DESIGN FOR SURVIVABILITY IN CRITICAL INFRASTRUCTURE SAFETY APPLICATIONS

A Dissertation
Presented in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
with a
Major in Computer Science
in the
College of Graduate Studies
University of Idaho

by
Ahmed Abdelhamid Ahmed Serageldin

August 2014

Major Professor: Axel Krings, Ph.D.

AUTHORIZATION TO SUBMIT DISSERTATION

This dissertation of Ahmed Abdelhamid Ahmed Serageldin, submitted for the degree of Doctor of Philosophy with a major in Computer Science and titled "Design for Survivability in Critical Infrastructure Safety Applications," has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor _____ Date _____
Axel Krings, Ph.D.

Committee Member _____ Date _____
Robert Rinker, Ph.D.

Committee Member _____ Date _____
Li Tan, Ph.D.

Committee Member _____ Date _____
Ahmed Abdel-Rahim, Ph.D.

Department Administrator _____ Date _____
Gregory Donohoe, Ph.D.

Discipline's College Dean _____ Date _____
Larry Stauffer, Ph.D.

Final Approval and Acceptance

Dean of the College
of Graduate Studies:

_____ Date _____
Jie Chen, Ph.D.

ABSTRACT

Transportation systems, and thus Intelligent Transportation Systems (ITS), are one of the critical infrastructures. At the core of ITS are safety critical applications, in which any fault, may it be of benign or malicious nature, could have far-reaching consequences. Therefore, reliability, security, and survivability are of paramount importance.

In this dissertation, we present survivability solutions for two types of ITS application domains. The first domain involves the Connected Vehicles Infrastructure (CVI), and the second domain a Weather Responsive System Infrastructure (WRSI). Both application domains have in common that they are part of safety critical infrastructures, and thus any failure can lead to injury or loss of life. Given the criticality of the systems, fault-tolerance and survivability considerations have to be designed into the systems, rather than in an add-on fashion.

Therefore, in our proposed solutions we will demonstrate survivability mechanisms that employ an approach known as *Design for Survivability*. Specifically, solutions based on redundancy in the context of hybrid fault models are proposed. The solutions presented for both application domains do not require any modification of existing infrastructure components or standards. In the absence of such solutions malicious faults could render the applications useless.

In the CVI application domain, a model to analyze and quantify the reliability of Dedicated Short Range Communication (DSRC) safety applications is introduced. An approach is given to utilize channel redundancy to mitigate against the impact of communication jamming. In addition to channel redundancy message dissimilarity, using different message types, is employed. The approaches are analyzed and the results show survivability improvements of the safety applications.

In the WRSI application domain, the main theoretical contributions are the combination and extension of the approaches introduced in previous work. The theory of certifying executions is extended by three concepts. First, the detection of dependency violations,

exceptions triggers, and sensor analysis are considered. Furthermore, a dual-bound threshold approach for detecting off-nominal executions is introduced. Lastly, profiling is augmented with the concept of behavior sets. Extensive evidence of the effectiveness of the solutions based on a one-year observation of the system in action is presented.

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my major professor Axel Krings, for his excellent guidance, patience, caring and providing me with an excellent atmosphere for doing research. Without his guidance and persistent help this dissertation would not have been possible.

I would like to thank Dr. Ahmed Abdel-Rahim for supporting my research and continually and convincingly conveyed a spirit of adventure in regard to practical work.

I would like to thank my committee members, Dr. Robert Rinker and Dr. Li Tan, for their support and guidance.

I am deeply grateful to University of Idaho, Computer Science faculty, staff and students for how they have impacted my life and studies and for providing enabling environment for me to work.

Dedication

I would like to dedicate this dissertation to praise GOD , and ask for his peace and blessing to reward all those who contributed to the completion of this research the best reward, and to make it a useful knowledge.

A special feeling of deepest gratitude to my family for their support, patient and encouragement towards the completion of this work.

My mother, Magda Serageldin.

My father, Abdelhamid Serageldin.

My wife, Norhane Elalayly.

My sons Adam, Abdelrahman and Eyad.

My brother, Tamer Serageldin.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

The following abbreviations were used in this dissertation:

ITS	Intelligent Transportation Systems
USDOT	United States Department of Transportation
RITA	Research and Innovative Technology Administration
ISTEA	Intermodal Surface Transportation Efficiency Act
IVHS	Intelligent Vehicle Highway Systems
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure
I2V	Infrastructure-to-Vehicle
I2I	Infrastructure-to-Infrastructure
CVI	Connected Vehicles Infrastructure
WRSI	Weather Responsive System Infrastructure
DSRC	Dedicated Short Range Communication
NTCIP	National Transportation Communications for Intelligent Transportation System Protocol
BSM	Basic Safety Message
VSC-A	Vehicle Safety Communications - Applications
ASTM	American Society for Testing and Materials
WLAN	Wireless Local Area Networks
FHWA	Federal Highway Administration
WAVE	Wireless Access in Vehicular Environments
RSU	Road Side Unit
OBU	On-Board Unit
FCC	Federal Communication Commission
CCH	Control Channel
SCH	Service Channel
EIRP	Effective Isotropic Radiated Power
VSC 2 C	Vehicle Safety Communications 2 Consortium
NHTSA	National Highway Traffic Safety Administration
VANET	Vehicular ad hoc Networks
MAC	Medium Access Control
PHY	Physical

OFDM	Orthogonal Frequency Division Multiplexing
BPSK	Binary Phase Shift Keying
QPSK	Quadrature Phase Shift Keying
QAM	Quadrature Amplitude Modulation
WSA	WAVE Service Advertisement
CA	Certificate Authority
ECDSA	Elliptic Curve Digital Signature Algorithm
WME	WAVE Management Entity
TA	Timing Advertisement
WSMP	WAVE Short Message Protocol
IPv6	Internet Protocol Version 6
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
WSM	WAVE Short Messages
SAE	Society of Automotive Engineers
HV	Host Vehicle
RV	Remote Vehicle
EEBL	Emergency Electronic Brake Lights
FCW	Forward Collision Warning
BSW	Blind Spot Warning
LCW	Lane Change Warning
DNPW	Do Not Pass Warning
IMA	Intersection Movement Assist
CLW	Control Loss Warning
ACM	À la Carte message
PVD	Probe Vehicle Data
PDM	Probe Data Management Message
CCI	Cross Channel Interference
WSMP-S	Wave Short Message Protocol Safety Supplement
RSA	Road Side Alert
WDoS	Wireless Denial of Service
PER	Packet Error Ratio
SJR	Signal-to-Jamming Ratio
JSR	Jamming-to-receiver Signal Ratio

AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
ESS	Environmental Sensor Stations
AES	Advanced Encryption Standard
SSL	Secure Sockets Layer
STMP	Simple Transportation Management Protocol
CSV	Comma Separated Value
LCS	Local Clarus Server
URL	Uniform Resource Locator
ST	Surface Status

Chapter 1

INTRODUCTION AND BACKGROUND

1.1 Intelligent Transportation Systems as Critical Infrastructure

Several presidential directives, including the 2003 Homeland Security Presidential Directive HSPD-7 for Critical Infrastructure Identification, Prioritization and Protection; and the 2013 Presidential Policy Directive PPD-21, identify Transportation Systems as a Critical Infrastructure. Intelligent Transportation Systems (ITS) are utilizing technology to increase traffic safety and environmental benefits. For example, according to the United States Department of Transportation (USDOT) ITS reduce traffic hazards, which cause about 43,000 deaths, 3 million injuries and consume over $230 billion dollars each year [1]. An ITS is defined according to the USDOT, Research and Innovative Technology Administration (RITA), as "the application of information technology to surface transportation in order to achieve enhanced safety and mobility while reducing the environmental impact of transportation". The ITS program was initialized and created by the U.S Congress as a national program to incorporate technology and advanced systems into the transportation infrastructure, e.g., to increase traffic safety and decrease pollution and fuel consumption. It was administered by the Department of Transportation in the Intermodal Surface Transportation Efficiency Act of 1991 (ISTEA) which originally named ITS as Intelligent Vehicle Highway Systems (IVHS) [1].

The ITS provides communication support to moving and stationary devices. The moving devices are associated with vehicles, while the stationary devices are associated with the roadside, i.e., the infrastructure. Thus it relates to Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Infrastructure-to-Vehicle (I2V) and Infrastructure-to-Infrastructure (I2I) communications.

1.2 Motivation

Given that the ITS is a critical infrastructure, that consists of safety critical applications, and that any fault, may it be of benign or malicious nature, could have far-reaching consequences, security and survivability are of paramount importance.

Real-time control systems, especially those governing critical infrastructures, such as transportation systems presented in this research, need to be reliable and secure under normal operating conditions and survivable under abnormal conditions. They should be designed and operated so that essential services will function even in the presence of component failure or external or internal manipulations of the system or the data it relies on.

1.3 Objectives and Research Contributions

In this dissertation, we present survivability solutions for two types of ITS application domains. The first domain involves the Connected Vehicles Infrastructure (CVI), and will be shown in Chapter 3. The second domain is the Weather Responsive System Infrastructure (WRSI), which will be presented in Chapter 4. Both application domains have in common that they are safety critical infrastructures, and thus any failure can lead to injury or loss of life. Given the criticality of the systems, fault-tolerance and survivability considerations have to be designed into the systems, rather than in an add-on fashion, therefore in our proposed solutions, we will demonstrate survivability mechanisms that employ an approach known as *Design for Survivability* [2]. The solutions presented to both application domains do not require any modification of the existing infrastructure components or standards.

Contributions of this dissertation follow the principle of design for survivability and can be stated for both application domains, which have in common that malicious faults could render the applications useless. The specific contributions are as follows:

Connected Vehicles

1. A model to analyze and quantify the reliability of Dedicated Short Range Communication (DSRC) safety applications is introduced.

2. An approach is given to utilize channel redundancy to mitigate against the impact of communication jamming.

3. In addition of channel redundancy, information redundancy is achieved by using message type dissimilarity.

4. The survivability mechanisms avoid the need for any modifications or deviation from the standards.

5. The approaches are analyzed and the results show their effectiveness in improving survivability of the safety applications.

Weather Response System

1. The main theoretical contribution is the combination of the approaches introduced in [3, 4, 5] into one comprehensive architecture with survivability and resilience characteristics.

2. The subsystem that monitors the application program is extended to three monitoring approaches: 1) detection of dependency violations, 2) identification of anomalies through exception triggers and data sensor analysis, and 3) detection of off-nominal, non-certified executions.

3. The theory is extended to use a *dual-bounded* certification threshold vector.

4. The theory of detection of off-nominal executions is extended to allow for certification of executions based on *Behavior Sets*.

Differences and Commonalities

The CVI solution addresses V2V, V2I, and I2V communications. In this solution, the main concern is value faults introduced during message exchange under malicious attacks. Specifically, we are concerned about malicious symmetric and asymmetric fault types, which will be discussed in detail in Chapter 2, Section 2.4.

The fault assumptions in the WRSI solution are that any fault type can be detected with high probability and the system can go to fail safe, therefore we can treat any type of faults as benign fault. The WRSI solution addresses fixed installed infrastructure, i.e I2I communication. Fault Types and assumptions will be discussed in detail in Chapter 2, Section 2.4.

It should be noted that in both application domains survivability and safety are very related. Furthermore, failures that would result in values or actions that would under normal circumstances result in compromise of safety, will in these cases have limited impact. This is due to the fact that in WRSI the devices are National Transportation Communications for Intelligent Transportation System Protocol (NTCIP) compliant, thus not allowing deviations outside of the safety parameter ranges, and in CVI the driver is still in charge (unlike in cases like active braking).

The reliability for the CVI and WRSI will be defined in Chapter 3 and Chapter 4 in detail. The CVI application reliability depends on the packet error probability and their impact on message exchanges. The WRSI application reliability is the probability that the application experiences nominal behavior, and if this is not the case then the application contingency management systems provides the proper response, e.g., go to fail-safe mode. Table 1.1 shows a summary of information related to the two proposed solutions.

1.4 Outline

The rest of the dissertation is organized as follows: Security and survivability of critical infrastructures with description about different fault types and assumptions will be introduced in Chapter 2. In Chapter 3 we will introduce the first application domain, which involves the Connected Vehicles Infrastructure. The second application domain, which is the Weather Responsive System Infrastructure, will be presented in Chapter 4. Finally, Chapter 5 discusses conclusions and future work.

	CVI	WRSI
Fault Types Considered	Malicious: asymmetric and symmetric (omissions)	All fault types are considered
Redundancy	Spacial	Time
Infrastructure Communication Support	V2V, V2I, and I2V	I2I
Technical Modification on existing standards	none	none
System Operation	1) Receive messages from different vehicles 2) Alerts the driver in a timely manner	1) Receive weather data from sources 2) The algorithm changes signal phase yellow timing
Potential Impact	The driver is still in charge (no active braking)	NTCIP compliance (adjustments stay within compliant ranges)
Limitations	System can work up to certain assumptions of jamming, then fail-safe mode	System can tolerate faults up to certain threshold then fail-safe mode

Table 1.1: Differences and Commonalities

Chapter 2

SURVIVABILITY IN CRITICAL INFRASTRUCTURES

2.1 Motivation of Survivability

Given the ITS's communication nature, may it be vehicle-to-vehicle or between vehicles and the fixed infrastructure, communication inherits the entire spectrum of potential threats.

Furthermore the attack vector cannot be fully predicted. For example, targeted jamming has been shown to be able to introduce Byzantine faults in wireless networks [6] and the safety applications of the ITS are not immune to such attacks either.

Given that the ITS is a critical infrastructure, then the degree to which ITS is able to provide critical services is broadly defined as survivability [7].

According to [7], survivability analysis was first introduced by [8] in the context of military command, control, and communication (C^3) systems in 1970s.

In order to understand the basic philosophy of this research it is important to understand the concept of *survivability* and *resilience*. There are no single agreed-upon definitions for system survivability or resilience. Instead, one may use as a starting point the vague notion that a system has to be able to tolerate diverse faults. This includes those faults typically considered in the area of fault-tolerant system design, such as faults resulting from component failure as a consequence of aging, fatigue, or break-down of materials. These faults may exhibit very predictable behavior and frequency. Software faults are more difficult to describe, however, they essentially cause the system to enter a state that deviates from the specification. In the last two decades there has been much attention on (humanly induced) malicious faults, e.g., hacking, denial of service, virus, Trojan horses, and spoofing. These kinds of faults may affect the software and even the hardware and can be totally unpredictable. They are the main target of security and survivability considerations.

Security addresses the standard concerns associated with confidentiality, integrity, and authentication, and often includes access control, nonrepudiation, availability, and privacy [9]. According to [10], traditional security mechanisms have limitation in maintaining and recovering services during and after an intrusion, i.e., the system is either safe or

compromised. Survivability on the other hand refers to the robustness under attack, which presents the main difference of survivability from the traditional security mechanisms, i.e., survivable system components must accomplish their mission even if the attack damages significant portion of the system [10].

The goal of fault tolerance to enable the services of a system to operate even in the presence of faults. Fault tolerance relates to the statistical probability of accidental faults or combination of faults, and typically does not include the malicious attack or intelligent adversary. Approaches to relax this assumption were presented in [11] where malicious act is considered as part of the fault spectrum, affecting dependability and security. Certain fault scenarios may be considered to be so unlikely to occur, that they may not be addressed, if the probability is below a certain threshold. Survivability on the other hand does not strictly rely on these statistical assumptions as intelligent adversaries may use pathological scenarios. These statistical assumptions can be extended to independence of faults. For example, in typical systems fault tolerance considerations may assume independence of faults if the probability of a fault in one system component affecting another is considered small. This may not hold in the case of pathological malicious behavior.

From what we stated above and according to [10], it is obvious that survivability can benefit from security and fault tolerance, and it provides a framework for both, security and fault tolerance, with other disciplines that can contribute to system survivability.

Survivability often takes a more mission-oriented view, in that the mission must survive, i.e., essential functionalities must perform to specification even in the presence of faults or malicious act [2]. This implies that the system needs to be designed with survivability considerations in mind.

2.2 Definitions of Survivability

Many definitions of survivability have been proposed, as summarized in [2, 7], which can be loosely partitioned into *qualitative* and *quantitative* definitions.

Qualitative definitions mainly focus on guaranteeing that essential functionalities of the system are maintained, in a timely manner, even in the presence of faults and attacks; “the

mission must survive".

According to [10], survivability is defined as "the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents. We use the term system in the broadest possible sense, including networks and large-scale systems of systems."

Quantitative definitions imply that survivability can be quantified, e.g., measured, and assume a formal model. Liu and Trivedi [7] proposed a general survivability quantification framework to a wide range of system architectures, applications, failure/recovery behaviors, and desired metrics, which is capable to better understand system steady state and transient behaviors under failures/attacks. They captured the definition of the T1A1.2 network survivability performance working group [12], in which by this definition, survivability depicts the time-varying behavior of the system after a failure occurs. The definition states: "Suppose a measure of interest M has the value m_0 just before a failure happens. The survivability behavior can be depicted by the following attributes: m_a is the value of M just after the failure occurs, m_u is the maximum difference between the value of M and m_a after the failure, m_r is the restored value of M after some time t_r, and t_R is the time for the system to restore the value m_0."

Liu and Trivedi [7] applied this definition on a telecommunication switching system consisting of n trunks (or channels). Their [7] definition of survivability is shown in Figure 2.1.

For a closer look at survivability, its definitions and implications, the interested reader is referred to [2]. The work presented here incorporates both qualitative and quantitative aspects of survivability, using the definitions of [10] and [7] respectively.

There are many definitions that address dependability and resilience in one from or another. Some are mathematically very precise, such as *reliability* $R(t)$, which is defined to be the probability that the system performs to its specification during the entire interval $[0, t]$, assuming that it was functioning at $t = 0$. The definitions of *survivability*, *resilience*, or *intrusion tolerance* are less precise and have been subject of much discussion in the dependability and security community. From our application point of view we care less about the specific definitions as about the general implications. Thus we focus on the overall

concept of being able to deal with benign or maliciously induced faults by adapting the system in a manner that 1) provides continues essential functionalities, 2) adapts to unexpected changes of system input or external events, 3) provides graceful degradation, and 4) satisfies security and safety requirements under diverse fault assumptions. These fault assumptions will be presented in Section 2.4.

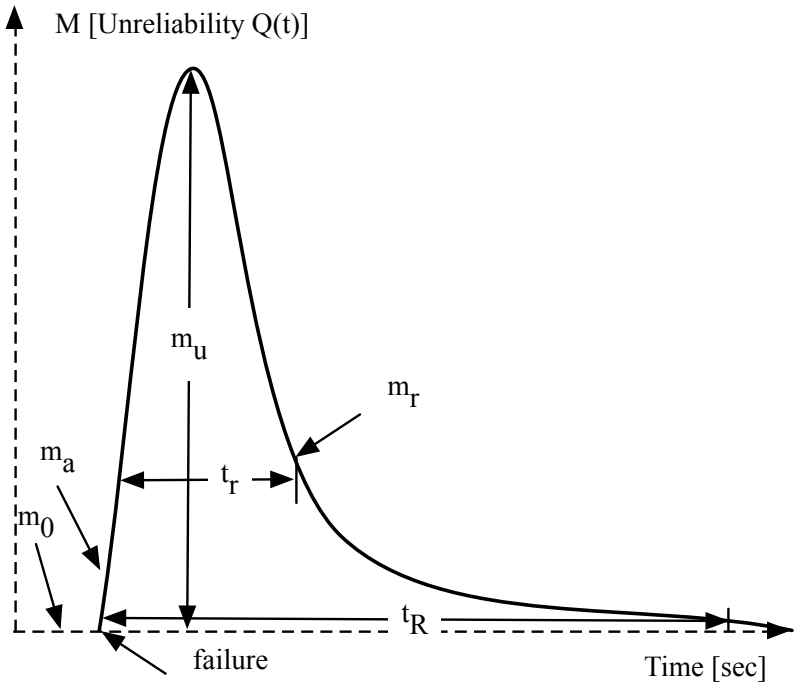

Figure 2.1: Quantitative Definition of Survivability [7]

2.3 Preserving Essential Functions in Survivable Systems

As the common thread in the definitions of survivability of the last section is the concept of essential functionalities of a system,i.e., “the mission must survive” [10].

In [10], they used the term mission to refer to the requirements or goals in order to preserve the main objectives of the system, also whether or not the mission has been accomplished is judged through the context of external conditions that may affect the achievement of that mission. For example, in the transportation system if the signal

controller, due to unexpectedly minor failure, were to cause unsafe signal timing, which would lead to a hazard in an intersection, then the system would be judged to fail its mission.

In the survivability definition, they also used the expression “in a timely manner” to address criticality of time defined in the mission. For example, in the Connected Vehicles Infrastructure, which will be shown in Chapter 3, we will show that the mission should be survivable during specific time periods. If the driver doesn’t receive the alert message in a specific time period, then the mission will fail.

In [10], they used the terms attack, failure, and accident to include all potentially damaging events. In survivability, the main focus is on the effect of a potentially damaging event. For a system to survive, it must react to, and recover from the damaging effect, perhaps even before the cause of the damage is identified.

In the Connected Vehicles Infrastructure solution, the main concern is faults introduced during message exchange under malicious attacks. Thus, we should be concerned about faults ranging from benign all the way to the worst case malicious fault types. The fault assumptions in the Weather Responsive System Infrastructure solution are that any fault type can be detected with high probability and the system can go to fail safe, therefore we can treat any type of faults as benign fault.

It should be noted that in order for a system to survive and fulfill its mission, the essential functions of the system that defines its mission need to survive, and not any particular subsystem or system component. Even if significant portions of the system are damaged or destroyed, a survivable system can fulfill its mission against damaging events such as attacks, failures, or accidents.

In the Connected Vehicles Infrastructure, the essential function to fulfill the mission is to alert the driver in a timely manner of the status of other vehicles under malicious attacks. In Weather Responsive System Infrastructure, the essential function is for the application to experiences nominal behavior, and if this is not the case due to any type of faults, then the application’s contingency management system provides the proper response, e.g., to enter a fail-safe mode.

2.4 Fault Models

At the core of addressing survivability is the capability of dealing with diverse faults. There are too many fault sources to list them individually and exhaustively. Therefore the notion of *fault models* is used, capturing the behavior of a fault, i.e., a *fault* can produce an *error*, that in turn can lead to a *failure* [13, 14]. The latter implies that the system does not perform its tasks to specifications anymore.

The diversity of faults and their consequences for a system have been the primary motivation for the definition of fault models. A fault model addresses the behavior of the faults and specifies the redundancy levels required to tolerate a single fault type or a mix of fault types.

One of the fundamental problems in the fault tolerant distributed computing is the ability of non-faulty processes to reach agreement on data values in the presence of faulty processes. The transmitting processor in these systems is required to transmit data to all other processors. If the transmitter is a faulty processor, then it may send conflicting values to other processes. This issue can cause disagreement among receivers of the sent message. This situation can be due to a faulty transmitter or can be due to a fault in the communication link, which will result in inconsistency or disagreement among non-faulty processors [15].

Many different fault models have been proposed over the years, ranging from the simple models that make no assumptions about the fault behavior [15, 16], to hybrid fault models considering multiple fault behavior [17, 18, 19]. The latter considers a mix of faults ranging from benign, symmetric to asymmetric faults [18], with potential transmissive and omissive behaviors [19].

In 1982, Lamport et al [16] introduced a distributed agreement called *Byzantine Agreement*. In this exact agreement a single-mode fault model was used, in which all non-faulty processes must arrive at a single consensus value. The total number of processors must be more than three times the maximum number of faults i.e., $N \geq 3t + 1$, where N is the total number of processors and t is the maximum number of faulty processors [15, 16]. The Byzantine Agreement algorithms of [16] assumes every fault to be worse case. This makes the algorithm very expensive in terms of the number of processors required and the

number of messages exchanged in each round of communication. The algorithm requires $t + 1$ rounds of message exchanges.

Meyer and Pradhan [17] partitioned the single-mode fault, introduced in *Byzantine Agreement*, into two modes. The first fault mode assumes *benign* faults, which are globally diagnosable and have no effect on other components. The second fault mode assumes *malicious* faults, i.e., Byzantine faults. The total number of faults at any given time is the sum of the total number of benign faults and the total number of malicious faults, i.e., $N > 3m + b$, and $r > m$ rounds, where N is the total number of processors, m is the total number of malicious faults, and b is the total number of benign faults.

Thambidurai and Park [18] partitioned *malicious* faults into *symmetric* and *asymmetric* faults. In symmetric faults the faulty value is identically received by all non-faulty processors. Thus the same faulty value is transmitted to all receivers. On the other hand, asymmetric faults are Byzantine faults, and therefore there are no restrictions on the fault behavior. Thus different values may be received by different non-faulty processors. The total number of faults at any given time is the sum of the total number of benign, symmetric, and asymmetric faults, i.e., $N > 2a + 2s + b + r$, and message paths $r \geq a$, where N is the total number of processors, a is the number of asymmetric faults, s is the number of symmetric faults, b is the number of benign faults, and r is the number of rounds of message exchange excluding initial transmission. Symmetric and asymmetric faults are often also termed *value* faults, as they imply incorrect values. This could also be the result of omissions. It should be noted that there is a follow-up paper [20] by Lincoln and Rushby that addresses a problem in the algorithm introduced in [18].

In 2000, Azadmanesh and Kieckhafer [19] developed a fault model that partitions Thambidurai and Park's [18] symmetric and asymmetric fault modes into *omissive* and *transmissive* versions. The symmetric faults of [18] are partitioned into *omissive symmetric* and *transmissive symmetric* faults. An omissive symmetric fault is the failure to deliver values to *all* non-faulty processors, i.e., no processor receives a value. In transmissive symmetric faults the same erroneous values is received by *all* non-faulty processors. The asymmetric faults of [18] are partitioned into *strictly omissive asymmetric* and *transmissive asymmetric* faults. A strictly omissive asymmetric faults results if some processors receive

the correct values and one or more others do not receive a value. It should be noted that no erroneous value is sent. However, the processors that have not received values will use a default value that is different from the sent value. A transmissive asymmetric is the classic Byzantine fault, i.e., under this fault scenario different values may be received by receivers.

The five-fault model of [19] constitutes the basis for the research presented. The evolution of hybrid fault models is shown in Figure 2.2.

