ABSTRACT

Title of dissertation: DEVELOPMENT AND EVALUATION OF METHODOLOGIES FOR VULNERABILITY ANALYSIS OF AD-HOC ROUTING PROTOCOLS

Shah-An Yang
Doctor of Philosophy, 2007

Dissertation directed by: Professor John S. Baras
Department of Electrical
and Computer Engineering

This thesis presents a number methodologies for computer assisted vulnerability analysis of routing protocols in ad-hoc networks towards the goal of automating the process of finding vulnerabilities (possible attacks) on such network routing protocols and correcting the protocols. The methodologies developed are (each) based on a different representation (model) of the routing protocol, which model predicated the quantitative methods and algorithms used. Each methodology is evaluated with respect to effectiveness feasibility and possibility of application to realistically sized networks. The first methodology studied is based on formal models of the protocols and associated symbolic partially ordered model checkers. Using this methodology, a simple attack in unsecured AODV is demonstrated. An extension of the Strands model is developed which is suitable for such routing protocols. The second methodology is based on timed-probabilistic formal models which is necessary due to the probabilistic nature of ad-hoc routing protocols. This second methodolgy

uses natural extensions of the first one. A nondeterministic-timing model based on partially ordered events is considered for application towards the model checking problem. Determining probabilities within this structure requires the calculation of the volume of a particular type of convex volume, which is known to be $\sharp P$-hard. A new algorithm is derived, exploiting the particular problem structure, that can be used to reduce the amount of time used to compute these quantities over conventional algorithms. We show that timed-probabilistic formal models can be linked to trace-based techniques by sampling methods, and conversely how execution traces can serve as starting points for formal exploration of the state space. We show that an approach combining both trace-based and formal methods can have faster convergence than either alone on a set of problems. However, the applicability of both of these techniques to ad-hoc network routing protocols is limited to small networks and relatively simple attacks. We provide evidence to this end. To address this limitation, a final technique employing only trace-based methods within an optimization framework is developed. In an application of this third methodology, it is shown that it can be used to evaluate the effects of a simple attack on OLSR. The result can be viewed (from a certain perspective) as an example of automatically discovering a new attack on the OLSR routing protocol.

Development and Evaluation of Methodologies for Vulnerability Analysis of Ad-Hoc Routing Protocols

by

Shah-An Yang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:
Professor John S. Baras, Chair/Advisor
Professor Virgil Gligor
Associate Professor Gang Qu
Associate Professor Richard J. La
Professor A. Udaya Shankar

ACKNOWLEDGMENTS

I would like to acknowledge my advisor, John S. Baras, for his numerous contributions to this manuscript.

I am grateful for the support of my research work and graduate studies through the following contracts and grants: Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-10-2-001 and U.S. Army Research Office under Award No. DAAD 190110494.

Table of Contents

List of Figures v

List of Abbreviations vii

1 Introduction 1
1.1 Outline . . . 1
1.2 Vulnerability Analysis of Ad-Hoc Routing Protocols : An Overview . 2
1.3 Main Contributions of this Dissertation . . . 4
1.4 Dissertation Outline . . . 6

2 A Formal Model for Ad Hoc Routing Protocol Vulnerability Analysis 8
2.1 Introduction . . . 8
2.2 Background . . . 9
2.2.1 Verification of Ad Hoc Routing Protocols . . . 10
2.2.2 Secure Ad Hoc Routing Protocols . . . 11
2.2.3 Securing Fixed Routing Protocols . . . 16
2.2.4 Formal Methods . . . 17
2.3 Criteria for Secure Routing . . . 27
2.4 Proposed extended strands model . . . 33
2.4.1 Partial Order Semantics and CTL . . . 33
2.4.2 Messages and events . . . 35
2.4.3 Role . . . 36
2.4.3.1 Logical Theories for Φ . . . 38
2.4.4 Protocols . . . 40
2.4.5 Strands . . . 40
2.4.6 Goal bindings . . . 41
2.4.7 Causal relations . . . 43
2.4.8 Constraint Program . . . 44
2.4.9 Semibundles . . . 45
2.4.10 Bundles . . . 45
2.4.11 Topologies . . . 46
2.4.12 Constraint program feasibility . . . 46
2.4.13 Algorithm for checking feasibility . . . 52
2.5 Search procedure . . . 61
2.5.1 Protocol specification language . . . 62
2.6 Implementation . . . 63
2.7 Results . . . 63
2.8 Discussion . . . 64

3 Probabilistic-Timed Formal Model 66
3.1 Introduction . . . 66
3.1.1 Related Work . . . 68
3.2 Review of OLSR . . . 69
3.2.1 Components of OLSR . . . 72
3.2.1.1 Local Topology Maintenance . . . 73
3.2.1.2 Information Dissemination . . . 74
3.3 Protocol Specification by Extended Executable Model . . . 74
3.3.1 Basic Formal Model . . . 75
3.3.2 Formal Timing Analysis . . . 77
3.3.3 Probabilistic Timing Analysis . . . 80
3.3.3.1 Preliminaries . . . 80
3.3.3.2 Deriving the Timing Model from the Formal Model . 85
3.3.3.3 Volume Computation . . . 88
3.3.3.4 Adapting Lasserre's Algorithm to Temporal Structure 90
3.3.3.5 Coupling Monte-Carlo Integration with Exact Computation . . . 103
3.4 Discussion . . . 109

4 Trace-Based Model within an Optimization Framework : OLSR Example 110
4.1 Introduction . . . 110
4.2 New Security Requirement . . . 111
4.2.1 Policy Iteration Formulation . . . 112
4.2.1.1 Cross-Entropy Optimization . . . 113
4.2.1.2 Multi-Layer Perceptrons . . . 120
4.3 Initial Experiments . . . 135
4.3.1 Experiment 1 . . . 136
4.3.2 Experiment 2 . . . 138
4.3.3 Experiment 3 . . . 141
4.4 Discussion . . . 142

5 Discussion and Future Work 144

Bibliography 147

List of Figures

2.1 Uncompromised path. 30

2.2 Example scenario. 64

3.1 OLSR Dependencies 70

3.2 Unit cube partitioned by coordinate order - 6 = 3!. 88

3.3 Lasserre's algorithm example in 2D. 90

3.4 Distance from point to line in constraint structure. 91

3.5 The constraint $x_j + d_{ji} \geq x_i$ forms a face of the polytope, but $x_i + d_{ij} \geq x_j$ is degenerate, being implied by $\phi + d_{0j} \geq x_j$ and $x_i + d_{i0} \geq \phi$. 93

