Evidence-based Development of Trustworthy Mobile Medical Apps

by

Priyanka Bagade

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2015 by the
Graduate Supervisory Committee:

Sandeep K. S. Gupta, Chair
Carole-Jean Wu
Adam Doupé
Yi Zhang

ARIZONA STATE UNIVERSITY

December 2015

ABSTRACT

Widespread adoption of smartphone based Mobile Medical Apps (MMAs) is opening new avenues for innovation, bringing MMAs to the forefront of low cost healthcare delivery. These apps often control human physiology and work on sensitive data. Thus it is necessary to have evidences of their trustworthiness i.e. maintaining privacy of health data, long term operation of wearable sensors and ensuring no harm to the user before actual marketing. Traditionally, clinical studies are used to validate the trustworthiness of medical systems. However, they can take long time and could potentially harm the user. Such evidences can be generated using simulations and mathematical analysis. These methods involve estimating the MMA interactions with human physiology. However, the nonlinear nature of human physiology makes the estimation challenging.

This research analyzes and develops MMA software while considering its interactions with human physiology to assure trustworthiness. A novel app development methodology is used to objectively evaluate trustworthiness of a MMA by generating evidences using automatic techniques. It involves developing the Health-Dev β tool to generate a) evidences of trustworthiness of MMAs and b) requirements assured code generation for vulnerable components of the MMA without hindering the app development process. In this method, all requests from MMAs pass through a trustworthy entity, Trustworthy Data Manager which checks if the app request satisfies the MMA requirements. This method is intended to expedite the design to marketing process of MMAs. The objectives of this research is to develop models, tools and theory for evidence generation and can be divided into the following themes:

- Sustainable design configuration estimation of MMAs: Developing an optimization framework which can generate sustainable and safe sensor configuration while considering interactions of the MMA with the environment.

- Evidence generation using simulation and formal methods: Developing models and tools to verify safety properties of the MMA design to ensure no harm to the human physiology.

- Automatic code generation for MMAs: Investigating methods for automatically generating trustworthy software for vulnerable components of a MMA and evidences.

- Performance analysis of trustworthy data manager: Evaluating response time performance of trustworthy data manager under interactions from non-MMA smartphone apps.

DEDICATION

To my daughter, Prisha.

ACKNOWLEDGMENTS

First of all, I would like to express deepest gratitude to my supervisor, Prof. Sandeep K. S. Gupta for his continuous guidance in my whole research without which this research would not have been possible. His pragmatism, patience, encouragement and strategic insights were key motivations throughout the graduate studies.

I am greatly indebted to Dr. Ayan Banerjee who has been my mentor throughout my Ph.D. Technical discussions with him have always been educationally enriching. I would also like to thank my committee members Prof. Carole-Jean Wu, Prof. Adam Doupe, and Dr. Yi Zhang. I also appreciate support of all my previous and current lab mates Madhurima, Zahra, Vinaya, Joseph, Sunit, Kusha, Prajwal, Imane, Javad, Conner, Babak, Jong, Srija and Subhashish for keeping the lab environment enjoyable and stress-free.

I am grateful to the financial support from national science foundation, CNS grant #1218505, IIS grant #1116385 and national institutes of health #EB019202 to carry out this research. I would also like to thank FDA to provide me an opportunity to conduct part of research at their facility in Maryland.

Finally I would like to thank my daughter Prisha, husband - Nilesh, my parents, in-laws, brother - Vishal, and friends Kshitija, Pradnya and Sudhanshu for continuous moral support during the whole journey of my graduate studies.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER

1 INTRODUCTION 1

1.1 Overview of the Research Approach and Contributions 3

1.2 Research Outcome 7

2 PRELIMINARIES 10

2.1 Mobile Medical Apps Examples 10

2.2 Trustworthy Properties of MMA 14

2.3 MMA Models and Requirements 16

2.3.1 MMA Models 16

2.3.2 MMA Requirements Provided by the Developer 18

3 RELATED WORK 19

3.1 MMA Development Frameworks 20

3.2 Evidence Generation 21

3.3 Techniques to Ensure Trustworthiness of MMA 24

4 SYSTEM MODEL AND TRUSTWORTHY FRAMEWORK FOR MMA ... 26

4.1 System Model 26

4.2 Traditional Android App Architecture 26

4.3 Problem Statement 28

4.4 Proposed Trustworthy Framework for Mobile Medical Apps 29

4.4.1 Change in App Architecture 31

4.4.2 Runtime Validation of TDM 32

4.4.3 Artificial Pancreas MMA Working with TDM 36

CHAPTER Page

5 EVIDENCE GENERATION: MMA SUSTAINABLE DESIGN 37

5.1 Safe and Sustainable MMA Configuration Optimization 37

5.1.1 Motivating Examples 37

5.1.2 Problem Formulation 39

5.1.3 Validation 42

5.2 Sustainable MMA Configuration Optimization 46

5.2.1 Validation 49

6 EVIDENCE GENERATION: HYBRID SIMULATION 51

6.1 Challenges of Hybrid Simulation 53

6.2 Practical Need for Processing Events at Accurate Time 55

6.3 *HyrefSim* Simulation Approach 59

6.4 System Model 60

6.4.1 *HyrefSim* Execution Model and Implementation 62

6.5 Estimation of Physical Event Time 64

6.5.1 Loss of Events 65

6.5.2 Clustering of Events 66

6.5.3 Simulator Progress 69

6.5.4 Error Bound of Simulation 69

6.6 Simulation Algorithm 70

6.6.1 Simulator Speedup 71

6.7 Usage and Performance Analysis of *HyrefSim* 73

6.7.1 Example 1: Blood Glucose Control using Infusion Pump 74

6.7.2 Example 2: Body Sensor Networks 79

6.8 Hardware Assisted MMA Physical System Emulation 83

7 EVIDENCE GENERATION: MMA DESIGN VERIFICATION - THEORETICAL APPROACH 88
7.1 Introduction 88
7.2 Model Checking using Reachability Analysis 88
7.3 Artificial Pancreas Cyber Physical Control System 94
7.4 Related Works 95
7.5 Our Approach 95
7.6 Preliminaries 98
7.7 The Exponential Box Spline and its Properties 99
7.8 Smooth and Compactly Supported Non-Linear Hybrid Automata 105
7.9 Reachability Analysis 107
7.9.1 Estimation of Trajectory 107
7.9.2 Estimating Discrete Transitions 113
7.10 Case-study: Artificial Pancreas 113
8 IMPLEMENTATION - AUTOMATED CODE GENERATION FOR MMAS 118
8.1 Health-Dev β Tool Architecture 118
8.2 Security-enabled MMA Implementation and TDM Generation 120
8.2.1 TDM App Generation 122
8.3 Performance Analysis 123
8.3.1 Quality of the Generated Code 123
8.3.2 Overhead Analysis of the Proposed App Model 126
8.3.3 MMA Development Effort Reduction using Health-Dev β 131
9 CONCLUSION AND FUTURE RESEARCH DIRECTIONS 132
REFERENCES 136

LIST OF TABLES

Table Page

5.1 Power Consumption of Sensor Query Task. 43

5.2 Radio Power Consumption Results with Plaintext Packets and Encrypted Packets. 44

6.1 Summary of Existing CPS Simulation Tools. 53

6.2 Instance of Hybrid Simulator Definition for Sensor Network. 63

7.1 Summary of Related Work on Reachability Analysis for Nonlinear Systems. 96

LIST OF FIGURES

Figure Page

1.1 Techniques used in MMA Development: Traditional and New Approach. . . 4

1.2 Trustworthy MMA Development. 8

2.1 Hybrid Automata Model for Artificial Pancreas MMA. 17

2.2 Finite State Machine Model for MMAs. 18

4.1 System model for Mobile Medical Applications. 26

4.2 Mobile Medical Application. 27

4.3 Traditional App Execution Model. 28

4.4 Conceptual Operating Model for MMAs. 29

4.5 Proposed App Execution with TDM. 30

4.6 Manifestation of the TDM Runtime Environment for Validating the Apps Operation Against Requirements. 33

4.7 App Request Processing by TDM for Artificial Pancreas MMA. 35

5.1 Skin Temperature Rise Depending on Sensor Location. 38

5.2 Cluster Scheduling Schemes for Implanted Sensors. 45

5.3 Infusion Pump Parameters for Avoiding Hypoglycemia. 46

5.4 Experimental Results for Sustainable Sensor Design. 50

6.1 Distributed Cyber-physical Systems with Computing Units Operating in Close Loop with Physical systems. 52

6.2 Difference in Temperature Profile for *FixSim* Type Simulation and *AccuSim* with Exact Event Times. 55

6.3 Difference in Energy Consumption of Sensors for *FixSim* Type Simulation and *AccuSim* with Exact Event Times. 57

6.4 False Clustering of Events. 58

Figure Page

6.5 Error in Estimation of Stored Energy by *Fixsim* increases with Increase in Simulation Time without Bounds. 59

6.6 The *HyrefSim* Execution Model. 64

6.7 Event Loss (Case a), Event Loss and False Clustering (Case b) and Zeno Behavior (Case C) in Physical Systems. The Top Row Shows an Ideal Simulator with Very Small Time Step. The Bottom Row is *Fixsim* with a Finite Time Step. 67

6.8 Blood Glucose Level Estimation using *HyrefSim* and *FixSim*. 76

6.9 Average Delay along with Error Bar in Processing Events by *HyrefSim* and *FixSim* Over Time with Respect to *AccuSim*. 77

6.10 Number of Events that are Processed by *Accusim* but Lost by *Hyrefsim* and *Fixsim* Over Time. 78

6.11 Variation of Simulation Error of *FixSim* with Respect to *HyrefSim* v.s. Simulation Length. 79

6.12 Simulation Error i.e., Difference in Total Amount of Insulin Delivery with Respect to *AccuSim* v.s. Simulation Length for *HyrefSim* and *FixSim*. 80

6.13 Variation in Event Queue Length with Respect to Time for *HyrefSim* and *FixSim*. 81

6.14 Simulation Execution Time of *HyrefSim* and *FixSim* for Different Real Time Simulation Lengths. 82

6.15 BSN Simulation Results with Different Networking Algorithms. 83

6.16 Number of Events Lost by *HyrefSim* and *FixSim*. 84

6.17 Average Delay in Event Processing for *HyrefSim* and *FixSim*. 84

Figure Page

6.18 Average Number of Events in the Event Queue for *HyrefSim* and *FixSim* Over Different Simulation Runtimes. 85

6.19 Resistor Grid and Diffusor Network Circuits for Representing Analog Behavior of Physical System. 86

6.20 50X50 Resistor Grid Circuit Output Plot 87

7.1 Comparison of Linear Interpolation and EB Spline based Image Computation Techniques. 93

7.2 Approach for Reach Set Computation of Nonlinear Hybrid System. 95

7.3 Support Zonotopes for Integer Translates of Exponential Box Splines 101

7.4 Approach for Estimation of the Trajectory of SCHA. 108

7.5 The Evolution of Images of Reach Set Over Time for the Artificial Pancreas SCHA. 116

8.1 Health-Dev β Tool. 118

8.2 CodePro Analytix Audit Report. 124

8.3 Experimental Setup for TDM Scalability. 127

8.4 Response Time of TDM with Increased Requests. 128

8.5 Response Time of TDM with Increased Request Payload Size. 129

8.6 Experimental Setup to Check Impact of TDM on Performance of Other App. 130

8.7 Impact of TDM on Performance of Non-health App. 130

Chapter 1

INTRODUCTION

Use of mobile medical apps (MMAs) in healthcare is becoming prevalent as they provide low cost healthcare delivery [1]. These apps often control human physiology by acting as a controller to the medical devices such as artificial pancreas which automatically controls the blood glucose level by infusing insulin, ventilators to support breathing action by blowing air in and out from the respiratory system. They also provide diagnosis using physiological data collected by implanted sensors. Such close interaction of MMAs with human physiology make them critical. Thus, despite of their ability to provide low cost healthcare delivery, getting these apps in the market is a tedious process e.g. design for the automated artificial pancreas came in 1999s [2]. However the actual app is marketed in 2014, which is a first generation app and capable of only predicting hypoglycemia (low blood glucose level)[3]. Due to involvement of human physiology, MMAs are technically challenging to implement and get market approval which requires to establish trustworthiness of these apps. Trustworthiness of MMAs can be viewed as having properties such as interaction safety, communication and storage security and wearables sustainability. Assuring these trustworthy properties in MMAs is challenging due to difficulty in extracting the interactions between MMAs and human physiology, which leads to lack of mathematical structure to objectively provide evidences of the MMAs. The evidence can be defined as a set of observations on certain properties of MMA software and its effects on the human body, generated through the usage of scientifically sound experiments, clinical studies, simulations and mathematical proofs. Traditionally, experiments and clinical studies are used to generate evidences of trustworthiness of MMAs. However they can be difficult and costly to be comprehensive due to requirement of approval from institutional review board

(IRB), patient availability. These techniques can also potentially harm the user and may take long time to establish their trustworthiness. Simulation and mathematical proofs can generate evidences before the actual use of the MMAs on the user. However, developing such techniques might increase development time and may require higher skills set from the developer. Thus, this research implements **tools and techniques to enable evidence-based MMA design and development of trustworthy MMAs.**

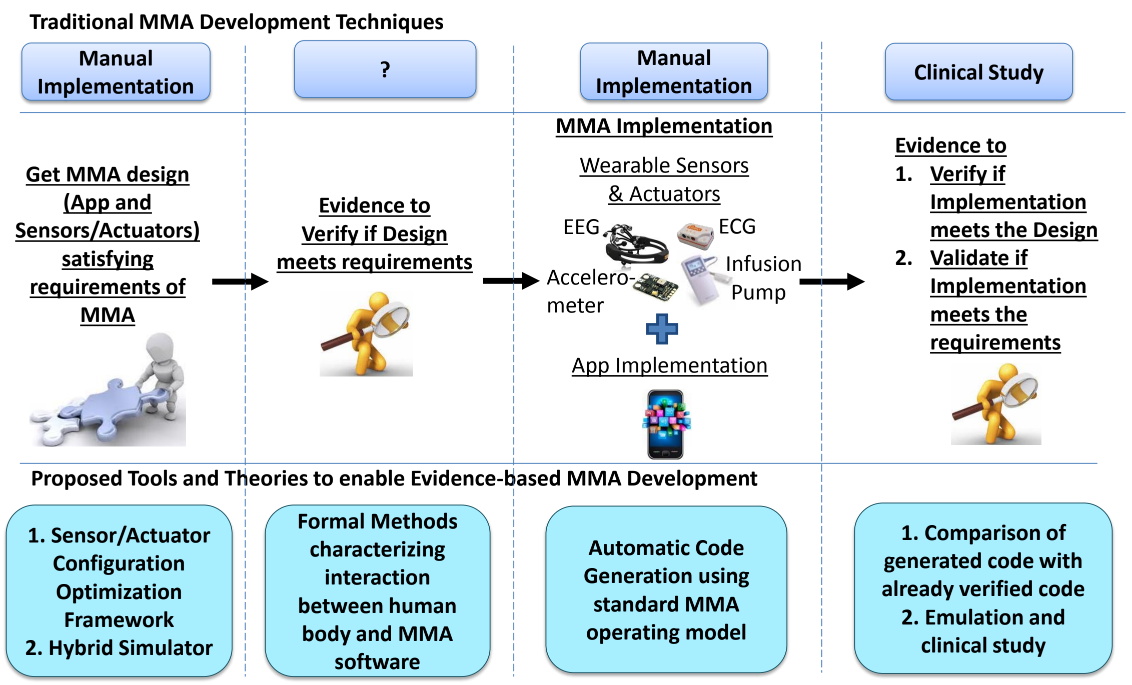
Simulating the MMA involves estimating its interaction with human physiology. It uses models of MMA software and human physiology to appraise the values of continuous variables of the system over certain time. These values are then used to estimate resource requirements, cost and verify the design against requirements which can provide evidence for the trustworthiness of MMA design. The nonlinear and spatio-temporal nature of the human physiology makes its models difficult to simulate. Further, the events generated in controller software and human physiology changes the system configurations. Thus inaccurate handling of such events might end up in getting sub-optimal design. Accurate handling of the events require infinitely small time step in current simulators, which leads to increased simulation time. Thus to overcome these challenges in accurately simulating MMAs, I implemented a simulator with a time refinement approach which predicts the event timing and accordingly modifies the time step to handle the event.

To obtain the evidence of the trustworthy properties of the MMA using mathematical structures, a state based abstraction of the MMA software is used. The discrete states of the abstraction represent the operating conditions of the software, while the transitions in the state are governed by changes in human physiology variables. An execution of this state based model results in valuations of the system variables. If the set of all possible valuations intersects with the unsafe set, the system will not satisfy the trustworthy properties. This analysis is typically called as reachability analysis. It requires solving non-linear, spatio-temporal differential equations of human physiology to accurately estimate values of system variables. The existing solutions for reachability analysis are for linear systems. Thus they can not be directly applicable to the human physiology based system which is non-linear in nature. Thus, this research has focused on developing reachability analysis of nonlinear systems using exponential box splines, which are traditionally being used for curve fitting in geometric modeling.

These automatic techniques can be used to develop trustworthy MMAs as well as to generate evidences. Such techniques can be used, if the implementation follows a standard model. However, the standard model can lead to generating similar types of MMAs. This might limit the functionalities of the smartphone that can be exploited by the developers e.g. different graphical user interfaces, data displaying techniques. Thus, this research considers that the app developer should be responsible for developing GUI and processing algorithms on smartphone end and the critical interfaces such as data communication, sensor/actuator code will be generated and handled by the trustworthy entity. *Thus, it is hypothesized that trustworthy mobile medical applications should have an operating model, where every instance of data communication to the sensor, data storage in the smartphone, control inputs to the actuator, interaction with the user, data communication to the EHR and even EHR access by physician has to pass through a certification entity, Trustworthy Data Manager (TDM).*

1.1 Overview of the Research Approach and Contributions

To ensure the trustworthiness of MMAs, the MMA software development process should provide evidences at every step of its life cycle [4]. Such evidence-based MMA development process (Figure 1.1) can be divided into four parts, a)MMA Design, b)MMA design verification, c)MMA Implementation, and d)verification and validation of implemented MMA. The process starts with obtaining MMA design which should satisfy trustworthy requirements of MMA. It is then followed by providing evidence to check if the obtained



Traditional MMA Development Techniques
Manual Implementation
?
Manual Implementation
Clinical Study
Get MMA design (App and Sensors/Actuators) satisfying requirements of MMA
Evidence to Verify if Design meets requirements
MMA Implementation
Wearable Sensors & Actuators
EEG
ECG
Accelero-meter
Infusion Pump
App Implementation
Evidence to
1. Verify if Implementation meets the Design
2. Validate if Implementation meets the requirements
Proposed Tools and Theories to enable Evidence-based MMA Development
1. Sensor/Actuator Configuration Optimization Framework
2. Hybrid Simulator
Formal Methods characterizing interaction between human body and MMA software
Automatic Code Generation using standard MMA operating model
1. Comparison of generated code with already verified code
2. Emulation and clinical study

Figure 1.1: Techniques used in MMA Development: Traditional and New Approach.

design meets the trustworthy requirements of MMAs. This verified design is then used for implementation which includes sensor code and smartphone app development. Two types of evidences are required for implemented code to verify if the implementation follows the design and to validate if the implementation ensures trustworthy requirements of MMAs. Traditional app development method uses manual implementation method while it does not provide any evidence for correctness of the design. The implemented code is generally validated using experiments such as clinical study which is time consuming and might lead to user safety hazards due to wrong design or implementation. This research fills the gap by proposing models, tools and methodologies to generate evidences with theoretical guarantees for ensuring correct working of MMAs before their experimental studies.

Research Contributions: The contributions are discussed as follows:

- *MMA Design Optimization*: MMAs interact with environment through wearable devices, e.g. wearable devices scavenge energy from unpredictable sources such as human heat, sun light etc. Also as they are implanted on human body, they interact with human physiology by controlling it using actuators or by heating the human skin at the sensor worn spot. Thus to assure sustainable and safe working of the MMAs, the design of the wearable devices should consider the interaction effects. Traditional MMA design techniques fail to consider these interactions, which can result into unreliable design. Thus, this research tries to answer following research question:

How to incorporate interactions of MMA with its environment while configuring the sensor/actuator?

Contribution: The research implements an optimization framework to get sensor configuration that supports privacy ensured continuous monitoring powered by energy scavenged from human body or sun light and does not cause any harm to the human physiology (Chapter 5). It takes hardware and software requirements of the sensor/actuator to generate the optimal design. The framework uses hybrid simulator to simulate human physiology to obtain the safe design. The research further extends the hybrid simulator to simulate events from human physiology as well as from controller app (Chapter 6). This whole system simulation can give better idea about the MMA design in terms of resource allocation requirements.

- *Design Verification*: This step provides an evidence for the design obtained in previous step. The software design verification usually involves logical reasoning by giving mathematical proof of the correctness of the design. In case of MMAs, as they directly interact with human physiology, considering the effect of MMA operation on human physiology is the important part in the reasoning process. Traditional MMA development techniques do not provide any of such method for the reasoning. Thus, this research focuses on:

How to generate evidence for the MMA design using models of human physiology?

Contribution: The technique incorporates formal methods for verification of the design (Chapter 7). In this technique the combined analysis of continuous dynamics of human physiology and discrete dynamics of the MMA controller software is done using reachability analysis. Reachability analysis computes the values of continuous variables of the system with a given controller at any point in time. These values are further used to check instability in controller design. The technique uses exponential box splines to represent the trajectory of these continuous variables of nonlinear systems. Exponential box splines are being traditionally used for curve fitting in geometric modeling. Their curve fitting property is used to express the nonlinear variation of the continuous variables in the system.

- *MMA Implementation*: Traditional app development process uses manual implementation which might result into software bugs. These errors can lead to safety and security issues e.g. bug in the infusion pump controller software can lead to excess drug infusion in the blood which can lead to distress, error in security algorithm can cause leakage of health data. In order to avoid these issues the app software should be developed in trustworthy environment. Thus, this research focuses on:

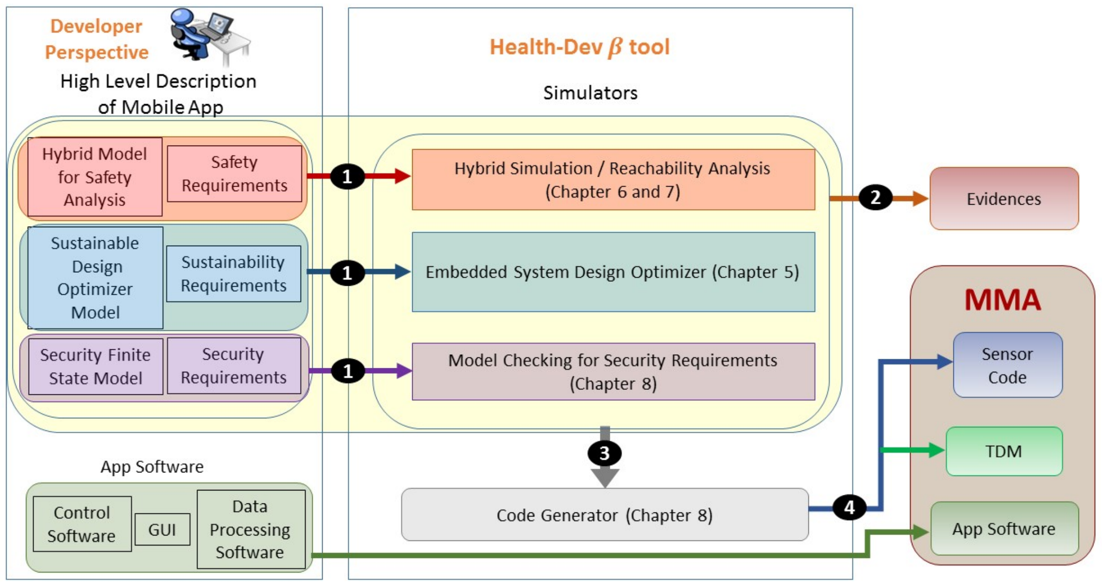
  How to reduce manual implementation errors in MMAs?

  Contribution: A medical app development framework, Health-Dev β, is developed that enables the automated generation of TDM app, sensor and actuator code, and communication code that meets the security and safety requirements (Chapter 8). The framework generates customizable code for sensor communication, data storage in smartphone and communication to the cloud from high level descriptions using a parameterized code frame data-base. The visualization software (GUI) can be independently developed by a manufacturer. Health-Dev β tool is equipped with a data-base of security enabled communication protocol code frames such as password based and physiological value based security.

- *Verification and Validation of implemented code*: To check if the implemented code conforms to the requirements and design of the MMA, the use of clinical studies in traditional app development methods can be hazardous to human physiology. Thus this research compares the implemented code with already verified code such as BSNBench [5] to verify if the implemented code satisfies the MMA design. Further, emulation platform is used to validate the MMA requirements of safety from MMA operation before actual use of the app. It uses field-programmable analog arrays (FPAAs) to emulate the interaction between MMA software and human physiology (Chapter 6 Section 6.8).

1.2 Research Outcome

The research outcome of this dissertation is as shown in Figure 1.2 which represents the trustworthy development of MMA along with evidence generation for their regulation. In this research, it is hypothesized that a trustworthy data manager (TDM) will provide interfaces to MMAs and validate if the requirements are met or not at runtime (discussed in Chapter 4). Given such TDM, the developer has to follow an app model where he is responsible for providing the smartphone app software, models for safety sustainability and security checking along with the requirements. Now the Health Dev β tool, an automated code generator that I have developed involves simulators. The models and requirements will be input to this simulator. The simulator uses hybrid simulator or reachability analysis for safety verification, embedded system design optimizer for sustainability requirements and a model checking algorithm for security requirements. Through these simulation it



Developer Perspective
High Level Description of Mobile App
Hybrid Model for Safety Analysis
Safety Requirements
Sustainable Design Optimizer Model
Sustainability Requirements
Security Finite State Model
Security Requirements
App Software
Control Software
GUI
Data Processing Software
Health-Dev β tool
Simulators
Hybrid Simulation / Reachability Analysis (Chapter 6 and 7)
Embedded System Design Optimizer (Chapter 5)
Model Checking for Security Requirements (Chapter 8)
Code Generator (Chapter 8)
1
2
3
4
Evidences
MMA
Sensor Code
TDM
App Software

Figure 1.2: Trustworthy MMA Development.

will generate evidences of trustworthiness which will be used for MMA regulation. The simulators to generate evidences are discussed as follows:

Energy sustainability checking : The developer has a choice to specify the energy budgets of the application in the form of overall caps as requirements. The Health-dev β code database [6] is supplied with energy consumption values for code snippets on different platforms. While selecting the appropriate code frames in the generator the energy cap is taken as one of the deciding factors. Such selection can give rise to complex optimization problems which can be solved by developing appropriate models of energy consumption of the respective components as shown in our previous work [7] and discussed in Chapter 5.

Safety verification: Safety verification can be used to prove the design safety using formal modeling and hybrid simulation techniques. The safety verification ensures the side-effects of interactions between human physiology and sensors within accepted limit. These interactions are typically governed by smartphone control algorithm when sensor acts as an actuator. In such a scenario, developer provides the hybrid automata which embeds the control algorithm while specifying high level design. Reachability analysis (discussed in Chapter 7) or hybrid simulation (discussed in Chapter 6) is then performed on the specified hybrid automata while considering optimized time constrained designs as initial set.

Security checking: Security properties can be expressed as a part of the high level specification of the TDM application. The finite state automata representing data access patterns should succinctly describe the desired operation of the app. Simulation of the FSA will show whether in any state the security properties are violated (discussed in Chapter 2 and Chapter 8).

After that all these models, requirements and simulators will be input to an automatic code generator which will generate a TDM consisting of all these along with codes for the sensors, wearables. This TDM, sensor codes along with the app software forms mobile medical app or MMA.

Chapter 2

PRELIMINARIES

2.1 Mobile Medical Apps Examples

MMAs communicate with physiological sensors implanted on user's body and use the collected physiological data for further processing such as diagnosis, controlling actuator operation, displaying data in user readable format. The apps also communicate with a cloud server to upload the health data to an electronic health record (EHR) which is a health data storage.

Since, MMAs control medical grade actuators and monitor health, they need to meet safety, security and sustainability requirements. Indeed, Food and Drug's Administration (FDA) classify smartphone health apps as medical devices. According to the definition of the MMAs published by FDA in September 2012, the healthcare apps can be classified into three types, a) displaying app - which displays the sensor data in graphical format, b) diagnostic app - which processes the collected physiological data and provides diagnostic feedback to the user, and c) controlling app which controls the actions of the body implanted actuator device depending the sensed physiological signals.

The techniques investigated in this research have been applied to following MMA examples to validate the proposed approach. The example MMAs are classified according to the FDA definition of smartphone medical apps.

Example 1 PETPeeves - Display app: *PETPeeves is an app aimed to help users alter their lifestyle to be healthier by presenting the user with a virtual pet whose mood changes based on the amount of exercise the user performs a week. The app uses accelerometer and*

ECG sensor to monitor the user's heart rate and accordingly calculates calories burned during exercise.

Example 2 BrainHealth - Diagnostic app: *The app consists of three activities-focus, mood change, and relaxation. Focus is aimed towards users who suffer from learning disabilities and need a boost in mental performance, motivation, and focus. Mood change is aimed towards users whom are not satisfied with their mood and want to achieve a more positive mood. Lastly, relaxation is aimed at any user who wants to learn how to relax in any situation. The app consists of a visual feedback system which uses a glob of goop. When the user is performing well in an activity, the glob is very structured and dense, however when the user's performance degrades, the glob breaks apart and the goop drifts to the edges of the screen. Neurofeedback or phsychostimulants are found to be an effective method for encouraging healthy behavior [8]. PETPeeves app uses this neurofeedback as additional input to give reward points to the user which acts as the bonus to pet's mood modifier.*

Example 3 Artificial Pancreas/Infusion Pump App - Controller App: *Automated control of blood glucose levels in human are often obtained using artificial pancreas. Artificial pancreas are distributed systems consisting of an infusion pump, glucosemeter, and a controller implemented in a mobile device such as smartphone. These distributed components are networked through the wireless communication channel and operate in a close loop to keep the drug concentration in the human blood within recommended limits. The different components of the automated control system may have skews in the clock rates as well as data transfer rates. This results in transport delays in the sensed values of glucosemeter and actuation delay in infusion. Thus, the continuous dynamic equation that represents the blood glucose level in the blood sensed by the glucosemeter due to infusion from the pump is given by equation 6.15.*

$$\begin{aligned} \dot{y_1} &= A_p y_1 + B_p \dot{Q} z_2 + B_p u(t - T_i), \\ z_1 &= C_p y_1(t - T_p), \\ \dot{y_2} &= A_s y_2 + B_s \dot{Q} z_1, \\ z_2 &= C_s y_2(t - T_r). \end{aligned} \tag{2.1}$$

Here y_1 and y_2 are the state space variables of the equation. y_1 consists of vectors of left heart, lung blood, lung tissue and right heart compartments through which infused drug passes. Newly infused drug merges with recirculated drug from Vessel Rich Group, Muscle, Fat and Residual drug which is represented by y_2 state space variable vectors. A_p, A_s, B_p, $\dot{Q}$, C_s, and C_p are constants. z_1 is the drug concentration in the blood while z_2 is the arterial drug concentration. The initial infusion rate $u = x_0$ is the input to the model and the output is the drug concentration in the blood. The time delays related to the infusion input ($Taya_i$) is due to the delay in actuation, the cardio-pulmonary transport delay T_p and the arterial, capillary and venous transport delays T_r manifest the delay in sensing the blood glucose level by the glucosemeter.

The discrete controller of the infusion pump has five states: a) basal, where infusion rate is I_0 and the blood glucose rate is between 120 ug/dl and 70 ug/dl, b) braking, where infusion rate is a fraction f of I_0, and blood glucose level goes below 70 ug/dl but is still above 20 ug/dl, c) correction bolus, where infusion rate is incremented by I_{cb}, and the blood glucose level is between 120 ug/dl and 180 ug/dl, d) bolus, where the infusion rate is incremented by $I_b > I_{cb}$, and the blood glucose level is above 180 ug/dl, and e) stop, when the infusion is stopped i.e., $I_b = 0$, since the blood glucose level drops below 20 ug/dl.

Example 4 Body sensor Networks: *This example considers a network of implanted sensors, which communicate in a cluster protocol. The implanted sensors form a cluster, where* *a cluster head collects all communication packets from the participating sensors and sends it out to a base station.*

Example 5 Medical Ventilator - Controller App: *Medical ventilators provide critical life support to the patients when they are unable to breathe properly. Due to diseased or injured lungs, respiratory system of the patient stops functioning correctly. In such a scenario, ventilator develops the required pressure for air flow through the patient's respiratory system as well as it supplies air with appropriate amount of oxygen concentration. The percentage of oxygen concentration in the ventilator supplied air is referred as* F_IO_2 *which is basically a fraction of inspired oxygen in the air. Ventilator system calculates* F_IO_2 *by measuring oxygen saturation,* S_PO_2 *, in the blood. Ventilators use non-invasive method to measure* S_PO_2 *using pulse-oximeter. If the blood oxygen saturation level drops below certain level, the patient suffers from hypoxemia and if it goes above the higher limit of target oxygen saturation, the patient suffers from hyperoxia. Ventilator needs to adjust* F_IO_2 *to maintain* S_PO_2 *in desired level. Blood cells are oxygenated in the lungs. Inhaled oxygen and patient physiology decides the oxygen saturation in the blood. Thus, to avoid development of hypoxemia or hyperoxia in the patient improper oxygen saturation, ventilator's control algorithm uses* S_PO_2 *as feedback to determine appropriate value of* F_IO_2*.*

For ventilators, initially the input oxygen level is set to some default value or a value decided by a physician as per the patient need. Then depending on the saturation level of oxygen in the lungs, the input oxygen level should change. To accommodate this control in ventilators, the oxygen level in the lungs is being measured using pulse-oximeter after predefined time interval. The output of pulse-oximeter acts as an input to the ventilator control algorithm. Next input oxygen level is decided depending on this feedback.

The patient model (patient physiology) gives the relation between the input oxygen level and the saturation oxygen level. We have considered the non-linear patient model from [9] *as defined in equation 2.2,*

$$\frac{\Delta S_a O_2(s)}{\Delta F_I O_2} = \frac{G_s G_P e^{-T_d s}}{1 + \tau_p s}, \tag{2.2}$$

Here,

$S_a O_2$: *arterial oxygen saturation*

$F_I O_2$: *input oxygen level*

T_d : *system transport lag (sec)*

G_p : *sensitivity of $P_a O_2$ to $F_I O_2$*

τ_p : *time constant (sec)*

G_s : *linearised sensitivity of the O_2 dissociation curve.*

$S_a O_2$ measured using pulse-oximeter is referred as $S_P O_2$. The patient model in equation 2.2 is in Laplace transform. We are using Hybrid Automata to represent the ventilator system and it can not compute the equations in Laplace transform format. Thus, we have converted it into a differential equation in order to represent continuous dynamics of the system (equation (2.3)).

$$\frac{dS_P O_2}{dt} = G_s G_P F_I O_2 e^{t/T_d}, \tag{2.3}$$

2.2 Trustworthy Properties of MMA

The interactions of MMAs with human physiology might lead to safety issues e.g. brunt skin due to sensor dissipated heat, in case of infusion pump, extra drug infused in patient's body can cause respiratory distress. Security vulnerability can be a major concern due to involvement of health data in wireless communication. Further, as these apps work with low power sensors and actuators which usually scavenge energy from unpredictable sources such as human heat, sunlight, their sustainable working is one of the important aspect of MMA development. While considering the importance of these safety, security and sustainability requirements in assuring trustworthy working of MMA, this research focuses on evaluating these requirements in app development process. These requirements are referred as trustworthy properties of MMAs and defined as:

Safety: In MMAs, safety issues can occur due to device failure, control operation and delay in treatment. This research mainly speculates interaction safety in MMAs that control human physiology through actuators. Considering V_h as the human physiology parameter controlled by MMA, safety can be defined with Equation 2.4 as,

$$f(V_h) \leq T_h \tag{2.4}$$

Here, T_h is the safety threshold value for V_h. $f(V_h)$ can be a blood glucose level in case of infusion pump or oxygen saturation level for ventilators, which should not exceed T_h.

Sustainability: Sustainability in sensors and actuators can be defined with respect to a) reduction in energy usage, b) increase in battery life and, c) operate on harvesting resources. This research considers sustainability as energy neutrality where energy required for sensor or actuator operation is obtained from scavenging source only. Let $P(t)$ be the power required for a sensor operation at time t and $H(t)$ be the storage capacity, then sustainability property can be defined using Equation 2.5.

$$\int_0^t P(t)dt \leq \int_0^t H(t)dt \tag{2.5}$$

Security: In MMAs, security is associated with storage and communication of health data. The aim of this research is to assure secure communication through wireless channel. Communication security is measured in terms of encryption algorithm input parameter size S_m, as in Equation 2.6.

$$min(S_m) \geq S_{Th}, \forall m \epsilon MMAs \tag{2.6}$$

Here, S_{Th} is the input parameter threshold for user selected encryption algorithm. S_m has to be atleast S_{Th} to ensure minimum desired security. S_m can be the size of the prime number in case of RSA protocol or polynomial size or number of chaff points in case of physiological value-based protocol [10].

2.3 MMA Models and Requirements

2.3.1 MMA Models

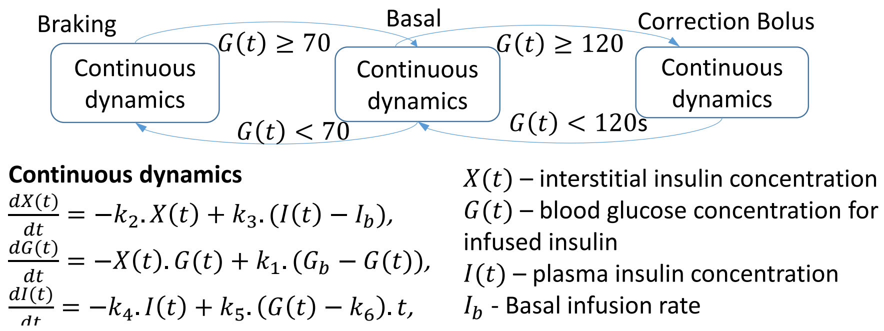
Following models represent high level specification of MMA software. They are used for generating evidences as well checking runtime requirements for the apps.

Hybrid Automata Model for Safety Checking:

Hybrid automata (HA) [11] is a mathematical construct that represents the behavior of the MMA in terms of discrete modes and continuous state variables. MMAs have controller software which has discrete modes. In each mode a certain control decision is evaluated and administered such decisions are often taken based on the values of certain continuous state variables which represent continuous dynamics systems in MMAs which is human physiology. The temporal behavior of MMA is governed by differential equations. Hybrid automata are popularly used to represent the discrete computing models and continuous variables in a single mathematical construct. Hybrid automaton also enables transitions between discrete modes which are decided by associated guard conditions on state variables. Figure 2.1 shows an example hybrid automata model for Artificial Pancreas MMA (Example 3). It has three discrete modes basal, correction bolus, and breaking. Each mode represents different infusion rates and associated continuous dynamics of the physiology. The transition between states is governed by blood glucose level, $G(t)$.

Sensor Design Optimization Formulation:

The low power wearable devices are more prone to safety and sustainability issues than smartphones. Thus, an optimizer is used to find the optimal safe and sustainable design for such devices for a defined time, hardware and software constraints. It simulates continuous dynamics of the human physiology to consider the interaction effect between human body and wearable sensors. Along with the human physiology safety constraints, the design



Braking
Continuous dynamics
$G(t) \geq 70$
Basal
Continuous dynamics
$G(t) \geq 120$
Correction Bolus
Continuous dynamics
$G(t) < 70$
$G(t) < 120$s
Continuous dynamics
$\frac{dX(t)}{dt} = -k_2.X(t) + k_3.(I(t) - I_b),$
$\frac{dG(t)}{dt} = -X(t).G(t) + k_1.(G_b - G(t)),$
$\frac{dI(t)}{dt} = -k_4.I(t) + k_5.(G(t) - k_6).t,$
$X(t)$ – interstitial insulin concentration
$G(t)$ – blood glucose concentration for infused insulin
$I(t)$ – plasma insulin concentration
I_b - Basal infusion rate

Figure 2.1: Hybrid Automata Model for Artificial Pancreas MMA.

optimizer algorithm also includes security and sustainability constraints to find optimal design for a particular device operation period.

The formulation can be for a given hardware and software constraints e.g. for given sensing power, communication power, data processing power, power required for encryption, scavenging source availability and battery capacity, find the data sending frequency when scavenging source is available and not available.