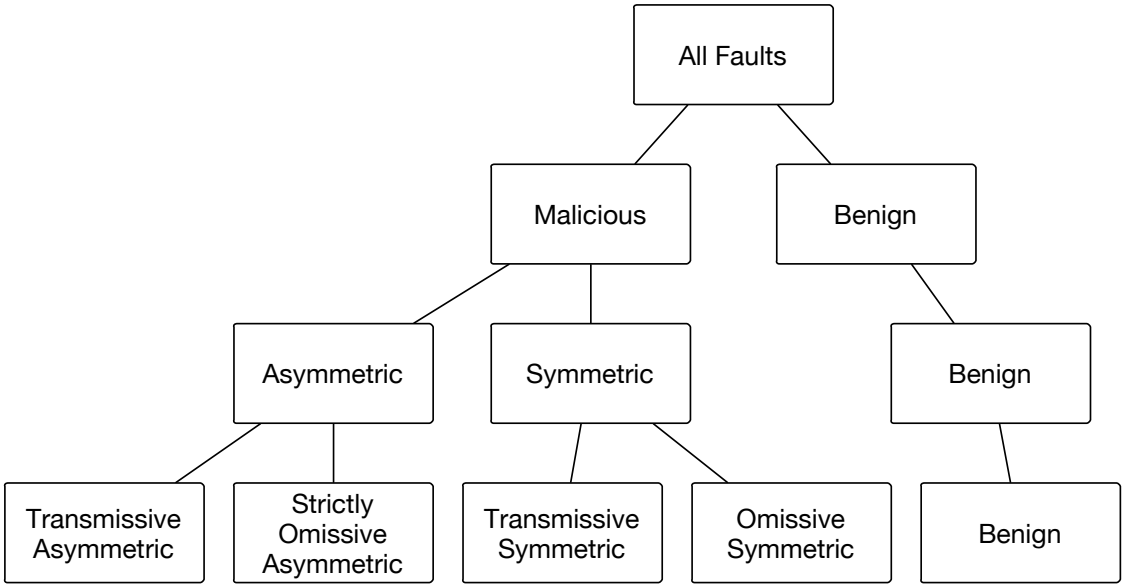

Figure 2.2: Hybrid Fault Models [19]

2.5 Fault Tolerance through Redundancy and Dissimilarity

It should be noted that in general the redundancy described in the context of fault models above is referred to as *spatial* redundancy. Other kinds of redundancy are *time* and *information* redundancy. The different types of redundancy were defined [14] as: "the addition of information, resources, or time beyond what is needed for normal system operation." The three types of redundancy are [21]:

Spatial Redundancy is to obtain information from multiple different sources. This type of redundancy is used to improve the reliability of data exchange and to increase the level

of information security. There are two types of spatial redundancy, the first is *hardware* redundancy and the second is *analytical* redundancy.

Time Redundancy is to perform a specific action more than once, separated in time. The repeated actions are checked and compared to increase reliability. In time redundancy related to communication, the same packet is sent more than once, at different times.

Information Redundancy is the addition of redundant data beyond the required data to implement a given function. A typical example of information redundancy is error correction codes.

According to [11], the fault assumption highly affect the choice and implementation of error detection, error handling and fault handling techniques. Since fault assumption affect the selected faults to tolerate, it relies on the independence of redundancies. Fault tolerance can be achieved by performing multiple computations through multiple channels, either sequentially or concurrently. These channels can be 1) of identical design to handle the hardware components that fail independently, or 2) the channels can be implemented to perform the same functions using different designs and implementations, i.e., through design diversity.

Since in safety critical applications, the system may be subject to a variety of faults, one should apply the concepts of redundancy. Redundancy however is only meaningful if failures of the individual redundant subsystems are independent, which is referred to as the *independence of fault* assumption. The lack of independence of faults is referred to as *common mode* faults [11]. In an attempt to provide independence of faults, *diversity* in the design of the system has been used under the assumption that diverse systems will be affected differently when subjected to the same faults.

Two different approaches have been used with the goal of achieving independence of faults through dissimilarity, i.e., *N-version* and *N-variant* designs.

The concept of N-version programming [22] dates back to the late 70s. In N-version programming it is assumed that several software development groups independently derive programs from the same specification. The concept of N-variant software is inspired by

N-version software, but in N-variant software different variants are generated in a more automated fashion [23, 24, 25, 26, 27, 28]. The goal of both, N-version and N-variant approaches is to ensure the survivability and resilience of the system under different fault types.

In [26, 27], an N-variant system framework was presented that used automated diversity to provide high assurance detection and disruption for large classes of attacks. They used diversity to make it impossible for an attacker to compromise all the variants with the same input.

In [28], a diversity index was introduced, based on ecological diversity studies to be able to quantify the value of data in terms of diversity. They used the similarities between internet malware and the organisms in an ecology system, to study the effect of the same attack on systems that lack data diversity. Specifically, it was stated that: "An ecological system is said to have high species diversity if many nearly equally abundant species are present. If a community has only a few species or if only a few species are very abundant, then species diversity is low". Therefore the diversity index can be used to measure the degree of heterogeneity. In [28], the mathematical measures to quantify diversity was summarized. First, they introduced the *Shannon index* [29] as one of the most popular diversity index used in ecological studies shown in Equation 2.1.

$$H_1 = \sum_{i \neq 1}^{s} P_i \log_2 P_i \tag{2.1}$$

where P_i is the relative proportion of the i^{th} specie in the collection and s is the count of species present in the collection.

Another popular diversity index is *Simpson's diversity index* [30], which measures the probability that two individuals randomly selected from a sample will belong the same species. Simpson's diversity index is shown in Equation 2.2.

$$H_2 = 2 \sum_{i>j=1}^{5} P_i P_j \tag{2.2}$$

where P_i and P_j are the relative proportion of the i^{th} and j^{th} specie.

Finally in [28], they stated that, Shannon [29] and Simpson [30] diversity indexes are the generalized form of the Renyi's entropy [31], which is shown in Equation 2.3.

$$H_2 = \frac{1}{1-\alpha} \log_2 \sum_{i=1}^{s} P_i^{\alpha} \tag{2.3}$$

where α provides different values to diversity.

In this dissertation, we will show how the selected safety critical application domains benefit from the redundancy and dissimilar implementations to enhance system survivability for different fault types.

In connected vehicles, generally the receiver needs to be able to receive messages from a transmitter in order to increase the application reliability under malicious attacks, e.g., in the presence of communication affected by jamming. In this application we are concerned about symmetric or asymmetric faults as a result of jamming. However, these two fault types can be the result of omissions. For example, if a vehicle brakes sharply and another vehicle following this lead vehicle has no visibility, e.g., due to fog, then a fault that results in the omission of the message indicating the brake event constitute an omissive symmetric fault. Similar examples can be argued for asymmetric faults resulting from omissions. We will discuss in more detail the communication system of the infrastructure and highlight its single point of failure, i.e., a single safety channel and a single message type. We concluded that redundancy is the best approach to overcome these fault modes. In this dissertation, the spacial redundancy is selected to tolerate theses kind of fault modes.

In the weather responsive system, yellow times are adjusted based on road condition data that is periodically accessed over the Internet. Any deviation from nominal behavior beyond a certain threshold will be considered a fault. Time redundancy will be the method to tolerate these faults.

Chapter 3

THE IMPACT OF DISSIMILARITY AND REDUNDANCY ON THE RELIABILITY OF DSRC SAFETY APPLICATIONS

3.1 Introduction

At the core of the ITS are safety applications, which require wireless communications, i.e., wireless signals. It should be obvious that the safety applications are directly affected by any degradation of communication reliability. Such degradation may be the result of adverse effects on the signals implementing communication, but it may also be the result of malicious act.

The mechanisms to increase survivability of ITS safety applications in CVI that will be presented in this chapter are based on data redundancy associated with applications using a specific kind of message, i.e., the Basic Safety Message (BSM) described below. The redundancy schemes are in line with the Vehicle Safety Communications - Applications (VSC-A) project [32] motivation, which considers data reliability to be essential for the robustness of the system.

This chapter describes how redundancy and dissimilarity can be used to mitigate effectively against jamming. Whereas the research in [33] assumed a homogeneous simplified channel power model, the research in [34, 35] is extended to consider the real impact of the inhomogeneous channels with dissimilar power ratings, as defined in standard American Society for Testing and Materials (ASTM) E2213-03 [36]. It will be shown how the redundancy scheme utilizes channels that have higher power ratings and how jamming, that would otherwise cause failure of the safety application, becomes largely ineffective. This is without introducing any mechanisms outside of the defined standards.

3.2 Contributions

A model to analyze and quantify the reliability of DSRC safety applications is introduced. An approach is given to utilize channel redundancy to mitigate against the impact of

communication jamming. In addition of channel redundancy, information redundancy is achieved by using message type dissimilarity. It should be noted that the survivability mechanisms avoid the need for any modifications or deviation from the standards. The approaches are analyzed and the results show their effectiveness in improving survivability of the safety applications.

3.3 Background and Related Work

Many ITS projects have been introduced worldwide, especially in the USA, Europe and Japan. Initially all projects were concerned with communication and service models, e.g., adopting known communication solutions such as 2G and Wireless Local Area Networks (WLAN), which led to the development of many standards like IEEE 802.11p and the IEEE 1609 standards family. Later most projects in real-world vehicular environments dealt with concepts and solutions optimized for interoperability between standards, performance of communications, and functionality of services [37]. This led to the adoption of 5.9 GHz DSRC over existing 900 MHz DSRC. To develop a national interoperable standard for 5.9 GHz DSRC, the Federal Highway Administration (FHWA) entered into cooperative agreement with the ASTM, leading to the publication of the ASTM E2213-03 standard [36] as approved standard for DSRC operations.

Channel allocation and the power characteristics are important to the concept of redundant communication for safety applications. The DSRC Wireless Access in Vehicular Environments (WAVE) system provides communication support to moving and stationary devices. In WAVE systems at least one of the engaged devices is associated with a vehicle, while the other may be any other WAVE device, e.g., another vehicle, roadside, or pedestrian. Thus it relates to V2V, V2I, and I2V communications. According to the ASTM E2213-03 standard [36] WAVE systems support many types of stationary or mobile devices. For stationary devices the WAVE standards define the Road Side Unit (RSU), which is permanently mounted. For mobile devices they define the On-Board Unit (OBU), which is mounted to a vehicle or any portable moving device [38].

The Federal Communication Commission (FCC) licensed 75 MHz of bandwidth at 5.9

GHz (5.850-5.925 GHz) to DSRC [1, 36]. It should be noted that Japan allocated 80 MHz (5.770-5.850 GHz) and Europe 50 MHz (5.875-5.925 GHz) with recommendation to add 20 MHz (5.855-5.875).

There are seven 10 MHz channels from (5.855-5.925 GHz), consisting of one Control Channel (CCH), i.e., channel 178 (denoted by CH178), and six Service Channels (SCH) with even numbers, i.e., CH172, CH174, CH176, CH180, CH182, and CH184. The remaining 5 MHz band (5.850-5.855 GHz) is reserved for future use. The first service channel, CH172, is a low power channel assigned to V2V communication, while the last channel, CH184, is a high power channel assigned to public safety applications, including road intersections [1, 36, 38]. Channels 174 and 176 can be combined to form CH175, and channels 180 and 182 could be combined to form CH181. Both channels, 175 and 181, are 20 MHz channels for higher data rate applications [1]. Table 3.1 shows a summary of information related to channel allocation.

Channel No	CH170	CH172	CH174	CH176	CH178	CH180	CH182	CH184
			CH175			CH181		
Channel Use	Reserved	SCH	SCH	SCH	CCH	SCH	SCH	SCH
Bitrate(Mbps)		3-27	3-27	3-27	3-27	3-27	3-27	3-27
			6-54			6-54		
Bandwidth(MHz)	5	10	10	10	10	10	10	10
			20			20		
Frequency Range(GHz)	5.850-5.855	5.855-5.865	5.865-5.875	5.875-5.885	5.885-5.895	5.895-5.905	5.905-5.915	5.915-5.925

Table 3.1: DSRC Channel Allocation

Since the power levels associated with different channels will play an important role in the assessment of survivability of our redundancy approach, the specific requirement in the standards need to be identified. The transmit power levels for public safety and Private RSU and OBU operations in DSRC channels were introduced in the ASTM E2213-03 standard [36]. It should be noted that the maximum allowable Effective Isotropic Radiated Power (EIRP) in accordance with FCC regulations is 44.8 dBm (30 Watt) for government, while the maximum allowable EIRP is 33 dBm (2 Watt) for non-government [39]. Table 3.2 shows a

summary of information related to channel power limits as introduced in the ASTM E2213-03 standard [36].

Max allowable Effective Isotropic Radiated Power (EIRP) 44.8 dBm (30 W) EIRP = P_t - L + G							
CH	Public / Private	Desc.	RSU		OBU		Min Gain
			Antenna i/p power dB_m	EIRB dB_m	Antenna i/p power dB_m	EIRB dB_m	
CH172	Public	Small and medium range operations	28.8	33	28.8	33	
	Private		28.8	33	28.8	33	
CH174	Public		28.8	33	28.8	33	
	Private		28.8	33	28.8	33	
CH175	Public		10	23	10	23	
	Private		10	23	10	23	
CH176	Public		28.8	33	28.8	33	
	Private		28.8	33	28.8	33	
CH178	Public	High Power	28.8	44.8	28.8	44.8	
	Private		28.8	33	28.8	33	
CH180	Public	Small zone operations	10	23			6
	Private		10	23	20	23	6
CH181	Public		10	23			6
	Private		10	23	20	23	6
CH182	Public		10	23			6
	Private		10	23	20	23	6
CH184	Public		28.8	40	28.8	40	
	Private		28.8	33	28.8	33	

Table 3.2: DSRC Channel Power Limits

Since we are only interested in the reliability of V2V safety applications, we will only present the maximum allowable power for public safety OBU operations and some RSU operations. Public Safety OBU operations in Channels CH172, CH174, and CH176 shall not exceed 28.8 dBm antenna input power and 33 dBm EIRP. Public Safety OBU operations in Channel CH175 shall not exceed 10 dBm antenna input power and 23 dBm EIRP. Public Safety OBU operations in Channel CH178 shall not exceed 28.8 dBm antenna input power and 44.8 dBm EIRP. Public Safety RSU and OBU operations in Channel CH184 shall not

exceed 28.8 dBm antenna input power and 40 dBm EIRP. The DSRC Channels CH180, CH181 and CH182 are used to implement small zone operations. Public Safety and Private RSU installation in these channels shall not exceed 10 dBm antenna input power and 23 dBm EIRP. OBU operations in Channels CH180, CH181 and CH182 shall not exceed 20 dBm antenna input power and 23 dBm EIRP. RSUs and OBUs shall transmit only the power needed to communicate over the distance required by the application being supported. Also it should be noted that, according to the ASTM E2213-03 standard [36], the receiver minimum input level sensitivity will be less than or equal to -85 dBm for 3 Mbit/s data rate, which is the lowest data rate for DSRC applications, and the sensitivity value varies according to the data rate used. The Packet Error Rate shall be less than 10% at a Physical Layer Service Data Unit length of 1000 bytes for rate-dependent input levels. Figure 3.1 shows a summary of information related to channels.

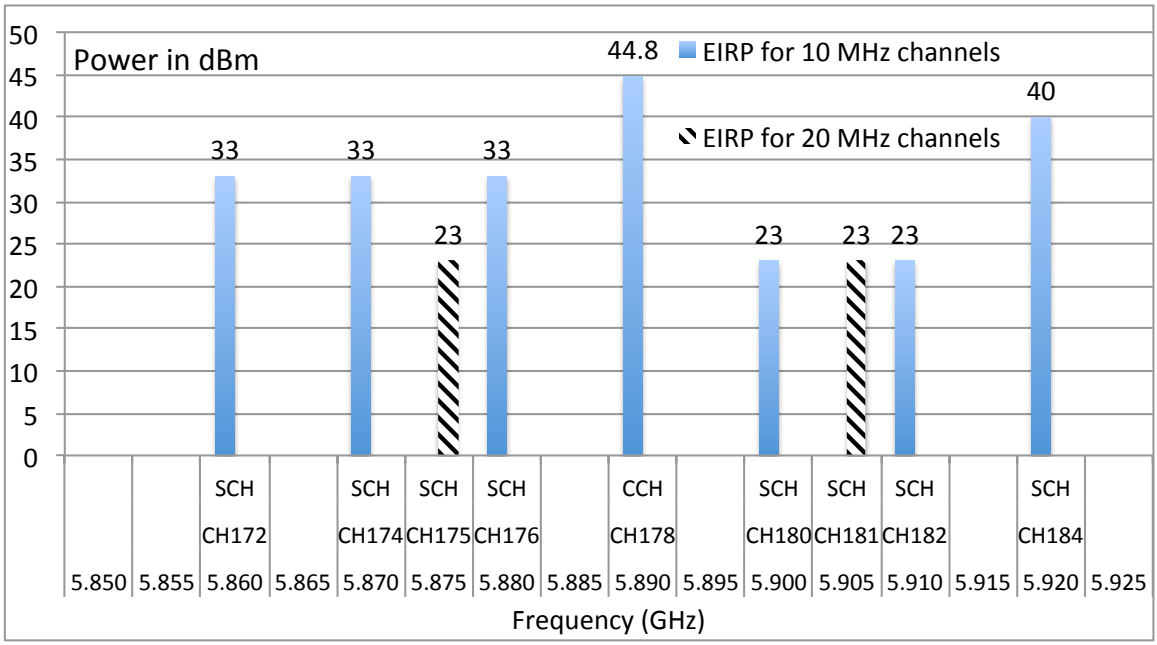

Figure 3.1: DSRC Channel Allocation and Power Limits

Testing communications related to vehicles was spearheaded by the VSC-A project [32]. The project was a collaborative effort in the area of WAVE safety applications initiated in December 2006 by USDOT and the Vehicle Safety Communications 2 Consortium (VSC 2 C), consisting of several vehicle manufactures (Ford, Mercedes-Benz, Toyota, Honda and

General motors). The VSC-A project final report was distributed by the USDOT National Highway Traffic Safety Administration (NHTSA), which provides information and results of testing V2V communication using DSRC at 5.9 GHz to improve the system and enable new communications-base safety applications. One of the most important goals in the VSC-A project was to develop and test BSM for V2V communication that can be used by safety applications to communicate in all directions of the host vehicle. It also proves the limitations of traditional safety systems such as radar.

There has been significant focus on the reliability of Vehicular ad hoc Networks (VANET). Research either focused on 1) applications with mechanisms utilizing the BSM messages, or 2) applications that use new messages to increase the functionality of BSM messages.

As an example of the first kind, redundancy was utilized in [40], where a non-interactive voting algorithm performed by the vehicle was introduced to detect malicious behavior. The algorithm depends on BSM message broadcasts from other vehicle's reaction to an event to infer on the truth in that event. A different redundancy approach was taken in [41], where a data-centric misbehavior detection scheme is introduced. It is not based on voting, but on observation of the movement of vehicles in response to their reaction to the event, such as a crash. However, both previous approaches will be affected by corruption or omission of the BSM messages they depend on.

As an example of the second kind, a collaborative protocol introducing a new message was used in [42] to deal with communication interruptions by moving obstacles as an effort to forward BSM messages. Such scenario can occur if a large vehicle blocks line-of-sight between two communicating vehicles. The blocking vehicle is made part of the message-forwarding scheme. In [43] a new message was introduced to disseminate data to other vehicles more efficiently. This message is involved in a grouping scheme based on roads. Communication between vehicles involves selected relay nodes with best line-of-sight within each group.

As it is not possible to give a comprehensive overview of all related work in general, we only gave representative examples. However, to the best of our knowledge, there is no research to date that uses redundant messages from the standard alone to overcome reliability

issues or malicious act. We will show an approach that uses BSM messages together with redundant messages from the existing standards to overcome BSM reliability issues.

3.4 WAVE Standards

Since the focus of this research is the investigation of survivability mechanisms based solely on existing standards, it is necessary to present their relevant details. Many standards have been developed to support the 5.9 GHz DSRC short to medium range communication for ITS Applications. Several ITS standards that support the WAVE architecture's different layers have already been published. Their most important aspects related to this research are discussed below and illustrated in Figure 3.2.

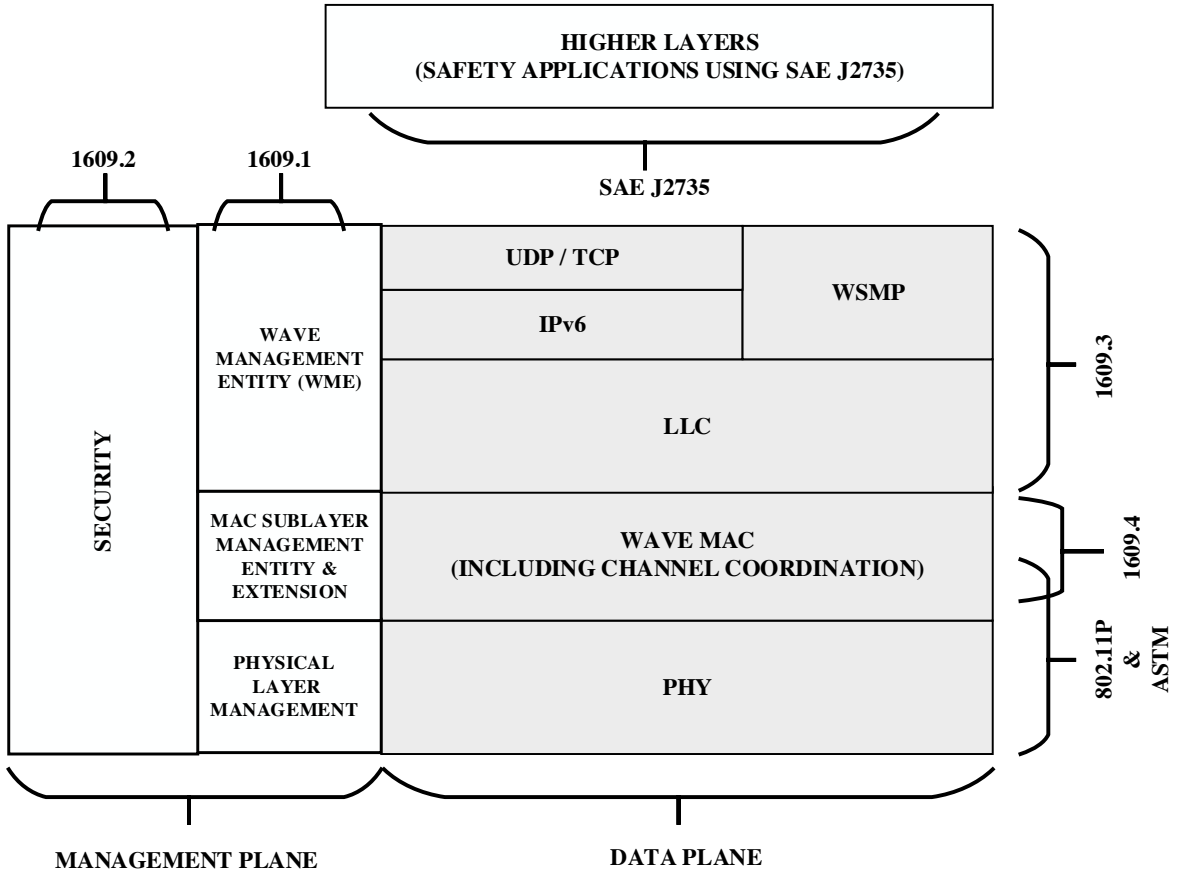

Figure 3.2: DSRC Protocol Architecture Related to WAVE Standards

3.4.1 ASTM E2213-03 Standard

The ASTM E2213-03 standard [36] describes the specification of the Medium Access Control (MAC) Layer and Physical (PHY) Layer using the DSRC services to be used in wireless communications. It is used in high-speed vehicle environments up to 200 Km/h and over short distances up to 1000 meters with very low latency and is based on the IEEE 802.11 and IEEE802.11a in the 5.9 GHz band. The standard supports a special implementation for the physical layer as introduced by IEEE 802.11a, and it uses the MAC layer of IEEE 802.11. The changes to the physical layer of IEEE 802.11a is that the Orthogonal Frequency Division Multiplexing (OFDM) will provide DSRC with data payload communication capabilities of 3, 4, 5, 6, 9, 12, 18, 24 and 27 Mbit/s, and in channel combinations it will be able to support 6, 9, 12, 18, 24, 36, 48 and 54 Mbit/s. The Binary or Quadrature Phase Shift Keying (BPSK/QPSK) and 16-Quadrature Amplitude Modulation (QAM) are used by DRSC, which support the mandated data rates of 3Mbps, 6Mbps and 12Mbps. These rates will be subject of our investigations. Based on the ASTM E2213-03 standard, the IEEE 802.11 working group developed the IEEE 802.11p [39], which is an amendment to include the specifications discussed by ASTM E2213-03 standard to support WAVE systems.

3.4.2 IEEE 1609 Standard Family

For the upper layers, the IEEE 1609 Work Group published a list of standards for wireless communications in vehicular environments.

The IEEE 1609.0 Standard

IEEE 1609.0 [38] is a draft guide for WAVE, which describes the DSRC/WAVE architecture for the devices in a mobile vehicular environment, and it provides an overview of the system, its components, and operations. Also it is considered a guide to other 1609 standards. IEEE 1609.0 defines the WAVE Service Advertisement (WSA) in which the application provider advertises a service to WAVE devices. The WSA has all the required information like service channel, priority, or repetition rate. When a WAVE device receives this advertisement, it will check whether the advertised application is of interest.

The IEEE 1609.2 Standard

IEEE 1609.2 [9] focuses on WAVE security services for applications and management messages. Due to the critical nature of safety application using WAVE devices and the wireless nature of communication, this standard addresses the need for privacy of application user data. The standard introduces new customized security mechanisms, rather than using the existing internet security mechanisms. While the existing internet standards are designed for flexibility and extensibility, we need the new mechanisms to optimize bandwidth and real-time low latency processing. Broadcast applications, which do not use encryption, should not include any personal identifying information, e.g., license plate numbers. Non-broadcast applications however encrypt messages to protect privacy. The standard suggests that there must be a method, which permits all the devices and applications in WAVE to be known and trusted by the Certificate Authority (CA), and all certificates must be only used by authorized entities. All applications must be granted authorization before using the safety channel. Basic Safety Messages are secured using digital signatures. The standard states that to minimize overhead on a congested channel the BSM uses implicit certificates with fast verification based on Elliptic Curve Digital Signature Algorithm ECDSA-256. Also it is stated that on receiving a BSM, the data validity period is 5 seconds. Due to the short validity time the VSC-A team suggested using a 224-bit key over the 256-bit key, which requires 50 percent less processing. The VSC-A team argued that a 224-bit key is enough to prevent forgery by attackers not having valid certificates [32].

The IEEE 1609.3 Standard

IEEE 1609.3 [44] for WAVE networking services is concerned with connectivity between vehicles to vehicles, vehicles to roadside or between any WAVE devices. The standard focuses on 1) network and transport layer protocols and 2) services supporting multi-channel connectivity between WAVE devices, providing addressing and data delivery services within a WAVE system. It defines service requests from higher-level layers that are accepted by the WAVE Management Entity (WME), which provides access to SCHs causing the transceiver device to be tuned to a specific channel during channel intervals. The service can be

requested from a provider, user, CCH Service, management services, or Timing Advertisement (TA) service. The standard defines two roles for the devices involved. The first is a provider, which advertises its services by transmitting WSA. The second is a user who is interested in the WSA, thus accepting the application messages on the specified SCHs. The standard classifies the types of devices using the allocated WAVE channels to 1) single-physical layer device (not capable of simultaneous operation on multiple radio channels), 2) multi-physical devices (capable of simultaneous operation on multiple radio channels), and 3) switching devices, which have one single-physical layer device capable of switching between channels. IEEE 1609.3 defines two protocol stacks that will be used in the WAVE system. The first is the WAVE Short Message Protocol (WSMP), designed for optimized operations. The second is the Internet Protocol Version 6 (IPv6), which supports transport protocols such as User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). The WAVE Short Messages (WSM) can be used on any channel, while the IP traffic is only used on the service channels.

The IEEE 1609.4 Standard

IEEE 1609.4 [45] for WAVE multi-channel operations is concerned with the specification of multi-channel wireless connectivity supported by the MAC sublayer between WAVE devices. It also describes multi-channel operation channel routing and switching for different scenarios. The standard defines channel coordination where switching devices are concurrently alternating access on the CCH and SCH intervals for data exchange. The channel access includes many options such as 1) continuous access, which requires no coordination because it allow continuous access to one channel, 2) alternating access between SCH and CCH, which requires coordination, 3) immediate SCH access, which allows access to SCH without waiting for the next SCH interval, and 4) extended SCH access, which allows access to SCH without pauses for CCH access. The standard specifies synchronization (for the above access options) based on common time references to perform channel coordination. Devices without local time sources can acquire timing information from other WAVE devices.