3.6 Effect of assigning complementary distance $d_{ji} \leftarrow -d_{ij}$ 96

3.7 3D cross-eye stereographic illustration: view left image in right eye and right image in left eye. $d_{ik} < d_{ij} + d_{jk}$. Consequently, the constraint $x_i + d_{ik} \geq x_k$ dominates $x_j + d_{jk} \geq x_k$ within the plane $x_i + d_{ij} = x_j$. In this case, the constraint within the polytope face, $x_j + d_{ik} - d_{ij} \geq x_k$, corresponds with the original constraint $x_i + d_{ik} \geq x_k$. Note that $x_j + d_{jk} \geq x_k$ is redundant within the plane $x_i + d_{ij} = x_j$. 98

3.8 3D cross-eye stereographic illustration: view left image in right eye and right image in left eye. $x_i + d_{ik} \geq x_k$ is implied by $x_j + d_{jk} \geq x_k$ and $x_i + d_{ij} \geq x_j$. In this case, the polytope face in the plane $x_i + d_{ij} = x_j$ has $x_j + d_{jk} \geq x_k$ as a constraint. $x_i + d_{ik} \geq x_k$ is implied and therefore degenerate. 99

3.9 A representative 2D volume is shown. Its volume (area) can be decomposed into a main rectangular region subtracting two triangular regions. . 101

3.10 Execution times vs number of dimensions. No data point given for 12 dimensions in Vinci because it crashes. 102

3.11 Plot of the function $\sqrt{\frac{1-p}{p}}$. The relative error goes towards infinity as p approaches 0 and 0 as p approaches 1. 105

3.12 Shows the computation of the distance from a given point (a, b) to the constraint $x_i + d_{ij} \geq x_j$ along a chosen direction vector. $\cos(\theta)$ can be obtained by taking the dot product of the direction vector with the normal vector. 106

3.13 Shows an example of hit and run sampling with poor convergence to the distribution. This should be uniform over a rectangle. The reason for this is that the vast majority of random chords are nearly vertical in this rather severe aspect ratio. 107

3.14 Shows comparison of convergence of Monte-Carlo integration using plain rectangular rejection method and hybrid rejection method. The key difference is that the hybrid method results in a sampling region 1.4 times the size of the target region whereas the plain rectangular method results in a sampling region 7.5 times the size of the target. This example is for a 6-dimensional figure. Greater gains should be achievable in higher dimensions, but the hit-and-run sampling method works poorly there. . . 108

4.1 Simple network with a malicious node. 111

4.2 Elite sampling instability. Elite samples from a distribution that is a composite of these two will be dominated by the distribution with the lower mean but higher variance. 119

4.3 Multi-layer perceptron. 121

4.4 Transfer function $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$. 130

4.5 Fraction of time Disconnected vs Training Epoch. In this training scenario, the malicious node may ignore any packet. The random initial training data for Epoch 0 begins with a mean value of 0.2. 135

4.6 Fraction of time Disconnected vs Training Epoch. In this training scenario, the malicious node may only ignore `TC` packets and must accept all `HELLO` messages. 137

4.7 Probability of ignoring `HELLO` messages vs timeout values. Values are averaged over response to `HELLO` messages from nodes 0 and 2, and all possible combinations of values of `NEIGHBOR_TYPE`. 139

4.8 Fraction of time Disconnected vs Training Epoch. In this scenario, the malicious node may only ignore `HELLO` messages. 140

4.9 Fraction of time Disconnected vs Training Epoch. In this scenario, the malicious node may only ignore `HELLO` messages. This scenario differs from the previous in that different inputs used for the decision maker. . . 141

4.10 Mean Fraction of time Disconnected vs Training Epoch. In this scenario, the malicious node may only ignore `HELLO` messages. OLSR has been modified to contain countermeasures to attackers that mysteriously drop packets. 142

List of Abbreviations

AODV	Ad hoc On-Demand Distance Vector Routing
MPR	Multi Point Relay
OLSR	Optimized Link State Routing Protocol

Chapter 1

Introduction

1.1 Outline

This dissertation develops and evaluates methodologies for the computer assisted vulnerability analysis of ad-hoc network routing protocols. This introductory chapter aims to describe the ideas motivating this work and its main contributions. Subsequent chapters will address the details of the techniques that support the results outlined in this chapter.

Several approaches towards the problem of ad-hoc routing protocol attack vulnerability analysis are explored in this thesis. This is pursued towards the eventual goal of computer-aided design tools that will relieve protocol designers from time-consuming, error-prone tasks. This is important in view of the fact that there is a clear trend towards more complex protocols and the proliferation of protocols, while at the same time pressure to reduce the costs of analysis is increasing. A possible solution is to automate steps of the analysis and this thesis aims in that direction.

The following section presents some basic ideas on vulnerability analysis of ad-hoc routing protocols.

1.2 Vulnerability Analysis of Ad-Hoc Routing Protocols : An Overview

The most basic and direct mechanism by which to perform computer assisted vulnerability analysis for a particular protocol is to start with a model of the protocol, a model of the attacker and a definition of what constitutes an attack. Then it is a matter of searching the combined execution space of the protocol and attacker for traces that satisfy the attack definition. Despite the simplicity of its description, developing implementable such mechanisms is a challenging undertaking, for several classes of protocols encountered in modern communication networks. Nevertheless, there are a number of ways of doing this, and traditionally, in the area of security protocols, formal models of both the protocol and the attacker are defined (for example, the Dolev-Yao model [24]) and security is specified as a set of properties that all executions must satisfy. Using suitable models, security criteria and tools, it is possible for some types of protocols and cases, to perform automatically the checking for security violations [50], [64].

Model checking works by enumerating (ideally exhaustively) the execution space, (see [18] for an introduction). The goal is to find within the execution space violations of desired properties or to show conversely that the execution space is free of violations. In some limited instances this is possible, but in most real problems and in the specific problem of ad-hoc routing protocol security, this does not appear to be possible. A particular obstacle is that of state space explosion, which is a result of the fact that the possible reachable states of a dependent composition of state machines is exponential in the number of machines being composed. There are

a number of universally applicable reduction techniques, such as symbolic variable representation and partial order reduction for concurrent machines, but none of these approaches defeats the state space explosion problem.

Applying (or extending and applying) model checking methods to ad-hoc routing protocols is significantly further complicated by the fact that ad-hoc routing protocols behave probabilistically and their behavior is time-dependent. In OLSR [20], there are behaviors, such as packet emission that occur based on a clock variable that is uniformly distributed over a given range. This distribution affects the behavioral distribution of OLSR. In such systems, the event timings can be described by systems of constraints on the event timings that may be encapsulated in objects referred to as *Difference Bound Matrices* (DBMs) [3][2][18]. Each entry in a DBM represents one temporal constraint between two time variables of the system. In systems such as OLSR where behaviors occur according to clocks that are random and uniform over intervals while adhering to ordering constraints, the probability of particular partial order traces can be computed by evaluating the volume of the polytope specified by the DBM. Being able to evaluate the individual probability of traces enables the formulation of probabilistic model checking and cost-based evaluation of protocols.