Finite State Machine for Security Checking:

Security properties can be expressed as a part of the high level specification of MMA. The finite state Machine (FSM) representing data access patterns should succinctly describe the desired operation of the app. Simulation of the FSM will show whether in any state the security properties are violated.

Given the FSM model of the PETPeeves application (Example 1) as shown in Figure 2.2, the simulator can check each transition condition and evaluate whether the model satisfies the security requirements. More complicated security rules such as a combination of data storage security and communication security may also be specified using the FSM models as shown for the BrainHealth case (Example 2).

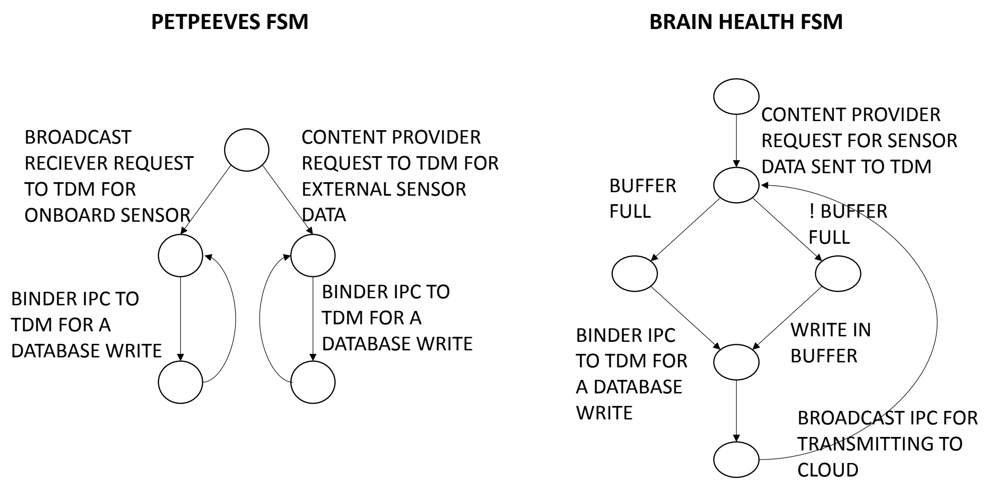

PETPEEVES FSM
BRAIN HEALTH FSM
BROADCAST RECIEVER REQUEST TO TDM FOR ONBOARD SENSOR
CONTENT PROVIDER REQUEST TO TDM FOR EXTERNAL SENSOR DATA
BINDER IPC TO TDM FOR A DATABASE WRITE
BINDER IPC TO TDM FOR A DATABASE WRITE
CONTENT PROVIDER REQUEST FOR SENSOR DATA SENT TO TDM
BUFFER FULL
! BUFFER FULL
BINDER IPC TO TDM FOR A DATABASE WRITE
WRITE IN BUFFER
BROADCAST IPC FOR TRANSMITTING TO CLOUD

Figure 2.2: Finite State Machine Model for MMAs.

2.3.2 MMA Requirements Provided by the Developer

The requirements for MMA can be obtained by consulting with medical professionals, doctors and MMA regulators. These requirements can written in English. There is no automated way to get the requirements in system threshold formats. Thus developer has to convert the requirements in system thresholds so as to use them during evidence generation using models of MMA discussed in Section 2.3.1. These requirements can be the safety regulations of mobile medical apps, the sensor sustainability requirements and the HIPAA security requirements.

Chapter 3

RELATED WORK

Mobile medical apps (MMAs) are involved in critical healthcare operations which might lead to risks or hazards to human physiology. Recently there has been efforts taken to monitor user physiology using MMAs. HeartMApp [12], an MMA, is developed to monitor patients with congestive heart failure condition. It uses ECG sensor to collect heart signal from that it follows up on when the patient is engaged in exercising activities such as deep breathing, walking etc. MT-Diet [13] an MMA, developed towards the aim of reducing obesity by computing calories from food plate using thermal camera and Myo sensors. Further, there are MMAs developed to monitor user's health by prompting user to enter information such as PainGauge [14] requires users to enter pain level every two hours, Mood assessment app [15] asks user to enter the current emotional level on the scale of 1 to 10. These apps send the collected data to cloud to be accessed by doctors for further diagnosis. It might lead to security issues such as tampering of health data or stealing of the data when communicated over wireless channel from sensor or to cloud. They can lead to safety issues by burning the user's skin due to sensor wearing. Also it is required that the sensor should operate for long time to avoid frequent battery charging. This research focuses on mitigating these risks of MMA using automated techniques. The risks addressed in this research are categorized as: a) safety violations, a user is not harmed due to the operation of the smartphone app (the controller app might harm user physiology), b) security violations, user's data is always kept private and free from unauthorized access (unauthorized access to the health data), and c) sustainability violations, lack of availability of apps due to resource constraints such as power, CPU, and memory. These are the trustworthy properties of MMAs which need to be evaluated before the actual use of the app to have dependable healthcare system.

3.1 MMA Development Frameworks

There has been several frameworks proposed for developing MMAs to reduce development time. The existing frameworks for mobile health app development can be classified considering their divergent development strategies as: a) support APIs, b) provide programming abstractions, and c) automated code generation. The frameworks are also evaluated against trustworthy requirements of MMAs. The mobile app development with APIs is popular due its ease of programming. There are frameworks proposed in literature to provide APIs to interface with sensors [16, 17, 18] without satisfying any trustworthy requirements. UPHIAC [19] and PRISM [20] frameworks provide health data security with APIs to interface with smartphone sensors and cloud for data storage. However these frameworks do not check for safety and sustainability requirements of health apps and provide API's to connect to external sensors. Programming abstractions for sensors are being widely used to allow developers to write minimal sensor code to include wearables in health apps [21, 22, 23, 24, 25]. Considering the limited energy availability on wearables, energy efficiency techniques are supported by Reflex [23], LittleRock [24] and Turducken [25]. However these methods require sensor programming from the developer. Thus to reduce health apps development time and to get bug-free code, there are automated sensor code generators proposed in literature [26, 27, 28, 29]. Still they require developers to write code for smartphone apps and its vulnerable components such as interfacing with sensors, actuators and cloud. This led to fully automated code generators for sensors and smartphone apps [30, 31, 32, 33]. However they only consider system safety and do not consider sensor sustainability and data security. Thus, an automated code generator for health apps is proposed which can ensure their trustworthiness without imposing any burden on developer.

Data Sharing with other apps: Secure data communication, sharing data with other apps, storage in cloud is one of the main concerns in MMAs. There has been lot of work done to address issue of data sharing in mobile apps and cloud while considering the pervasive monitoring systems [34, 35, 36, 37, 38]. Open mHealth architecture[39] promotes the use of having single framework which can collect data from various apps used by the patient/user and share with the physician instead of asking each app to upload data separately. It mainly focuses on data transfer from apps to cloud. However it does not check for security vulnerabilities during the data transfer. bHealthy [37] promotes the use data sharing between apps to give better feedback to the user. Mobius[34], a middleware for interfacing with complex data management, supports unified data messaging and abstraction for mobile apps. Another interface (Simba) [35] interfaces the complexities of synchronizing data to the cloud with minimal work from the developer. A Service-Oriented Context-Aware Middleware (SOCAM) has been proposed [36] to enable rapid prototyping of context-aware services in pervasive computing environments. SOCAM provides a middleware of components to define context providers, interpreters, and the interaction between different components. All of these approaches use APIs to provide the interfaces and require an app to change structural design to adapt to the interface. Configuration of the APIs is tedious and inaccuracy in setting the API parameters correctly can result in failure of the system. Thus, to reduce development effort a non-invasive interface provider framework is needed which requires very less configuration. Thus this research proposes TDM app to provide critical interfaces for MMAs. It allows developer to use the interfaces in the same way as communicating with other app on the smartphone and avoids possible configuration issues.

3.2 Evidence Generation

Simulations can be used as an evidence for verifying MMA design against requirements. Lack of accurate simulation of smartphone app and human physiology interactions in a MMA may result in suboptimal design. This can cause significant hazards to the physical environment e.g., wrong infusion of insulin can cause hypo or hyper-glycemia problems [40]. Simulation of MMA interactions with human physiology are challenging and traditional approaches of interfacing domain specific simulators may have simplifying assumptions that adversely affect efficient MMA design. As MMA has smartphone app, sensor/actuator code as cyber system and human physiology, environment as physical system, the currently available tools to simulate cyber physical systems (CPS) can be used to simulate MMAs. WCPS [41] focuses on simulating wireless civil infrastructural control systems and the effects of network delays and data loss on control. Truetime [42] is a Matlab-based simulator used for simulating the effect of network performance on continuous control systems. Further, to capture the wireless network characteristic more accurately, NCSWT [43] and PiccSIM [44] integrate control system simulators with NS-2 for wireless network simulation. tools such as WCPS, GISOO [45] use TOSSIM [46] and COOJA [47] respectively to have more realistic wireless network models. Other network simulators model both packet level and continuous traffic flows in the network[48, 49, 50, 51, 52]. However, a common disadvantage of these simulators is that they do not consider events from continuous systems. These events can change parameters of the control system and hence change the nature of continuous dynamics that govern the system variables. Further, they do not consider any form of time refinement to accurately estimate event timings and hence take the fixed time step approach towards simulation.

Several individual efforts are taken to assure trustworthy properties in MMAs using formal methods [53]. Inaccurate control input to actuator computed by MMA can cause serious harm to human physiology. This issue can be viewed as verification hybrid model and existing techniques such as reachability analysis can be used to verify the control algorithm before implementation [54, 55, 56, 57]. In these methods, due to linearization of systems, the available linear solutions are directly applicable. These approximation errors are nonlinear in terms of discretization and may even increase exponentially with increase in simulation time. To improve the reachability analysis with linearized systems, the state space is decomposed into linearization domains and considered overlapping linear regions[58, 59]. This method increases the number of state variables, increasing the computation complexity. Also the reach set approximation error increases with increase in size of linearization domains. HyperTech [60] proposes the use of interval numerical method to approximate the reach set for nonlinear Hybrid system. It uses rectangular shape to represent the non-rectangular region shape, resulting into increased wrapping error. Also this error keeps on growing with each iteration as reachability analysis considers sets from previous time, which adds the errors from previous time step as well. To avoid these linearization errors, recently few methods have been proposed which compute reachable sets of non-linear system without linearizing them. One of the method proposes polynomial Zonotope to capture nonlinearity of the system [61]. It converts nonlinear system equations into polynomial differential equations. This approach models error while converting nonlinear system equations into polynomial differential equations by adding uncertainty. The verification of the parameterized hybrid systems using first order discrete dynamics logic has been proposed in [62]. However it demonstrates reachability analysis of linear systems only. Further, nonlinear systems can have discontinuous invariant conditions. Thus, the use of Satisfiability Modulo Theory (SMT) is proposed to encode these conditions for nonlinear hybrid systems [63]. However, the reachability analysis for these nonlinear systems with the encoded invariant is not specified. Thus, this research proposes a reachability analysis algorithm for nonlinear systems using exponential box splines to reduce error of linearization in reach set estimation.

3.3 Techniques to Ensure Trustworthiness of MMA

The security violation of the apps can be caused by compromising health data by sharing it with non-legitimate app. To combat this issue, Android uses sandboxing to secure direct data access between apps. Recently, there have been efforts taken to assure data security in mobile apps with companies such as Happtique [64], however most of these approaches are subjective. Moreover, the proposed regulation system had failed to effectively detect serious security flaws in apps such as password stored in plain text. Further, there has been many efforts taken to secure data communication over wireless channel for in body implanted sensors and smartphone [65, 10, 66, 67, 68]. The security algorithms which use physiological parameters to encrypt the data are considered to be unbreachable [10, 66]. However they can lead to possible threats by regenarating the parameters using physiological models [69]. Mocana Atlas platform [70] supports password-based authentication to data storage. It checks for app tampering through penetration checking which might take long time or attack might happen in between checks. Thus runtime validation of the apps requests is necessary to avoid security breaching.

The availability of the sensors can be jeopardized by low battery capacity. This issue is mainly addressed by using schemes such as throttling, duty cycling, sustainable sensing [71]. All these efforts to combat safety, security and sustainability issues focus independently on each issue and try to find solution for it. However in case of MMAs, we need a solution which should consider the inter-effect of safety, sustainability and security issues on each other.

The current commercial app market approval interface also does not require the apps to conform to any kind of safety, sustainability or security standards. While it is still unclear how safety and security in health apps, and sensor sustainability will be regulated or otherwise overseen, it is clear that an objective means for assessing them before market approval is needed. It is hypothesized that the objective assessment of MMAs requires standardization. It should focus on areas of causing issues such as interfacing sensors with smartphones, inefficient sensor code implementation, insecure data storage in smartphone and Personal Health Record (PHR), interactions of sensors and actuators with the human body. This creates a need to have a MMA development framework which can ensure trustworthy properties while generating the evidences at every step of the development process. Thus the proposed research focuses on building such framework which can specifically focus on trustworthy properties in MMAs.

Chapter 4

SYSTEM MODEL AND TRUSTWORTHY FRAMEWORK FOR MMA

4.1 System Model

There can be multiple mobile medical apps working in a smartphone as shown in Figure 4.1. They might be communicating with sensors, actuators, clouds and even with other apps to share health data. If all the communication at interfaces is trustworthy, we can call the MMA reliable. Figure 4.2 shows one instance of a mobile medical app. It communicates with physiological sensors and actuators implanted on user's body through wireless channel. These apps use the sensor collected physiological data for further processing such as diagnosis, controlling actuator operation, displaying data in user readable format and uploading it to an PHR in cloud server.

4.2 Traditional Android App Architecture

Current state of the art operating model of Android smartphone employs application sandboxing (shown in Figure 4.3). In this method, an application is assigned a Unique

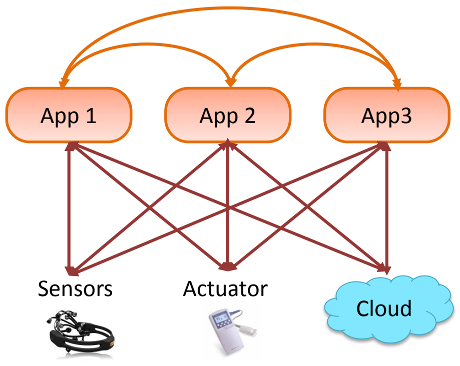


Figure 4.1: System model for Mobile Medical Applications.

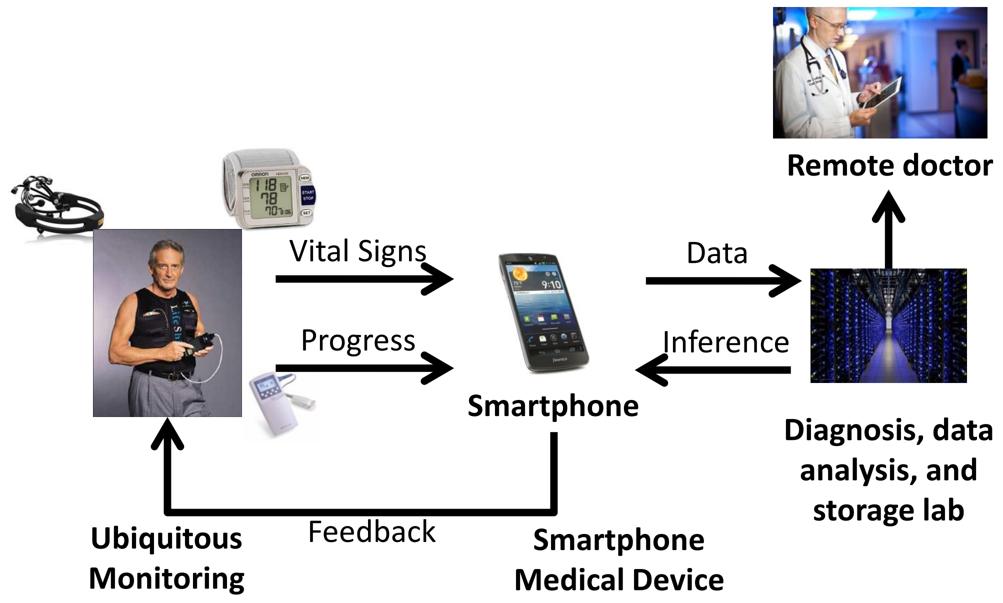


Figure 4.2: Mobile Medical Application.

User ID similar to the Linux operating system policy of assuring data access security between two users. In such an execution model, an application A runs in a dedicated process which is divided into a set of n threads. Each process has its own encrypted space in the available pool of resources. Thus, the memory is divided amongst the total number of applications and the memory addresses are encrypted using an ID, *uuid*, that is unique to the process p_i^A associated with application A. Also each thread instantiated by a process is associated with the same *uuid*. The isolation of memory space and threads entails that an application A_i cannot read or write in the memory allocated to a different application A_j. The only way two processes can communicate is via an Inter Process Communication (IPC) call where an app can ask for specific permissions to read data from another app using its own *uuid*. The advantage of such a sandboxing strategy is that all data access to an app can be monitored and traced back to its source. Further, any access to the external storage, sensors, actuators, smartphone peripherals, and the communication radio, has to be initiated via IPC calls. Moreover, access permissions for these components have to be

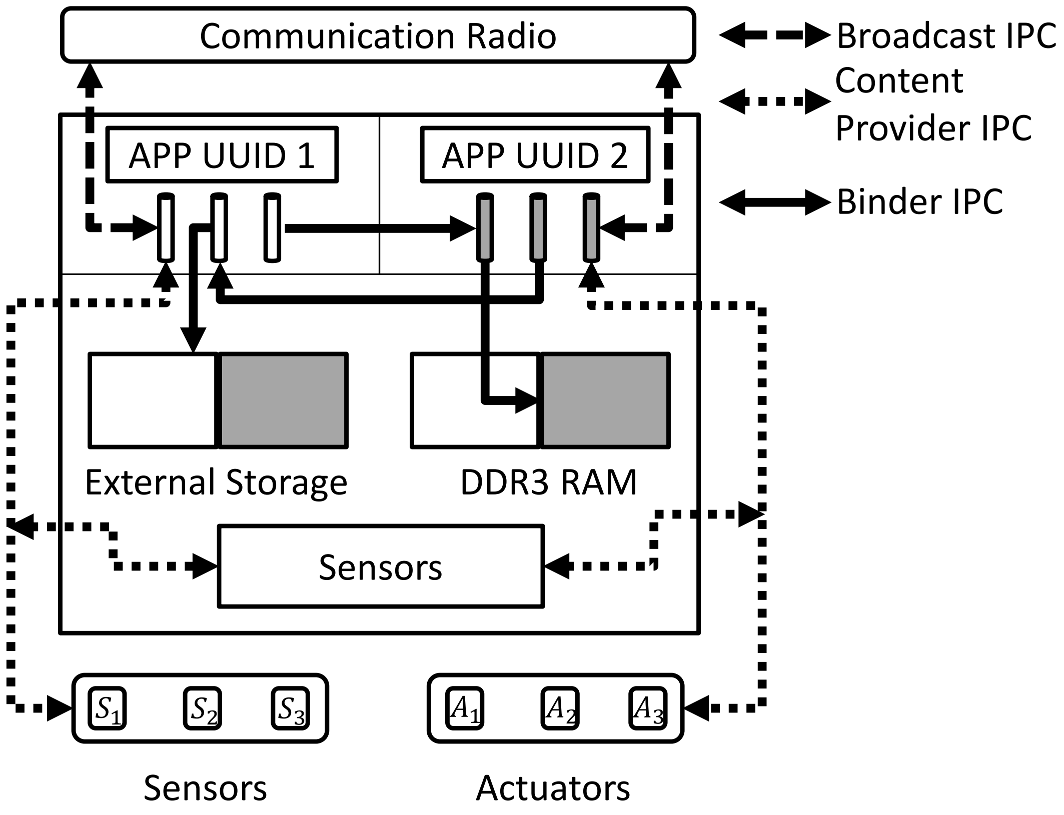


Figure 4.3: Traditional App Execution Model.

set while packaging the code into an encrypted executable. Thus, the mobile app developers are in charge of assuring privacy of the health data when communicated over wireless channel or stored in cloud. They also have to check for tampering of MMAs frequently to avoid their malfunctioning. However, developers may not be security experts. Further, to develop MMAs which communication with sensors/actuators, developer has to learn the programming language specific to the wearable devices.

4.3 Problem Statement

Given the definitions of trustworthy properties in MMAs and current app working model, the following research problem has been addressed in this dissertation: *how to ob-*

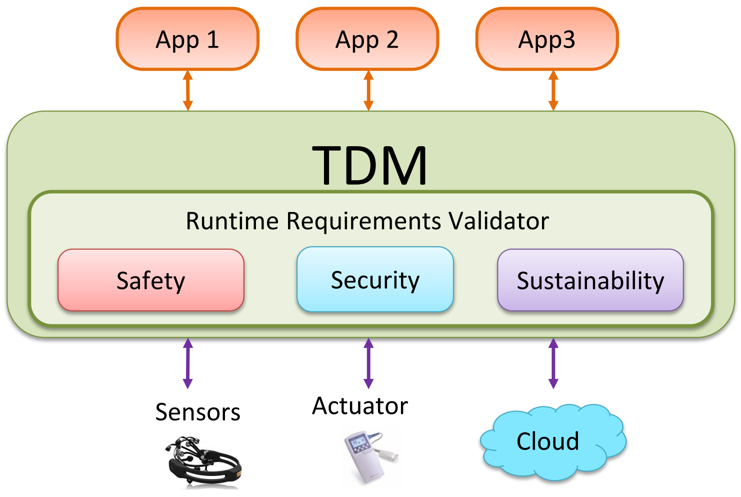


Figure 4.4: Conceptual Operating Model for MMAs.

jectively assure patient safety, wearables sustainability and data security in MMAs given the divergent app development methodologies?

One possible approach to solve this problem is to use automated techniques. Such techniques can be used, if the implementation follows a standard model. Thus an app model is proposed as a standard operating model for MMAs, discussed in detail in Section 4.4.

4.4 Proposed Trustworthy Framework for Mobile Medical Apps

It is hypothesized that the trustworthy MMAs should have an operating model as shown in Figure 4.4, where every instance of data communication to the sensor, actuator and cloud occurs through a certification entity, Trustworthy Data Manager (TDM). It includes runtime requirements validator to ensure trustworthiness of every request it processes.

App model requirements: The proposed app model should satisfy following requirements:

- *Safety*: Any form of control input from MMA should be tested for patient safety.

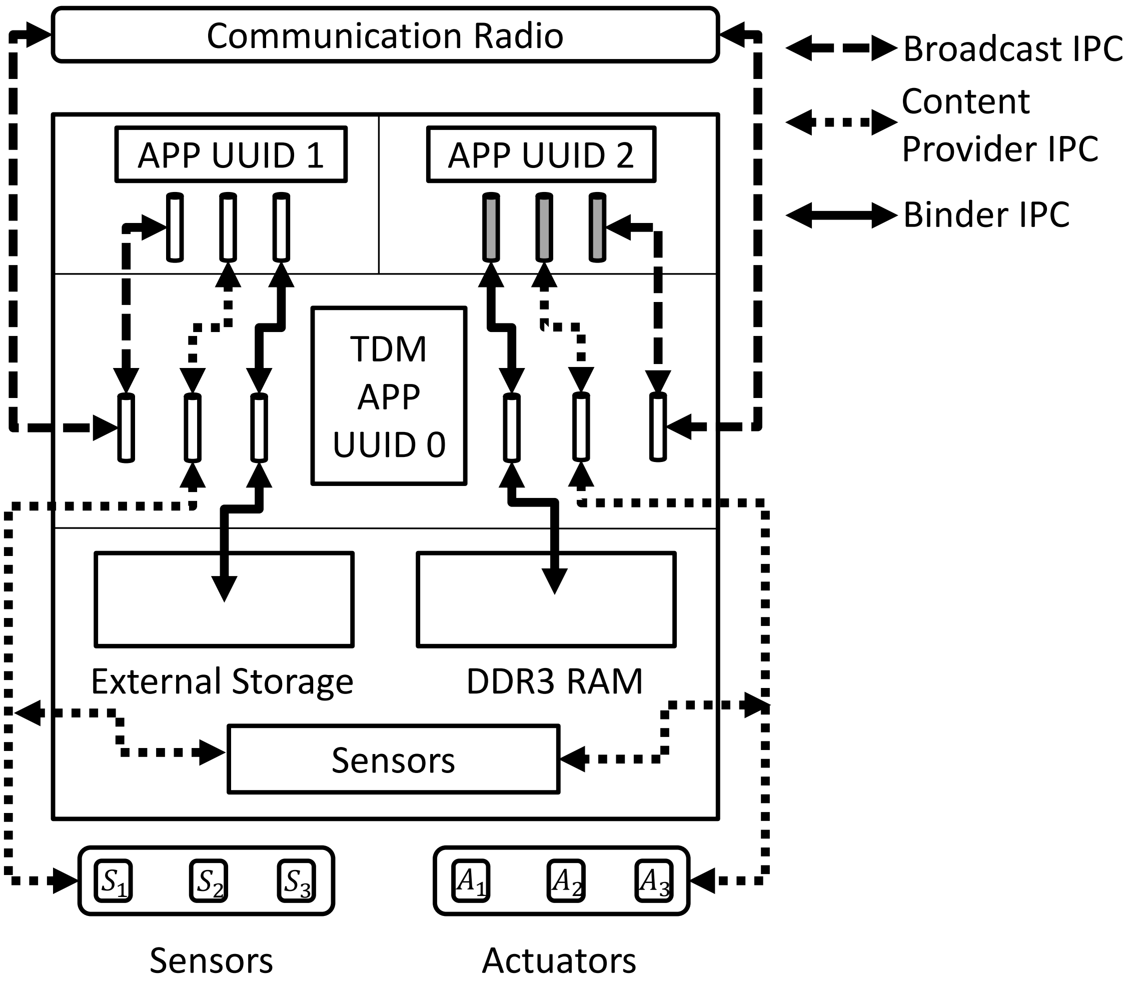


Figure 4.5: Proposed App Execution with TDM.

- *Sustainability*: Any sensor communicating with TDM should support long term availability with the sustainable design.
- *Security*: Any form of data communication should be privacy ensured by TDM.
- *TDM overhead*: Working of TDM app should not degrade performance of any MMA or non-health app.

Assumption: *Here the assumption is, all trustworthy MMAs follow app model.*

The app model enables the development of MMAs in the form of a suite, where participating apps are certified by the regulatory agencies against trustworthy requirements. The app model registry will provide guidelines to other apps to become a part of the application suite e.g. the types of sensors being used by currently available apps in the application suite, sensor sampling frequency, energy management algorithms. If any app does not follow the registry guidelines of existing application suite and it utilizes sensor data from one of the sensors being currently used in application suite, the new app should be able to modify its requirements or consider re-developing those. If any app can not follow the registry guidelines due to its mandatory requirements, it can create new application suite.

With the app model, the smartphone software itself is in charge of only the graphical and algorithmic aspects of the application. Any form of control input is tested for patient safety, sensor configuration change is tested for sustainability and data communication is privacy ensured by TDM using runtime validation models. These models include hybrid automata based model checking for safety assurance, optimization algorithm for sensor sustainability and finite state automata model for ensuring security. Further, data collected by different applications are kept in secured databases (similar to application sandboxing).

4.4.1 Change in App Architecture

Traditional app architecture has explained in Chapter 4 using Figure 4.3. It is hypothesized that safety and security risks can be mitigated if MMAs follow the high level model shown in Figure 4.4. In this model all data storage and peripheral communication in a MMA are handled by the Trustworthy Data Manager. The TDM is a separate app that runs in background (Figure 4.5). Using such a standard model of operation of MMAs enable runtime safety, sustainability and security checks. The TDM can match the outcomes of vulnerable operations of a MMA with their expected behavior expressed using models. This enables detection of code modification and runtime deviations in execution. Further, as all the data communication from MMA occurs through TDM, developer does not need to implement the security protocols. The runtime validation of MMA input is done using finite stet automata and reachability analysis techniques. The app model reduces app development time significantly as developer does not need to implement communication interfaces.

4.4.2 Runtime Validation of TDM

The TDM runtime process monitoring and validation functionality is not only agnostic to user preferences but also has enough embedded intelligence so that it can detect anomalous behavior of an app as shown in Figure 4.6. An app can register with a TDM by first establishing a Binder IPC channel with the TDM. Through the Binder channel it passes the UUID to TDM. The app can have its Multi-Tier Model encrypted using the UUID so that no other app except the TDM can access its models. TDM has a broadcast listener implemented which continuously listens for app requests. Once it receives a request from MMA, it will invoke a message parser which separates the requests into two parts: ID and Data. Corresponding to every ID there is an algorithm for runtime requirement checking. The algorithm will load requirements for the specific ID from the database. These requirements can be the safety regulations of mobile medical apps, the sensor sustainability requirements and the HIPAA security requirements. Simultaneously a model simulator will take the data as input and simulate a model for certain time to generate system parameter. Models may include Finite State Automata based representation of secure data management schemes, hybrid automata models of safety assessment of control algorithms and optimization algorithm to check sustainable sensor configuration. The TDM will be equipped with simulators that can execute the models and estimate the expected operating condition or state of the app for current phone context(CPC). Now if the output of the simulator which is a system parameter matches the requirements, the TDM will create new

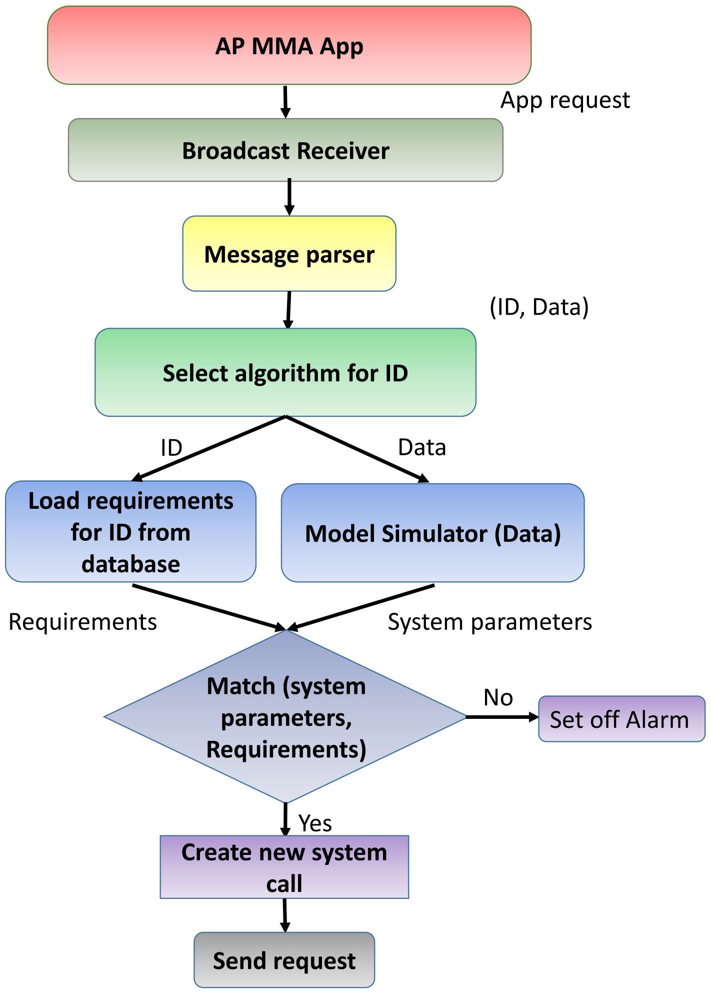


Figure 4.6: Manifestation of the TDM Runtime Environment for Validating the Apps Operation Against Requirements.

system call to issue the initial app request. If the requirement is not matched, the TDM will set off an alarm.

The runtime validation functionality of the TDM can be expressed using functions $f_{safety}, f_{sustainability}$ and $f_{security}$ as shown in Equations 4.1, 4.2 anf 4.3.

$$f_{safety} =< M_1^P(CPC, V_h), Th > \tag{4.1}$$

$$f_{sustainability} =< M_2^P(CPC, P), B_{min} > \tag{4.2}$$

$$f_{security} =< M_3^P(CPC, \tau_{M_3}), True\ or\ False > \tag{4.3}$$

Here, M_1, M_2 and M_3 are models used by TDM to validate the app request against safety, sustainability and security. V_h is the physiological parameter controlled by the app and Th is the safety threshold value. P is the power required for the sensor for specified operation in the system call whereas B_{min} is the minimum threshold value for the sensor battery level. Data access request τ_{M_3}, is checked against M_3 to see if the request is valid or not for a given CPC.

A more accurate estimation of the actual state of the app can be obtained by tracing the system calls made by the app. The app makes different types of system calls such as Broadcast Provider for radio transmission or Content Provider for accessing the sensors and actuators. All these system calls goes through the TDM app in our proposed method. and hence the TDM can trace the state of the app from these system calls. These two states can be compared in runtime. If there is a match then the app behaves as the proposed model and hence its system call is trustworthy and the TDM allows the requested operation. If the app operation does not match the model then a default operation is performed. This default operation has to be specified by the App and is kind of a fail safe mode of the application. If this fail safe mode is ever reached an exception will be considered by the TDM and the app

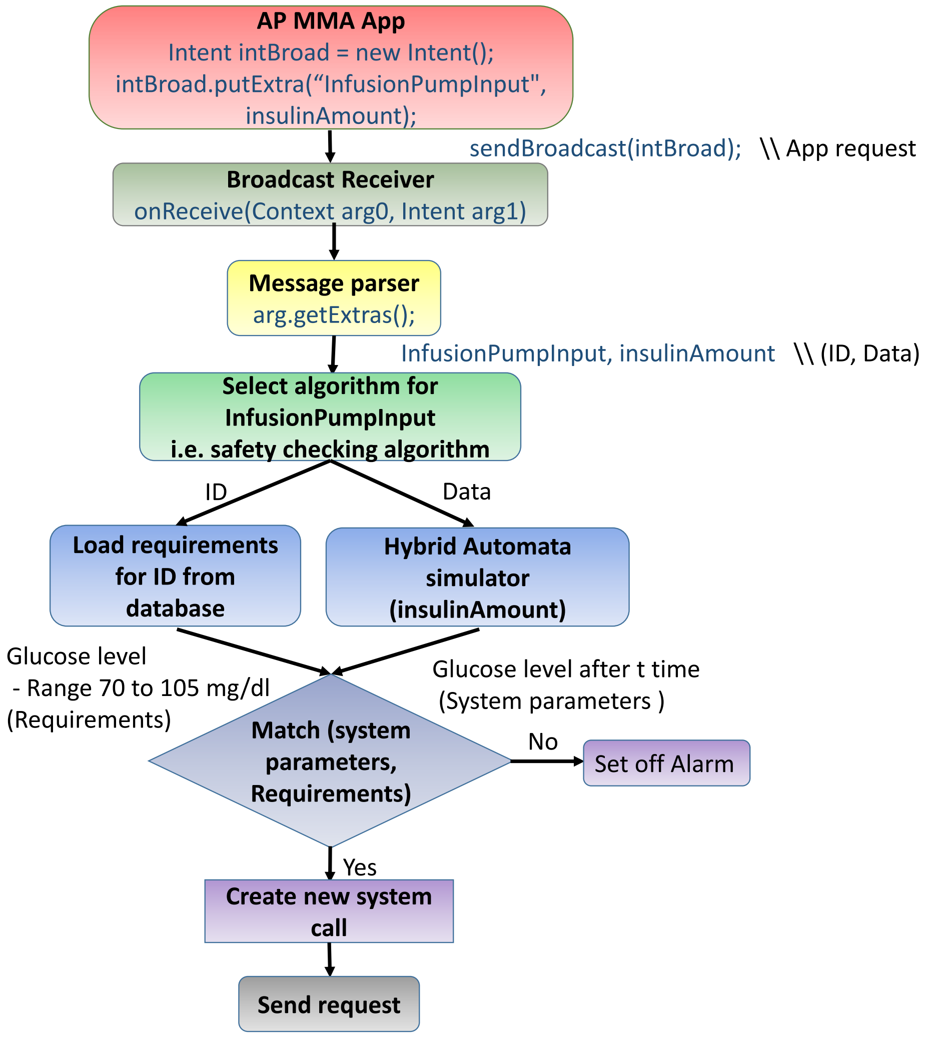

AP MMA App
Intent intBroad = new Intent();
intBroad.putExtra("InfusionPumpInput", insulinAmount);
sendBroadcast(intBroad); \\ App request
Broadcast Receiver
onReceive(Context arg0, Intent arg1)
Message parser
arg.getExtras();
InfusionPumpInput, insulinAmount \\ (ID, Data)
Select algorithm for InfusionPumpInput i.e. safety checking algorithm
ID
Data
Load requirements for ID from database
Hybrid Automata simulator (insulinAmount)
Glucose level - Range 70 to 105 mg/dl (Requirements)
Glucose level after t time (System parameters)
Match (system parameters, Requirements)
No
Set off Alarm
Yes
Create new system call
Send request

Figure 4.7: App Request Processing by TDM for Artificial Pancreas MMA.

user will be notified. Fail safe modes are also essential for handling false positives from the TDM runtime monitor. This is an important problem in healthcare domain when an useful action performed by the app is prevented because of false suspicion of vulnerability. In such cases fail safe modes of apps or appropriate user intervention through alarms will be considered.

4.4.3 Artificial Pancreas MMA Working with TDM

Artificial pancreas MMA is a controller app to keep the blood sugar within desired limit as explained in Example 3 from Chapter 2. The app sends request to change insulin concentration to infusion pump. Given new app working model, all MMA request pass through TDM. Figure 4.7 shows the MMA request processing by TDM for artificial pancreas app. Artificial Pancreas MMA sends the message of change in insulin concentration to TDM using *Intent* message. Broadcast listener implemented in TDM receives this message and invokes a message parser which separates the requests into two parts, ID and Data. For artificial pancreas, the ID is infusion pump input and data is amount of insulin to be infused. As for this example the request is for infusion pump to change drug concentration, the algorithm for checking if the requested insulin amount is safe or not is called. The algorithm loads requirements for the specific ID from the database i.e. glucose safe range. Simultaneously a model simulator takes the data as input and simulate a model for certain time to generate system parameter. For safety verification hybrid automata model for artificial pancreas is used (Figure 2.1 from Chapter 2). The simulator takes amount of insulin to be infused as input, simulated hybrid automata for that input and outputs blood sugar level after certain amount of time. Now this output matches the requirements i.e. the sugar level within required range, the TDM creates new system call to issue the initial app request. If the requirement is not matched, the TDM will set off an alarm.

Chapter 5

EVIDENCE GENERATION: MMA SUSTAINABLE DESIGN

Sensors aesthetically embedded in accessories such as jewelry, piercings or contact lenses are being used by MMAs to collect physiological data. These symbiotic wearable wireless sensors are envisioned to operate on scarce harvested energy resources from the human body. In addition to the hardware and software constraints arising from the form-factor and low energy operations, there are safety requirements such as avoidance of physical injury. The design implications of these requirements are non-intuitive and may involve estimation of human physiological dynamics. The physical impact of a sensor operation can be controlled by appropriate design of multiple sensor components such as processor, radio, and optimization of data algorithm. For example, the risk of thermal injury to tissue can be reduced by limiting the sensing frequency, the computation power, and the radio duty cycle of body worn sensor. Hence, it is a challenging task to trace back a cause of a physical impact to hardware and software design decisions in a sensor. This research proposes a novel non-linear optimization framework to consider safety and sustainability requirements that depend on the human physiology and derive system level design parameters of a sensor [7].

5.1 Safe and Sustainable MMA Configuration Optimization

5.1.1 Motivating Examples

Single sensor thermal safety: Thermal effects of a sensor on the human body follows a complex pattern and varies with the location and placement of the sensor. Sensors worn on the arm or on the chest, where the skin is not very sensitive to the heat energy, for short periods of time does not cause severe heat related problems. Sensors such as Shimmer,

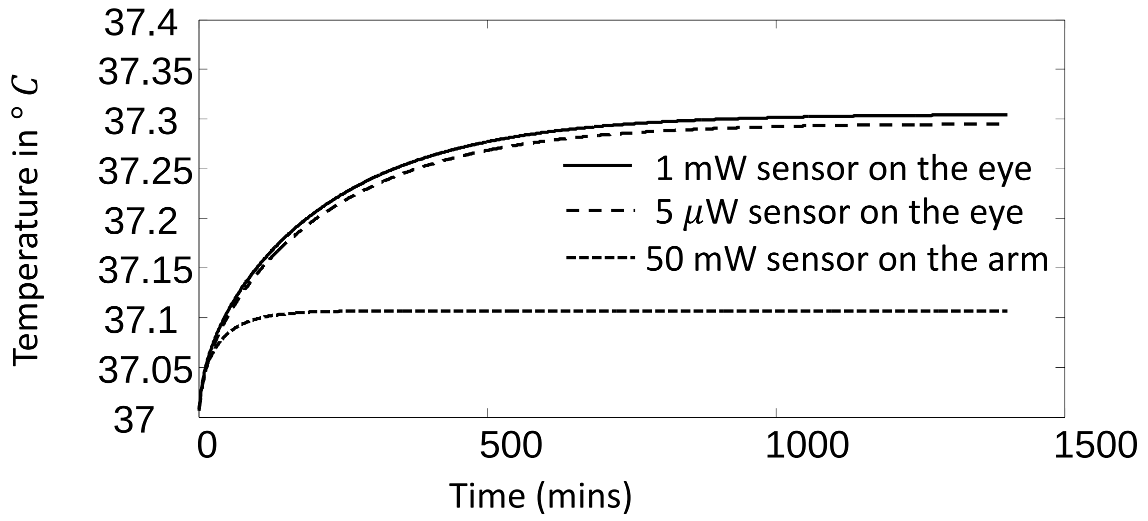


Figure 5.1: Skin Temperature Rise Depending on Sensor Location.