3.4.3 The SAE J2735 DSRC Message Set Dictionary Standard

The Society of Automotive Engineers (SAE) standard J2735 [46] was introduced for message exchange in ITS applications. This standard specifies the message set, its frames, and data elements for use by applications in 5.9 GHz DSRC to support interoperability between WAVE devices. It uses a dense encoding of messages and the general design goal is to maximize the support for short broadcast style messages. In this research we will only define five of a total of fifteen messages shown in Table 3.3, which will be used in our proposed solutions.

	From Infrastructure	From Vehicle
To Infrastructure		ProbeVehicleData (PVD) SignalRequestMessage (SRM)
To Vehicle	MapData (MAP) NMEA_Corrections (NMEA) ProbeDataManagement (PDM) RoadSideAlert (RSA) SignalPhaseAndTiming (SPAT) SignalStatusMessage (SSM) TravelerInformationMessage (TIM)	À la Carte message (ACM) BasicSafetyMessage (BSM) CommonSafetyRequest (CSR) EmergencyVehicleAlert (EVA) IntersectionCollisionAvoidance (ICA) RTCM_Corrections (RTCM)

Table 3.3: The SAE Fifteen Message Communication Topology

The five messages used are listed below and will be defined in detail in Section 3.6:

- Message (MSG_A_la_Carte)
- Message (MSG_BasicSafetyMessage)
- Message (MSG_ProbeDataManagement)
- Message (MSG_ProbeVehicleData)
- Message (MSG_RoadSideAlert)

3.5 Safety Application Scenarios

Since the focus of this research is the reliability of DSRC safety applications, we selected the used scenarios in our research from real world applications, i.e., real-word scenarios listed by the VSC-A project. These scenarios have been tested and analyzed by the VSC-A project, which includes the vehicle manufacturers, and have led to the development of the safety applications [32]. The scenarios described involve a Host Vehicle (HV) and one or more Remote Vehicles (RV). Our interest is the status of the host vehicle as it is affected by the status of the remote vehicles. The applications associated with crash scenarios based on [32] are: Emergency Electronic Brake Lights (EEBL), Forward Collision Warning (FCW), Blind Spot Warning + Lane Change Warning (BSW+LCW), Do Not Pass Warning (DNPW), Intersection Movement Assist (IMA), and Control Loss Warning (CLW). The applications and associated crash scenarios sorted by composite crash rankings are illustrated in Table 3.4 and Table 3.5. The composite crash rankings were determined by taking the average of the crash rankings by frequency, cost, and functional years lost for each scenario [32]. Three of the scenarios are depicted in Figure 3.3.

3.5.1 Scenario 1: Lead Vehicle Stopped

The scenario shown in Figure 3.3a, uses the FCW application, which alerts the driver of the host vehicle of an impending rear-end collision with a remote vehicle traveling ahead in the

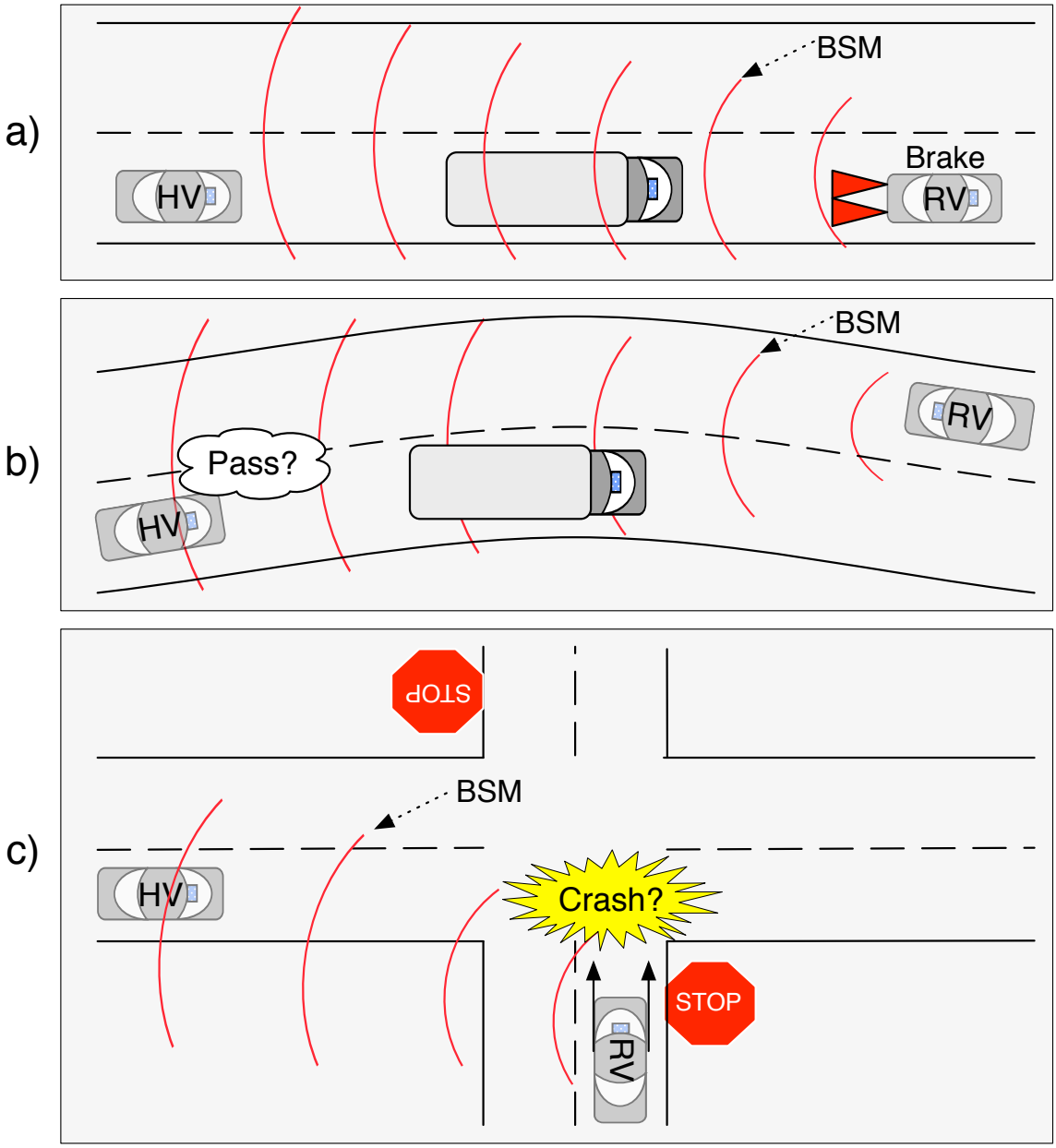

Figure 3.3: Selected Crash Scenarios Identified in [32]

No	Crash Scenarios	Crash Category		
		High Frequency	High Cost	High Years
1	Lead Vehicle Stopped	✓	✓	✓
2	Control Loss without Prior Vehicle Action	✓	✓	✓
3	Vehicle(s) Turning at Non-Signalized Junctions	✓	✓	
4	Straight Crossing Paths at Non-Signalized Junctions			✓
5	Lead Vehicle Decelerating	✓	✓	
6	Vehicle(s) Changing Lanes Same Direction	✓		
7	Vehicle(s) Making a Maneuver Opposite Direction			

Table 3.4: VSC-A Crash Scenarios Sorted by Composite Ranking [32]

same direction and on the same lane. For example, when a remote vehicle brakes hard, in the figure this is the first vehicle labeled RV, it broadcasts this event via a BSM message to the surrounding vehicles. The vehicles following the remote vehicle will use this information to alert the driver about a possible collision. This may be very useful in situations with low visibility, e.g., heavy fog or vision obstruction by large vehicles. The algorithm in the remote vehicle may transmit this event before the next scheduled transmission time with higher priority than routine BSM broadcasts.

3.5.2 Scenario 2: Vehicle(s) Making a Maneuver – Opposite Direction

This scenario uses the DNPW Application which alerts a host vehicle attempting a passing maneuver that is not safe. In Figure 3.3b the passing zone of HV is occupied by the RV traveling in the opposite direction.

3.5.3 Scenario 3: Straight Crossing Paths or Turning at Non-Signalized Junctions

Figure 3.3c shows crossing or turning at non-signalized junctions, which uses the IMA application. This application alerts the host vehicle that it is not safe to proceed due to high collision probability with a remote vehicle in the intersection. The host vehicle

No	Safety Applications / Crash Scenarios	EEBL	FCW	BSW	LCW	DNPW	IMA	CLW
1	Lead Vehicle Stopped		✔					
2	Control Loss without Prior Vehicle Action							✔
3	Vehicle(s) Turning at Non-Signalized Junctions						✔	
4	Straight Crossing Paths at Non-Signalized Junctions						✔	
5	Lead Vehicle Decelerating	✔	✔					
6	Vehicle(s) Changing Lanes Same Direction			✔	✔			
7	Vehicle(s) Making a Maneuver Opposite Direction					✔		

Table 3.5: VSC-A Safety Applications Related to Crash Scenarios [32]

communicates with all nearby remote vehicles and receives their broadcasted BSM. After that the in-vehicle unit analyzes all data received from other vehicles and predicts their future paths. If the analysis detects the probability of a collision, a warning is issued to the host vehicle's driver. Such warning is issued if the data in the BSM of the RV suggests to the HV that the RV is not stopping.

3.6 Redundancy-Based Survivability Architecture

The discussion above has the common thread that the BSM message is the main mechanism used by all safety applications. The BSM message is the main mechanism to communicate critical data used by all safety applications. This message is limited to one specific channel and thus represents a single point of failure. There are many ways this channel can be affected and possible faults may originate from simple obstacles, jamming, or the channel congestion phenomenon following a channel switch [44, 45], to name a few. To increase the message exchange reliability in the ITS safety applications, we propose an alternative, redundant approach. Specifically, first we propose message dissimilarity using other messages from the SAE J2735 standard [46] capable of providing the application with all required data as BSM. Second we propose channel redundancy by transmitting the proposed

messages on different channels, i.e., other than the BSM's safety channel. The alternate channels used for redundancy have higher power ratings than the safety channel. The use of redundant channels results in large reliability gains for safety applications in the presence of any failure in the safety channel such as the effect of jamming.

3.6.1 BSM and Message Dissimilarity

BSM is defined in SAE J2735 [46] and is a V2V message. This message is used by a variety of applications in an exchange of safety data regarding the vehicle state. The message is broadcasted by each vehicle to other surrounding vehicles at a rate of 10 times per second, or other rates depending on the application. The broadcast range of a BSM message is about 300 meters which depends on the transmitting power on the used channel.

A BSM message consists of two parts as shown in Table 3.6. Part I is mandatory and contains the most required fields for safety applications, including position (latitude, longitude, elevation and accuracy), motion (speed, heading, angle and acceleration), brake system status and vehicle size. Part II of the message is optional and is used when required by the application.

BSM				
Part I (Mandatory)	Position	Motion	Brake System Status	Vehicle Size
	Latitude Longitude Elevation Accuracy	Speed Heading Angle Acceleration		
Part II (Optional)	Vehicle Events Object	Vehicle Path History Object	Vehicle Path Prediction Object	Vehicle Relative Positioning RTCM 1002 Data Object

Table 3.6: The BSM Message

As defined by [46] BSM messages are transmitted on a pre-agreed channel, i.e., CH172, using the WSM protocol. It is not required for senders to advertise for this service, and also not required from the receiver to confirm or take any action to join this service. To facilitate

BSM functional redundancy, we need to identify messages that have the same structure and information to support safety applications. We identified two different suitable messages, i.e., À la Carte message (ACM) and Probe Vehicle Data (PVD) message, from the fifteen total messages defined in SAE J2735.

Redundancy Using ACM

The first message is the À la Carte Message, which is a V2V message shown in Table 3.7. As its name suggests, it can include any data frames, data elements, or any external content defined in the standard in a field called (ALLInclusive). All message fields can be added as required. For example, we can add the content of the BSM message, i.e., (BSMblob) [46], to get an ACM message containing equivalent information. The message has all the flexibility of the BSM and can even support more data than BSM if desired by an application.

ACM		
msgID	ALLInclusive	CRC

Table 3.7: The ACM Message

Redundancy Using PVD

The second message is Probe Vehicle Data. It is a V2I message, a unicast from the OBUs to an RSU using the WSM protocol on a Service Channel determined by the RSU. All PVD messages are authenticated and no acknowledgment from the RSU is required. A PVD message as shown in Table 3.8 contains information about the full position vector, vehicle type, and most importantly, it has a vector of snapshots, which define the vehicle's traveling behavior. Each snapshot contains 1) a full report of the vehicle position (longitude, latitude, elevation and accuracy), 2) the time in milliseconds, 3) its motion (speed, heading and transmission state), 4) the confidence information about time, position and speed, 5) the VehicleStatus field, which contains all the vehicle's sensor reading including the brake status, and 6) the VehicleSafetyExtension field, which includes path history, events, timing and path

prediction. In short, the PVD message contains a superset of the information found in the BSM message and is thus suitable for providing BSM data redundancy.

PVD				
Full Position Vector	Snapshots , sequence [1..32]			Vehicle Type
	Full Position Vector (Latitude, Longitude, Elevation, Heading,Speed, transmission, Accuracy)	Vehicle Status (all sensors)	Vehicle Safety Extension (Events, Path History, Path Prediction, RTCM)	

Table 3.8: The PVD Message

What specific information is to be included in the PVD message and which vehicle's message is relevant is controlled by a message named Probe Data Management Message (PDM). The PDM can add more privilege to the use of PVD by controlling data collected from the vehicles as follows. PDM is an I2V message broadcast from the RSU to OBUs. The PDM can 1) control the time/distance OBUs join the RSU and begin to send data using the SnapshotTime and SnapshotDistance fields, 2) control the coverage pattern using the direction HeadingSlice field, 3) instruct specific classes of OBUs to collect data from using the Sample field, and 4) indicate the frequency OBUs will send data using the TxInterval field.

3.6.2 Safety Channel and Channel Redundancy

As shown in the previous subsection, in terms of information content the ACM and PVD messages contain all the required fields to support the functionality of BSM in safety application. However, to eliminate the aforementioned single point of failure (BSM is limited to CH172) they should be on different channels. In [1] it was stated that "both public safety and non-public safety users should be eligible for licensing on all channels, subject to priority

for safety/public safety". This is confirmed also in [38], i.e., any of the control or service channels could be configured for use as a safety channel.

Given the flexibility of channel assignments mentioned above we suggest that the redundant channels should be far away in the frequency spectrum from the BSM safety channel to increase resilience against natural and malicious external interference such as shadowing or jamming. This separation assumption is proven by the VSC-A project. In validation of the DSRC PHY protocol with regards to Cross Channel Interference (CCI) the VSC-A project exposed in a field test that the interference in a band adjacent to the target band causes more performance degradation than similar interferer in a band further from the target band. The VSC-A team concluded that no change is needed in PHY protocol, and that CCI concerns should be addressed in higher layers [32]. This is in agreement with our approach, which resolves this redundancy issue in the application layer.

In order to use different channels in the redundancy scheme it is important to elaborate on the WAVE radio switching device to understand the details of channel accesses by WAVE devices, in order to make intelligent decisions about channel spacing and redundancy. According to [38, 45], in the channel switching based on time division multiplexing a single WAVE device is required to exchange information on a SCH while participating on the CCH. Switching devices was introduced in subsection 3.4.2. Access to channels is based on 100 ms periods, for CCH and SCH intervals. It is divided into 50 ms for each interval as shown in Figure 3.4. This however imposes significant capacity constraints on V2V safety

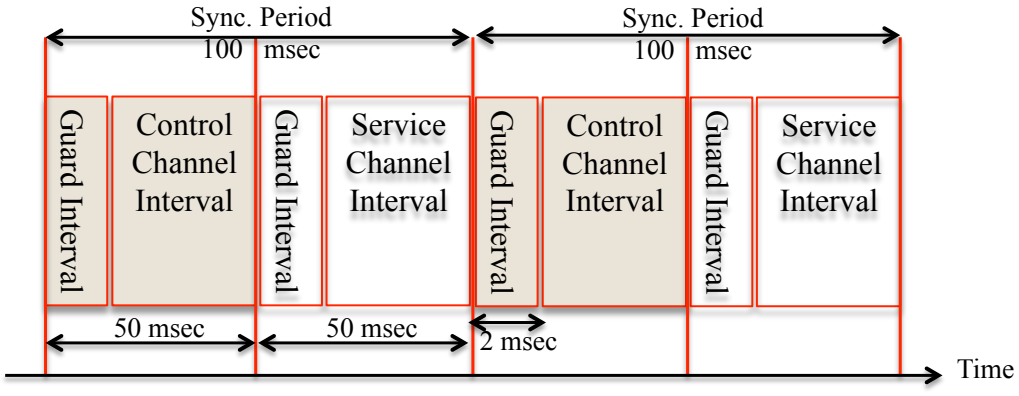

Figure 3.4: Channel Switching

communication, because the safety channel will be available less than half the time for safety messages. One of the goals of the VSC-A research was to avoid the capacity constraint by defining one dedicated channel for safety messages, i.e., an always-on safety channel, which according to [1] is CH172. Having a full-time access safety channel removes the need for channel switching and doubles the channel access time. However, the implementation of this concept requires that each OBU be equipped with two radios [32]. Therefore we assume using at least two WAVE radio devices per OBU for best performance. Dissimilar redundancy can be achieved by using the first device dedicated to CH172, the always-on safety channel, for exchanging BSM with full performance. The second device will be a switching radio device that exchanges information on any M other SCH while participating on CCH as shown in Figure 3.5. Below we will present solutions that implement redundancy for the special cases of $M = 2$ and 3, i.e., dual and triple channel redundancy. With a total of 6 service channels, in addition to the control channel, the maximum redundancy level is 7. However, it should be noted that the message overhead will grow linear with the number of redundant channels, imposing extra usage of the dedicated limited bandwidth.

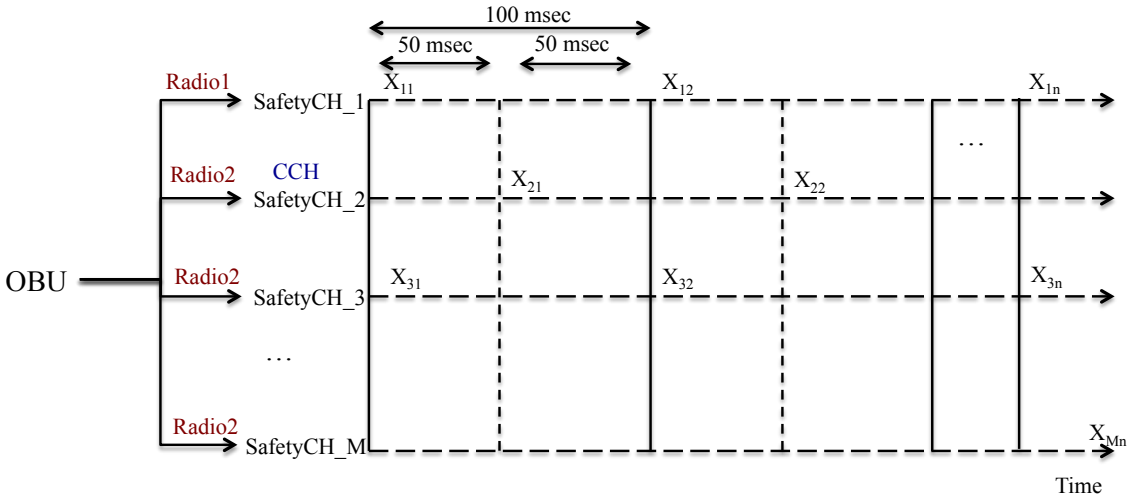

Figure 3.5: Channel Redundancy Using Dual Radio

Dual Redundant Channel Selection

There are two important factors that affect our selection to redundant channel, 1) the channel distance in the frequency spectrum, and 2) the maximum allowed channel transmitting power, shown in Figure 3.1. As stated in [45] any device listens to control channel CH178 by default. Furthermore, CH178 is optimally spaced from CH172 in terms of interference isolation. In addition, the EIRP of CH178 is higher than that of CH172, i.e., 44.8 dBm and 33 dBm respectively. Therefore CH178 lends itself as optimal candidate for the redundant channel as any other choice of channels would require additional switches of devices to monitor that channel. One way to manage access of CH178 for redundant messages in this scheme is to use the Wave Short Message Protocol Safety Supplement (WSMP-S) [45]. As requested by the VSC-A project [32] the WSMP-S will only be sent with WAVE short message that carry safety payloads to enable enhanced channel switching between vehicles and provide information to receiving vehicles about channel switching operations of the transmitting vehicles. This will maximize channel capacity by making choices concerning transmit channel and timing for safety messages, also it will avoid missing safety messages due to traditional channel switching. Therefore WSMP-S header can be used to arbitrate the control channel for safety messages. In our case these are the redundant counterparts to the BSM messages, which should take precedence over lower priority messages sent over control channel.

For the reasons described above, one candidate for a redundant analog to the BSM messages is the ACM, which is to be sent on the CCH with higher priority to take precedence over other messages. This implements a system with dual redundancy utilizing dissimilarity, i.e., two different messages on two different channels, to increase survivability of safety applications. Should there be a need to increase redundancy levels beyond two, e.g., as the result of conflicting values due to benign or malicious reasons, or out of concern that both mechanisms fail, a third redundancy level is required.

Triple Redundancy Involving the ITS Infrastructure

As shown in Figure 3.1, the most applicable choice for the third redundant channel is using CH184. The advantages of using CH184 is twofold. First it maximizes the spectrum separation to the other channels used in the redundancy scheme, which provides higher resilience to interference. Second, the EIRP of CH184 is higher than that of CH172, i.e., 40 dBm and 33 dBm respectively.

In the last subsection we introduced Dual Redundancy using ACM, which is a V2V message redundant to BSM on a different channel. Both messages used in dual redundancy are V2V involving message exchange between 2 vehicles. To make the system more resilient, diversity will be introduced as a third approach to involve the infrastructure. Involving the ITS infrastructure is not a new concept. For example, the RSU as an active actor has been recommended in the CICAS-V project [47] for signalized intersections in which the RSU alerts approaching vehicles of possible collisions.

The RSU can serve as a third mechanism in the redundancy scheme to communicate safety information. Specifically, the RSU can use the collected PVD messages and respond to the OBU in case of a detected hazard. In reference to the SAE J2735 there will be local systems that can be authorized to collect data directly from the RSU [46]. We recommend this system to be used for collision detection, which triggers a Road Side Alert (RSA) message to be broadcasted.

The RSA is an I2V message sent from the RSU to OBUs to alert travelers about nearby hazards. For urgent and critical messages the RSA is sent as periodic broadcasts using the WSM protocol on a high power channel, either CCH or SCH. In case of lower urgency the IP protocol can be used to send this message as a periodic broadcast over a service channel. This message can be embedded and used as a building block for any other DSRC message, e.g., it is used by Emergency Vehicle Alert message. The RSA has a FullPositionVector field, which describes the location of the hazard and whether it is fixed or moving. The message also contains the heading and priority. We can use the *ITIS.ITIScodes* fields to send alerts to vehicles if the infrastructure detects a hazard. For the implementation we suggest the use of the high power channel CH184 as discussed in the beginning of the subsection.

Implication of Triple Redundancy

To demonstrate this redundancy scheme a triple redundant application of the scenario in Figure 3.3c, i.e., the Straight Crossing Paths or Turning at Non-signalized Junctions, will be used. The motivation to use this scenario and not FCW is that now the RSU is involved, which is more likely situated in intersections. Consider the Intersection Movement Assist application used in the host vehicle and the scenario shown in Figure 3.6a.

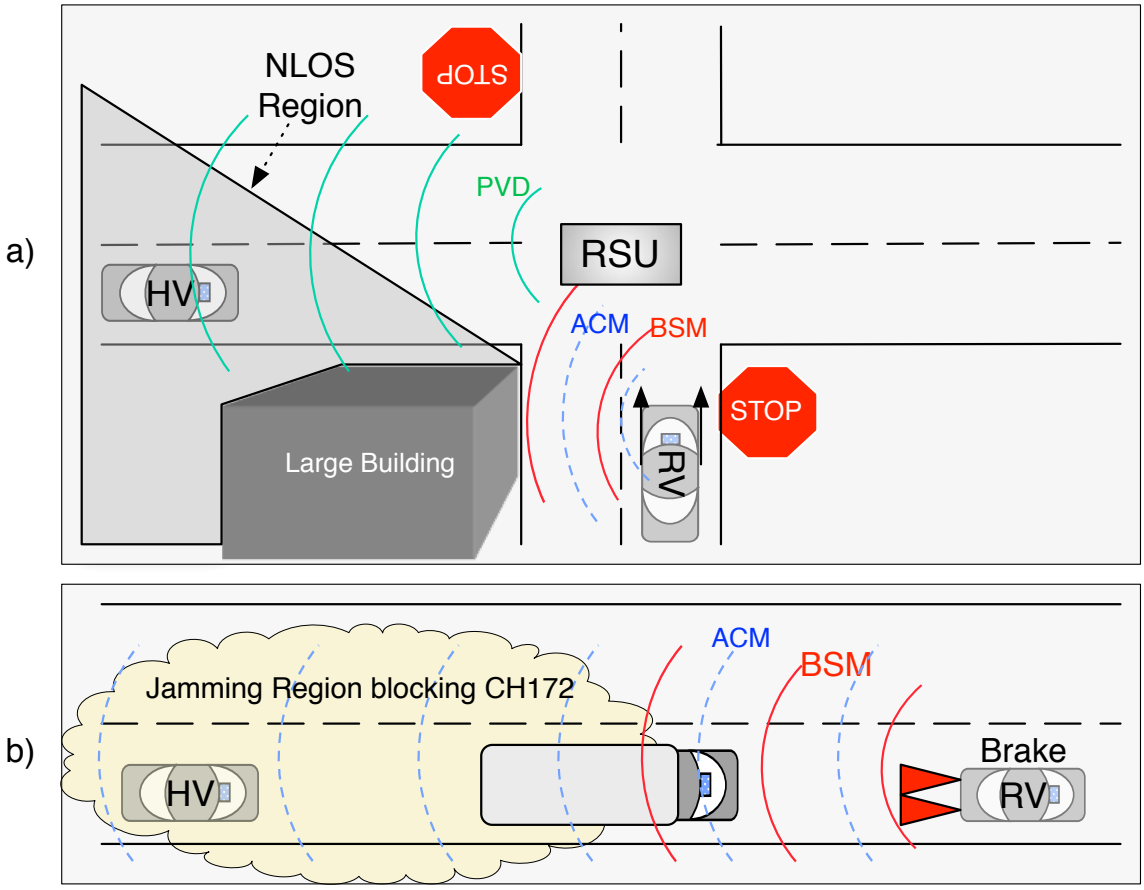

Figure 3.6: Demonstration of Triple Redundancy Mechanism

In the traditional scenario, which only uses BSM messages, the host vehicle would receive a BSM message from a remote vehicle crossing in its path. If the BSM message is blocked by an obstacle or the channel is jammed by an attacker, the host vehicle will not be aware of a possible impending collision. Using the redundant scheme the hazards condition will only occur if the BSM and all redundant message mechanisms fail or are compromised.

