Therefore, trajectory τ must have passed at least $\left\lfloor \frac{\|\tau\|}{\|E\|+1} \right\rfloor$ cycles of transitions in A without temporal overlapping. According to the theorem of SNZ-LHA system decidability, every cycle of transitions takes at least ε seconds to pass, the lemma hence holds. □

Lemma 6. *Suppose the LHA system only consists of one LHA A. Let $I = 2(\lceil \frac{T}{\varepsilon} \rceil + 1)(\|E\|+1)+1$, where T is the finite-horizon for finite-horizon reachability model checking, ε is defined in the definition of SNZ-LHA system, and $\|E\|$ is the number of transitions in A, then the SMC-procedure on model checking finite-horizon reachability terminates at the $(I+1)$th iteration.*

Proof: Suppose there is automaton state $(v,\sigma) \in \chi_{I+1} \backslash \chi_I$, then $\forall \tau.(v,\sigma) \rightsquigarrow_\tau \chi_0$, we have

$$\|\tau\| \geq \lfloor \frac{I+1}{2} \rfloor \qquad \text{(due to Lemma 4)},$$

then trajectory duration

$$\begin{aligned}
& \delta_\tau \geq \left\lfloor \frac{\|\tau\|}{\|E\|+1} \right\rfloor \varepsilon \qquad \text{(due to Lemma 5)} \\
\geq & \left\lfloor \frac{\lfloor \frac{I+1}{2} \rfloor}{\|E\|+1} \right\rfloor \varepsilon = \left\lfloor \frac{\lfloor \frac{2(\lceil \frac{T}{\varepsilon} \rceil + 1)(\|E\|+1)+2}{2} \rfloor}{\|E\|+1} \right\rfloor \varepsilon \\
= & \left\lfloor \frac{\lfloor (\lceil \frac{T}{\varepsilon} \rceil + 1)(\|E\|+1)+1 \rfloor}{\|E\|+1} \right\rfloor \varepsilon \\
\geq & \left\lfloor \frac{(\lceil \frac{T}{\varepsilon} \rceil + 1)(\|E\|+1)}{\|E\|+1} \right\rfloor \varepsilon = \left\lfloor \lceil \frac{T}{\varepsilon} \rceil + 1 \right\rfloor \varepsilon \\
\geq & \left\lfloor \frac{T}{\varepsilon} + 1 \right\rfloor \varepsilon > \frac{T}{\varepsilon} \varepsilon = T.
\end{aligned}$$

This violates the assumption that φ_1 and φ_2 already include the requirement that global time $t \in [0,T]$ for the case of finite-horizon reachability model checking. Hence $\chi_{I+1} \backslash \chi_I = \varnothing$, which means SMC-procedure terminates at the $(I+1)$th iteration. □

With the above definitions and lemmas, we can now prove the decidability of SNZ-LHA system (Theorem 1): Because the LHAs in the system can be combined into a single LHA within polynomial time [AHH96], and the resulted LHA A still satisfies the preconditions of SNZ-LHA system, hence the theorem holds due to Lemma 6.

3.7.2 Proof of Theorem 2

Proof of Theorem 2: If trajectory τ passes through a transition twice, then it passes through a cycle of transitions $\mathcal{C}$. Given $\mathcal{C}$ must comply with one of the following patterns:

Case 1: ε-Minimal Dwelling Time pattern, then τ must stayed in one location on $\mathcal{C}$ for more than ε.

Case 2: ε-Alternating Cyber-Value pattern, then τ must have changed a state variable x_l's value from s to s' and back to s, where x_l, s, and s' are described in Definition 5. According to Definition 5, this costs at least ε amount of time.

Case 3: ε-Alternating Physical-Value pattern, then τ must have changed a state variable x_l's value from s to s' and back to s, where x_l, s, and s' are described in Definition 6. According to Definition 6, since x_l is continuous and $|\dot{x}_l| \leq R$, altering x_l from s to s' and back to s takes at least $\frac{2|s-s'|}{R} \geq \varepsilon$ amount of time.

In summary, there must be $\delta_\tau \geq \varepsilon$. Therefore, $\mathcal{S}$ is SNZ-LHA-System, and hence is decidable for finite-horizon reachability model checking. ■

3.7.3 NP-Hardness of SNZ-LHA System Reachability Model Checking

We can reduce the well-known NP-Hard problem of Directed Hamiltonian Cycle to the problem of finite-horizon reachability model checking of an SNZ-LHA-System.

The Directed Hamiltonian Cycle problem is as follows: Given a directed graph $G = (V_G, E_G)$, where V_G and E_G are its vertex set and directed edge set respectively, does G contain a directed Hamiltonian cycle?

Given an instance of Directed Hamiltonian Cycle problem P_G on directed graph $G = (V_G, E_G)$, we can construct an SNZ-LHA-System $A = (\vec{x}, \vec{x}^0, V, v^0, inv, dif, E, act, L, syn)$ in polynomial time, where

$V = V_G$, i.e., each vertex in G is regarded as a location in A. Denote $n = |V| = |V_G|$, and hence denote $V = V_G = \{v_i\}$, where $i = 1, 2, \ldots, n$.

$E = E_G$, i.e., each edge in G is regarded as an edge in A.

$\vec{x} = (x_1, x_2, \ldots, x_n, t)$, i.e., we assign a variable x_i for each $v_i \in V = V_G$ (where $i =$

$1, 2, \ldots, n$); and t is a timer for fixed dwelling time guarantee, which will be explained later.

$\vec{x}^0 = (0, 0, \ldots, 0)$, i.e., x_i (where $i = 1, 2, \ldots, n$) and t are all initialized to 0.

v^0 can be any location. Without loss of generality, let us pick $v^0 = v_1$.

For each $v_i \in V$ (where $i = 1, 2, \ldots, n$), $inv(v_i)$ corresponds to the inequality proposition of

$$0 \leq t \leq 1. \tag{3.7}$$

For each $v_i \in V$ (where $i = 1, 2, \ldots, n$), $dif(v_i)$ corresponds to the following continuous activities:

$$\begin{aligned} \dot{x_i} &= 1, \\ \dot{x_k} &= 0, \text{ for each } k \neq i \text{ and } k \in \{1, 2, \ldots, n\} \\ \dot{t} &= 1. \end{aligned} \tag{3.8}$$

For each edge $e = (v_i, v_j) \in E$ (where $i, j \in \{1, 2, \ldots, n\}$), $act(e)$ corresponds to the following discrete actions:

$$t = 1, \tag{3.9}$$

$$t' = 0. \tag{3.10}$$

Formulae (3.9), (3.10), (3.7), and (3.8) together imply every location $v \in V$ has a *fixed* dwelling time of 1: once entered, one has to stay exactly 1 unit of time, and then move to another location. This further implies LHA A complies with the ϵ-Minimal Dwelling Time Pattern, where $\epsilon = 1$. According to theorem of decidable design pattern (Theorem 2), LHA A henceforth is a SNZ-LHA-System.

Finally, we choose $L = \varnothing$ and $syn = \varnothing$ as they are irrelevant to our proof.

With SNZ-LHA-System A at hand, our finite-horizon SNZ-LHA-System reachability model checking problem P_A checks whether the following state space S is reachable within finite-horizon $T = n + 1$:

$$\begin{aligned} S &= \{(v_1, \vec{x}) | \text{where } 1.5 < x_1 < 2, \\ &\qquad \text{and } 0.5 < x_i \leq 1 \text{ for each } i = 2, 3, \ldots, n\}. \end{aligned}$$

A "yes" answer to P_A implies there is a cyclic trajectory on A that traverse each vertex $v \in V$ exactly once and returns to the initial location of v_1. This trajectory hence corresponds to a Hamiltonian Cycle in G, hence a "yes" answer to P_G. Conversely, a "yes" answer to P_G implies there is a Hamiltonian Cycle in G, along this cycle, we can traverse A to reach S, hence a "yes" answer to P_A.

From above, we prove Directed Hamiltonian Cycle problem can be reduced to finite-horizon reachability model checking of an SNZ-LHA-System problem in polynomial time. As Directed Hamiltonian Cycle problem is NP-Hard, finite-horizon reachability model checking of an SNZ-LHA-System problem is hence NP-Hard. ■

3.8 Summary

Through our case study on Airway-laser MCPS, we show that online model checking of short-run future behavior can effectively address the challenge of lacking tractable patient model in MCPS hybrid systems model checking. By focusing on online and short-run future, many originally hard to describe/predict human body parameters become describable and predictable; and many variable parameters become fixed numerical values, which greatly simplifies verification. The MCPS devices cooperate to build the online hybrid system model and conduct model checking. Also, the online model checking can go real-time if

the proposed hard/soft real-time system co-design patterns are followed. Based on the patterns, MCPS devices can continuously coordinate to predict the future faults, and take appropriate actions to avoid the adverse consequences. Our empirical evaluations based on real-world human subject traces show that our online model checking and co-design approach is feasible and effective.

Chapter 4

Safety-assured Device Coordination in MCPS under Network Uncertainties

This chapter continues to study coordination problem in MCPS. However, our emphasis of this chapter is on dealing with network uncertainties, e.g., message loss and varied transmission delay. Specifically, we will investigate how to coordinate MCPS devices to support the safety-enhancement mechanisms documented in the standard (i.e., closed-loop control and safety interlock), even in the presence of network uncertainties. We start this chapter by giving the background and motivation of this work in Section 4.1. Section 4.2 presents the system model and device interaction diagram in our approach. We then develop the detailed algorithm to make sure the interaction mechanism preserves patient safety in Section 4.3. Evaluation of our solution is carried out in Section 4.4. Finally, conclusion is drawn in Section 4.5.

4.1 Background and Motivation

So far, most medical devices in the market have been produced for isolated use, and lack interoperability. To this end, Medical Device Plug-and-Play (MDPnP) Interoperability

Program [Med04] was launched in 2004 to promote medical device interoperability and safe device integration. The continuous effort of the program has led to the publication of the *Integrated Clinical Environment (ICE)* architecture standard [AST09] (see Section 2.1), which involves a controller, called *supervisor*, to coordinate medical devices for high-acuity patients. Following the ICE architecture, composition of the controller, network, along with the network-enabled medical devices yields a new type of system. In this dissertation, such systems are referred to as *Medical Cyber-Physical Systems (MCPS)* [LS10].

The goals of device coordination in MCPS include improving treatment safety and efficiency, as well as providing advanced decision support for caregivers. To enable these goals, one of the fundamental and effective coordination mechanisms is *closed-loop control*. In traditional control theory, closed-loop controller utilizes the measured output of a *process* as feedback to automatically adjust its following control signals to actuators, so as to obtain the desired system output. Closed-loop control is also utilized in MCPS to automatically maintain patient's physiologic variables within safe ranges, thus protecting patients from harmful human operations and reducing human intervention. Examples of such MCPS include closed-loop Patient-Controlled Analgesia system for safe opioid delivery [A$^+$10], and closed-loop insulin delivery system for maintaining healthy glucose level [Hov06].

Most of the existing research on closed-loop MCPS is based on reliable network condition, that is, messages exchanged between medical devices never get lost. However, such assumption may not be true. Especially recently, there has been an increasing interest in utilizing *wireless ad-hoc networks* to connect devices in MCPS. One one hand, more and more medical devices have been equipped with wireless communication capabilities (e.g., Bluetooth and WiFi), as clinical staff and patients prefer wire-free connection [KSYS10b]

which poses less restrictions on their movement. Since wireless devices have limited communication range, multi-hop ad-hoc network is sometime inevitable in order to connect distant devices. On the other hand, the clinical environment is quite dynamic. Patients can be transferred from one place to another; medical devices can be frequently turned on/off or moved around. Faced with such high dynamicity, ad-hoc network is more flexible for device connection compared with wired approach.

Despite the above-mentioned advantages, the side effect of using wireless ad-hoc networks is network uncertainties, e.g., message loss and varied message transmission delay between controller and other devices [BMJ+98]. This thus affects the controller design in the closed-loop MCPS. Suppose a "start" signal from the controller encounters a long delay to reach a stopped ventilator which supports patient's respiration, patient may suffer irreversible organ damage due to delayed oxygen supply. Therefore, it is necessary to design a more robust closed-loop controller that is capable of combating message loss and varied message transmission delay without affecting patient safety.

In traditional networked control systems, there exists a wealth of work dealing with message loss and delay, such as [LG04][XTLS07][A+11]. However, they can not be applied in MCPS. The reasons are two-fold. First, they rely on an accurate model (e.g., linear time-invariant model) about the process to be controlled. On the contrary, such accurate model about patient is often unavailable because of the complication of biochemical reactions happening in human body [L+12b]. Second, besides closed-loop control, there may exist additional safety and clinical requirements in MCPS which will affect the inter-device relationship and the controller design. For example, *safety interlock* mechanism proposed in the ICE standard [AST09] can be used in MCPS to forbid several devices entering certain

hazardous states simultaneously (some application examples of safety interlock can be found in Section 2.1). The interleaving of closed-loop control and safety interlock requirements thus requires us to jointly consider those mechanisms while designing the controller. What's more, traditional networked control systems do not involve user interactions; whereas MCPS coordination needs to meet the clinician's usage requirements.

In the area of MCPS, we have summarized the existing work on handling network uncertainties in Section 2.3.2. The work most closely related to ours is [KSM+10]. Although it succeeds in tolerating network failures, its drawback is that it is *time-triggered*, that is, coordination is only initiated periodically. As a consequence, surgeon's requests for using medical devices may not get response until the system's coordination activity starts. Obviously this long response delay will hinder its application in many delay-sensitive clinical contexts.

The mechanism we propose is *event-triggered*, which means each surgeon's request can initiate coordination activity immediately. In addition, to make our proposed mechanism capable of handing network uncertainties, we let the controller plan the future running modes for each device whenever the controller makes decision. This planning approach is based on prediction of the patient's worst-case future states. Even though an accurate patient model may not be available for prediction (this problem has been pointed out in Chapter 3), we commonly can estimate the short-term patient's state reliably [L+12b]. Based on such estimation knowledge, the controller is able to appropriately plan the future running modes for each device to satisfy the closed-loop control requirement. Furthermore, we ensure that the combination of devices' future running modes will not result in violation

of safety interlock requirements. In case control messages from the controller can not be received promptly, each device obeys the planned modes to operate. As all the planned modes are verified to be hazard-free, our mechanism avoids violating any closed-loop control and safety interlock requirements simultaneously. This planning-based approach works similarly to the Model Predictive Control (MPC) in the control theory [Cam04], which periodically determines the control decision for the future finite horizon based on the estimation of the process's future states. However, in MPC, only the first control decision is necessary to be sent to the actuators for real use, as the communication between controller and actuators are reliable. Moreover, it relies on an accurate model of the physical process, which is generally unavailable in MCPS.

To evaluate our approach, we construct a concrete MCPS, i.e., Airway-laser MCPS, in the simulation. Human subject trace-based simulation results confirm that patient safety can be guaranteed in any circumstances of network uncertainty. In addition, we test the performance of our approach by contrasting with an existing approach named NASS framework (proposed in [KSM$^+$10]). Results show that our approach achieves high clinical efficiency, and drastically outperforms NASS in terms of response time and communication cost (with maximum decease by a factor of 10 and 100, respectively).

4.2 System Overview

A MCPS consists of a supervisor and a collection of medical devices that are connected through a network. As mentioned in Section 2.1, medical devices can be categorized into two groups: *monitoring devices* and *delivery devices* [L$^+$12a]. The first group, such as heart rate and blood pressure sensors, monitors patient's physiological state or physical environment

parameters. The second group, such as infusion pump and laser scalpel, delivers therapy to patients and is capable of changing patient's state. The supervisor is installed with certain device control mechanism for a specific clinical procedure. In what follows, we will use *device* to refer to either a medical device or a supervisor, and use supervisor and controller interchangeably.

4.2.1 System Model

A MCPS Sys is modeled as a 4-tuple $\{\mathcal{S}, \mathcal{D}, \mathcal{H}, \mathcal{R}\}$. $\mathcal{D}$ is the set of medical devices. $\mathcal{S}$ is the supervisor that coordinates the medical devices. For each medical device $d \in \mathcal{D}$, $d = \{State^d, \xrightarrow{s}, Mode^d, \xrightarrow{m}, mode_0^d, Space^d\}$, where $State^d$ is the set of *device states* (states, for short); $\xrightarrow{s} \subset State^d \times State^d$ is a relation denoting the set of valid state transitions; $Mode^d$ is the set of supervisory modes (modes, for short) specified by the supervisor for d; $\xrightarrow{m} \subset Mode^d \times Mode^d$ is a relation denoting the set of valid mode transitions; $mode_0^d \in Mode^d$ is d's initial mode; and $Space^d$ is a function that defines the operational state space for each mode. When d is assigned with mode $mode^d \in Mode^d$ by the supervisor, it is restricted to operate within $Space^d(mode^d)$. Within $Space^d(mode^d)$, state transition is free to take place. It should be noted that mode transition implies state transition, that is, a medical device is forced to change its state if mode transition happens. Also note that for monitoring devices, their states are the sensor measurement results.