TelosB, with power consumption around 50 mW, cause negligible temperature rise in the human body ($\approx$ 0.01 °C) for a prolong operating time of 24 - 48 hours [5]. Symbiotic sensors on the other hand will have a power cap of 5 μW [72], but can have a higher thermal effect due to their location. We conducted a simulation study that shows a non-intuitive result as depicted in Figure 5.1. The skin temperature rise of a sensor with 1 mW power dissipation installed on the cornea is greater than that of a sensor with 50 mW power worn on the human arm for 24 hrs by almost 0.2 °C. Another interesting fact is that the reduction of sensor power does not significantly change the temperature rise. As shown in Figure 5.1, if power of the sensor is reduced to 5 μW [72] from 1 mW, the skin temperature reduces only by 0.01 °C. This result is due to the fact that the average temperature rise of the human tissue around a sensor is not only a function of the power consumption of the sensor, the frequency of sensing, the data transmission rate but is also a function of the blood perfusion rate and hence varies based on the location of installation.

Networked Infusion Pump: In a networked autonomous infusion pump, a wireless controller samples the blood glucose levels from a glucose meter and computes the future infusion rate to stabilize blood glucose concentration in the human body. The interaction between insulin and glucose can be modeled as the spatio-temporal [73],

$$\frac{\partial d}{\partial t} = \triangledown(D \triangledown d) + \Gamma(d_B(t) - d) - \lambda d, \tag{5.1}$$

where $d(x, t)$ is the tissue drug concentration at time t and distance x from the infusion site, D is the diffusion coefficient of the blood, Γ is the blood to tissue drug transfer coefficient, $d_B(t)$ is the prescribed infusion rate at time t, and λ is the drug decay coefficient. The design of the controller involves finding five parameters, the optimal sampling rate of glucose sensor, infusion change or increment step by which the controller increases infusion, allowable bolus rate, the set point $d_B(t)$, and the delay in taking control decisions.

5.1.2 Problem Formulation

The principal research problem that we seek to answer in this paper is -
Given a set of design requirements expressed in the form of a set of favorable states, find a sensor design that always keeps the system in one of the favorable states.

Optimization approach: We frame the problem of finding the sensor design that satisfies regulatory requirements in an optimization framework. Let us consider a contact lens glucose sensor placed on the retina which consumes P_s power for sensing, P_c for data transmission, P_{sec} for executing security protocol. The sensor has an energy storage device of capacity B_c and its stored power at time t is denoted by $P_b(t)$. The energy available from the scavenging source is denoted by $E(t)$ at time t. There will be constraints on the frequency at which the sensor can sense and communicate data. Let us consider that the sensing frequency is f_s and the communication frequency is f_c. The aim is to determine P_s, P_c, P_{sec}, B_c, f_s and f_c from an optimization formulation that minimizes the temperature rise of the human body part and also never depletes the storage device.

Let us consider that the specific absorption rate of the tissue is SAR, which is directly proportional to the radio power and inversely proportional to the square of the distance between the sensor and the point at which temperature is to be calculated. The change in temperature of the tissue is represented using Penne's bioheat equation [74] as follows,

$$\rho C_p \frac{dT}{dt} = K \nabla^2 T - b(T - T_b) + \rho \mathrm{SAR} + P_{avg}, \tag{5.2}$$

where ρ is the mass density, C_p is the specific heat, K is the thermal conductance, and T is the temperature of the body part, A is the surface area of contact with the device, T_r is its temperature, b is the blood perfusion constant, T_b is the blood temperature, and P_{avg} is the average power required by the sensor.

The Penne's bioheat equation [75] can be written as a temporal differential equation by discretizing it over $N \times N$ space grid. It is discretized over time and space using the finite difference time domain(FDTD) technique [74].

At grid points (i, j), the temperature $T(i, j)$ is:

$$\begin{aligned} \frac{dT(i,j)}{dt} &= \left[-\tfrac{(b\delta^2+4K)}{\rho C_p \delta^2}\right] T(i,j) + \tfrac{SAR}{C_p} + \tfrac{b}{\rho C_p} T_b + \tfrac{P_{avg}}{\rho C_p} \\ &+ \tfrac{K}{\rho C_p \delta^2}[T(i+1,j) + T(i,j+1) + T(i-1,j) \\ &+ T(i,j-1)]. \end{aligned} \tag{5.3}$$

If we consider $\boldsymbol{T}$ to be a vector consisting of the temperatures at each grid point, then Equation 6.16 is analogous to Equation 6.17, which is in the form of linear time invariant differential equation.

$$\dot{\boldsymbol{T}} = A\boldsymbol{T} + B, \tag{5.4}$$

where, matrix A will have dimensions $N^2 \times N^2$ and its each row will be as,

$$A = \begin{array}{c} \begin{matrix} \cdot\cdot & X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 & \cdot\cdot \end{matrix} \\ \begin{pmatrix} \cdot\cdot & \alpha & \cdot\cdot & \alpha & \beta & \alpha & \cdot\cdot & \alpha & \cdot\cdot \end{pmatrix} \end{array} \tag{5.5}$$

where, $\alpha = \frac{K}{\rho C_p \delta^2}$, $\beta = -\frac{(b\delta^2+4K)}{\rho C_p \delta^2}$, $X_1 = N(j-1) + i$, $X_2 = N - 2$ zeros, $X_3 = N(j+1) + i$, $X_4 = Nj + i$, $X_5 = Nj + (i-1)$, $X_6 = N - 2$ zeros, $X_7 = Nj + (i+1)$. The dotted elements of matrix A are all zero. The computed matrix B of dimension $N^2 \times 1$ is as follows,

$$B = \begin{array}{c|c} & 1 \\ 1 & \\ \vdots & \begin{pmatrix} \frac{SAR}{C_p} + \frac{b}{\rho C_p} T_b + \frac{P_{avg}}{\rho C_p} \\ \vdots \\ \frac{SAR}{C_p} + \frac{b}{\rho C_p} T_b + \frac{P_{avg}}{\rho C_p} \end{pmatrix} \\ N^2 & \end{array}. \tag{5.6}$$

The solution to Equation 6.17 is a non linear function of time given by,

$$\boldsymbol{T}(t) = e^{A(t-t_0)}\boldsymbol{T}(0) + e^{At}A^{-1}(e^{At} - e^{At_0})B, \tag{5.7}$$

where t_0 is the starting time.

The average power consumption of the sensor over a time τ can be obtained using Equation 5.8,

$$P_{avg} = (P_s f_s + (P_c + P_{sec})f_c). \tag{5.8}$$

A principal requirement of the sensor to operate is the energy neutrality constraint. A system is energy neutral for a time τ if the storage device level remains unchanged after the execution, i.e., $P_b(0) = P_b(\tau)$. Thus, effectively all the power required for computation comes from scavenging sources (Equation 5.14),

$$(P_s f_s + (P_c + P_{sec})f_c)(\tau - t_0) \leq \int_{t_0}^{\tau} E(t)dt. \tag{5.9}$$

The power from the scavenging sources are not directly provided to the sensor but has to be stored in a storage device. The storage device should have a capacity greater than the peak power consumption of the sensor. Also, the available charge at any time should be greater than the power requirements of the sensor,

$$P_b(t) \geq P_{avg} \text{ and } B_c \geq max\{P_s, P_c, P_{sec}\} \tag{5.10}$$

The aim of the optimization analysis is to find P_s, P_c, P_{sec}, B_c, f_s and f_c such the temperature rise is maximized without violating dangerous levels. In order to frame it as an optimization problem as shown in Equation 5.15, we have to convert the above requirement into an objective function. Let us assume that $\boldsymbol{T}$ should not increase by more than ϵ°C. In such a case, the objective function can be framed as minimization of the quantity, $abs(\epsilon - max(\boldsymbol{T}(t) - \boldsymbol{T}(0)))\forall t \in [t_0 \ldots \tau]$, where τ is the final time.

For the energy scavenging unit the energy $E(t)$ can be given by the linear model $E(t) = \alpha(max(\boldsymbol{T}) - T_{amb})$, where α is the thermal resistance of the human body and T_{amb} is the ambient temperature.

$$
\begin{aligned}
&\textbf{Find } P_s, P_c, P_{sec}, f_s, f_c, \text{ and } B_c \text{ that minimizes}\\
&abs(\epsilon - max(\boldsymbol{T}(t) - \boldsymbol{T}(0)))\forall t \in [t_0 \ldots \tau]\\
&\textbf{such that}\\
&\text{[Penne's Equation]}\ , \forall t : \boldsymbol{T}(t) = e^{A(t-t_0)}\boldsymbol{T}(0) + e^{At}A^{-1}(e^{At} - e^{At_0})B, \qquad (5.11)\\
&\text{[Energy Neutrality]}, (P_s f_s + (P_c + P_{sec})f_c)\tau \leq \int_0^\tau E(t)dt,\\
&\text{[Storage Constraint]}, \forall t : P_b(t) \geq P_{avg},\\
&\text{[Capacity Constraint]}, \forall t : B_c \geq max\{P_s, P_c, P_{sec}\}.
\end{aligned}
$$

5.1.3 Validation

The optimization problem is solved in MATLAB using *fmincon*, which starts with initial values of variables and minimizes the scalar function within the specified constraints. We show the usage of the optimization technique on three examples: a) single sensor operation, b) cluster head scheduling for implanted sensor networks, and c) designing infusion control algorithm.

Single sensor example We use the problem formulation of Section 5.1.2. The objective function of Equation 5.15 is used as a scalar function for *fmincon*. ϵ is considered as 0.1 °C. Energy sustainability of sensors are described by the battery constraints and energy neutrality constraints.

We consider the initial values of design parameters as, sensing power $P_s = 7.1\mu$W, data transmission power $P_c = 50$ mW, power for executing security protocol $P_{sec} = 12$ mW, sampling frequency $f_s = 100$ Hz, and frequency of data transmission $f_c = 6/3600$ Hz. For forming the complete optimization problem, we set feasible design considerations for aforementioned parameters as $P_s = 5\ \mu$W, $P_c = 1\ m$W, $P_{sec} = 5\ m$W, $f_s = 10$ Hz, and $f_c =$ 0.001 Hz. These are minimum values of design parameters for currently available sensors. Using these values, we solved the optimization problem and obtained the optimized design

Table 5.1: Power Consumption of Sensor Query Task.

Platform	Consumed Power	Time
TelosB	5.46 mW	500 ms
Imote2 (13 MHz)	156.15 mW	286 ms
BSN v3	6.586 mW	235 ms
SHIMMER 2R	7.145 mW	452.8 ms

Table 5.2: Radio Power Consumption Results with Plaintext Packets and Encrypted Packets.

Platform	Consumed Power Encrypted data(mW)	Consumed Power Plain data(mW)
TelosB	58.2	47.6
Imote2	209.4	198.3
BSN v3	70.7	59.7
SHIMMER 2R	72.4	60.1

parameters as $P_s = 8.3\,\mu$W, $P_c = 61$ mW, $P_{sec} = 13.3$ mW, $f_s = 112$ Hz, and $f_c = 1/360$ Hz which satisfied all the constraints.

To implement and experimentally validate our optimization design, we have bench-marked commercially available sensing platforms TelosB, Shimmer, Imote2, and BSN v3. Table 5.1 enlists the power consumption of respective platform for sensing once. We measured sensing power and radio power while both plain data and encrypted data are being sent. Table 5.2 shows the bench-marking results for the radio power.

From Tables 5.1 and 5.2, the Shimmer platform is the best match for our optimized power consumption constraint. Thus, we have used it for validation.

After obtaining optimal design for sensors and choosing testing platform as Shimmer, we need to develop software code for it with our optimized design. For code development, we have used the automated code generator, Health-Dev [6]. It takes requirements of wearable sensors in the form of models in Advanced Architectural Description language (AADL) and generates downloadable code for them. The automated code generation reduces manual implementation errors.

We input sampling frequency as 112 Hz, data transmission frequency of once every 6 mins, and choice of sensor as ECG sensor to Health-Dev [6]. Then, we have downloaded the generated code into Shimmer platform and wore it to measure ECG signal continuously for 12 hours. We conducted two sets of experiments: a) without any radio duty cycling and b) with the optimize sensing frequency and duty cycle. At the end of the testing period, we measured the energy required and rise in temperature of the body part. With no duty cycling and at a sampling rate of 250 Hz, the energy consumption was 432 J and the temperature rise was (≤ 0.167 °C) on the arm. With the optimized design, the sensor consumed 11 J of energy and there was no measurable temperature rise. The power available from body heat was 9 J and we considered an energy storage unit with initial energy 4 J.

Networked implanted sensors In this example, we consider a network of implanted sensors, which communicate in a cluster protocol. The implanted sensors form a cluster, where a cluster head collects all communication packets from the participating sensors and sends it out to a base station. Here, for each sensor, the sensing power P_s is fixed to 3 μW, communication power P_c is fixed at 5 μW and there is no security protocol, hence $P_{sec} = 0$. The sensing frequency is set at 10 Hz while the communication frequency is set at 0.5 Hz. Since the cluster head has to collect data from all other sensors, it has the most power consumption and the tissue around the cluster head is heated the most. To keep the temperature of the surrounding tissue within limits, the cluster head has to be frequently rotated. In this example, we find out the optimal cluster head rotation scheme that can keep the temperature within thresholds at different parts of the body.

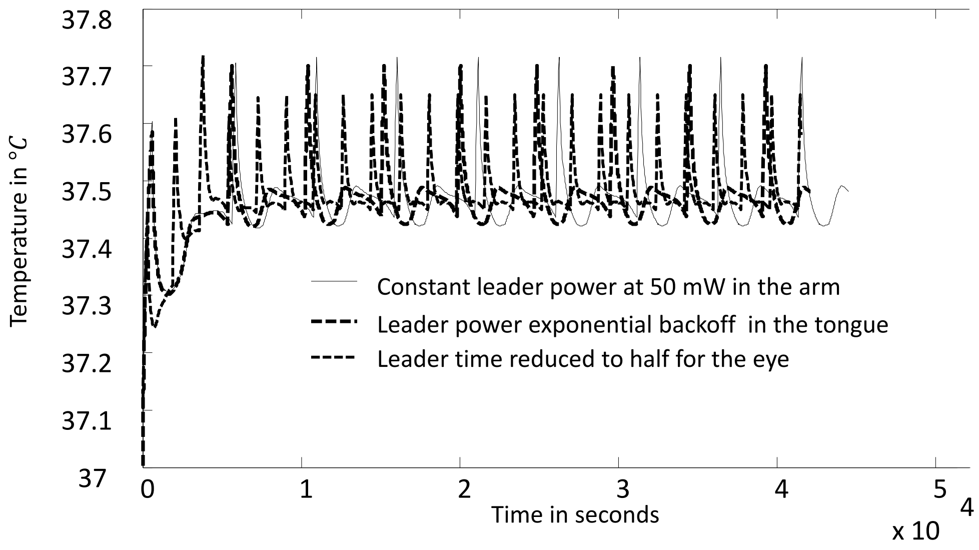


Figure 5.2: Cluster Scheduling Schemes for Implanted Sensors.

We observe that as the location within the body changes, different leader rotation schemes are needed to keep the temperature within thresholds as shown in Figure 5.2. In the arm where the blood perfusion is the lowest, the leader can be changed every 600 s to keep the temperature below 37.8 °C. However, for the same threshold a cluster in the tongue will have to have its leader power reduced to half, every time the temperature is greater than 37.6 °C in addition to changing its leader every 600 s. If the implant is in the eye, the leader has to be changed every 300 s.

Designing infusion pump control algorithm For the infusion pump case study, we consider a requirement that the drug concentration should not be above 1300 μg/min. Our aim is to derive the design parameters discussed in Section 5.1.1 such that this requirement is satisfied. In addition to the five design parameters, we also consider the packet delivery ratio as a measure of the wireless channel characteristics. Equation 5.1 is of the same form as the Penne's bioheat equation and hence can be solved using FDTD. The form of the

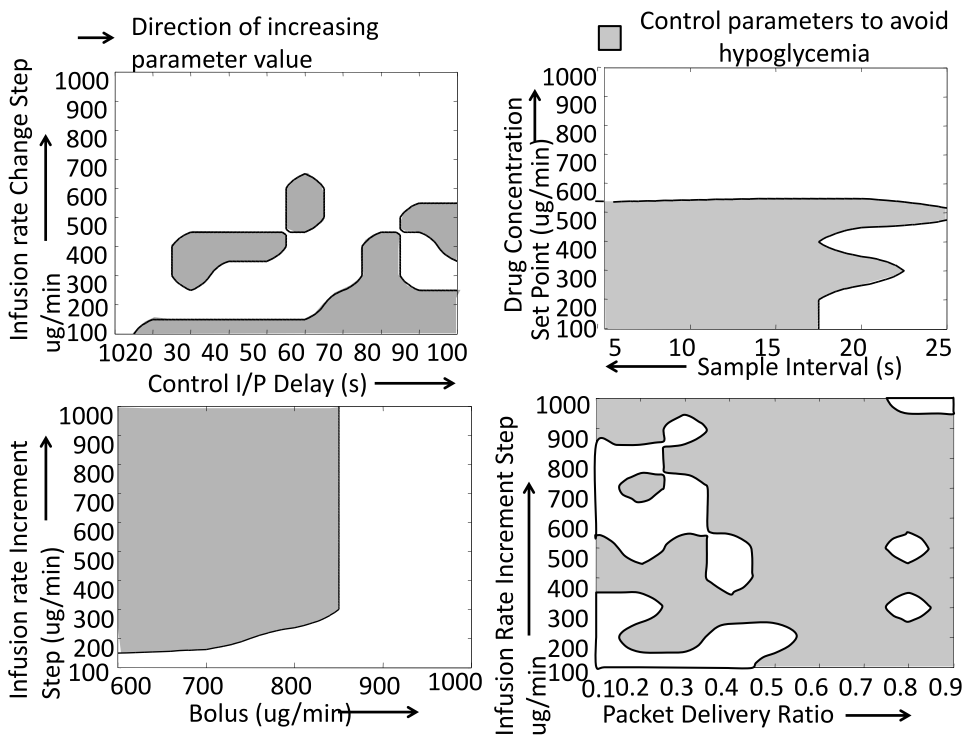


Figure 5.3: Infusion Pump Parameters for Avoiding Hypoglycemia.

objective function will thus be the same as that in Equation 5.15. Figure 5.3 shows the results of 10000 optimization runs. Since several local minimums are possible, we get a set of design parameters (within the gray bounded region) that satisfy the requirements.

5.2 Sustainable MMA Configuration Optimization

Wearable sensors typically scavenge energy from light or human body heat. The availability of scavenging sources is unpredictable. Thus, to have sustainable working of the sensors, we form an optimization problem to achieve energy neutrality which satisfies Equation 2.5. In this problem formulation, we optimize the data sending frequency of the sensor depending on the availability of scavenging source. It mainly considers the energy obtained from the solar or light and does not focus of battery design or its efficiency.

Optimization approach: Let us consider a wearable sensor which consumes P_s power while sensing and sending the data over wireless channel. $P_b(t)$ denotes the number of packets can be sent with available battery charge at time t. The energy available from the scavenging source is denoted by $E(t)$ at any time t. R represents the random sequence to show the availability of scavenging source. Time slot allocated to each entry in R is denoted by τ, e.g., if $R = [1, 0, 0, 1]$, where, 1 denotes scavenging source is available and 0 as source is not available, each of the 1 and 0 will occur for τ time. The availability of scavenging source puts constraints on the sensor sending frequency. Let us consider that the sending frequency is f_c when scavenging source is available and f_d when scavenging source is unavailable. The aim is to determine f_c and f_d that achieves energy neutrality.

We consider scavenging source as light for formulating the optimization problem. Models of solar traces are used to form the sequence R. For solar power, single diode equation to simulate the available electrical power generated from a single PV (photo voltaic) panel[76] is used, whose current-voltage characteristic is given by Equation 5.12,

$$I = I_{pv} - I_o(e^{\frac{\varepsilon(V+IR_S)}{nkT_c}} - 1), \tag{5.12}$$

where, I is the total current generated in PV panel, I_{pv} is the photo generated current, while I_o is the dark saturation current with respect to the ambient weather pattern. Further k denotes Boltzmann's gas constant, ε denotes the charge on one electron, T_c denotes PV cell temperature in Kelvin, n denotes the number of solar cells in the PV, finally V and R_s denote junction thermal voltage and series resistance of the PV panel.

The solar panel on sensor starts charging only when the sensor processor is awake for computations. This infers that, the data sending frequency is directly related to charging or discharging of the battery. Thus, f_c can act as charging rate and the f_d as discharging rate.

Given a solar light availability sequence R, we have derived the sequence R_f, which is the sequence of sensor sending frequencies while executing the sequence R. e.g., if $R = [1, 0, 0, 1]$, $R_f = [f_c, f_d, f_d, f_c]$. The scavenged energy obtained during this period is

Find f_c and f_d that optimizes

$(\sum R)\tau * (I^2 r) - \sum(P_s * R_f)$

such that

$$\begin{aligned}
&\text{[Solar Energy]}, \forall t : E(t) = R * \tau * (I^2 r), \\
&\text{[Energy Neutrality]}, P_b(k-1)\tau - \tau R_f(k\tau) \geq 0, \\
&\text{[Storage Constraint]}, \forall t : P_b(t) \geq 0, \\
&\text{[Battery Capacity Constraint]}, \forall t : P_b(t) \leq P_{bMax}.
\end{aligned} \tag{5.15}$$

given by Equation5.13,

$$E(t) = R * \tau * (I^2 r) \tag{5.13}$$

where, I is the current calculated using Equation 5.12 and r is the internal resistance of the solar panel.

A principal requirement of the sensor operation is the energy neutrality constraint. A system is energy neutral for a sequence R of scavenging source availability, if the storage device level remains unchanged after the execution, i.e., $P_b(0) = P_b(length(R) * \tau)$, where τ is the time for which each element in R will execute. Also, battery capacity at any time t is limited by P_{bMax} which is the maximum number of packets to be sent when battery is fully charged. Thus, effectively all the power required for computation should not exceed power obtained from scavenging source given by Equation 5.14 which is similar to Equation 2.5,

$$(\sum R)\tau * (I^2 r) > \sum(P_s * R_f), \tag{5.14}$$

where, k is the time at which the data is sent.

The aim of the optimization analysis is to minimize f_c and f_d given energy constraints. In order to frame it as an optimization problem as shown in Equation 5.15, we have to consider an objective function. It can be framed as $(\sum R)\tau * (I^2 r) - (\sum P_s * R_f)$, such that the power consumption in sensor should not go above the scavenged power (Equation 5.15).

The optimizer algorithm gives a set of time constrained sustainable designs. Developer can chose any design depending on available sensors and their configuration settings.

5.2.1 Validation

To validate the sustainable optimized sensor design obtained using the proposed optimizer, we used the MSP430 Solar Energy Harvesting Development toolkit (MSP430 SEH) [77]. It harvests energy from solar or ambient light and stores in the rechargeable battery for use when harvesting source is not available. The fully charged battery can send upto 400+ data samples. We emulate the general light source for the experimentation using a smartphone flash light. The light source variation allows the sensor to charge and send signal to compute device. The data collected on the compute device includes the remaining packets the sensor can send in the existing energy state of the harvester. Using the remaining packets data, the optimizer can set the frequency of packet sending for the sensor.

The optimization problem from Section 5.2 is used to find sustainable design for MSP430 SEH. This sensor scavenge energy from light, thus we used random sequence for R to indicate light availability. Matlab *fmincon* function is used to optimize the problem in Equation 5.15 with initial values for sending frequency when light is available, $f_c = 0.2Hz$, sending frequency when dark, $f_d = 0.2Hz$, battery capacity in terms of number of packets can be sent, $P_b = 360$, maximum number of packets sent when battery is fully charged, $P_{bMax} = 400$ and time step for each entry in R, $\tau = 20sec$.

After optimizing Equation 5.14 using *fmincon* function, the minimal values of frequencies are obtained, $f_c = 1.6Hz$ and $f_d = 0.76Hz$. The sensor is programmed with these optimal frequencies. The flash light sequence is set to R in smartphone application. The

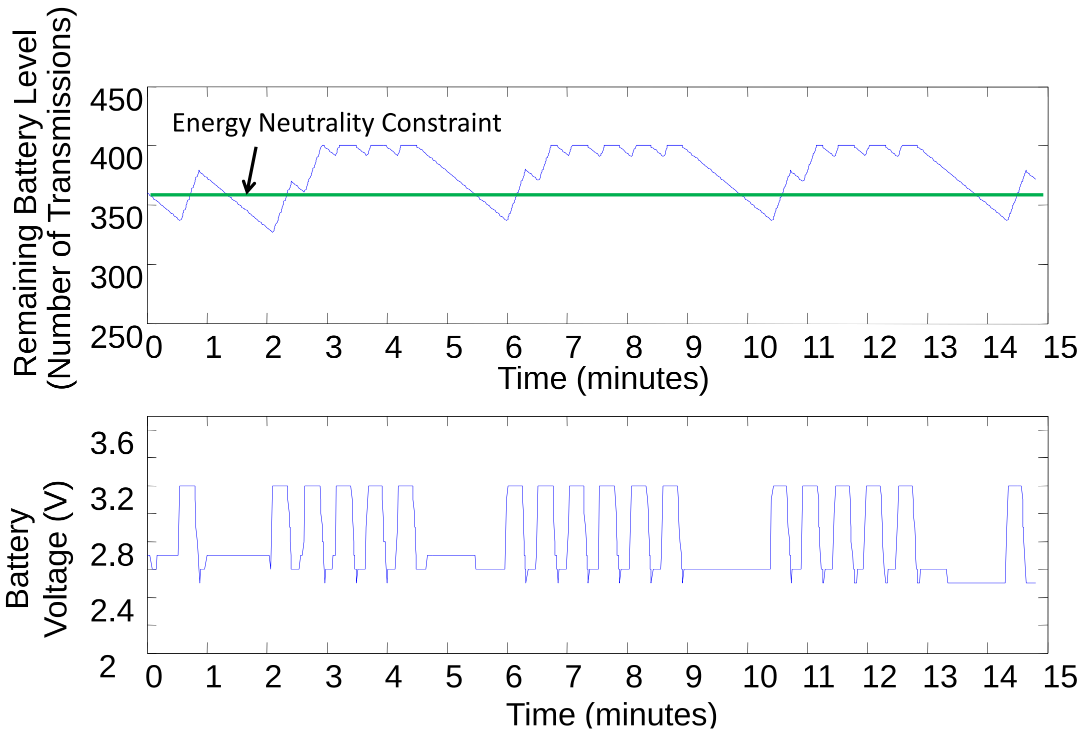


Figure 5.4: Experimental Results for Sustainable Sensor Design.

sensor harvests energy according to the flash light sequence and sends the data to compute device. The variation in battery voltage due to on-off light sequence in R is as shown in Figure 5.4. The results show that energy neutrality is achieved for a given light sequence R and satisfy our sustainability requirement defined in Equation 2.5.

Chapter 6

EVIDENCE GENERATION: HYBRID SIMULATION

The sensor configuration framework described in Section 5.1 uses hybrid simulator only to simulate continuous dynamics of the human physiology. However, if we want to simulate the MMA system design with controller app software, sensors and human physiology, we need to consider events from both continuous dynamics of human physiology as well as events from discrete system which is controller software. Thus, this research focuses on how to accurately simulate the events from both continuous an discrete domains [78].

In MMAs, controller algorithm in smartphone which is a computing system and physical dynamics form a closed loop control system, which is a cyber-physical system. The interactions between these two systems are guided by two types of events: a) *random discrete computing events*, that arise from the interaction of a user with the MMA, and b) *physical events*, that arise due to threshold crossing of continuous system variables of human physiology system. These events reconfigure the computing and physical system parameters. This research focuses on accurate simulation of cooperation between computing and physical systems, their interactions, through a unified hybrid simulation approach.

Simulations play an important role in: a) estimating resource requirements and designing their organization, b) estimating cost, c) comparing strategies, and d) verifying the design against requirements. While analytical techniques such as model checking and formal requirements verification may provide more rigorous evaluation framework, they often have limited solutions for complex systems including delayed differential dynamics. For such cases, simulation provides a time efficient and scalable solution. Lack of accurate simulation of MMA interactions with human physiology may result in sub-optimal

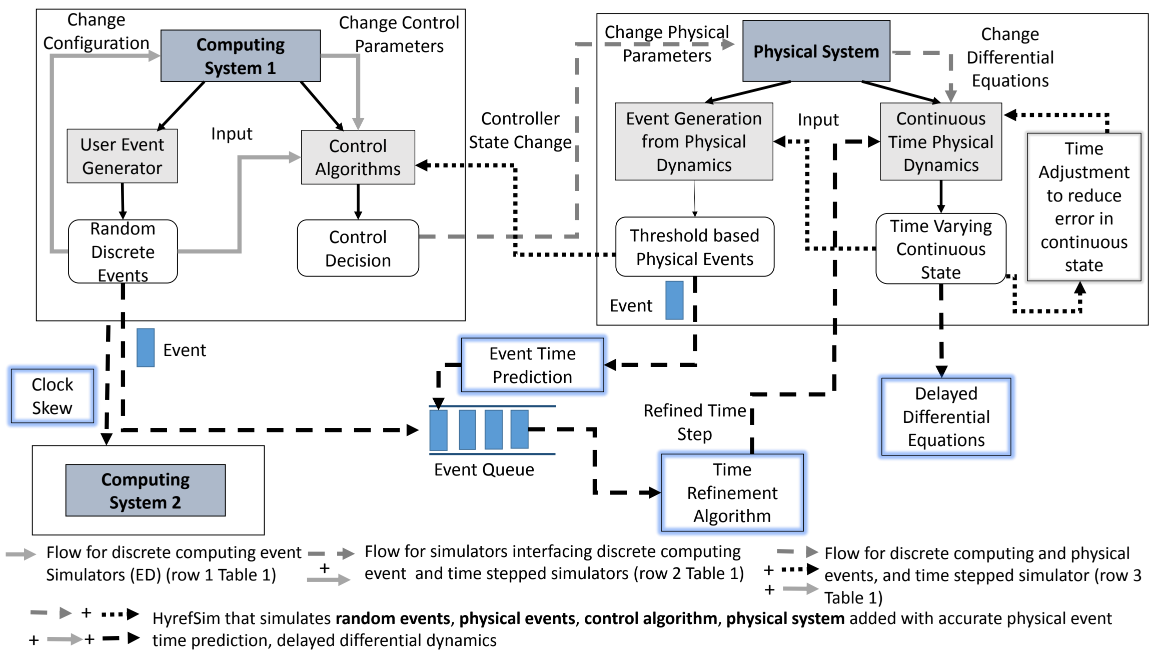

Change Configuration
Computing System 1
Change Control Parameters
User Event Generator
Input
Control Algorithms
Random Discrete Events
Control Decision
Controller State Change
Change Physical Parameters
Physical System
Change Differential Equations
Event Generation from Physical Dynamics
Input
Continuous Time Physical Dynamics
Time Adjustment to reduce error in continuous state
Threshold based Physical Events
Time Varying Continuous State
Event
Event
Clock Skew
Event Time Prediction
Refined Time Step
Delayed Differential Equations
Event Queue
Computing System 2
Time Refinement Algorithm
Flow for discrete computing event Simulators (ED) (row 1 Table 1)
Flow for simulators interfacing discrete computing event and time stepped simulators (row 2 Table 1)
Flow for discrete computing and physical events, and time stepped simulator (row 3 Table 1)
HyrefSim that simulates random events, physical events, control algorithm, physical system added with accurate physical event time prediction, delayed differential dynamics

Figure 6.1: Distributed Cyber-physical Systems with Computing Units Operating in Close Loop with Physical systems.

design. This can cause significant hazards to the physical environment e.g., wrong infusion of insulin can cause hypo or hyper-glycemia problems [40], burning of the skin due to over-heating of the wearable sensors. Simulation of MMA interactions with human physiology are challenging and traditional approaches of interfacing domain specific simulators may have simplifying assumptions that adversely affect efficient MMA design.

Traditionally there are two different paradigms of simulation: a) event driven (ED), that progresses by processing events that can change system variables and generate new events, and b) finite horizon time stepped (FHT), that progresses by increasing time by a small fixed amount and evaluating the dynamics of system variables. ED simulators operate in discrete time and hence cannot simulate continuous dynamics of human physiology while FHT operate in continuous time and can only process events at the start of a time slot. For

Table 6.1: Summary of Existing CPS Simulation Tools.

Existing CPS simulation tools	**Discrete computing events**	**Physical system**	**Physical events & time adjustment to reduce error**	**Event time prediction**
Omnet++ [79], Situation Calculus based simulator [80]	✓	×	×	×
GISOO [45], Picc-SIM [44], iSEE [81], Matlab+EPANET [82]	✓	✓	×	×
NCSWT [43], Modelica [83], WCPS [41], Truetime[42]	✓	✓	✓	×
HyrefSim	✓	✓	✓	✓

MMAs, a hybrid approach is the most optimal where computing events are handled by ED at exact event times while continuous physical system is handled by FHT.

6.1 Challenges of Hybrid Simulation

Co-simulation of computing and physical events: In MMAs, random discrete computing events originating from the networked computing systems may result in change in controller configuration or may also induce mode transition for a given controller. Events in computing domain can be efficiently handled by discrete event simulators such as ns2 or OMNET++ [79] (first row in Table 6.1). Changes in the physical system variables due to a random discrete computing event in control algorithms is a property unique to MMAs. For cyber-physical systems (CPS), researchers have tackled this problem by interfacing a discrete event simulator such as ns2 with a physical simulator such as Matlab, second row in Table 6.1). These techniques can be applied to MMAs as they also form a CPS.

Time step adjustment problem: Simulating MMAs require accurate estimation of: a) solutions of differential equations using a time stepped approach, and b) physical event timings. Existing hybrid simulators (third row Table 6.1) use tools such as Simulink to estimate physical dynamics and state change in controllers due to physical events. Such simulators dynamically adjust simulation time step in order to reduce error in estimation of differential dynamics. However, they have an inherent assumption that a time step that ensures accuracy in differential dynamics can also accurately estimate physical event timings, which is often not the case as shown in Section 6.2. A slight difference in event processing times can have long lasting impact in MMA simulation, by progressively increasing error in estimating the physical system variables.
Artifacts of wrong time step adjustment: Wrong estimation of physical event timings may lead to: a) event delays, when an event scheduled to be processed within a time slot of an FHT is pushed towards the end of a time slot, b) event loss, when events scheduled to occur within a time slot of an FHT is lost at the end of a time slot since the differential dynamics fails to satisfy threshold crossing conditions, and c) false clustering of events, when multiple events scheduled to occur at different times within a time slot of an FHT are grouped at the end of a time slot, often resulting in conflicting control requests. Examples of such effects are shown in Section 6.2.

A way to avoid such errors is to have an arbitrarily small time step, however, that will drastically increase the simulation time. Hence, simulation approaches that combine physical simulators such as Simulink and discrete event simulators such as ns2, are either limited in their capabilities to process events at accurate times or take a prohibitively long processing time. This is due to a lack of a method for refining simulation time step to reduce error in predicting physical event timings. An associated side-effect of this gap is that a time efficient unified simulator that can process random events from computing domain and physical events from continuous domain is difficult to develop.

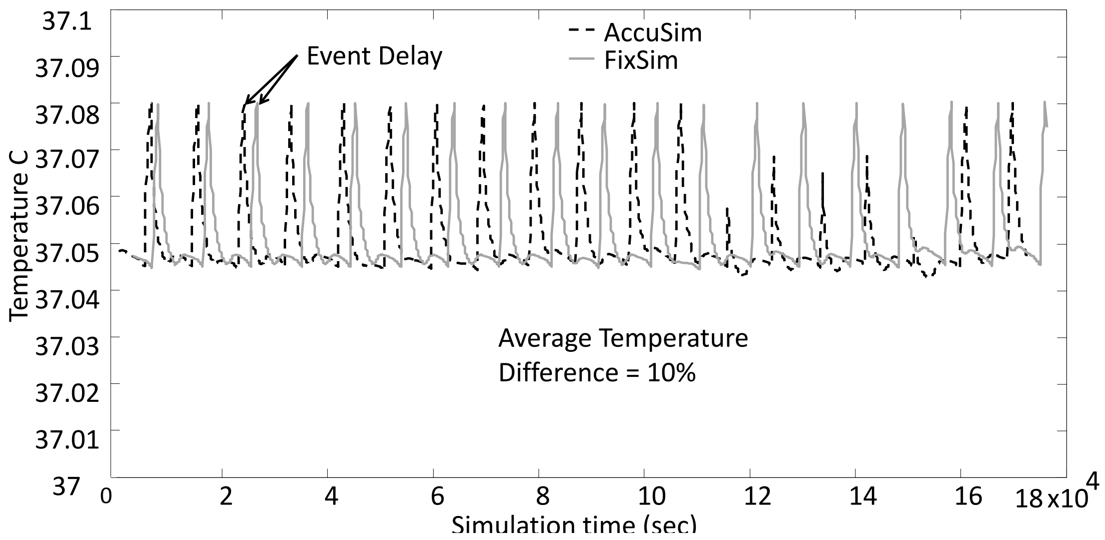


Figure 6.2: Difference in Temperature Profile for *FixSim* Type Simulation and *AccuSim* with Exact Event Times.

6.2 Practical Need for Processing Events at Accurate Time

Let us consider a example of network of implanted sensors 4 MMA, from Chapter 4 that sense physiological signals from different parts of the human body and send it back to the smartphone outside the body for storage, processing and diagnosis. The sensors form a cluster where a cluster head is randomly chosen to communicate with the smartphone. All other sensors are slaves and they only communicate with the cluster head. Communication of sensors with the cluster head leads to power consumption. Since the cluster head is the most power hungry sensor, it has to be periodically re-selected or rotated i.e., some other slave sensor takes up the task of a cluster head.

As these sensors operate inside the human body, power dissipation causes temperature rise of the tissue. This can be modeled using the Penne's bioheat equation [74]. The rotation of cluster head can also be triggered by tissue temperature rise. This is an example of physical event triggered action in the computing domain. Let us further assume that the implanted network has to be designed such that it lasts at least a year without the need for recharging. Hence, another strategy is employed such that a cluster head with energy less than 30% of initial energy has to be rotated (low energy events). We need to design a cluster head rotation scheme that never causes a temperature rise of 0.15 °C above the normal body temperature and can have at most one sensor that is not capable of being a cluster head after an year of operation. The design parameters are placement, power consumption, and energy storage size of the sensors.

Our aim is to consider the simulation paradigm of existing CPS simulators and evaluate if event loss, delay, and false clustering can cause any significant difference in design cost or operation of the system. Existing CPS simulators interface physical simulator with computing or network simulators, which use fix time step. Here, we refer to existing simulators as *FixSim*. We compare *FixSim* with a hypothetical simulator that has infinitesimally small time step and can process physical events at their exact times of occurrence.