In Figure 3.6a the redundant schemes are provided using the ACM and the PVD involving the RSU.

The communication associated with FCW of Figure 3.3a is depicted in Figure 3.6b. Assume that channel CH172 is the target of a jamming attack. This will prevent the host vehicle from receiving BSM messages indicating that the remote vehicle is braking hard. Without redundancy HV cannot alert the driver. ACM is utilizing a different channel, i.e., CH178, and assuming that jamming does not reach the frequency spectrum of this channel the safety application will succeed.

The same arguments can be applied to Scenario 2 introduced in Section 3.5.2, in which vehicle(s) make passing maneuvers. The redundancy of the previous case applies and if an RSU is present triple redundancy can be used.

To determine the effectiveness of the redundant schemes one can lean on reliability analysis. If one describes the redundant system as a parallel system, which is defined to fail only if all redundant components fail, then the unreliability of the combined system is the product of the unreliabilities of the individual components [48]. Whereas this product rule only applies when using the assumptions of failures of electronic components, and not for non-exponential failure behavior, it still provides some intuition. A more precise model would need to consider more complicated hazard functions, as described in [49].

3.7 Wireless Communication and Jamming

Whereas communications are affected by many aspects of benign environmental phenomena, the adversarial model addressed in this research is malicious act. Much research has focussed on dealing with the environment issues related to signal degradation in DSRC communications. In this research we focus on the impact of jamming as a malicious act. Many different jammer types have been introduced and characterized in [50, 51], ranging from constant jammers, which constantly disrupt communication brute force, to intelligent jammers that are protocol-aware and able to target specific data or control packets.

Since DSRC is a wireless protocol, it inherits all problems from the shared wireless media, including malicious act such as Wireless Denial of Service (WDoS). A common

attack in wireless communication is jamming, which can be launched, using off-the-shelf equipment, to interfere or block legitimate transmission by emitting radio signals that do not obey the standardized MAC protocol.

A jammer is defined by [50] to be "an entity who is purposefully trying to interfere with the physical transmission and reception of wireless communications". Jamming cannot be avoided by regular security mechanisms such as authentication, digital certificates, or encryption, because the jammer is often disregarding higher layers, focusing on disrupting the physical communication at the lower layers. Several jamming types have been identified in [50, 51]. Our considerations focus on the following two types:

Constant Jammer: This type of jammer emits a constant radio signal interfering with legitimate communication, violating the underlying MAC protocol. This is considered the worst case of jammer by many researchers as it indiscriminately affect the signal of ongoing communication. However, it is the least energy efficient and is relatively easy to detect and locate.

Random Jammer: Here the attacker jams for t_j and sleeps for t_s seconds. The jam and sleep periods may be unpredictable, e.g., t_j and t_s can be samples of two random variables T_j and T_s, respectively, following different distributions [51]. Random jammers consume less energy than constant jammers, but can be harder to detect.

In this research, we investigate the safety application reliability as it is affected by constant and random jammers. We picked the constant jamming because it can create wide blind spots and induce immense performance degradation [52], also constant jammers are generally considered the worst case jammers in that their effect is indiscriminatory, even though they are easy to detect compared to more sophisticated jammers [50]. Random jamming was picked as its impact on reliability is limited, depending on sleeping period.

One important factor in jamming is the power that the adversary uses to disrupt. We assume that the jammer capabilities are limited to the technical specifications of the vehicles On-Board Unit, which is the device installed in vehicles and is readily available for purchase, i.e., its jamming effect is limited by the transmission power model of such devices as specified in the ASTM E2213-03 standard [36]. This is due to the fact that our current field implementation uses such device as a jammer.

3.8 Quantitative Analysis of Impact of Redundancy

Application reliability is highly dependent on the message exchanges and requirements of the specific application considered. For our research we selected the FCW application, as it is the highest ranked safety application based on crash frequency, cost and functional years lost according to the VSC-A project [32] as illustrated in Table 3.4 and Table 3.5..

3.8.1 Forward Collision Warning

FCW application was introduced in subsection 3.5.1 and shown in Figure 3.3a, which alerts the driver of the host vehicle of an impending rear-end collision with a remote vehicle traveling ahead in the same direction and on the same lane, where vehicles are traveling at constant speed.

The timing issues related to the FCW application host and remote vehicles of Figure 3.3a are shown in Figure 3.7. The position of the jammer in this scenario is assumed to be right next to the RV. A hypothetical situation would be an adversary with a jammer causing the event that leads to braking, e.g., by launching an obstacle into the moving traffic. Starting with the moment of hard braking at time t_{brake} the RV emits BSM messages every 100ms. The HV needs to be alerted of the potential collision with the RV early enough to react. The reaction time is the time from the driver receiving an alert to his/her reaction, i.e., the time from t_{react} to t_{brake}. Reaction is only possible if the HV receives at least one BSM message from the RV, which is the minimum the application requires to detect the event, before t_{react}. Specifically, as demonstrated using Figure 3.7, the HV must receive at least one of the first x BSM messages, i.e., BSM_1, ..., BSM_x, must be received before it is too late to react, at time t_{react}. Thus t_{react} is the deadline for the FCW application to warn the driver of a possible collision, leaving enough reaction time to brake. Any BSM message received after that will arrive too late for the driver to react. Typical reaction times are within 0.9s [53] and 1.3s [54].

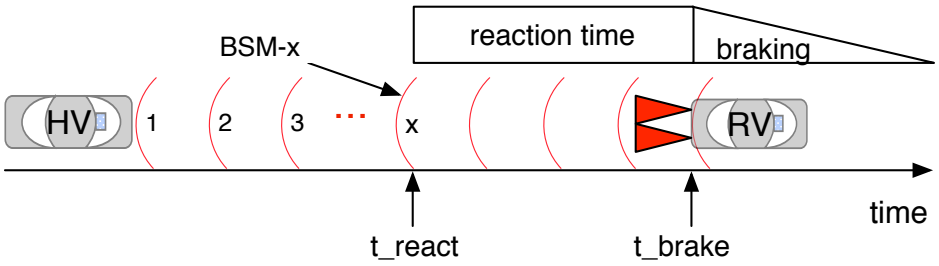

Figure 3.7: BSM Propagation during FCW

3.8.2 Impact of Jamming

Figure 3.8 shows FCW scenarios, where the host vehicle's reception of the BSM messages is affected by jamming, i.e., the jamming signal degrades the signal to noise ratio at the receiver of HV. This degradation however is related to the length of two distance vectors, i.e., the HV-to-jammer distance and the HV-to-RV distance. These distances change, as the vehicles are moving and the jammer is by our assumption stationary. We assume the distance between the HV and RV is constant, even during braking. This is over-conservative, but it accounts for special cases where brakes could be applied aggressively in conjunction with the gas pedal during brief periods.

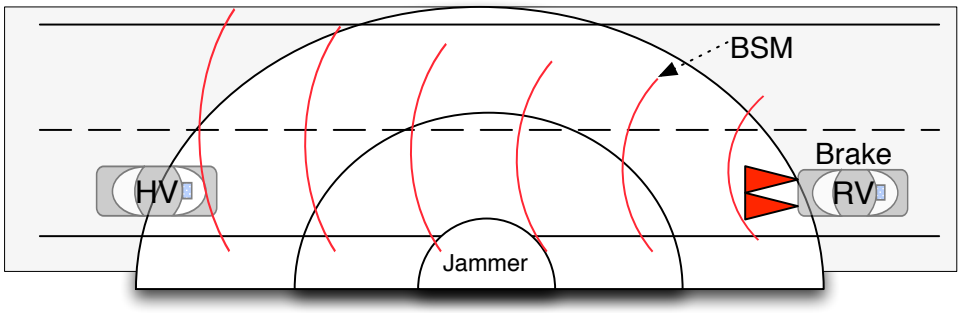

Figure 3.8: FCW under Jamming

Three interesting jamming scenarios are shown in Figure 3.9. Whereas the figure shows the timeline, it should be clear that these times relate to distances. In Figure 3.9a) the jammer is positioned right next to the RV as it brakes. As the HV approaches the jammer, the jamming effect on the reception increases. In Figure 3.9b) the jammer is positioned behind

the HV, and thus as the HV drives, the distance from the jammer gets larger. A larger distance between the HV and the jammer can also be the result of the jammer retreating further away from the road, as seen in Figure 3.9c). The distances between the HV and RV and where and how far from the HV the jammer is positioned has great impact on the application reliability.

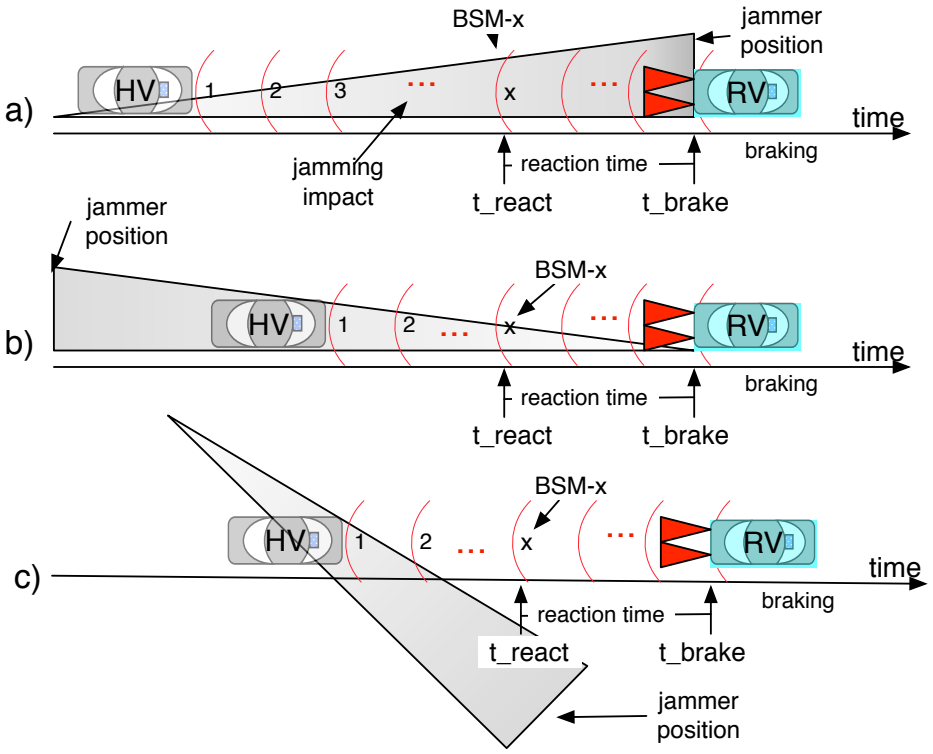

Figure 3.9: Jammer Positions

3.8.3 Application Reliability Quantification

The FCW application reliability is directly linked to the probability of the HV receiving BSM messages before it is too late to react. Thus the application reliability depends on the Packet Error Ratio (PER), or packet error probability and their impact on message exchanges. In line with the standard definition of reliability, i.e., $R(t)$ is the probability that the system is working to specifications during the entire time interval $[0, t]$ [14], we can define the FCW application reliability as the probability of receiving at least one BSM message before t_{react}, i.e., one of BSM$_i$, for $i = 1, .., x$. Since the application fails only if no BSM message is

received before t_{react}, and since the reliability of one BSM is independent of that of another BSM, we use the unreliability $Q(t) = 1 - R(t)$, i.e., the probability of all x messages being lost, which is

$$Q(t) = \prod_{i=1}^{x} Q_i(t_i) \tag{3.1}$$

where Q_i is the probability that BSM message i was not received, i.e., the PER of BSM$_i$, and t_i is the time BSM$_i$ should be received. Note that this time is linearly related to the distance between HV and the jammer when BSM$_i$ should be received.

In order to obtain the application unreliability indicated in Equation 3.1 we need the values of Q_i. Packet error probabilities are derived from the Signal-to-Jamming Ratio (SJR), which depend on signal powers and distances, as it applies for each BSM$_i$. We assume that jamming noise dominates any other noise. The SJR is given in [51] by

$$SJR = \frac{P_t G_{tr} G_{rt} R_{jr}^2 L_j B_j}{P_j G_{jr} G_{rj} R_{tr}^2 L_r B_r} = \frac{P_t G_{tr} R_{jr}^2 L_j}{P_j G_{jr} R_{tr}^2 L_r} \tag{3.2}$$

Also we can use the Jamming-to-receiver Signal Ratio (JSR), which is the inverse of SJR

$$JSR = \frac{P_j G_{jr} G_{rj} R_{tr}^2 L_r B_r}{P_t G_{tr} G_{rt} R_{jr}^2 L_j B_j} = \frac{P_j G_{jr} R_{tr}^2 L_r}{P_t G_{tr} R_{jr}^2 L_j} \tag{3.3}$$

where subscript j refers to the jammer, r to the receiver and t to the transmitter. The transmission power of node y is denoted by P_y, the antenna gain from node y to z by G_{yz}, the distance between nodes y and z by R_{yz}, the communication link's signal loss by L_r, the jamming signal loss by L_j, and the nodes y bandwidth by B_y. After cancellation of terms that are equal, due to the assumption that the jammer and OBU have equal capabilities, the SJR to the right of the equation remains. As stated before we assume that distance between the HV and RV is constant, even during braking. Using the standard definition of EIRP we get

$$SJR_{dB} = EIRP(t)_{dB} - EIRP(j)_{dB} + 20\log R_{jr} - 20\log R_{tr} \tag{3.4}$$

The impact of the SJR is now used to calculate the PER. However, we need to consider modulation for different bit rates. As stated in subsection 3.4.1, the DSRC according to ASTM E2213-03 standard [36], uses Orthogonal Frequency Division Multiplexing and uses Binary Phase Shift Keying or Quadrature Phase Shift Keying and 16-Quadrature Amplitude Modulation, which support the mandated data rates of 3Mbps, 6Mbps and 12Mbps. These rates will be subject of our investigations, i.e., for 3Mbps using BPSK with coding rate 1/2, for 6Mbps using QPSK with coding rate 1/2, and for 12Mbps 16-QAM with coding rate 1/2, as defined in [36] and shown in Table 3.9. Assuming Additive white Gaussian noise (AWGN) channel model, the Bit Error Rate (BER), or Bit Error Ratio $P_{b(PSK)}$ for BPSK and QPSK can be expressed using the complementary error function *erfc*() [55] as

$$P_{b(PSK)} = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N}}\right) \tag{3.5}$$

where E_b / N is the ratio of average energy per bit to noise power spectral density.

For 16-QAM we have the following BER with $k = \log_2 16 = 4$

$$P_{b(QAM)} = \frac{3}{2k} erfc\left(\sqrt{\frac{kE_b}{10N}}\right) \tag{3.6}$$

This is related to the SJR by

$$\frac{E_b}{N} = SJR\frac{B}{R} \tag{3.7}$$

where R is the channel information data rate and B is the channel occupied bandwidth, as shown in Table 3.10.

The packet error probability P_p is now approximated by

$$P_p = 1 - (1 - P_b)^N \tag{3.8}$$

where N is the number of bits of the BSM message. Whereas, this equation assumes independence of faults. It can still serve as an approximation, since jamming is considered constant over the jamming time and is reflected in the BER. For details about the impact of

bit-to-bit dependence on packet error rate the reader is referred to the literature, e.g., [56].

Information Data Rate (Mbits/s)	Modulation	Coding Rate	Coded bits per Subcarrier N_{BPSC}	Coded bits per OFDM symbol N_{CBPS}	Data bits per OFDM symbol N_{DBPS}
3	BPSK	1/2	1	48	24
6	QPSK	1/2	2	96	48
12	16-QAM	1/2	4	192	96

Table 3.9: Data Rate and Modulation Parameters.

3.8.4 Impact of Redundancy on Unreliability $Q(t)$

Considering only benign faults, a system consisting of N redundant subsystems C_j, $j = 1, .., N$, fails only if all N subsystems fail, i.e., it functions as long as at least one subsystem functions up to specifications [14]. The unreliability of such system is therefore the product of the unreliabilities of the subsystems. In our case the application unreliability Q_{C_j} of each channel $C_j(t)$ is defined by Equation 3.1 and thus

$$Q_N(t) = \prod_{j=1}^{N} Q_{C_j}(t) = \prod_{j=1}^{N} \prod_{i=1}^{x} Q_i(t_i) \tag{3.9}$$

This equation assumes independence of faults. However, its usage is argued as a good approximation due to the fact that jamming of different channels is assumed to be by different radios and the transmission of dissimilar messages is not time-synchronized, e.g., they are not coordinated to overlap.

3.9 Results

3.9.1 Impact of Jamming without Considering Channel Power Limits

The JSR for two constant jammers is plotted in Figure 3.10, for the scenario of Figure 3.9a). The assumptions for the graph are as follows: P_t was set to 20dBm, P_j to 10 dBm and 15dBm, R_{tr} is set to the safety distance between vehicles of 3s, or 45.9m, corresponding to a

Parameter	Value	Parameter	Value
Number of Subcarriers, Total (N_{ST})	52 (48 Data Subcarrier + 4 Pilot Subcarrier)	Information Data Rate	3, 4.5, 6, 9, 12, 18, 24, and 27 Mbit/s (3, 6, and 12 Mbit/s are Mandatory)
Subcarrier Frequency Spacing (ΔF)	156.25 KHz (10 MHz / 64 total OFDM subcarriers)	Modulation	BPSK OFDM, QPSK OFDM, 16-QAM OFDM, 64-QAM OFDM
T_{FFT}	6.4 μs (1/ΔF)	Coding Rate	1/2, 2/3, 3/4
Guard Interval (T_{GI})	1.6 μs (T_{FFT}/4)	Channel Bandwidth	10 MHz (Occupied Bandwidth 8.3 MHz)
OFDM Symbol Duration	8 μs (T_{GI} + T_{FFT})	CH172 Transmit Power Level	33 dBm EIRP, 28.8 dBm i/p power
PLCP preamble duration (T_{PR})	32 μs	CH178 Transmit Power Level	44.8 dBm EIRP, 28.8 dBm i/p power
Duration of the SIGNAL BPSK-OFDM symbol (T_{SIG})	8 μs (T_{GI} + T_{FFT})	CH184 Transmit Power Level	40 dBm EIRP, 28.8 dBm i/p power
Packet Size	300 bytes (2400 bits)	Jammer Transmit Power Level	33 dBm EIRP, 28.8 dBm i/p power

Table 3.10: Configuration Parameters.

vehicle speed of 35mph, with an assumed reaction time of 1s. R_{jr} is the varying distance from the jammer as the HV moves. All other parameters, G, L and B are assumed equal for both, thus canceling each other out. The impact of thermal noise compared to the large jamming power is assumed negligible. If we assume a total safety distance of 3s and subtract 1s of reaction time, this only leaves the first 2 seconds to receive BSM messages before it is too late to react. Since the interval between two BSM messages is 0.1s, a maximum of 20 BSM messages could possibly be received, and thus the last message that may be received in Figure 3.9a) is BSM_{20}.

As can be seen in the graph, the impact of the jammer increases with the message index, with BSM_1 least affected by jamming.

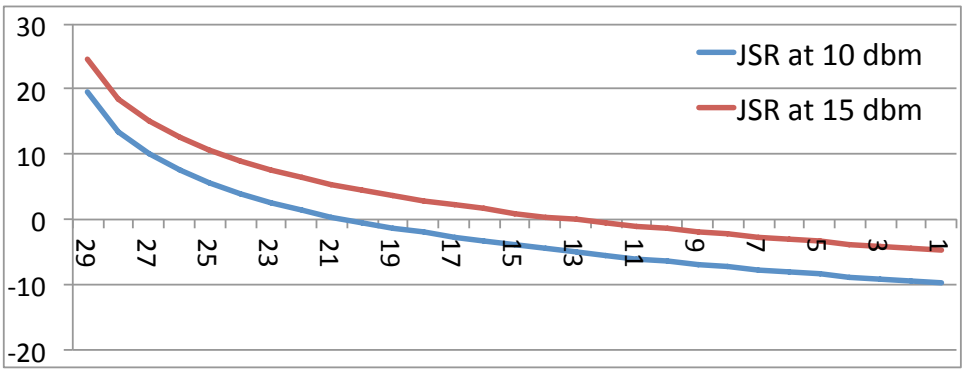

Figure 3.10: Jamming-to-Signal Ratio in dB Related to Messages BSM_i

Impact of Constant Jammer on $Q(t)$

The impact of the JSR is now used to calculate the PER. The BSM messages are sent on the 6Mbps Channel CH172 using QPSK 1/2 encoding [36][38]. The bit error probability P_b for QPSK can be expressed using Equation 3.5. We assume a BSM message length of 300 Bytes, giving $N = 2400$ bits. The packet error rate P_p is the unreliability Q_i used in Equation 3.1. Its impact on the FCW application's unreliability $Q(t)$ in the case of a constant jammer is shown in Figure 3.11. The x-axis labels i indicate the total number of BSM messages that were sent by t_i and may be received before t_{react}, whereas the y-axis is the corresponding unreliability $Q(t) = \prod_1^x Q_j(t_j)$ for $x = i$. For the 15dBm jammer the application unreliability is close to 1 (total failure) during most of the plot. However, in the 10dBm case the unreliability decreases drastically. The final unreliabilities, with 20 BSM messages sent, for the 15dBm jammer scenario was 0.993, which is unacceptable. For the 10dBm jammer case however the jammer has insignificant impact, i.e., the probability of missing all 20 BSM messages due to jamming was 10^{-18}.

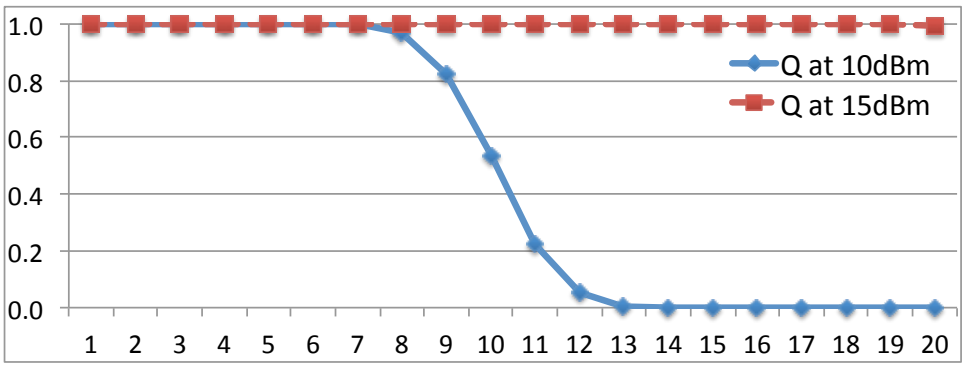

Figure 3.11: $Q(t)$ under Constant Jamming over Number of BSM Messages Sent

Impact of Random Jammer on $Q(t)$

Figure 3.11 was for the worst case jamming scenario, i.e., a constant jammer. The reliability in the presence of a random jammer is highly affected by the probability that a BSM is sent during a sleep period. To simplify matters, let P_s be the probability that an entire BSM falls into a sleeping period.

If a BSM message is sent during any sleep time before the reaction time t_{react}, the application reliability is at least as high as the probability of receiving that unjammed BSM message. Thus, the application unreliability as it is affected by random jamming is

$$Q_{rand}(t) = \prod_{1}^{x}(1 - P_s)Q_i(t_i) \tag{3.10}$$

where $Q_i(t_i)$ is the unreliability of BSM reception at t_i during jamming. Equation 3.10 shows that the unreliability is dominated by the probability that at least one BSM falls in the sleeping period. The impact of sleeping probability P_s on unreliability is shown in Figure 3.12. For the 15dBm jamming scenario the unreliability, which was unacceptable in Figure 3.11, falls off very fast with increasing sleeping probability P_s. In fact, increasing jamming power has little impact on the graph, i.e., it is P_s that impacts $Q(t)$. It is obvious that $Q(t)$ in the 10dBm case is already insignificantly small, even with $P_s = 0$. This special case of random jamming, i.e., where sleeping probability is zero, is equivalent to constant jamming. Recall that the unreliability for constant jamming in Figure 3.11 was 10^{-18} for the

20 messages.

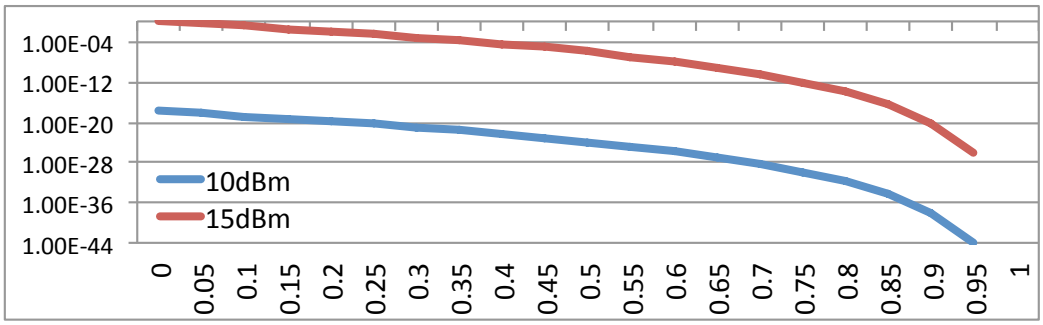

Figure 3.12: Impact of Sleeping Probability (x-axis) on Q(t) (y-axis)

Impact of Redundancy on $Q(t)$

A dual-redundant system can be defined by adding redundancy using ACM, as described in Subsection 3.6.1. The redundant channels are CH172 and CH178 with individual unreliabilities denoted by $Q_{172}(t)$ and $Q_{178}(t)$ respectively. This leads to an application unreliability $Q_{dual}(t) = Q_{172}(t)Q_{178}(t)$, which can be simplified to $Q_{dual}(t) = Q(t)^2$ if we assume that both channels have the same reliabilities. If we extend the redundancy level by one, e.g., by including redundancy using PVD, we have a triple-redundant system, which for equal reliabilities results in $Q_{triple}(t) = Q(t)^3$.

The unreliability of a system with redundant channels is unaffected by jamming as long as one channel is unjammed, i.e., jamming has no effect unless it covers all channels. Now assume that all channels are jammed. Figure 3.13 shows the impact of redundancy on unreliability of such scenario as a function of the number of BSM messages sent before t_{react}, which in our case is 20. It can be seen that as the redundancy level goes up, the unreliability during lower power jamming goes down. However, as expected, redundancy in the presence of all channels jammed at full power has limited benefit. The real benefit is when the power of the jammer is spread over all redundant channels, and that impact will be significant.