Given that the behaviors of ad-hoc routing protocols are probabilistic (stochastic), it may not make sense to apply notions of correctness towards their evaluation in model checking. For example, it may be the case that with very low probability, packets are continually dropped and the routing algorithm cannot converge over a given time window. This gives no indication that the algorithm is inherently faulty.

It makes more sense in this context to describe the average behavior of the algorithm or the likely behaviors of the algorithm. It may make sense to call an algorithm faulty if under appropriate conditions, convergence occurs at a frequency less than some specified threshold P. Thus the model checking problem becomes one of computing the probability that certain conditions are satisfied. This removes one of the benefits of model checking, which is that instances of violating traces are produced. Instead, an entire collection of traces along with their probabilities are produced and can only be considered a violation in aggregate.

As stated earlier, it will not be possible in general to enumerate all the states. However, since the relevant features of the execution space apply to aggregates of traces, it may make sense to consider a partial exploration of the state space by random sampling of traces. This alternative approach is supported by the theory of simulation-based optimization methods [23][9].

1.3 Main Contributions of this Dissertation

The initial effort of the research reported was towards the development of partially-ordered symbolic formal models of ad-hoc routing protocols and attacks, in order to make progress in the direction of computer assisted vulnerability analysis of these protocols. A contribution from this initial effort is the development of an extension of the Strands model [29], in a manner suitable for describing ad-hoc routing protocols and attacks. On the other hand, despite considerable and diverse efforts, we have not been able to overcome (the apparently) insurmountable obstacle

of state space explosion for such systems so that they can handle problems of size and complexity typically encountered in practice. Our research efforts compiled considerable evidence that the development of such automated methods does not appear to be feasible. The research from this initial effort led to the pursuit of probabilistic-timed models for ad-hoc routing protocols and attacks.

The second major component of the work was therefore towards the development of a probabilistic model checker. The goal was to discover to what extent a methodolgy could be developed in this direction, so as to accomplish computer assisted vulnerability analysis of interesting classes of routing protocols. The focus on developing such a method yielded a significantly enhanced algorithm for computing the volume of DBMs. This is another contribution of the dissertation with diverse applications to probabilistic model checking. This is achieved by reducing the branching factor in Lasserre's recursion. In general formulations for Lasserre's recursion, the problem of eliminating redundant constraints is equivalent in complexity to the linear programming problem. Due to the special shortest-path structure of DBMs, redundant constraints can be eliminated in quadratic time in the number of clock variables using the improved algorithm. This results in significantly reducing in the computational time, although this improvement does not change the fundamental complexity class of the volume computation problem. This result can be applied towards counting the number of linear extensions of a partial order. It is also shown how this algorithm can be combined with Monte-Carlo methods to improve the convergence time of Monte-Carlo integration for this class of problems. Nevertheless, in the course of this part of the research we accumulated substantial

evidence, that computer assisted vulnerability analysis of ad-hoc routing protocols could not be developed by probabilistic model checking methodologies alone so as to handle cases of practical complexity.

Given the complexity issues posed by the above, probabilistic model checking does not appear cabable as a method, to succeed in automated attack discovery within the proposed framework. Since a probabilistic framework makes sense for this problem, the problem was reposed using sample based estimates within an optimization framework. This last framework resulted in some successes, in that computer assisted evaluation of the effects of a number of vulnerabilities in OLSR were performed. The scope of these vulnerabilities were rather narrow, but the algorithms were able to determine that the effects could be substantial. In addition our methods indicated a proposed modification to the OLSR protocol and we demonstrated that this modification circumvents the vulnerability which was revalidated within the attack discovery framework.

1.4 Dissertation Outline

The research of this thesis pursued computer assisted vulnerability analysis from three different perspectives. In the remainder of the thesis, the three main efforts of research are described in detail and the results are described together with their inter-relationships and implications. In Chapter 2, the focus is on developing a formalism for a symbolic partial order model checker. The protocol examined in Chapter 2 is AODV and the approach is roughly inspired by the Strands model,

which is extended appropriately. In Chapter 3, a probabilistic timed model for model checking is developed and the detailed mechanics of this model are developed along with a significantly computationally improved algorithm for computing the critical probabilities. In Chapter 4, a different approach is taken in order to bypass the insurmountable complexity problems encountered with the previous methods. Sampling methods are explored and a simulation-based optimization technique is used to evaluate quantitatively the effects of some simple novel attacks on the routing protocol OLSR. Chapter 5 summarizes the results, discusses limitations and describes promising directions for future work. Among the latter, the most promising appears to be (based on the evidence we compiled) a combination of formal (symbolic partial-order) checker and simulation-based (trace-based) optimization methodology.

Chapter 2

A Formal Model for Ad Hoc Routing Protocol Vulnerability Analysis

2.1 Introduction

A participant in an ad hoc network may communicate directly only with its immediately adjacent neighbors. It relies on routing services provided by its neighbors, and in turn neighbors of its neighbors recurisvely, to communicate with distant members of the network. Due to this dependency, a malicious node is in a position to disrupt services in the network, even when there is enough redundancy to avoid relying on the routing services of the malicious node directly. Current approaches to securing ad-hoc routing protocols focus on using secure signatures to authenticate routing information and some built-in mechanisms for resisting or detecting Byzantine attacks [49] [54] [72] [38]. However, authentication is not always sufficient to safeguard the network from malicious nodes, as in the case of publically accessible wireless networks or in the case of physical compromise, to which mobile devices are thought to be more susceptible. Also, methods for detecting Byzantine attacks are often susceptible to attack themselves.

This chapter focuses on developing formal models for ad hoc networks with the objective of understanding and evaluating vulnerability to Byzantine or insider attacks as well as their effects (cf. causing loss of connectivity). Such a model may aid protocol engineers in mitigating their effects or reducing the number of

vulnerabilities of this type. In addition, formal models have been used previously in intrusion detection to protect routing in fixed networks. A formal model for ad hoc networks may also prove useful for constructing intrusion detection systems in a mobile environment.

This work proposes an extension of the Strand model that captures the behavior of ad hoc routing protocols. It inherits from the Strand model the partial ordering semantics, which is useful for mitigating the state space explosion problem. In this extended Strands model, it is possible to formulate Byzantine attacks as security goals. However, neither regularity nor formulation of this problem in a decidable infinite structure is obvious. Using this Strand model, AODV has been modeled along with one of its security goals, and the semi-decision procedure is able to discover a particular attack on AODV. Nevertheless, it is our conclusion, based on the results presented and the evidence we have accumulated, that the success of methods developed along such lines will be limited to relatively small networks and simple attacks (vulnerabilities).

2.2 Background

Formal models for the analysis of ad hoc routing protocols has been appliet to the case of fixed topologies for a small number of nodes [10] [4]. Additionally several frameworks have been proposed for formally examining the security of ad-hoc routing protocols [69] [16] [1]. The work of the author in [69] is presented here.