Event Delay

FixSim results in a different temperature profile with a higher temperature rise on an average and also predicts 10% higher energy consumption of the sensors (Figures 6.2 and 6.3). *FixSim* will always process a physical event at the end of a time slot. Hence cluster head rotations due to temperature rise above a certain threshold are always delayed. Hence, *FixSim* estimates that sensors remain cluster head for a longer period of time. This error in estimation leads to an error in computing the thermal profile of the sensors and also their energy consumption. Hence, if we design the sensor network using the estimations of *FixSim*, we have to: a) over provision energy storage for the sensors, and b) reduce the power consumption of sensors to prevent higher temperature rise. Based on fuel cell cost [84], we will have to spend $ 3 extra on sensors to accommodate for 10 % more fuel cell capacity,

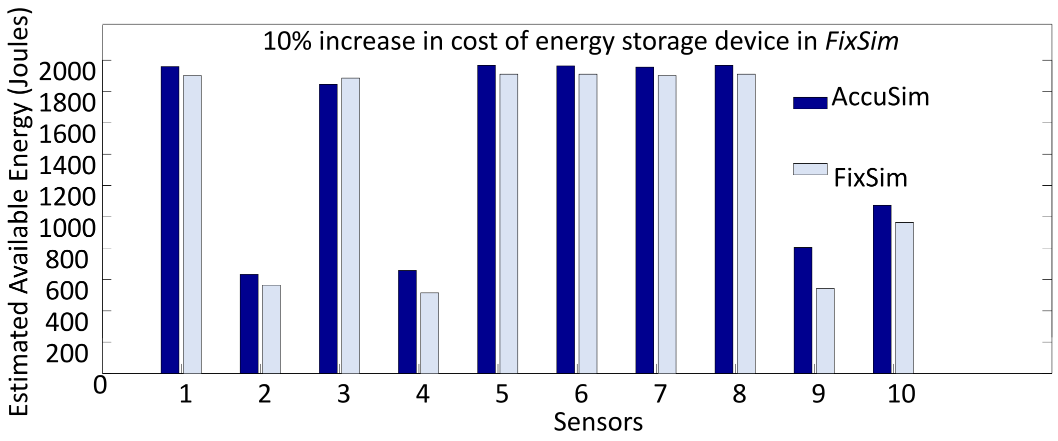


Figure 6.3: Difference in Energy Consumption of Sensors for *FixSim* Type Simulation and *AccuSim* with Exact Event Times.

and also have to reduce the power consumption to keep the temperature within the safety threshold of 37.15 °C.

Event Loss

FixSim fails to process three critical low energy events over a period of 24 hours. Error due to event delay accumulates over time, which eventually leads to loss of events in *FixSim*. From our experiments we found that for a period of 1 day *FixSim* missed three times more low energy events and predicted that only one cluster head will be out of energy. However, if we process the events at the exact time of occurrence, *AccuSim* predicts that three cluster heads will be out of energy and non-operational. Hence, if we design using the estimation given by *FixSim* then it can jeopardize the system performance.

False Clustering of Events

False Clustering of events in FixSim induces loss of events or renders events meaningless.

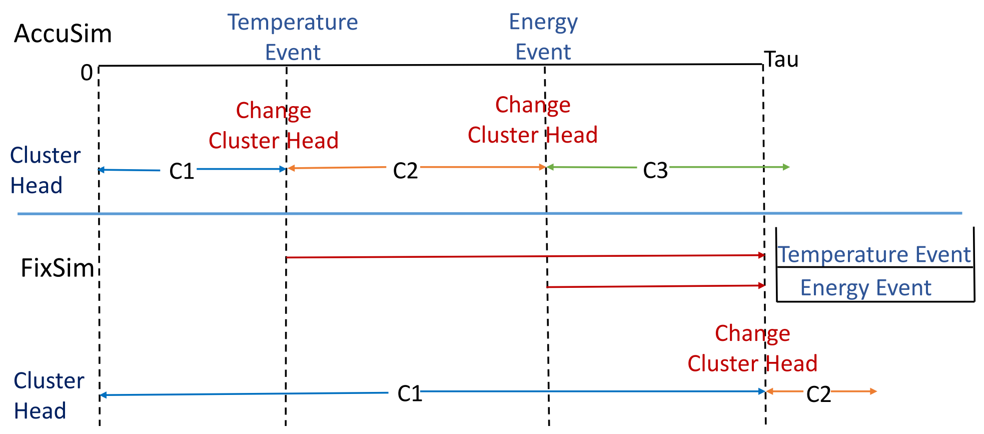


Figure 6.4: False Clustering of Events.

A case of event clustering is as shown in Figure 6.4, where within a time slot first the cluster head $C1$ has to be rotated due to temperature rise then $C2$ is selected, which had to be rotated due to a low energy event. In *FixSim*, these two events will occur at the same time slot. The simulator will consider the events in the order that they arrived at the event queue. Hence, rotation of $C1$ will be handled first. However, the low energy event for $C2$ will not make sense to the simulator since $C2$ has not yet served as cluster head and hence its energy is not reduced yet. Thus, the low energy event will not be processed and hence for the next slot cluster head $C2$ will be drained of energy. If the energy drain occurs within a time slot then *FixSim* will have an in-feasible solution.

A case for further concern is that the errors from the above-mentioned artifacts are unbounded. Figure 6.5 plots the difference between the energy estimation by *FixSim* and that by *AccuSim* that has exact event times. The error increases with respect to simulation steps t without bounds and at a rate of $O(t^5)$ (determined experimentally).

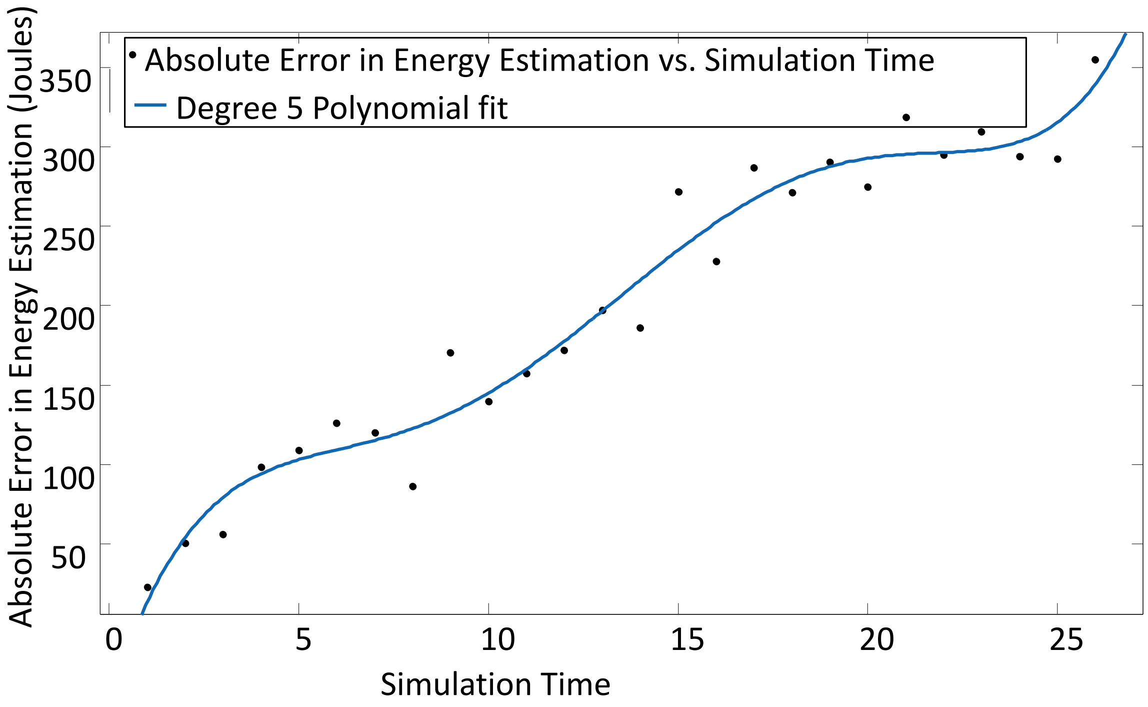


Figure 6.5: Error in Estimation of Stored Energy by *Fixsim* increases with Increase in Simulation Time without Bounds.

6.3 *HyrefSim* Simulation Approach

The proposed *HyrefSim* employs a time refinement approach towards hybrid simulation of MMAs. The primary simulator is ED and it processes events from both computing and physical operation (Figure 6.6). After an event is processed, the ED passes control to the FHT until the next event is generated. For each time step in FHT, it not only evaluates the continuous system variables but also uses sophisticated predictive models of physical systems to determine the exact time of occurrence of physical events within one step in the future. The time step is then dynamically adjusted to account for the predicted physical event at the correct time within an user specified error margin. This time refinement strategy not only minimizes event delay, event loss, and false clustering of events, but also ensures that the simulation progresses over time. However, before going to the details of the simulator, let us first consider an example and determine whether simulation errors related to event loss, event delay and false clustering can cause significant sub-optimality in the MMA design.

6.4 System Model

The discrete operation of a DCPS is expressed by a set of n_d discrete variables $\mathcal{D} = \{d_1, d_2 \ldots d_{n_d}\} \in \mathcal{Z}^{n_d}$, while the continuous operation is represented by a set of n_c continuous variables $\mathcal{S} = \{s_1, s_2 \ldots s_{n_c}\} \in \mathcal{R}^{n_c}$. Thus, at any time t the system is described by the evaluations of discrete and continuous variables $\mathcal{D} \bigcup \mathcal{S}$.

Events in a DCPS can occur from two sources (Figure 6.1): a) events generated by the computing unit at random times, and b) events generated due to threshold crossing of continuous system variables. In the proposed simulator, we consider the physical system to be represented by a set of linear delay differential equations. We define an event as -

Definition 1 *Event: An event is a tuple* $\{t_e, f_e, f_v\}$*,*

- **Event time***:* t_e *at which the event occurs,*
- **Discrete reset function** $f_e : D \rightarrow \mathcal{Z}^{n_d}$ *that changes discrete variables,*
- **Continuous reset function** $f_v : S \times f_e(D) \rightarrow \mathcal{R}^{n_c}$ *that assigns values to the continuous variables at the event time* t_e*.*

Definition of hybrid simulator: The proposed hybrid simulator evaluates continuous variables by simulating a set of DDEs. At each event, the simulation of DDEs are halted and the discrete and continuous reset functions are executed to reconfigure the DDE.

Definition 2 *Hybrid Simulator:*

A hybrid simulator is a tuple $\{\mathcal{E}, \mathcal{V}, \mathcal{I}, \{\mathcal{A}_s, \mathcal{A}_s^{\tau_1} \ldots\}, \mathcal{B}_s, t_{sim}, \{G_c, C_c\}, \mathcal{E}_s, \epsilon, \epsilon_r\}$,

- **Event set** $\mathcal{E}$*: where each event is expressed by the tuple in Definition 1, the event set has m different types of events,*

- **Variable set** $\mathcal{V} = \mathcal{D} \bigcup \mathcal{S}$*, with* $n = n_d + n_c$ *variables,*

- **Discrete event generator** $\mathcal{I} : t_{sim} \rightarrow \mathcal{E}$ *takes the current time as input and generates a discrete event as output in some future time.*

- **Physical dynamics:** *The continuous variables vary according to a DDE Equation 6.1,*
$$\frac{d\mathcal{S}(t)}{dt} = \mathcal{A}_s\mathcal{S}(t) + \mathcal{A}_s^{\tau_1}\mathcal{S}(t-\tau_1)\ldots + \mathcal{B}_s, \tag{6.1}$$
where $\mathcal{A}_s^{\tau_i}$ *is an* $n_c \times n_c$ *matrix* $\forall i$ *and* $\mathcal{B}_s$ *is an* $n_c \times 1$ *vector,*

- **Current simulation time:** t_{sim}*,*

- **Physical event generation function set** $G_c : 2^{\mathcal{S}} \rightarrow \mathcal{R}$ *is a set of m event generation functions, each function combines different continuous variables and evaluates to a single real variable that can be compared to a threshold.* $C_c \subset \mathcal{R}^m$ *is a vector of m real threshold values corresponding to each unique event,*

- **Physical event labeling function** $\mathcal{E}_s : (G_c, C_c, \mathcal{S}(t_{sim})) \rightarrow \mathcal{E}$ *is a function that evaluates an event generation function* $g_k \in G_c$ *and generates an event* e_k *if* $c_{k+1} > g_k(\mathcal{S}(t)) > c_k$.

- **Simulation resolution** ϵ

- **Continuous state error margin** ϵ_r

For the network of implanted sensor example in Section 6.2, Definition 1 can be instantiated as, Event(event time, *Cluster Rotation Algorithm*, *Power Estimation Algorithm*). *Cluster Rotation Algorithm* sets one sensor as the cluster head and all others as slaves and accordingly adjusts their power levels. *Power Estimation Algorithm* uses the power level settings from *ClusterRotation* and set appropriate power consumption estimates to the sensors.

Definition 2 can be instantiated for sensor network example as described in Table 6.2.

6.4.1 HyrefSim Execution Model and Implementation

The execution of *HyrefSim* is a specific logic for progression of time. We have implemented *HyrefSim* in MATLAB and the implementation logic is shown in Figure 6.6.

HyrefSim combines simulation paradigms from both event driven (ED) and Finite Horizon Time (FHT) simulation. Events generated from both of computing and physical domain are simulated in event-based simulator, ED. Continuous dynamics of the physical system is simulated using time-stepped simulator, FHT, with variable time step. The time step is either calculated by simulator depending on the given error bound on simulation results, a technique used by existing simulators such as Simulink, or obtained from the time refinement strategy proposed in this paper. With the defined time step, FHT simulator simulates: (i) DCPS continuous dynamics, (ii) interactions between and cyber and physical systems of the DCPS, and (iii) threshold conditions on continuous variables of the DCPS. The ED simulator processes events from both discrete and continuous simulators.

The simulator starts with an initial set of events en-queued in the event queue of ED. These events are generated by the discrete event generator in Definition 2. Examples include Markov chain models of random events. All events in the event queue are sorted according to their execution time and processed by the ED in order. The initial control is given to the ED. It processes an event by computing the discrete and continuous reset functions f_e and f_v and then determines the next event time from the queue. Then the ED transfers control to the FHT. The FHT simulates the physical dynamics until the next event time using a time step that reduces DDE estimation error below ϵ_r.

Table 6.2: Instance of Hybrid Simulator Definition for Sensor Network.

HyrefSim definition components	**Sensor network example parameter**
$\mathcal{E}$ (set of events)	cluster head change or reduce cluster power level
$\mathcal{V}$ (variable set)	*Discrete variables:* radio power level, sampling rate and *Continuous variables:* human body temperature T, energy consumption in sensors
$\mathcal{I}$ (Discrete event generator)	Sensor turn ON turn OFF
Physical dynamics	The change in temperature of the human body tissue is represented using Penne's bioheat equation [74] as follows, $$\rho C_p \frac{dT(t)}{dt} = K \nabla^2 T(t) - b(T(t-\tau) - T_b) + \rho \mathrm{SAR} + P_c, \quad (6.2)$$ where ρ: mass density, C_p: specific heat, K: thermal conductance, T: temperature of the body part, P_c: power generated by the sensor processor, b: blood perfusion constant, T_b: blood temperature, τ : delay in propagating the temperature from the human blood to the sensor, and SAR: specific absorption rate of the body
t	simulation time
Physical event generation function	The body temperature (T) goes beyond threshold ($37.07°C$), $T > 37.07$. Thus, here the function G_c is a single valued identity function, $G_c(T) = T°C$.
$\mathcal{E}_s$ (Physical event labeling function)	When the condition $T > 37.07$ becomes true, continuous state simulator generates a cluster head change event

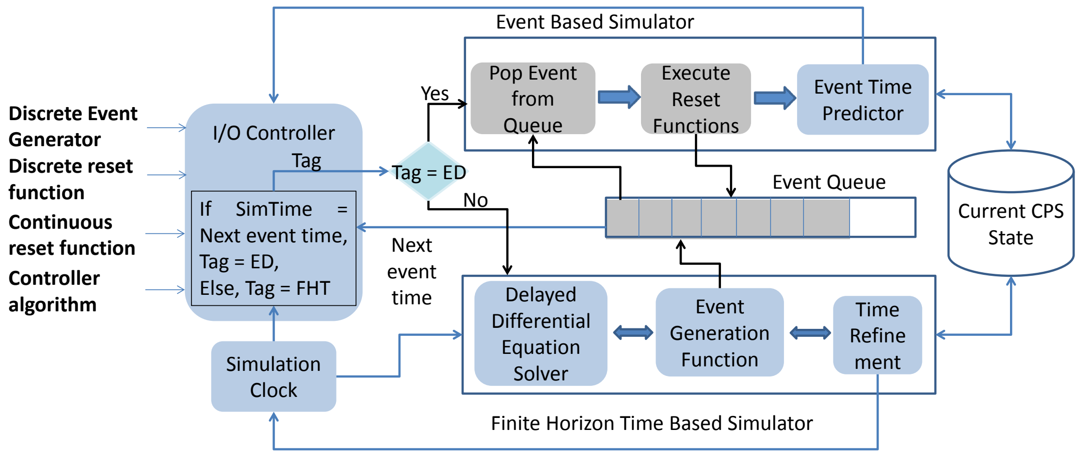


Figure 6.6: The *HyrefSim* Execution Model.

The FHT increments simulation time and keeps track of the current system state. For each step it employs a predictor to check for any physical event one step in the future. If there is a physical event occurring within a simulation time step, the FHT employs the time refinement technique to compute the exact time of the event. This is done by numerically solving the DDE with proper boundary conditions. The FHT then resets the simulation time step to the computed time.

If the FHT generates an event, it pushes the event to the event queue and passes control to the ED simulator. The ED simulator processes the event and again calls the FHT to simulate for a time duration until the next event in the queue. The simulation time step is also reset to the previous value.

6.5 Estimation of Physical Event Time

We present a mathematical discussion on how to configure *HyrefSim* simulation time step such that: a) it minimizes loss of events and delay, b) it avoids false clustering of events, c) minimizes physical system dynamics estimation error, and d) ensures that simulation progresses even if the physical dynamics are ill-formed resulting in Zeno behavior.

6.5.1 Loss of Events

Figure 6.7 shows two cases, a) the behavior of *AccuSim* that processes events at exact time of occurrence, and b) the behavior of *FixSim*. As shown in Figure 6.7 (Case a), due to a large time step of the simulator, two events can occur within a time step. In this case, the system variable v_1 increases at time t and crosses the threshold C_1. At this time a physical event is triggered that causes v_1 to decrease at a rate proportional to v_1 following Equation 6.1. *AccuSim* processes this event and v_1 falls below C_1 and eventually reaches a stable value. However, a *FixSim* simulator due to its large time step misses this event. In such a case, v_1 continues to increase and reaches C_2. A physical event is triggered which reduces v_1 by an even greater value proportional to its magnitude. *FixSim* might process this last event and start reducing v_1 with a larger value. This might lead to instability in the variation of v_1 which is not observed by the *AccuSim*. To avoid this scenario *HyrefSim* predicts the occurrence of such events and re-adjusts simulation time step to process such events.

The event can only be lost if the event generator function $g_k(\mathcal{S}(t))$ is approaching towards a constraint $c_k \in C_c$ for the current simulation state $\mathcal{S}(t)$ and within the next time step $t + \Delta t$, $g_k(\mathcal{S}(t + \Delta t))$ crosses the constraint.

$$||g_k(\mathcal{S}(t + \Delta t)) - g_k(\mathcal{S}(t))|| \leq ||c_k - g_k(\mathcal{S}(t))||. \tag{6.3}$$

where, $||A||$ denotes the magnitude of A if A is a real number, or it denotes a vector with magnitudes of each element in A if A is a vector. If we consider that the event generator functions are linear in nature i.e., $g_k(\mathcal{S}(t)) = A_c^k \mathcal{S}(t) + B_c^k$, where A_c^k is an $n_c \times n_c$ matrix while B_c^k is an $n_c \times 1$ vector then we the time refinement to capture the lost event is given by the following theorem,

Theorem 6.5.1 Time step to prevent event loss:

Given a simulator $\{\mathcal{E}, \mathcal{V}, \mathcal{I}, \{\mathcal{A}_s, \mathcal{A}_s^{\tau_1} \ldots\}, \mathcal{B}_s, t_{sim}, \{G_c, C_c\}, \mathcal{E}_s, \epsilon, \epsilon_r\}$*, there will be no event loss iff, the simulation time step* Δt *is the minimum of all threshold crossing inter-*

vals $\Delta_k t$ *-*

$$\Delta_k t \in A_c^{k^{-1}}(\sum_{\forall i} A_s^{\tau_i}\mathcal{S}(t_{sim} - \tau_i) + B_s)^{-1}[||c_k \pm \epsilon_r - B_c^k||]. \tag{6.4}$$

Proof: *Let us rewrite Equation 6.3 as follows -*

$$g_k(\mathcal{S}(t + \Delta t)) - g_k(\mathcal{S}(t)) \leq c_k - g_k(\mathcal{S}(t)). \tag{6.5}$$

Let us consider that we have an error margin of ϵ_r *in determining the thresholds. Further, let us focus on just one function* g_k *then we can obtain* $\Delta_k t$ *such that no event generated by that function is missed. We can rewrite Equation 6.5 as follows,*

$$g_k(\mathcal{S}(t + \Delta t)) - g_k(\mathcal{S}(t)) \leq c_k - g_k(\mathcal{S}(t)) \pm \epsilon_r \tag{6.6}$$

$$g_k(\textstyle\sum_{\forall i} A_s^{\tau_i}\mathcal{S}(t_{sim} - \tau_i) + B_s) \leq c_k \pm \epsilon_r$$

$$A_c^k(\textstyle\sum_{\forall i} A_s^{\tau_i}\mathcal{S}(t_{sim} - \tau_i) + B_s)\Delta_k t + B_c^k \leq c_k \pm \epsilon_r$$

$$\Delta_k t \in (A_c^k)^{-1}(\textstyle\sum_{\forall i} A_s^{\tau_i}\mathcal{S}(t_{sim} - \tau_i) + B_s)^{-1}[c_k \pm \epsilon_r - B_c^k]$$

If we consider the other two operator in OP then we can derive the same equation as Equation 6.6. Finally, if we want to ensure that no events are lost for any condition function then we can take the minimum $\Delta_k t$ *that will ensure that we are always within the error margin* ϵ_r *of the threshold. This proves the if part of the theorem. The only if part of the theorem can be easily proven by evaluating the difference* $g_k(\mathcal{S}(t + \Delta t)) - g_k(\mathcal{S}(t))$ *by using Equation 6.4 for* Δt*.*

Main insight:

$$\Delta t \propto \frac{\text{difference in event threshold}}{\text{change in event generating function X change in continuous state}}$$

6.5.2 Clustering of Events

Case b in Figure 6.7, shows that a big time step can cause false clustering of events, when the event causing thresholds of two events are crossed within the simulation time step. Two events may occur at the same time due to two main causes: a) *true clustering*: when two events are not distinguishable in time within the specified simulation resolution

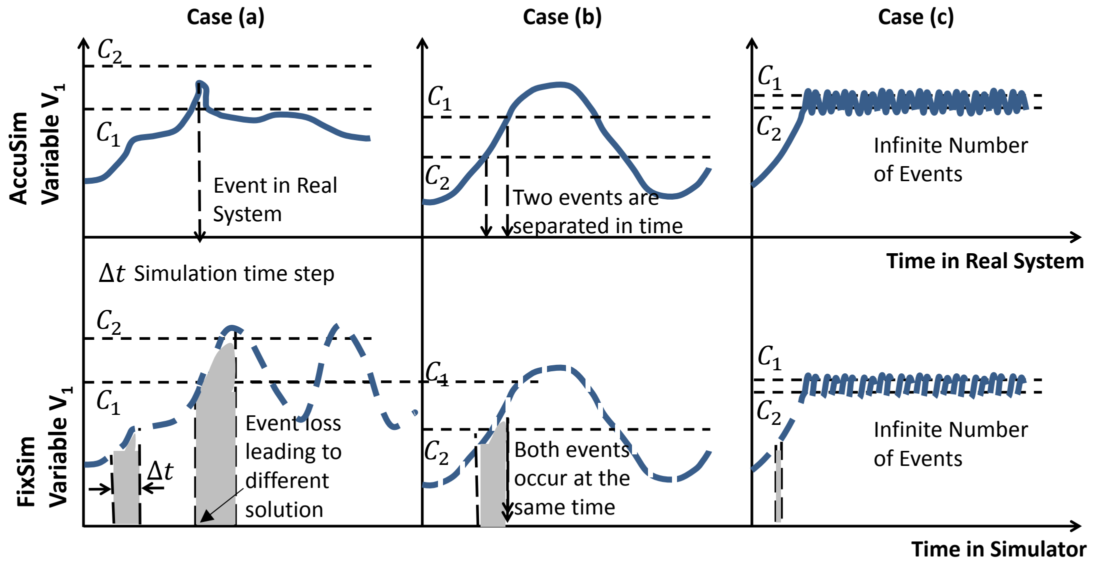


Figure 6.7: Event Loss (Case a), Event Loss and False Clustering (Case b) and Zeno Behavior (Case C) in Physical Systems. The Top Row Shows an Ideal Simulator with Very Small Time Step. The Bottom Row is *Fixsim* with a Finite Time Step.

of ϵ, and b) *false clustering*: the event causing conditions of two events are satisfied within the simulation time step. For the first condition *HyrefSim* uses the following theorem to detect whether events are distinguishable.

Theorem 6.5.2 Evaluate true clustering of events: *Two events e_i and e_j occurring due to satisfaction of event generation thresholds $g_i(\mathcal{S}(t_{sim} + \Delta t)) > c_i$ and $g_j(\mathcal{S}(t_{sim} + \Delta t)) > c_j$ occur within ϵ time difference iff $A^i_c = A^j_c$ and $||c_i - c_j|| = ||B^i_c - B^j_c|| \pm \epsilon_r$.*

Proof: *Let us consider that at the current simulation time t_{sim} of the simulator the continuous state variables of the system are $\mathcal{S}(t_{sim})$. If the simulation is executed with a time step of Δt then two constraints $g_i(\mathcal{S}(t_{sim} + \Delta t))OPc_i$ and $g_j(\mathcal{S}(t_{sim} + \Delta t))OPc_j$ are satisfied for any time step Δt. This can only happen if the rate of change of functions g_i and g_j are equal with respect to changes in $\mathcal{S}$, i.e., $A^i_c = A^j_c$. Further, the amount of change required* *to satisfy the constraints also have to be equal. Hence, we get the following equations -*

$$\|c_j - g_j(\mathcal{S}(t_{sim}))\| = \|c_i - g_i(\mathcal{S}(t_{sim}))\| \pm \epsilon_r \tag{6.7}$$

$$\|c_j - c_i\| = \|A_c^j \mathcal{S}(t_{sim}) + B_c^j - A_c^i \mathcal{S}(t_{sim}) - B_c^i\| \pm \epsilon_r$$

$$\|c_j - c_i\| = \|B_c^j - B_c^i\| \pm \epsilon_r$$

The only if part is an easy extension since $A_c^i = A_c^j$.

Main insight: *Two events are truly clustered in time if the event generation functions have same derivatives, and the difference in event thresholds are within error margin ϵ_r.*

To distinguish between two events in time, *HyrefSim* derives the time step as follows-

Theorem 6.5.3 Temporal distinction of falsely clustered events: *Consider two physical events e_i and e_j occurring due to satisfaction of thresholds by event generating functions $g_i(\mathcal{S}(t_{sim} + \Delta t)) > c_i$ and $g_j(\mathcal{S}(t_{sim} + \Delta t)) > c_j$. If these events are apart in time by at least ϵ then they can be distinguished using a simulation time step given by,*

$$\Delta t < (A_c^j - A_c^i)^{-1} (\frac{dS(t)}{dt})^{-1} \|c_j - c_i - (B_c^j - B_c^i)\|. \tag{6.8}$$

Proof: *If two events are distinguishable by at least ϵ time then there exists a time δt such that,*

$$\|c_j - g_j(\mathcal{S}(t_{sim}))\| > \|g_j(\mathcal{S}(t_{sim} + \delta t)) - g_j(\mathcal{S}(t_{sim}))\| \tag{6.9}$$

$$\|c_i - g_i(\mathcal{S}(t_{sim}))\| < \|g_i(\mathcal{S}(t_{sim} + \delta t)) - g_i(\mathcal{S}(t_{sim}))\| \tag{6.10}$$

for some events e_i and e_j. Let us consider the $\geq$ and $>$ operators and replace $g_j(\mathcal{S}(t_{sim} + \delta t)) = A_c^j(A_s\mathcal{S}(t_{sim}) + B_s)\delta t + B_c^j$. If we add both sides of the equations in Equation 6.9 and consider that event j occurs after event i then we get, -

$$\begin{aligned} c_j &+ A_c^i(\sum_{\forall i} A_s^{\tau_i} \mathcal{S}(t_{sim} - \tau_i) + B_s)\delta t + B_c^i \\ &> c_i + A_c^j(\sum_{\forall i} A_s^{\tau_i} \mathcal{S}(t_{sim} - \tau_i) + B_s)\delta t + B_c^j \\ \delta t &< \frac{c_j - c_i - (B_c^j - B_c^i)}{(A_c^j - A_c^i)(\sum_{\forall i} A_s^{\tau_i} \mathcal{S}(t_{sim} - \tau_i) + B_s)} \end{aligned} \tag{6.11}$$

We arrive at the same type of equation if we consider the $\leq$ and $<$ operator in OP. Thus, any δt satisfying Equation 6.8, will separate the events e_i and e_j in time.

Main insight:

$\Delta t \propto$ difference in event threshold$\div$(difference in the change in event generating function $\times$ change in continuous state)

6.5.3 Simulator Progress

In a malformed hybrid simulation, there may exist two events $e_1 \in \mathcal{E}$ and $e_2 \in \mathcal{E}$, which have conflicting control decisions. That is e_1 increases the rate of increase of a variable v_1 and e_2 decreases it by a large value. Let us also consider that the threshold conditions for these events c_1 and c_2 respectively are such that $c_1 - c_2 \leq \dot{\mathcal{S}(t_{sim})} \times \delta t$, for a very small δt. Then there can be infinitely many events occurring within a very small time as shown in Case c of Figure 6.7. This is an artifact of time refined simulation since for an FHT simulator the time steps are fixed and even if infinite events occur theoretically they are rejected. *HyrefSim* tackles this problem by limiting the smallest simulation time step to a chosen value.

6.5.4 Error Bound of Simulation

The continuous variables in *HyrefSim* are evaluated at every time step. Time refining ensures that the simulator does not miss any events. Discretization into time steps can lead to errors in estimating the values of the continuous variables. The time refinement in *HyrefSim* is done on top of the simulation time step computation used by traditional FHT simulators such as Simulink. The time step of the simulator is selected such that the error in differential dynamics estimation is less than the error margin ϵ_r.

For linear time invariant systems of the form,

$$\dot{\mathcal{S}} = A_s\mathcal{S} + B_s, \tag{6.12}$$

where $\mathcal{S}$ is the n_c dimensional vector obtained from the transient model, the simulation time step is given by,

$$\Delta t \leq \frac{\epsilon_r}{||A_s||_\infty ||\mathcal{S}(t)||_\infty + ||B_s||_\infty}, \tag{6.13}$$

where $||A||_\infty$ is the maximum element in A. The value of Δt depends on the maximum value of the elements in $\mathcal{S}(t)$. The maximum value of $\mathcal{S}(t)$ can be determined from the simulation settings and is a user specified parameter. Since the time step of *HyrefSim* is less than the Δt it has similar error bound as traditional simulators. However, we will see in the examples that *HyrefSim* actually reduces the error due to accurate reduced time steps for handling events.

6.6 Simulation Algorithm

In this section, we derive the simulator algorithm from the properties discussed in the previous section. At the most basic level the simulator maintains an event queue, which is populated initially with random discrete computing events generated using the discrete event generator function. The simulator executes a loop where it first de-queues an event, processes its operation on the discrete and continuous variables, and updates the event queue according to future events. It then finds out the next event time and calls the continuous function simulator, FHT, to execute from the current simulation time to the next event time. The recurring steps taken by FHT are as follows:

a) It computes the time step Δt based on the simulation resolution ϵ_r according to Equation 6.13. This is a standard step taken by most physical system simulator.

b) The FHT then employs the time refinement method to determine whether it has lost any events or not. It first computes the threshold crossing intervals $\Delta_k t$ for all event generating functions $g_k \in G_c$ and thresholds c_k using Equation 6.4. If the minimum threshold crossing interval $min(\Delta_k t) \forall g_k \in G_c$ is greater than the current time step Δt determined by the FHT then we can conclude that no event occurs within the time step. *HyrefSim* keeps the simulation time step as Δt. Otherwise it uses the minimum threshold crossing interval $min(\Delta_k t)$ as the time step to process the nearest event.

c) It then computes the continuous variables $\mathcal{S}(t + \Delta t)$.

d) The condition generation functions G_c is then evaluated. The condition generation can lead to generation of multiple events at the same time.

e) In such a scenario, the time refining methodology is called again to distinguish between events. The simulator first checks the equality constraint in Theorem 6.5.2. If the equality constraint is satisfied within the simulation resolution ϵ then it uses a random priority for the events. Else it uses Equation 6.8 to further refine the time step. It goes back in time by Δt and uses a finer time discretization with step size determined from Equation 6.8, which distinguishes the two events. Note that the simulator only re-simulates from current time t to time $t + \Delta t$ with the finer time step and then reverts back to the coarser time step determined from Equation 6.13.

f) On each event generation, the FHT pushes the event into the event queue and passes the control back to the ED immediately to process the events.

6.6.1 Simulator Speedup

HyrefSim uses time refinement as opposed to *FixSim*. *FixSim* is expected to operate faster than *HyrefSim* since it does not reduce the time step to account for events at their correct time of occurrence. The degradation in execution time is given by the following theorem:

Theorem 6.6.1 Simulator execution time degradation: *The degradation of execution time of HyrefSim with respect to FixSim is bounded and* **does not increase with simulation time, number of control modes, or number of continuous variables**. *The degradation is bounded by a factor directly proportional to the square of the error margin ϵ_r, and maximum difference in the rate at which any two constraints in G_c are approached for* *unit change in $\mathcal{S}$ and inversely proportional to the smallest difference in event generating thresholds.*

Proof: *FixSim computes the time step of simulation Δt following Equation 6.13. It then increments simulation time by Δt and finishes the simulation when current time is equal to T_{sim}, the total simulation time.*

On the other hand, the hybrid simulator can start iterations using a time step Δt from Equation 6.13. It has the capability to do time refining so that if p is the total number of times there is a possibility of lost event, it can refine just one time step Δt by δ_c. Thus, it runs a total of $\frac{T_{sim}}{\Delta t} + \frac{p\Delta t}{\delta_c}$ number of iterations. If the HyrefSim is poorly configured then it might have zeno behavior. However, to eliminate zeno behavior we need to distinguish events in time by a finite temporal interval. Thus, we can consider that in the worst case HyrefSim has events that are separated by the minimum time δt obtained from Equation 6.8. Thus, the execution time degradation of HyrefSim is given by -

$$\begin{aligned} d_x &\leq \frac{T_{sim}/\Delta t + \frac{T_{sim}}{\delta t}\frac{\Delta t}{\delta_c}}{T_{sim}/\Delta t} \\ &\leq 1 + \frac{\Delta t^2}{\delta t \delta_c} \\ &< 1 + \frac{\epsilon_r^2 max_{\forall i,j}(\|A_c^j - A_c^i\|_\infty)}{min_{\forall i,j}(c_j - c_i) min_{\forall k}(c_x - \|B_c^k\|_\infty)} \end{aligned} \tag{6.14}$$

Main insight: *The degradation in speed up is directly proportional to the maximum difference in rate at which two thresholds are approached and inversely proportional to the minimum difference in thresholds and the minimum state change needed to cross a threshold.*

Theorem 6.6.1 is intuitively true since larger the error margin larger can be the time discretization and hence the simulation executes faster. Further, if constraints are approached faster then after the time refining process potentially lost events will be captured earlier and the hybrid simulator can stop the time refining process and continue executing with larger time steps.

6.7 Usage and Performance Analysis of *HyrefSim*

In this paper, we show two examples, network controlled infusion pump, and body implanted sensor networks. Three different simulation settings were considered for comparison:

- *FixSim*: This simulator has fixed time step and can adjust the time step to reduce error in estimating the differential equations. It resembles traditional simulators such as Simulink that have the capability to dynamically change the simulation time step. However, this simulator is agnostic of the error in the timing of events.
- *HyrefSim*: This is the proposed simulator that has the time adjustment based on estimation error of continuous dynamics as well as accurate prediction of event timings. The initial simulation time step of this simulator is kept to be same as *FixSim*.
- *AccuSim*, *FixSim* with fine time step: This simulator is similar to *FixSim* but the time step is much lower than *FixSim*. The assumption is that for a very small time step, the *FixSim* can estimate event timings accurately and can also reduce the estimation error in the differential equations. Hence, we can consider *FixSim* with high resolution as an optimal simulator.

To evaluate *HyrefSim* we define the following metrics: a) accuracy or error in predicting continuous variables, b) average event delay, c) average event queue size, d) event loss: average number of events processed by *HyrefSim* that are not processed by *FixSim* but processed by *AccuSim*, and average number of events processed by *FixSim* that are not processed by *HyrefSim* and *AccuSim*, and e) execution time of simulator and speedup with respect to *AccuSim* and *FixSim*. In this section, we show two examples: 1) infusion pump controller and b)sensor network.

6.7.1 Example 1: Blood Glucose Control using Infusion Pump

Automated control of blood glucose levels in human are often obtained using artificial pancreas. Artificial pancreas are distributed systems consisting of an infusion pump, glucosemeter, and a controller implemented in a mobile device such as smartphone. These distributed components are networked through the wireless communication channel and operate in a close loop to keep the drug concentration in the human blood within recommended limits. The different components of the automated control system may have skews in the clock rates as well as data transfer rates. This results in transport delays in the sensed values of glucosemeter and actuation delay in infusion. Thus, the continuous dynamic equation that represents the blood glucose level in the blood sensed by the glucosemeter due to infusion from the pump is of the form of a DDE (Equation 6.15).

$$\begin{aligned} \dot{y_1} &= A_p y_1 + B_p \dot{Q} z_2 + B_p u(t - T_i), \\ z_1 &= C_p y_1(t - T_p), \\ \dot{y_2} &= A_s y_2 + B_s \dot{Q} z_1, \\ z_2 &= C_s y_2(t - T_r). \end{aligned} \tag{6.15}$$

Here y_1 and y_2 are the state space variables of the equation. y_1 consists of vectors of left heart, lung blood, lung tissue and right heart compartments through which infused drug passes. Newly infused drug merges with recirculated drug from Vessel Rich Group, Muscle, Fat and Residual drug which is represented by y_2 state space variable vectors. A_p, A_s, B_p, $\dot{Q}$, C_s, and C_p are constants. z_1 is the drug concentration in the blood while z_2 is the arterial drug concentration. The initial infusion rate $u = x_0$ is the input to the model and the output is the drug concentration in the blood. The time delays related to the infusion input (T_i) is due to the delay in actuation, the cardio-pulmonary transport delay T_p and the arterial, capillary and venous transport delays T_r manifest the delay in sensing the blood glucose level by the glucosemeter.

The discrete controller of the infusion pump has five states: a) basal, where infusion rate is I_0 and the blood glucose rate is between 120 ug/dl and 70 ug/dl, b) braking, where infusion rate is a fraction f of I_0, and blood glucose level goes below 70 ug/dl but is still above 20 ug/dl, c) correction bolus, where infusion rate is incremented by I_{cb}, and the blood glucose level is between 120 ug/dl and 180 ug/dl, d) bolus, where the infusion rate is incremented by $I_b > I_{cb}$, and the blood glucose level is above 180 ug/dl, and e) stop, when the infusion is stopped i.e., $I_b = 0$, since the blood glucose level drops below 20 ug/dl.

In addition to these states there can be discrete random events related to consumption of meal. The user may opt to take in a bolus shot of insulin 30 mins before taking a meal. According to recent research, food intake behavior of human users have Markovian properties. Hence, the food intake event generator was modeled using a Markov chain. There are three different levels of meal: small typically signifying breakfast, medium corresponding to dinner, and large corresponds to a midday lunch. The corresponding Markov chain representation has three states: a) small, b) medium, and c) large. The transition probabilities are given by the matrix $A = \begin{pmatrix} 0.1 & 0.2 & 0.7 \\ 0.5 & 0.4 & 0.1 \\ 0.6 & 0.3 & 0.1 \end{pmatrix}$.

The assumption is that a small meal is more likely to be succeeded by a large meal. A large meal is more likely to be succeeded by a small meal. Medium meals are less likely to be succeeded by a large meal. The time between successive meals is obtained from an exponential distribution with a mean time between meals of 8 hrs. For each meal size the user requests a bolus dosage of insulin proportionate to the size of the meal.