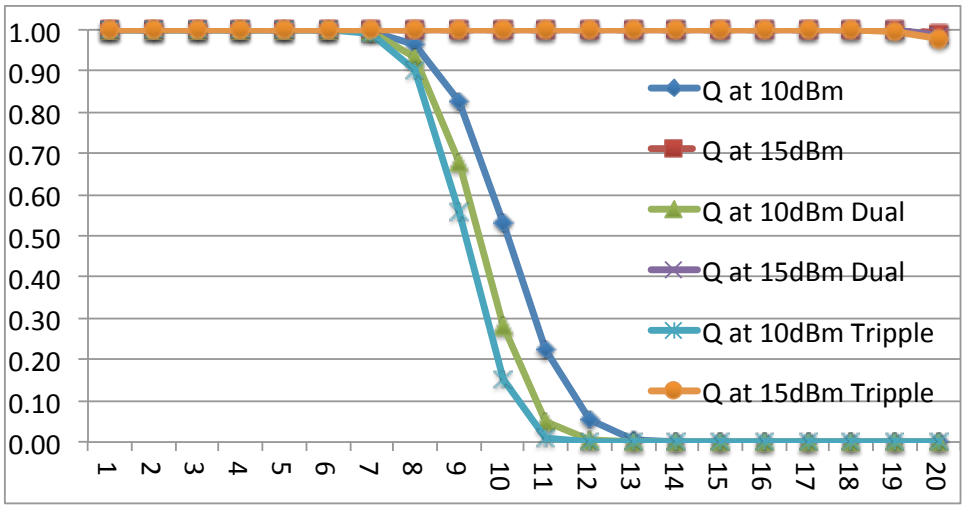

Figure 3.13: Impact of Redundancy on $Q(t)$ (y-axis) for BSM Messages (x-axis)

3.9.2 Impact of Jamming Considering Different Channel Power Limits

Impact of Redundancy on $Q(t)$ under Constant Jamming

The impact of constant jamming on the PER of the safety channel CH172, the first redundant channel, i.e., control channel CH178, and the second redundant channel CH184 is shown in Figure 3.14. As can be seen in the graph, the impact of the jammer increases with the message index, with BSM_1 least affected by jamming.

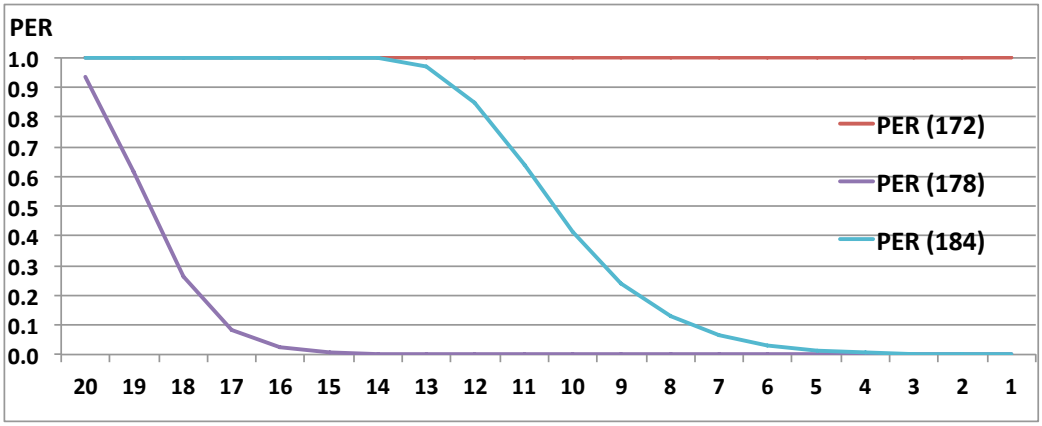

Figure 3.14: PER for BSM_i during Jamming for Different Channels

The assumptions for the graph are as follows: the EIRP of the transmitter and jammer are

33dBm, R_{tr} is set to the safety distance between vehicles of 3s, or 46.9m, corresponding to a vehicle speed of 35mph, with an assumed reaction time of 1s. R_{jr} is the varying distance from the jammer as the HV moves. The impact of thermal noise compared to the large jamming power is assumed negligible. We assume a BSM message length of 300 Bytes, giving $N = 2400$ bits. If we assume a total safety distance of 3s and subtract 1s of reaction time, this only leaves the first 2 seconds to receive BSM messages before it is too late to react. Since the interval between two BSM messages is 0.1s, i.e., BSM messages are broadcast every 100ms [46], a maximum of 20 BSM messages could possibly be received, and thus the last message that may be received in Figure 3.7 is BSM_{20}. A summary of the parameter used in the derivation of the application reliabilities is shown in Table 3.9 and Table 3.10. This data was extracted from ASTM E2213-03 standard [36].

As can be seen in Figure 3.14, channel CH172 is completely jammed, i.e., PER $= 1$, and thus any safety application only relying on this channel will fail. For channel CH184 the PER only starts deteriorating starting with message 6, implying that the lower numbered messages are unlikely to be corrupted. Channel CH178 however is mostly resilient to jamming as corruption begins with message 16, i.e., all lower numbered message have very high probability of being delivered uncorrupted.

The impact of constant jamming on the PER of the safety channel CH172, the first redundant channel, i.e., control channel CH178, and the second redundant channel CH184, using 3Mbps and 6Mbps rates, is shown in Figure 3.15. As the HV approaches the jammer the PER of the safety messages increases. It can be seen in the graph that the impact of the jammer increases with the message index, with BSM_1 least affected by jamming. However, the exponential deterioration affects channels differently. Channel CH172 is (for all practical purposes) completely jammed for 3Mbps, with even worse results for 6Mbps and 12Mbps (not shown in the figure). Channel CH184 for 3 Mpbs has very low PER (less than 10^{-3}) for the first 4 messages, and only starts showing practical impact with message 5. For 6Mbps however, even the best PER achieved for message 1 is already slightly over 0.3, which is violating the acceptable rate of the standard [36]. The most reliable channel is CH178, which only starts seeing deterioration for 3Mbps and 6Mbps starting with messages 15 and 9 respectively. All channels with 12Mbps experienced unacceptable PER for all messages, and

they were not depicted in the figure.

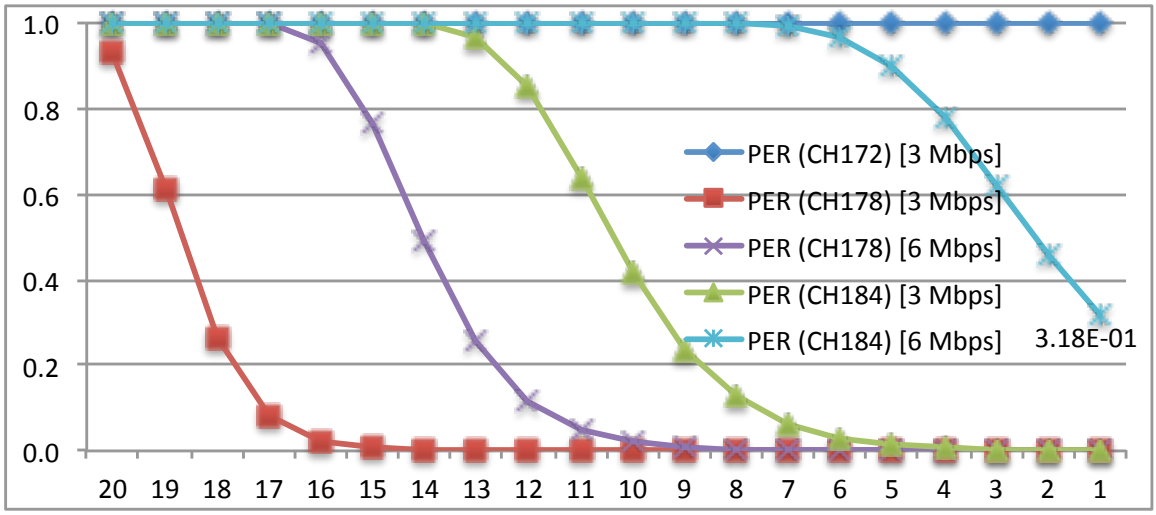

Figure 3.15: PER of Safety Message i (x-axis) Using 3Mbps and 6Mbps for Different Channels Affected by Constant Jamming

By using Redundant approach, the unreliability of a system with redundant channels is unaffected by jamming as long as one channel is unjammed, i.e., jamming has no effect unless it covers all channels.

The unreliability of the FCW safety application, defined in Equation 3.1 and Equation 3.9, for 3Mbps communication, is shown in Figure 3.16. Note that the product of the equation is dominated by the product terms with smallest unreliability.

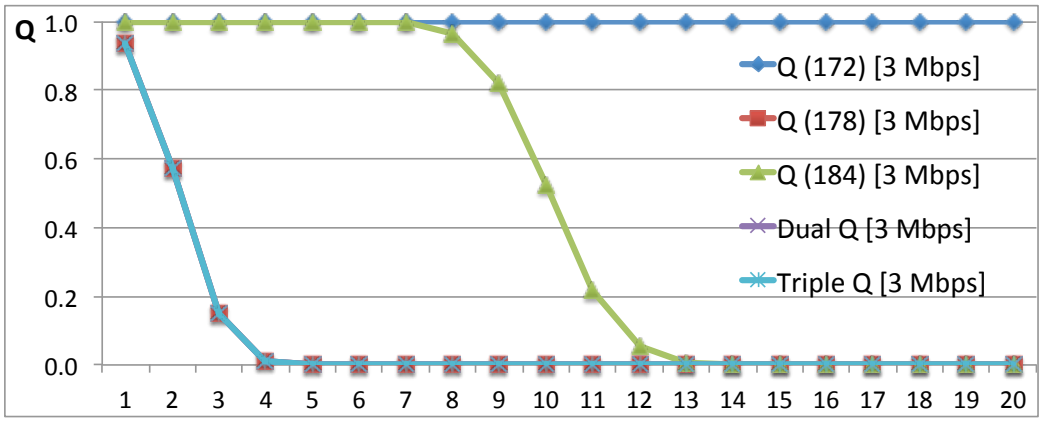

Figure 3.16: Unreliability Q of Different 3Mbps Jammed Configurations

Only using safety channel CH172, the FCW application fails totally, as no error-free packets were received. On the other hand, the first redundant channel, i.e., control channel CH178, is extremely robust. This can be observed when one considers the time window in which safety messages could be potentially received, which is given in the x-axis of Figure 3.16. When the safety distance between the HV and the RV in Figure 3.7 allows a message window greater than three messages, the FCW receives messages with very high probability. This point is reached for channel CH184 when the message window grows beyond thirteen. Since channel CH178 is used in the dual and triple redundant schemes, its unreliability dominates that of the schemes, resulting in FCW to work reliably.

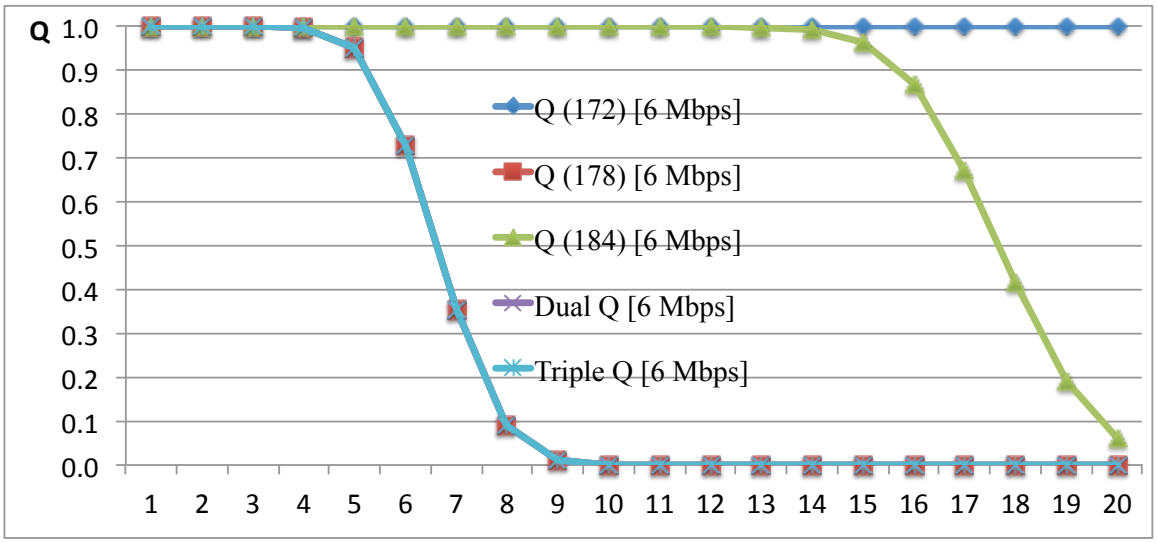

Figure 3.17: Unreliability Q of Different 6Mbps Jammed Configurations

In Figure 3.17, which considers 6Mbps communication, similar behavior can be observed. However, only channel CH178, and the redundancy schemes using it, allows FCW to work reliably. In the figure, the plot for the unreliability of CH178, dual and triple-redundancy overlap. Channel CH184 is borderline, as only one BSM provides reasonable unreliability of 0.06, i.e., the BSM at x-axis label 20. Therefore, in general, we suggest to not use this channel for 6Mbps or higher.

The dual-redundant schemes for different data rates are compared in Figure 3.18. For the FCW application the 3Mbps and 6Mbps communication is not affected by jamming, i.e.,

given the assumed minimal safety distance between the vehicles the unreliability of jamming of both falls below 10^{-43}. The 12Mbps communication however fails as unreliability remains close to one. This is a very important observation, which makes us conclude that safety applications should not use this data rate, as communication fails under jamming, i.e., in the figure the application unreliability stays close to one during the entire time before it is too late to react.

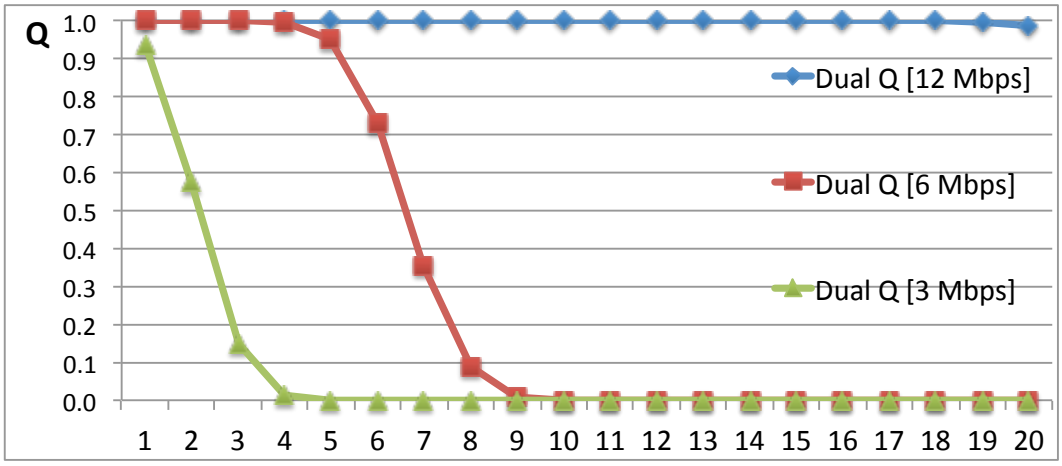

Figure 3.18: Impact of Data Rate of Dual Configuration on Unreliability Q during Jamming

Given the packet error rates of Figure 3.15 the FCW unreliabilities were derived for triple-redundant configurations, as shown in Figure 3.19. The unreliabilities shown reflect the number of messages, i.e., terms, used in Equation 3.1. Thus, the best unreliabilities are achieved when all 20 messages are used, where the dominating messages are the first ones received, i.e., the message with lowest PER in Figure 3.15, which is message 1. Most importantly, for 12Mbps even the triple-redundant implementation results in unacceptable unreliability close to one. When using lower data rates, i.e., 3Mbps and 6Mbps, all triple configurations can, for all practical purposes, completely overcome jamming.

Figure 3.19 also shows the unreliability of a triple-redundant configuration using different data rates, which overlap with the 6Mbps plot. Here CH172 and CH184 use 3Mbps, but CH178 uses 6Mbps. The rational for using a higher rate for control channel CH178 is that this channel is used by all applications and thus bandwidth is precious. CH178, even with the

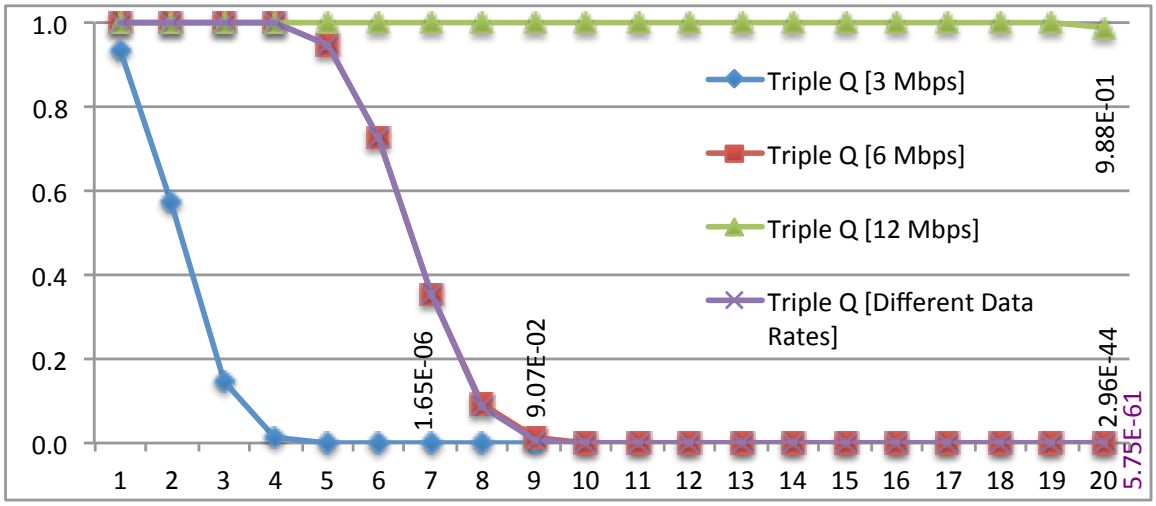

Figure 3.19: Unreliability Q of Different Triple Redundant Configurations, Constant Jammer, over Total Number of BSM Sent

higher rate, is providing the dominating terms for Equation 3.1 and Equation 3.9, which results in extremely low unreliabilities.

Impact of Redundancy on $Q(t)$ under Random Jamming

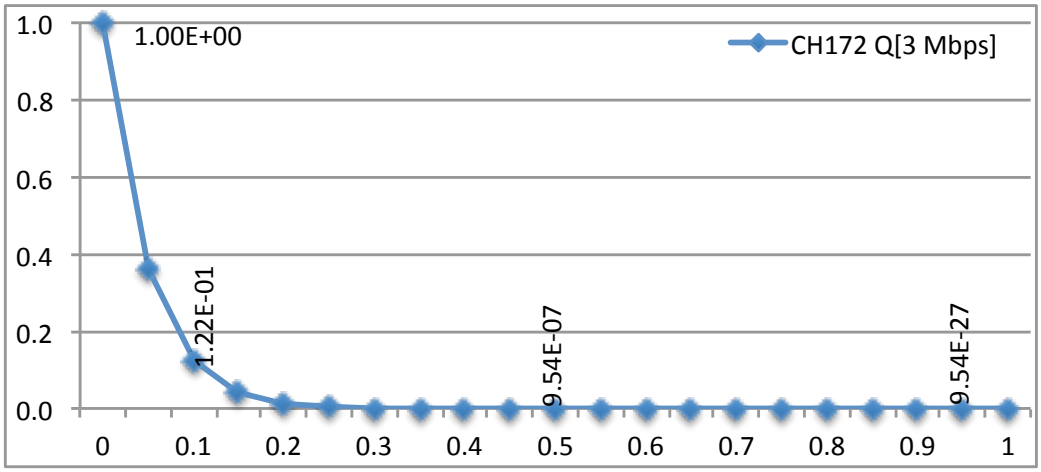

Figure 3.20: Unreliability Q of CH172 Using 3Mbps under Random Jamming, over Sleeping Ratio

The unreliabilities of random jamming for different sleep ratios are shown in Figure 3.20 for CH172 using 3Mbps, and for different triple redundant scenarios in Figures 3.21 and 3.22.

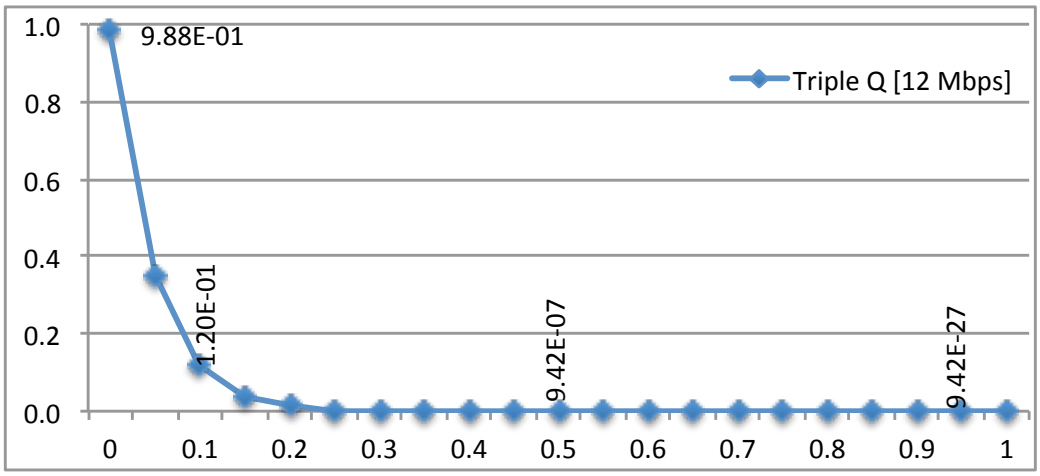

Figure 3.21: Unreliability Q of 12Mbps Configuration under Random Jamming, over Sleeping Ratio

The most important observation is that the unreliabilities now are dominated by the sleep ratios. All scenarios, no matter whether the data rates are 3, 6, or 12Mbps, are unaffected by jamming unless the sleeping times are small, e.g., less than 25% in Figures 3.20 and 3.21. The justification for this is that as the sleeping times increase the probability for messages to not experiencing jamming is high. Thus even the 12Mbps scenario, which was not usable in the constant jammer case, is immune to random jamming, if the sleep ratio is above 25%.

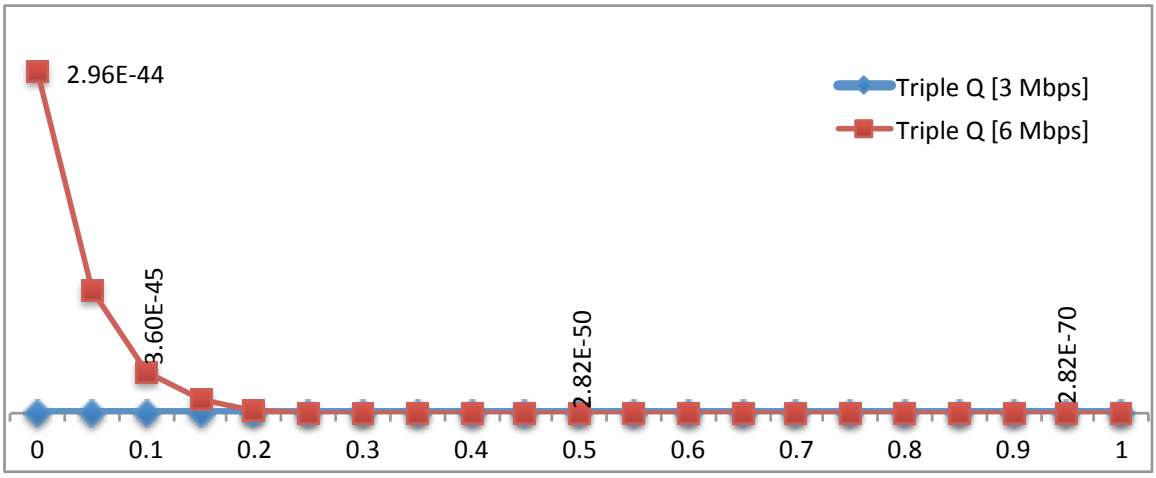

Figure 3.22: Unreliability Q of 3 and 6Mbps Configurations under Random Jamming, over Sleeping Ratio

Extreme resilience against random jamming can be observed in Figure 3.22 for triple redundant configurations using 3 and 6Mbps. One should note that the unreliabilities are insignificantly low, as even the constant jammer, which is a special case of random jammer with sleeping time zero, could cope in this configuration. All results for random jammers do not even consider the time the jammer would need to switch channels, e.g., to switch between CH178 and CH184, which is bound by 2ms [36]. In site of message delays of approximately 6.3ms, 3.5ms and 2.3ms for 3Mbps, 6Mbps and 12Mbps rates respectively, considering maximum message length, such channel switching would effectively count as non-jamming time.

Chapter 4

A SURVIVABLE REAL-TIME TRAFFIC CONTROL SYSTEM

4.1 Introduction and Background

Advanced traffic signal systems that adapt to changing traffic conditions in real time are at the core of most ITS traffic management applications. In this chapter, we describe a resilient real-time weather-responsive traffic signal control system which we refer by WRSI as introduced in chapter 1. The WRSI intents to improve the efficiency and safety of traffic signal operations during inclement weather conditions. The system receives and analyzes road weather information from an integrated surface transportation weather observation data management system and adapts signal timing in response to changes in road surface conditions and/or visibility level.

The control system described here has to address these fault tolerance and resilience requirements during the execution of two tasks. The first task consists of the system accessing near real-time atmospheric, weather, visibility, and road surface condition information from the FHWA Clarus data system [57], whereas the second task adapts signal timing in response to inclement weather based on this information. Since the system access data on domains outside its secured local communication networks, the data exchange architecture needs to be designed in a way that is resilient against cyber attacks and intruders.

The control system was designed with resilience consideration in mind, utilizing two essential software design approaches: *Design for Survivability* [2, 58] and a *Measurement-Based Methodology* [3, 59]. The first approach is derived from the concept of *Design for Testability*, which addressed the impossibility of completely testing VLSI circuits by designing them with testability in mind [2]. Now, survivability considerations are similarly integrated into the design, rather than in an add-on fashion. The second approach, i.e., Measurement-Based Methodology, was proposed for critical applications that rely on measurements of operational systems and on dependability models to provide quantitative survivability with certain user-defined confidence levels. The software design incorporates

self-monitoring techniques for fault detection and recovery to maximize the resilience of the system.

4.2 Contributions

As stated in the introduction, the contributions of this paper are of theoretical and practical nature. The main theoretical contribution is the combination of the approaches introduced in [3, 4, 5] into one comprehensive architecture with survivability and resilience characteristics. Furthermore, the subsystem that monitors the application program is extended to three monitoring approaches: 1) detection of dependency violations, 2) identification of anomalies through exception triggers and data sensor analysis, and 3) detection of off-nominal, non-certified executions. The theory of the latter is extended to allow for certification of executions based on *Behavior Sets*. Furthermore, a dual-bound threshold approach for detecting off-nominal executions is introduced.

The practical significance is in the description of the actual system with extensive evidence to the resilience of the architecture based on observation of the system in action and data collection by the system during the year 2012.

4.3 Related Work

Run-time monitoring refers to the process of monitoring the system's behavior in real-time. The goal is to determine whether the system performs its tasks to specifications or if there are anomalies in the execution patterns. The latter could indicate that the system is compromised. Has the software experienced a fault, has the system been attacked, or is it executing correctly in a fashion that we just have not observed before? These questions have plagued the dependability and security communities for decades. Fault detection and treatment have been researched by the dependability and software engineering communities. Attack recognition, i.e., intrusion detection, is a very complex problem and detecting patterns or anomalies has been a constant hot topic in the intrusion detection community, e.g., signature-based approaches or anomaly detection. Especially in anomaly detection the critical issue is where one should set the threshold for deciding what is normal and what is

not [60]. It should be noted that the methods of intrusion detection, i.e., the claim made about the mechanisms used for detection, has not been without controversy [61].