$\mathcal{H}$ is the set of hazards. A hazard $h \in \mathcal{H}$ is the combination of states of a medical device subset $\mathcal{D}^h$, where $\mathcal{D}^h \subseteq \mathcal{D}$, i.e., $h = \bigwedge_{d_i \in \mathcal{D}^h} state_h^{d_i}$. If all medical devices $d_i \in \mathcal{D}^h$ enter state $state_h^{d_i}$ simultaneously, then hazard h will be induced. Essentially a hazard h is a conjunctive predicate of the device states. Note that if multiple medical devices stay in

their initial modes $mode_0^d$, there must be no hazards yielded. Otherwise, it is nonsense that the initial system configuration is unsafe.

In essence, hazards are risky system states. The supervisor $\mathcal{S}$ runs a set of supervisory control rules (or control rules, for short), denoted by $\mathcal{R}$, to coordinate medical devices to circumvent the hazards. A control rule $r \in \mathcal{R}$ consists of a *condition field* and a *decision field*, i.e.,

$$\big(\bigwedge_{d_i \in \mathcal{D}^r} state_r^{d_i} \big) \wedge cmd^r \Longrightarrow mode_r^{d_r}. \tag{4.1}$$

The condition field defines the states for a medical device subset $\mathcal{D}^r$ as well as the input command cmd^r from caregivers. If the condition field is satisfied, the supervisor will execute the rule to designate a new mode $mode_r^{d_r}$ to device d_r.

The advantage of this system model is that it accommodates both closed-loop control and interlock requirements. On one hand, control loop can be closed by obtaining the feedback of monitoring devices' state update. Thus, supervisor can determine new modes for delivery devices based on the control rules, and then controls actuators' behavior. On the other hand, our definition of hazard is able to specify the violation of both closed-loop control requirement and interlock requirement. We say closed-loop control requirement is violated when patient's physiological variable under control falls beyond the desired safe range. For example, in airway-laser MCPS, $state^{\text{Oxim}} < 90\%$ is a hazard indicating the failure of closed-loop control.

System Model Example: The system model for Airway-laser MCPS is shown in Fig. 4.1. Detailed description of Airway-laser MCPS and how the surgery is carried out can be found in Section 2.1.3. Surgeon may trigger the usage of laser scalpel (Laser) from time to time. When laser is allowed to be used by supervisor, laser scalpel enters ALLOWED mode.

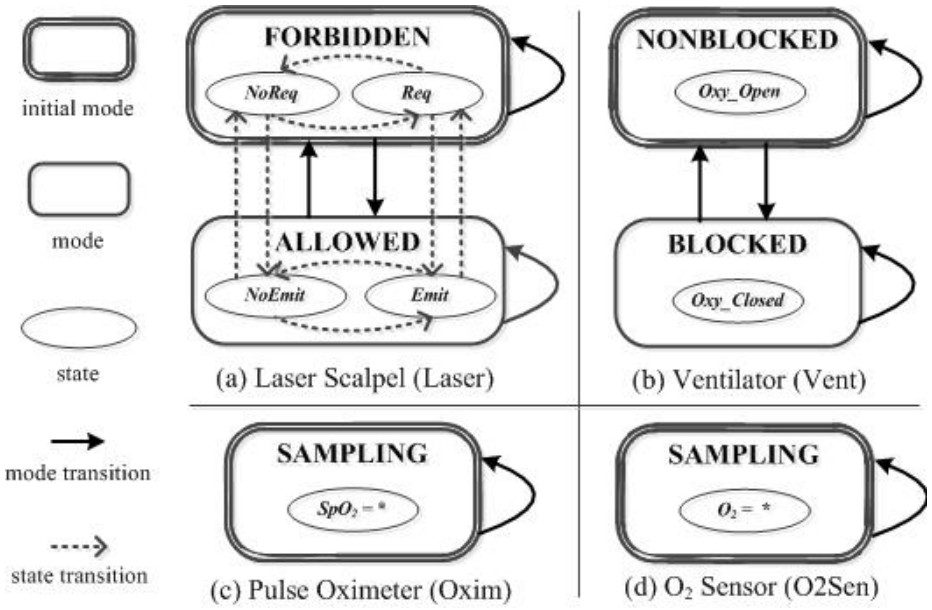


Fig. 4.1: System model for Airway-laser MCPS ($*$ represents any valid sampling result)

Otherwise, it stays in FORBIDDEN mode. For ventilator (Vent), supervisor specifies two modes: NONBLOCKED and BLOCKED. Initially ventilator operates in NONBLOCKED mode providing continuous oxygen supply for patient. But supervisor may force ventilator to suspend oxygen supply, i.e., switch to BLOCKED mode, when it decides to enable laser usage. Pulse oximeter (Oxim) and O_2 sensor (O2Sen) are monitoring devices; they always stay in SAMPLING mode, periodically providing SpO_2 and O_2 sampling results.

4.2.2 Device Interaction

The interactions between supervisor and medical devices are shown in Fig. 4.2. We define two types of messages for the interaction:

1. DOWNLINK: Each DOWNLINK message is sent by the supervisor, and contains a planned operation sequence for a delivery device associated with the supervisor's local time for generating the sequence.

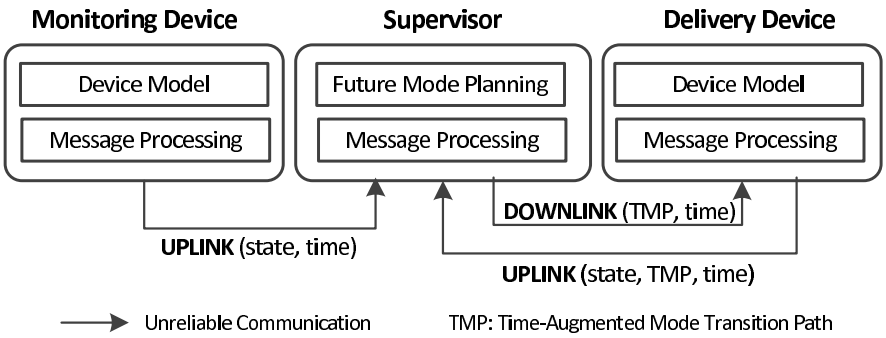


Fig. 4.2: Device interaction diagram

2. UPLINK: Each UPLINK message is constructed by medical devices to report their state change, along with the occurrence time about the change, to the supervisor. The state change is commonly triggered by user's operation (e.g., surgeon requests to use laser) or the generation of new sampling results in monitoring devices.

By using feedback from monitoring devices (encapsulated in UPLINK messages), the control loop in MCPS can be closed. Subsequently, supervisor can adjust its control decision by sending DOWNLINK messages to delivery devices. As wireless ad-hoc networks are not reliable, a mode update message may not be received by a delivery device. Missing critical modes would endanger patient's life. To deal with this problem, our basic idea is to let supervisor plan a sequence of safety-assured future modes for each delivery device. We call this mode sequence a *time-augmented mode transition path* (TMP). Therefore, in the case of missing DOWNLINK messages from the supervisor, each delivery device is still able to spontaneously follow the sequence to take mode transition, and meanwhile maintains safety property.

In addition to message loss and varied transmission delay, wireless ad-hoc network can also introduce *message out-of-order* problem, i.e., messages sent earlier may arrive at destination later [BMJ+98]. To deal with this problem, we require supervisor and each medical device to be *synchronized* in CPU clock (we will loosen this assumption in Section 4.3.4). Furthermore, we associate each message with a time stamp indicating when it is generated. An additional module, i.e., *Message Processing*, has been constructed in each device to handle the message out-of-order problem. Suppose two messages m_1 and m_2 are sent from the same source node, and the generating time of m_1 is earlier than m_2. Message Processing module in the destination inspects each message. Once it finds m_1's time stamp is smaller but arrives later than m_2, m_1 will be discarded. In fact, there exist multiple solutions to resolving the message out-of-order problem, e.g., using sequence number. The rationale behind our approach is that time synchronization will also benefit the calculation of TMP as will be introduced in the next section.

4.3 Algorithm Design

This section describes in detail how supervisor computes the future mode sequence, i.e., *time-augmented mode transition path* (TMP), for each delivery device.

4.3.1 Time-augmented Mode/State Transition Path

Now we give a definition for TMP. For a delivery device d, a mode $mode^d$ actually defines a state space in which device d is allowed to operate. A *time-augmented mode* $\langle mode^d, t\rangle$ associates a time t with mode $mode^d$; it restricts device d to be in $mode^d$ starting from time t. A mode transition path $mp^d = mode_0^d \rightarrow mode_1^d ... \rightarrow mode_l^d$ is a sequence of mode transitions, where each transition from $mode_i^d$ to $mode_{i+1}^d$ is valid. By associating each

mode in the path with a time, we obtain a *time-augmented mode transition path* (TMP) $tmp^d_{t_0}$.

$$\begin{aligned} tmp^d_{t_0} &= \langle mode^d_{t_0}, t_0\rangle \rightarrow \langle mode^d_{t_1}, t_1\rangle \rightarrow \ldots \rightarrow \\ &\quad \langle mode^d_{t_i}, t_i\rangle \rightarrow \ldots \rightarrow \langle mode^d_{t_l}, t_l\rangle, \end{aligned}$$

where $t_i < t_{i+1}$ for all $0 \leq i < l-1$, and l is the length of the path. A TMP specifies not only the order of mode transition for a device, but also when each transition should take place. We use $tmp^d_{t_0}[t]$ to indicate the specific mode in the path at local time t, where $t \geq t_0$. When delivery device d receives $tmp^d_{t_0}$ (contained in DOWNLINK message) from the supervisor at time t, it checks whether $t_0 \leq t < t_1$ hold or not. If yes, d will immediately switch to $mode^d_{t_0}$ and accept $tmp^d_{t_0}$ as its TMP. Otherwise, it implies that d misses certain mode transition deadline required by the supervisor, and thus the received $tmp^d_{t_0}$ is not accepted.

Now let us consider d is a monitoring device. Instead of its mode, we are more interested in its state transition, i.e., the variation of its sampling results, which reflects how the monitored parameter changes. A *time-augmented state transition path* (TSP) for d from time t_0, denoted by $tsp^d_{t_0}$, is a sequence of sampling result and sampling time pairs. More formally,

$$tsp^d_{t_0} = \langle state^d_{t_0}, t_0\rangle \rightarrow \ldots \langle state^d_{t_i}, t_i\rangle \ldots \rightarrow \langle state^d_{t_l}, t_l\rangle,$$

where t_i is the sampling time of result $state^d_{t_i}$. Similarly, $tsp^d_{t_0}[t]$ represents the d's state in the path at time t.

The following introduces three basic operations on TMP/TSP. The first operation is called *extraction*. If we want to know the specific mode (or state) in $tmp^d_{t_0}$ (or $tsp^d_{t_0}$) at a

specific time t, where $t \geq t_0$, we can extract it by $tmp^d_{t_0}[t]$ (or $tsp^d_{t_0}[t]$).

The second operation is called *truncation*, denoted by $trunc()$. A t-truncation of $tmp^d_{t_0}$ (or $tsp^d_{t_0}$), where $t \geq t_0$, is a sub-path of $tmp^d_{t_0}$ (or $tsp^d_{t_0}$) starting from t. Suppose $t_{i-1} \leq t < t_i$, we have

$$\begin{aligned} trunc(tmp^d_{t_0}, t) &= \langle mode^d_{t_{i-1}}, t\rangle \rightarrow \langle mode^d_{t_i}, t_i\rangle \rightarrow \dots \\ &\rightarrow \langle mode^d_{t_l}, t_l\rangle. \end{aligned} \tag{4.2}$$

The third one is called *appending*, denoted by $append()$, which adds a time-augmented mode (or state) at the end of a TMP (or TSP). For example,

$$\begin{aligned} tmp^d_{t_0}.append(\langle mode^d_{t_{l+1}}, t_{l+1}\rangle) = \langle mode^d_{t_0}, t_0\rangle \rightarrow \dots \\ \rightarrow \langle mode^d_{t_l}, t_l\rangle \rightarrow \langle mode^d_{t_{l+1}}, t_{l+1}\rangle. \end{aligned} \tag{4.3}$$

As a result, the length of the TMP/TSP will be incremented by 1.

4.3.2 TMP Calculation Algorithm

Because of the network uncertainty problem, a TMP sent to the delivery device may not be received and thus may not take effect. In this case, the supervisor does not know the exact TMP that is being used by the delivery device. However, it is able to *estimate* all the possible TMPs the delivery device is using, based on the past delivered TMP. Meanwhile, to reduce the estimation space, we require each delivery device to report their current TMP while they are sending UPLINK messages to the supervisor (see Fig. 4.2). For a delivery device $d_i \in \mathcal{D}$, the Future Mode Planning module in the supervisor maintains a set of TMPs, denoted as $\mathcal{TP}^{d_i}(t)$, which includes all the possible TMPs used by d_i at time t according to the supervisor's estimation.

Update Supervisor's Local Estimation

$\mathcal{TP}^{d_i}(t)$ is initialized to $\{\langle mode_0^{d_i}, t_e\rangle\}$ while the closed-loop control setup is accomplished at time t_e. Here, $mode_0^{d_i}$ is d_i's initial mode. Afterwards, there are two conditions when the supervisor should update $\mathcal{TP}^{d_i}(t)$. First, when the supervisor generates a new TMP $n\_tmp_t^{d_i}$ for d_i (see Alg. 1), then

$$\mathcal{TP}^{d_i}(t) = \mathcal{TP}^{d_i}(t) \cup \{n\_tmp_t^{d_i}\}. \tag{4.4}$$

Second, once the supervisor receives a UPLINK message from d_i, which includes d_i's most recently received TMP $tmp_{t_m}^{d_i}$, then $\mathcal{TP}^{d_i}(t)$ is updated to contain the t-truncation of all the TMPs generated no earlier than t_m, i.e.,

$$\mathcal{TP}^{d_i}(t) = \{trunc(n\_tmp_{t_k}^{d_i}, t) | t_m \leq t_k \leq t\}. \tag{4.5}$$

On the contrary, for a monitoring device $d_j \in \mathcal{D}$, the Future Mode Planning module maintains a worst-case estimation of d_j's future state, i.e., TSP $tsp_t^{d_j}$. Updating $tsp_t^{d_j}$ happens once a UPLINK message is received from d_j which contains d_j's up-to-date feedback $state_{t_s}^{d_j}$. At first, we use the worst-case prediction function to generate a TSP $tsp_{t_s}^{d_j}$ as the estimation of d_j's state starting from time t_s. Then, by removing the states from t_s to t, we can obtain $tsp_t^{d_j}$, i.e., $tsp_t^{d_j} = trunc(tsp_{t_s}^{d_j}, t)$. Note that the worst-case prediction function is dependent on the medical devices' states and the patient's characteristics. Actually, the assumption of having the worst-case state prediction is not unreasonable, because it is common practice that the surgeon should estimate patient's future state according to the past treatment and drugs delivered. However, our previous work found that long-term estimation of future state is generally impossible [L$^+$12b]. So, once beyond our estimation

range, we assign the *worst valid sampling value* of d_j, denoted as $WORST^{d_j}$, into $tsp_{t_s}^{d_j}$. For example, for pulse oximeter, $WORST^{\text{Oxim}} = 0\%$ and our estimation range for SpO_2 is next 6 seconds. Then, a TSP starting from 10 second could be $tsp_{10}^{\text{Oxim}} = \langle 98\%, 10\rangle \rightarrow \langle 95\%, 12\rangle \rightarrow \langle 93\%, 14\rangle \rightarrow \langle 0\%, 16\rangle$.

Algorithmic Details

Algorithm 1: Algorithm for calculating new TMP at time t

Input: $\mathcal{TP}^d(t)$ or tsp_t^d for all $d \in \mathcal{D}$
Output: A new TMP $n\_tmp_t^{d_r}$ for device d_r

1 $n\_tmp_t^{d_r} \leftarrow NULL$;
2 **foreach** $d_i \in \mathcal{D}\backslash\{d_r\}$ **do**
3 | Lock $\mathcal{TP}^{d_i}(t)$;
4 **end**
5 **foreach** d *is a delivery device in* $\mathcal{D}$ **do**
6 | $State_{t_h}^d = \bigcup_{tmp_t^d \in \mathcal{TP}^d(t)} \left(Space^d(tmp_t^d[t_h])\right)$;
7 **end**
8 **foreach** d *is a monitoring device in* $\mathcal{D}$ **do**
9 | $state_{t_h}^d = tsp_t^d[t_h]$;
10 **end**
11 **for** $t_h = t$ **to** $+\infty$ **do**
12 | **if** *there is a control rule* r *enabled at time* t_h *&& no hazards will be generated after executing* r **then**
13 | | $n\_tmp_t^{d_r} \leftarrow n\_tmp_t^{d_r}.append(\langle mode_r^{d_r}, t_h\rangle)$;
14 | **end**
15 **end**
16 **if** *Safety hazards can potentially happen during* $[t_{last}, +\infty)$ **then**
17 | $n\_tmp_t^{d_r} \leftarrow NULL$;
18 | Go to line 11 to examine the alternative rules that can be enabled;
19 **end**
20 **foreach** $d_i \in \mathcal{D}\backslash\{d_r\}$ **do**
21 | Unlock $\mathcal{TP}^{d_i}(t)$;
22 **end**

As the system execution progresses, the Future Mode Planning module in the supervisor will continuously update $\mathcal{TP}^{d_i}(t)$ and $tsp_t^{d_j}$ for each delivery device d_i and monitoring device d_j, respectively. Meanwhile, the module will check whether there is any control rule enabled.