2.2.1 Verification of Ad Hoc Routing Protocols

Consider a protocol where each node has k states and assume there are n nodes in the network. Then there are k^n possible concurrent states of these nodes, though not all of these states are necessarily reachable. Since k can be immense (for example, AODV specifies sequence numbers as 31-32 bit integers which is approximately 2-4 billion states), the exponential state space explosion problem is acute. Compounding this difficulty is the effect of topology on the behavior of the protocol. Since the number of non-isomorphic topologies is also exponential in n, the number of states increases further by another factor exponential in n. An explicit representation of the protocol by a state machine that incorporates all these topologies and transitions between topologies will be enormous.

Thus, model checking by state enumeration is only possible for a small, fixed number of nodes under special assumptions and clever reductions of the state space. Unfortunately, this approach is still intractable for the numbers of nodes found in real networks. Clearly, state enumeration model checking techniques alone are insufficient for the development of computer assisted (automated) vulnerability analysis for realstic ad hoc networks. A possible solution to the problem is to use hand proofs to establish lemmas that reduce the verification problem to one that requires only model checking a small, finite, representative fragment of the execution space of the protocol.

A notable example of this technique is the work on verifying the loop freedom of AODV [10] [53]. The authors used formal models of AODV to discover conditions

leading to the formation of routing loops by model checking finite models of AODV in SPIN [36]. They proceeded to design a repair for AODV that eliminates this problem and verified that the repaired version of AODV is loop free under some assumptions (prior to restarting its AODV process, a node must ensure that all of its neighbors detect the restart). The verification combines a hand proof performed using the mechanical proof assistant HOL with finite state model checking performed by SPIN.

2.2.2 Secure Ad Hoc Routing Protocols

The preceding discussion pertains to ad hoc routing protocols that have no security provisions, but recently, there has been intense interest in securing ad hoc routing protocols [72] [49] [54] [38] [71]. For these new protocols, the amount of formal analysis is limited, and contributions and progress in formal modeling of these kinds of protocols is one of the objectives of this dissertation. The following briefly surveys some approaches to securing routing algorithms in rough chronological order.

The more recent secure ad hoc routing algorithms borrow from secure algorithms for fixed networks. Perlman proposed a link state routing protocol that is robust to Byzantine failure, but has a very high overhead associated with public key encryption [58].

For the mobile case, Zhou and Haas propose a threshold cryptography system for key management that can be distributed across an ad hoc network that permits subversion of t from a total of $3t + 1$ nodes while maintaining security [72].

This scheme has the added benefit of working under weak assumptions about the synchronicity of network communication. However, it does not directly address the issue of subverted insiders attacking the routing algorithm itself rather than the certificate authority (CA).

Marti describes a technique for detecting misbehavior in ad hoc networks by having nodes promiscuously listen to make sure that packets are correctly forwarded by their neighbors[49]. However, wireless collisions almost guarantee that not all forwarded messages will be heard, so a node receives a rating based on the frequency of missed message forwarding observations and if the frequency exceeds a certain threshold, the node is labeled malicious. This approach does not make any formal guarantees of correctness and identifies several of its own vulnerabilities, the most critical of which is exploiting the watchdog reporting mechanism itself.

Two algorithms based on DSR, SRP and Ariadne, attempt to achieve routing security [54] [38]. The claim is that source based routing protocols are easier to secure than distance vector routing protocols because the explicit representation of the route allows greater fine grain control. For example, SRP requires nodes to authenticate the destination during each route discovery. Both SRP and Ariadne use message authentication codes to achieve authentication of route replies. They differ in what assumptions they make about the requirements on the key management infrastructure and what security guarantees they provide.

The routing algorithm SRP guarantees against malicious route replies while requiring only a security association between the source and destination nodes by using a message authentication code on queries and replies. The route discovery

phase of the algorithm is proven to work in the presence of any number of malicious nodes as long as they do not collude. However, malicious nodes can still corrupt error messages and interfere with the route maintenance phase. For example, since intermediate nodes are not authenticated, a malicious node might listen for a source routed packet and synthesize a route error message to invalidate the route. This route may not even include the malicious node.

Ariadne, based on DSR, makes stronger assumptions on the network infrastructure than SRP. Ariadne authenticates not only the endpoints in the communication, but also the intermediate nodes, and it will not be vulnerable to the same types of attacks as SRP and does not assume non-colluding compromised nodes. While the assumptions on clock synchronization have been criticized as unrealistic in mobile ad hoc networks [71], Ariadne can run using pair-wise shared secret keys or RSA asymmetric keys without requiring clock synchronization. The distribution of $n(n-1)/2$ shared secret keys could be established using the distributed threshold cryptography CA described previously or with PKI. Managing $n(n-1)/2$ shared secret keys is costly (in constrast SRP requires only shared keys between communicating endpoints). Also, the high computational cost of authenticating with PKI could be a security liability leading to denial of service where attackers inject many messages that require authentication into the network.

When using the TESLA [59] asymmetric cryptographic primitive with Ariadne, clock synchronization is required, but the timing requirements are loose. There only needs to be a known upper bound Δ on the clock disparity between the nodes in the network and an upper bound τ on the network traversal time. $2\Delta + \tau$ gives

a lower bound on authentication delay (in turn becoming a lower bound on route discovery delays) since authentication depends on the disclosure of an appropriate hash chain key. τ could be arbitrarily long in large, busy networks. Key disclosure, which requires periodic flooding of the network at a rate proportional to n, will increase network delays, thus increasing τ. This does not compromise the security of Ariadne, but affects its delay and power utilization. While TESLA achieves an asymmetric primitive with lower computational cost than conventional public key algorithms, it does so by increasing bandwidth requirements. A final criticism of TESLA is that by requiring unauthenticated messages to be buffered until key disclosure, it opens a risk of a denial of service attack based on filling this buffer.

In work related to Ariadne on packet leashes [39], the time synchronicity assumptions are less realistic. The peril of relying on synchronized clocks for security has been pointed out previously [31]. GPS has been proposed as a solution to the clock synchronization problem, in addition to being able to provide geographical packet leashes; however GPS has its own security and practicality concerns. GPS signals are too weak to provide information indoors and problematic in cities where the line of sight to the GPS satellites may be blocked by surrounding buildings. The accuracy of GPS measurements can be influenced by other environmental conditions and mobility. Finally, the weak GPS signal is vulnerable to jamming by homemade, portable devices [5].