Using the automated infusion control system we compare the performance and execution time of *HyrefSim* and *FixSim*. The initial simulation time step of *FixSim* is kept at

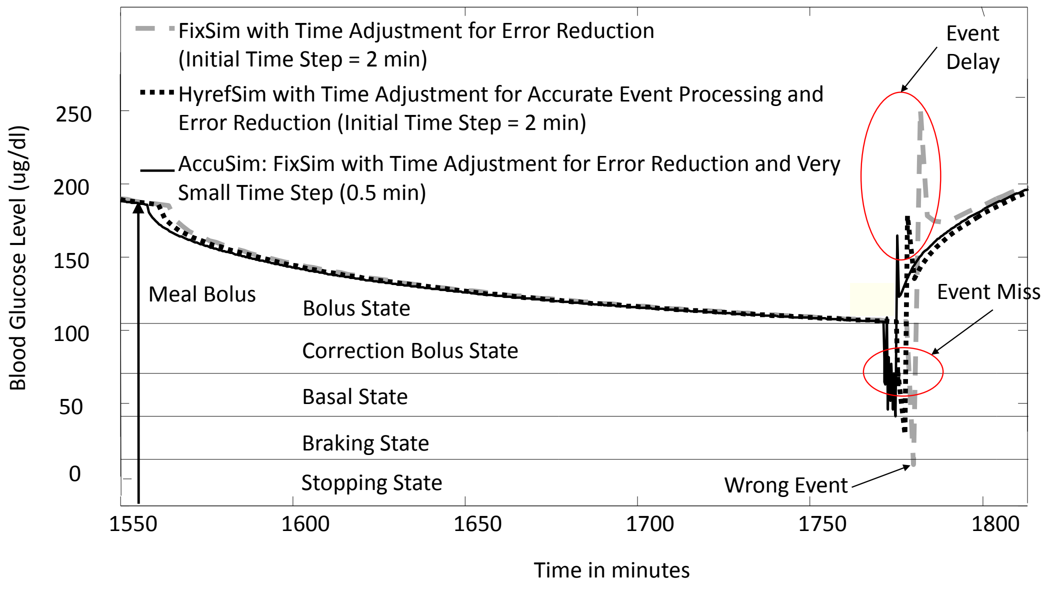


Figure 6.8: Blood Glucose Level Estimation using *HyrefSim* and *FixSim*.

2 mins, which guarantees an error less than 10% in estimating the continuous dynamics of Equation 6.15. The time step of *AccuSim* is kept at 0.5 seconds. The purpose of the first experiment was to show conditions when there are event losses, delays, and wrong event processing. Figure 6.8 shows a snippet of a simulation of two days worth of infusion control.

Event delay: The snippet shows a case where a meal bolus has been administered and the blood glucose level starts to fall. Initially there is a short delay corresponding to the normal insulin action delay of around 10 mins. As seen from Figure 6.8, *FixSim* with high resolution is the first one to detect the drop in blood glucose level as expected. *HyrefSim* predicts the drop closer to the actual time than *FixSim*. This is attributed to the event detection algorithm in *HyrefSim*. Figure 6.9 shows the average event delay for *HyrefSim* and *FixSim* with respect to *AccuSim*. *FixSim* always has higher delays.

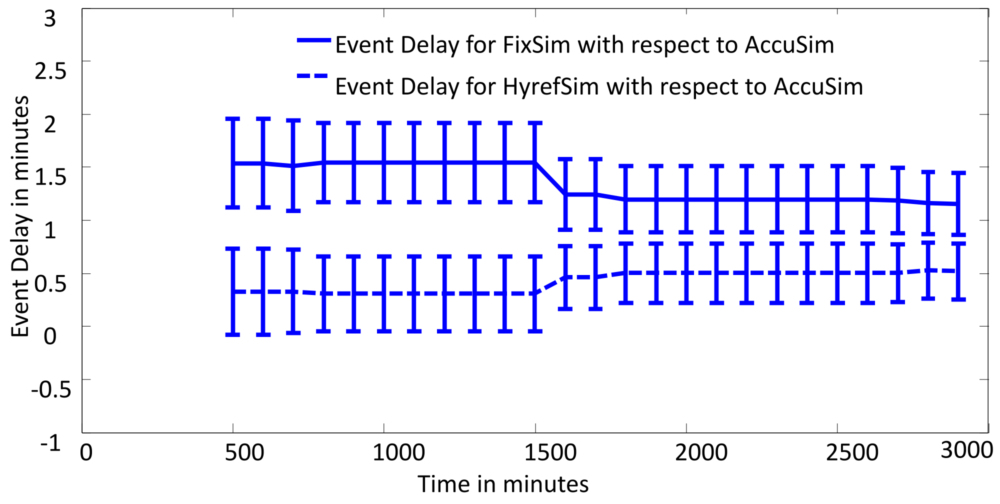


Figure 6.9: Average Delay along with Error Bar in Processing Events by *HyrefSim* and *FixSim* Over Time with Respect to *AccuSim*.

Event loss: More interesting cases are observed when the blood glucose level falls below the bolus state condition and reaches the correction bolus level. The *HyrefSim* captures this event earlier than the *FixSim*, and hence reduces the bolus level and decelerates the drop in blood glucose level. However, *FixSim* detects this event much later and hence even if it reduces the bolus during the correction bolus level the fall rate of blood glucose level is much higher. Due to this higher fall rate, in the next time steps the glucose level falls higher in *FixSim* than *HyrefSim*. Thus, when *HyrefSim* capture the transition from correction bolus to braking state, *FixSim* fails to capture such transition and goes straight to stopping state. Since *HyrefSim* captures the braking state transition, it reduces the bolus amount and further decelerates the glucose fall. Therefore, *HyrefSim* predicts a higher blood glucose concentration than *FixSim* and does not process the stopping state. As seen from Figure 6.8, *HyrefSim* simulation is closer to *FixSim* with higher resolution. The average number of events lost for different lengths of run is shown in Figure 6.10. We see that *HyrefSim* does not loose events while *FixSim* on an average looses 35% of the events.

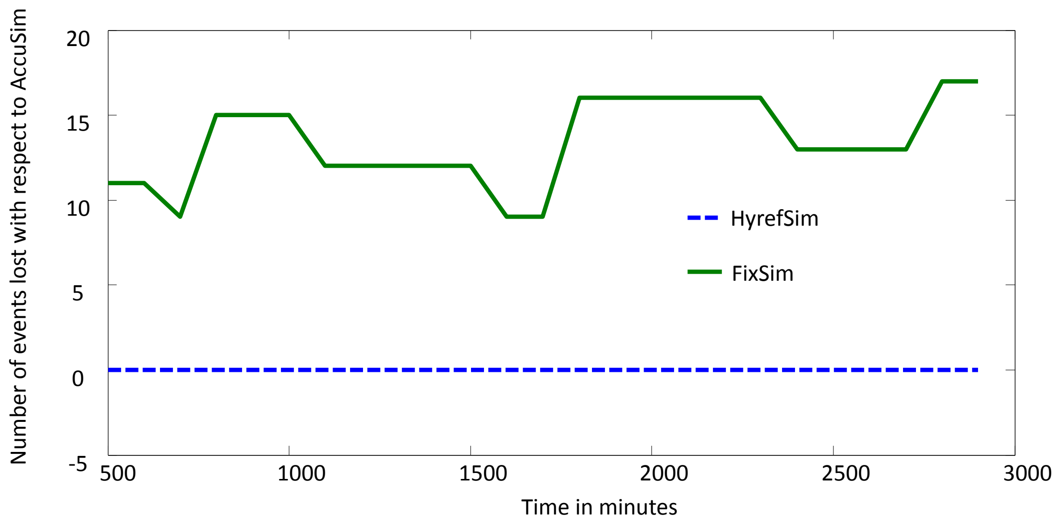


Figure 6.10: Number of Events that are Processed by *Accusim* but Lost by *Hyrefsim* and *Fixsim* Over Time.

Accuracy: Since *HyrefSim* simulator predicts events much earlier than *FixSim*, it predicts 65mL lower insulin infusion over the period of 2 days (Figure 6.11). The amount of insulin infused by a control system is a metric of safety for the insulin pump. Insulin control systems that achieve the same control outcomes with lesser amount of insulin delivery are preferred. Thus, a control configuration can be deemed un-preferable by *FixSim* while it is in practice a good design. Further, error of *FixSim* increases nonlinearly with respect to *HyrefSim* (Figure 6.11). The error in simulation i.e., the difference in total amount of insulin delivery with respect to *AccuSim* for *HyrefSim* and *FixSim* is shown in Figure 6.12.

Average event queue length: In presence of discrete random meal events, multiple events can occur within a simulation time step of *FixSim*. *HyrefSim* has the capability of temporally distinguishing such events even if they occur within one time step. The average event queue length is a good indicator of false clustering of events. Figure 6.13, shows the average length of the event queue for *FixSim* and *HyrefSim* simulation cases. The solid line for *HyrefSim* coincides with the dashed line in *FixSim* for most of the cases, however, in

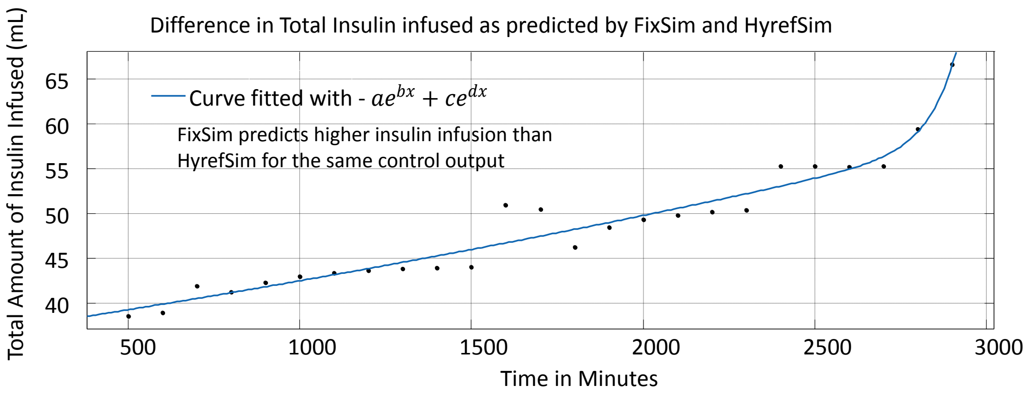


Figure 6.11: Variation of Simulation Error of *FixSim* with Respect to *HyrefSim* v.s. Simulation Length.

some cases *FixSim* has higher queue size than *HyrefSim*. This gives a clear indication of false clustering of events.

Speedup: Figure 6.14, shows the execution time of *HyrefSim* and *FixSim* for different lengths of infusion delivery. The maximum degradation of speedup for *HyrefSim* is 15%, which is due to the fact that *HyrefSim* spends time in computing refined time steps for the simulator and also processes more events than *FixSim*. This is in accordance with Equation 6.14.

6.7.2 Example 2: Body Sensor Networks

We simulated various routing algorithms for the example discussed in Section 7.1, for example computing minimum temperature, scheduling workloads, moving cluster head, reducing radio power level. The interaction of implanted sensors with human body causes temperature in tissues surrounding the sensors. The temperature rise in the human tissues is governed by the following heat transfer processes: 1) heat transfer due to radiation from the sensors which depends on the operating temperature of the processor, 2) conductive heat transfer from the processor, 3) electro-magnetic radiation absorption by the body part, and

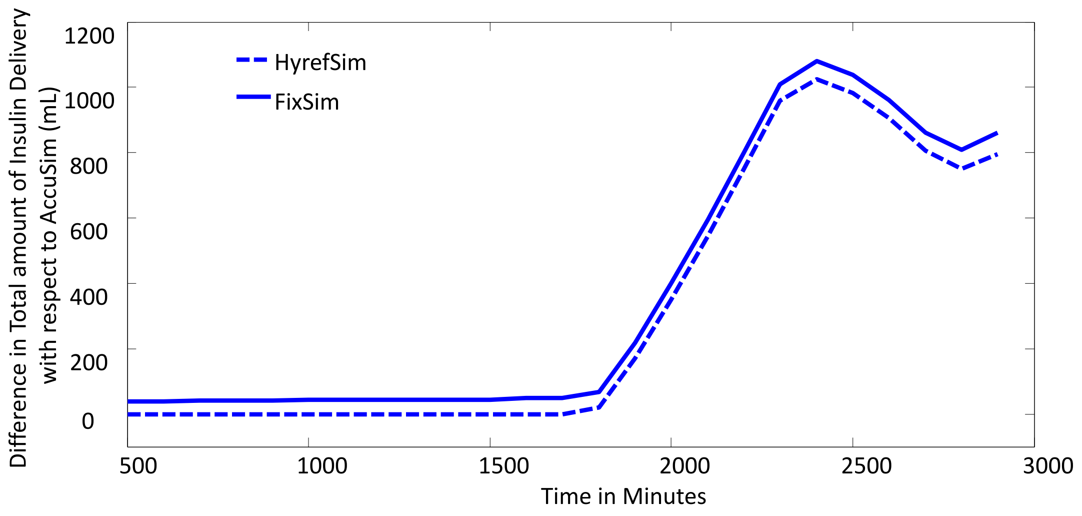


Figure 6.12: Simulation Error i.e., Difference in Total Amount of Insulin Delivery with Respect to *AccuSim* v.s. Simulation Length for *HyrefSim* and *FixSim*.

4) convective heat extraction by blood. These physical processes are combined in a single partial differential equation known as Penne's equation [74] which gives the temperature variation of the human body part over space and time as in equation 6.2 .

To simulate Penne's bioheat equation using FHT simulator, we have discretized it over $N \times N$ grid. It is discretized over time and space using electromagnetic modeling technique, finite difference time domain(FDTD) [75],

$$\begin{aligned} \dot{T^m(i,j)} &= \left[-\frac{(b\delta^2+4K)}{\rho C_p \delta^2}\right] T^m(i,j) + \frac{SAR}{C_p} + \frac{b}{\rho C_p} T_b \\ &+ \frac{P_c}{\rho C_p} + \frac{K}{\rho C_p \delta^2}[T^m(i+1,j) + T^m(i,j+1) \\ &+ T^m(i-1,j) + T^m(i,j-1)], \end{aligned} \tag{6.16}$$

Equation 6.16 is analogous to Equation 6.17 which is in the form of linear time invariant differential equation.

$$\dot{T_m} = AT_m + B, \tag{6.17}$$

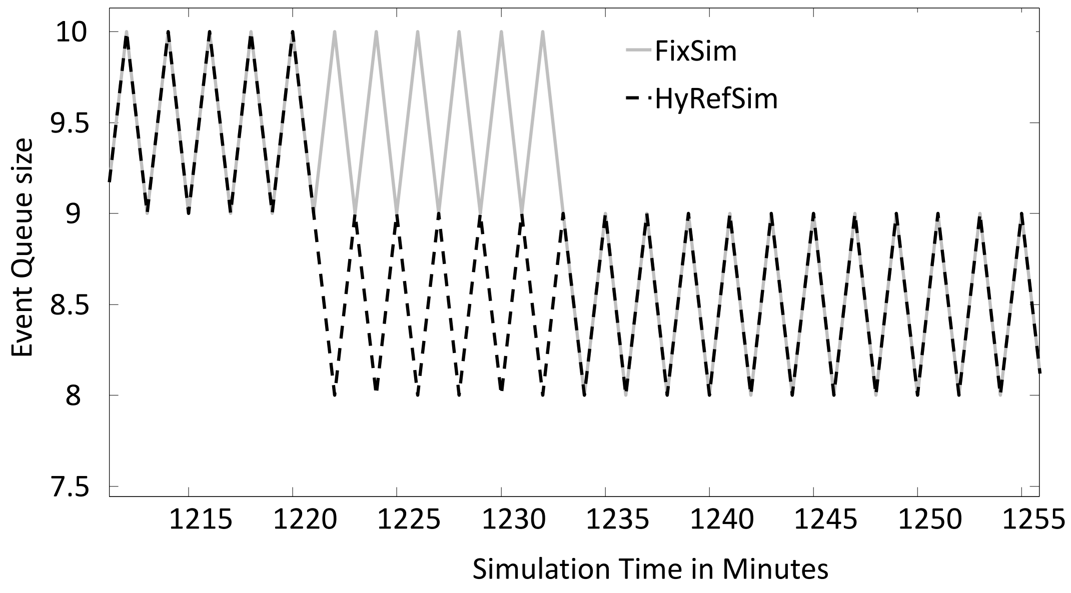


Figure 6.13: Variation in Event Queue Length with Respect to Time for *HyrefSim* and *FixSim*.

where, matrix A will have dimensions N^2XN^2 and its each row will be as, $A =$
$$\begin{array}{ccccccccc} \cdot\cdot & X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 & \cdot\cdot \\ \left(\cdot\cdot\right. & \alpha & \cdot\cdot & \alpha & \beta & \alpha & \cdot\cdot & \alpha & \left.\cdot\cdot\right) \end{array}$$
where, $\alpha = \frac{K}{\rho C_p \delta^2}$, $\beta = -\frac{(b\delta^2+4K)}{\rho C_p \delta^2}$, $X_1 = N(j-1)+i$, $X_2 = N-2$, $X_3 = N(j+1)+i$, $X_4 = Nj+i$, $X_5 = Nj+(i-1)$, $X_6 = N-2$, $X_7 = Nj+(i+1)$. The dotted elements of matrix A are all zero. The computed matrix B of dimension N^2X1 is as follows,
$$B = \begin{array}{c|c} & 1 \\ 1 & \frac{SAR}{C_p} + \frac{b}{\rho C_p}T_b + \frac{P_c}{\rho C_p} \\ \vdots & \vdots \\ N^2 & \frac{SAR}{C_p} + \frac{b}{\rho C_p}T_b + \frac{P_c}{\rho C_p} \end{array}$$

By simulating Equation 6.16, the events such as skin temperature increased beyond threshold can be detected. The networking decisions made using events generated by continuous dynamics of the human body ensure safety of the overall system. All the events generated from discrete and fluid-flow workload as well as from continuous dynamics of

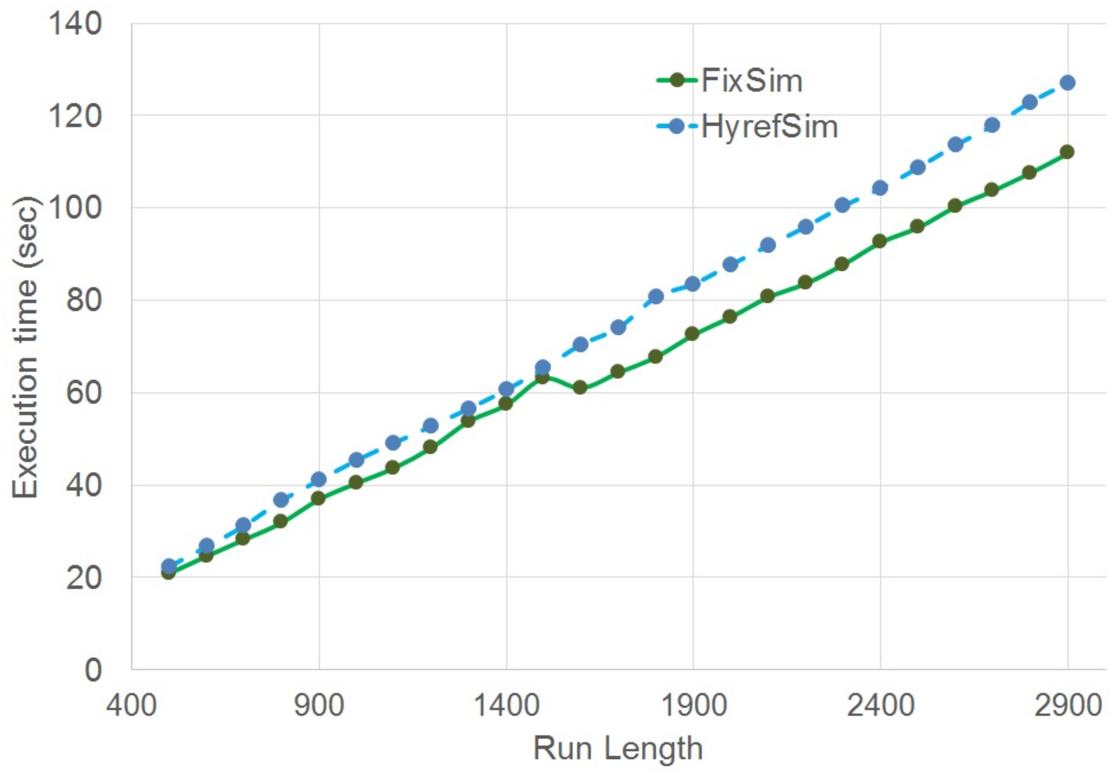


Figure 6.14: Simulation Execution Time of *HyrefSim* and *FixSim* for Different Real Time Simulation Lengths.

the human body are used in making network decisions using networking algorithms. Using this example, we studied various networking algorithms for the example BSN setting such as: (i)Thermal map of the BSN for a given leader sequence (ii)Thermal map of the BSN for defined tissue temperature threshold limit (iii)Thermal map of the BSN to reduce radio power by half Figure 6.15 shows the simulation results for these algorithms simulated using proposed hybrid simulator.

Event Loss: Figure 6.16 shows the number of events lost by both *HyrefSim* and *FixSim*. The figure shows that *HyrefSim* processes all the events while *FixSim* ignores many.

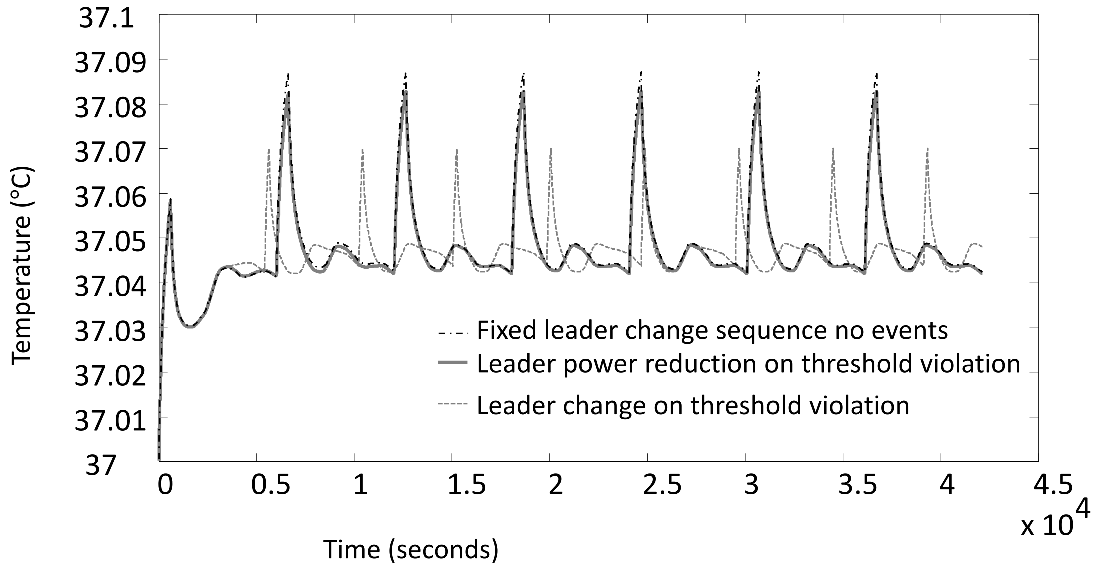


Figure 6.15: BSN Simulation Results with Different Networking Algorithms.

Event delay: Figure 6.17 shows the average delay in processing events for both *HyrefSim* and *FixSim* with respect to *AccuSim*. On an average *HyrefSim* processes events at times more close to times processed by *AccuSim*.

Average false event clustering: The number of events falsely clustered by a simulator can be found out by monitoring the average size of event queue. From the event loss results we see that *HyrefSim* computes all events that *AccuSim* would have processed. Figure 6.18 shows that the average length of the event queue for *HyrefSim* is much lower than that of *FixSim*. This shows that *HyrefSim* avoid false clustering of events.

6.8 Hardware Assisted MMA Physical System Emulation

Hardware implementation of physical systems in MMA will help to test the physical behavior of the overall system to ensure safety. Physical systems e.g. human physiology models are represented in the form of differential equations. In order to verify working of MMA before its actual implementation, these differential equations are required to be

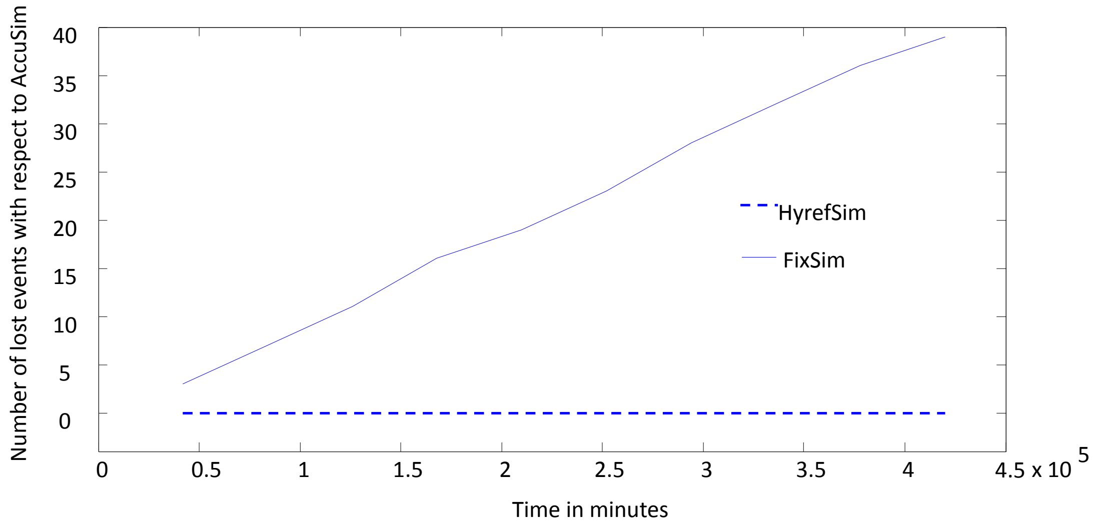


Figure 6.16: Number of Events Lost by *HyrefSim* and *FixSim*.

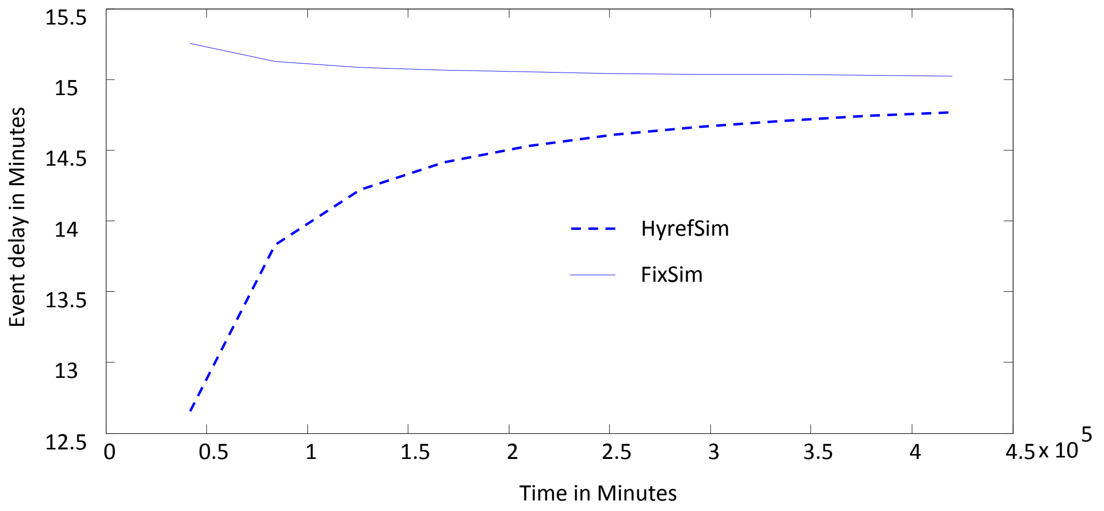


Figure 6.17: Average Delay in Event Processing for *HyrefSim* and *FixSim*.

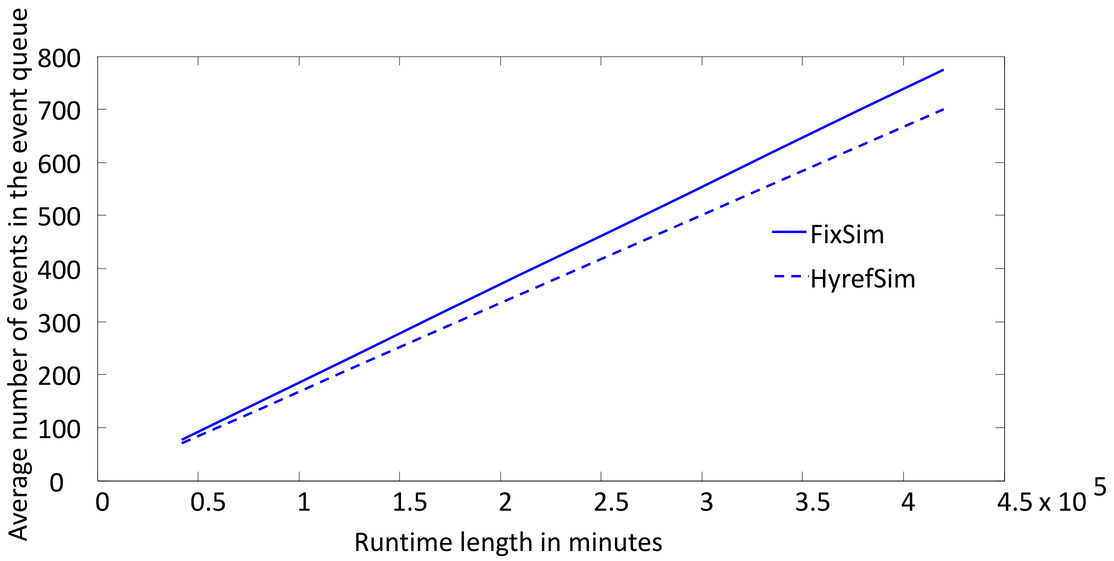


Figure 6.18: Average Number of Events in the Event Queue for *HyrefSim* and *FixSim* Over Different Simulation Runtimes.

simulated. Hardware implementation of such systems speeds up the process of simulation by order of magnitude as compared to software.

Pennes bioheat equation [74] is discretized over time and space using electromagnetic modeling technique, finite difference time domain(FDTD) [75] as,

$$\begin{aligned} T^{m+1}(i,j) &= \left[1 - \tfrac{\delta_t b}{\rho C_p} - \tfrac{4\delta_t K}{\rho C_p \delta^2}\right] T^m(i,j) \\ &+ \tfrac{\delta_t}{C_p} SAR + \tfrac{\delta_t b}{\rho C_p} T_b + \tfrac{\delta}{\rho C_p} P_c \\ &+ \tfrac{\delta_t K}{\rho C_p \delta^2}\left[T^m(i+1,j) + T^m(i,j+1) + T^m(i-1,j) + T^m(i,j-1)\right] \end{aligned} \tag{6.18}$$

Diffusor network can be implemented in field programmable analog arrays (FPAA) to represent physiscal system using differential equations [85, 86, 87, 88]. Diffusor network circuit can be simplified in resistor grid circuit as shown in Figure 6.19.

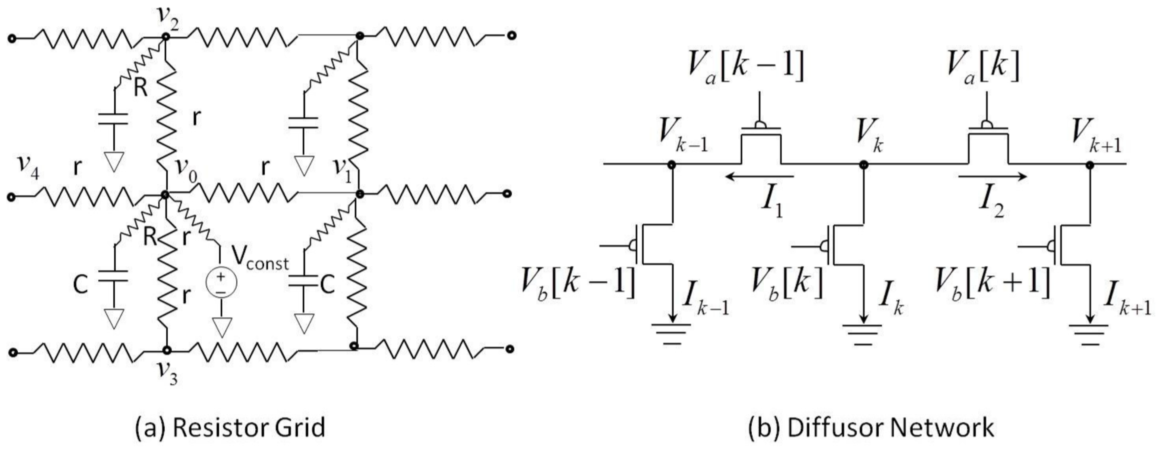


Figure 6.19: Resistor Grid and Diffusor Network Circuits for Representing Analog Behavior of Physical System.

By applying Kirchoff's Current law to resistor grid circuit (Figure 6.19 a), we get following equation,

$$\begin{aligned} V_0 = {} & \tfrac{rr_1}{R(4r_1+r)} e^{\frac{-t}{RC}} V_0 \\ & + \tfrac{r}{4r_1+r} V_{const} \\ & + \tfrac{r_1}{4r_1+r}(V_1 + V_2 + V_3 + V_4) \end{aligned} \tag{6.19}$$

For establishing equivalence between equations (6.19) and (6.18), we need to consider voltage and temperature as equivalent quantities. By putting the appropriate values of physiological parameters in equation (6.18) and comparing the terms with equation (6.19), we can calculate values of components in the resistor gird circuit.

We have simulated the 50X50 resistor grid circuit in Cadence ICFB environment to get values of voltages at each node in the circuit. Figure 6.20 shows the MATLAB plot of these values.

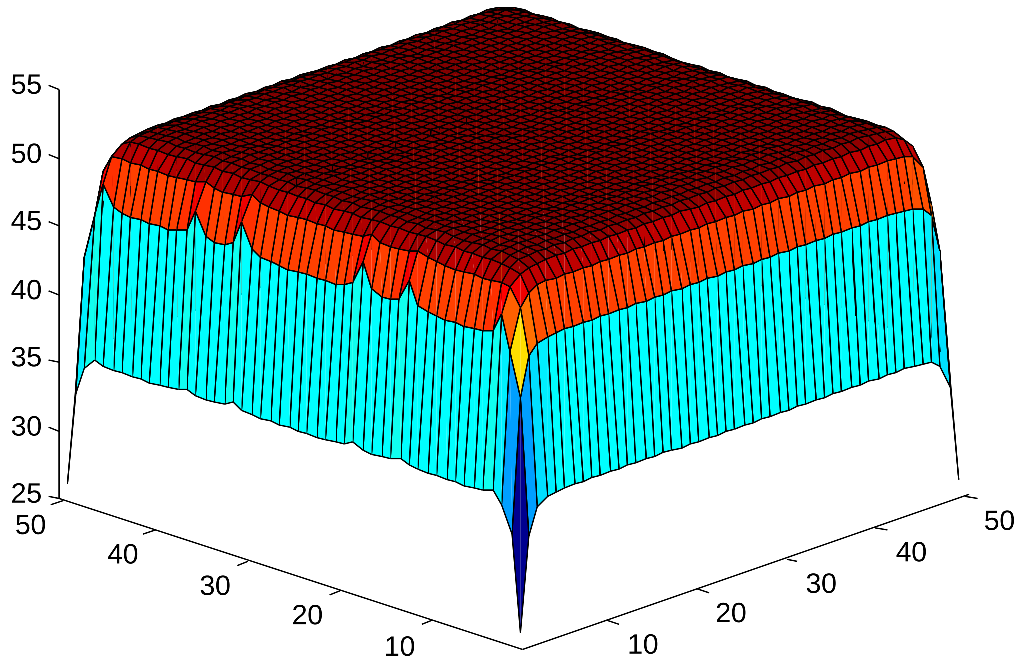


Figure 6.20: 50X50 Resistor Grid Circuit Output Plot

The plot verifies the simulation result with the actual physical behavior of human physiology according to Penne's Bio Heat equation.

Chapter 7

EVIDENCE GENERATION: MMA DESIGN VERIFICATION - THEORETICAL APPROACH

7.1 Introduction

Mobile medical apps (MMAs) work closely with human physiology which might lead to safety hazards such as heating of human skin or hypoglycemia due to low insulin dose in case of infusion pump app. To avoid these issues, the design of the MMA should be verified before its actual implementation. Typically a model based approach is taken to verify safety of a MMA design. In this regard, this research considers verification of MMA design using formal methods which incorporates models of human physiology and MMA control algorithms [89]. This method is based on theoretical safety verification of the MMA controller software. It is more rigorous and time consuming than simulation method for system design verification, however it provides eternal safety guarantees.

7.2 Model Checking using Reachability Analysis

When the smartphone acts as a controller to an actuator device, it obtains feedback from physiological signals to determine the control inputs to the actuator. Thus, it directly interacts with the human physiology typically in a complex non-linear manner. In this regard, poor controller design can cause instabilities in the human physiology leading to hazardous conditions such as hypoglycemia. Thus the combined analysis should be done on continuous dynamics of human physiology and discrete dynamics of controller software. Hybrid Automata(HA) is able to represent both continuous and discrete dynamics of the system. Reachability analysis on hybrid automata computes the values of continuous variables of the system with a given controller at any point in time. These values are further used to check instability in controller design. However, a hybrid automata representation of the interaction between smartphone and physiology will be non-linear and have multiple independent dimensions. Current hybrid system research have given limited focus on non-linear hybrid automata (discussed in Related works). The proposed research will perform reachability analysis of non-linear hybrid automata and use it to perform safety checks of MMAs.

Hybrid systems are important modeling abstractions for analyzing continuous evolution of a system under supervision of a discrete decision algorithm. Typically they are useful for checking system evolution against safety requirements of cyber-physical control systems. Reachability analysis of hybrid systems estimates the *reach set*, which comprises of all possible valuation of the system variables for a set of initial conditions. Although the problem of finding the *reach set* is undecidable, for linear dynamics there are efficient over-approximation techniques [90]. However, most of the mission critical cyber-physical (CP) control systems such as artificial pancreas [91], ventilators [92] have non-linear continuous dynamics. Research efforts are invested in estimating nonlinear dynamics using piecewise linearization. In some cases such as diffusion dynamics, the solution may not be efficiently interpolated using linear interpolants [93]. In such cases a shape preserving interpolation such as exponential splines [94] appears to be a better choice. In this paper, we present a reachability analysis technique for non-linear hybrid automata using exponential box splines and show their usage for the artificial pancreas CP control system example.

The biggest problem in estimating the *reach set* is to represent the initial set and compute the image of the continuous dynamics for each of the points in the initial set. This problem is mitigated for linear hybrid systems in a computationally efficient manner through the use of zonotopes, which are sets of vectors representing directions in a co-ordinate system [90]. A subset of the set of "n-dimensional" vectors of real numbers can be expressed as a linear combination of vectors in a zonotope i.e., if $Z = \{\overrightarrow{z_1}, \overrightarrow{z_2} \ldots \overrightarrow{z_m}\}$ is a zonotope then a vector in the linear span of Z can be expressed as $\overrightarrow{v} = \sum_{i=1}^{m} t_i \overrightarrow{z_i}$ such that $0 \leq t_i < 1, \forall t_i$. Zonotopes are further closed under linear transformation. If A is a real valued matrix then $A\overrightarrow{v} = \sum_{i=1}^{m} t_i A\overrightarrow{z_i}$. Thus the image of the vector v under linear transformation can be expressed as a linear combination of the vectors in zonotope $Z_A = \{A\overrightarrow{z_1}, A\overrightarrow{z_2}, \ldots A\overrightarrow{z_m}\}$. This property is utilized by the reachability analysis technique to efficiently compute the image of a set I in each step under linear dynamics of the form $\dot{\overrightarrow{v}} = A\overrightarrow{v} + B$, where B is a real valued vector. The method expresses the set I as a collection of zonotopes, finds the image of the corner points of the polygon spanned by the zonotope vectors with respect to the linear transformation, and then takes the convex hull of the image of the corners.

Zonotope based solution is not applicable for non-linear hybrid automata since the zonotopes are not closed under non-linear transformation. Driven by the simplicity and computational efficiency of such a solution researchers have considered piece-wise linearization of the non-linear continuous dynamics. To find the image of a set I it is first divided into regions corresponding to the linear pieces of the nonlinear dynamics and for each region the zonotope based solution is applied. Several methods have been proposed for piece-wise approximation based reachability analysis [55, 56, 57]. The error in estimating reach set using these methods often increases in a nonlinear fashion with respect to the simulation time and reducing the error magnitude requires larger number of pieces resulting in higher simulation time. To use the computationally efficient solution yet not loose on accuracy we have to find a non-linear counterpart of zonotopes.

In this regard, we propose a novel methodology using exponential box splines to compute image of a set of vectors with real numbers under a non-linear transformation. Exponential splines are traditionally used for curve fitting in geometric modeling [95, 96]. An important property of exponential splines in curve fitting is there shape preserving nature that is useful for estimating complex curves. In this paper, we will apply the exponential box (EB) spline based reachability analysis technique to medical control system of artificial pancreas. The artificial pancreas involve non-linear diffusion dynamics and exponential splines have been shown to work well for such dynamics [93]. In this paper, we use their curve fitting property to estimate nonlinear dynamics. Exponential box splines are distributions on smooth and continuous functions that map a continuous function to a real number. In its simplest form an exponential spline is an exponential weight function: $\frac{e^{\vec{\theta} \odot \vec{v}}}{k}$, where θ is a constant vector, and k is a real constant.