If the execution of a rule $r \in \mathcal{R}$ will not yield hazards by considering all the combinations of r's resultant state space $Space(mode_r^{d_r})$ with other devices' possible states, the supervisor will execute the rule and continue to generate the future modes for d_r, i.e., obtain a new TMP denoted by $n\_tmp_t^{d_r}$.

Alg. 1 shows the detailed steps for calculating $n\_tmp_t^{d_r}$ at time t. We use t_h to denote the future time instance. As t_h increases from t to the infinity, the algorithm generates all the planned modes for the new TMP. Procedure from line 2-4 is executed only once, which locks $\mathcal{TP}(t)$ for all the delivery devices except d_r. Hence, it only allows for executing control rules to update d_r's future modes. Line 5-7 estimate each delivery device's state space at time t_h, denoted by $State_{t_h}^d$; Line 8-10 estimate the worst-case state, denoted by $state_{t_h}^d$, for each monitoring device. Line 11-15 mean that once a new control rule is executed, the resultant mode is appended into $n\_tmp_t^{d_r}$, denoted by $mode_{t_h}^d$. The premise for executing the control rule is that no safety hazard can be caused when we examining all the possible combinations of device states, i.e., by combining d_r's mode $mode_r^{d_r}$ resulted from executing rule r with the estimated states of other devices. This process of calculating the new TMP is repeated until the length of $n\_tmp_t^{d_r}$ can not be increased any more.

Let us denote the last mode in $n\_tmp_t^{d_r}$ and the corresponding switching time by $mode_{last}^{d_r}$ and t_{last}, respectively. Line 16 in Alg. 1 poses an additional requirement for accepting $n\_tmp_t^{d_r}$. That is, from t_{last} on, the combination of d_r's state space $mode_{last}^{d_r}$ and the estimated states of other devices can not result in any hazards. If such requirement is not satisfied, $n\_tmp_t^{d_r}$ is potentially unsafe to be accepted by d_r. Therefore, it should be discarded. Meanwhile, if there exist multiple rules that can be enabled during the process of calculating $n\_tmp_t^{d_r}$, the supervisor has to run the whole algorithm again to check whether

the alternative rules can lead to a safe $n\_tmp_t^{d_r}$ or not. If no safe $n\_tmp_t^{d_r}$ can be found after the exhaustive search, the algorithm returns no new TMP. In other words, the existing TMP for d_r is not updated.

Finally, $\mathcal{TP}(t)$s will be unlocked for future calculation of new TMP (line 20-22). After the algorithm finishes, the generated $n\_tmp_t^{d_r}$ will be immediately sent to device d_r. Once receiving $n\_tmp_t^{d_r}$ (in the DOWNLINK message) at time t_r, device d_r should obey the new TMP to operate in the future. In other words, it updates its TMP to $trunc(n\_tmp_t^{d_r}, t_r)$. According to this TMP, corresponding modes will be extracted and delivered to the medical device for use at the specified time. Therefore, on the medical device's side, network uncertainties are transparent to the application layer.

4.3.3 Proof of the Algorithm's Correctness

For a monitoring device $d \in \mathcal{D}$, suppose there are two states $state^d$ and $state_*^d$, and $state^d$ is no worse than $state_*^d$, from clinical point of view. For example, $state^{\text{Oxim}} = [SpO_2 = 96\%]$ is no worse than $state_*^{\text{Oxim}} = [SpO_2 = 94\%]$ in the Airway-laser Surgery, as patient has higher blood oxygen level. Apparently, we have the following lemma.

Lemma 7. *If a control rule $r \in \mathcal{R}$ is safe to be executed when d is in $state_*^d$, then r is also safe to be executed in the case when d's state is $state^d$.*

Based on this lemma, we can then obtain the following theorem related to Alg. 1.

Theorem 8. *Alg. 1 makes sure that all the generated TMPs are safe to be used by delivery devices at any future time instances.*

Proof: Suppose current time is t_c when a delivery device d receives an DOWNLINK message from the supervisor, which contains TMP $tmp_{t_m}^d$. Apparently we have $t_m < t_c$ due to the

delay of message transmission. If $tmp^d_{t_m}$ is accepted by d, the implication is that none of the TMPs generated by the supervisor between t_m and t_c have been received by d. In the following, we will prove that it is safe for d to use $tmp^d_{t_m}$ at time t_c. Three cases are possible to happen during time period of $[t_m, t_c]$.

Case 1: The most recent time that the supervisor runs Alg. 1 is t_m. On the one hand, for all delivery device $d_i \in \mathcal{D}\backslash\{d\}$, any TMPs accepted by d_i during $[t_m, t_c]$ period must be contained in $\mathcal{TP}^{d_i}(t_m)$. At time t_m, Alg. 1 utilizes $\mathcal{TP}^{d_i}(t_m)$ to calculate new TMP for d and safety is ensured while executing control rules in the calculation. Therefore, the resultant TMP for d is safe for d to use, no matter which TMP is finally used by d_i at time t_c. On the other hand, for all monitoring device $d_j \in \mathcal{D}\backslash\{d\}$, at time t_m the supervisor uses worst-case prediction to generate d_j's TSP. The real sampling results of d_j during $[t_m, t_c]$ period must be no worse than the values in the TSP. Lemma 7 implies that the generated $tmp^d_{t_m}$ must be safe to d. In summary, regardless of the real TMP used by d_i and the real sampling results of d_j, $tmp^d_{t_m}$ generated by Alg. 1 is safe for d to use.

Case 2: After t_m, the supervisor runs Alg. 1 to update d's TMP again. Since the new TMP has not been received by d, there is no need to check the safety of new TMP.

Case 3: After t_m, there exists a delivery device $d_i \in \mathcal{D}\backslash\{d\}$ whose TMP is updated by the supervisor at time t_h. Here, $t_m < t_h \leq t$. Because of Equation 4.4 and 4.5, $\mathcal{TP}^d(t_h)$ always includes t_h-truncation of $tmp^d_{t_m}$, no matter whether the supervisor generates new TMPs for d or not during $(t_m, t]$ period. Let us denote by $n\_tmp^{d_i}_{t_h}$ the new TMP generated by Alg. 1 for d_i at t_h. While the supervisor calculates $n\_tmp^{d_i}_{t_h}$, it already considers $tmp^d_{t_m}$ to check the safety of control rule execution. Therefore, even though the new TMP $n\_tmp^{d_i}_{t_h}$ will be accepted by d_i during $(t_h, t_c]$, safety can still be ensured.

By summarizing the above cases, we conclude that safety is assured when delivery device d consumes the modes in $tmp^d_{t_m}$ for future use. Since d and t_c are arbitrary, we then proved Theorem 8. ■

4.3.4 Discussions on Clock Synchronization Errors

So far, the device interaction protocol (Fig. 4.2) and Alg. 1 have been developed based on the assumption that all the devices in MCPS are *perfectly* synchronized in CPU clock. Hence, when a medical device consumes a mode at the time specified in the TMP, no safety hazards will happen. In reality, however, perfect time synchronization is generally impossible, which is a fundamental fact in distributed systems [KS11]. In the following, we will discuss how to handle the problem of synchronization error.

Even though perfect clock synchronization assumption does not hold, we find that the state-of-the-art synchronization protocols have been able to achieve synchronization with very high precision. Meanwhile, there are solutions for both wired networks (e.g., [Mil91][KB03]) and wireless networks (e.g., [EGE02]). In other words, by introducing clock synchronization protocols into MCPS, we can consider there is an upper bound for the synchronization error between any pair of devices in the system. Let us denote this upper bound by Δ. Now we will modify Alg. 1 by taking this upper bound into account.

In line 12 of Alg. 1, a new control rule r is found to be enabled at time t_h. The new mode to be generated after executing rule r is $mode^{d_r}_r$. However, due to the synchronization errors in the real world, it is possible that rule r can be enabled during $[t_h - \Delta, t_h]$. Therefore, in order to avoid missing the potential rules, at every time instance t_h we need to check whether a rule can be yielded by combining d_r's state at time t_h with other devices' states

during $[t_h - \Delta, t_h + \Delta]$. In other words, we have to find the *earliest* possible time when a rule can be enabled. If such a rule r is found, then it is necessary to check whether hazards can happen or not by combining the resulted $mode_r^{d_r}$ (after executing the rule) with other medical devices' state space in time period $[t_h - \Delta, t_h + \Delta]$. If no hazards can be resulted, the execution of rule r is guaranteed to be safe, regardless of the discrepancy of the local clocks among different devices. Since the maximum time difference between the supervisor and any medical device is Δ, the state space within time period $[t_h - \Delta, t_h + \Delta]$ certainly contains the real operational state of each medical device in $\mathcal{D}\backslash\{d_r\}$. Therefore, after modifying Alg. 1 by the above approach, the generated new TMP will also be safe to be used by the delivery devices.

4.4 Evaluations

This section evaluates the safety property and performance of our approach. Because of the restrict regulatory constraints, we are not able to conduct real medical experiment. Thus, we choose to use simulation, in which we construct a well-known medical scenario, Airway-laser Surgery, and build an ad-hoc network to support Airway-laser MCPS. The system model for Airway-laser MCPS has been depicted in Fig. 4.1.

4.4.1 Trace Analysis

In the simulation, we collect real human traces to serve as the measurements for pulse oximeter and O_2 sensor. However, we confront one problem. That is, due to the limitation of existing devices and traces, we are not able to collect traces about airway O_2 concentration. Fortunately, CO_2 data can be found, and thus we use CO_2 data to replace O_2 measurements because of the negative correlation between O_2 and CO_2 measurements in

patient's trachea [L$^+$12b], i.e., high (low) O_2 concentration implies low (high) CO_2 concentration. To be specific, the trace we use is called *HKPolyU Trace*, retrieved from an emulated Airway-laser Surgery (refer to Section 3.5.1 for the details about how to obtain the trace). Note that the time span of the HKPolyU Trace is 1200 seconds, i.e., the Airway-laser Surgery lasts 1200 seconds. Because of the negative correlation between O_2 and CO_2 measurements, in the sequel, we will replace all the parameters regarding O_2 by CO_2 values. For example, the O_2 threshold ($\Theta_{O_2} = 30\%$) in the control rules is replaced by CO_2 threshold ($\Theta_{CO_2} = 15$ mmHg).

Recall that the Future Mode Planning module relies on the worst-case prediction to obtain the TSPs of the monitoring devices. A reasonable way of constructing the worst-case prediction functions for SpO_2 and CO_2 is to obtain the *worst-case variations in one sampling period*. Then we can work out the whole TSP by considering how many sampling periods have passed. For pulse oximeter, the lower SpO_2 implies worse patient's physiological condition. So we are interested in its *maximum decrease in one sampling period*. Similarly, we should figure out the *maximum decrease in one sampling period* for constructing CO_2's TSP. These results are obtained through statistic analysis of the traces (see Tab. 4.1). Note that the worst-case prediction function depends on the states of medical devices.

Table 4.1: Trace analysis results for the worst-case prediction

Parameter	**Unit**	**Sampling Period**	**Maximum Decrease in One Sampling Period**
SpO_2	%	2 sec	1 ($state^{\text{Vent}} = Oxy\_Open$)
			1 ($state^{\text{Vent}} = Oxy\_Closed$)
CO_2	mmHg	2 sec	29 ($state^{\text{Vent}} = Oxy\_Open$)
			10 ($state^{\text{Vent}} = Oxy\_Closed$)

4.4.2 Simulation Settings

Let us denote the routing paths for supervisor-laser scalpel, supervisor-ventilator, supervisor-pulse oximeter, and supervisor-O_2 sensor by $path^{s2l}$, $path^{s2v}$, $path^{s2p}$, and $path^{s2o}$, respectively (see Fig. 4.3). We create two types of network conditions to mimic the wireless ad-hoc networking effects. First, we vary the message loss probabilities of $path^{s2p}$ and $path^{s2o}$ (denoted by $P(path^{s2p})$ and $P(path^{s2o})$ respectively) from 0 to 0.9. Thus, the control loop in the system may become open because of missing feedback signals. Furthermore, we consider all the routing paths are symmetric. Second, we intentionally disconnect path $path^{s2l}$ or $path^{s2v}$, so that we can investigate the system's behavior in the case of network disconnection. Specifically, $path^{s2l}$ is disconnected from 500-510 second and 1000-1010 second; $path^{s2v}$ is disconnected from 800-810 second. Aside from the network setting, we also need to model the surgeon's behavior during the surgery. As real surgeon's behavior is very complex, in our experiment we choose a simple model, that is, surgeon decides whether to initiate a request in every D_i seconds, where D_i follows a uniform distributed in $[2, 5]$ seconds. The probability of initiating a request is 0.5 at the beginning of D_i, and the request will not be released until D_i ends. For each following test, we will run 20 times to cancel out the randomness generated by surgeon's behavior.

4.4.3 Results and Analysis

MCPS not only help to enhance patient safety, but also have to provide satisfiable quality of medical services (e.g., usability). So in the following, we will evaluate not only the system's safety property, as well as its service-related metrics, such as *clinical efficiency* and *average response time*. In the context of airway-laser surgery, clinical efficiency means

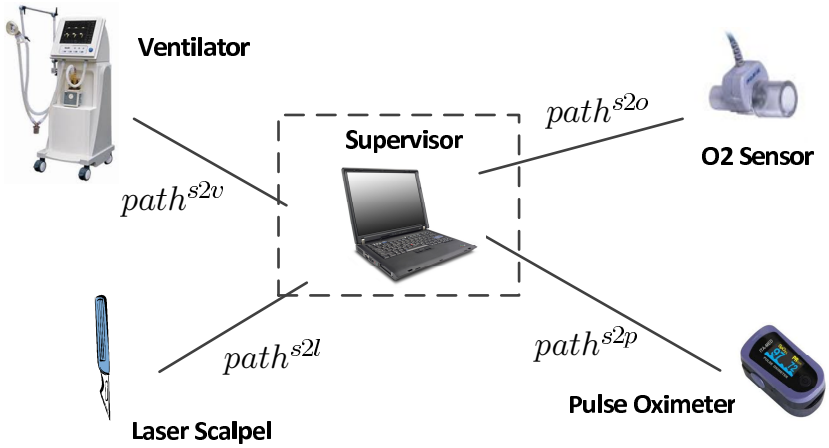


Fig. 4.3: Simulation setting

the percentage of time when the laser scalpel is in the ALLOWED mode, in which the surgeon can use the laser to cut patient's trachea. Average response time means the average delay from the time of surgeon's request to use laser to the time when the request is met. Meanwhile, we also evaluate the *communication cost* of our proposed approach, i.e., the total number of messages that have been sent, regardless of whether message reception is successful or not. In the simulation our approach (denoted by ModePlan) is compared with NASS [KSM+10], which is a state-of-the-art MCPS architecture with capability of tolerating network failures.

Safety Property

Along the 1200-second surgery, we collected the states of all the medical devices to check whether hazards happened or not. The results show that no hazards had happened in any circumstances, which validates the safety property of our approach. This also proves both the closed-loop control and interlock requirements can be satisfied. Due to space limit, we only show one system state diagram (Fig. 4.4) between 500-510 second (disconnection) in

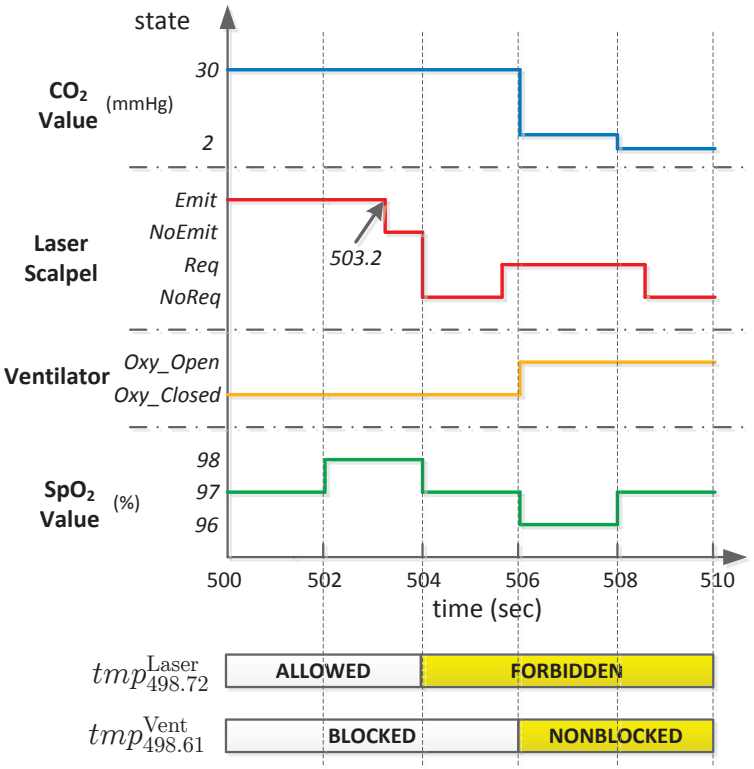


Fig. 4.4: System state diagram when $P(path^{s2p}) = 0$ and $P(path^{s2o}) = 0$.

the case when the packet loss probabilities of $path^{s2p}$ and $path^{s2o}$ are 1. We can find from the figure that, no hazards happen. Furthermore, even though laser scalpel is disconnected from the supervisor at 500 second, it is still able to allow surgeon to use laser, i.e., $state^{\text{Laser}} =$ *Emit* from 502-503.2 second. This is because laser scalpel receives from the supervisor a TMP $tmp^{\text{Laser}}_{498.72}$ at 498.72 second, which allows the laser scalpel to continue operating in the ALLOWED mode till 504 second.