A source based routing algorithm similar to SRP and Ariadne but not based on DSR has been proposed [6]. This algorithm assumes that there is a facility that provides pairwise shared secret keys on demand via some CA, which may be

distributed. There are two main differences between this work and the previously discussed works based on DSR. Instead of constructing routes during request propagation as in DSR, the proposed algorithm constructs hop by hop authenticated routes during reply propagation. The second and more important difference is the idea of being able to locate nodes in routes exhibiting Byzantine behavior. Essentially, by using probes to perform a binary search along the route, it is possible to identify a faulty link (not a node) in $log(n)$ faults where n is the length of the route. When such a link is identified, a weight is assigned to it that causes it to be avoided in the next route discovery phase. The authors claim that this guarantees that if there exist routes that are not controlled by adversaries, the routes will be discovered in finite time. However, this scheme appears to suffer from the same problem as the previously discussed misuse detection scheme, namely abuse of the detection system itself. For example, an attacker could use jamming to create delays in the fault probes in to incriminate certain otherwise well behaving links.

An approach based on the distance vector routing algorithm AODV rather than any source based algorithm has also been proposed [71]. This approach assumes that there is a PKI and that each node is able to securely determine if a given public key belongs to another given node. Security is improved over AODV, which has no security provisions by signing routing messages and using hash chains to sign mutable fields of routing messages. However, this work does not address the problem of malicious insiders or wormhole attacks.

The objective of all of these secure ad hoc routing algorithms can be seen as improving the survivability of mobile wireless networks. Survivability is character-

ized by three objectives: resistance, recognition, and recovery [65]. The protocols reviewed above focus mainly on resistance and partially on recognition. However, there is little analysis on how well the protocols recover from attacks, where recovery is defined as the ability to maintain services during attack and the ability to completely restore services after the attack. The techniques must be hardened under attack models in the wireless context.

2.2.3 Securing Fixed Routing Protocols

The preceding survey of approaches to securing routing in ad hoc networks excludes some ideas that might be useful, but are currently used only for fixed topology networks. The following two techniques use formal methods to increase security in wired networks, but have no analagous ad hoc wireless equivalents.

One such model for wired networks is that of filtering postures, which allows a global set of constraints on network traffic in terms of security goals to be automatically converted into a set of local filtering configurations [32]. The types of packets that may flow from one location to another in the network are defined in terms of constraints on the packet headers, which may be efficiently represented in terms of binary decision diagrams (BDD) [12]. This also allows for efficient automatic verification of local filtering configurations. This approach depends upon the fixed topology of the network, an assumption that does not apply in wireless networks. The idea of using binary decision diagrams on packet header fields might be useful in similar wireless network packet filtering problems.

Another topology dependent application of formal models to securing routing in fixed networks is sensor based intrusion detection [52]. For a given topology and a given set of sensor positions within the network, the approach constructs a model of all possible routing advertisements that a given sensor will observe. Impossible routing advertisements are automatically flagged as malicious, and other routing advertisements can be verified by probing the network via sending packets to other sensors. The approach has certain limitations and is not able to detect all possible attacks, in addition to sometimes generating false positives. Obviously, this approach will not work in wireless networks, but one interesting question it raises is whether or not one may formulate an equivalent wireless sensor intrusion detection problem.

2.2.4 Formal Methods

The previous two approaches use formal models in order to analyze security properties of wired networks. While there is no standard formal model for ad hoc networks, there is a vast literature on formal models in general [18] [26] [56][51] [48]. The following surveys just a few of the most prominent ones. Ideas from these models will be important in formulating a model for ad hoc routing protocols.

The basic model for concurrent programs is the labeled transition system (LTS), of which variants can be defined based on different notions of state. A LTS is a tuple $(Q, \Sigma, \delta, q_0, F)$ where

Q set of states (possibly infinite)

Σ set of transition labels

$\delta \subseteq Q \times \Sigma \times Q$ transition relation

$q_0 \in Q$ start state

$F \subseteq Q$ set of terminal states where

$$\forall q \in F \; \forall a \in \Sigma \; \forall q' \in Q \;\; \neg\delta(q, a, q')$$

This appears no different from the description of a finite state machine (FSM), and if Q and Σ are finite, then this indeed represents a nondeterministic FSM. However, Q can be infinite, for example including an infinite tape as with Turing machines. Σ may also be infinite. One characterization of Q that is useful for describing protocols is as a set of assignments of values to a set of variables $V = v_1, \ldots, v_n$. Let each variable v_i take on values from the domain D_i, then Q is the set $D_1 \times \ldots \times D_n$.

Process algebras have been used to reason about the properties of concurrent, communicating systems. The π-calculus is a prominent example of a process algebra [51] [56]. An important concept from process algebras like the π-calculus is bismulation, which defines a notion of equivalence between states. There is an elegant description of bisimilarity in terms of a simple two player game. Let $M = (Q, \Sigma, \delta, q_0, F)$, $M' = (Q', \Sigma', \delta', q_0', F')$ and initially let $x = q_0$ and $y = q_0'$.

1. If $x \in F \oplus y \in F'$ then player 1 is the winner.

2. If $x \in F \wedge x' \in F'$ then player 2 is the winner.

3. Otherwise $x \notin F \wedge x' \notin F'$. Player 1 chooses an enabled transition from either M or M'. That is, player 1 chooses either α and x' such that $(x, \alpha, x') \in \delta$ or β and y' such that $(y, \beta, y') \in \delta'$.

4. If player 1 chooses a transition of machine M, (x, α, x'), then player 2 must choose a transition of machine M' having the same label $(y, \alpha, y') \in \delta'$. If there are no enabled transitions for M' with the label α, then player 1 is the winner. Similarly if player 1 chooses a transition o machine M'.

5. The game repeats with $x = x'$ and $y = y'$.

M is bisimilar to M', written $M \sim M'$ iff player 2 always has a winning strategy. It is important to note that bisimilarity is more restrictive than language equivalence in automata because bisimilarity models a reactive system.

An interesting analogy for the packet filtering and sensor based intrusion detection problem exists in the context of ad hoc networks formulated as a bisimulation. Let E be a LTS that represents every possible interaction a node may exhibit in an ad hoc network under some protocol. It may not even be possible to characterize E since it must capture the behavior of every reachable state of the network and the routing protocol in general has unbounded state. Nevertheless, the formulation proceeds as follows. Let M be the behavior of a node and let E' be an unconstrained machine (single state q with transition relation containing (q, α, q) for all labels α) with the same labels set as E. First, if $E \sim E'$, then the hostile environment is indistinguishable from the secure environment and nothing can be done. Assume this is not the case for protocols of interest. Then a machine M' such

that $M' \times E \sim M \times E$ is equivalent to M, but it will not necessarily be the case that $M \sim M'$, and preferably not the case that $M' \times E' \sim M \times E'$. This yields a criterion for testing machines M' which include packet filtering mechanisms on top of existing algorithms.