An advanced property of exponential box splines that is somewhat similar to zonotopes is that any smooth and continuous function $f(\vec{v})$ can be specified as a summation of linearly shifted exponential box splines: $f(\vec{v}) = \sum_{\vec{\alpha_i}} f(\vec{\alpha_i}) \frac{e^{\vec{\theta} \odot (\vec{v} - \vec{\alpha_i})}}{k}$. The $\vec{\alpha_i}$ are chosen is such a manner that $\vec{v}$ can be expressed as their linear combination. Once the initial choice of $\vec{\theta}$, k and $\vec{\alpha_i}$s are made, the next image can be expressed as: $f(f(v)) = \sum_{f(\vec{\alpha_i})} f(f(\vec{\alpha_i})) \frac{e^{\vec{\theta} \odot (f(\vec{v}) - f(\vec{\alpha_i}))}}{k}$. Here the exponential box spline function remains unchanged. In a sense it can be considered as a weight function similar to the t_i values for zonotopes. The $\vec{\alpha_i}$s are similar to the zonotopes that represent the initial set I in the linear case. This enables us to use an approach similar to the linear case for the reachability analysis of non-linear systems.

Before we go into the details of the EB spline based reachability analysis technique, let us first see the usefulness of this technique in reducing image estimation error and computation time by applying it to the classical non-linear system example of a Brusselator.

Example 6 *A brusselator is the model of auto-catalytic reactions and is a very common example of non-linear system used by many researchers [97]. The model of the brusselator has two variables v_1 and v_2 and an example dynamics is expressed using Equation 7.1.*

$$\dot{v_1} = 1 + v_1^2 v_2 - 2.5 v_1 \tag{7.1}$$

$$\dot{v_2} = 1.5 v_1 - {v_1}^2 v_2$$

The initial set of the brusselator is a zonotope $Z = \{(0, 2), (2, 0)\}$. *The time was discretized with an interval of 0.5s. We computed the image of the initial set for* $t = h$ *following two approaches: a) linear interpolation of the non-linear function in Equation 7.1, and b) exponential box spline based representation of the non-linear function in Equation 7.1. Note that Equation 7.1 is only non-linear with respect to* v_1. *We represented the non-linear variation with respect to* v_1 *using four linear segments:* $\{(0 - 0.75), (0.75 - 1.5), (1.5 - 2.25), (2.25 - 3)\}$. *We then divided the initial set zonotope into four parts corresponding to the linear segments as shown in Figure 7.1, and then applied the corresponding linear transformation on these zonotopes. The resultant image is shown with solid lines with star shaped corners. For the exponential box spline based implementation we assumed* $\alpha = \{(0, 0), (0, 1.5), (0, 1), (0.25, 0.75), (1, 1), (1.5, 1), (2, 1.25), (1, 0), (1.5, 0)\}$. *We computed image of the corners of the zonotope using the exponential box spline based representation and took the convex hull. The resultant image is shown using dashed lines with square shaped corners.*

In terms of computation complexity both the linear interpolation as well as EB spline technique required 9 computations of the non-linear function. From this simple analysis we see that EB spline based technique does not increase the computation complexity. In terms of accuracy we compare the amount of over approximation by both the techniques. The linear interpolation technique covers 25% more area while the EB spline based technique covers 13% more area.

There have been several efforts in computing reach set for nonlinear hybrid systems. These techniques can be classified based on their estimated error bounds on reach set as techniques where: a) approximation error is nonlinear in terms of discretization of the initial set [57], b)approximation error is exponential with respect to simulation time [56], c)approximation error depends on initial set size [60], and d) approximation error linearly depends on the size of the approximated domain i.e. size of each piece [58]. Our proposed

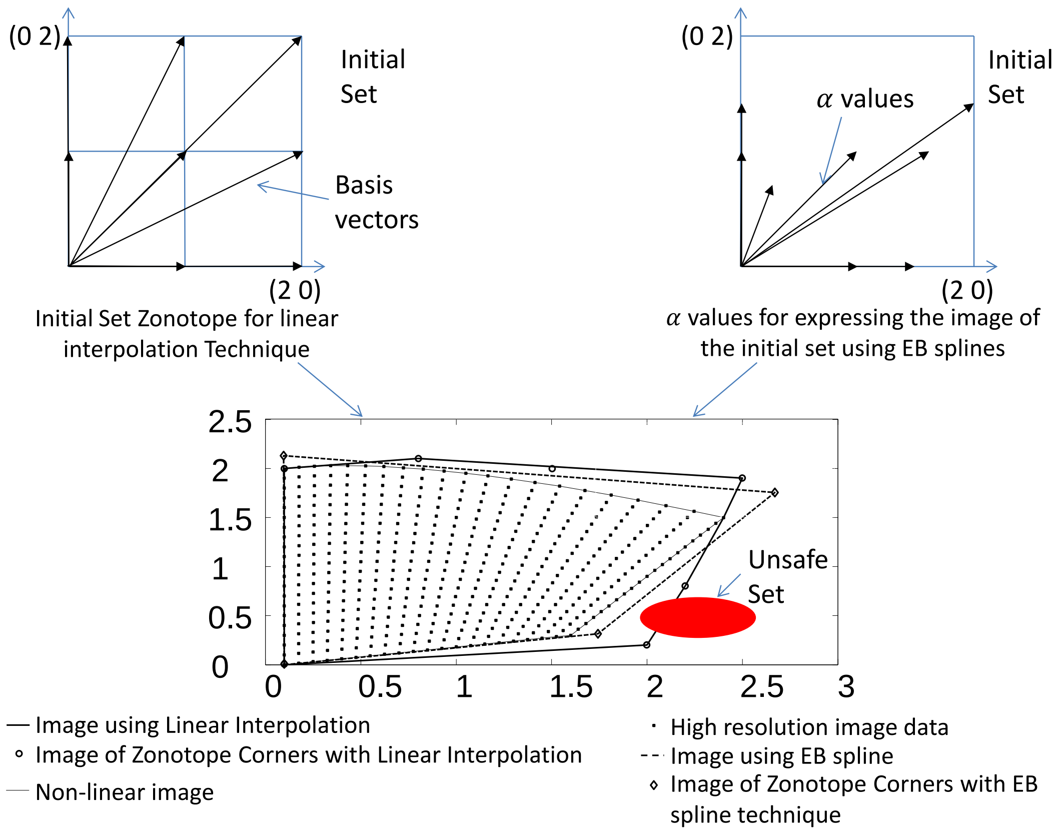


Figure 7.1: Comparison of Linear Interpolation and EB Spline based Image Computation Techniques.

methodology has error bound of $O(\tau + h)$, where τ is time discretization step and h is set discretization step. The error bound is linear and does not depend on size on the initial set. Also, the error does not increase with increase in simulation time.

Practical Purpose: Reachability analysis of hybrid systems are used for verifying safety properties of a CP control system. A subset of the set of real numbers can be denotes as an unsafe set as shown in Figure 7.1. If the image set intersects with the unsafe set then the system is considered to be unsafe for the given initial conditions. The initial conditions are reflective of the design of the CP control system. If the reach set is over approximated to a large extent then control system designs that are apparently safe will be considered unsafe. Such false negatives drastically reduce the design space and hence can potentially increase the cost of the design. Hence, a technique with a tighter approximation on the reach set is essential especially for designing cost effective mission critical systems. In our approach, we make use of the shape preserving property of exponential splines to tightly approximate non-linear dynamics such as diffusion processes. We show that EB spline technique can have a better approximation than piecewise linearization approach using the classical Brusselator example. We also show the usage of our EB spline based reachability analysis technique in the artificial pancreas example.

7.3 Artificial Pancreas Cyber Physical Control System

To validate our approach of reachability analysis using exponential box splines, we have applied the technique to a realistic system artificial pancreas (AP). It is a medical device, which is used for maintaining blood glucose level by inducing controlled amount of insulin in the blood. AP uses glucose sensor to sense the glucose level in the blood and accordingly computes the insulin concentration level. The nonlinear dynamics of the AP are represented using nonlinear equations 7.2, 7.3 and 7.4.

$$\frac{dX(t)}{dt} = -k_2.X(t) + k_3.(I(t) - I_b), \tag{7.2}$$
$$\frac{dG(t)}{dt} = -X(t).G(t) + k_1.(G_b - G(t)), \tag{7.3}$$
$$\frac{dI(t)}{dt} = -k_4.I(t) + k_5.(G(t) - k_6)^+.t, \tag{7.4}$$

Here, $\frac{dX(t)}{dt}$ gives the rate of the variation in the interstitial insulin concentration, $\frac{dG(t)}{dt}$ is the rate of change of blood glucose concentration for the infused insulin concentration X and $\frac{dI(t)}{dt}$ is the variation in plasma insulin concentration. Inaccurate infusion of insulin can harm the patient severely, e.g. if the glucose concentration G goes above $180mg/dl$, it can lead to hyperglycemia while low glucose level i.e. below $60mg/dl$ can cause hypoglycemia.

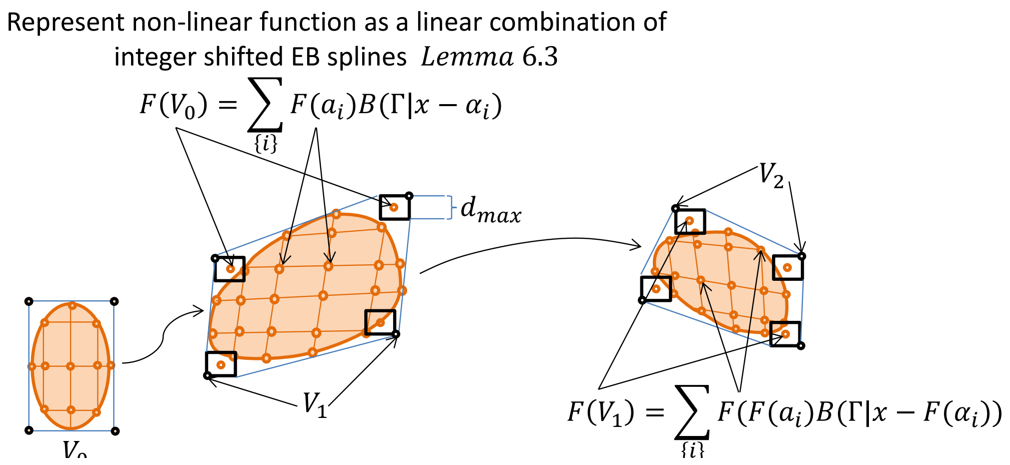

Represent non-linear function as a linear combination of integer shifted EB splines *Lemma* 6.3
$F(V_0) = \sum_{\{i\}} F(a_i)B(\Gamma|x - \alpha_i)$
d_{max}
V_2
V_1
V_0
$F(V_1) = \sum_{\{i\}} F(F(a_i)B(\Gamma|x - F(\alpha_i))$

Figure 7.2: Approach for Reach Set Computation of Nonlinear Hybrid System.

7.4 Related Works

Table 7.1 classifies the research on computing reachable set for nonlinear system.

7.5 Our Approach

Any approach to reachability analysis of non-linear systems requires an approximation of the solution to the differential equation that expresses the continuous dynamics of the hybrid system. Our aim is to perform reachability analysis of mission critical systems such as artificial pancreas and ventilators. These systems have dynamics that require shape preserving interpolants for accurate representation [93]. Further, reachability analysis requires estimation of the image of a set and not merely a single point. In a set there can be infinite number of points and hence the problem of finding the exact reach set is intractable. The typical approach towards estimation of the image of a set is to approximate the set using a polytope defined by a set of zonotopes. The next step is to compute the image of the corners of the polytope. For linear dynamics it is sufficient to take the convex hull of the images of the corners of the polytope to over-approximate the image. For non-linear differential dynamics we have to find an approximation that allows us to compute the image in a

Table 7.1: Summary of Related Work on Reachability Analysis for Nonlinear Systems.

Technique	**Type of nonlinearity**	**Error bound**	**Example used for modeling**
Zero order hold approximation [55]	Linear dynamics	$O(\tau^2)$, τ is time discretization step	Drug Infusion Pump
Partitioning of state space [56]	Second order differential equation	$O(h^2)$, h is partition size	Van der Pol oscillator, Biquad lowpass filter
Piecewise linear [57]	Lipschitz continuous	$O(h^2)$, h is mesh size	Van der Pol oscillator
Linear interpolation [58]	Lipschitz continuous	$O(h^2)$, h is mesh size	Biochemical network
Interval numerical method [60]	All types of nonlinear dynamics	Error varies with different nonlinear functions	Thermostat with delay, two-tank system, Air traffic conflict resolution
Use of Polynomial zonotopes [61]	Differential equations which can be represented as polynomials	$O(h^2)$	Van der Pol oscillator,, Biological aging model
Verifying Hybrid systems with parameters using first order logic [62]	Linear dynamics, No reachability for non-linear	Not applicable	Train Control System
Encoding hybrid systems [63]	Polynomial dynamics with discontinuous invariant	Not applicable	Braking control system of trains
Proposed Approach with EB Splines	Compactly supported continuous and smooth functions such as diffusion dynamics	$O(\tau+h)$, τ is time discretization step and h is set discretization step	Brusselator, Artificial Pancreas

similar way by computing image of specific points in the set and representing other points as a combination of these specific images. Hence, we need an interpolant that satisfies the following properties:

- **Property 1:** The interpolant should have shape preserving property [94].
- **Property 2:** Any image of a point can be represented as a weighted combination of shifted interpolants (Lemma 2).
- **Property 3:** There exists real number $h > 0$ such that an image of a set can be expressed as a summation of interpolants whose domains are shifted by integer multiples of h. (Lemma 2 and Theorem 7.9.1).

Property 1 ensures that the error in approximating the image of a fixed point is low and property 2 ensures that we can take an approach similar to linear systems for computing the image of a set for non-linear functions. Based on these two properties we now consider our approach to derive the image of a set as shown in Figure 7.2.

Given an initial set V_0 the first step is to over-approximate the set using a polytope (square as shown in Figure 7.2). We then select a set of points $\overrightarrow{\alpha_i}$ such that all the corners of the polytope $\{\overrightarrow{v_0}, \overrightarrow{v_1}, \dots \overrightarrow{v_n}\}$ can be expressed as linear combinations of these $\overrightarrow{\alpha_i}$ values. Utilizing the property 2 of the interpolant we express the image of the corner points $F(V_0)$ as a weighted sum of the images of the $\overrightarrow{\alpha}$ values. If $\overrightarrow{\alpha_i}$ s are further integer shifts then this amounts to representing the image using linearly independent interpolants (property 3 similar to piece-wise linearization).

Since we are only computing the image of the corners there can be cases especially for non-convex dynamics that a point inside the polytope has an image that lies outside the convex hull created by the image of the corners. In our approach, we tackle such conditions for a restricted class of non-linear dynamics whose first derivatives are bounded. In such cases we give a simple solution in Theorem 7.9.1. The solution is to enclose the images of the corners of the polytope by a polygon with sides d_{max} given by the bound on the first derivative and the maximum distance between the $\overrightarrow{\alpha}$ values i.e., the set discretization factor h (Lemma 4). The convex hull of the resulting polytope encloses the image of the initial set V_0.

There may be several interpolants that have the above-mentioned properties. For example Bezier curves of higher dimensions meet the above three properties [98]. In our example, the dynamics are diffusion dynamics expressed using error functions [91]. Exponential splines have been shown to work well with such dynamics [93] and hence we show our reachability analysis approach using exponential box splines and use it for the artificial pancreas case study. Finding other splines that fit these properties or which can be used with our technique is a question for future research.

7.6 Preliminaries

In this section, we discuss some of the preliminary concepts needed to understand our approach. Let us consider a co-ordinate space of dimension n, denoted by $\mathcal{R}^n$. We define a variate as $\mathcal{V}$, which is an ordered collection of n real variables v_i or an $n - tuple$. Thus, a valuation of the variate $\mathcal{V}$, is a member of the set $\mathcal{R}^n$, denoted using the vector $\overrightarrow{x}$ symbol over any letter. A subset of $\mathcal{R}^n$ is denoted by capital letters $\mathbb{X}$. The variation of each variable in the variate over time is defined by a non-linear differential equation of the form $\frac{dv_i}{dt} = g(\mathcal{V}, t)$ where the multivariate function $g(\mathcal{V}, t) = g(v_1, v_2, \ldots, v_n, t)$ is continuous and smooth within a domain $\mathcal{P} \subset \mathcal{R}^n$. Continuous and smooth functions are defines as Definition 3.

Definition 3 *A multi-variate function $g(v_1, v_2, \ldots, v_n, t)$ is called continuous and smooth if it is infinitely differentiable within the domain $\mathcal{P}$.*

Note that if $g(\mathcal{V}, t)$ is continuous and smooth within a domain $\mathcal{P}$ then the solution to the differential equation expressing each variable in $\mathcal{V}$ is also smooth and continuous within that same domain. We further restrict the class of functions to compactly supported smooth and continuous multivariate functions defined as follows:

Definition 4 *A multi-variate function $g(\mathcal{V}, t)$ is called compactly supported if and only if it is smooth and continuous and it is identically zero beyond a certain compact set in $\mathcal{R}^n$ i.e., there exists a subset $\mathbb{X} \subset \mathcal{R}^n$, also called the support, such that -*

$$\begin{aligned} g(\overrightarrow{y}, t) &\neq 0, \forall \overrightarrow{y} : \overrightarrow{y} = \sum_{j=1\ldots|X|} t_j \overrightarrow{x_j}, 0 \leq t_j < 1 \\ &= 0, \text{ otherwise }. \end{aligned} \tag{7.5}$$

Here $|X|$ denotes the number of elements in the set $\mathbb{X}$. Also the vector $\overrightarrow{y}$ as defined in Equation 7.5 is said to lie in the linear span $\langle X \rangle$ of the vectors in the set $\mathbb{X}$.

Compactly supported smooth and continuous functions are rapidly decreasing and infinitely differentiable. The set of such functions with domain $\mathcal{R}^n$ is denoted as $\mathcal{D}(\mathcal{R}^n)$. We define the distribution on a function in $\mathcal{D}(\mathcal{R}^n)$ as a linear mapping from $\mathcal{D}(\mathcal{R}^n)$ to real numbers. For example, a simple distribution can take the function $g(\mathcal{V}, t)$ as input and return $g(\{0, \ldots, 0\}, 0)$.

For a vector with n real numbers, we divide the set $\mathcal{R}^n$ into mutually exclusive cells $\mathbb{H} \subset \mathcal{R}^n : \bigcup_{\forall \mathbb{H}} \mathbb{H} = \mathcal{R}^n$. We define $\mathbb{H}^{\square}$ as the boundary of the cell $\mathbb{H}$ and $\mathbb{H}^{\boxdot}$ as the interior of the cell $\mathbb{H}$ such that $\mathbb{H}^{\square} \bigcup \mathbb{H}^{\boxdot} = \mathbb{H}$. Further, for two cells $\mathbb{H}_i$ and $\mathbb{H}_k$, $\mathbb{H}_i^{\boxdot} \bigcap \mathbb{H}_k^{\boxdot} = \emptyset$ and $\mathbb{H}_i$ and $\mathbb{H}_k$ are connected if $\mathbb{H}_i^{\square} \bigcap \mathbb{H}_k^{\square} = n - 1$. In the simplest case we can envision a cell as an n dimensional hypercube.

7.7 The Exponential Box Spline and its Properties

An exponential box spline is a distribution on the set $\mathcal{D}(\mathcal{R}^s)$. For every exponential box (EB) spline, we associate a *defining set* Γ. An element $\gamma \in \Gamma$ consists of two parts: a) an n-tuple $\overrightarrow{x_\gamma}$ and b) a real number λ_γ. The set $\mathbb{X}_\Gamma = \{\overrightarrow{x_\gamma} : \gamma \in \Gamma\}$ is the support of the EB spline while the set $\Lambda_\Gamma = \{\lambda_\gamma : \gamma \in \Gamma\}$ defines a scaling factor the EB spline distribution. For simplicity of symbols we also consider $\mathbb{X}$ as a matrix and Λ_Γ as a vector. An EB spline [95] over the defining set Γ is defined as follows:

Definition 5 *The n-dimensional exponential box spline $B(\Gamma)$ over a set $\Gamma \subset \mathcal{R}^n \times \mathcal{R}$ is a distribution on the set $\mathcal{D}(\mathcal{R}^n)$ such that -*

$$\int_{\mathcal{R}^n} B(\Gamma|\overrightarrow{x})\phi(\overrightarrow{x})d\overrightarrow{x} = \int_{(0,1)^{|\Gamma|}} \phi(\mathbb{X}_\Gamma \overrightarrow{t})e^{\Lambda_\Gamma \odot \overrightarrow{t}} d\overrightarrow{t}, \qquad (7.6)$$
$$\forall \phi \in \mathcal{D}(\mathcal{R}^n),$$

where $\overrightarrow{t} \in \mathcal{R}^{|\Gamma|}$ and $\odot$ is the dot product of two vectors.

Note that $|\Gamma|$ can be greater than n. In such cases $|\Gamma|$ will have at least n directional vectors that are orthogonal and the others can be used to span polygonal spaces. Let us consider that $|\Gamma| = n$ and if its linear span is equal to the coordinate space, $\langle\Gamma\rangle = \mathcal{R}^n$, then each element within $\mathbb{X}_\Gamma$ should point to unique orthogonal directions in the co-ordinate space. This means that *the dot product of any two elements,* $\overrightarrow{x_{\gamma_1}} \odot \overrightarrow{x_{\gamma_2}} = 0$. In such a case the EB spline can be represented using Lemma 1 [95].

Lemma 1 *If $|\Gamma| = n$ and $\langle\Gamma\rangle = \mathcal{R}^n$ then the EB spline $B(\Gamma|\overrightarrow{x})$ is given by -*

$$B(\Gamma|\overrightarrow{x}) \quad = \frac{e^{\overrightarrow{\theta} \odot \overrightarrow{x}}}{|det(\mathbb{X}_\Gamma)|}, \text{ if } \overrightarrow{x} = \textstyle\sum_{\gamma\in\Gamma} t_\gamma \overrightarrow{x_\gamma}, 0 \le t_\gamma < 1 \qquad (7.7)$$

$$= 0, \text{ otherwise .} \qquad (7.8)$$

Here $\overrightarrow{\theta} \odot \overrightarrow{x_\gamma} = \lambda_\gamma, \forall \gamma \in \Gamma.$

Proof: *The proof can be simply obtained by replacing $B(\Gamma|\overrightarrow{x})$ from Equation 7.7 to Equation 7.6 and doing a change of variables by replacing $\overrightarrow{x}$ with $\mathbb{X}_\Gamma \overrightarrow{t}$.*

Thus, we see that the EB spline is a compactly supported function whose support is the zonotope formed by the vectors in $\mathbb{X}_\Gamma$. In the following we give an example of EB spline and discuss its properties:

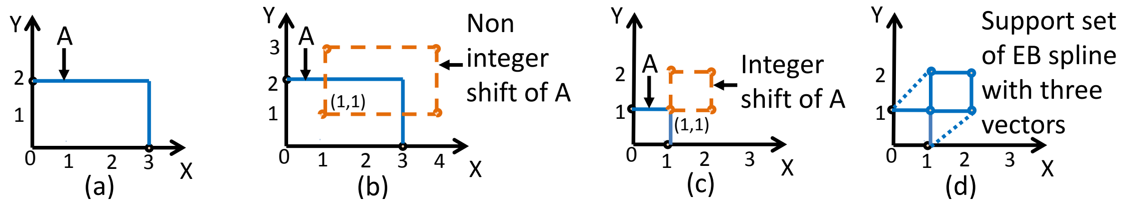


Figure 7.3: Support Zonotopes for Integer Translates of Exponential Box Splines

Example 7 *Figures (a) and (b) show support zonotopes for integer translates of exponential box splines with* $\Gamma = [\{(0,2), 0.5\}, \{(3,0), 1.5\}]$, *Figure (c) shows support zonotopes for integer translates of exponential box splines with* $\Gamma = [\{(0,1), 0.5\}, \{(1,0), 1.5\}]$ *and Figure (d) shows support zonotopes for exponential box splines with* $\Gamma = \{(0,1), (1,0), (1,1)\}$. *Let us consider that the defining set of the EB spline be* $\Gamma = [\{(0,2), 0.5\}, \{(3,0), 1.5\}]$. *Here,* $|\Gamma| = 2$, *and* $\langle\Gamma\rangle = \mathcal{R}^2$. *The support of the EB spline is shown in Figure 7.3(a). Note that* $det(\mathbb{X}_\Gamma) = -6$, $\overrightarrow{\theta} = \{0.5, 0.25\}$ *and the value of the function* $B(\Gamma\lceil\overrightarrow{x})$ *is non-zero within the zonotope formed by the elements in* $\mathbb{X}_\Gamma$ *and zero outside.*

An important characteristic of the EB spline is that integer shift operation on an EB spline leads to a linearly independent EB spline. Let us choose an $\overrightarrow{\alpha} \in \mathcal{R}^n$ such that there exists an $h > 0 : h^{-1}\overrightarrow{\alpha} \in \mathcal{Z}^n$, i.e, the set of n-tuple of integers. This method of generating α is equivalent to discretizing a set with an interval h. The EB spline $B(\Gamma\lceil\overrightarrow{x} - \overrightarrow{\alpha})$ is called an integer shift of the EB spline $B(\Gamma\lceil\overrightarrow{x})$. In Example 7 if we consider $\overrightarrow{\alpha} = \{1, 1\}$ then we get the EB spline shown in Figure 7.3(b) marked B. Note that $B(\Gamma\lceil\overrightarrow{x})$ marked A and $B(\Gamma\lceil\overrightarrow{x} - \overrightarrow{\alpha})$ have overlapping support zonotopes which means that they are not linearly independent. This is because 1, 1 is not an integer shift for both $\overrightarrow{x_\gamma} \in \mathbb{X}_\Gamma$. Now let us consider a $\Gamma = [\{(0,1), 0.5\}, \{(1,0), 1.5\}]$ such that $det(\mathbb{X}_\Gamma) = 1$. Now the $\alpha = \{1, 1\}$ is an integer shift for both the vectors in the defining set. As shown in Figure 7.3 (c), the support zonotopes of $B(\Gamma\lceil\overrightarrow{x})$ and $B(\Gamma\lceil\overrightarrow{x} - \overrightarrow{\alpha})$ do not overlap, which means that they are linearly independent.

Thus, exponential box splines are quite similar to sinusoids whose harmonics are linearly independent. Just as Fourier series representation decomposes a function into a series of sinusoids, EB splines also have a similar property that any continuous function can be approximated as a linear combination of integer shifts of EB splines [95].

We first consider the representation of an exponential function using EB splines.

Corollary 1 *For any defining set* Γ, *such that* $|\Gamma| > n$, *and for any* $\mathbb{J} \in \Gamma$ *which consists of only directional vectors, we can select integer shifts of EB spline* $\alpha \in Z_h^n$ *such that -*

$$e^{\overrightarrow{\theta_J} \odot \overrightarrow{x}} = \sum_{\forall \overrightarrow{\alpha}} C_h^{\Gamma}(\mathbb{J})^{-1} e^{\overrightarrow{\theta_J} \odot \overrightarrow{\alpha}} B(\Gamma \lceil \overrightarrow{x} - \overrightarrow{\alpha}), \tag{7.9}$$

where, $C_h^{\Gamma}(\mathbb{J}) = h^{-n} \Pi_{\gamma \in \Gamma} \int_0^h e^{(\lambda_\gamma - \overrightarrow{\theta_J} \odot \overrightarrow{x_\gamma}) \overrightarrow{t}} d\overrightarrow{t}$.

Proof: *We prove this through induction. Let us consider that* $\{\gamma' \in \Gamma' | \Gamma' = \Gamma \backslash \gamma\}$ *such that* $\mathbb{X}_{\lneqq}' = \mathbb{X}_{\lneqq} \backslash \overrightarrow{x_\gamma}$ *consists of only directional vectors and* $|\Gamma'| = n$. *From Equation 7.9,* $C_h^{\Gamma'} J = 1$, *since* $\lambda_{\gamma'} - \overrightarrow{\theta_J} \bigodot \overrightarrow{x_{\gamma'}} = \{0, 0, \ldots n \text{ zeros }\}$. *Further, if* $\mathbb{X}_{\lneqq}'$ *only has orthogonal and directional vectors then* $det(\mathbb{X}_{\lneqq}') = 1$. *Further, since* α*s are integer shifts then the shifted EB splines are all linearly independent and hence for each* $\overrightarrow{x}$ *there is only one EB spline that is non-zero. Using Equation 7.7 for* Γ', *we see that Equation 7.9 holds. Now let us consider* Γ *with one* $\overrightarrow{x_\gamma}$ *that is not orthogonal to at least one vectors in* $\mathbb{X}_{\Gamma'}$. *We can express the summation as -*

$$\begin{aligned}
& \sum_{\forall \overrightarrow{\alpha}} e^{\overrightarrow{\theta_J} \odot \overrightarrow{\alpha}} B(\Gamma \lceil \overrightarrow{x} - \overrightarrow{\alpha}) && (7.10) \\
= & \sum_{\forall \overrightarrow{\alpha}} \int_0^1 e^{\lambda_\gamma t} e^{\overrightarrow{\theta_J} \odot \overrightarrow{\alpha}} B(\Gamma' \lceil \overrightarrow{x} - \overrightarrow{\alpha} - t \overrightarrow{x}_\gamma) dt \text{ using Equation 7.6} \\
= & C_h^{\Gamma'}(\mathbb{J}) \int_0^1 e^{\lambda_\gamma t} e^{\overrightarrow{\theta_J} \odot (\overrightarrow{x} - t \overrightarrow{x_\gamma})} dt \text{ since Equation 7.9 holds for } \Gamma' \\
= & C_h^{\Gamma'}(\mathbb{J}) \int_0^1 e^{\lambda_\gamma t - \overrightarrow{\theta_J} \odot \overrightarrow{x_\gamma} t} dt e^{\overrightarrow{\theta_J} \odot \overrightarrow{x}} \\
= & C_h^{\Gamma}(\mathbb{J}) e^{\overrightarrow{\theta_J} \odot \overrightarrow{x}}
\end{aligned}$$

Using this corollary we prove the following lemma for continuous smooth functions -

Lemma 2 *Consider a defining set* $\Gamma : \langle\Gamma\rangle = \mathcal{R}^n$*, and* $\mathbb{X}_\Gamma \subset \mathcal{Z}^n$*. The defining set has s elements i.e.,* $|\Gamma| = s$*. Let us also consider the set* $\mathcal{Z}^n_h = \{\overrightarrow{\alpha} : \exists h > 0, h^{-1}\overrightarrow{\alpha} \in \mathcal{Z}^n\}$*. Then for any continuous compactly supported smooth multivariate function* $f(\overrightarrow{x})$ *with bounded first derivatives and a set* $\mathbb{A} \subset \mathcal{R}^n$*,*

$$|f(\overrightarrow{x}) - C^\Gamma_h(\mathbb{J})^{-1} \sum_{\overrightarrow{\alpha}\in\mathcal{Z}^n_h} f(\overrightarrow{\alpha}) B_h(\Gamma|\overrightarrow{x} - \overrightarrow{\alpha})| = O(h), \tag{7.11}$$

$\forall\overrightarrow{x} \in \mathbb{A}$ *where* $\mathbb{J} \subset \Gamma : \langle J\rangle = \mathcal{R}^n \& |J| = n$*.*

Proof: *Before we go into the proof we need to define* $\mathbb{J} \subset \Gamma$ *as a set of n basis vectors or directions in the coordinate space that can span the entire co-ordinate space. However,* Γ *can have other vectors as well which gives different shapes to the support zonotope and hence helps to map the domain of the function* $f(\overrightarrow{x})$*. Figure 7.3 (d) shows one such support if* $\mathbb{X}_\Gamma = \{(0,1),(1,0),(1,1)\}$*. Then* $\mathbb{X}_J = \{(0,1),(1,0)\}$*. Here* $\overrightarrow{\theta_J} \bigodot \overrightarrow{x_\gamma} = \lambda_\gamma$*.*

Let $\overrightarrow{x} \in \mathbb{A}$*, since* $\overrightarrow{\alpha_i}$ *s have to span* $\overrightarrow{x}$*, we get -*

$$\begin{aligned} \overrightarrow{x} &= \sum_i t_i \overrightarrow{\alpha_i} \\ \|\overrightarrow{x} - \overrightarrow{\alpha}\| &\leq h \sum_i t_i h^{-1}\overrightarrow{\alpha_i} \\ &\leq h \sum_i t_i \overrightarrow{z_i}, \overrightarrow{z_i} \in \mathcal{Z}^n \leq O(h). \end{aligned} \tag{7.12}$$

Utilizing this result we get -

$$\begin{aligned}
& \| \quad \frac{f(\overrightarrow{x})}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}}} - \frac{f(\overrightarrow{\alpha})}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{\alpha}}}\| \qquad (7.13)\\
= \quad & \|\frac{1}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}}}[f(\overrightarrow{x}) - e^{\overrightarrow{\theta_J}\bigodot(\overrightarrow{x}-\overrightarrow{\alpha})}f(\overrightarrow{\alpha})]\|\\
\leq \quad & \|\frac{1}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}}}[f(\overrightarrow{x}) - e^{O(h)}f(\overrightarrow{\alpha})]\|\\
& \textit{from Equation 7.12}\\
\leq \quad & \|\frac{1}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}}}[f(\overrightarrow{x}) - (1+O(h))f(\overrightarrow{\alpha})]\|\\
& \textit{for a small enough } h\\
\leq \quad & \|\frac{1}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}}}[f(\overrightarrow{x}) - f(\overrightarrow{\alpha}) - O(h)f(\overrightarrow{\alpha})]\|\\
\leq \quad & \|\frac{1}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}}}[d_{max}(\overrightarrow{x} - \overrightarrow{\alpha}) - O(h)f(\overrightarrow{\alpha})]\|\\
& \textit{where } d_{max} \textit{ is the upper bound on the derivative}\\
\leq \quad & \frac{1}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}}}O(h) \textit{ from Equation 7.12.}
\end{aligned}$$

From Corollary 1 we have -

$$e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}} = \sum_{\forall\overrightarrow{\alpha}} e^{\overrightarrow{\theta}\bigodot\overrightarrow{\alpha}} C_h^{\Gamma}(\mathbb{J})^{-1}B(\Gamma\lceil\overrightarrow{x} - \overrightarrow{\alpha}). \qquad (7.14)$$

Now let us consider the function $f(\overrightarrow{x})$ as follows -

$$\begin{aligned}
& \| \quad f(\overrightarrow{x}) - C_h^{\Gamma}(J)^{-1}\sum_{\forall\overrightarrow{\alpha}} f(\overrightarrow{\alpha})B(\Gamma\lceil\overrightarrow{x} - \overrightarrow{\alpha})\| \qquad (7.15)\\
= \quad & \|\sum_{\forall\overrightarrow{\alpha}} \frac{f(\overrightarrow{x})}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}}} e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{\alpha}} C_h^{\Gamma}(J)^{-1}B(\Gamma\lceil\overrightarrow{x} - \overrightarrow{\alpha})\\
- \quad & \sum_{\forall\overrightarrow{\alpha}} \frac{f(\overrightarrow{\alpha})}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{\alpha}}} e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{\alpha}} C_h^{\Gamma}(J)^{-1}B(\Gamma\lceil\overrightarrow{x} - \overrightarrow{\alpha})\|\\
\leq \quad & max\|\frac{f(\overrightarrow{x})}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}}} - \frac{f(\overrightarrow{\alpha})}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{\alpha}}}\| \sum_{\forall\overrightarrow{\alpha}} e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{\alpha}} \frac{B(\Gamma\lceil\overrightarrow{x} - \overrightarrow{\alpha})}{C_h^{\Gamma}(J)}\\
\leq \quad & \frac{1}{e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}}}O(h)e^{\overrightarrow{\theta_J}\bigodot\overrightarrow{x}} \textit{ from Equation 7.13 and 7.14,}\\
\leq \quad & O(h).
\end{aligned}$$

The lemma allows us to approximate the function $f(\overrightarrow{x})$ over a set $\mathbb{A}$. Ideally to obtain $f(\mathbb{A})$ we have to compute countably infinite number of points. The Lemma 2 allows us to divide the set $\mathbb{A}$ into a grid with grid size h and then compute $f(\overrightarrow{x})$ at each grid point and approximate the value of $f(\overrightarrow{x})$ in between each grid point using an exponential box spline. The most important result is that if $f(\overrightarrow{x})$ has bounded first derivatives then the error of this approximation linearly increases with h the discretization factor. We use this result to obtain a tight approximation of the reach set of a non-linear hybrid automata. We next discuss the definition of non-linear hybrid automata and its reachability analysis.

7.8 Smooth and Compactly Supported Non-Linear Hybrid Automata

Non-linearities in hybrid automata can arise in several manners. However, we restrict our hybrid automata such that the continuous dynamics is expressed using compactly supported continuous and smooth functions. Thus, we define a Smooth and Compactly Supported Non-Linear Hybrid Automata (SCHA), as follows -

Definition 6 *A Smooth and Compactly Supported Non-Linear Hybrid Automata (SCHA) is a tuple* $\{\mathcal{V}, n, \mathcal{L}, m, Inv, \mathcal{F}, Re\}$ *such that -*

- $\mathcal{V}$ *is a variate consisting of n number of real valued variables* $v_i \in \mathcal{R}$.
- $\mathcal{L}$ *is a set of m modes* $\{l_1 \ldots l_m\}$.
- $Inv : \mathcal{L} \to 2^{\mathcal{R}}$*, is a mapping from a mode to a set of subsets of the real set* $\mathcal{R}$ *such that -*
  - $Inv(l_i) \bigcap Inv(l_j) = \emptyset$, $\forall l_i, l_j \in \mathcal{L}$, *and* $l_i \neq l_j$.
  - $\bigcup_{\forall l_i \in \mathcal{L}} Inv(l_i) = \mathcal{R}^n$
  - $\forall l_i \in \mathcal{L}$ *the cells in the invariant set* $Inv(l_i)$ *are all connected (Section 7.6).*

- $\mathcal{F} : \mathcal{L} \times \mathcal{V} \rightarrow \mathbb{D}(\mathcal{R}^n)$ *is a mapping that associates a smooth and continuous compactly supported multi-variate function* $g_{l_i} : \mathcal{V} \times \mathcal{R} \rightarrow \mathcal{R}$ *to each variable* $v_j \in \mathcal{V}$ *in mode* l_i *such that -*

$$\frac{dv_j}{dt} = g^j_{l_i}(v_1 \dots v_n, t). \tag{7.16}$$

- $Re : \mathcal{L} \times \mathcal{V} \rightarrow \mathcal{R}^n$ *is a reset function that sets initial conditions of the real variables in* $\mathcal{V}$ *at each mode* $l_i \in \mathcal{L}$.

The definition 6 only defines the components of an SCHA. But for an SCHA to be executed we need to define three entities: a) trajectory at a given location, b) execution over a given time, and c) discrete state transition.

Definition 7 Trajectory*: The trajectory of an SCHA for a location* l_i*, from an initial vector* $\overrightarrow{v_0} \in Inv(l_i)$ *for a duration* $t_f > 0 \in \mathcal{R}$ *is the set of maps* $\eta : \mathcal{V} \times [0, t_f] \rightarrow \mathcal{R}^n$ *such that -*

1. $\eta_j(\overrightarrow{v_0}, \tau)$ *is a solution to the Equation 7.16* $\forall \tau \in [0, t]$ *and* $\forall j \in \{1 \dots n\}$*, i.e,* $\frac{dv_j}{dt} = \frac{d\eta_j(\overrightarrow{v_0}, \tau)}{d\tau} = g^j_{l_i}(v^0_1 \dots v^0_n, \tau)$ $\forall \eta_j \in \eta$*. And*

2. $\eta(\mathcal{V}_0, \tau) \in Inv(l_i)$ $\forall \tau \in [0, t]$.

We denote the duration of a trajectory as $\eta.dur$.