Clinical Efficiency

Fig. 4.5 shows the clinical efficiency diagram in different settings. As we can see, the clinical efficiency decreases as the message loss probability becomes worse. Because the supervisor conservatively uses the worst-case prediction to estimate future SpO_2 and CO_2

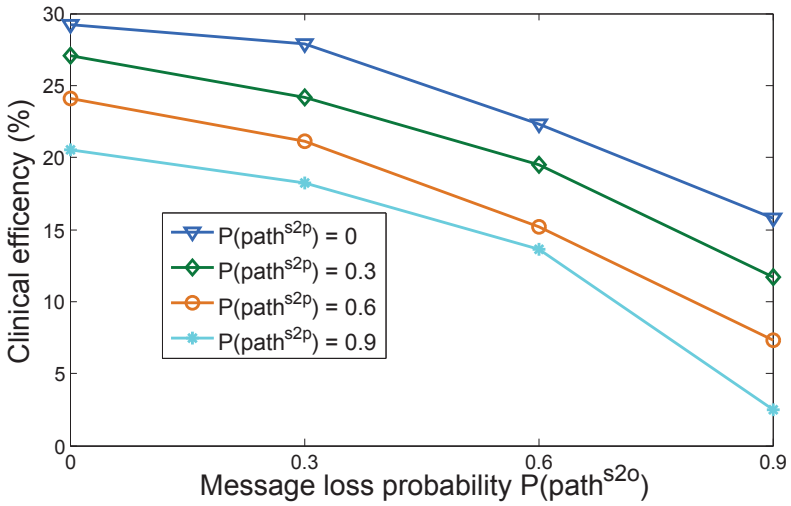


Fig. 4.5: Clinical efficiency

values, more severe message loss leads the supervisor to underestimate the real SpO_2 and CO_2 values more frequently, and give less time for the laser scalpel to operate in ALLOWED mode. However, under the moderate message loss condition, i.e., $P(path^{s2o}) = 0.6$ and $P(path^{s2p}) = 0.6$, our approach still achieves comparable efficiency value (15.20%) in contrast with the best case scenario where no message loss exists. This result proves our approach's capability of tolerating message loss.

Average Response Time

Response time affects the surgeon's experience and clinical efficiency. Fig. 4.6 plots the results of average response time for our approach and NASS when message loss probabilities vary. In the figure, NASS 200 and NASS 400 refer to the NASS framework with cycle length equal to 200ms and 400ms, respectively. Besides, mode vector length is set to a large value 20 for NASS 200 and NASS 400 to eliminate the impact of vector length [KSM+10]. We can see that ModePlan has much shorter response time compared to NASS. This difference

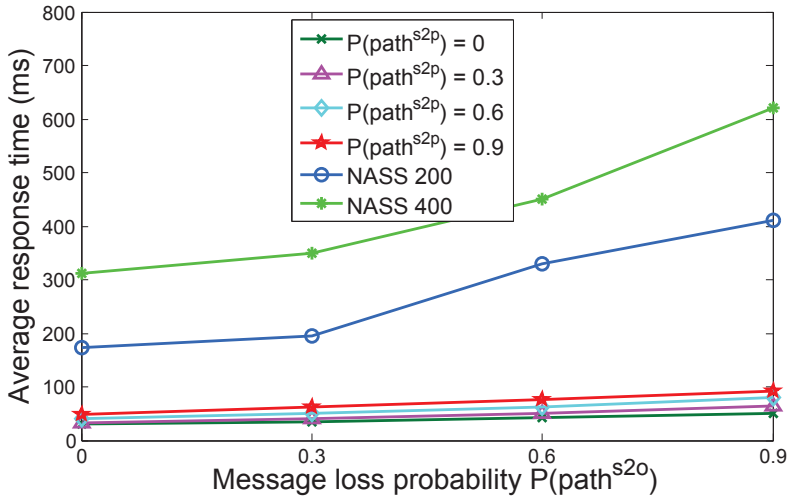


Fig. 4.6: Average response time

is mainly due to the periodic interaction mechanism used by NASS. In NASS, supervisor's response to device state change has to be deferred to the next cycle, resulting in longer delay, no matter how long the cycle length is. However, no such delay is posed by ModePlan. In fact, the response delays in NASS 200 and 400 are 5-10 times of that of ModePlan, according to the simulation results.

Communication Cost

As NASS needs periodic message exchange between the supervisor and medical devices, we can easily determine the total message number for the cases of NASS 200 and 400. To be specific, in each cycle, there are 8 messages exchanged. So in total, NASS 200 geneates $1200 * \frac{1000ms}{200ms} * 8 = 48000$ messages within the 1200-second surgery. Similarly, we can figure out the total message number for NASS 400, that is, $1200 * \frac{1000ms}{400ms} * 8 = 24000$. The results of communication cost for ModePlan are depicted in Fig. 4.7. As network condition becomes worse the total message number decreases slightly, because worse network

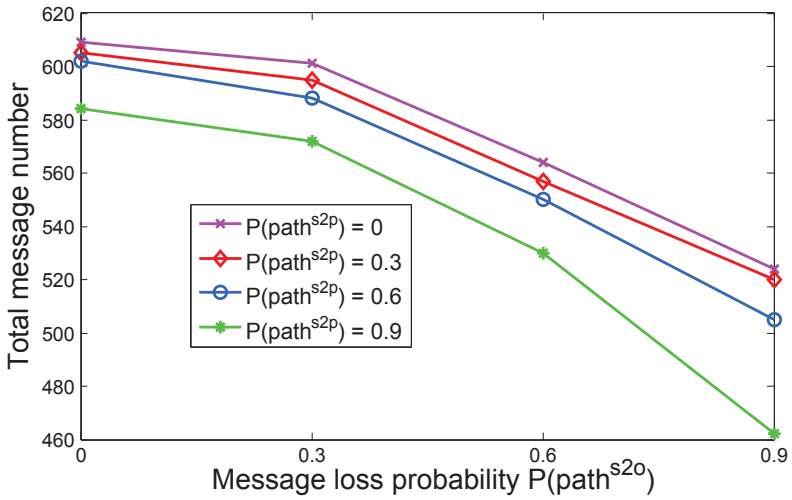


Fig. 4.7: Communication cost

condition results in less chances of control rule execution. Hence, less mode/state update messages are generated. In contrast with NASS, our approach maintains significantly lower communication cost (around a factor of 100 less).

4.5 Summary

Medical Cyber-Physical Systems (MCPS) often involve closed-loop control to achieve stability of patient's physiological state. In this chapter, we endeavor to enable closed-loop control for MCPS running over unreliable networks, rather than wired connections. We propose a novel coordination mechanism for MCPS to meet the closed-loop control requirements in the presence of network uncertainties, e.g., message loss and varied transmission delay, in the unreliable networks. The center of our proposed mechanism is a robust controller design. Additionally, we take into account the safety interlock constraints coming from the real medical applications. Our approach makes sure of patient safety when closed-loop control and safety interlocks co-exist in one MCPS. Joint consideration of closed-loop

control and safety interlocks broadens our approach's applicability to general MCPS cases. Furthermore, our approach is also practical in real medical environments, as it does not rely on perfect time synchronization among the devices. Finally, we demonstrated many advantages of our approach by trace-driven simulations, e.g., high clinical efficiency, low response delay to surgeon's request, and low communication overhead.

Chapter 5

Coordination for Context-awareness in Safety-critical MCPS

This chapter aims at enabling context-awareness for safety-critical CPS, e.g., MCPS. We will study how to coordinate devices towards context-aware medical systems, which are able to adapt the behaviors according to the user's situations. In this work, one sort of user situation that falls in our special interest is human error. To be precise, we treat human errors as contexts to the MCPS. Those contexts then serve as implicit inputs to the system and trigger the devices to coordinate their behaviors intelligently to prevent the potential harms as a result of human errors. In Section 5.1, we present the motivation under this work and give an overview about our proposed approach. Section 5.2 points out the challenges coming from context uncertainty, and introduces a system design to address these challenges. Starting from Section 5.3, we switch to one medical case named Patient-Controlled Analgesia, and aim to prove the effectiveness of our approach through this case study. We develop a context-aware MCPS for PCA and elaborate its design in Section 5.4. The system is evaluated in Section 5.6. Finally, this chapter is concluded in Section 5.7.

5.1 Motivation and Overview

The large number of medical accidents have attracted tremendous attentions from the society and regulatory departments [KCD00]. Undoubtedly, there is an urgent need to reduce the accidents. Recently, there has been an increasing interest in integrating and coordinating networked medical devices to improve medical care safety and quality. The integration and coordination of medical devices combined with patient physiologic dynamics have generated a new type of system — Medical Cyber-Physical System (MCPS) [LS10]. Such systems are typical Cyber-Physical Systems (CPS), as they involve both cyber components (e.g., embedded medical device software and network) and physical components (e.g., patient's body). Tight coupling of cyber and physical components in MCPS requires us to jointly consider them in the design of medical device coordination mechanism.

So far, most of the research in MCPS follows this principle, and has been successful in many medical cases, like Airway-laser Surgery [KSM$^+$10] [L$^+$12b] and Patient-Controlled Analgesia [A$^+$10]. However, one drawback of these research is that they all neglected the factors related to the users of MCPS. In fact, medical procedures in the hospital highly demand human's intervention. Timely, safe, and effective delivery of treatment to patient is possible only if the right users (e.g., surgeon, nurse) are performing right operations in the right situations. Without knowledge of user's status, a MCPS may not be able to coordinate medical devices appropriately or effectively. For example, without awareness of user's identity, an operation conducted by an unauthorized user will be accepted by the MCPS. On the contrary, such erroneous operation should be denied to avoid the resulting harms to patient.

In recognition of the impacts of user factors on medical care safety and quality, we propose, for the first time, to incorporate those factors into the MCPS design. Specifically, we leverage the contextual information that can characterize users' situation when they are interacting with MCPS, and make the system adaptive to these contexts. With the help of various contexts, we are able to recognize the occurrence of errors generated by the users (called *user errors*), and thus control medical devices properly to resist those errors and provide safeguard for patient. However, developing such context-aware MCPS is non-trivial as it faces the following two challenges. First, it is domain-specific, and involves tremendous medical knowledge. For example, we should understand how people interact with the MCPS, how user errors occur, and the responsibility of each person, etc. Then, we can find out what contextual information can be used to infer user errors (high-level contexts) [Dey01], and decide how the MCPS adapts its behavior to the contexts. Inappropriate design will lower the effectiveness of context-awareness or even cause safety leaks. Second, *context uncertainty* is an inevitable problem, and can potentially lead to inappropriate coordination of medical devices that will harm the patient. It is well acknowledged by the pervasive computing community that the contexts we obtain are commonly subject to uncertainty [B$^+$10], and automatic and complete resolution of uncertainty is impractical in complex environments, like hospital [DM05]. Suppose a life-supporting medical device, like ventilator, is stopped improperly because of false recognition of an uncertain context, patient's life would be threatened.

Regarding the first challenge, communication and collaboration with medical experts could be helpful, and furthermore, we can identify the requirements for context-aware M-CPS through extensive survey of related medical reports. To deal with the second challenge

(i.e., context uncertainty), there exist a wealth of research on modeling and reasoning on uncertain contexts [B+10] [RAMC+04b], as well as mitigation of uncertainty [XC05] [DM05]. However, none of them has investigated how to prevent the adverse impacts of context uncertainty, especially on patient safety in medical settings. Our approach to solving this problem is based on the following two principles. First, it is crucial to increase the certainty degree of context. Depending on the specific aspects of uncertainty (e.g., confidence, accuracy, inconsistency), corresponding approaches, such as multi-modal sensing [RMJ+11] and context inconsistency resolution [XC05], can be applied. Thereby, the likelihood of false context detection can be reduced. Second, we revise the existing "context-action" style of adaptation to contexts, and propose a *"context-assessment-action"* style for MCPS. In other words, prior to executing the context-triggered actions, we involve a safety assessment step which evaluates the potential consequence of the actions based on solid domain knowledge. An action is prohibited unless it is validated to be safe to the patient. As a result, although context uncertainty can potentially induce risky actions, each action finally allowed to be executed is assured to be free of harm.

Following the above principles, we then investigated a specific medical procedure, named *Patient-Controlled Analgesia (PCA)*, and developed a context-aware MCPS for it. PCA is a widely adopted procedure for post-operative analgesia in hospital, but the error rate in PCA is estimated to be higher than 4% [M+09]. Because of the rigorous regulatory and ethical issues, we cannot test the system on real patients. Instead, we deployed it in a ward-like environment and invited several students in the tests. Evaluation results show that in comparison with the existing system proposed in [A+10][KAL+10] (non-context-aware), we are able to prevent 95% more adverse events while always preserving patient safety.

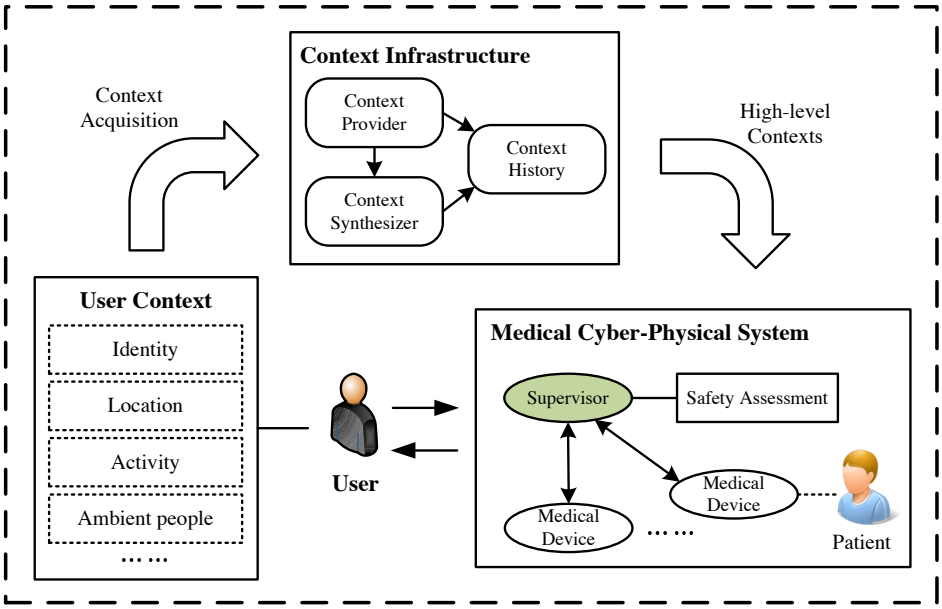


Fig. 5.1: Overview of context-aware MCPS.

Meanwhile, we are able to suppress false positive detection of user errors under a very low rate. These promising results manifest the great potential of enabling context-awareness for MCPS in reducing medical accidents.

5.2 Context-aware MCPS Design

In this section, we first give an overview of the context-aware MCPS in our design. Then, we will elaborate how the MCPS changes its behavior with respect to the contexts.

5.2.1 System Overview

It is a trend that future medical devices will be equipped with networking capabilities [LS10]. MCPS aims at improving treatment safety and efficiency by integrating and coordinating the networked medical devices. Based on the standard proposed by the MDP-nP Interoperability program [AST09], a central controller, called *supervisor*, is introduced

in MCPS to oversee the states of medical devices and coordinate the devices (see Section 2.1). In this work, we point out an additional role (i.e., system's user) that should not be neglected in the MCPS design. Here, *users* of MCPS include physician, nurses, patients themselves, and any relevant people that can potentially use the system. It is worth noting that users are *not* limited to the medical professionals who offer medical care to patients. By considering this additional role, we propose a holistic approach to designing MCPS. To be specific, we empower the MCPS to perceive user's status (e.g., identity, activity, ambient people) during user's interaction with MCPS, and adapt its device coordination decision accordingly. In this paper, we regard the relevant user status information as implicit *contexts* to the MCPS [Dey01], and then propose to build context-aware MCPS. Fig. 5.1 depicts an overview of the context-aware MCPS. A Context Infrastructure has been constructed to support the acquisition and inference of these contexts. The obtained high-level contexts then will be sent to the MCPS, which will accordingly adjust its device coordination decision to assist the user and provide further safeguard to patient.