Temporal logic is an extension of first order logic that includes temporal operators in addition to the regular boolean operators [48]. This is a linear time formulation, so every behavior σ is a sequence of states $< s_0, s_1, \ldots >$. Define the suffixing operator as $\sigma^i =< s_i, s_{i+1}, \ldots >$. Let φ be a unary relation on the states Q. In temporal logic, relations are interpreted on sequences, where truth on a sequence is defined as

$$\sigma \models \varphi \text{ iff } \varphi(s_0) \text{ holds.}$$

State relations can be composed using Boolean operators in the natural way. The notion extends to sets of behaviors naturally. Let Σ be a set of behaviors. Then

$$\Sigma \models \varphi \text{ iff } \forall \sigma \in \Sigma \;\; \sigma \models \varphi.$$

Temporal logic defines the following operators for describing properties of a program. Let φ and ψ be *temporal* formulas.

$\Box$ henceforth. $\sigma \models \Box\varphi$ iff $\forall i \;\; \sigma^i \models \varphi$

$\Diamond$ eventually. $\sigma \models \Diamond\varphi$ iff $\sigma \models \neg\Box\neg\varphi$.

$\bigcirc$ nextime. $\sigma \models \bigcirc\varphi$ iff $\sigma^1 \models \varphi$.

$\mathcal{U}$ until. $\sigma \models \varphi\mathcal{U}\psi$ iff $\exists j \;\; \sigma^j \models \psi \wedge \forall i < j \;\; \sigma^i \models \varphi$.

There are fairly simple rules of inference for this logic that can be used to establish invariance properties and liveness properties.

The temporal logic of actions (TLA) is a specification language for concurrent systems that allows systems to be specified with temporal logic [44][45]. The fundamental unit of execution is the action. Let $\mathcal{A}$ be an action, which is a binary relation on $Q \times Q$.

$$\sigma \models \mathcal{A} \text{ iff } \mathcal{A}(s_0, s_1) \text{ holds.}$$

A cannonical definition of a program in TLA can be written

$$Init \wedge \Box[\mathcal{N}]_v \wedge L.$$

$Init$ is a unary state relation specifying the initial conditions, $\mathcal{N} = \bigvee_{i=1}^{n} \mathcal{A}_i$ and L is a formula specifying the fairness of the system. The notation $[\mathcal{N}]_v$ is a shorthand denoting stuttering invariance. The specified system either stutters or executes one of the n actions $\mathcal{A}_i$.

Subsequently, temporal logic has been categorized into variants depending on the underlying view of time (see the survey by Emerson [26]). In Linear Temporal Logic (LTL), which has the same semantics as the temporal logic above, time is a totally ordered set $(S, <)$ of cardinality ω which is usually assumed to be isomorphic with $(\mathbb{N}, <)$. The other view of time is as an infinite branching tree structure where along each path in the tree is a linear time line. Computational Tree Logic (CTL) extends LTL by adding the following operators.

A for all futures

E for some future

CTL also restricts formulas to disallow Boolean combinations and nestings of the linear-time operators. In terms of expressiveness, CTL and LTL are incomparable. There are properties that may be expressed in LTL that are not expressible in CTL and vice versa. However, CTL has lower computational complexity for model checking than LTL.

Note that there is no difficulty in defining TLA actions in CTL by translating $\Box$ into $\mathbf{A}\Box$ and by using $\mathbf{A}\bigcirc$ to operate on state predicates of the next state. However, this is not true in general for LTL formulas. For example, the LTL formula $\neg\Box\neg P$ expresses something different from the CTL formula $\neg\mathbf{A}\Box\neg P$. While the LTL formula specifies that all executions eventually reach state P, as in $\Diamond P$, the CTL formula merely states that there is an execution that reaches state P.

There has also been interest in the area of using optimization methods for automated theorem proving [17]. This approach is based on the insight that satisfiability in propositional logic, which is the basis for many automatic theorem provers, can be modeled as a 0-1 integer programming problem. However, it is difficult to automate proofs of this type due to the computational complexity, even in the case where the theory is decidable. Consider, for example, the decidable theory $(\mathbb{Z}, +, \leq, 0)$, which permits the elimination of quantifiers. Any algorithm that can determine satisfiability of sentences in this theory has a lower bound on computational complexity of $2^{2^{cn}}$ where n is the length of the sentence and c is a constant > 0 [30]. This means that no matter what algorithm is used, even using optimization methods, the computational cost is still doubly exponential. Since protocols are complex, sentences used to describe them are necessarily long, and so automated

theorem proving with temporal logic remains infeasible. The other approach to analyzing programs is model checking, in which every system is assumed to be a finite automaton on possibly infinite strings.

A useful formalism for model checking that subsumes CTL and some of its variants is the Mu-calculus [27]. The Mu-calculus introduces the idea of least (μ) and greatest (ν) fixpoint operators. From these and the operators $\mathbf{A}\bigcirc$ and $\mathbf{E}\bigcirc$, all the rest of the CTL can be defined. Mu-calculus model checking is grounded in the Knaster-Tarski theorem [22], which guarantees the existence of solutions to the fixpoint operator and also provides a bounded algorithm for calculating these fixpoints.

Theorem 2.2.1 (Knaster-Tarski)**.** *Let L be a complete lattice (*$\sup S$ *and* $\inf S$ *exist for all $S \subseteq L$), and $\Phi : L \to L$ be an order preserving map ($\forall a, b \in L$ if $a \leq b$ then $\Phi(a) \leq \Phi(b)$). Then*

$$\sup\{x \in L | x \leq \Phi(x)\}$$

is the maximal fixpoint of Φ.

Proof. Let $H = \{x \in L | x \leq \Phi(x)\}$, $\alpha = \sup H$. α exists because L is a complete lattice. Then

$$\forall x \in H \;\; x \leq \Phi(x) \leq \Phi(\alpha)$$

by order preservation of Φ. So $\alpha \leq \Phi(\alpha)$ because $\alpha = \sup H$, meaning $\forall x \in L \; (\forall y \in H \; x \geq y) \to \alpha \leq x$. So $\alpha \in H$. By order preservation, since $\alpha \leq \Phi(\alpha)$, $\Phi(\alpha) \leq \Phi(\Phi(\alpha))$. Hence $\Phi(\alpha) \in H$ and $\Phi(\alpha) \leq \alpha$ meaning $\Phi(\alpha) = \alpha$ and α is indeed a fixpoint. α is obviously maximal because H contains all fixpoints of Φ and

$\alpha = \sup H$. □

Corollary 2.2.2. Corollary. *Let L be a* finite *complete lattice of size n, and $\Phi : L \to L$ an order preserving mapping. Then $\Phi^{n-1}(\top)$ (where $\top = \sup L$) is the maximal fixpoint of Φ.*

Proof. $\top \geq \Phi(\top)$ so by order preservation and induction $\Phi^i(\top) \geq \Phi^{i+1}(\top)$ for all $i \geq 0$. At each value of i, either $\Phi^i(\top) > \Phi^{i+1}(\top)$ or $\Phi^i(\top) = \Phi^{i+1}(\top)$ meaning $\Phi^i(\top)$ is a fixpoint. Clearly, the first holds at most for $n-1$ values of i because L has only n distinct elements so $\Phi^{n-1}(\top)$ is a fixpoint of Φ.