Detection of off-nominal executions implies that one knows what a nominal execution looks like. We do not attempt to mimic anomaly detection, but utilize the detection of previously observed executions patterns, e.g., profiles, versus those we have just not seen before. Instead of focusing on “what is abnormal”, we focus on “what is normal”. Thus everything outside of previously identified, i.e., nominal, behavior is simply assumed off-nominal.

The research presented here is based on early work using frequency spectra of observed system executions [62]. They presented a real time approach to detect system behavior deviating from normal activities, with focus on attacks on the system software. Similar approaches were taken in [63], where signatures related to observed frequency behaviors were constructed for attack signature detection. Both approaches used profiling based on injected execution handles, an approach that is also used in Unix systems, e.g., when compiling C programs with the -g option. The concepts were later combined into a measurement based methodology for embedded software systems [59], which was the starting point for this research.

4.4 Real-Time Control Application

The traffic control infrastructure is augmented with capabilities driven by performance and safety improvement goals.

4.4.1 Control System Components and Operation

The real-time weather responsive system is shown in Figure 4.1. The non-shaded components are the existing ITS system whereas the shaded components are additions and are the components that implement real-time weather response and system resilience. The traffic lights in an intersection are controlled by a traffic controller hosted in a cabinet located in the intersection. The traffic controller is connected to a switch, or hub, to the ITS control network, which is either physically totally separated or connected to the internet via a

firewall. It should be noted that the separation of the ITS control network and the internet is critical and any access through the firewall has to be extremely limited and under strict compliance with security policies.

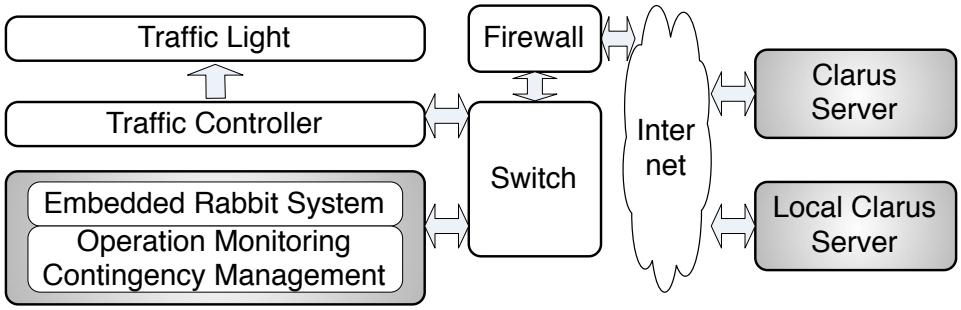

Figure 4.1: Overview of the Real-Time Weather Response System

Weather data is collected by the Clarus system from a network of Environmental Sensor Stations (ESS) of participating states. The network of ESS can be viewed at [57] by the general public. This ESS data is accessible via the internet from the Clarus server, after it undergoes quality and consistency checks based on Clarus quality checking algorithm [64]. Due to this quality check, survivability considerations do not include verification of the original Clarus data. An embedded Rabbit-based system located in the traffic signal system in the intersection retrieves the Clarus system data from the Clarus server or a local mirror site, analyses the relevant data, and computes changes to the signal timing. Upon approval, the signal timing changes are made in the traffic controllers by the embedded Rabbit system. Signal timing plan adaptations include changes such as modified all-red or yellow clearance intervals or traffic signal efficiency parameters such as minimum green, maximum green, passage time as well as different coordination parameters. Suggested changes depend on multiple factors such as approach speed, pavement surface conditions, visibility, and the mode of signal operations.

The embedded Rabbit System shown in Figure 4.1 is based on a Rabbit RCM4300 micro-controller running Dynamic C[1] version 10.7 supporting a variety of services including Advanced Encryption Standard (AES) and Secure Sockets Layer (SSL). It is the core

[1]Dynamic C and Rabbit are registered trademarks of Digi International Inc. See documentation at www.rabbit.com

hardware in the system that communicates with the traffic controller through the ethernet switch. To facilitate communications, the controller and microprocessor must follow the NTCIP communication standard (AASHTO 2005), a family of standards for transmitting data and messages between different devices used in Intelligent Transportation Systems.

The Dynamic Object Simple Transportation Management Protocol (STMP)/UDP/IP Ethernet protocol stack is used to facilitate the NTCIP-based communication between the microprocessor and the traffic controller. A computer, connected to the microprocessor through the cabinet serial connection, is used to setup and add the control logic to the microprocessor. Because the microprocessor is directly connected to the traffic controller through the ethernet port of the switch, the connection is not sensitive to the cabinet configuration.

4.4.2 The Clarus System Weather Data Support

The data that is needed to implement real-time weather responsiveness comes from sensors of the ESS. The Clarus System shown in Figure 4.1 maintains the location of all ESS. The ESS most suitable for the specific traffic signal system, e.g., the one with closest proximity to the intersection, needs to be identified and a subscription for that ESS is generated. The subscription, which may include data from a single or multiple specified ESSs, is made available via the Clarus System's subscription web site in the format of a Comma Separated Value (CSV) file. It should be noted that the data is not queried from a data base server, but simply accessed directly over the web and is, unless protected from general access by a password, publicly readable. Specifically, a list of observations, i.e., the actual CSV files, is made available in regular intervals typically ranging from 5 to 15 minutes. The specific observations in the list depend on the capabilities associated with the ESS associated with the subscription. Within a subscription the observation files follow the file naming convention $date\_time.csv$. An observation file contains data for specific *Observation Type IDs* (ObsTypeID). The first line is a header line describing the values present in each line of data. A relevant subset of these values is used later by the system to calculate changes to be made to the traffic controller. Since a subscription is not limited to contain data from only one ESS but can be specified to contain data from multiple sensors, e.g., to include neighboring ESS,

the control algorithms of the weather responsive system can take advantage of data fusion, thus taking advantage of a "larger view".

4.4.3 Software System Architecture

An overview of the software system that controls the weather responsive system is shown in Figure 4.2, where shaded areas refer to external hardware interfaces. The Rabbit system,

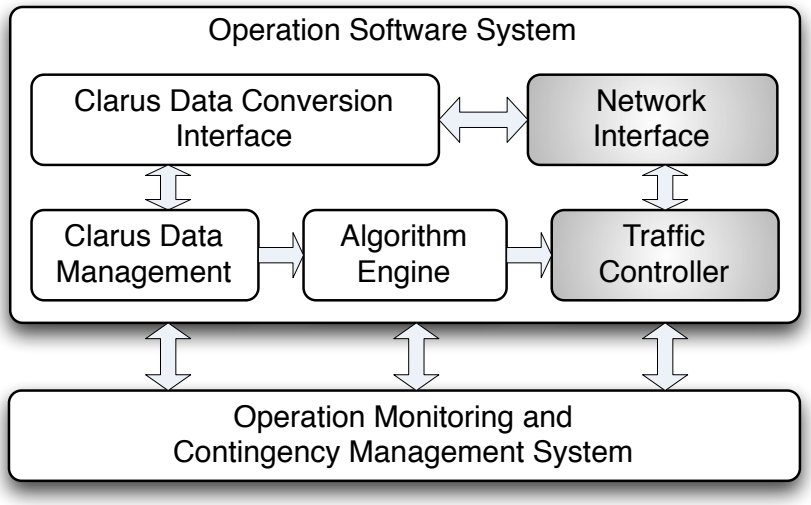

Figure 4.2: Overview of the Software System Architecture

which we will refer to simply as "Rabbit", executes the application control software, which consists of the *Operational Software System* and an *Operation Monitoring* and *Contingency Management System*. The operation software system connects to either a *Local Clarus Server* (LCS), which is simply a local mirror supplying the Clarus subscription data, or the Clarus System, using the *Network Interface* to the internet. In regular intervals that are specified by the Clarus subscription the Clarus data is read and converted by the Rabbit, the desired sensor data is extracted, and specific algorithms compute changes to the control parameters of interest, e.g., yellow timing adjustments. The traffic controller is then updated. All this is monitored at run-time via the instrumentation telemetry by the *Operation Monitoring and Contingency Management System*. That is, the Rabbit monitors the execution of its software in real-time by sensor points that are injected into the software.

4.5 Formal Model of Systems Architecture

The system architecture is guided by the design methodology and general principles shown in [3, 59]. It starts with the view of a general system as two distinct abstract machines that define the implementation in the development of any software system. The first, called *Operational Machine* is the machine that interfaces directly with the hardware interface. It executes a set of operations O, e.g., "Get Data" or "Update Controller", with cardinality $|O|$. As the embedded system provides services to the hardware, the services cause the operational machine to perform a series of actions referred to as *operations*. Each operation causes the operational machine to perform a specific action. The transition from one operation to another marks an *operational epoch*. Thus, the purpose of this operational machine is to articulate exactly what the software system must do to provide the necessary services dictated by the software system requirements.

The second abstract machine, called *Functional Machine*, is a set of functionalities that describe exactly how each system operation is implemented. Each operation o_i in O uses one or more functionalities f_j from a set F of functionalities with cardinality $|F|$. Similar to the operational epoch the functional epoch is defined by transitions from one functionality to another. Functionalities are implemented by code modules, which in our case are written in Dynamic C, a C-like language with a unique multitasking environment as will be described later. The implementation of the functionalities in code results in a set of code modules M of cardinality $|M|$.

As the system operates, operations cause functionalities, implemented by modules, to be invoked, or functionalities cause operations to be performed. During system operation, i.e., while the application is running, the operational and functional machines can be monitored in realtime, assuming appropriate instrumentation is in place to allow this. In our case, the execution of the application running on the Rabbit is monitored in real-time by three different monitoring mechanisms as shown in Figure 4.3. The first mechanism, described by a Profiling Model, is based on analysis of realtime execution profiles. It will be used to describe measurement of typical behavior as the basis for what to expect, with a certain probability of error, in the future. The second mechanism, covered by a Dependency Model,

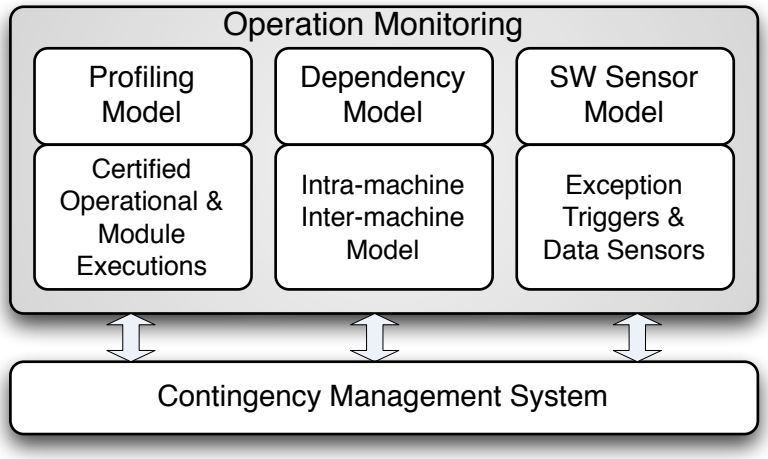

Figure 4.3: Using Profiling, Dependencies, and Data Sensor Monitoring

is monitoring for violations of state dependencies between the machines and within the machines. Any violation indicates an abnormal execution. The third mechanism, referred to as the Software Sensor Model, is based on the analysis of data supplied by specific data sensors within the software. These software sensors supply information that can be used for analysis or direct actions. All three mechanisms allow for the detection of off-nominal, unexpected, or invalid executions, which in turn are used by the Contingency Management System. We now describe each of the three models in detail.

4.5.1 Profiling-based Model

If one counts the invocations of operations, functionalities and modules over a specific period of time one can derive the respective *operational, functional* and *module profile*. These profiles will be used later in the analysis that may expose off-nominal executions.

To stay compatible with the notation used in [4, 59] we will use letters u, q and p for operational, functional and module profiles respectively. The notation is introduced using module profiling as an example. Let p_i denote the probability that the system is executing module m_i. Then $\mathbf{p} = (p_1, p_2, ..., p_{|M|})$ is the module profile of the system, i.e., it is the probability vector of the modules in M.

Non-synchronized Profiling

During execution of the system we are interested in observing the module profile over n epochs. Here we assume that n is not synchronized to a particular higher level machine, e.g., the operational machine's epoch.

This observed profile is denoted by $\hat{\mathbf{p}} = (\hat{p}_1, \hat{p}_2, ..., \hat{p}_{|M|})$, where $\hat{p}_i = c_i/n$ is the fraction of system activity due to invocations of module m_i and c_i is the count of invocations of m_i. As the software executes, invocations of modules are continuously monitored and module profiles are generated and analyzed. We want to keep track of these profiles. Let $\hat{\mathbf{p}}^k$ denote the k^{th} observed profile. Thus $\hat{\mathbf{p}}^k$ is the k^{th} observed module profile, observed over n epochs, which was preceded by $\hat{\mathbf{p}}^{k-1}$, observed over the previous n epochs, and so forth.

To get a feel for the expected evolving profile of the system, we want to establish the module profile equivalent of an "h-day moving average" in financial stock movements, i.e., we will derive a centroid that will serve as a reference for observed profiles. For that, just as in [59], we consider h sequences of n epochs each and define a centroid $\overline{\mathbf{p}} = (\overline{p}_1, \overline{p}_2, ..., \overline{p}_{|M|})$, where

$$\overline{p}_i = \frac{1}{h} \sum_{j=1}^{h} \hat{p}_i^j \tag{4.1}$$

Thus $\overline{\mathbf{p}}$ is a $|M|$-dimensional vector, and using the above financial metaphor, each element represents the "h-day moving average" of a specific stock (module), where a day is measured as n epochs. Furthermore, just as in the stock market, we don't know what the future brings but find it useful to track the past in order to establish "nominal", i.e., expected, behavior.

Synchronized Profiling

In the previous discussion the profiles reflect the behavior of the system. However, it is a single behavior. If there are multiple behaviors that a machine may exhibit, then one has to consider sets of behaviors, which we refer to as *Behavior Sets*. Let's consider the case where modules may exhibit different behavior during an operational epoch. Therefore, assume we synchronize module epochs to the operational machine, specifically an operational epoch.

Thus we make the assumption that n is the number of module epochs expiring during one operational epoch. In our application n is the number of module epochs during the 15 minute operation epoch at which the Clarus data is fetched. We now adapt the notation of the non-synchronized case and will switch from lower to upper case letters when considering behavior sets.

Now the observed profile is $\hat{\mathbf{P}} = (\hat{\mathbf{P}}_\mathbf{1}, \hat{\mathbf{P}}_\mathbf{2}, ..., \hat{\mathbf{P}}_{|\mathbf{M}|})$, where $\hat{\mathbf{P}}_\mathbf{i}$ is the behavior set of module m_i, i.e., it is a set of different profiles $\hat{p}_i = c_i/n$, which again represents the fraction of system activity due to invocations of module m_i and c_i is the count of invocations of m_i during the operational epoch of length n. Analogous to the non-synchronized case, let $\hat{\mathbf{P}}_i^k$ denote the behavior set of the k^{th} observed profile. Thus $\hat{\mathbf{P}}^k$ is the k^{th} set of observed module profiles, observed over n epochs, which was preceded by $\hat{\mathbf{P}}^{k-1}$, observed over the previous n epochs, and so forth.

Considering h sequences of n epochs each, we define a centroid of sets $\overline{\mathbf{P}} = (\overline{P}_1, \overline{P}_2, ..., \overline{P}_{|M|})$, where

$$\overline{P}_r = \overline{P}_r \cup p_i, \;\; 1 \leq r \leq |M| \;\;\; p_i = \frac{1}{h} \sum_{j=1}^{h} \hat{p}_i^j \tag{4.2}$$

for each behavior i. Thus $\overline{\mathbf{P}}$ is a $|M|$-dimensional structure of sets, and again using the above financial metaphor, each element represents the "h-day moving average" of a specific *set of stocks* (module), where a day is measured as n epochs, and again we want to track the past in order to establish "nominal", i.e., expected, behavior from a *set of behaviors*.

It should be noted that if each behavior set consists of only one element, then essentially $\overline{\mathbf{P}}$ is the same as $\overline{\mathbf{p}}$.

4.5.2 Dependency-based Model

The above discussion about profiles does not capture any dependencies *between* operations, functionalities, and modules, nor does it capture dependencies *among* them. We will refer to the first case as interdependencies and the latter as intradependencies.

Interdependencies

The relationship between operations, functions, and modules is defined by a graph $\mathcal{G}^{OFM}$, where the superscript simply indicates that the graph maps from O to F to M. The term *interdependencies* stems from the fact that $\mathcal{G}^{OF}$ and $\mathcal{G}^{FM}$ are bipartite graphs and $\mathcal{G}^{OFM}$ is a tripartite graph. An example is depicted in Figure 4.4, which shows three operations o_1, o_2 and o_3. The operations utilize specific functionalities, e.g., o_1 uses functionalities f_1 and f_2. Incidentally, f_2 is also used by o_3. The functionalities are implemented by modules, e.g., f_3 is implemented by module m_4, whereas f_4 is realized by m_4, m_5, and m_6. Checking

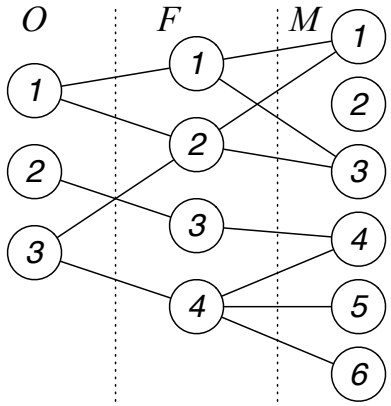

Figure 4.4: Mappings in ($O \times F \times M$)

interdependencies allows to identify any violation of mappings. For example if during the service of functionality f_1 module m_2 would be invoked, then at the functionality level one can detect a violation, since checking the graph one knows that m_2 is not utilized for f_1. Similarly, at the module level the violation would be detected as m_2 finds out that it should not be called as part of f_1 services.

Violations of interdependencies may be the result of scenarios where the mapping from specification to code is different than the reverse mapping, i.e., from code back to specification. In the latter case the code does more than it is supposed to do.

Sometimes there is a *one-to-one* and *onto* mapping from operations to functionalities, which is the case in our application. Then the mapping of ($O \times F \times M$) can be reduced to

$(O \times M)$, which is defined by $\mathcal{G}^{OM}$. We will refer to this mapping and its implied simplification as *Mapping Simplification Assumption* throughout the paper.

Intradependencies

It is not only of interest to know which functionality is used by an operation or which modules are used by a functionality, but also to know the dependencies within operations, functionalities, or modules. Those intradependencies can be defined by precedence graphs and are shown within the shaded areas of Figure 4.5. Specifically, dependencies between operations are defined by graph $\mathcal{G}^O = (O, \prec^O)$, where $\prec^O$ defines a precedence relation on the operations in O, i.e., if o_j depends on o_i then (o_i, o_j) is in $\prec^O$. Any violation of the precedence indicates a problem in the control flow of operations. We define similar graphs for functionalities and modules. Thus $\mathcal{G}^F = (F, \prec^F)$ and $\mathcal{G}^M = (M, \prec^M)$ are the graphs defining calling relationships between functionalities and modules respectively. It should be noted that $\mathcal{G}^M$ is the static call graph of modules in M created by the compiler. The operational, functional, and module dependency graphs are used to detect invalid or previously unobserved transitions. In Figure 4.5 the intradependencies are shown with solid

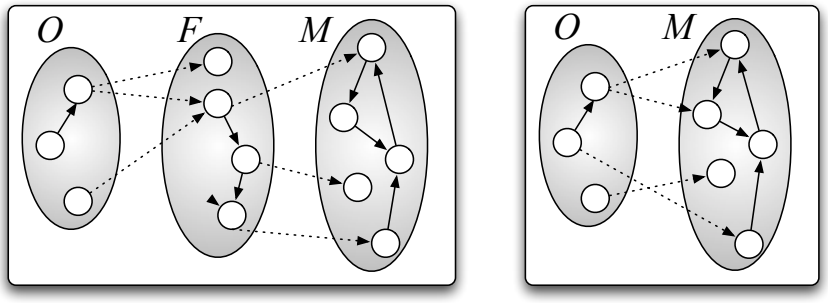

Figure 4.5: Dependencies within Operations, Functionalities, and Modules. Complete Model (Left), Simplified Model with One-to-One and onto Mapping in $(O \times F)$ (Right)

arrows. The interdependencies are indicated by dotted arrows. Whereas the left part of Figure 4.5 shows the complete dependencies, the simplified model under the Mapping Simplification Assumption is to the right.

4.5.3 Sensor-based Model

Not every behavior can be extracted from profiles or dependencies. Sometimes, specific data sensors are needed in order to observe specific data values or trigger exceptions, e.g., as the result of abnormal, missing, or unknown data items.

Exception Triggers

An *exception trigger array* has been implemented to identify and profile exceptions. An example of such trigger is the detection that a file that is supposed to be accessed does not exist, that specific external sensor data is no longer available, or any other observation that will cause the program to adapt. In general, any error condition can be viewed as a exception trigger.

In our application the driving motivator for exception triggers lies in the uncertainty of the availability of data provided by ESS. This can be due to ESS sensor failure, or simply a change in ESS configuration or hardware. Recall that there are large numbers of diverse data available from ESS and which specific data is available depends foremost on the capability of the ESS.

Data Sensors

Data sensors serve primarily for observation for analysis of specific numeric values. An example in our application is the value for the adjustment of the yellow period as computed by the algorithm engine shown in Figure 4.2. The correctness of data that is used in the computation is not questioned as Clarus provides quality and consistency checks based on its quality checking algorithm. However, the computed adjustment values provided by the data sensors can be analyzed to determine if they are feasible or make sense. Due to the NTCIP compliance of the traffic controller no safety violations can be forced, e.g., by selecting dangerously small yellow periods. But it could be possible that for some reason, may it be benign or malicious, the values are constantly too large. This would constitute a denial of service attack. Thus, no matter what the reason for the extreme durations may be, the data sensor makes analysis and thus contingency management possible.

4.5.4 Reduction of Nondeterministic Executions

One of the challenges in monitoring a system is dealing with the effects of nondeterminism of the executions. Typical sources of nondeterminism are interrupts, or even more importantly, context switching in multiprocessing based on time slicing. The Rabbit system uses a single processor in which multitasking is not achieved using time slicing; rather it is implemented using a model defined by *costatements*. A costatement is defined as a task in a nonpreemptive multitasking model. In practice, the main program of a control applications runs costatements (the tasks) in an endless loop, cycling from one costatement to the next. Each costatement has a statement counter, i.e., a program counter, which indicates which instruction of the costatement will execute when it gets a chance to run. Execution is switched from one costatement (of the infinite loop) to the next in a round-robin fashion when the currently executing costatement "yields" to the next costatement using explicit commands such as *yield*, *abort* or *waitfor(event)*. Note that these yielding mechanisms represent a model that is based on good behavior. The state of a costatement is called a *costate*. We will use the terms *costatement* and *costate* interchangeably.

An execution model in which there are no externally initiated task switches executes with a low level of nondeterministism, i.e., a task switch is explicitly demanded by the currently executing task: the *active costatement*. On the other hand this means that it is possible for a costatement to cause starvation by not yielding. However, a special mechanism called watchdog can be used to force timer interrupts. In this case the system deviates from its otherwise nonpreemptive execution model.

As operations, functionalities, and modules are called from within exactly one costatement at a time, it is possible to precisely determine the functionality and module that are being executed on behalf of a specific operation. Thus, the dispatching model results in executions with a low degree of nondeterminism, which is very desirable when working with profiles.

With the introduction of costates we can now extend the definitions of profiles presented in Subsection 4.5.1 to profile on a costate-basis. Thus, the observed profile $\hat{\mathbf{p}} = (\hat{p}_1, \hat{p}_2, ..., \hat{p}_{|M|})$, the k^{th} module profile $\hat{\mathbf{p}}^k$, the centroid $\overline{\mathbf{p}} = (\overline{p}_1, \overline{p}_2, ..., \overline{p}_{|M|})$ and d_k,

i.e., the distance from $\hat{\mathbf{p}}^k$ from centroid $\overline{\mathbf{p}}$, can now be defined on a costate-basis. For a costate α this leads to notation $\hat{\mathbf{p}}[\alpha]$, $\hat{\mathbf{p}}^k[\alpha]$, $\overline{\mathbf{p}}[\alpha]$ and $d_k[\alpha]$ respectively. Now it is possible for each costate α to have its own profiling, which is not affected by any non-determinism due to costate (task) switching, i.e., profiles of costates do not interfere.

4.6 Run-Time Monitoring

The run-time monitoring employs three monitoring approaches:

1. Validation of Dependencies

2. Detection of anomalies through data sensor analysis

3. Detection of off-nominal executions

Validation of dependencies is quite simple. Given the interdependencies and intradependencies as discussed in subsection 4.5.2, the system can detect any violation of mappings from operations to functionalities to modules in $\mathcal{G}^{OFM}$, the simplified $\mathcal{G}^{OM}$, and any violations of precedence in each $\mathcal{G}^{O}$, $\mathcal{G}^{F}$ and $\mathcal{G}^{M}$. For example, any call graph precedence violation, i.e., a module call sequence that is not in the precedence relation of $\mathcal{G}^{M}$, indicates a call sequence that is not intended. The reason for such call sequence cannot be extracted from the simple detection, however, possible reasons could be incorrect function pointers or perhaps a code injection attack.

Detection of anomalies in values returned by code-embedded data sensors is highly dependent on the sensor type. For example, in our application one data sensor is the actual adjustment to the yellow period of the traffic light. If the rate of change of the period is not in character with the environment parameters, e.g., the surface temperature, then perhaps the value is not correct. A simple range check of parameters may detect values that are outside of the expected range.

Detection of trigger events, coming from the second type of sensors, is more straightforward. The trigger events are simple signals that indicate certain events. They can be used to initiate specific actions, or can simply serve as a tracking mechanism, e.g., to keep track of how many times certain events have occurred over a specific time interval.

Detection of off-nominal executions implies that observed profiles are checked to establish if they meet an expected certified behavior, as will be described in detail in the following subsection.

4.6.1 Certified Executions

Nominal behavior can be refined to a costate level. Thus, given that different parts of the system execute in different costates, e.g., the application control loop is in one costate, the granularity of run-time monitoring is that of a costate, and thus more accurate than that of a system lacking that refinement.

The specifics of the instrumentation and how simple data structures can be used to achieve costate-based profiling is described in [3]. Using the data from the instrumentation, i.e., the observed profiles, one can detect off-nominal executions. However, rather than identifying off-nominal behavior, we "certify" nominal executions. Here we describe a dual-bound approach to execution certification introduced in [4] and expand it to consider behavior sets. For ease of presentation we first introduce the notion without behavior sets. Furthermore, we discuss certification using modules as an example, but the principle can be extended to functionalities or operations.