The Context Infrastructure can be constituted by both medical devices and other normal devices deployed in the environment, e.g., light sensor and computer. Moreover, in the infrastructure, devices can play different roles. Several typical roles are explained as follows. *Context Providers* represent the devices acting as sources of context information, e.g., physical or logical sensors. *Context Synthesizer* obtain the contexts from Context Providers, and are able to perform inference and deduce high-level contexts based on the existing ones. *Context History* stores various important historical contexts for future use. In addition, it is necessary for the Context Infrastructure to provide support for context uncertainty, e.g., representation and inference of uncertain contexts.

5.2.2 System Design

Now that we have an overview of the context-aware MCPS. In the following, we will specify the detailed design for context-aware MCPS. Medical applications differ from traditional context-aware applications (e.g., smart home, smart office) in that patient safety is a big concern. The designed context-aware system has to never violate the safety requirements. However, such goal is not easy to achieve, as the design requires lots of medical knowledge, and the context uncertainty problem will further complicate the design task. To make the task tractable, we divide it into two sub-tasks and tackle them separately. First, we need to identify the contexts to which the MCPS should adapt. Then, appropriate devices can be found to construct the Context Infrastructure to obtain those contexts. Second, we should define the adaptation rules for the MCPS. In other words, we should specify how the system changes its coordination behavior corresponding to the contexts.

Sub-task 1: Context Identification

To deliver medical treatment to patient, MCPS commonly has to rely on human's intervention, e.g., an airway-laser surgery requires surgeon to operate the laser knife to open patient's trachea. During the manual operation of medical devices, users are prone to make mistakes. In fact, human errors are a long recognized problem in hospital [KCD00][Bog94], and they are usually followed by tragic consequences. According to [Bog94], such human errors that emerge when users are interacting with medical systems are referred to as *user errors*. Obviously, if users operate correctly, there is no need for the MCPS to adapt its behavior. Therefore, we are concerned about the situations when user errors occur. So the objective of Context Infrastructure is to provide context support for recognizing the user

Table 5.1: Adaptation rules for the MCPS

Context	**Intended Action**	**Assessment Result**	**Final Action to be Executed**
Commission error	Reject the user operation	Safe	Reject the user operation and notify the user
		Unsafe/ Unknown	Accept the user operation
Omission error	Autonomously operate the medical device	Safe	Autonomously operate the medical device and notify the user
		Unsafe/ Unknown	Remind the user of the expected user operation

errors. Then, preventive actions can be taken by the MCPS to resist the user errors.

Now let us analyze how user errors happen so that we can find out what contexts can be utilized for recognizing them. Diagnosis and treatment for a type of disease commonly have standard procedures, and almost every medical device has its intended use and specified use conditions. From the human behavioral point of view, user errors can be classified into two types [Lat07][Bog94]:

- **Errors of commission:** users perform erroneous operations on the medical devices;
- **Errors of omission:** users do not perform certain critical operations as expected.

In practice, there exist a variety of factors that can cause the above errors, such as stress, fatigue, inadequate knowledge, etc. Usually these two types of user errors are *high-level contexts* inferred from low-level sensory data. Once these errors have been detected by the Context Infrastructure, the MCPS will be notified to take appropriate actions.

Sub-task 2: Adaptation to Contexts

Tab. 5.1 describes how the MCPS adapts its behavior to the detected user errors. For the commission errors, the MCPS should take actions to directly reject the erroneous user

operations before they take effect and harm the patient. However, the tricky thing is that context uncertainty problem can cause false detection of user errors. For example, a correct operation may be regarded as a commission error by the Context Infrastructure. Hence, it is necessary to check the safety of each context-triggered action prior to executing it. Now we adopt a *"context-assessment-action"* style for the MCPS to adapt to contexts, instead of the "context-action" style used by the traditional context-aware systems (e.g., smart home). In other words, we force the MCPS to assess the safety of an intended action first rather than taking the action directly. A special component in the MCPS, named *Safety Assessment*, has been constructed to handle the safety assessment based on solid medical knowledge and real-time patient state (see Fig. 5.1). Following this "context-assessment-action" style, rejection of a user operation is disallowed unless it is certain to be safe. Furthermore, once a user operation is rejected, it is also important to notify the user of the rejection result and the corresponding reasons. Such feedback increases the user's awareness level, helping the user adjust subsequent decisions. On the other hand, in the cases where the rejection is evaluated to be unsafe or the potential impact of rejection is unknown, MCPS *conservatively* ignores the detected commission error, i.e., it chooses to accept the user's operation.

In the case of having detected an omission error, the MCPS will follow the "context-assessment-action" style to adjust its behavior as well. Once the Context Infrastructure determines a certain critical operation is missed by the user, the MCPS should autonomously operate the medical devices on behalf of the user. But, prior to taking this autonomous operation, the Safety Assessment component should assess its potential risk. Such action is allowed only if the assessment result validates it is safe to the patient. Otherwise, the system will simply remind the user of the omission error, asking the user to conduct further

check and make a judgement. Again, when an autonomous operation has been taken by the system, the user should be notified.

The above MCPS adaptation rules look simple at the first glance, but actually have several underlying design rationales. First, involving a safety assessment step is reasonable and practical. During our discussion with medical professionals, they pointed out that user operations in the hospital actually can be categorized. There exist many operations that do not have stringent time requirement. Hence, rejection of such operations can be considered to be safe as long as the user can be notified. For example, in Patient-Controlled Analgesia, rejecting user's request for additional pain medication will not compromise patient safety [Gra05][KAL$^+$10]. However, the challenge here is that the safety assessment step requires lots of medical knowledge, but sometimes it is very difficult to judge the potential risk due to the inadequacy of our understanding on medicine and patient's reaction [L$^+$12b]. In the latter case, the exact assessment result is *unknown*, and we conservatively regard the action as unsafe. Thus, we let the MCPS ignore the detected user errors, and choose to 'trust' user's behavior. Second, whenever the MCPS rejects a user operation or autonomously takes a device operation, it is desirable to notify the user of such actions. Lack of such information is possible to lead the user to make latent mistakes in the future, as user operations are likely to have inter-dependencies.

Finally, it is worth noting that the certainty level of contexts is also very important. Even though involving a safety assessment step assures every execution of the context-triggered action is free of harm, it is not desirable that such actions are not necessary, i.e., they are triggered by contexts that do not occur in reality (i.e., false positive context detection). For example, in Patient-Controlled Analgesia, frequent rejection of patient's

request of pain medication will result in under-medication and patient frustration on PCA therapy [D'A08]. In this sense, it is crucial to mitigate the degree of context uncertainty to some extend. Therefore, we can avoid high rates of false context detection. Since a context can be uncertain in many different aspects (e.g., accuracy, confidence level and ambiguity) [B$^+$10], corresponding uncertainty resolution/mitigation techniques should be applied. For example, based on data fusion, we can improve our confidence level on a detected context by fusing data from multi-modal sensors.

In the following sections, we will apply the design proposed in the above to a concrete medical case, i.e., Patient-Controlled Analgesia (PCA). On the one hand, we will prove the applicability of our approach in designing context-aware MCPS. On the other hand, through this case study, we intend to show the benefits of enabling context-awareness for MCPS.

5.3 User Errors in PCA and Insufficiency of Existing Systems

A comprehensive description about PCA and an MCPS developed for PCA (called PCA-MCPS) has been introduced in Section 2.1.2. In this section, we will point out some common user errors emerging in the PCA process. In addition, some drawbacks about the existing PCA-MCPS will be presented. These drawbacks will be overcome by our context-aware PCA-MCPS proposed in the next section.

5.3.1 User Errors in PCA

PCA is a medical procedure involving multiple roles of people in hospital (e.g., prescriber, pharmacist, nurse, and patient) and a sequence of operations. Refer to Section 2.1.2

for the detailed description of PCA process. Protecting patients from overdose is a great concern in addition to relieving patient's pain level. Unfortunately, inappropriate operations conducted by unauthorized people are likely to cause safety issues. The following lists several major types of user errors that occur in the current PCA practice.

- **E1: Drug misuse.** Pharmacist may mix up drugs, as some opioids for PCA use have similar names and packaging, e.g., *morphine* and *hydromorphone* [Coh05]. Selection of syringes/bags filled with incorrect drug type/concentration and loading into PCA pump can easily lead to overdose.

- **E2: Misprogramming.** Misprogramming refers to incorrect setting of any PCA pump parameters, e.g., dose limit. Typical programming errors include choosing wrong unit (mg vs. mL), and making wrong decimal point (10.0 mg vs. 1.00 mg).

- **E3: PCA-by-Proxy.** The intended user of PCA pump is patient. *PCA-by-Proxy* means that someone other than the patient presses the bolus button to initiate medication injection. These people are usually patient's family members and friends, who press bolus button with good will of relieving patient's pain. However, PCA-by-Proxy is dangerous, as it can generate extra medication injection which is not demanded by patient. For ease of narration, we will refer to patients themselves pressing bolus button as *Non-PCA-by-Proxy*.

Obviously, the above user errors all belong to errors of commission. In the current PCA practice, they often can not be noticed until serious consequences happen to patient. As a result, in United States, there has been a large number of adverse PCA-related events reported to the FDA [Coh05] [US 10].

5.3.2 State-of-the-Art MCPS for PCA and its Drawbacks

To combat the safety issues resulted from the above user errors, University of Pennsylvania has developed a closed-loop control based PCA system (shown in Fig. 2.2) [A$^+$10][KAL$^+$10]. Such a system is called *PCA-MCPS*. The system composes a PCA pump, a supervisor, and a pulse oximeter, and contains a closed control loop. The pulse oximeter continuously monitors patient's blood oxygen saturation (SpO2) as an indicator of patient's response to analgesic drug. Based on the feedback of patient's SpO2, the supervisor is able to automatically stop the PCA pump when patient's SpO2 falls below certain threshold, thus preventing the risk of further drug delivery.

Unfortunately, PCA-MCPS is still not sufficient to safeguard patients. For example, drug misuse and misprogramming can lead to significant overdose [Coh05]. However, there may exist a profound delay for the respiratory depression developed by the patient and observed by the supervisor [Gra05]. In this case, it will become too late to stop PCA pump, and thus fails to prevent overdose. Furthermore, PCA-by-Proxy can generate unwanted drug delivery which can lead patient to suffer more serious nausea [A$^+$10]. Unfortunately, the existing PCA-MCPS is not capable of detecting and preventing PCA-by-Proxy. To solve the above problems, we propose to enable context-awareness for the PCA-MCPS. Specifically, the context-aware PCA-MCPS is sensitive to the occurrence of user errors and can proactively prevent the harmful impacts induced by user errors, instead of taking posterior actions (i.e., stop PCA pump) after patient has developed observable adverse response.

Table 5.2: System adaptation and context infrastructure support

Error	Assessment Result	Final Action	Low-level Context	Supporting Device
E1	Safe	Reject the loaded syringe and alert	Syringe information	RFID reader and tags
E2	Safe	Reject the setting and notify another nurse	User Identity	Nanotron sensor
			Distance	Nanotron sensor
E3	Safe	Reject the bolus request and alert	User identity	Nanotron sensor
				Microphone sensor

5.4 Context-aware PCA-MCPS Design

Based on our approach proposed in Section 5.2, we will introduce the detailed design for a context-aware PCA-MCPS in this section. To be specific, we will present how the PCA-MCPS adapts to contexts as well as the construction of Context Infrastructure.

5.4.1 System Adaptation

Recall that there are three types of user errors (E1, E2, and E3) in PCA. Tab 5.2 describes how the context-aware PCA-MCPS adapts its behavior to these errors. Since these errors all belong to commission errors, it is necessary for the Safety Assessment component to evaluate the safety of rejecting user's operation. As shown in the second column of Tab. 5.2, all the assessment results are 'safe'. This is because PCA does not have strict time constraints; rejecting/delaying user's operation does not generate significant impact on patient safety.

When error E1 has been detected, the system will force the PCA pump to block the drug delivery from the loaded syringe/bag, because drugs of wrong type or concentration

must be banned from being used. Meanwhile, it will warn the responsible nurse to remove this error. To combat error E2, an effective approach is recommended by [Coh05], that is, we can involve two nurses checking the programming setting independently, thus reducing the chance of accepting incorrect programming parameters. So when the first nurse finishes programming PCA pump, the system intentionally considers misprogramming happens, and thus temporarily rejects the programming setting. Afterwards, the system will notify another responsible nurse to double-check the pump setting. The error state of programming will be cleared once the second responsible nurse appears in the PCA pump's proximity and has confirmed the pump setting. In the case of error E3, the system will reject user's request for bolus dose, and additionally warn the user of the PCA-by-Proxy error, which will help the user avoid PCA-by-Proxy error in the future.

5.4.2 Supporting Devices for Context Acquisition

In Tab. 5.2, we also describe the low-level contexts (fourth column) needed to infer the user errors (high-level contexts), as well as the devices for acquiring these low-level contexts (fifth column). To recognize error E1, we choose RFID technology. RFID tags attached to drug syringes/bags contain drug types and concentration information. Thus, drug misuse can be easily detected. For recognition of E2, we involve two nurses checking programming settings independently. In order to differentiate nurse's identity, Nanotron sensors will be used. Nanotron sensors are developed by our lab, and can be attached to people/objects for identification and measuring mutual distance (aka. ranging). Fig. 5.2 shows a picture of Nanotron sensor. Its ranging capability will also help the system detect nurse's appearance in the PCA pump's proximity. To recognize E3, user identity needs to be known by the

system. As the delivery of bolus dose can directly affect patient's safety and experience, we propose to use two types of sensors, i.e., Nanotron sensor and microphone sensor, to reliably detect user identity. The latter one is deployed beside the PCA pump to capture the voice of ambient people so as to recognize their identities. As either type of the sensors can not perfectly detect user identity, by fusing the results from both sensors, we can improve our confidence about the detected user identity, i.e., mitigate context uncertainty. Such data fusion method is also known as *multi-modal sensing* [RMJ+11].

Remark: In the above, we just present one way of choosing low-level contexts and the supporting devices. Of course, there exist alternative choices, or even better ones. For example, instead of microphone sensor, fingerprint sensor could be embedded in the bolus button for identifying the person pressing it. We argue that the primary goal of this paper is to prove the concept of context-awareness for MCPS and its benefits, rather than discussing the system implementation details.

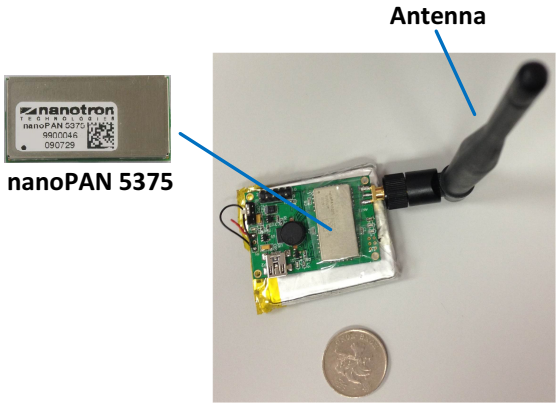


Fig. 5.2: Nanotron sensor developed by our lab

5.4.3 Context Representation and Reasoning

Now that appropriate devices have been found for context acquisition, we need find a way to represent the captured contexts and perform context reasoning. With respect to context uncertainty, in this paper we are only focused on the *confidence level* which reflects the likelihood that a detected context occurs in reality. Specifically, our approach of modeling uncertain contexts is based on *first-order probabilistic logic (FPL)*.

Context Representation

Based on FPL, a context C is represented in predicate format: ContextType(Entity1, Entity2, ...), where ContextType is the type of the context and correlates several entities. For example, we can use predicate Location(Dr. CHAN, Room 703) to express context "location of Dr. CHAN is room 703". For details about context presentation in first-order logic, please refer to [RC03]. Here, we are more concerned about modeling uncertainty factor. To measure the degree of our confidence on a context C, we choose a probabilistic approach, and denote the corresponding uncertain context by UC. So we represent UC in the following way: $Prob_c :: C$, where $Prob_c$ is a probability value in $[0, 1]$ and reflects our confidence level about context C. This expression style actually follows the conventions in ProbLog [RKT07], which is a probabilistic logic programming language. The following shows one example of uncertain context in our system: 0.9 :: Location(Dr. CHAN, Room 703), which says the probability that Dr. CHAN's location is room 703 is 0.9.

Context Inference

Another advantage of using FPL is its strong expressiveness, i.e., it allows us to express/infer new contexts from the existing ones by using *logical connectives*, e.g., negation

($\neg$), conjunction ($\wedge$) and implication ($\rightarrow$), as well as *quantifiers*, e.g., existential quantifier ($\exists$) and universal quantifier ($\forall$). Furthermore, it allows us to perform context inference without losing probabilistic information. For example, a typical probabilistic reasoning rule is written as follows:

$$Prob_d :: C_{new} \text{ :- } C_1, C_2, ..., C_n,$$

where C_{new} is a new context derived from contexts $C_1, C_2, ..., C_n$, and $Prob_d$ denotes our confidence about the derivation. In other words, $Prob_d$ is the probability that C_{new} is true if $C_1, C_2, ..., C_n$ are all true. Basically, the clause following symbol :: is equivalent to $(C_1 \wedge C_2 \wedge ... \wedge C_n) \rightarrow C_{new}$. Hence, if we know the probability of C_i, $1 \leq i \leq n$, probabilistic logic is able to automatically calculate the probability of C_{new} based on the rule above. We call the device performing such derivation a Context Synthesizer.