Let α be the maximal fixpoint of Φ. Let $\beta = \Phi^{n-1}(\top)$. $\top \geq \alpha \geq \beta$ since β is a fixpoint and α is the maximum fixpoint. By order preservation, $\Phi(\top) \geq \Phi(\alpha) = \alpha$. By reapplication, $\Phi^{n-1}(\top) \geq \alpha$. Therefore $\Phi^{n-1}(\top) = \alpha$. □

The Knaster-Tarski theorem applies to complete lattices, and one such complete lattice is the powerset of a finite set of state predicates under the subset relation. There are syntactic restrictions on Mu-calculus formulas that can be used to ensure that they are in fact monotone. The fixpoints of these formulas can be calculated, which yields a technique for model checking.

The verification results above apply mainly to finite state systems. There are also some positive model checking results for infinite state systems [15]. The decidability of bisimulation equivalence and model checking for systems resembling pushdown automata has been established [28]. However, it is completely unclear how to represent routing protocols as pushdown automata.

Some practical demonstrations of Mu-calculus model checking exist. One tech-

nique uses binary decision diagrams [12] to great effect as a symbolic representation for the state of a system [14]. Binary decision diagrams allow Boolean operations on formulas describing the set of states to be performed in linear time with respect to the size of the diagrams. In many problems, binary decision diagrams perform well, but in the worst case, they can reach sizes that are exponential in the number of variables.

SPIN is another example of model checking that uses LTL and automata rather than CTL and BDDs [36]. Again, SPIN, like Mu-calculus model checking, assumes that the automata are always finite so that all properties of the system are automatically decidable. It uses the language Promela for specifying communicating automata and computes their asynchronous product yielding a global automata. It converts the LTL formulas used to specify the desired properties of the system into Büchi automata and checks that the language of the system automaton is included in the language of the automaton generated from the LTL formula. SPIN has many performance optimizations for storing automata, and performs partial order reduction for asynchronous systems. It also includes a mechanism called bit-state hashing for dealing with automata that are too large to fit into memory. In these cases, bit-state hashing allows for model checking with an incomplete, but nearly exhaustive coverage of the states.

There are many possible future directions for the area of formal methods[19]. One such example is combining theorem proving with model checking where model checking is used as a decision procedure within a deductive framework. Another point is that tools should be be oriented towards finding errors rather than proving

correctness, which is rather consistent with the model checking approach. Additionally, methods and tools should be specialized to particular aspects of a system and need not be good at analyzing all aspects of a system.

One approach that lies between theorem proving and model checking has been used to verify secrecy and nonrepudiation in security protocols [64]. The tool Athena is based on the Strand model [29], which provides a framework for proving properties of security protocols. Athena has several advantages over other approaches that would be useful in the study of routing protocols. For instance, since the model of variables is symbolic, it is possible to describe an infinite range of executions within a finite expression. Also, partial order reduction is an inherent part of the model. Rather than representing time as a linear or branching structure, time is implicit and only appears as a set of constraints on the causal ordering of events. Athena is so efficient at checking protocols that it is possible to search the space of possible protocols (for the classes where it can be applied) to automatically generate them [60].

As a combination of model checking and theorem proving, Athena uses only symbols and simple operations on these symbols to represent the variables occurring in the system. This model can be expressed in first order logic [68]. Athena then uses a model checking procedure to check the satisfiability of these logical formulas that describe scenarios in their system. It manipulates the data variables symbolically, but explores the control paths explicitly. Consequently, only a finite number of control paths can be checked by Athena. It is possible to show that this suffices to capture all possible behaviors of a protocol that could satisfy a given formula.

However, it is doubtful that the same result applies to the problem of interest here (i.e. ad-hoc routing protocols), because in routing algorithms, the number of control paths includes all the infinite possible sequences of topology changes.

An alternative to verifying protocols is subjecting them to automatically generated fault oriented tests [34] [35]. The protocols are modeled as finite state machines and rather than checking the entire state space for violations of the correctness criteria, one only checks states reachable from faults (low level anomalous, but correct behavior). Once incorrect states reachable from faults are identified, a backwards search is performed from the fault to determine if it is reachable. This search is performed for each type of fault and each message of the system and yields a set of tests that lead to error states. The idea of restricting the search to faults helps to mitigate the effect of state space explosion in blind searches of the protocol state space. This work places emphasis on simulation of the system using the test sequences leading to realistic examples of protocol errors.

2.3 Criteria for Secure Routing

Attacks may come in many forms in an ad hoc routing protocol. It is possible that an attacker may compromise nodes and make them behave arbitrarily. It is also possible for an attacker to jam the physical medium. Survivability demands of an ad hoc routing protocol that it is able to maintain service under hostile conditions. Eavesdropping is always possible by passive snooping on the broadcast medium, so it is not an attack that is considered here. The attacks of interest are those that

disrupt routing services provided by the protocol even if there exist uncompromised routes through the network.

Routing protocols must provide routing service and may depend upon lower layer mechanisms to provide reliable, in-order communication with adjacent nodes. The MAC layer should also be formally verified to guarantee these properties, but that is not within the scope of the current discussion.

Given the above assumptions, secure routing protocols should provide routing service even in the presence of attackers. All of the secure routing algorithms referenced in Section 2.2.2 require authentication of routing messages. Based on whether or not they may generate these authenticated messages, there are two categories of attackers.

Outsider: A malicious outsider is unable to generate authentic routing messages. However, it is able to replay messages that are generated by legitimate parties and to degrade communication between nodes within its broadcast range by jamming the lower layers of communication.

Insider: A malicious insider can perform all the attacks that an outsider can. Additionally, a malicious insider has the keys necessary to generate authentic messages for its own identity. If malicious insiders cooperate and share their keys, each insider may generate any message appearing to originate from any of the compromised nodes.

In secure environments, it is possible to apply sufficient safeguards to the nodes themselves to prevent compromise, and it is only necessary to consider outsider

attacks. However, malicious insiders are a more realistic model in less controlled environments, such as public access networks.

An ideal routing protocol should be able to provide routing service in the presence of any number of attackers, as long as there is a stable, uncompromised path through the network. There are two criteria that define an uncompromised path through the network.

Safety: Every node in the path is normal, that is non-malicious and its key has not been disclosed. If there are malicious nodes in the path, then there is always a way for that node to prevent route discovery of that path.

Liveness: Communication is possible along the path. This is necessary because it is possible for malicious nodes not belonging to the path to compromise it by either jamming lower protocol layers or denial of service attacks at the routing message level. For example, continual routing messages that fill message queues or require processing may prevent legitimate routing messages from ever being processed, as a malicious node is unlikely to respect fairness requirements. More precisely, there is an upper bound on the amount of delay between nodes in the path.