Definition 8 Execution*: The execution* β *of an SCHA from an initial state* $(l_0, \overrightarrow{v_0}) \in \mathcal{L} \times \mathcal{R}^n$ *for a duration t is a concatenation of finite or infinite number of trajectories* $\eta^0\eta^1\eta^2 \dots$ *such that -*

- $\eta^0(\overrightarrow{v_0}, 0) = \overrightarrow{v_0}$,

- $\eta^{k+1}(., 0) = Re(\eta^k(., \eta^k.dur))$ *for* $k \geq 1$*, and*

- $\beta.dur = \sum_{\forall k} \eta^k.dur$, *where $\beta.dur$ is the duration of the execution. $Re(.)$ is a reset function, which allows reseting the values of the system variables after an execution (Definition 6).*

Definition 9 Discrete mode transition: *For two modes $(l_i, l_j) \in \mathcal{L}$, and the initial vector $\overrightarrow{v_{l_i}}$ for the mode l_i a discrete transition takes place from l_i to l_j at a state $(l_i, \eta^{l_i}(\overrightarrow{v_{l_i}}, \tau)) \in \mathcal{L} \times \mathcal{R}^n$ and at time τ if $\eta^{l_i}(\overrightarrow{v_{l_i}}, \tau) \in Inv(l_i) \bigcap Inv(l_j)$ and $\eta^{l_i}(\overrightarrow{v_{l_i}}, \tau) = Lim_{\tau' \to \tau} \eta^{l_i}(\overrightarrow{v_{l_i}}, \tau')$ where $\eta^{l_i}(\overrightarrow{v_{l_i}}, \tau') \in Inv(l_i)^o$ for some $\delta > 0$ such that $\tau' \in [\tau - \delta, \tau]$.*

7.9 Reachability Analysis

Reachability analysis of a hybrid automata concerns with the computation of an execution of the SCHA from an initial set of continuous states $\mathcal{V}_0$ at time t=0 to a final time t = t_f. In an execution the SCHA undergoes discrete mode transitions. Thus, the reachability analysis requires to compute two parameters: a) the trajectory, and b) a discrete mode transition. We discus these two aspects in the following sections.

7.9.1 Estimation of Trajectory

The estimation of the trajectory requires solution of the differential equation 7.16. All forms of non-linear differential equation may not have closed form solutions. However, we do not need closed form solutions for the equations. We propose to discretize time into intervals of δt and consider that the differential is independent of time within this δt interval. With such an assumption the non-linear image of an initial vector $\mathcal{V}_0$ is given by Lemma 3.

Lemma 3 *Consider an initial mode of the SCHA as l_0 and an initial vector $\overrightarrow{v_0}$. Let $\delta t > 0$ be a small real number which is the interval by which time increases i,e. $t_{k+1} = t_k + \delta t \; \forall k \in \mathcal{Z}$. Let the vectors $\overrightarrow{v_k}$ and $\overrightarrow{v_{k+1}}$ be the valuation of the system parameters at a time t_k and t_{k+1} respectively. For any component v_j of $\overrightarrow{v_k}$, we estimate $v_j(t_{k+1})$, value of the component v_j at*

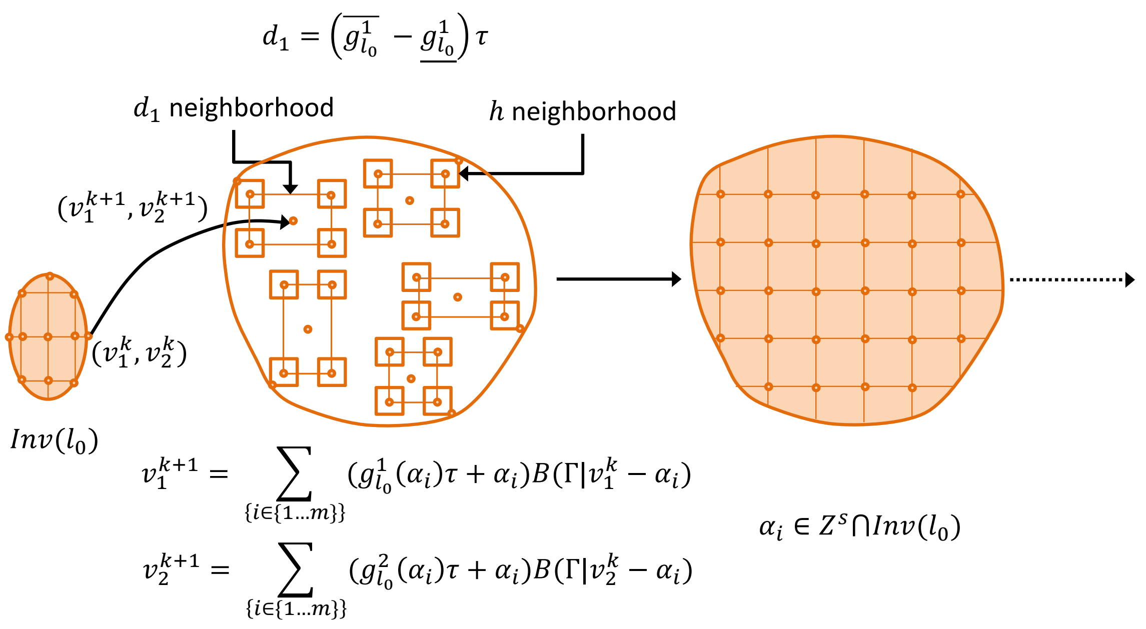

$d_1 = \left(\overline{g^1_{l_0}} - \underline{g^1_{l_0}}\right)\tau$
d_1 neighborhood
h neighborhood
(v_1^{k+1}, v_2^{k+1})
(v_1^k, v_2^k)
$Inv(l_0)$
$v_1^{k+1} = \sum_{\{i \in \{1 \ldots m\}\}} (g^1_{l_0}(\alpha_i)\tau + \alpha_i) B(\Gamma | v_1^k - \alpha_i)$
$v_2^{k+1} = \sum_{\{i \in \{1 \ldots m\}\}} (g^2_{l_0}(\alpha_i)\tau + \alpha_i) B(\Gamma | v_2^k - \alpha_i)$
$\alpha_i \in Z^s \cap Inv(l_0)$

Figure 7.4: Approach for Estimation of the Trajectory of SCHA.

time t_{k+1} as -

$$v_j(t_{k+1}) = g^j_{l_0}(v_1 \ldots v_n, t_k)\delta t + v_j(t_k). \tag{7.17}$$

An initial set $\mathbb{A} \in Inv(l_0) \subset \mathcal{R}^n$ consists of vectors $\mathcal{V}$. The image of a set $\mathbb{A}$ is defined as follows -

Definition 10 *The image of a set of vectors $\mathbb{A}$ for a mode l_i of the SCHA at time t_k is the set of vectors $Img(\mathbb{A})$ such that for any variate $\mathcal{V}$ whose valuation lies in $\mathbb{A}$ there exits a vector $\overrightarrow{F(l_i, \mathcal{V})} \in Img(\mathbb{A})$. The jth component v^f_j of the vector $\overrightarrow{F(l_i, \mathcal{V})}$ is given by -*

$$v^f_j = g^j_{l_i}(v_1 \ldots v_n, t_k)\delta t + v_j, \tag{7.18}$$

where v_j is the jth component of $\mathcal{V}$.

According to the definition of SCHA the function $g^j_{l_i}$ has finite upper and lower bounds since F is required to have bounded first derivatives. Thus, we can theoretically limit the image $\overrightarrow{F(l_i, \mathcal{V})}$ of a variate $\mathcal{V}$ using the following lemma.

Lemma 4 *Given a $\mathcal{V}$ (v_j is the jth component of $\mathcal{V}$) the image $\overrightarrow{F(l_i, \mathcal{V})}$ for a given mode l_i lies inside a polytope with its center at $\mathcal{V}^c$ and side length along a dimension i is given by d_i such that:*

- *the ith component v_i^c of $\mathcal{V}^C$ is given by Equation 7.18*
- *d_i is given by*

$$d_i = (\bar{g}_{l_i}^j - \underline{g_{l_i}^j})\tau, \tag{7.19}$$

where, $\bar{g}_{l_i}^j$ and $\underline{g_{l_i}^j}$ are the upper and lower bounds of the function $g_{l_i}^j$ respectively.

According to the definition of image of a set, we should consider every point $\overrightarrow{x}$ in the set $\mathbb{A}$ and compute Equation 7.18 to obtain the image $\overrightarrow{x}^f$. Then over-approximate the image by surrounding with a polytope whose sides are given in Equation 7.19. However, this process is not tractable since there are countably infinite points inside the set $\mathbb{A}$. Lemma 2 allows us to discretize $\mathbb{A}$ into a uniform mesh and use the image of only the mesh points to estimate the image of the set A. The following theorem helps us to device an algorithm:

Theorem 7.9.1 *Given a continuous compactly supported smooth function $f : \mathcal{R}^n \rightarrow \mathcal{R}^n$ with bounded first derivatives and a set $\mathbb{A} \subset \mathcal{R}^n$, there exists a set of points $\psi = \{\overrightarrow{\alpha} : h^{-1}\overrightarrow{\alpha} \in \mathcal{Z}^n, h > 0\}$ such that the image of the set $\mathbb{A}$ over the function f, $f(\mathbb{A})$ lies within the h neighborhood of the convex hull of the set of points $\psi^f = \{f(\overrightarrow{\alpha}) : \overrightarrow{\alpha} \in \psi\}$.*

Proof: *We know from Lemma 2 that any continuous compactly supported smooth function with bounded first derivatives can be represented using a linear combination of exponential box splines as shown in Equation 7.11, where $|\Gamma| = n$, $\langle\Gamma\rangle = \mathcal{R}^n$, and $\mathbb{X}_\Gamma \subset \mathcal{Z}^n$. Let us consider that* $\mathbb{X}_\Gamma = \begin{pmatrix} n & 0 & \cdots & 0 \\ 0 & n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & n \end{pmatrix}$. *Let us consider a point $\overrightarrow{x} \in A$. Since $|det(\mathbb{X}_\Gamma)| > 1$,* *the integer translates of the exponential box spline $B(\Gamma\lceil\overrightarrow{x})$ are not independent. This means that at the point $\overrightarrow{x}$ the support of more than one integer translates overlap. Let us consider that there are q such integer translates of EB spline $B(\Gamma\lceil\overrightarrow{x}-\overrightarrow{\alpha_i}) : i \in 1\ldots q$, whose support zonotopes overlap at the point $\overrightarrow{x}$. Note that $\overrightarrow{\alpha_i}$'s can be easily obtained from the knowledge of $\overrightarrow{x}$ and iteratively examining overlap of the support zonotopes of integer translates of the EB spline. Further, $\mathbb{X}_\Gamma$ contains the set of all directions in the co-ordinate space and hence $\mathbb{J} = \mathbb{X}_\Gamma$ the value of $C_h^\Gamma(J) = 1, \forall \mathbb{J}$. Thus, using Lemma 2 and 7.7 we can write for any point $\overrightarrow{x} \in \mathbb{A}$,*

$$\begin{aligned} f(\overrightarrow{x}) &\leq \sum_{i=1}^{q} f(\overrightarrow{\alpha_i}) B(\Gamma\lceil\overrightarrow{x}-\overrightarrow{\alpha_i}) + O(h), \\ &\leq \sum_{i=1}^{q} f(\overrightarrow{\alpha_i}) \frac{e^{\overrightarrow{\theta_\Gamma}\odot(\overrightarrow{x}-\overrightarrow{\alpha_i})}}{|det(\mathbb{X}_\Gamma)|} + O(h), \end{aligned} \tag{7.20}$$

If we can find $\overrightarrow{\theta_\Gamma}$ such that -

$$\sum_{i=1}^{q} \frac{e^{\overrightarrow{\theta_\Gamma}\odot(\overrightarrow{x}-\overrightarrow{\alpha_i})}}{|det(\mathbb{X}_\Gamma)|} = 1, \tag{7.21}$$

then $f(\overrightarrow{x})$ lies in the $O(h)$ neighborhood of the convex hull of the points $f(\overrightarrow{\alpha_i})$. Further, since the dimension of the co-ordinate space is n, there will be at least n integer translates whose support zonotopes must overlap at any point $\overrightarrow{x} \in \mathbb{A}$. Thus, we can derive a system of n linear equations of the form -

$$\sum_{j=1}^{n} \theta_j(v_j - \alpha_i^j) = ln(\frac{|det(\mathbb{X}_\Gamma)|}{n}), \qquad \forall i \in \{1\ldots n\}, \tag{7.22}$$

such that its solution will satisfy Equation 7.21. Here θ_j, v_j, and α_i^j are components of $\overrightarrow{\theta}$, $\overrightarrow{x}$, and $\overrightarrow{\alpha_i}$. Since, any solution to the linear system of Equations 7.22 satisfies Equation 7.21, any $\overrightarrow{x} \in \mathbb{A}$ lies within the h neighborhood of the convex sum of the points $f(\overrightarrow{\alpha_i})$.

Lemma 4 and 2 and Theorem 7.9.1 can be used to estimate the image of a set of vectors $\mathbb{A}$. The basic idea for the estimation is to employ two types of dicretization: a) discretize time to approximate the solution of the differential equations in Equation 7.16, and b) discretization of the set $\mathbb{A}$ into integer points in $\mathbb{A}$ and estimate the $Img(\mathbb{A})$ using Lemma 2. We next discuss an algorithm to progressively compute the trajectory of a given mode l_i in SCHA.

Algorithm for Estimation of a Trajectory

The basic idea is depicted using the Figure 7.4. Given a set $\mathbb{A} \subset \mathcal{R}^n$, we first over-approximate the set $\mathbb{A}$ using a polytope specified as a linear combination of a set of zonotopes, $\mathbb{X}_\Gamma$. We define exponential box splines with $\Gamma = \{\mathbb{X}_\Gamma, \Lambda_\Gamma\}$ as the defining set for the EB splines. For each point $\overrightarrow{x_p}$ on the corners of the polytope that over approximates $\mathbb{A}$, we find unique set of $\overrightarrow{\alpha}$s such that $\overrightarrow{x_p}$ falls in the linear span of the chosen $\overrightarrow{\alpha}$s. We then use Equation 7.22 to determine $\overrightarrow{theta}$ for each corner point. We then compute the image of a corner vector $\overrightarrow{x_p}$, using the EB spline representation of Equation 7.11. The function f in the EB spline representation is the time discretization of Equation 7.17. We build a polytope around $Img(\overrightarrow{x_p})$ following Lemma 4. We then build hyper-rectangles of side length h with each vertex of the polytopes as their center. The reach set is the convex hull of the vertices of the hyper-rectangles. We continue this process until the final time t_f is reached or there is a state transition.

Algorithm 1 shows the pseudocode of this algorithm using exact steps in order to prove properties on the error of such estimation. The input to the algorithm is a SCHA, and invariant set Inv related to a mode l, an initial set $\mathbb{A}$, a final time t_f, a time discretization τ, and an $h > 0$.

In the following theorem we prove that the Algorithm 1 indeed obtains the reach set within a given approximation bound.

Algorithm 1 (R_{set}, t) = CalcTraj(SCHA,$Inv(l)$,$\mathbb{A}$,t_f,τ,h)

```
1: for each $i \in \{1 \ldots n\}$ do
2:     Find $int_j = max(v_j) - min(v_j)$ for each variable in the variate $\mathcal{V}$
3: end for
4: Over-approximate the set $\mathbb{A}$ as set $A^d$ using a set of zonotopes $\overrightarrow{z_k}$.
5: $R_{set} = \mathbb{A}$
6: t = 0
7: For each element in $A^d$, derive $\overrightarrow{\alpha}$s that span the element. There can be at most $2n\ \overrightarrow{\alpha}$.
8: For each element in $A^d$ compute the $\overrightarrow{\theta}$ values using Equation 7.22.
9: while $t \le t_f$ & $R_{set} \not\subset Inv(l)$ do
10:     $\forall \overrightarrow{v} \in A^d$ compute $Img(\overrightarrow{v}, \tau)$ using Equation 7.11 and 7.18,
11:     Build polytopes with each point in $Img(\mathcal{V}, \tau)$ as center and of lengths given in Lemma 4.
12:     Let $Vert(Img(\mathcal{V}), \tau)$ be the set of vertices of the polytopes
13:     For each vertex in $Vert(Img(\mathcal{V}), \tau)$ build a hyper-rectangle of side $h$ with the vertex at the center
14:     $Chull$ = convex hull of the vertices of the hyper-rectangles.
15:     $R_{set} = R_{set} \bigcup Chull$
16:     t = t + $\tau$
17: end while
18: return($R_{set}, t$)
```

Theorem 7.9.2 *Given an SCHA, and mode l, an initial set of continuous states $\mathbb{A}$, a final time t_f, a time discretization τ, and a number $h > 0$, the Algorithm 1 outputs the reach set with an over approximation error of $O(\tau + h)$.*

Proof: *The proof is a direct consequence of the Theorem 7.9.1 and the Lemma 4. The Algorithm 1 first approximates the image of a point using Lemma 4, thus guaranteeing that the actual image lies within the polytope of side length given in Equation 7.19. Then it considers the h-neighborhood of each vertex of the polytope, and takes the convex hull of the resultant vertices of the neighborhood. According to Theorem 7.9.1, the image of the set of point in $\mathbb{A}$ is contained by the h-neighborhood of the convex hull of the actual images* *of the point. Since the polytopes over approximate the actual image, the image of the set $\mathbb{A}$ should also lie within the h-neighborhood of the vertices of the over approximating polytopes. Thus, the Algorithm 1 returns a reach set with an approximation error $O(h + \tau)$.*

7.9.2 Estimating Discrete Transitions

The invariant set of the SCHA can be specified as a collection of cells. Thus, estimating when the reach set crosses an invariant set can be determined by intersection of a convex hull with a polygon. We assume that all the invariant sets are compact sets. In such a scenario the intersection of a polygon and convex hull can be computed in $O(plog(p))$ time, where p is the number of facets of the convex hull, using several algorithms in existing research [99]. The intersections will result in division of the convex hull into several subsets. We can use Algorithm 1 to obtain the reach set of multi-mode SCHA. The algorithm for tackling transitions is straightforward and similar to the linear zonotope based reachability analysis case.

In this regard, we consider Algorithm 2 to compute the reachability of multi-mode SCHA.

7.10 Case-study: Artificial Pancreas

We apply Definition 6 to Artificial Pancreas (AP) which is used to control blood glucose level for diabetics patients. We instantiated the definition 6 is as follows:
- $\mathcal{V} = \{X, G, I, t\}$ is a vector with $\{X, G, I, t\}$ as real valued variables. X represents the variation in the interstitial insulin concentration, G is the blood glucose concentration and I is the plasma insulin concentration.
- $\mathcal{L} = \{Basal, Breaking, Correction\}$ are the discrete modes of the AP hybrid automata. *Basal* is the initial state. It infuses insulin with a specified constant rate. It predicts the blood glucose concentration G one hour ahead. If the prediction shows that glucose con-

Algorithm 2 R_{set} = CalcReach(SCHA,l_0,$\mathbb{A}$,t_f,τ,h)

```
1: $CurrentState = l_0$
2: $t = 0$, $timeLeft = t_f$, $t_j = 0$;
3: $InitSet(l_0) = \mathbb{A}$
4: while $t < t_f$ do
5:     for each element $l$ in $CurrentState$ do
6:         $timeLeft = timeLeft - t_j$
7:         $CurrentState = CurrentState\ l$
8:         $(TrajSet, t_j) = CalcTraj(SCHA, Inv(l), InitSet(l), timeLeft, \tau, h)$
9:         for each mode $m$ in SCHA do
10:            if ( then$TrajSet \bigcap Inv(m) \neq \phi$)
11:                $CurrentState = CurrentState \bigcup m$
12:                $InitSet(m) = InitSet(m) \bigcup (TrajSet \bigcap Inv(m))$
13:            end if
14:        end for
15:        $R_{set} = R_{set} \bigcup TrajSet$
16:        $t = t + t_j$
17:    end for
18: end while
19: return($R_{set}$)
```

centration will become very low, the system goes in *Braking* mode otherwise remain in *basal* mode only. In *Braking* mode, the risk factor is calculated using nonlinear function of G given in equation 7.3. Based on the risk factor, the insulin dosage, I reduces. If G remains in the range of $60mg/dl$ to $120mg/dl$, system goes back to *Basal* mode, otherwise if G becomes more than $180mg/dl$, system goes in *Correction* mode. Similar to *Basal* model, *Correction* mode also predicts G one hour ahead. It calculates the bolus and adds it with Basal rate. In this mode, if G goes below $20mg/dl$, system goes in *Braking* mode and if G is in safe rage i.e. in between $60mg/dl$ to $120mg/dl$, it goes in *Basal* mode.

- Different invariant conditions are defined for each of the mode of AP hybrid automata.

The system continues remaining in the same mode until its invariant condition is satisfied. The *Inv* set is mapping from a mode to $\mathcal{R}$ subset as: $\{Inv(Basal) = ([60/k, 120/k])\}$, $\{Inv(Braking) = ([0, 60])\}$, and $\{Inv(Correction) = ([120/k, \infty])\}$.

- The variables $\{X, G, I, t\}$ in vector $\mathcal{V}$ vary over smooth and compactly supported multivariate functions expressed using Equations 7.2, 7.3, and 7.4. These functions contain patient specific constants, k_1, k_2, k_3, k_4, k_5, and k_6. I_b represents the basal plasma insulin concentration while G_b is the basal glucose concentration in the blood.

- Reset function is defined for each mode in $\mathcal{L}$ to set the initial conditions as: $\{Re(Basal) = ([I = I_0])\}$, $\{Re(Braking) = ([I = I_{reduced}])\}$, and $\{Re(Correction) = ([I = I_{increased}])\}$. For the *Basal* mode, the insulin rate I is set to the constant rate, I_0. *Braking* mode reduces the insulin rate considering the predicted risk factor. This reduced insulin rate $I_{reduced}$ is set as new I. In *Correction* mode, Bolus is added with the basal rate which increases the current I value. It acts as reset condition for *Correction* mode.

We apply our proposed reachability analysis technique to the artificial pancreas SCHA, using MATLAB. We consider an initial set of states where the glucose concentration varies from 20 mg/dl to 180 mg/dl, the infusion rate varies from 200 units/min to 400 units/min, and the interstitial glucose concentration varies from 0.1 g/dl to 1 g/dl. The evolution of the reach set is shown in Figure 7.5. The hyperglycemia threshold is shown by the grey plane. Any image on the right of the plane is unsafe. As we see that in several cases the image crosses to the right side of the unsafe planes leading to hyper-glycemia however, there were no hypoglycemic stages of the patient. In our previous research [55], we have performed reachability analysis of artificial pancreas with piecewise linearization. The reach set diverged as simulation time increased as opposed to the converging as seen in Figure 7.5.

We have proposed a methodology of reachability analysis for nonlinear hybrid systems using exponential box splines. The continuous dynamics of nonlinear hybrid automata

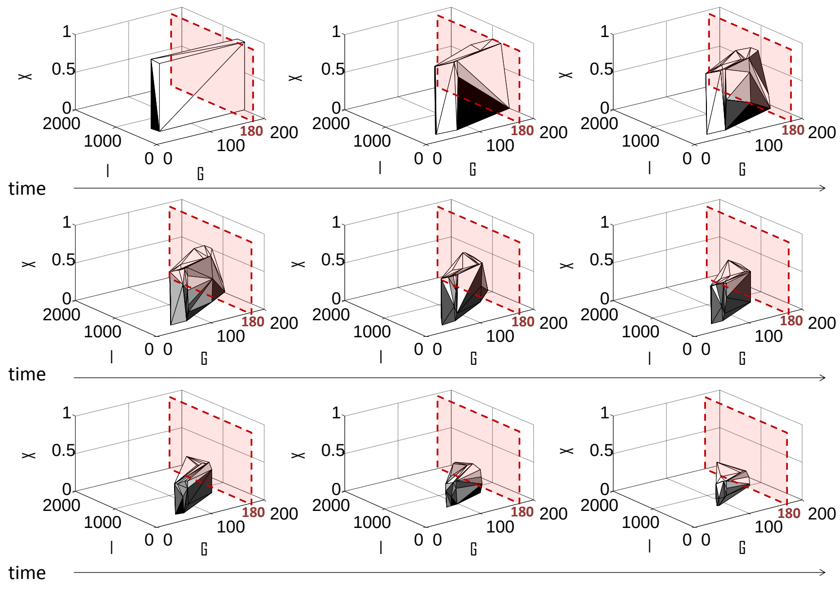


Figure 7.5: The Evolution of Images of Reach Set Over Time for the Artificial Pancreas SCHA.

considered in this paper is represented using compactly supported continuous and smooth functions and have bounded first derivatives. We have proposed an algorithm for computing reach set of such systems with an approximation error $O(\tau + h)$, where τ is time discretization step and h is set discretization step. Application of our reachability analysis algorithm to the Brusselator example, shows that we can approximate the non-linear images more accurately than other commonly used piecewise linearization techniques. We further applied this algorithm to Artificial Pancreas, a nonlinear hybrid system with non-linear diffusion dynamics.

In our previous work, we had proposed a reachability analysis of spatio-temporal hybrid automata (STHA) [54]. The dynamics involved partial differential equation but they were linear in nature. An important future work is to consider the non-linear partial differential equations in STHA and employ exponential box spline based approximations. The main advantage of our approach is the approximation order is linear with respect to discretization factor.

Chapter 8

IMPLEMENTATION - AUTOMATED CODE GENERATION FOR MMAS

8.1 Health-Dev β Tool Architecture

Health-Dev β framework is proposed in this research [100] to allow the developer to implement MMAs following the app model in Figure 4.4 and automatically ensuring safety from interactions, sustainability in sensor and security of health data. It uses an automatic

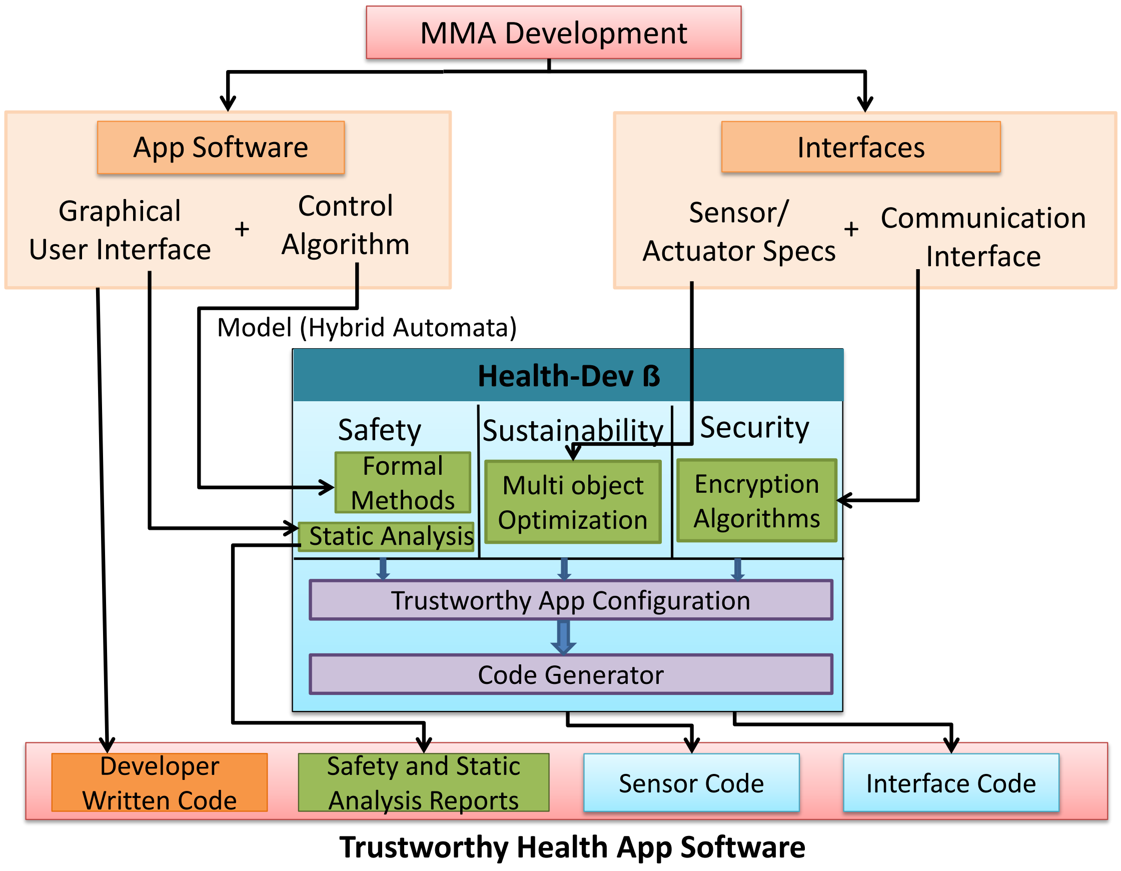


Figure 8.1: Health-Dev β Tool.

code generator to generate critical parts of the MMAs such as interfacing code, sensor or actuator code etc.

The *Health-Dev* β tool for trustworthy development of health apps has the architecture as shown in Figure 8.1. A developer can use such an architecture to either verify whether the developed MMA satisfies trustworthiness requirements or automatically generate critical parts of the code that can impose safety, sustainability or security vulnerabilities. As an input, Health-Dev β requires a high level specification of the MMA. The high level specification language used in the proposed framework is Architectural Analysis and Description Language (AADL). The high level design can be further optimized and tested using following tools:

a) sustainable sensor design optimization module: The specified sensor configuration can be optimized for energy neutrality. It is assumed that sensor operates on battery as well as scavenges energy from sources such as sunlight, body heat etc. The energy neutrality problem is modeled as a multi-objective optimization problem as discussed in Chapter 5. The proposed hybrid simulator can be used to estimate resource requirements and designing their organization for MMAs.

b) MMA system verification: For the verification of the controller of MMA, the proposed research uses (discussed in Chapter 7)model checking using rechability analysis. It is a theoretical safety verification technique which represents MMA system using hybrid automata and uses reachability analysis to determine patient safety.

c) security-enabled for sensor code and TDM app generation module: The interface specification can be used by the automatic code generator to generate sensor interface and communication code for the health app. In our previous work, we have developed Health-Dev [6], which can convert an AADL design into sensor and smartphone implementations. In addition to using the standard software primitives, Health-Dev is extended to have security plug-in, which contains pre-verified code for data communication security algorithms such as physiology value based security [66], AES encryption [101] and private-public key infrastructure [102].

The generated sensor and smartphone code can then be also passed through performance analyzers to validate the code against type and memory related errors. The performance analyzer, the MMA simulation, the reachability analysis, sustainable optimized sensor design and the security primitives generate a certification report stating their findings. The code and the certification report can then be reviewed by an expert personnel in the field to certify the application which can then be uploaded to a dedicated medical app store. If the MMA fails certification, the developer can redesign and again use the same architecture. The key feature of this architecture is that it will be implemented in a modular fashion so that the user can use different modules individually and generate only partial reports.

8.2 Security-enabled MMA Implementation and TDM Generation

MMA security vulnerabilities arise from three main sources: a) poor configuration management of APIs when developing smartphone applications, b) poor sensor code with common software errors such as unreachable code, array overflow, and inadequate input validation, and c) insecure wireless communication between smartphone and sensor. The problem is not in the technique since there are robust algorithms to avoid these security vulnerabilities. The problem as explained in [103], is an implementation problem arising from two aspects: a) misconfiguration of APIs in smartphone, and b) scarcity of resources in a sensors. These two factors coupled with the real time requirements of MMAs make it extremely non-intuitive to implement a secure MMA.

It is hypothesized that models of health apps can be optimized to obtain the correct configurations of implementation modules and then an automatic code generator can be used to reduce software errors. There are two parts of the Health-Dev β code generator:

sensor code generator and the TDM app generator. Both the code generators consist of three parts: a) high level specification module, b) a meta model consisting of code frames that can be filled by functions with appropriate attributes, and c) a database of template code with attributes that can be appropriately parameterized to fit the security needs of the application.

High-level specification: The high-level specification allows user to specify sensor and TDM app specification and translate it to respective models. In this regard, Health-Dev β tool uses Architecture Analysis and Design Language (AADL). It has been shown that the usage of AADL to specify a sensor construct defined with input and output ports [104]. The instance of that sensor consists of identifier, algorithm and communication components. It also allows to specify routing of the data through algorithms.

Meta-Model: Meta-model highlights the information content in a model. It is an abstraction of a model which is itself an abstraction of real-life system. Meta-modeling provides method to analyze, create constructs, frames, rules to model a given set of problems. The high-level specification is written under such regulation that allows parser to create a meta-model of a given model. The parser extracts out all the relevant information from the model which is further used in code generation. e.g. meta model for sensor includes platform, communication protocol, sensor type, sensing and sending frequency, sampling size etc. Meta model for smartphone can include platform, communication protocol, inbuilt sensor type etc.

Template code with attribute: Template codes are platform specific code which are used as a base to generate required code. It contains reference entities or attributes which helps parser to inject required frame (piece of a code) and brings modularity in generation. This also makes template code light-weight and generic by storing the functionalities away from it. Based upon the information obtained from meta-model, generator calls an appropriate template code and replaces its attributes by respective frames. The frame can be an algorithm or a function.

Code frame: Code frames are generic implementation of a function which could be used by generator after it is instantiated and parameterized. They are used for specifying and sequencing of algorithms. The generator injects the frame into the specific location making the generation more modular. e.g. the code generator has a repository of algorithm frames namely, peak detection, Fast Fourier Transform (FFT) and Mean. These frames are declared and sequenced in a copy of a template code by generator.

Code generator: The high level specification adheres to meta-model is parsed to create an instance of meta-model and is fed to generator as an input. Based upon the given meta-model generator pulls a copy of a platform specific template code with attributes. It applies preprocessing technique to template file, defining and expanding the required expressions. Appropriate code frames are instantiated and parameterized before getting injected into a specific location in template file.

8.2.1 TDM App Generation

The code generator also generates a TDM app which ensures the secure wireless communication. It includes the security algorithm specified by the user in the input. The code generator maintains a code template for generating TDM app. On getting the specification of the security algorithm, it pulls the algorithm from database and inserts in the code. It contains encryption-decryption algorithms such as PEES [66], Advanced Encryption Algorithm (AES) to secure the physiological data over vulnerable wireless channel. In TDM code generation, code generator uses the Application Programming Interfaces (APIs) to allow communication between external wireless mote and an Android smartphone via Bluetooth. It consists of two components, Bluetooth API and sensor handler. Bluetooth API ensures connection establishment and data communication between mote and smartphone while sensor handler acts as a manager and registers all user assigned sensor, different algorithms associated with each sensor and handling of data received from the particular sensor. For each of the wireless communication calls the generator appends the security protocol.

8.3 Performance Analysis

The performance of the MMAs can be decided by two factors: a)quality of the code and b)latency in serving the app request. To check the quality of the smartphone and sensor code in the health app, static analysis tools are used to check software errors in generated code. I proposed to perform interference analysis on MMAs in the proposed app model to check the latency in serving requests from the apps.

8.3.1 Quality of the Generated Code

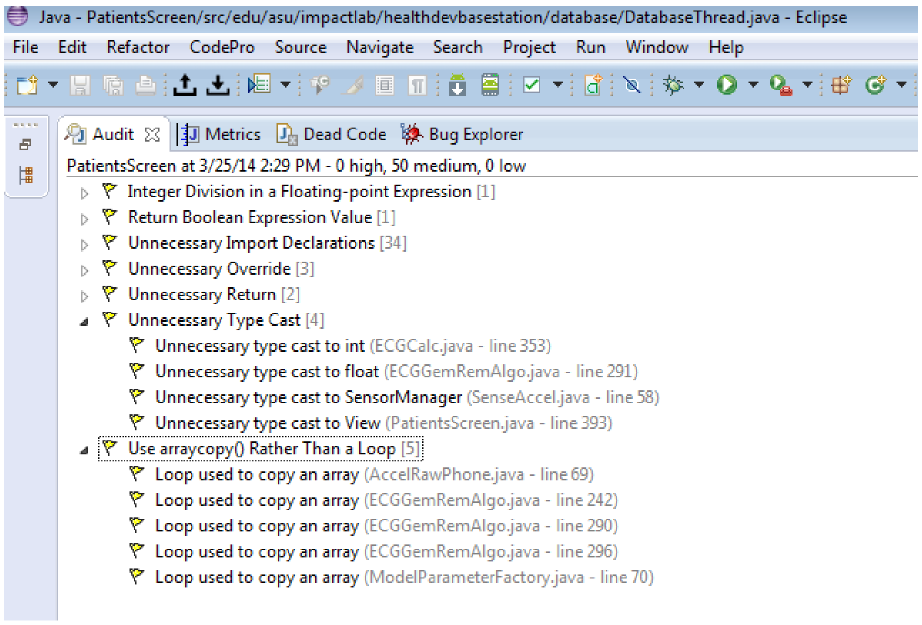
As software plays a major role in safer operation of Mobile Medical apps, there is a need to have a metric to measure the quality and efficiency of the generated code for sensor and smart phone for a personalized health monitoring system. In this regard, to validate the software of mobile medical apps, I have used static analysis tools for both smartphone and sensors code.

Static analysis for smartphone: The software for the smartphone apps typically use high level languages such as java, C#, objective C. The compilers for these languages include advanced features to find bugs or errors in the code as compared compilation features provided by low level languages such as C, nesC. However, the complexity involved in high level languages such as multithreading, methods, classes, various design types make available compilers inefficient to capture all the bugs. Thus, static analysis tools are recommended to use along with the usual compiler feedback to make code more robust.

There are many static analysis tools available such as Dexter [105], FlowDroid [106], Lint [107], Find Bugs [108], CodePro Analytics [109], PMD [110] for Android. These tools are plugins with the Android framework, which makes them easy to use. Dexter [105] and FlowDroid [106] mainly deals with security issues in Android apps such as information leakage. Lint [107] is a inbuilt static analysis tool for Eclipse which captures layout performance problems, manifest and icon problems, translation issues etc. Find Bugs [108], CodePro Analytics [109] and PMD [110] report duplicate code, generate unit test cases, computes code metrics, dead code, well explained defect reports, misunderstood API methods etc.

As an example we are show casing the results from CodePro Analytics used with our Health-Dev generated smartphone code. Figure 8.2 shows the audit report generated using CodePro Analytix tool. The report includes typecast errors, efficient ways to copy arrays as well as all the unnecessary parts added in the code such as imports, variables, return statements etc. The report also puts flags in front of each reported bug to show its severity. The audit report helps developers to reduce these errors to improve code quality and efficiency. The tool also points out the dead code sections and their exact location in all the classes using tree hierarchy.

Static analysis for sensors: The wearable sensors used in health apps typically do not have a good runtime exception capture support. For example, in TinyOS which is used by most commercially available sensors there is absolutely no support to capture array out of bound errors, null pointer errors, or race conditions in runtime. Hence safety of sensor code from runtime errors has to be ensured during the implementation and compilation phase. To avoid these errors, static analysis should be done on the developed code. There are number of such tools available for sensor code validation. Since TinyOS is a popular operating system designed for low-power wireless sensor networks, we are considering it as an example to perform static analysis in this paper. We have used TinyOS Compiler, cXprop, Safe TinyOS, Frama-c tools to do static analysis on Health-Dev [6] generated TinyOS-based sensor code. We observed that, in each case there is an increase in the code


Java - PatientsScreen/src/edu/asu/impactlab/healthdevbasestation/database/DatabaseThread.java - Eclipse
File Edit Refactor CodePro Source Navigate Search Project Run Window Help
Audit
Metrics
Dead Code
Bug Explorer

```
PatientsScreen at 3/25/14 2:29 PM - 0 high, 50 medium, 0 low
  Integer Division in a Floating-point Expression [1]
  Return Boolean Expression Value [1]
  Unnecessary Import Declarations [34]
  Unnecessary Override [3]
  Unnecessary Return [2]
  Unnecessary Type Cast [4]
    Unnecessary type cast to int (ECGCalc.java - line 353)
    Unnecessary type cast to float (ECGGemRemAlgo.java - line 291)
    Unnecessary type cast to SensorManager (SenseAccel.java - line 58)
    Unnecessary type cast to View (PatientsScreen.java - line 393)
  Use arraycopy() Rather Than a Loop [5]
    Loop used to copy an array (AccelRawPhone.java - line 69)
    Loop used to copy an array (ECGGemRemAlgo.java - line 242)
    Loop used to copy an array (ECGGemRemAlgo.java - line 290)
    Loop used to copy an array (ECGGemRemAlgo.java - line 296)
    Loop used to copy an array (ModelParameterFactory.java - line 70)
```

Figure 8.2: CodePro Analytix Audit Report.

size when Safe TinyOS is applied. This is because it uses safety annotation and stores error messages in ROM (stores code) of a micro-controller. However, cXprop used in Safe TinyOS reduces average code size cost. In order to include safety checks code size is increased and can be seen as a trade-off between safety and code size. Whereas cXprop only performs optimization without safety checks and hence the code size is found to be reduced.

Most of the embedded micro-controllers in the sensors lack memory protection and it makes the use of dynamic memory allocation risky as it may incur race conditions. The code generator ensures that the variables in the generated code don't incur race condition and makes the code safe from incurring overlap of heap and stack segment of a memory. To further evaluate the generated code, static code analyzer Frama-c [111] was used on a TinyOS application. It was observed that no error related to array (out-of-bounds) and pointers (bad pointer access) were found ensuring safety and optimal use of memory of the micro-controller.