5.4.4 Multi-modal Sensing

In contrast with reasoning about new contexts, multi-modal sensing improves the quality of an existing context. Suppose there are k types of sensors detecting a single context. The context obtained by sensor of type i is denoted by UC_i. Multi-modal sensing fuses the results from those k types of sensors, and we denote the fused context by UC_f. The equation for calculating UC_f's confidence level, denoted by $P(UC_f)$, is listed as follows.

$$\begin{aligned} P(UC_f) &= F_f\big(P(UC_1), P(UC_2), \ldots, P(UC_k)\big) \\ &= 1 - \prod_{i=1}^{k}(1 - P(UC_i)), \end{aligned} \tag{5.1}$$

where F_f is a fusion function and $P(UC_i)$ is the confidence level on context UC_i. This equation essentially means that the fused context happens if and only if any of the k types

of sensors has detected it. Actually data fusion of this manner belongs to *decision fusion* and assumes independence among the sensors. In other words, we fuse the decision values (i.e., probabilities of the context) locally determined by the k types of sensors, rather than fusing raw sensory data.

5.4.5 Detection of Contexts

After going through context reasoning and fusion, finally we will obtain a confidence value for each error listed in Tab. 5.2. Let us denote the confidence value for user error E_i by $P(E_i)$, where $1 \leq i \leq 3$. To decide the acceptance of E_i, we associate E_i with a threshold θ_i. Once $P(E_i) \geq \theta_i$, we believe E_i is happening. Then, PCA-MCPS should launch corresponding safety assessment and adaptation actions. Otherwise, E_i is ignored. However, no matter what value of θ_i is selected, inevitably it will result in false positive (FP) and false negative (FN) detection of these errors. In fact, the values of θ_i reflect our system's sensitivity to the contexts. For example, too low threshold may lead to too many false recognitions of user errors, thus resulting in frequent rejection of user's operation, which will annoy the user. On the contrary, too large threshold will lower the effectiveness of adopting context-awareness. Therefore, we need careful selection of θ_i and we will discuss how θ_i affects the context-aware system in the evaluation section.

5.5 Implementation of the Prototype System

In this section we will introduce a prototype context-aware PCA-MCPS we developed based on the above design. We will describe the implementation details for each individual device, as well as uncertain context representation and inference. Besides, we will introduce how to calculate the confidence level for error E3 (PCA-by-Proxy) to exemplify the technique

of multi-modal sensing for context uncertainty mitigation. We omit the details of confidence level calculation for other user errors due to space limitation.

5.5.1 Emulated PCA Pump

As medical devices need to go through rigorous approval by regulatory agencies [L+06], like FDA in U.S., finding a PCA pump that is modifiable in software is not easy. It is even harder to use a modified PCA pump for real analgesic therapy. Therefore, we use a laptop to emulate a PCA pump. Clicking the mouse connected to the laptop corresponds to pressing the bolus button in a real PCA pump. Meanwhile, nurse can program the emulated pump by using the laptop keyboard to key in various parameters. The emulated PCA pump can accept the same settings as a real one. Besides, to mimic the situation of loading bag/syringe onto the PCA pump, we place a RFID tag onto the laptop. The RFID tag contains both drug type and concentration information related to the syringe/bag.

5.5.2 Bluetooth-enabled Pulse Oximeter

Another medical device in the PCA-MCPS is pulse oximeter. In our prototype, we choose a Bluetooth-enabled pulse oximeter, Nonin Model 4100 [Non08] (shown in Fig. 5.3). It is placed on patient's finger clip to measure patient's SpO2. Wireless pulse oximeter poses less restrictions on patient's movement, compared with wired one. We developed a special receiving program which could establish Bluetooth connection with the pulse oximeter as well as receive the stream of SpO2 data continuously. Sampling period of the pulse oximeter is 1/3 second.

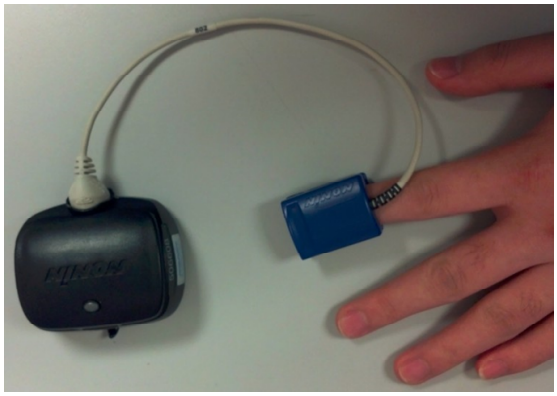


Fig. 5.3: Nonin Model 4100 Bluetooth pulse oximeter

5.5.3 User Identification based on Voice

To detect PCA-by-Proxy, a microphone sensor is mounted on the PCA pump, and continuously records the voice from the ambient environment. The tool we utilized for speaker recognition is Alize/LIA_SpkDet [Bon08], which determines speaker's identity based on voice features rather than speech contents (i.e., *text-independent speaker recognition*).

First of all, a 5-minute voice log about the patient are required to train a speaker model for the patient. Afterwards, while bolus button is pressed by someone, the system will check whether there exist some people other than the patient in the PCA pump's proximity, based on the recorded voice file about the most recent 3 minutes. If no speech has been detected in the voice file, the probability of PCA-by-Proxy, denoted by $Prob_v$, is 0 (see Equ. 5.2). Otherwise, we are able to compute a score for the voice file, indicating the degree that the voice matches the patient's speaker model. After normalization of the score, we obtain a value $p \in [0, 1]$. Essentially, p reflects our confidence level about the situation that there exist someone in the proximity and talking with the patient. Reasonably, we can use p to

characterize the chance that someone else can press the bolus button (i.e., PCA-by-Proxy).

$$Prob_v = \begin{cases} 0, & \text{no speech,} \\ p, & \text{otherwise.} \end{cases} \tag{5.2}$$

Nevertheless, merely relying on microphone sensor can not reliably detect PCA-by-Proxy, e.g., when PCA-by-Proxy happens but nobody speaks during the voice recording period. In the following, we will incorporate another type of sensor, i.e., Nanotron sensor, to improve the detection of PCA-by-Proxy.

5.5.4 Nanotron Sensor

The Nanotron sensor developed by our lab consists of a nanoPAN 5375 RF module, a battery board, and an antenna. As Fig. 5.2 shows, the size of the sensor is relatively small, thus allowing easy attachment to people or objects. The main component in nanoPAN 5375 RF module is a nanoLoc TRX transceiver [Nan08], which is capable of performing identification and precise ranging based on IEEE 802.15.4a Chirp Spread Spectrum (CSS) technique [KPGT10].

Now we describe how to calculate the probability of PCA-by-Proxy based on Nanotron sensors. As there exist many random factors in the real world [RAMC$^+$04b], accurate calculation of the probability is very difficult. Here we just present a method for rough calculation. If nobody, except the patient, appears within a certain range of the PCA pump, e.g., 1.5 meters, then we are safe to believe PCA-by-Proxy is impossible to happen. This range is called *critical range*, and denoted by D_c. The region centered at PCA pump and within D_c is called a *critical region*. We assume any person Per_i appearing within the critical region has certain probability to press the bolus button. Let us denote this

probability by $Prob(D_i)$, where D_i is the distance, measured by Nanotron sensors, between Per_i and the pump. We further assume that people closer to the PCA pump has higher probability of pressing the bolus button. Thus we can calculate $Prob(D_i)$ according to the following equation.

$$Prob(D_i) = \begin{cases} \frac{\pi * {D_c}^2 - \pi * (D_i - 2)^2}{\pi * (D_i + 2)^2 - \pi * (D_i - 2)^2}, & 2 < D_i < D_c + 2, \\ \frac{\pi * {D_c}^2}{\pi * (D_i + 2)^2 - \pi * (D_i - 2)^2}, & 0 \leq D_i \leq 2, \\ 0, & D_i \geq D_c + 2. \end{cases} \tag{5.3}$$

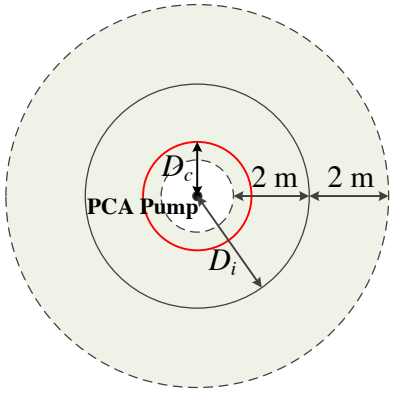


Fig. 5.4: Calculation based on distance information.

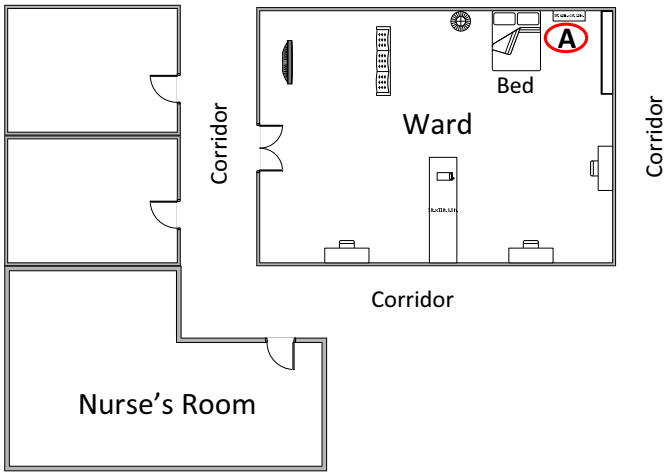


Fig. 5.5: Layout of the system deployment environment.

Since in reality, the ranging error of Nanotron sensor is 2 meters (see specification in [Nan08]), person Per_i can appear in any location whose distance to the PCA pump is in range $[max\{0, D_i-2\}, D_i+2]$. In the first case of Equ. 5.3, D_i falls in the range $(2, D_c+2)$, which is pictorially demonstrated in Fig. 5.4. Red dot in the center is the PCA pump. The region inside the red circle indicates the critical region. The region in grey covers all the locations in which person Per_i may appear. Given the 2-meter ranging error, the size of the grey region is $S_c = \pi * (D_i+2)^2 - \pi * (D_i-2)^2$. We denote the size of the overlapping between the critical region and the grey region by $S_o = \pi * {D_c}^2 - \pi * (D_i-2)^2$. Therefore, $Prob(D_i) = \frac{S_o}{S_c} = \frac{\pi * {D_c}^2 - \pi * (D_i-2)^2}{\pi * (D_i+2)^2 - \pi * (D_i-2)^2}$. For the second case of Equ. 5.3 where $0 \leq D_i \leq 2$, we have $S_o = \pi * {D_c}^2$. Thus, $Prob(D_i) = \frac{S_o}{S_c} = \frac{\pi * {D_c}^2}{\pi * (D_i+2)^2 - \pi * (D_i-2)^2}$. For the third case where $D_i \geq D_c + 2$, as Per_i now becomes so far away from the PCA pump, size of the overlapping area $S_o = 0$. Therefore, it is impossible for Per_i to press the bolus button, i.e., $Prob(D_i) = \frac{S_o}{S_c} = 0$.

We denote by $\mathcal{P}$ the set of people (except patient) whose distance to the PCA pump is less than $D_c + 2$ (for case 1 and 2 in Equ. 5.3). We consider Non-PCA-by-Proxy happens as long as none of the nearby person $Per_i \in \mathcal{P}$ presses the button. Hence, the probability of PCA-by-Proxy, denoted by $Prob_d$, can be determined by the following equation.

$$Prob_d = 1 - Prob(\text{Non-PCA-by-Proxy}) = 1 - \prod_{Per_i \in \mathcal{P}} (1 - Prob(D_i)). \tag{5.4}$$

5.5.5 Context Reasoning and Fusion

Context representation and reasoning in our system follow the FPL described in Section 5.4.3. We build a special server as Context Synthesizer, which provides the context reasoning and fusion services. A ProbLog-based inference engine [DTA13] has been implemented on

the server to support those derivations. Finally, by fusing the intermediate confidence levels determined from the Nanotron sensors and microphone sensor (multi-modal sensing), we can obtain the ultimate confidence level $Prob_f$ on PCA-by-Proxy by the following equation (based on Equ. 5.1):

$$Prob_f = 1 - (1 - Prob_d)(1 - Prob_v). \tag{5.5}$$

5.6 Evaluations

This section introduces how we deploy the context-aware PCA-MCPS and evaluate the system. Because of the rigorous regulatory and legal constraints, currently our tests are not based on real PCA therapy. However, we still endeavor to involve human in the test, and create a lot of situations similar to the real PCA scenarios. The prototype system has been deployed on the 8th floor of our department. The Intelligent Home lab is chosen as a *ward*, and a nearby office room as *nurse's room*. The layout of our deployment environment is depicted in Fig. 5.5. The emulated PCA pump is placed at location A, near the bed where patients lie. The same location is chosen for placing the RFID reader whose power is properly adjusted, so as to read the RFID tags on the syringes/bag loaded to the PCA pump. Nurses can walk back and forth between ward and nurse's room to change their locations.

5.6.1 Adaptation to Error E1 and E2

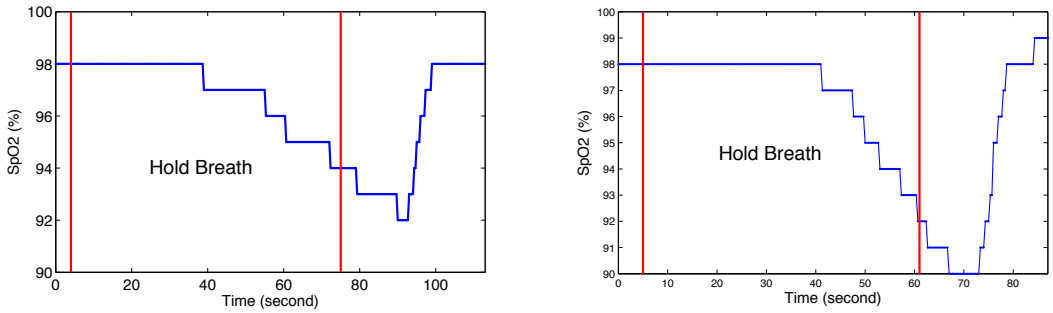
First of all, we demonstrate our system's capabilities of adapting to error E1 (drug misuse) and E2 (misprogramming). We invite one student to be patient lying on the bed, and another two students as nurses. Let us use NS1 to denote the primary nurse responsible for programming the PCA pump, and NS2 to denote the secondary nurse for

double-checking the pump setting, respectively. While NS1 chooses a RFID tag representing incorrect drug syringe/bag and places on the PCA pump, error E1 can be immediately and reliably detected, and then the system will sound alarm informing the nurse to remove the error. Afterwards, once NS1 finishes programming the PCA pump, the system will postpone the acceptance of the pump setting. Instead, it will remind NS2 to double-check the pump settings. The Nanotron sensors, associated with PCA pump and nurses, are able to detect the appearance of NS2 in the PCA pump's proximity (e.g., 1 meter). Once NS2 confirms the pump setting (may also modify the setting), the setting will be accepted by the pump, and from now on, the pump will be activated for patient use.

5.6.2 Adaptation to Error E3

Considering the diversity of human's response to analgesic drugs, our test involves two students as patients, one male and one female. To mimic the effect of receiving analgesic drug, we ask the students to hold breath for a while, e.g., 1 minute, thus causing SpO2 to drop. Fig. 5.6 shows two samples of SpO2 change during breath holding for the male and female students, respectively. Apparently from the figures, we can see the female student's SpO2 decreases much faster than the male. According to the SpO2 readings during student's breath holding, we set the threshold of SpO2 in the supervisor to 94%. In other words, SpO2 lower than 94% is considered to be risky and will trigger the supervisor to stop the PCA pump.

After initiating PCA therapy, we invite one more student to be patient's relative, denoted by RS. The student may press the bolus button from time to time to create PCA-by-Proxy errors. Meanwhile, RS can also change his location in the ward, i.e., change his distance



(a) Male patient's SpO2 (b) Female patient's SpO2

Fig. 5.6: Change of patient's SpO2 while breath is held (patients start holding breath at the first vertical line, and resume respiration at the second).

to the PCA pump. Specifically, we fix 50 locations in the ward for RS with distance to the PCA pump varying from 0 to 5 meters, and allow RS to talk with the patient from time to time. During the test, we record i) patient's SpO2 value when the bolus button is pressed, ii) real identity of the person pressing the bolus button, iii) final action of the system's adaptation. From statistics of the records, we find that no bolus requests have been accepted when patients' SpO2 is lower than threshold 94%, regardless of the identity of the person pressing the button. This result shows our system's capability of preventing additional drug delivery when patient's condition is not good enough.