Certifying module behavior per costate is based on module profiles, $\hat{\mathbf{p}}^k[\alpha]$. The distance of the observed costate profiles $\hat{\mathbf{p}}^k[\alpha]$ from $\overline{\mathbf{p}}[\alpha]$ can be used so that departure beyond it indicates non-certified behavior of costate α. Specifically, we define two threshold vectors

$$\epsilon^{max}[\alpha] = (\epsilon_1^{max}[\alpha], ..., \epsilon_{|M|}^{max}[\alpha]) \tag{4.3}$$

$$\epsilon^{min}[\alpha] = (\epsilon_1^{min}[\alpha], ..., \epsilon_{|M|}^{min}[\alpha]) \tag{4.4}$$

where $\epsilon_i^{max}[\alpha]$ and $\epsilon_i^{min}[\alpha]$ are the upper and lower threshold values of m_i, representing a dual-bound threshold. Every observed profile that is in the region between the two vectors is assumed nominal. Thus we certify a profile $\hat{\mathbf{p}}^k[\alpha]$ to be a *nominal profile* if

$$\epsilon^{min}[\alpha] \leq \hat{\mathbf{p}}^k[\alpha] \leq \epsilon^{max}[\alpha] \tag{4.5}$$

i.e., if $\epsilon_i^{min}[\alpha] \leq \hat{p}_i^k[\alpha] \leq \epsilon_i^{max}[\alpha]$ for every $1 \leq i \leq |M|$. The values of threshold vectors $\epsilon^{max}[\alpha]$ and $\epsilon^{min}[\alpha]$ are experimentally determined while the system is in *test mode*. Test mode here assumes a controlled environment in which the system runs normal and is closely observed while no fault occurs and no attacks on the system take place. In practice this means that, while in normal operation, the profiles are tracked over time to derive (or calculate) the desired threshold vectors. In the simplest case the threshold vectors can be the minimal and maximal observed values of each $\hat{p}_i^k[\alpha]$.

If one needs to tune the sensitivity of the thresholds, one can introduce weight functions w, defined per costate, to be multiplied with the threshold vectors. Then a nominal execution of module m_i is defined as

$$w\epsilon_i^{min}[\alpha] \leq \hat{p}_i^k[\alpha] \leq w'\epsilon_i^{max}[\alpha]. \tag{4.6}$$

An alternative representation of execution certification is based on the deviation of the observed profile from the mean that has to be within the threshold vectors. Then an execution of module m_i is nominal if

$$|\hat{p}_i^k[\alpha] - \overline{p}| \leq \epsilon_i^{max}[\alpha] - \epsilon_i^{min}[\alpha]. \tag{4.7}$$

4.6.2 Certified Executions with Behavior Sets

Now we consider that there may be more than one nominal behavior of the system. For example if during one operational epoch the system may exhibit one of several known behaviors, the nominal behavior is not unique anymore. Assume we are processing data files that have several known sizes, e.g., size a or b. Then one would expect the observed profiles of the executions in both cases to be different, yet both are nominal. If on the other hand a file is corrupted and its size deviates largely from size a or b, then one would like to detect this, e.g., by observing a profile vastly different from that of nominal executions. Thus we need to extend the notation of certified execution to consider behavior sets, introduced in subsection 4.5.1. Recall the notational convention of changing lower case letters to upper case letters when behavior sets are used.

When using behavior sets the elements of threshold vectors are sets. Let $E^{min}[\alpha]$ and $E^{max}[\alpha]$ denote the behavior threshold vectors, i.e.
$E^{min}[\alpha] = (E_1^{min}[\alpha], E_2^{min}[\alpha], ..., E_{|M|}^{min}[\alpha])$ and
$E^{max}[\alpha] = (E_1^{max}[\alpha], E_2^{max}[\alpha], ..., E_{|M|}^{max}[\alpha])$. For a module m_i there will be at least one threshold value $\epsilon_{i,r}^{min}[\alpha] \in E_i^{min}[\alpha]$ and $\epsilon_{i,s}^{max}[\alpha] \in E_i^{max}[\alpha]$. The second subscript refers to an element number in the behavior set, e.g., element r and s.

The execution of module m_i is nominal if for some $\hat{p}_{i,t}^k[\alpha] \in \hat{P}_i^k[\alpha]$ and some $\epsilon_{i,r}^{min}[\alpha] \in E_i^{min}[\alpha]$ and $\epsilon_{i,s}^{max}[\alpha] \in E_i^{max}[\alpha]$

$$\epsilon_{i,r}^{min}[\alpha] \leq \hat{\mathbf{p}}_{i,t}^k[\alpha] \leq \epsilon_{i,s}^{max}[\alpha]. \tag{4.8}$$

4.7 System Operation & Contingency Management

In this section we describe the system as it operates and present data that was collected as part of the system mission and its monitoring over months of operation, especially the winter months of 2012.

4.7.1 System State Space and Transition Violations

As the system operates in the field it goes through state transitions, which are monitored by the *Operation Monitoring and Contingency Management System* (shown in Figure 4.2), using the three monitoring approaches introduced in the beginning of Section 4.6.

The system state space is, in general, complex, as it is induced by the state space of operations, functionalities, and modules. In our application, using the Mapping Simplification Assumption it reduces to the operation state space and module state space. The module state diagram is shown in Figure 4.6. A total of 25 system states can be observed, i.e., $S_0, ...S_{24}$. For example, state S_0 indicates that the software is in function *main*(). There are two types of transitions between two states S_i and S_j, represented by arcs between states. The solidly drawn arcs represent *calls*, whereas dotted arcs represent *returns*, e.g., a call will cause the transition from S_1 to S_2, whereas a return will cause the transition from S_2 back to S_1. The state machine is derived directly from the software call graph during compile time of the program.

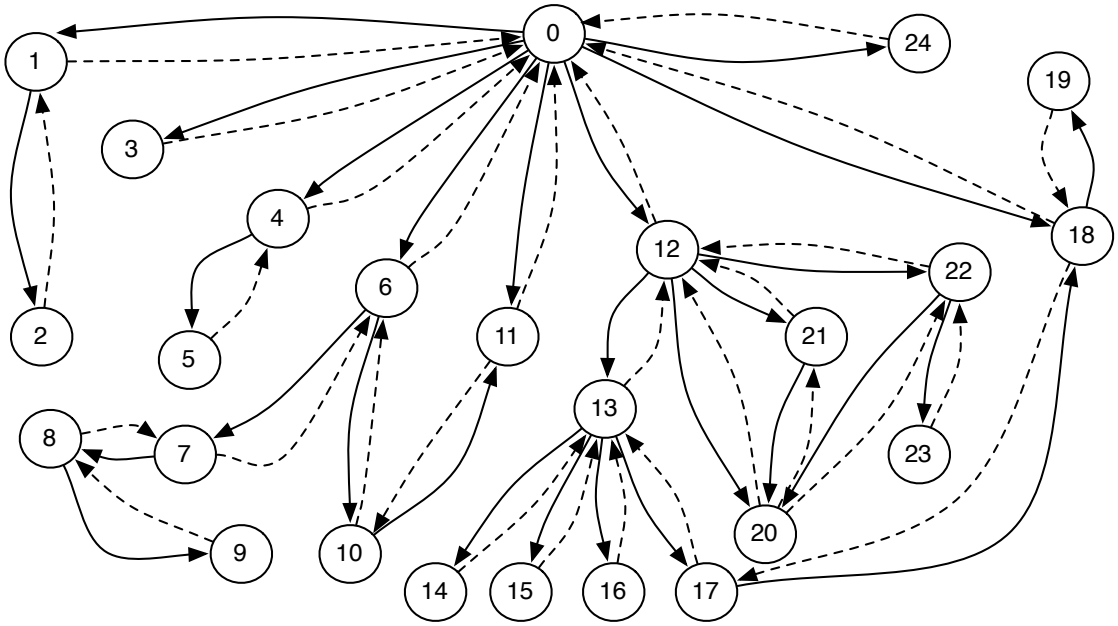

Figure 4.6: System Module State Machine

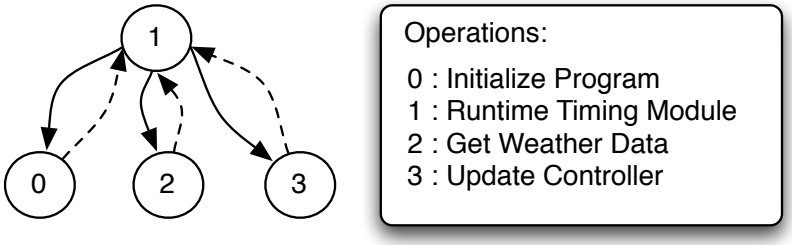

Figure 4.7: System Operation State Machine

The operation state space, shown in Figure 4.7, is much simpler and has only four states corresponding to the operations indicated in the figure. The combined system state space is thus the state space induced by the mappings in Figure 4.5, considering intradependencies and interdependencies. With respect to Figure 4.5, the diagrams in Figure 4.6 and Figure 4.7 represent the module and operation intradependency graphs respectively.

As the system software executes in costatements the transitions are verified to detect interdependency and intradependency violations described in subsection 4.5.2. Any violation of the system state transitions indicates a serious problem. For example, an intradependence violation of the module states implies that the system is calling modules that it should not be calling or it is returning to modules other than intended, e.g., as the result of a buffer overflow. Another example is if a module is called under an operation that should utilize it, or a module returns to another module that is not operating under the same operation. Both cases can be deducted from the interdependency mappings. Upon detection of any dependence violation the contingency management system initiates fail-safe mode and issues a notification about the nature of the violation.

4.7.2 Application Control

The operations of the application control system was depicted in Figure 4.7. The actions of the actual control sub-system, which is running in the costatement that implements the real-time traffic control application, are shown in the simplified flow chart in Figure 4.8.

The operation epoch is 15 minutes, which is the fixed time interval at which the real-time weather condition data is available from the Clarus server for specific subscriptions (indicated by subscription numbers). First the Rabbit determines the time and composes the Uniform Resource Locator (URL) that contains the comma separated values, a *csv* file. It then uses a recovery block strategy [65] to get the data from a set of data base servers. In our current implementation this is a LCS and the actual Clarus server shown in Figure 4.1, thus implementing a dual redundant system. Time redundancy is implemented as follows. First the LCS is queried and if the data cannot be retrieved within a certain amount of retries, then the Clarus server is tried. If it fails to provide the data after a certain amount of retries, the Contingency Management System initiates *fail-safe mode*, which is a forward recovery

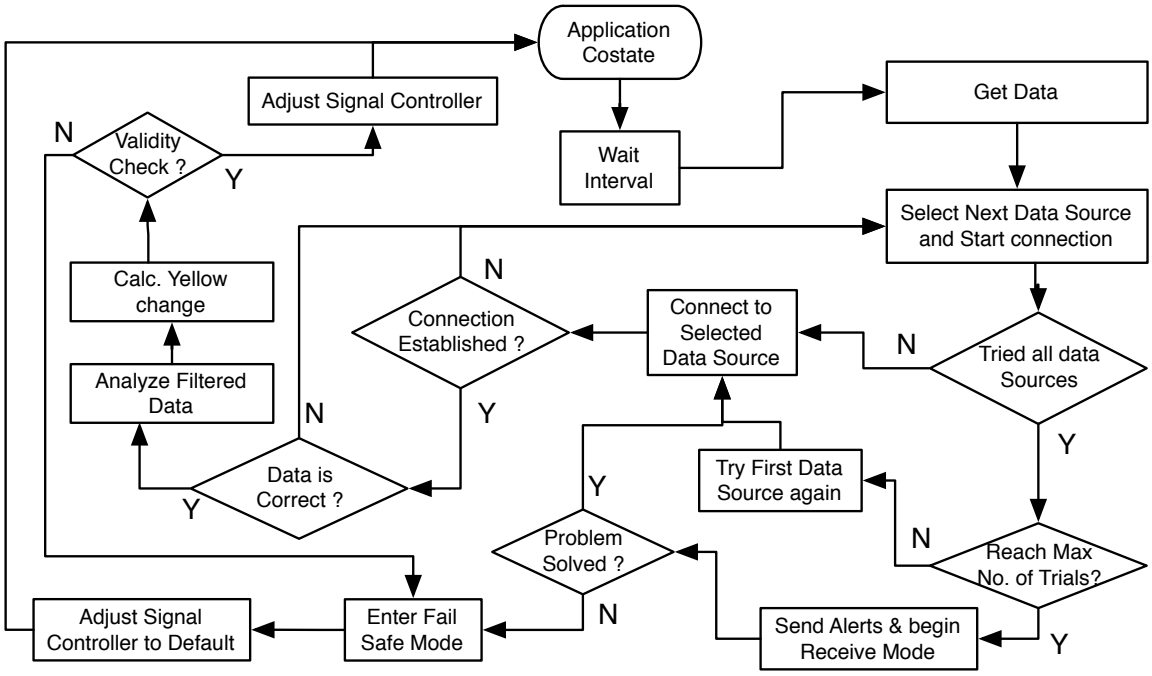

Figure 4.8: Flowchart of Application Control Costatement

mechanism bringing the system into a desired default state. Once entering fail-safe mode the system attempts to reestablish normal operation again.

If data acquisition was successful via one of the alternatives, then the traffic controller computes the adjustments that reflect the environment parameters and adjusts the signal controller accordingly.

Certain assumptions must be made about the quality of the data supplied by the Clarus server. We simply assume that the data received is correct since Clarus is designed using data quality checking algorithms [64]. On the other hand, the computed signal adjustment values, as computed by the Rabbit, are not assumed to be correct, as a fault may have occurred during computation. Therefore detection of anomalies through data sensor analysis, the second monitoring approach in Section 4.6, is implemented. It tests the computed signal changes for range violations from what was expected. If a violation is detected the contingency management system enters fail-safe mode. There are many other checking mechanisms implied in the flowchart of Figure 4.8, including reaction to network connection problems, data corruption, loss of time synchronization, e.g., after reboot as the result of

power failure, inability of finding valid Clarus data subscriptions, changing the LCS internet address, etc. Some of these issues require the contingency management system to enter a *receive mode*, in which configuration information, e.g., the IP of a new LCS or Clarus subscription number, is communicated to the system.

4.7.3 Application Control Performance

The system is installed in Northern Idaho and has been observed over the most interesting period, which are the winter months, as adverse weather conditions are common. The adjustments of the yellow time of the traffic signals has been observed over several months. As the environment conditions worsen the yellow time is increased to improve safety, e.g., during ice or snow a longer yellow period allows more time to safely clear the intersection. The adjustment value computed by the Rabbit is communicated to the traffic controller, which in turn makes changes to a default value according to the percentage given by the adjustment value.

The yellow adjustment values for 53 interesting operational epochs during November and December of 2012 are shown in Figure 4.9. The adjustments are in the range of 10% to 50% of the default value. However, how does one know if these values are correct? Figure 4.10 shows the data of the data sensor analysis and it indeed confirms the adjustments to be reasonable. The figure shows the values *Surface Status*, *Surface Temperature*, which are Clarus parameters, *Weather Conditions*, and the adjustment values called *Yellow*. The first observation is that value *Yellow* follows the weather conditions. Next, looking at the surface temperature one can see that *Yellow* is increased as the temperature decreases. An alternate reference is the *Surface Status* (*ST*), a Clarus value that shown in the lowest graph of the figure. Larger values of ST indicate deteriorating conditions, whereas conditions improve as ST become smaller. The values for ST are in the interval $[1, 14]$. However, if one carefully examines the graph, one can see that there are five cases where $ST = 0$, which represents an error condition, i.e., the Clarus file was not available. The contingency management system recognized the fault and adjusted it to the most recent value, as can be observed in Figure 4.9.

In addition to the data sensor analysis we also have exception triggers. Figure 4.11 shows five trigger and the count, again based on the 53 observed operational epochs, indicates how

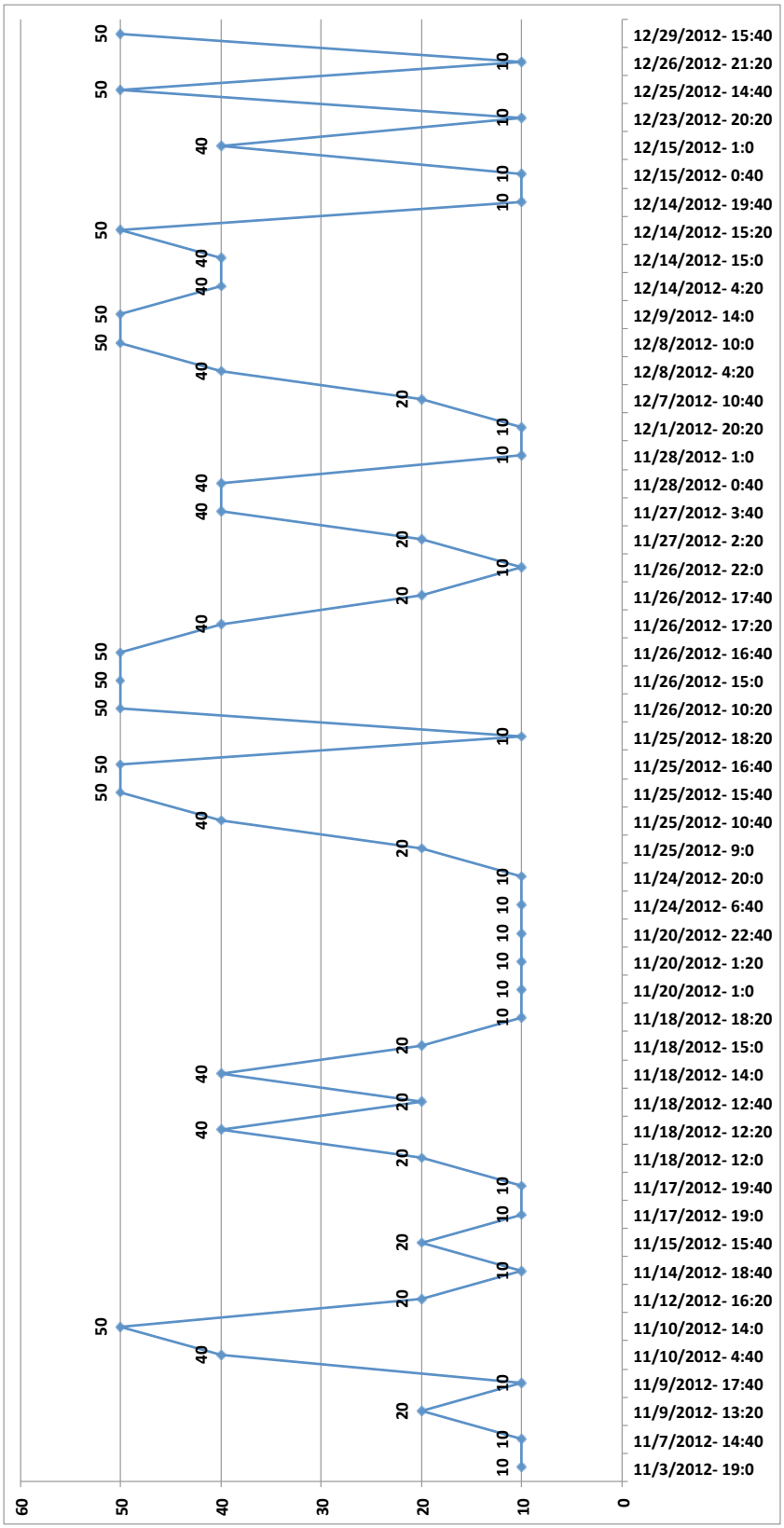

Figure 4.9: Adjustments of Yellow Period in % Over Winter Months

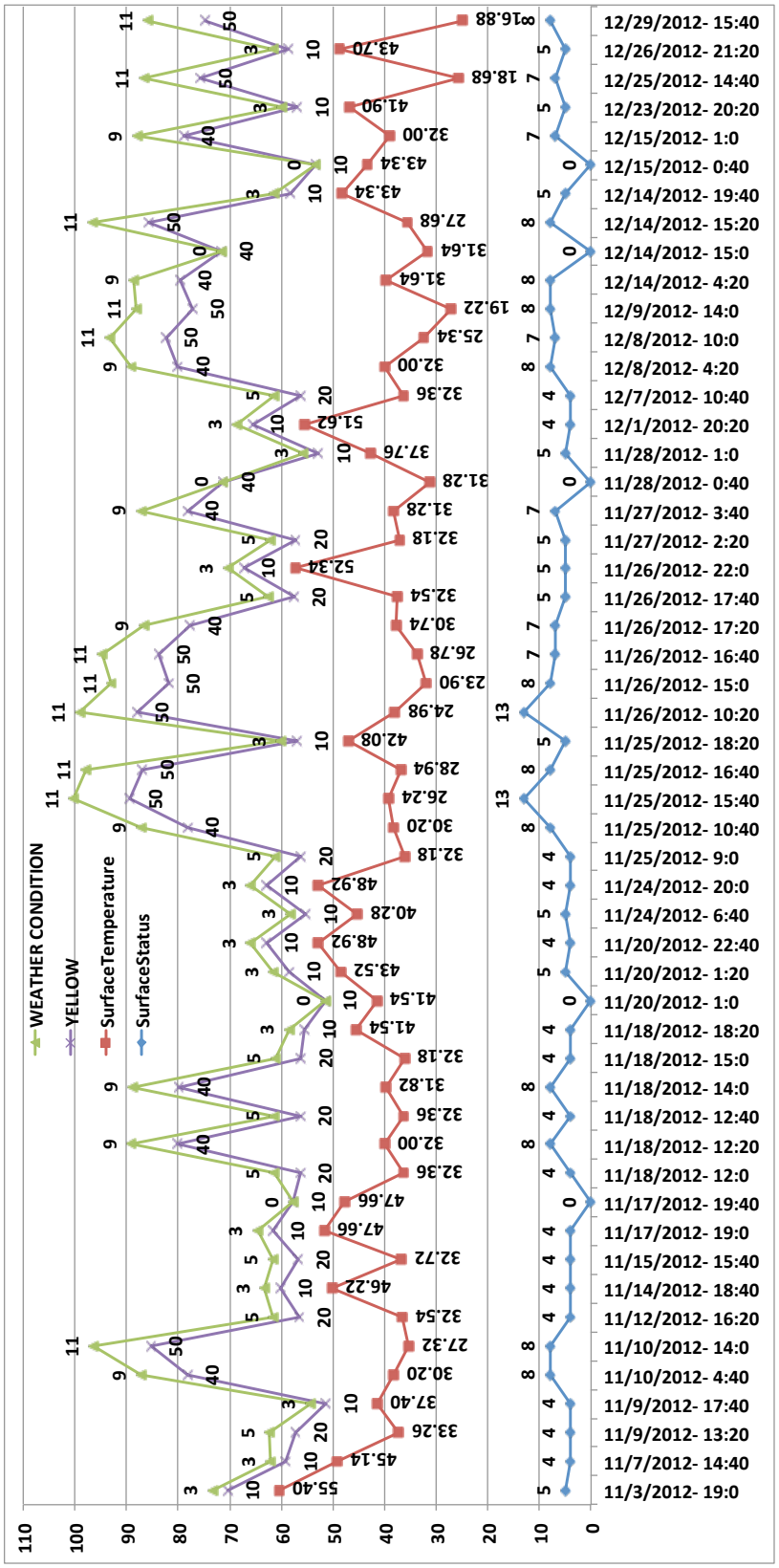

Figure 4.10: Yellow Adjustments During Winter Months Related to Weather Conditions

often exceptions were triggered. For example only two exceptions were never triggered. A trigger in this case implies that data needed for the computation of the yellow adjustments were missing in the data files. In fact, two of the five triggers were fired 47 times, i.e., out of 53 epochs, 47 did not include the specific sensor data, and 5 times the entire Clarus files were empty. However, the algorithm engine computing the adjustment could adapt to these

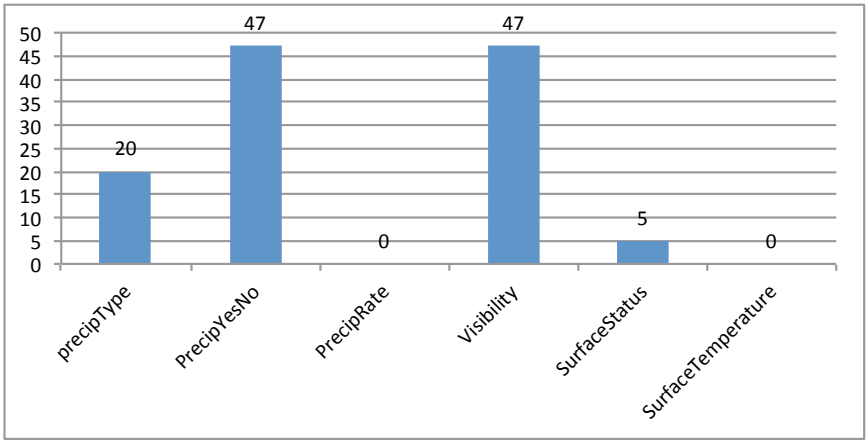

Figure 4.11: Exception Triggers

scenarios since even in the absence of data the system could produce reasonable adjustment values. The data in the figure was actually observed as the consequence of changes in the data supplied from the ESS or Clarus. Since we have no control over what ESS sensor data may be missing, perhaps due to defects or product changes, the algorithm computing the adjustments to the application must be able to tolerate the missing data, which represent an omission fault. The exception triggers are used to verify the adjustment values, e.g., as seen in Figure 4.9, and provide adaptation.

4.7.4 Certified Executions for Resilient Operation

The third monitoring approach is check for off-nominal executions. Due to the fact that costatements are non-preemptive, the observed modular profiles of one costate is not affected by executions of another costate. This however only holds in the absence of parallelism. Due to the fact that the embedded system has a single processor, there cannot be true parallelism

that could violate the assumptions that at any given time the system is in only one state. Thus profiles based on costates represent mutually exclusive measurements.

The threshold vectors that are the basis for determining certified executions are established during normal system operation. This is consistent with the general approach discussed in [66], where execution sequences based on the specifications were used to determine the Markov chain and the state probabilities. However, in our case the Markov chain does not need to be explicitly derived, since it is the static module call graph that is generated automatically at compile time.

Figure 4.12 shows the profile related to 14 key modules out of a total of 25 modules. All but module m_{23} behave consistent in that their minimum, average, and maximum frequencies are equal. Only m_{23}, a module that filters Clarus data, experienced variation. Further examination of the behavior of m_{23} over time is shown in Figure 4.14, where the 52 operational epochs of Figures 4.9 and 4.10 are used. The counts of invocations of m_{23} is indicated, together with the minimum and maximum counts that would typically be used as the basis for the threshold function. However, m_{23} actually has two behaviors, i.e., one for non-empty files with no sensor data and another for standard file size. This causes the minimum and maximum to drift, as they are monotonically non-increasing and non-decreasing functions. As a result it is difficult to define effective thresholds for detection of off-nominal executions. This was solved by using a behavior set of size two, as shown in Figure 4.15. One behavior threshold reflects the small file size and another normal file size. The figure also identifies four readings that are off-nominal, by far overreaching the other readings. These files were abnormal data files of much larger size and are treated as if they were data falsification attempts.