8.3.2 *Overhead Analysis of the Proposed App Model*

For the proposed app model (Section 4.4), inter-app communication, sensor access and memory access occur through the TDM using IPC. It leads to large payloads on IPC calls. In Android, IPC calls are scheduled using the weighted fair scheduling policy [112], where bandwidth is assigned to each request in proportion to the payload as per Equation 8.1, where R is the total bandwidth in the system and w_i is the payload size of the request.

$$Bandwidth = \frac{R * w_i}{\sum\limits_{i=1}^{N} w_i} \tag{8.1}$$

If an app does not receive the required amount of bandwidth then it may be stalled and if these access requests are not served within 10 sec of their initialization, they will be prompted for killing. Such canceling of requests or un-responsiveness is unacceptable for safety critical health apps. Thus, to determine if the proposed framework, is scalable, the response time of the TDM is evaluated, which is defined to be the difference in time between a IPC request being sent and processed by the TDM. Bearing payload size in mind and considering the working of CFS, we have defined the response time model as:

$$ResponseTime = \frac{\sum\limits_{i=1}^{N} payload\_size_i + overhead}{R} \tag{8.2}$$

Overhead in smart phone can be created from events such as user interactions, incoming SMS, network connectivity, or phone calls. To validate the model, following experiments were performed to collect latency data due to TDM:

1. Varying number of requests processed by the TDM.
2. Varying the payload size of requests processed by TDM while keeping number of requests constant.
3. Effect of TDM processing on performance of the other apps on smartphone.

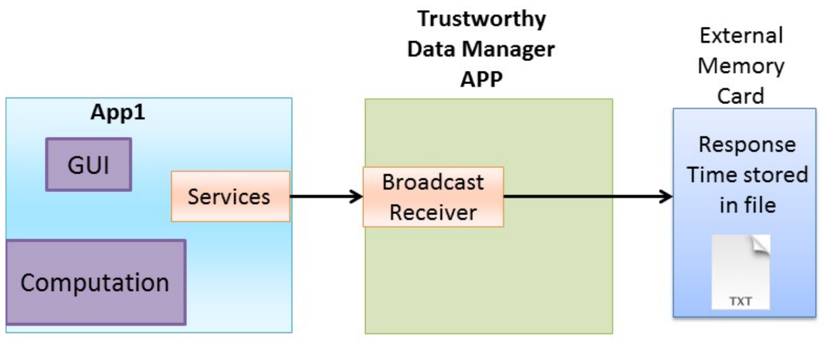


Figure 8.3: Experimental Setup for TDM Scalability.

Experimental Setup Two apps are developed, App1 and TDM for scalability analysis (Figure 8.3). App1 has broadcast service implemented and TDM receives the request through broadcast receiver. With each service request to TDM, App1 sends $\{t_{generation}$, App1$\}$, where $t_{generation}$ is the time at which the service is generated. When TDM receives and processes the request, the current system time, $t_{processing}$, is noted. Equation 8.3 gives the response time of the TDM to serve that particular request. The response time is then written to the text file on SD card. These experiments are performed on Samsung Galaxy Blaze android smartphone.

$$Response\_Time = t_{generation} + t_{processing} \tag{8.3}$$

Experiment 1: Varying number of requests: For the experimental purpose the number of requests generated by App 1 are varied as 10, 100, 500 and 10000. For each variation, the response time of requests are noted in the file on smartphone SD card. Total 12 experiments are performed in random order to reduce the error of repeating the experiments in sequential order. After calculating the average response time for a given request size, the averages are plotted as shown in Figure 8.4.

The response time approximately increases with increase in number of requests. This can be due to the request generation time is less than request transfer time over the available bandwidth. Also, the request is then processed at TDM end. Thus every time new request

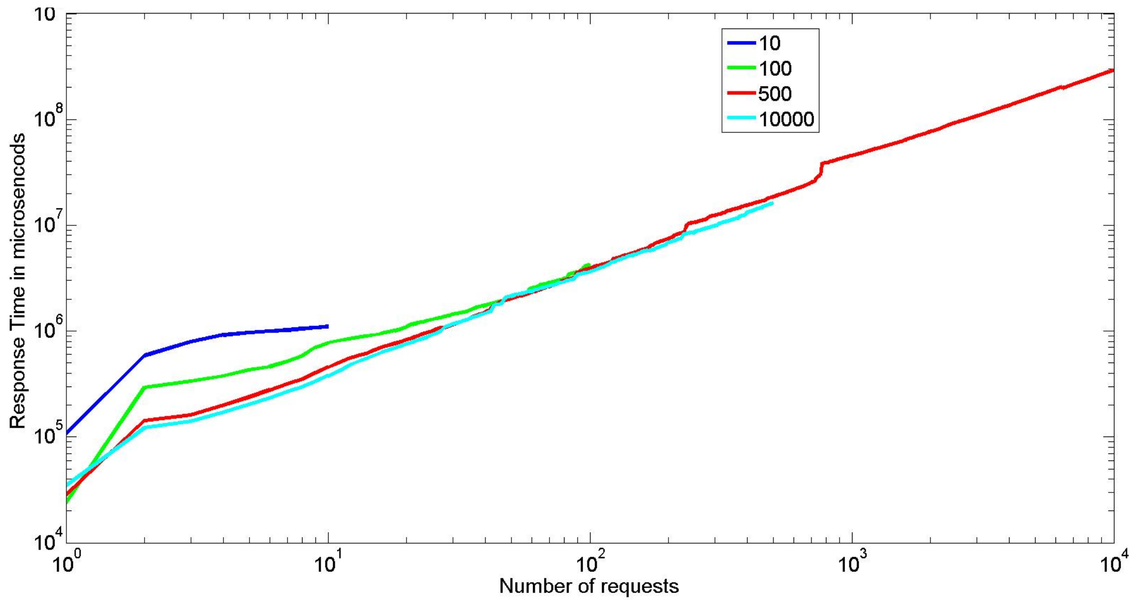


Figure 8.4: Response Time of TDM with Increased Requests.

is generated by App1 has to wait until previous request is processed. As per Equation 8.1, the bandwidth is allocated is dependent on payload size. In these experiments, payload size was constant. Thus every time same bandwidth is allocated to requests irrespective of the number of requests generated, the transfer time is constant in this case. Thus the increase in response time w.r.t. increase in requests is due to processing time at TDM end.

Experiment 2: Varying payload size of the requests: The request payload is varied as 128 bytes, 512 bytes, 1 KB, 4 KB, 16 KB and 256 KB. Number of requests generated was kept constant to 50. The payload size variation is achieved by sending array size equal to desired payload size with the request. Total 12 experiments are performed for these payload sizes with random order and then average response time is calculated. The result is as shown in Figure 8.5. The response time for payload sizes 128 bytes, 512 bytes, 1 KB, 4 KB and 16 KB is approximately in same range. However, it suddenly increases for payload size 256KB. It is due to at 256KB payload, heap memory gets filled up for each request, thus before serving the next request garbage collector runs and clears the memory.

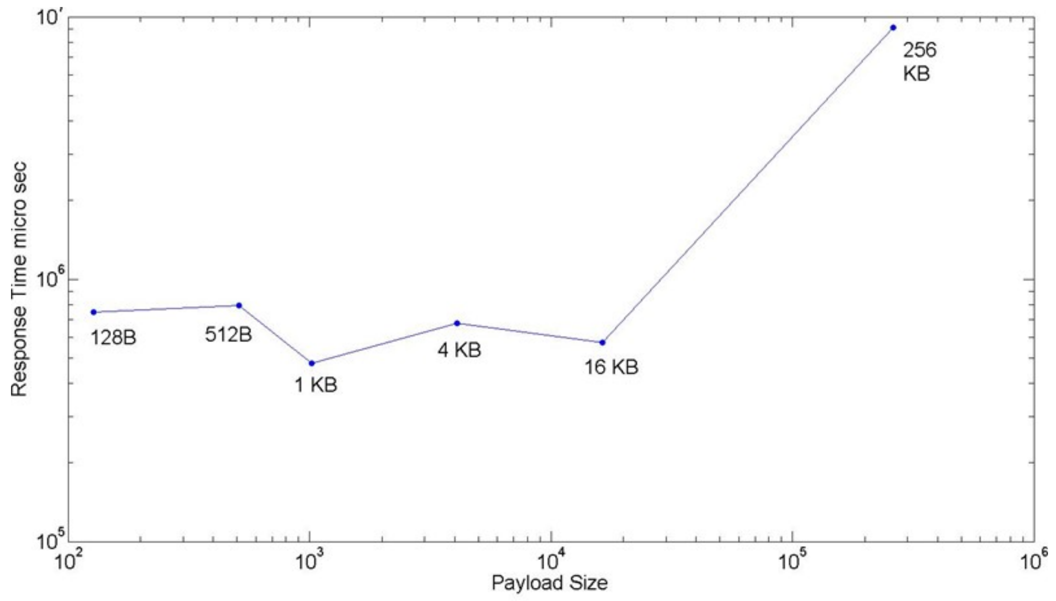


Figure 8.5: Response Time of TDM with Increased Request Payload Size.

For every request this garbage collector time has added. It led to increase in TDM response time.

Experiment 3: Effect on response time of other app: In this experiment along with App1 and TDM, one more app is implemented, non-health app which does not communicate with TDM (Figure 8.6). However this third app also accesses SD card location and write to text file.

During the experiment, the non-health app first ran without TDM processing requests and noted its response time. Then TDM has started to serve 10000 requests and again non-health app has started. It has been observed that, there is no impact of TDM working on non-health app performance (Figure 8.7). **Effect of non-health app on response time of TDM:** As smartphone is not a dedicated medical device, it might have multiple applications working such as games, phone calls, texting, internet etc.

Overhead analysis for TDM runtime validation: To check the overhead of runtime validation in TDM app, we have implemented the safety and sustainability models to get their

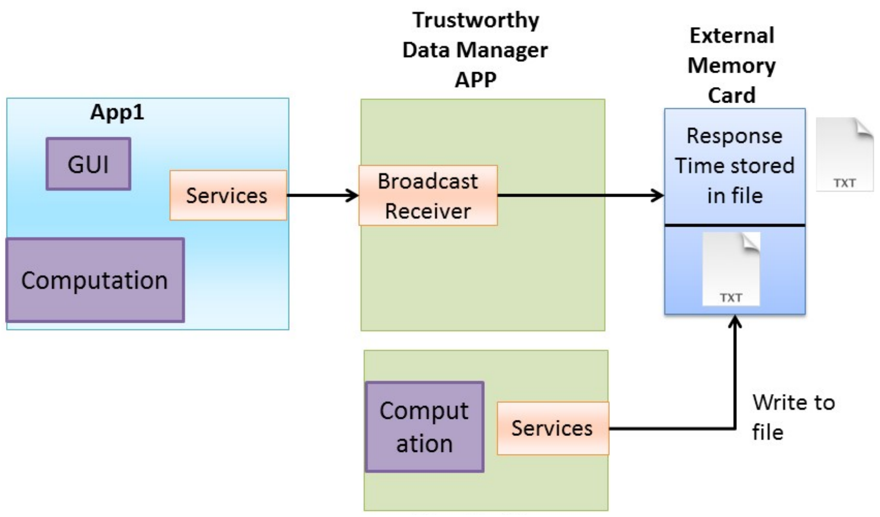


Figure 8.6: Experimental Setup to Check Impact of TDM on Performance of Other App.

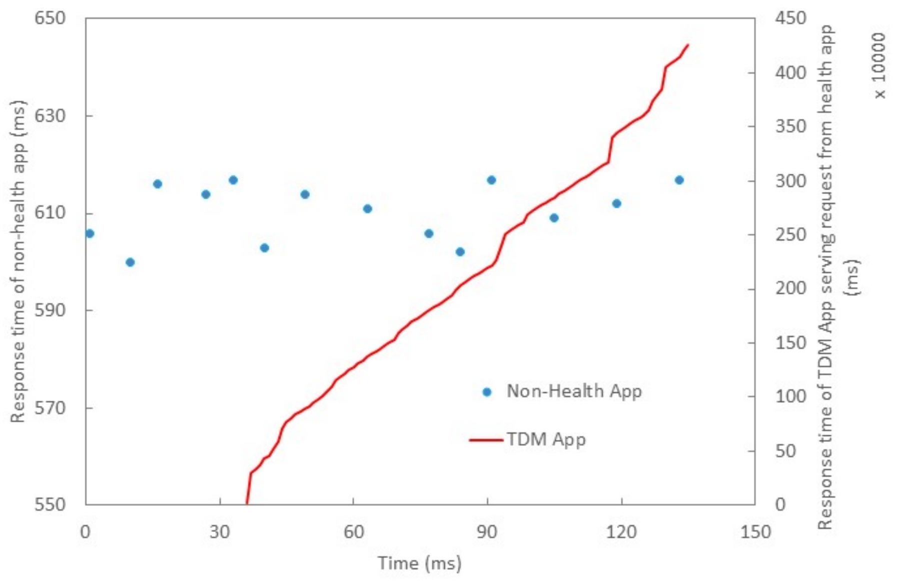


Figure 8.7: Impact of TDM on Performance of Non-health App.

execution time. Runtime security validation uses FSM which checks for the program flow with no complex computation involved thus it is assumed to no time. For safety analysis, pharmacokinetic model[113] for the analgesic infusion pump discussed in Example 3 from Chapter 4 is implemented as Android service. It took 10.894 seconds to check for the given drug concentration value if the blood glucose level will remain within safety thresholds. For sustainability, the objective function in the optimization problem formulation (Equation 5.15) is implemented as Android service to check for the given sensor configuration change if the sensor will satisfy energy neutrality constraint. The execution time for the sustainability model was 0.03 seconds.

8.3.3 MMA Development Effort Reduction using Health-Dev β

Developing trustworthy health apps while considering its interactions with the environment might increase burden on developer. Thus to validate that Health-Dev β tool doesn't hinder the development process but accelerates it by automated code generation, efforts required are estimated using Constructive Cost Model (COCOMO) [114], a cost-effort estimation tool and compared with manual implementation. As an example, we considered development of PetPeeves [37] app which requires developing user interface, ECG sensor code and communication interface between smartphone and ECG sensor. Manual implementation of PetPeeves app required 2553 lines of code (LOC) without any S3 assurance. With Health-Dev β, the communication and sensor code is automatically generated with TDM, leaving developer to write only 1678 LOC including sensor specifications. COCOMO estimated the effort required with Health-Dev β tool as 6.9 developer-month, whereas for manual implementation it came out as 12.1 developer-month which is almost 1.8 times more. The developer cost is also increased by 1.8 times for manual implementation. This shows that the proposed tool not only saves the development efforts but guarantees correct software working.

Chapter 9

CONCLUSION AND FUTURE RESEARCH DIRECTIONS

This research has focused on developing tools and techniques to enable evidence-based development of mobile medical apps. It hypothesizes that MMA should follow a standard operating model to objectively assure patient safety, wearables sustainability and health data security given the divergent app development methodologies. In this app model, every request from MMA such as sensor or actuator data access, cloud communication, data from other app, should pass through a trustworthy data manager which can ensure whether the request satisfies MMA trustworthy properties. Given such a standard operating model, the Health-Dev β can use automatic techniques to develop and generate evidences for trustworthy MMAs. It uses an optimization framework to get safe and sustainable sensor and actuator configuration. Evidence for correctness of these configurations in terms of resource allocation requirements, assuring safety and long term operation can be generated using hybrid simulator which modifies the time step of the simulation to accurately simulate system events. Further better evidence for MMA design can be hatched using reachability analysis of hybrid automata model of the app using exponential box splines approach to take into account non-linearity of the human physiological dynamics. Such dependable MMA design is then implemented using automated code generator to reduce manual implementation errors. Overhead analysis of the app model is performed in order to ensure that the model does not degrade the performance of MMA or any other non-health app in the smartphone. The current app model has considered Android architecture, however it can be easily extended to iOS.

Future research directions for trustworthy development of MMAs are described as following:

- **Extensions in TDM:**

  a) As smartphone is not a dedicated medical device, it might have multiple applications working such as games, phone calls, texting, internet etc. The bandwidth allocated to each request in smartphone is proportional to the payload size of the request as per Equation 8.1. If IPC calls for the non-health app has very large payload, it might take most of the available bandwidth. Also the proposed app model can increase the number of IPC calls since it requires that all memory access, sensor access, and communication radio access occurs through the TDM. This might limit the the availability of bandwidth to process critical MMA requests. If Android scheduler is modified to consider request priorities with highest priority given to TDM, there will be no degradation in the response time for TDM requests. The immediate extension to the TDM approach can be to explore solutions such as priority based scheduler for medical apps to reduce response or stall times of MMAs.

  b) Assuming TDM as the most trustworthy entity in the smartphone for MMAs, all the requests for memory access, sensor access, and communication radio access occurs through it. However, it can act as a single point of failure problem. Any malicious app can modify the code for TDM to affect its functionality. Current TDM design does not check for security attack on it. One possible solution can be to have replicas of TDM in a smartphone. It can allow to check the behavior of the TDM for every request by processing it on multiple TDM replicas. depending on the majority of the same output, failed or compromised TDM can be detected. However this method can lead to overhead of TDM replicas in smartphone. Thus it is necessary to find a solution to avoid overhead of TDM replicas and single point of failure of the proposed approach.

- **Controller Synthesis to obtain safe controller design**: The proposed reachability analysis technique uses non-linear continuous dynamics and by that it reduces error in estimating controller safety properties. However, the proposed technique assumes sensors and actuators with no delay, zero sensing error, zero precision error of the display devices and estimate the safety of the controller design. In practice, the obtained safe controller design may not be implementable or even may not be safe when these delays and errors are considered. Thus the immediate extension to this technique can be to come up with a mathematical model (variation of hybrid automata) which can consider these delays and errors so that the non-linear reachability analysis technique can be applied to it to get implementable controller design.

- **Reachability Analysis of Non-linear spatio-temporal hybrid systems**: The unintentional interactions in human physiology and controller in medical app are usually spatio-temporal and non-linear in nature. This research has focused on developing a technique to consider non-linearity in physical dynamics in reachability analysis. However, it fails to capture the spatial nature of the system dynamics while considering only temporal factors. Spatial dynamics play an crucial role in physical systems, e.g. in case multi-drug infusion system, spatial parameter can predict the blood glucose concentration at different locations in human body accurately. Thus to get more accurate evidence for MMA design safety, the reachability analysis of nonlinear systems should be extended to include spatial dynamics.

- **Clinical Study**: Clinical study on Health-Dev [6] generator code for ECG app has been performed by IMPACT lab in St. Luke's hospital. Similar study can be performed on Health-Dev β generated code with TDM for controller MMAs such as infusion pump apps. This study will act as a validation to the trustworthy MMA development approach proposed in this research, e.g. check if the sensor works for the specified time without recharging the battery for a given optimized sensor design obtained from the proposed framework or whether the safe controller as certified by the Health-Dev β does not lead to hypoglycemia ever in clinical settings.

REFERENCES

[1] FDA. Mobile medical applications. http://www.fda.gov/medicaldevices/productsand medicalprocedures/connectedhealth/mobilemedicalapplications/default.htm.

[2] Claudio Cobelli, Eric Renard, and Boris Kovatchev. Artificial pancreas: past, present, future. *Diabetes*, 60(11):2672–2682, 2011.

[3] Artifical Pancreas. The artificial pancreas aces new tests. http://www.diabetesforecast.org/2014/mar/the-artificial-pancreas-aces.html.

[4] Priyanka Bagade, Ayan Banerjee, and Sandeep K. S. Gupta. Evidence-based development approach for safe, sustainable and secure mobile medical app. In *Wearable Electronics Sensors*, pages 135–174. Springer, 2015.

[5] Sidharth Nabar, Ayan Banerjee, Sandeep K. S. Gupta, and Radha Poovendran. Evaluation of body sensor network platforms: a design space and benchmarking analysis. In *Wireless Health 2010*. ACM.

[6] Ayan Banerjee, Sunit Verma, Priyanka Bagade, and Sandeep K. S. Gupta. Health-dev: Model based development pervasive health monitoring systems. In *Wearable and Implantable Body Sensor Networks (BSN), 2012 Nineth International Conference on*, pages 85 –90, may 2012.

[7] Priyanka Bagade, Ayan Banerjee, and Sandeep K. S. Gupta. Optimal design for symbiotic wearable wireless sensors. In *Wearable and Implantable Body Sensor Networks (BSN), 2014 11th International Conference on*, pages 132–137. IEEE, 2014.

[8] Thomas R. Rossiter and Theodore J. LaVaque. A comparison of eeg biofeedback and. *Journal of Neurotherapy*, 1995.

[9] Mike Borrello. Modeling and control of systems for critical care ventilation. In *American Control Conference, 2005. Proceedings of the 2005*, pages 2166–2180. IEEE, 2005.

[10] Krishna Kumar Venkatasubramanian, Ayan Banerjee, and Sandeep K. S. Gupta. Pska: usable and secure key agreement scheme for body area networks. *Information Technology in Biomedicine, IEEE Transactions on*, 14(1):60–68, 2010.

[11] Thomas A Henzinger. *The theory of hybrid automata*. Springer, 2000.

[12] Mark Di Sano, Andres Perez, Miguel A Labrador, Ponrathi Athilingam, and Federico Giovannetti. Heartmapp: a mobile application to improve chf outcomes and reduce hospital readmissions. In *Proceedings of the conference on Wireless Health*, page 20. ACM, 2015.

[13] Junghyo Lee, Ayan Banerjee, Prajwal Paudyal, and Sandeep K. S. Gupta. Mt-diet: Automated diet assessment using myo and thermal. In *Late-Breaking Research Abstract at the conference on Wireless Health*, page 20. ACM, 2015.

[14] Divya Periyakoil. Paingauge: A feasibility study comparing the effectiveness of a mobile health app and a pain diary in assessing pain. In *143rd APHA Annual Meeting and Exposition (October 31-November 4, 2015)*. APHA, 2015.

[15] Le Minh Khue, Eng Lieh Ouh, and Stan Jarzabek. Mood self-assessment on smartphones. In *Proceedings of the conference on Wireless Health*, page 19. ACM, 2015.

[16] Xiang Chen, A.B. Waluyo, I. Pek, and Wee-Soon Yeoh. Mobile middleware for wireless body area network. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 5504 –5507, 31 2010-sept. 4 2010.

[17] Bonifaz Kaufmann and Leah Buechley. Amarino: a toolkit for the rapid prototyping of mobile ubiquitous computing. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 291–298. ACM, 2010.

[18] IOIO for Android. Android development tools. https://www.sparkfun.com/products/retired/10748.

[19] T Laakko, J Leppänen, J Lähteenmäki, A Nummiaho, et al. Mobile health and wellness application framework. *Methods Inf Med*, 47(3):217–222, 2008.

[20] Tathagata Das, Prashanth Mohan, Venkata N Padmanabhan, Ramachandran Ramjee, and Asankhaya Sharma. Prism: platform for remote sensing using smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 63–76. ACM, 2010.

[21] Waylon Brunette, Rita Sodt, Rohit Chaudhri, Mayank Goel, Michael Falcone, Jaylen Van Orden, and Gaetano Borriello. Open data kit sensors: a sensor integration framework for android at the application-level. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 351–364. ACM, 2012.

[22] Felix Xiaozhu Lin, Ahmad Rahmati, and Lin Zhong. Dandelion: a framework for transparently programming phone-centered wireless body sensor applications for health. In *Wireless Health 2010*, pages 74–83. ACM, 2010.

[23] Felix Xiaozhu Lin, Zhen Wang, Robert LiKamWa, and Lin Zhong. Reflex: using low-power processors in smartphones without knowing them. *ACM SIGARCH Computer Architecture News*, 40(1):13–24, 2012.

[24] Bodhi Priyantha, Dimitrios Lymberopoulos, and Jie Liu. Enabling energy efficient continuous sensing on mobile phones with littlerock. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 420–421. ACM, 2010.

[25] Jacob Sorber, Nilanjan Banerjee, Mark D Corner, and Sami Rollins. Turducken: hierarchical power management for mobile devices. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 261–274. ACM, 2005.

[26] Jun Bum Lim, Beakcheol Jang, Suyoung Yoon, Mihail L. Sichitiu, and Alexander G. Dean. Raptex: Rapid prototyping tool for embedded communication systems. *ACM Trans. Sen. Netw.*, 7:7:1–7:40, August 2010.

[27] Elaine Cheong, Edward A. Lee, and Yang Zhao. Viptos: a graphical development and simulation environment for tinyos-based wireless sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys '05, pages 302–302, New York, NY, USA, 2005. ACM.

[28] M.M.R. Mozumdar, F. Gregoretti, L. Lavagno, L. Vanzago, and S. Olivieri. A framework for modeling, simulation and automatic code generation of sensor network application. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON '08. 5th Annual IEEE Communications Society Conference on*, pages 515 –522, june 2008.

[29] BaekGyu Kim, Linh TX Phan, Oleg Sokolsky, and Insup Lee. Platform-dependent code generation for embedded real-time software. In *Compilers, Architecture and Synthesis for Embedded Systems (CASES), International Conference on*, pages 1–10. IEEE, 2013.

[30] Mersini Paschou, Evangelos Sakkopoulos, and Athanasios Tsakalidis. easyhealthapps: e-health apps dynamic generation for smartphones & tablets. *Journal of medical systems*, 37(3):1–12, 2013.

[31] Sam Procter and John Hatcliff. An architecturally-integrated, systems-based hazard analysis for medical applications. In *Formal Methods and Models for Codesign (MEMOCODE), 2014 Twelfth ACM/IEEE International Conference on*, pages 124–133. IEEE, 2014.

[32] Priyanka Bagade, Ayan Banerjee, Sunit Verma, Joseph Milazzo, and Sandeep K. S. Gupta. A holistic tool for developing a wireless home-based health monitoring system. *Biomedical Instrumentation & Technology*, 47(s1):64–71, 2013.

[33] Sunit Verma, Joseph Milazzo, Yu Xie, Priyanka Bagade, Ayan Banerjee, and Sandeep K. S. Gupta. Model-based wireless health system design tool. In *Proceedings of the conference on Wireless Health*, page 19. ACM, 2012.

[34] Byung-Gon Chun, Carlo Curino, Russell Sears, Alexander Shraer, Samuel Madden, and Raghu Ramakrishnan. Mobius: unified messaging and data serving for mobile apps. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 141–154. ACM, 2012.

[35] Nitin Agrawal, Akshat Aranya, and Cristian Ungureanu. Mobile data sync in a blink. In *Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems*, San Jose, CA, 2013. USENIX.

[36] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A middleware for building context-aware mobile services. In *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, volume 5, pages 2656–2660. IEEE, 2004.

[37] Joseph Milazzo, Priyanka Bagade, Ayan Banerjee, and Sandeep K. S. Gupta. bhealthy: A physiological feedback-based mobile wellness application suite. In *Proceedings of the conference on Wireless Health*. ACM, 2013.

[38] Joseph Milazzo, Priyanka Bagade, Ayan Banerjee, and Sandeep K. S. Gupta. Sharing is caring: A data exchange framework for colocated mobile apps. In *Third International Conference on Mobile and Wireless Networks*, 2014.

[39] Deborah Estrin and Ida Sim. Open mhealth architecture: an engine for health care innovation. *Science(Washington)*, 330(6005):759–760, 2010.

[40] D Russell Wada and Denham S Ward. The hybrid model: a new pharmacokinetic model for computer-controlled infusion pumps. *Biomedical Engineering, IEEE Transactions on*, 41(2):134–142, 1994.

[41] Bo Li, Zhuoxiong Sun, Kirill Mechitov, Gregory Hackmann, Chenyang Lu, Shirley J Dyke, Gul Agha, and Billie F Spencer Jr. Realistic case studies of wireless structural control. In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems*, pages 179–188. ACM, 2013.

[42] Anton Cervin, Dan Henriksson, Bo Lincoln, Johan Eker, and Karl-Erik Årzén. How does control timing affect performance? *IEEE control systems magazine*, 23(3):16–30, 2003.

[43] Emeka Eyisi, Jia Bai, Derek Riley, Jiannian Weng, Wei Yan, Yuan Xue, Xenofon Koutsoukos, and Janos Sztipanovits. Ncswt: An integrated modeling and simulation tool for networked control systems. *Simulation Modelling Practice and Theory*, 27:90–111, 2012.

[44] T Kohtamaki, Mikael Pohjola, Jenna Brand, and Lasse M Eriksson. Piccsim toolchain-design, simulation and automatic implementation of wireless networked control systems. In *Networking, Sensing and Control, 2009. ICNSC'09. International Conference on*, pages 49–54. IEEE, 2009.

[45] Behdad Aminian, José Araújo, Mikael Johansson, and Karl H Johansson. Gisoo: a virtual testbed for wireless cyber-physical systems. In *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE*, pages 5588–5593. IEEE, 2013.

[46] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003.

[47] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. In *Local Computer Networks, Proceedings 31st IEEE Conference on*, 2006.

[48] Jong Suk Ahn and Peter B Danzig. Packet network simulation: speedup and accuracy versus timing granularity. *IEEE/ACM Transactions on Networking (TON)*, 4(5):743–757, 1996.

[49] G Kesidis, A Singh, D Cheung, and WW Kwok. Feasibility of fluid event-driven simulation for atm networks. In *Global Telecommunications Conference, Communications: The Key to Global Prosperity*, volume 3, pages 2013–2017, 1996.

[50] Cameron Kiddle, Rob Simmonds, Carey Williamson, and Brian Unger. Hybrid packet/fluid flow network simulation. In *Parallel and Distributed Simulation, 2003.(PADS 2003). Proceedings. Seventeenth Workshop on*, pages 143–152. IEEE, 2003.

[51] Benjamin Melamed, Shuo Pan, and Yorai Wardi. Hns: A streamlined hybrid network simulator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 14(3):251–277, 2004.

[52] Jason Liu. Parallel simulation of hybrid network traffic models. In *Principles of Advanced and Distributed Simulation, 2007. PADS'07. 21st International Workshop on*, pages 141–151. IEEE, 2007.

[53] Priyanka Bagade, Ayan Banerjee, and Sandeep K. S. Gupta. Validation, verification and formal methods for cyber-physical systems. In *Cyber-Physical Systems: Foundations, Principles and Applications*. Elsevier, 2015.

[54] Ayan Banerjee and Sandeep K. S. Gupta. Spatio-temporal hybrid automata for safe cyber-physical systems: A medical case study. In *IntlConfon Cyber-Physical Systems (Accepted for Publication)*, 2013.

[55] Priyanka Bagade, Ayan Banerjee, and Sandeep K. S. Gupta. Safety assurance of medical cyber-physical systems using hybrid automata: A case study on analgesic infusion pump. *Workshop on Medical Cyber-Physical Systems*, 2013.

[56] Eugene Asarin, Thao Dang, and Antoine Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43(7):451–476, 2007.

[57] Eugene Asarin, Thao Dang, and Antoine Girard. Reachability analysis of nonlinear systems using conservative approximation. In *Hybrid Systems: Computation and Control*, pages 20–35. Springer, 2003.

[58] Thao Dang, Oded Maler, and Romain Testylier. Accurate hybridization of nonlinear systems. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 11–20. ACM, 2010.

[59] Thao Dang, Colas Le Guernic, and Oded Maler. Computing reachable states for nonlinear biological models. In *Computational Methods in Systems Biology*, pages 126–141. Springer, 2009.

[60] Thomas A Henzinger, Benjamin Horowitz, Rupak Majumdar, and Howard Wong-Toi. Beyond hytech: Hybrid systems analysis using interval numerical methods. In *Hybrid systems: Computation and control*, pages 130–144. Springer, 2000.

[61] Matthias Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 173–182. ACM, 2013.

[62] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. In *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 216–232. Springer, 2007.

[63] Alessandro Cimatti, Sergio Mover, and Stefano Tonetta. A quantifier-free smt encoding of non-linear hybrid automata. In *Formal Methods in Computer-Aided Design (FMCAD), 2012*, pages 187–195. IEEE, 2012.

[64] Pamela Lewis Dolan. http://exclusive.multibriefs.com/content/health-app-certification-program-halted.

[65] Fei Hu, Qi Hao, Marcin Lukowiak, Qingquan Sun, Kyle Wilhelm, Stanislaw Radziszowski, and Yao Wu. Trustworthy data collection from implantable medical devices via high-speed security implementation based on ieee 1363. *Information Technology in Biomedicine, IEEE Transactions on*, 14(6):1397–1404, 2010.

[66] Ayan Banerjee, Sandeep K. S. Gupta, and Krishna Kumar Venkatasubramanian. Pees: physiology-based end-to-end security for mhealth. In *Wireless Health*, page 2. Citeseer, 2013.

[67] Saleem Garawi, Robert SH Istepanian, and Mosa Ali Abu-Rgheff. 3g wireless communications for mobile robotic tele-ultrasonography systems. *Communications Magazine, IEEE*, 44(4):91–96, 2006.

[68] Robert SH Istepanian, Emil Jovanov, and YT Zhang. Guest editorial introduction to the special section on m-health: Beyond seamless mobility and global wireless health-care connectivity. *Information Technology in Biomedicine, IEEE Transactions on*, 8(4):405–414, 2004.

[69] Priyanka Bagade, Ayan Banerjee, Joseph Milazzo, and Sandeep K. S. Gupta. Protect your bsn: No handshakes, just namaste! In *Body Sensor Networks (BSN), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013.

[70] Mocana. Harden mobile apps with the mocana atlas platform whitepaper. https://www.mocana.com/harden-mobile-apps-with-the-mocana-atlas-platform.

[71] Gyuhae Park, Tajana Rosing, Michael D Todd, Charles R Farrar, and William Hodgkiss. Energy harvesting for structural health monitoring sensor networks. *Journal of Infrastructure Systems*, 14(1):64–79, 2008.

[72] Yu-Te Liao, Huanfen Yao, A. Lingley, B. Parviz, and B.P. Otis. A 3- μ W cmos glucose sensor for wireless contact-lens tear glucose monitoring. *Solid-State Circuits, IEEE Journal of*, 47(1):335–344, Jan 2012.

[73] Trachette L. Jackson and Helen M. Byrne. A mathematical model to study the effects of drug resistance and vasculature on the response of solid tumors to chemotherapy. *Mathematical Biosciences*, 164(1):17 – 38, 2000.

[74] H. H Pennes. Analysis of tissue and arterial blood temperature in the resting human forearm. In *Journal of Applied Physiology*, volume 1.1, pages 93–122, 1948.

[75] Qinghui Tang, Naveen Tummala, Sandeep K. S. Gupta, and Loren Schwiebert. Communication scheduling to minimize thermal effects of implanted biosensor networks in homogeneous tissue. *Biomedical Engineering, IEEE Transactions on*, 52(7):1285–1294, July 2005.

[76] Remus Teodorescu Dezso Sera and Pedro Rodriguez. Pv panel model based on datasheet values. In *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pages 2392–2396. IEEE, 2007. sera2007pv.

[77] MSP430 solar energy harvesting development tool from Texas Instruments. http://www.ti.com/tool/ez430-rf2500-seh.

[78] Priyanka Bagade, Ayan Banerjee, and Sandeep K. S. Gupta. Hyrefsim: Hybrid simulator for cyber-physical networks. *(In preparation)*.

[79] Madalin Gavrilescu, Gabriela Magureanu, Dan Pescaru, and Alex Doboli. A simulation framework for psoc based cyber physical systems. In *Computational Cybernetics and Technical Informatics (ICCC-CONTI), 2010 International Joint Conference on*, pages 137–142. IEEE, 2010.

[80] Vivek K Singh and Ramesh Jain. Situation based control for cyber-physical environments. In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pages 1–7. IEEE, 2009.

[81] Wei Yan, Yuan Xue, Xiaowei Li, Jiannian Weng, Timothy Busch, and Janos Sztipanovits. Integrated simulation and emulation platform for cyber-physical system security experimentation. In *Proceedings of the 1st international conference on High Confidence Networked Systems*, pages 81–88. ACM, 2012.

[82] Sahra Lin, Jing andSedigh and Ann Miller. Integratedcyber-physical simulation of intelligent water distribution networks. *Matlab/Book2 (EPLeite, ed.), Intech*, 2011.

[83] Dan Henriksson, Hilding Elmqvist, et al. Cyber-physical systems modeling and simulation with modelica. In *International Modelica Conference, Modelica Association*, volume 9, 2011.

[84] Haruki Tsuchiya and Osamu Kobayashi. Mass production cost of pem fuel cell by learning curve. *International Journal of Hydrogen Energy*, 2004.

[85] B.E. Shi. Cnn models of current mode neuromorphic networks. In *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, volume 3, pages 337 –340 vol. 2, may 2001.

[86] B.E. Shi. Pseudoresistive networks and the pseudovoltage-based cocontent. volume 50, pages 56 – 64, jan. 2003.

[87] P.D. Smith and P. Hasler. A programmable diffuser circuit based on floating-gate devices. In *Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on*, volume 1, pages I – 291–4 vol.1, aug. 2002.

[88] F. Baskaya, B. Gestner, C. Twigg, Sung Kyu Lim, D.V. Anderson, and P. Hasler. Rapid prototyping of large-scale analog circuits with field programmable analog array. In *Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on*, pages 319 –320, april 2007.

[89] Ayan Banerjee, Priyanka Bagade, and Sandeep K. S. Gupta. Approximation of reach set of a non-linear hybrid automata using exponential box splines. *(In preparation)*.

[90] Coleri Sinem, Ergen Mustafa, and Koo T. John. Lifetime analysis of a sensor network with hybrid automata modelling. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 98–104, New York, NY, USA.

[91] Yi Wang, Kent M. Eskridge, and Andrzej T. Galecki. Unknown title. *Journal of Health*, 2(3):188–194, 2010.

[92] P.S. Crooke, J.R. Hotchkiss, and J.J. Marini. Linear and nonlinear mathematical models for noninvasive ventilation. *Mathematical and Computer Modelling*, 35(1112):1297 – 1313, 2002.

[93] Reza Mohammadi. Exponential b-spline solution of convection-diffusion equations. *Applied Mathematics*, 4:933 – 944, 2013.

[94] L.M. Koci and G.V. Milovanovi. Shape preserving approximations by polynomials and splines. *Computers & Mathematics with Applications*, 33(11):59 – 97, 1997.

[95] Amos Ron. Exponential box splines. *Constructive Approximation*, 4(1):357–378, 1988.

[96] Nira Dyn and Amos Ron. Local approximation by certain spaces of exponential polynomials, approximation order of exponential box splines, and related interpolation problems. *Transactions of the American Mathematical Society*, 319(1):381–403, 1990.

[97] Toshimitsu Ushio and Shigemasa Takai. Control-invariance of hybrid systems with forcible events. *Automatica*, 41(4):669–675, apr 2005.

[98] Helmut Pottmann. The geometry of tchebycheffian splines. *Computer Aided Geometric Design*, 10(34):181 – 210, 1993.

[99] Michael Ian Shamos and Dan Hoey. Geometric intersection problems. In *Foundations of Computer Science, 1976., 17th Annual Symposium on*, pages 208–215. IEEE, 1976.

[100] Priyanka Bagade, Ayan Banerjee, and Sandeep K. S. Gupta. Health-dev beta a trustworthy health app development tool. *(In preparation)*.

[101] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer, 2002.

[102] Sattam S Al-Riyami and Kenneth G Paterson. Certificateless public key cryptography. In *Advances in Cryptology-ASIACRYPT 2003*, pages 452–473. Springer, 2003.

[103] Ayan Banerjee, Krishna Kumar Venkatasubramanian, and Sandeep K. S. Gupta. Challenges of implementing cyber-physical security solutions in body area networks. In *Proc. of Intl. Conf. on Body Area Networks*, 2009.

[104] Ayan Banerjee and Sandeep K. S. Gupta. Analysis of smart mobile applications for healthcare under dynamic context changes. *Mobile Computing, IEEE Transactions on*, 2014.

[105] Dexter. http://dexter.dexlabs.org/.

[106] Flowdroid. http://sseblog.ec-spride.de/tools/flowdroid/.

[107] Lint. http://developer.android.com/tools/help/lint.html.

[108] Findbugs. http://findbugs.sourceforge.net/.

[109] Codeproanalytics. https://marketplace.eclipse.org/content/codepro-analytix.

[110] Pmd. http://pmd.sourceforge.net/.

[111] Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-c. In George Eleftherakis, Mike Hinchey, and Mike Holcombe, editors, *Software Engineering and Formal Methods*, volume 7504 of *Lecture Notes in Computer Science*, pages 233–247. Springer Berlin Heidelberg, 2012.

[112] Cheng-Kang Hsieh, Hossein Falaki, Nithya Ramanathan, Hongsuda Tangmunarunkit, and Deborah Estrin. Performance evaluation of android ipc for continuous sensing applications. *ACM SIGMOBILE Mobile Computing and Communications Review*, 16(4):6–7, 2013.

[113] D.R. Wada and D.S. Ward. The hybrid model: a new pharmacokinetic model for computer-controlled infusion pumps. *Biomedical Engineering, IEEE Transactions on*, 41(2):134 –142, feb. 1994.

[114] Cocomo model. http://csse.usc.edu/tools/COCOMOII.php.