In addition, we are interested in *false negative* and *false positive* rates of PCA-by-Proxy detection, which are denoted by R_{fn} and R_{fp}, respectively. The higher R_{fn}, the greater possibility that our system accepts bolus requests from unauthorized people. The higher R_{fp}, the poorer user satisfaction on PCA, because more bolus requests from patients can be rejected. Also, we denote the threshold for detecting PCA-by-Proxy by θ. Now let us investigate how R_{fn} and R_{fp} change when we vary the value of θ from 0.1 to 0.9, as well as enable/disable the speaker recognition function (SRF) in the microphone sensor. Fig. 5.7

and 5.8 plot the variations of R_{fn} and R_{fp} for the male and female patients, respectively. First, increasing threshold θ will cause false negative rate to increase, whereas false positive rate goes in an opposite direction. This result is not surprising, because a higher threshold will cause more real PCA-by-Proxy errors to be ignored by the system. Second, we can find from the figures that while enabling SRF, we are able to significantly reduce the rate of false negative. This is because utilizing multi-modal sensing helps to improve the confidence level on PCA-by-Proxy, thus suppressing a lot of rejections of bolus requests from real patients. However, a side effect of introducing SRF is that false positive rate is slightly increased sometimes (e.g., $\theta = 0.5$ for the male patient), because SRF is sensitive to the ambient noise and could regard it as patient's utterance (see Equ. 5.2). Consequently, SRF may introduce a very small value in the overall confidence level on PCA-by-Proxy. Obviously, the increase of confidence level will lead to higher false positive rate.

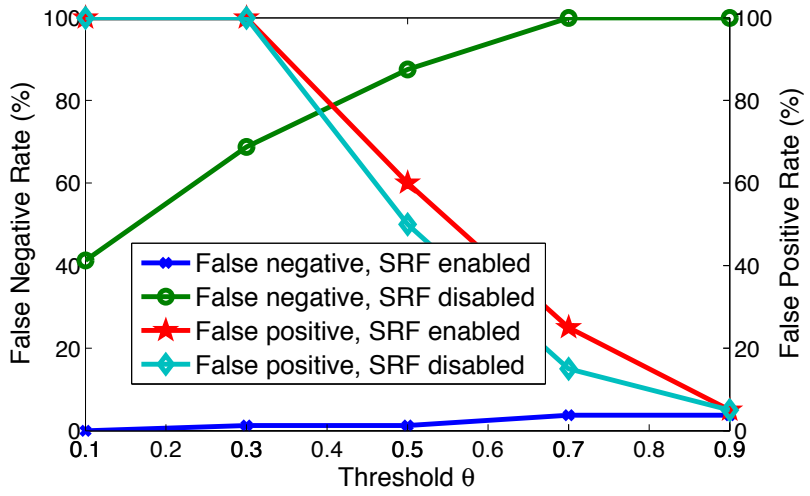


Fig. 5.7: False negative and false positive rates for the male patient.

In fact, through these findings, we know that it is necessary to adjust θ to balance the rates of false negative and false positive, so as to meet the design requirements. Finally, we

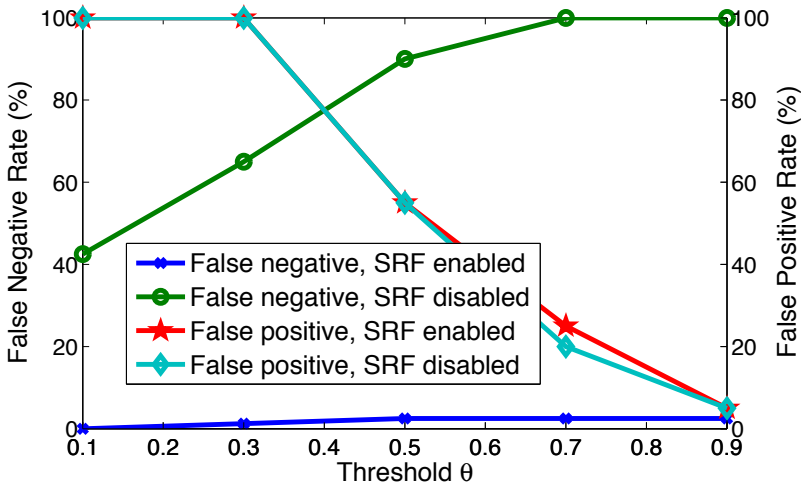


Fig. 5.8: False negative and false positive rates for the female patient.

Table 5.3: Statistics when $\theta = 0.9$ and SRF enabled

Patient	**Number of Real Non-PCA-by-Proxy**	**Number of Real PCA-by-Proxy**	**False Negative Rate**	**False Positive Rate**
Male	80	20	5%	3.75%
Female	80	20	5%	2.5%

can see from Fig. 5.7 and 5.8 that when setting θ to 0.9 and enabling SRF, we are able to achieve very low false negative and false positive rates simultaneously. Under this setting, the statistical results are listed in Tab. 5.3. Overall we recorded 80 real Non-PCA-by-Proxy cases and 20 real PCA-by-Proxy cases for both male and female patients. On the one hand, 95% of the real PCA-by-Proxy cases can be successfully detected and the corresponding bolus requests can be rejected for both male and female patients, i.e., $R_{fn} = 5\%$. Compared with the existing PCA-MCPS [A$^+$10][KAL$^+$10], which has no capability of preventing PCA-by-Proxy, this is obviously a significant improvement. On the other hand, the cost for this improvement is just very low false positive rates (rejecting patient's bolus requests), which are 3.75% and 2.5% for male and female patients, respectively. Under such low rates, we

claim that patient's experience will not be affected too much.

5.7 Summary

In this chapter, we have studied how to enable context-awareness for a special type of safety-critical system, i.e., Medical Cyber-Physical System (MCPS). In recognition of the impact of user factors on medical care, we treat user status as contexts to the MCPS when users are interacting with the MCPS. We design detailed rules for MCPS to adapt its coordination behaviors to the contexts. On one hand, although contexts obtained by the system are commonly subject to uncertainty, our approach does not introduce extra safety problems, compared with the non-context-aware systems. On the other hand, it exploits the contexts to perform appropriate actions to reduce various risks of human errors. To evaluate the feasibility and effectiveness of our approach, we have applied it in a widely-adopted medical case, named Patient-Controlled Analgesia (PCA), and built a context-aware MCPS for it. By testing the system under many scenarios of user errors, we demonstrate that our system has a stronger capability of preventing the adverse events, in contrast with the existing PCA systems. We believe the methods we developed for context-aware MCPS are also promising to be applied for building other context-aware systems with strong safety concerns.

Chapter 6

Conclusions and Future Directions

In this chapter, we conclude this dissertation by summarizing our contributions in Section 6.1 and pointing out some directions for further research in Section 6.2.

6.1 Conclusions

Cyber-Physical Systems (CPS) have been receiving more and more attention from the academia and industry, due to the crucial role they play in the economically vital domains, such as transportation, medical care, agriculture, energy, aerospace, and building. However, there still exist grand challenges to be addressed before coming to widespread applications. One of the challenging and fundamental problems considered in this dissertation is coordination. Individual and distributed CPS devices need to coordinate their activities so as to perform certain tasks posed by the CPS applications, and thus bind into a whole system.

This dissertation gets started with the investigation of special properties with respect to the CPS device coordination problem. In comparison with traditional distributed systems, coordination problem in CPS is more complicated due to i) the tight integration and interaction of cyber and physical components, ii) safety-critical and time-critical properties, and iii) a variety of uncertainties associated with the harsh cyber and physical environments,

like incomplete information and network condition variations. Therefore, coordination in CPS has to combat various uncertainties in order to meet various application requirements.

Having understood the characteristics of CPS coordination, we then developed a 3D framework assisting us to identify new and meaningful research problems in CPS coordination. The outcome of our endeavor is three specific coordination problems arising from the real situations of Medical Cyber-Physical Systems (MCPS), which are a typical CPS application in medical care.

The first coordination problem we studies is how to allow medical devices to collaborate to verify certain properties of a MCPS (in Chapter 3). Traditional formal verification, e.g., model checking, is commonly conducted offline prior to the running of a system. The reason why we have to coordinate MCPS devices in an online manner to perform verification is that there is no accurate model about patient's dynamics or tractable model to be fed into the existing verification tools. Our approach to solve this problem is based on an observation we found during extensive analysis of human-related medical traces, that is, patient's physiological state is quite predictable in the coming short period of time. For example, it is easy to predict the blood oxygen level curve in the next 4 seconds: it cannot plunge from 100% to 10%; instead, it has to be smooth, which can be effectively described with existing tools, such as linear regression. This observation thus motivated us to transform the traditional offline model checking to an online manner where we dynamically generate the system model for the coming short time span and then examine whether various desired properties are fulfilled by the system. Coordination of MCPS devices in this way yields a new approach of performing formal verification for CPS, when uncertainty about the physical components hinders the traditional offline verification.

Furthermore, we proposed several system co-design patterns which could guide the M-CPS design so as to integrate the online model checking procedure as a real-time task. In cases when the online model checking results fail to meet the safety property or can not terminate before the specified deadline, we force the system switch to a fall-back plan to circumvent any potential risks. In this way, we can build the online model checking as an add-on safety-enhancement service to the system. This co-design approach was supported by solid theoretical analysis. In addition, we took into account the unreliable communications between medical sensors and central controller, and improved our approach to be applicable to the unreliable communication environments. At last, we make our work complete by further discussing the impact of inaccuracy of online patient modeling. The feasibility and effectiveness of our approach have been validated through experiments by using real human traces and a state-of-the-art model checking tool, named PHAVer.

The second coordination problem we studied in this dissertation is how to coordinate medical devices in MCPS to assure patient safety in the presence of network uncertainties, and at the same time allow low delay to respond clinician's requests (in Chapter 4). Although unreliable communications have been considered in Chapter 3, in this work we study a more general case where the communication link between any pair of MCPS devices is unreliable, e.g., messages may get lost or encounter varied transmission delay. Research on this problem is of great importance due to the increasing adoption of wireless technologies in medical settings.

According to the Integrated Clinical Environment (ICE) standard [AST09], we find there exist two types of basic coordination mechanisms, e.g., closed-loop control and safety interlock, for enhancing treatment safety. Hence, our primary goal is to make these two

mechanisms capable of tolerating message loss and delay. Our approach to achieving this goal is based on a central controller, called supervisor, which plans the future operations for each medical devices. In case messages encounter serious delay or loss, each medical device is able to autonomously consume the planned operations at right time to behave. On one hand, the planning is carried out based on worst-case prediction of patient's future states. On the other hand, the supervisor makes sure that each generated future plan never violates the safety interlock constraints. Therefore, patient safety is assured. Furthermore, we hide the underlying networking and coordination complications for the applications in medical devices. From the application's point of view, closed-loop control and safety interlock are running over perfect network condition. Also, it is worth noting that our approach does not require perfect clock synchronization, thus very practical. The safety property and performance of our proposed solution has been evaluated by extensive trace-driven simulations. We believe that this research will make original contributions to achieving the vision of future operation rooms without any wires.

The third coordination problem (in Chapter 5) we studied differs from the previous ones in that the uncertainties originate from the external environment of MCPS, rather than internal entities, such as patient (in Chapter 3) and network (in Chapter 4). In this work, we propose to treat user's situations as contexts to MCPS. By enabling MCPS aware of and adapting to those contexts, we can significantly increase the system's intelligence and autonomy, and creates opportunities for proactively coordinating medical devices to prevent the risks of human errors. However, the challenge in enabling context-awareness comes from the uncertainty associated with the contexts. The limitations of sensing and reasoning techniques often lead the detected contexts to be uncertain. If not well handled,

uncertain contexts could induce unsafe operations of medical devices.

To solve this problem, we proposed an approach with two principles. First, we should increase the degree of certainty for the contexts. Thus, the likelihood of false context detection can be reduced. Second, the traditional adaptation paradigm of "context-action" should be revised to a "context-assessment-action" paradigm to fit the safety-critical medical environment. Specifically, in the revised paradigm, each context-triggered action should be assessed prior to its real execution. The "assessment" step, conducted based on solid medical knowledge, prevents the execution of any action that is suspected to be unsafe. As a result, we can make good use of context information to improve system's intelligence without introducing additional safety loopholes. To validate the effectiveness and advantages of our approach, we have developed a context-aware MCPS for Patient-Controlled Analgesia based on the above principles. Evaluation results have shown that our system exhibits much stronger ability of reducing adverse events.

6.2 Future Directions

Now that we have summarized all the work presented in this dissertation, in this subsection we will point out several directions that are worth further exploration in the future.

First, let us revisit the proposed online model checking approach for MCPS. Model checking originally was proposed to examine the conceptual design of a system. In the traditional offline model checking, the design of a system can be well modeled. However, when moving to the online approach, the system model is dynamically generated when the system is running. Note that there is a trick here. As we know, sometimes there is a discrepancy between the conceptual design and real implementation of MCPS devices. If

the implemented devices fail to conform to the original design, the generated online model can not capture the real system behavior. Consequently, the results yielded by online model checking can not reflect the real properties of the system. In this sense, our approach must rely on an assumption that the real implementation of each MCPS device perfectly complies with its design. As our future research, we will investigate how to remove this assumption, and meanwhile make sure the online model checking is still feasible and effective. One promising solution is to use model-driven approach to automatically generate codes for MCPS devices, so that the real MCPS devices' behaviors are always consistent with their models [LS10]. However, this future research can be considered as orthogonal to our online model checking and system co-design approach.

Second, with regard to the approach we proposed to handle network uncertainties in MCPS, all the MCPS devices must be synchronized in clock. Therefore, the planned operations can be safely consumed by the medical devices at right time. In the future, we plan to remove this time synchronization requirement, thus making our approach applicable in more general cases. However, the price is that the complexity will be significantly increased. In addition, we have been thinking of a MCPS scenario where each pair of devices communicating through wired links, and meanwhile we can construct wireless links to backup the wired ones. Once wired links occasionally get disconnected, wireless links can serve as secondary choice supporting continuous execution of the system. Such idea of combining wireless and wired connections would be very promising in improving the system's resilience to network problems. Furthermore, it can avoid the low performance issue when the system purely relies on wireless communication, because wireless link conditions may become poor in certain extreme environments and cause the system to be suspended for clinical use.

Finally, we would like to investigate how to enable MCPS adaptive to other types of contexts, in addition to the human errors considered in Chapter 5. Typical examples of such contexts are the computational and physical environments of the MCPS devices. Suppose a medical sensor is detached from patient's body, its reading should be discarded or regarded as invalid. So making MCPS aware of the 'attach' state of medical sensors can be helpful for the system to understand the real situation of the patient. Meanwhile, the external physical environment, such as temperature and humidity, can also affect the functioning of medical devices. It is apparent that there exists a great opportunity of improving the MCPS by adapting to those new types of contexts. However, still we will have to face the difficulties in tackling the uncertainties associated with those contexts. Since our current adaptation rules for handling context uncertainty are specifically designed for human errors, new rules will be needed for the new contexts. We believe that by exploring towards this new direction, we will be able to extend context-awareness to other safety-critical CPS application domains, such as avionics and transportation.

References

[A+92] Rajeev Alur et al. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. *Hybrid Systems*, 736:209–229, 1992.

[A+10] David Arney et al. Toward patient safety in closed-loop medical device systems. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 139–148, 2010.

[A+11] Rajeev Alur et al. Compositional modeling and analysis of multi-hop control networks. *IEEE Transactions on Automatic Control*, 56:2345–2357, 2011.

[AHH96] Rajeev Alur, Thomas A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.

[AK03] Panos J. Antsaklis and Xenofon D. Koutsoukos. Hybrid systems: Review and recent progress. *Software-Enabled Control: Information Technology for Dynamical Systems*, pages 273–298, 2003.

[Alu99] Rajeev Alur. Timed automata. *Lecture Notes in Computer Science*, 1633:8–22, 1999.

[AS03] Babak Azimi-Sadjad. Stability of networked control systems in the presence of packet losses. In *Proceedings of 42nd IEEE Conference on Decision and Control*, pages 676–681, 2003.

[AST09] ASTM International. Medical devices and medical systems – essential safety requirements for equipment comprising the patient-centric integrated clinical environment, part 1: General requirements and conceptual model, 2009.

[B+96] Johan Bengtsson et al. Verification of an audio protocol with bus collision using uppaal. *Lecture Notes in Computer Science*, 1102:244–256, 1996.

[B+08] S. Baker et al. Medical-grade, mission-critical wireless networks. *IEEE EMB Magazine*, 27(2):86–95, 2008.

[B+09] F. Brunicardi et al. *Principles of Surgery.* McGraw-Hill, September 2009.

[B+10] Claudio Bettinia et al. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6:161–180, 2010.

[BCL+94] Jerry R. Burch, Edmund M. Clarke, David E. Long, Kenneth L. McMillan, and David L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13:401–424, 1994.

[BGT+06] Yingyi Bu, Tao Gu, Xianping Tao, JunLi, Shaxun Chen, and Jian Lu. Managing quality of context in pervasive computing. In *Proceedings of the Sixth International Conference on Quality Software*, pages 193–200, 2006.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking.* The MIT Press, 2008.