4.7.5 Reliability Considerations

It is important to address how the addition of the real-time weather control application affects the reliability of the target application, i.e., the traffic control system. The reliability of the traffic control system is not affected by the addition of the embedded system since none of the components of the original traffic signal control system are modified. The embedded system implements only *added value*, but not basic functionality. In fact, as mentioned

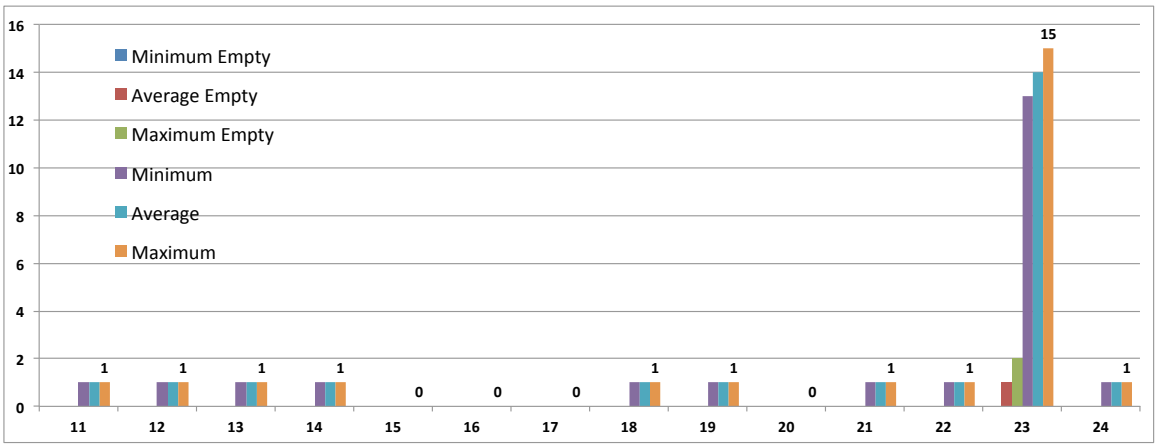

Figure 4.12: Profiles of Key Modules (Module Numbers on x-axis) with 2 Nominal Behaviors.

before, the traffic controller is NTCIP compliant. Thus action movie scenarios like an "all-green intersections" or "split second yellow timing" causing accidents are not possible (with or without the embedded system). Any attempt to assign parameters that violated NTCP compliance are simply ignored by the traffic controller, which we verified during testing of the embedded system. The only physical connection with the existing infrastructure is via the connection to the switch, as indicated in Figure 4.1. The embedded

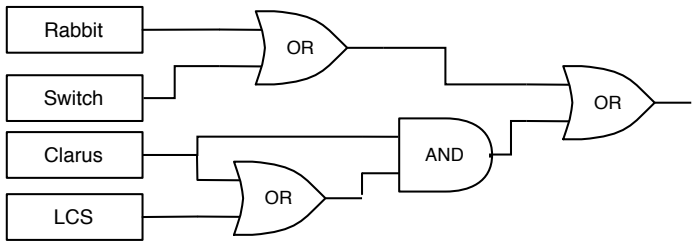

Figure 4.13: Simplified Fault Tree of System

system and its other components, as shown in Figure 4.1, can be modeled by the simple Fault Tree shown in Figure 4.13. The failure scenario is that the embedded system fails to provide added functionality. It should be noted that if the Clarus server (or the network to it) fails, then the LCS server is of no use anymore either, since it is only a mirror site, whereas if the LCS fails, the Clarus server can provide services.

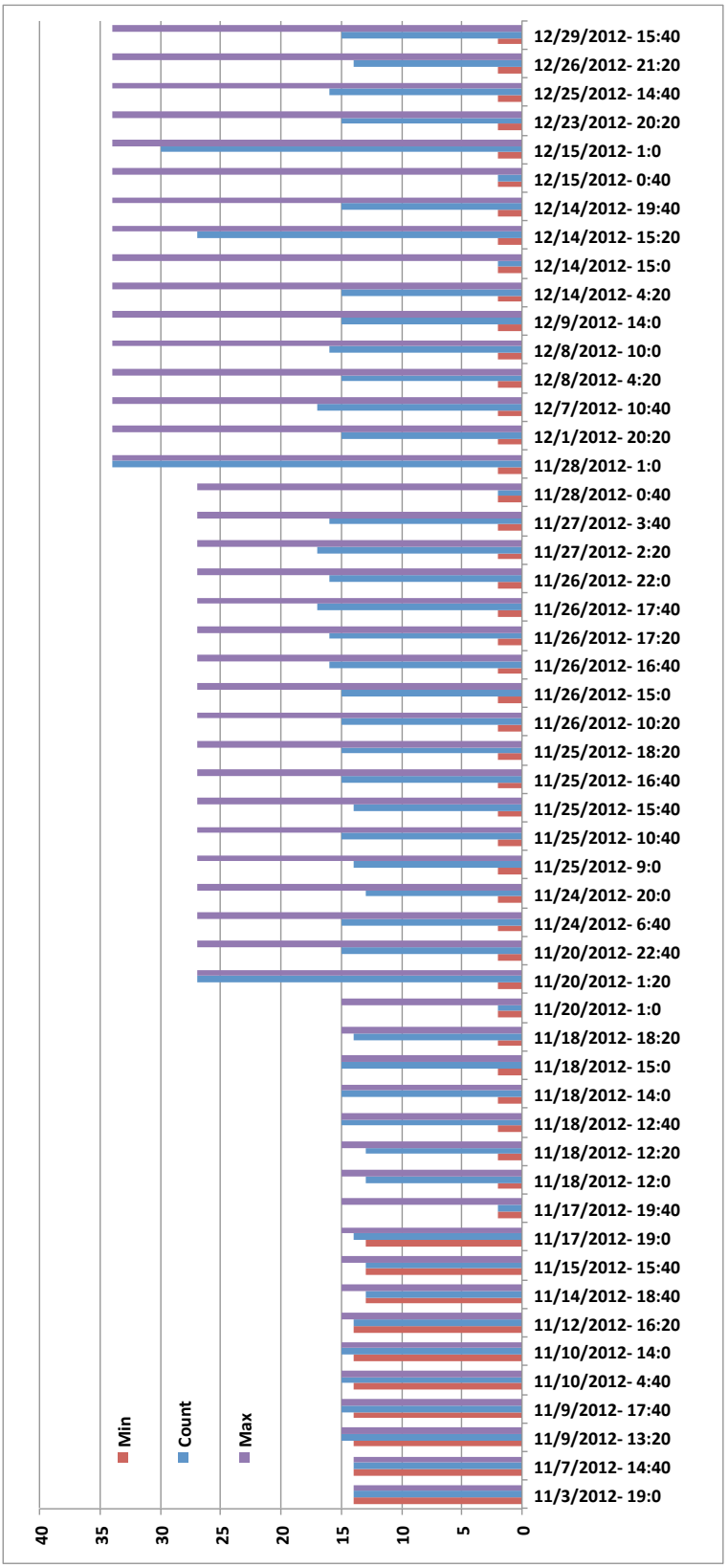

Figure 4.14: Profiles of Module m_{23} with Behavior Set Size Equal to 1 over Operational Epochs.

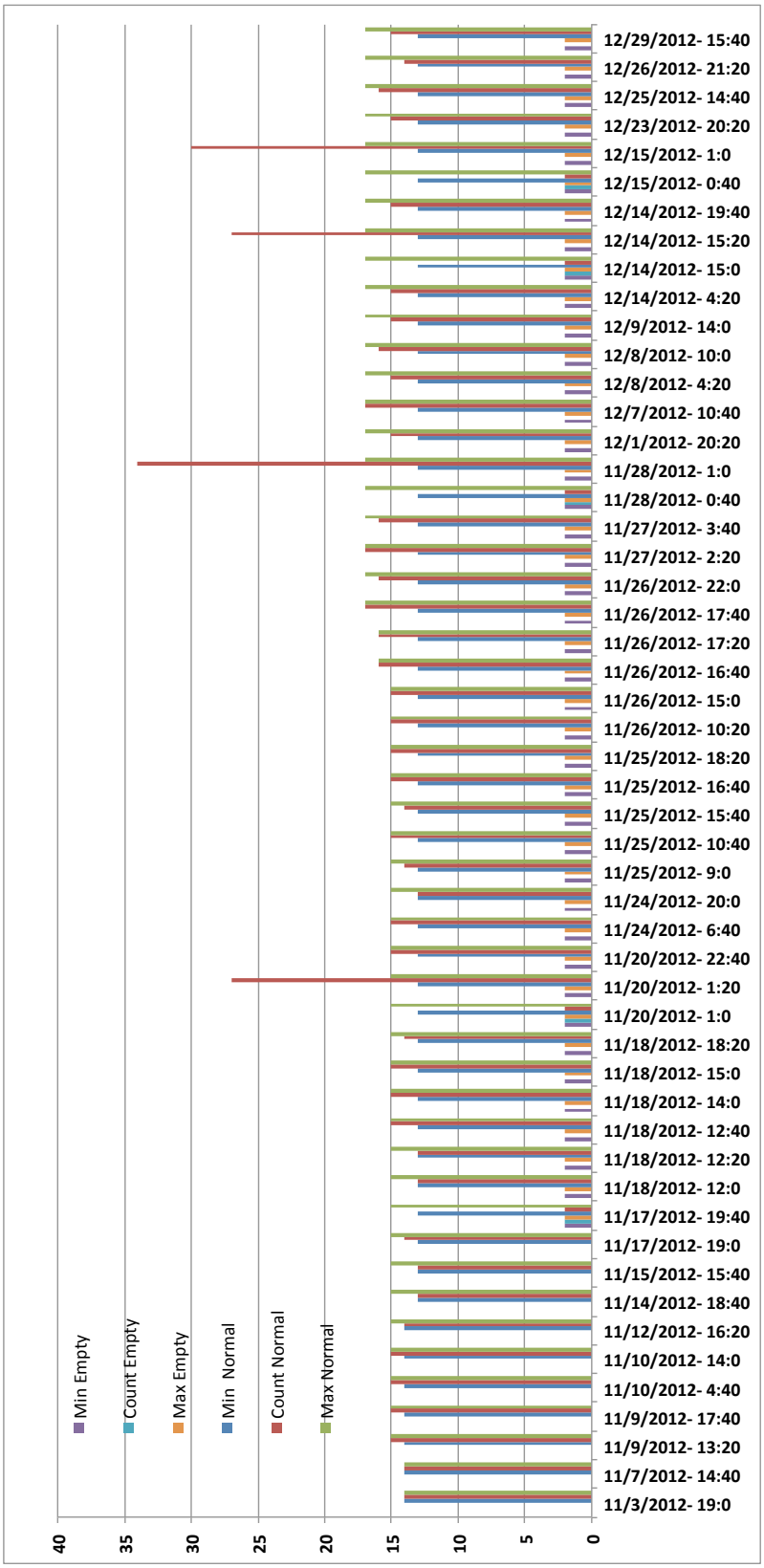

Figure 4.15: Profiles of Module m_{23} with Behavior Set Size Equal to 2 over Operational Epochs.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion and Future Work

The principle of design for survivability has been presented in this dissertation. Two types of ITS critical safety application domains have been discussed. The first application domain involved the Connected Vehicles Infrastructure. The second application domain was the Weather Responsive System Infrastructure. In order to preserve the survivability for the two application domains, we used as a starting point the vague notion that a system has to be able to tolerate diverse faults.

In the Connected Vehicles Infrastructure, a new approach to increase survivability of safety applications using DSRC has been presented. The concept of dissimilarity of communication mechanisms has been used to increase resilience against interference as the result of natural phenomena and malicious act. The case of constant and random jamming and their impact on safety applications were investigated under consideration of the data rates used. The dual or triple redundant mechanisms presented do not introduce concepts that deviate from existing standards. They only use message exchanges that rely on different message types using channels that are maximally spaced in the spectrum. The information in the standards relevant to the suggested mechanisms were presented to support and justify the decisions taken. The redundancy schemes introduced overcome the impact of jamming, assuming that the jammer's capabilities are limited to the technical specifications of the vehicle's OBU transmission power model. It was shown how different data rates were affected by jamming. In fact the dual-redundant scheme using channels CH172 and CH178 can provide sufficient FCW application reliability in the presence of jamming. This is the case for either using 3Mbps or 6Mbps communication. For constant jamming it was observed that the control channel dominates the reliability due to its high power. As a result it allowed to use higher data rates, up to 6Mbps, on that channel. This in turn would allow the usage of higher data rates in other channels, as the control channel reliability has greatest effect on the application reliability. In triple redundancy we suggest using channel CH184 for data rates

no higher than 3Mbps for DSRC safety applications. Furthermore, given the results for the unreliability of 12Mbps communication, we conclude that the use of this data rate is also not advisable for DSRC safety applications that may be exposed to jamming attacks. For random jamming it could be shown that reliability is highly dependent on the sleeping ratios. For sleeping ratios above 25% random jamming has no effect on reliability for all data rates. However, for lower sleeping rates, random jamming causes reliability characteristics closer to that of constant jamming. We acknowledge that using redundancy imposes extra overhead/usage of the dedicated limited bandwidth, which is intended to be used by multiple DSRC applications. However, our main concern is to give high priority consideration to safety applications over any other type of application.

In the Weather Responsive System Infrastructure, any fault type could be detected with high probability. Therefore we can treat any type of faults as benign fault. The architecture of a resilient control application operating in a critical infrastructure has been presented. The theoretical basis for effective run-time monitoring was given and the system has been observed over time. The experience gained so far indicates that the three monitoring approaches that were introduced allow for adaptation. As the fault and attack vector that the system is exposed to is unknown, time and testing using fault injection will ultimately be the judge for its effectiveness. Further operation in the field will allow us to study the sensitivity of the certification parameters that implement the thresholds of nominal executions. However, it should be noted that this application, due to its NTCIP compliance cannot compromise safety. If it enters fail-safe mode due to off-nominal executions due to unknown origin, it only ceases to supply the added value. If, against all expectations, it were to completely fail and behave pathological it could not overwrite settings and violate safety margins.

Current efforts focus on field implementations and analysis of the overhead of the mechanisms, in addition to comparing the experimental with the analytical results.

REFERENCES

[1] Amendment of the Commission's Rules Regarding Dedicated Short-Range Communication Services in the 5.850-5.925 GHz Band (5.9 GHz Band), Federal Communications Commission FCC 03-324, 2004.

[2] A. Krings, *Survivable Systems*, in Information Assurance: Dependability and Security in Networked Systems. Morgan Kaufmann Publishers, Yi Qian, James Joshi, David Tipper, and Prashant Krishnamurthy Editors, in press, 2008, ch. 5, pp. 113-146.

[3] A. Krings et al., *A Measurement-based Design and Evaluation Methodology for Embedded Control Systems*, in Proc. of the 7th Annu. Workshop on Cyber Security and Information Intelligence Research (CSIIRW'11), Oak Ridge National Laboratory, October 12 - 14, 2011 ©ACM. doi: 10.1145/2179298.2179335.

[4] A. Krings, A. Serageldin and A. Abdel-Rahim, *A Prototype for a Real-Time Weather Responsive System*, in the 15th Int. Conf. on Intelligent Transportation Systems (ITSC), Anchorage, Alaska, September 16-19, 2012, pp. 1465-1470.

[5] A. Serageldin, A. Krings, and A. Abdel-Rahim, *A Survivable Critical Infrastructure Control Application*, in Proc. of the 8th Annu. Cyber Security and Information Intelligence Research Workshop (CSIIRW'13), Oak Ridge National Laboratory, January 8 - 10, 2013 ©ACM. doi: 10.1145/2459976.2460015.

[6] V. Balogun and A. Krings, *On The Impact of Jamming Attacks on Cooperative Spectrum Sensing in Cognitive Radio Networks*, in Proc. of the 8th Annu. Cyber Security and Information Intelligence Research Workshop (CSIIRW'13), Oak Ridge National Laboratory, January 8 - 10, 2013 ©ACM. doi: 10.1145/2459976.2460011.

[7] Y. Liu and K.S. Trivedi, *A General Framework for Network Survivability Quantification*, in Proc. 12th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer

and Communication Systems (MMB) together with 3rd Polish-German Teletraffic Symp. (PGTS), September 2004.

[8] H.Frank, *Survivability Analysis of Command and Control Communications Networks – part I, II*, in IEEE Transactions on Communications, May 1974, Vol. 22, No. 5, pp. 589-605.

[9] IEEE Standard for Wireless Access in Vehicular Environments – Security Services for Applications and Management Messages, IEEE Standard 1609.2^{TM}, 2013.

[10] R. J. Ellison et al., *Survivable Network Systems: An Emerging Discipline*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Technical Report CMU/SEI-97-TR-013, November 1997, Revised: May 1999.

[11] A. Avizienis et al., *Basic Concepts and Taxonomy of Dependable and Secure Computing*, in IEEE Transactions on Dependable and Secure Computing, Jan. - March 2004, Vol. 1, No. 1, pp. 11-33.

[12] T1A1.2 Working Group on Network Survivability Performance, *Technical Report on Enhanced Network Survivability Performance*, Technical report No. 68, February 2001.

[13] V. Nelson, *Fault-Tolerant Computing: Fundamental Concepts*, in IEEE Computer, July 1990, Vol. 23, No. 7, pp. 19-25.

[14] W. B. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*, Addison-Wesley Publishing Company, New York, 1989.

[15] M. Pease, R.Shostak, and L. Lamport, *Reaching Agreement in the Presence of Faults*, in Journal of the ACM (JACM), April 1980, Vol. 27, No. 2, pp. 228-234.

[16] L. Lamport, R.Shostak, and M. Pease, *The Byzantine Generals Problem*, in ACM Transactions on Programming Languages and Systems (TOPLAS), July 1982, Vol. 4, No. 3, pp. 382-401.

[17] F.J. Meyer and D.K. Pradhan, *Consensus with Dual Failure Modes*, in IEEE Transactions on Parallel and Distributed Systems., Apr. 1991, Vol. 2, No. 2, pp. 214-222.

[18] P. Thambidurai, and Y.K. Park, *Interactive Consistency with Multiple Failure Modes*, in Proc. 7th Symposium on Reliable Distributed Systems, Columbus, OH, Oct. 1988, pp. 93-100.

[19] M.H. Azadmanesh, and R.M. Kieckhafer, *Exploiting Omissive Faults in Synchronous Approximate Agreement*, in IEEE Transactions on Computers, Oct. 2000, Vol. 49, No. 10, pp. 1031-1042.

[20] P. Lincoln and J. Rushby, *The Formal Verification of an Algorithm for Interactive Consistency under a Hybrid Fault Model*, in Proc. 5th Int'l Conf. Computer Aided Verification, 1993, pp. 292-304, .

[21] D. Curiac et al., *Redundancy and its Applications in Wireless Sensor Networks: A Survey*, in WSEAS Transaction on Computer, Apr. 2009, Vol. 8, No. 4, pp. 705-714

[22] A. Avizienis, *The Methodology of N-version Programming*, in Software Fault Tolerance, edited by M. Lyu, John Wiley & Sons, 1995, Ch. 2, pp. 23-42.

[23] L. Tan, and A. Krings, *An Adaptive N-variant Software Architecture for Multi-Core Platforms: Models and Performance Analysis*, in The 11th International Conference on Computational Science and its Applications (ICCSA 2011), Santander, Spain, June 20-23, 2011, in Lecture Notes on Computer Science, LNCS 6783, Springer Verlag, pp. 490-505.

[24] L. Tan, and A. Krings, *A Hierarchical Formal Framework for Adaptive N-variant Programs in Multi-core Systems*, in IEEE 30th International Conference on Distributed Computing Systems Workshops (ICDCSW), Genova, 21-25 June 2010, pp. 7-12.

[25] C.M. Jeffery and J.O. Figueiredo, *Towards Byzantine Fault Tolerance in Many- core Computing Platforms*, in IEEE 13th Pacific Rim International Symposium on Dependable Computing, Melbourne, Qld, 17-19 Dec. 2007, pp. 256-259.

[26] B. Cox et.al., *N-Variant Systems: A Secretless Framework for Security through Diversity*, in USENIX-SS'06 Proceedings of the 15th conference on USENIX Security Symposium, Vancouver, BC, Aug. 2006, Vol. 15, No. 9, pp. 105-120.

[27] A. Nguyen-Tuong et.al., *Security through Redundant Data Diversity*, in IEEE International Conference on Dependable Systems and Networks (DSN) With FTCS and DCC, Anchorage, AK, 24-27 June 2008, pp. 187-196.

[28] S. Chamotra et.al., *Data Diversity of a Distributed Honey Net Based Malware Collection System*, in IEEE International Conference on Emerging Trends in Networks and Computer Communications (ETNCC), Udaipur, 22-24 April 2011, pp. 125-129.

[29] C. E. Shannon, *A Mathematical Theory of Communication*, in Bell System Technical Journal, July 1948 ©American Telephone and Telegraph Co., Vol. 27, pp. 379-423.

[30] E. H. Simpson, *Measurement of Diversity*, in Nature Journal, 30 April 1949, Vol. 163, pp. 688-688, doi: 10.1038/163688a0.

[31] A. Renyi, *On Measures of Entropy and Information*, in Fourth Berkeley Symposium on Mathematical Statistics and Probability, 1961, Vol. 1, pp. 547-561.

[32] F. Ahmed-Zaid et.al., Vehicle Safety Communications-Applications (VSC-A) Final Report, Crash Avoidance Metrics Partnership on behalf of the Vehicle Safety Communications 2 Consortium, Farmington Hills, MI, Tech. Rep. DOT HS 811 492 A, Sponsored by NHTSA, RITA, U.S. Department of Transportation, September 2011.

[33] A. Serageldin , H. Alturkostani, and A. Krings, *On the Reliability of DSRC Safety Applications: A Case of Jamming*, in IEEE International Conference on Connected Vehicles and Expo (ICCVE), Las Vegas, NV, 2-6 Dec. 2013, pp. 501 - 506.

[34] A. Serageldin, and A. Krings, *The Impact of Dissimilarity and Redundancy on the Reliability of DSRC Safety Applications*, in IEEE 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Victoria, BC, Canada, 13-16 May 2014, pp. 417 - 424.

[35] A. Serageldin, and A. Krings, *The Impact of Redundancy on DSRC Safety Application Reliability under Different Data Rates*, in IEEE 6th International Conference on New Technologies, Mobility and Security, (NTMS), Dubai, United Arab Emirates, March 30 - April 2, 2014, pp. 1-5.

[36] Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems - 5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications, ASTM E2213-03, 2010.

[37] C. Makaya and S. Pierre., *Emerging Wireless Networks: Concepts, Techniques, and Applications*, CRC Press, Taylor and Francis, New York, December 12, 2011.

[38] IEEE Guide for Wireless Access in Vehicular Environments (WAVE) – Architecture, IEEE Standard 1609.0^{TM}, 2013.

[39] IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments, IEEE Standard $802.11p^{TM}$, 2010.

[40] Crescenzo et.al., *Non-interactive Malicious Behavior Detection in Vehicular Networks*, in Proceedings of the IEEE International Conference on Vehicular Networking Conference (VNC), Jersey City, NJ, USA, 13-15 Dec. 2010, pp. 278-285.

[41] S.K. Harit, G. Singh, and N. Tyagi, *Fox-Hole Model for Data-centric Misbehaviour Detection in VANETs*, in Third International Conference on Computer and Communication Technology (ICCCT), Allahabad, India, 23-25 Nov. 2012, pp. 271-277.

[42] O. Abumansoor, and A. Boukerche, *A Secure Cooperative Approach for Nonline-of-Sight Location Verification in VANET*, in IEEE Transactions on Vehicular Technology, Jan. 2012, Vol. 61, No. 1, pp. 275-285.

[43] L. C. Tung, and M. Gerla, *An Efficient Road-Based Directional Broadcast Protocol for Urban VANETs*, in Proceedings of the IEEE International Conference on Vehicular Networking Conference (VNC), Jersey City, NJ, USA, 13-15 Dec. 2010, pp. 9-16.

[44] IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Networking Services, IEEE Standard 1609.3^{TM}, 2010.

[45] IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-Channel Operation, IEEE Standard 1609.4^{TM}, 2010.

[46] Dedicated Short Range Communications (DSRC) Message Set Dictionary, Society of Automotive Engineers, DSRC Committee, SAE J2735, November 2009.

[47] M. Maile, and L. Delgrossi, *Cooperative Intersection Collision Avoidance System for Violations (CICAS-V) for Avoidance of Violation-Based Intersection Crashes*, Mercedes-Benz Research and Development North America, Inc., Paper Number 09-0118, 2009.

[48] R. Sahner, S. Trivedi and A. Puliafito, *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*, Kluwer Academic Publishers, 1996.

[49] S. Zhanshan and A. Krings, *Multivariate Survival Analysis (I): Shared Frailty Approaches to Reliability and Dependence Modeling*, in Proc. IEEE Aerospace Conference, Big Sky, MT, 1-8 March 2008, pp. 1-21.

[50] W. Xu et.al., *The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks* in Proceedings of the 6th ACM International Symposium on Mobile ad hoc Networking and Computing, 2005, pp. 46-57.

[51] K. Pelechrinis, M. Iliofotou, and S.V. Krishnamurthy, *Denial of Service Attacks in Wireless Networks: The Case of Jammers*, in IEEE Communications Surveys & Tutorials, 2^{nd} Quarter 2011, Vol.13, No.2, pp. 245-257.

[52] O. Puñal, A. Aguiar, and J. Gross, *In VANETs we Trust?: Characterizing RF Jamming in Vehicular Networks*, in Proceedings of the ninth ACM International Workshop on Vehicular Inter-Networking, Systems, and Applications, 2012, pp. 83-92.

[53] G. Johansson and K. Rumar, *Drivers' Brake Reaction Times*, in Human Factors: The Journal of the Human Factors and Ergonomics Society, 1971, Vol. 13, No. 1, pp. 23-27.

[54] D. B. Fambro et.al., *Driver Perception-Brake Response in Stopping Sight Distance Situations*, in Transportation Research Record: Journal of the Transportation Research Board, 1998, Vol. 1628, No. 1, pp.1-7.

[55] B. Sklar, *Digital Communications, Fundamentals and Applications*, Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1988.

[56] C. Trabelsi, and A. Yongacoglu, *Effect of Bit-to-Bit Dependence on Packet Error Rate Using Asynchronous DC-CDMA for Mobile Packet Radio Networks*, in International Journal of Wireless Information Networks, 1995, Vol. 2, No. 3, pp. 149-163.

[57] The Clarus System: http://www.clarus-system.com/

[58] A. Krings, *Design for Survivability: A Tradeoff Space*, in Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research (CSIIRW'08): Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead, Oak Ridge National Laboratory, May 12-14, 2008 ©ACM. doi: 10.1145/1413140.1413154.

[59] J. Munson, A. Krings and R. Hiromoto, *The Architecture of a Reliable Software Monitoring System for Embedded Software Systems*, in Proceedings of the 5th International Topical Meeting on Nuclear Plant Instrumentation Controls, and Human Machine Interface Technology, 12-16 Nov 2006, Vol. 43, No. 47, pp. 765-774.

[60] V. Chandola, A. Banerjee, and V. Kumar, *Anomaly Detection: A Survey*, in ACM Computing Surveys (CSUR), July 2009, Vol. 41, No. 3, Article 15, 2009 ©ACM. doi: 10.1145/1541880.1541882.

[61] J. Allen et.al., *State of the Practice of Intrusion Detection Technologies*, Software Engineering Institute,Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-99-TR-028, 2000.

[62] S. Elbaum and J. Munson, *Intrusion Detection Through Dynamic Software Measurement*, in Proceedings of the 1st conference on Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, April 9-12, 1999, Vol. 1, pp. 41-50.

[63] A. Krings et.al., *A Two-Layer Approach to Survivability of Networked Computing Systems*, in Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, (SSGRR'2001), L'Aquila, Italy, Aug. 06-12 2001, pp. 1-12.

[64] M. Limber, S. Drobot, and T. Fowler, *Clarus Quality Checking Algorithm Documentation Report*, Final Report, RITA Intelligent Transportation Systems Joint Program Office, Technical Report FHWA-JPO-11-075, December 21, 2010.

[65] B. Randell, and J. Xu, *The Evolution of the Recovery Block Concept*, In Software Fault Tolerance, John Wiley & Sons Ltd, 1994, pp.1-22.

[66] J. A. Whittaker, and J.H. Poore, *Markov Analysis of Software Specifications*, in ACM Transactions on Software Engineering and Methodology (TOSEM), January 1993, Vol.2, No.1, pp. 93-106.