[BKS03] Thomas Buchholz, Axel Küpper, and Michael Schiffers. Quality of context: What it is and why we need it. In *Proceedings of the 10th Workshop of the OpenView Univeristy Association (OVUA'03)*, Geneva, Switzerland, 2003.

[BMJ+98] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *ACM MobiCom'98*, 1998.

[Bog94] Marilyn Sue Bogner, editor. *Human error in medicine.* CRC Press, 1st edition, 1994.

[Bon08] Jean-François Bonastre. ALIZE/SpkDet: a state-of-the-art open source software for speaker recognition. In *Proc. IEEE Odyssey, ISCA Speaker Recognition Workshop*, 2008.

[BPZ00] Michael S. Branicky, Stephen M. Phillips, and Wei Zhang. Stability of networked control systems: Explicit analysis of delay. In *Proceedings of the American Control Conference*, 2000.

[C+11] George Coulouris et al. *Distributed Systems: Concepts and Design*. Addison-Wesley, 5th edition, 2011.

[Cam04] E. F Camacho. *Model predictive control*. Springer, 2nd edition, 2004.

[CCKM01] Paul Castro, Patrick Chiu, Ted Kremenek, and Richard Muntz. A probabilistic room location service for wireless networked environments. *Lecture Notes in Computer Science*, 2201:18–34, 2001.

[CGP00] E. M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, 2000.

[Coh05] Michael R. Cohen. Patient-controlled analgesia safety issues. *J. Pain Palliat. Care Pharmacother.*, 19(1):45–50, 2005.

[D'A08] Yvonne D'Arcy. Keep your patient safe during PCA. *Nursing*, 38:50–55, Jan 2008.

[DD07] Jerry A. Dorsch and Susan E. Dorsch. *Understanding Anesthesia Equipment, 5th ed*. Lippincott Williams and Wilkins, 2007.

[Dey01] Anind K. Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[DM05] Anind K. Dey and Jennifer Mankoff. Designing mediation for context-aware applications. *ACM Transactions on Computer-Human Interaction*, 12(1):53–80, 2005.

[DTA13] DTAI group of KULeuven. ProbLog. `http://dtai.cs.kuleuven.be/problog`, 2013.

[EGE02] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th symposium on operating systems design and implementation (OSDI)*, 2002.

[EKS06] Arvind Easwaran, Sampath Kannan, and Oleg Sokolsky. Steering of discrete event systems: Control theory approach. *Electronic Notes in Theoretical Computer Science*, 144(4):21–39, 2006.

[F+02] B. Finkbeiner et al. Collecting statistics over runtime executions. *ENTCS*, 70:36–54, 2002.

[Fre05] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems past HyTech. *Proceedings of HSCC'05*, LNCS 2289:258–273, 2005.

[G+00] Ary L. Goldberger et al. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101:215–220, 2000.

[GC10] Rachana Ashok Gupta and Mo-Yuen Chow. Networked control system: Overview and research trends. *IEEE Transactions on Industrial Electronics*, 57:2527–2535, 2010.

[GK09] Farid Golnaraghi and Benjamin C. Kuo. *Automatic Control Systems*. Wiley, 9th edition, 2009.

[Gra05] J. A. Grass. Patient-controlled analgesia. *Anesthesia & Analgesia*, 101(5S):S44–S61, November 2005.

[Gri08] Kristina Grifantini. "Plug and Play" Hospitals – Medical devices that exchange data could make hospitals safer. `http://www.technologyreview.com/news/410429/plug-and-play-hospitals/`, July 2008.

[GS01] Philip Gray and Daniel Salber. Modelling and using sensed context information in the design of interactive applications. *Lecture Notes in Computer Science*, 2254:317–335, 2001.

[GSC11] Dip Goswami, Reinhard Schneider, and Samarjit Chakraborty. Co-design of cyber-physical systems via controllers with flexible delay constraints. In *Proceedings of the 16th Asia and South Pacific Design Automation*, pages 225–230, 2011.

[H+07] J. P. Hespanha et al. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95:138–162, 2007.

[H+10] J. Huang et al. Beyond co-existence: Exploiting WiFi white space for ZigBee performance assurance. *Proceedings of ICNP*, pages 305–314, October 2010.

[HBB02] Jeffrey Hightower, Barry Brumitt, and Gaetano Borriello. The location stack: a layered model for location in ubiquitous computing. In *Proceedings Fourth IEEE Workshop on Mobile Computing Systems and Applications*, pages 22–28, 2002.

[HHWT97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Hytech: A model checker for hybrid systems. *Lecture Notes in Computer Science*, 1254:460–463, 1997.

[HKMP02] David Harel, Hillel Kugler, Rami Marelly, and Amir Pnueli. Smart play-out of behavioral requirements. In *Proceedings of the 4th International Conference on Formal Methods in Computer-Aided Design (FMCAD '02)*, pages 378–398, London, UK, 2002.

[Hof07] Robert Matthew Hofmann. Modeling medical devices for plug-and-play interoperability. Master's thesis, MIT, 2007.

[Hol90] Gerard J. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall, 1990.

[Hov06] R. Hovorka. Continuous glucose monitoring and closed-loop systems. *Diabetic Medicine*, 23:1–12, 2006.

[HP00] Klaus Havelund and Thomas Pressburger. Model checking java programs using java pathfinder. *International Journal on Software Tools for Technology Transfer*, 2(4):366–381, 2000.

[JPM11] Zhihao Jiang, Miroslav Pajic, and Rahul Mangharam. Model-based closed-loop testing of implantable pacemakers. In *IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS)*, pages 131–140, 2011.

[KAL+10] Andrew King, Dave Arney, Insup Lee, Oleg Sokolsky, John Hatcliff, and Sam Procter. Prototyping closed loop physiologic control with the medical device coordination framework. In *Proceedings of SEHC'10*, pages 1–11, 2010.

[KB03] Hermann Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91:112–126, 2003.

[KCD00] L. Kohn, J. Corrigan, and M. Donaldson. *To err is human: building a safer health system.* National Academy Press, Washington, 2000.

[KCMFL13] Benjamin A. Kohl, Sanjian Chen, Margaret Mullen-Fortino, and Insup Lee. Evaluation and enhancement of an intraoperative insulin infusion protocol via in-silico simulation. In *IEEE International Conference on Healthcare Informatics (ICHI 2013)*, 2013.

[KHC10] Eunju Kim, Sumi Helal, and Diane Cook. Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9:48–53, 2010.

[KKH+08] Nicholas Kottenstette, Xenofon Koutsoukos, Joe Hall, Panos Antsaklis, and Janos Sztipanovits. Passivity-based control design for Cyber-Physical Systems. Technical Report ISIS-08-904, ISIS Vanderbilt, 2008.

[KPGT10] Eirini Karapistoli, Fotini-Niovi Pavlidou, Ioannis Gragopoulos, and Ioannis Tsetsinas. An overview of the ieee 802.15. 4a standard. *IEEE Communications Magazine*, 48:47–53, 2010.

[KS11] Ajay D. Kshemkalyani and Mukesh Singhal. *Distributed Computing: Principles, Algorithms, and Systems.* Cambridge University Press, 2011.

[KSM+10] Cheolgi Kim, Mu Sun, Sibin Mohan, Heechul Yun, Lui Sha, and Tarek F. Abdelzaher. A framework for the safe interoperability of medical devices in the presence of network failures. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 149–158, 2010.

[KSYS10a] Cheolgi Kim, Mu Sun, Heechul Yun, and Lui Sha. A medical device safety supervision over wireless. *Autonomic Systems*, pages 21–40, 2010.

[KSYS10b] Cheolgi Kim, Mu Sun, Heechul Yun, and Lui Sha. A medical device safety supervision over wireless. *Autonomic Systems*, pages 21–40, 2010.

[L+06] Insup Lee et al. High-confidence medical device software and systems. *Computer*, 39(4):33–38, 2006.

[L+07] Guo-Ping Liu et al. Networked predictive control of systems with random network delays in both forward and feedback channels. *IEEE Transactions on Industrial Electronics*, 54:1282–1297, 2007.

[L+12a] Insup Lee et al. Challenges and research directions in medical cyber-physical systems. *Proceedings of the IEEE*, 100(1):75–90, 2012.

[L+12b] Tao Li et al. From offline toward real-time: A hybrid systems model checking and cps co-design approach for medical device plug-and-play (mdpnp). In *Proc. of ICCPS*, pages 13–22, 2012.

[L+13] Bo Li et al. Realistic case studies of wireless structural control. In *Proceedings of ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'13)*, 2013.

[Lat07] Robert J. Latino. Briefings on patient safety. *Patient Safety Monitor*, pages 6–12, nov 2007.

[LC13] Tao Li and Jiannong Cao. Reliable closed-loop control in medical cyber-physical systems over wireless ad-hoc networks. In *Proceedings of 22nd International Conference on Computer Communications and Networks (ICCCN)*, 2013.

[LG04] Xiangheng Liu and Andrea Goldsmith. Wireless network design for distributed control. In *IEEE Conference on Decision and Control*, 2004.

[LMN07] K. G. Larsen, M. Mikucionis, and B. Nielsen. *Uppaal tron user manual*, 2007.

[LNKM11] Chen Li, Masao Nagasaki, Chuan Hock Koh, and Satoru Miyano. Online model checking approach based parameter estimation to a neuronal fate decision simulation model in caenorhabditis elegans with hybrid functional petri net with extension. *Mol. BioSyst*, 7:1576–1592, 2011.

[LPY97] Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1:134–152, 1997.

[LS10] Insup Lee and Oleg Sokolsky. Medical cyber physical systems. In *ACM/IEEE Design Automation Conference (DAC)*, pages 743–748, 2010.

[LSD+02] Hui Lei, Daby M. Sow, II John S. Davis, Guruduth Banavar, and Maria R. Ebling. The design and applications of a context service. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6:45–55, 2002.

[LY05] Fu-Chun Liu and Yu Yao. Modeling and analysis of networked control systems using hidden markov models. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, pages 928–931, 2005.

[Lyg04] John Lygeros. Lecture notes on hybrid systems. `http://robotics.eecs.berkeley.edu/~sastry/ee291e/lygeros.pdf`, 2004.

[M+03] M. J. Moran et al. *Fundamentals of Engineering Thermodynamics*. Wiley, 2003.

[M+09] Brian Meissner et al. The rate and costs attributable to intravenous patient-controlled analgesia errors. *Hosp. Pharm.*, 44(4):312–324, 2009.

[MA04] Luis A. Montestruque and Panos Antsaklis. Stability of model-based networked control systems with time-varying transmission times. *IEEE Transactions on Automatic Control*, 49:1562–1572, 2004.

[Mac01] P. E. Macintyre. Safety and efficacy of patient-controlled analgesia. *Br. J. Anaesth.*, 87(1):36–46, 2001.

[MBS07] J. Xavier Mazoit, K. Butscher, and K. Samii. Morphine in postoperative patients: Pharmacokinetics and pharmacodynamics of metabolites. *Anesthesia and Analgesia*, 105(1):70–78, 2007.

[Med04] Medical Device Plug-and-Play Interoperability program. `http://www.mdpnp.org`, 2004.

[Mil91] D.L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39:1482–1493, 1991.

[Nan08] Nanotron Technologies GmbH. *nanoLOC TRX Transceiver (NA5TR1) Datasheet*, Apr. 2008. Version 2.0.

[Non] Nonin Medical, Inc. Nonin 9843 Pulse Oximeter and CO2 Detector. `http://www.nonin.com/documents/IFUManuals/7409-001-03%20ENG.pdf`.

[Non08] Nonin Medical, Inc. Nonin Model 4100 Bluetooth Pulse Oximeter. `www.nonin.com/documents/4100%20Specifications.pdf`, 2008.

[OHG02] Nuria Oliver, Eric Horvitz, and Ashutosh Garg. Layered representations for human activity recognition. In *Proceedings of Fourth IEEE International Conference on Multimodal Interfaces*, pages 3–8, 2002.

[P+05] Amir Padovitz et al. An approach to data fusion for context awareness. *Lecture Notes in Computer Science*, 3554:353–367, 2005.

[PMS+12] Miroslav Pajic, Rahul Mangharam, Oleg Sokolsky, David Arney, Julian M. Goldman, and Insup Lee. Model-driven safety analysis of closed-loop medical systems. *IEEE Transactions on Industrial Informatics*, 2012.

[Q+09] Zhengwei Qi et al. A hybrid model checking and runtime monitoring method for C++ web services. In *Fifth International Joint Conference on INC, IMS and IDC*, pages 745–750, 2009.

[RAMC04a] Anand Ranganathan, Jalal Al-Muhtadi, and Roy H. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, 3:62–70, 2004.

[RAMC+04b] Anand Ranganathan, Jalal Al-Muhtadi, Shiva Chetan, Roy Campbell, and M. Dennis Mickunas. Middlewhere: a middleware for location awareness in ubiquitous computing applications. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 397–416, 2004.

[RC03] Anand Ranganathan and Roy H. Campbell. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, 7:353–364, 2003.

[RKT07] L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2462–2467, 2007.

[RLSS10] Ragunathan (Raj) Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736, 2010.

[RMJ+11] Nirmalya Roy, Archan Misra, Christine Julien, Sajal K. Das, and Jit Biswas. An energy-efficient quality adaptive framework for multi-modal sensor context recognition. In *Proceedings of IEEE PerCom*, pages 63– 73, 2011.

[RR96] Olivier Roux and Vlad Rusu. Uniformity for the decidability of hybrid automata. *Static Analysis*, 1145:301–316, 1996.

[S+09] G. Sauterand et al. Lightweight hybrid model checking facilitating online prediction of temporal properties. In *Proceedings of the 21st Nordic Workshop on Programming Theory*, pages 20–22, Kgs. Lyngby, Denmark, 2009.

[SGLW08] Lui Sha, Sathish Gopalakrishnan, Xue Liu, and Qixin Wang. Cyber-physical systems: A new frontier. In *IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing*, pages 1–9, 2008.

[Sin93] M. Singhal. A taxonomy of distributed mutual exclusion. *Journal of Parallel and Distributed Computing*, 18:94–101, 1993.

[SM03] Debashis Saha and Amitava Mukherjee. Pervasive computing: a paradigm for the 21st century. *IEEE Computer*, 36:25–31, 2003.

[SRA04] K. Sen, G. Rosu, and G. Agha. Online efficient predictive safety analysis of multithreaded programs. In *Proceedings of 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, pages 123–138, Barcelona, Spain, 2004.

[T+13] Feng Tan et al. Guaranteeing proper-temporal-embedding safety rules in wireless cps: A hybrid formal modeling approach. In *Proceedings of 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 1–12, 2013.

[Tab09] Paulo Tabuada. *Verification and Control of Hybrid Systems*. Springer, 2009.

[THM05] Mohan Manubhai Trivedi, Kohsia Samuel Huang, and Ivana Mikić. Dynamic context capture and distributed video arrays for intelligent spaces. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35:145–163, 2005.

[US 10] US FDA, Center for Devices and Radiological Health. *White paper: Infusion pump improvement initiative*, April 2010.

[W+07] Q. Wang et al. Building robust wireless LAN for industrial control with the DSSS-CDMA cell phone network paradigm. *IEEE Trans. on Mobile Computing*, 6(6):706–719, June 2007.

[W+11] Y. Wang et al. WiCop: Engineering WiFi temporal white-spaces for safe operations of wireless body area networks in medical applications. *Proceedings of RTSS'11*, pages 170–179, 2011.

[Wu03] Huadong Wu. *Sensor data fusion for context-aware computing using dempster-shafer theory.* PhD thesis, The Robotics Institute, Carnegie Mellon University, 2003.

[X+06] Yuanqing Xia et al. H_∞ control for networked control systems in presence of random network delay and data dropout. In *Proceedings of Chinese Control Conference*, pages 2030–2034, 2006.

[XC05] Chang Xu and S. C. Cheung. Inconsistency detection and resolution for context-aware middleware support. In *Proc. of ESEC/FSE-13*, pages 336–345, 2005.

[XTLS07] Feng Xia, Yu-Chu Tian, Yanjun Li, and Youxian Sung. Wireless sensor/actuator network design for mobile control applications. *Sensors*, 7:2157–2173, 2007.

[Yan06] T.C. Yang. Networked control system: a brief survey. In *IEE Proceedings of Control Theory and Applications*, volume 153, 2006.

[YHL05] Dong Yue, Qing-Long Han, and James Lam. Network-based robust H_∞ control of systems with uncertainty. *Automatica*, 41:999–1007, 2005.

[ZCHG13] Daqiang Zhang, Min Chen, Hongyu Huang, and Minyi Guo. Decentralized checking of context inconsistency in pervasive computing environments. *The Journal of Supercomputing*, 64:256–273, 2013.

[ZSCH05] Liqian Zhang, Yang Shi, Tongwen Chen, and Biao Huang. A new method for stabilization of networked control systems with random delays. *IEEE Transactions on Automatic Control*, 50:1177–1181, 2005.

[ZY07] Wen-An Zhang and Li Yu. Output feedback stabilization of networked control systems with packet dropouts. *IEEE Transactions on Automatic Control*, 52:1705–1710, 2007.