Given an uncompromised path, a secure routing algorithm should be able to discover the route along the path. In the case where there may be malicious insiders, the requirement is very strong, and there is a useful abstraction that gives a simplified model of the network for the purpose of verification. Let $G = (V, E)$ be the topology of the network and let $R = \{p_1, \ldots, p_N\}$ be a chain of uncompromised nodes

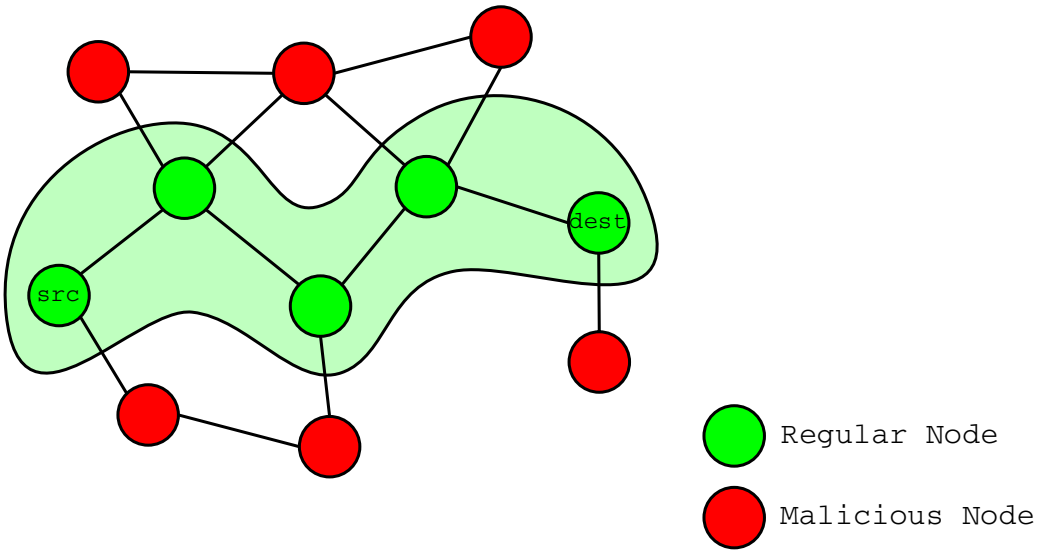

Figure 2.1: *Uncompromised path.*

where p_1 is the source and p_N is the destination where $< p_1, \ldots, p_N >$ is a stable path in the network. There may be any number of malicious insiders surrounding the path that are able to generate arbitrary authentic messages originating from any node not in R. They are also able to replay messages originating in R. The behavior of normal nodes outside R is a subset of the behavior of these malicious nodes. By verifying that routing security is achieved under attack by malicious nodes, one automatically shows that the same routing security is achieved by including any number of normal nodes along with the malicious nodes.

Checking the above scenario with any bounded N is finite state because each node has a limited RAM, though the number of states may be very large. The only state maintained by malicious outsiders is replayable messages. It is assumed that the clocks of the normal nodes are kept roughly synchronized, such that there is an upper bound Δ on the difference between the clocks and that each action takes a

nonzero amount of time, meaning that only a finite number of messages can be sent over any time interval. Let f be an upper bound on the frequency of messages that can be produced by the nodes in the chain. Then Δf gives the maximum number of messages useful for replay (simple timestamps can be used to discard messages older than $\sim \Delta$), hence the number of messages stored by the environment for replay is also finite. This shows that for any particular N and a given set of initial conditions, the system is finite state and the problem is automatically decidable. However, N is dependent on the real topology of the network. Different topologies may entail different values of $N = 2, \ldots, K$, but in practice there should be some upper bound on K, the maximum path length through a network. Additionally, the initial conditions may determine whether or not the path will be discovered. These initial conditions are dependent on the protocol itself, but since the state of the normal nodes is finite, there are still only finitely many states to check. Another approach to dealing with the problem of arbitrary path length is to use a hand proof to perform an induction on the number of nodes in this chain, which should be possible since the intermediate nodes of the chain behave similarly.

In the case where the malicious nodes are not insiders, the model becomes much more subtle. It no longer suffices to consider chains of normal nodes, because the routing algorithm needs only to discover any path through the uncompromised part of the network. This entails state reachability of the surrounding nodes over an arbitrary network. Again, it is possible to assume that there is a rough time synchronization limiting the number of useful replayable messages to Δf, but f is roughly proportional to the size of the network, which can be large in realistic

scenarios. The messages that are generated by the nodes may also depend upon the topological changes of the networks. This means that any state machine model of the network must include transitions between topologies, which grow exponentially with the number of nodes. It is unlikely that any approach based solely on model checking will be adequate for analyzing this problem. A combined approach of hand proof with model checking is more suitable. Another approach to reducing the problem could be to search only for conditions that may lead to faults.

In both cases, with or without malicious insiders, the definition of security comprises two conditions: safety and liveness. The safety condition asserts that the discovered route is stable. Otherwise, paths that are discovered can be broken by the attacker as they are formed. This can result in indefinite denial of routing service. The liveness condition asserts that it is possible to discover the route.

These conditions can be expressed formally with sentences of CTL. The safety condition can be expressed with the sentence

$$\mathbf{A}\Box(R \rightarrow \mathbf{A} \bigcirc R)$$

where R is a state predicate describing the state in which the routing table has an uncompromised route to the destination. The precise definition of R depends on the specific topology and how the routing protocol defines routes. The liveness condition is expressed in terms of a collection of assertions parameterized on a well founded set W. The state predicate R is similar to the one defined above and asserts that in the current state, the route exists. The predicates φ_i where $i \in W$ are a chain of assertions that describe a set of actions, whose inevitable outcome is R.

The liveness condition can be expressed with the sentence

$$\mathbf{A}\Box(\mathbf{A}\varphi_i \ \mathcal{U} \ R \vee \bigvee_{j<i} \varphi_j)$$

The definition of state predicates φ_i and W depends on the particular protocol and topology. These two rules are complete for describing safety and liveness properties [48], so these sentences should be sufficient for describing any routing protocol.

2.4 Proposed extended strands model

The Strand model exploits the nonbranching behavior of security protocols where each participant executes a fixed sequence of input and output events. However, this will not be entirely appropriate for routing protocols. Since the behavior of routing protocols contains branches, it is necessary to extend the Strand model. This work presents an extension of the strand model that enables it to capture the behavior of routing protocols for the purposes of verification.

Node: Since the term "node" has special significance in the Strand model, a "node" of the network will be referred to as a participant to avoid confusion where necessary.

2.4.1 Partial Order Semantics and CTL

The model of concurrent execution for the proposed model is similar to the Strand model, where events are partially ordered by the transitively closed causality relation. Since constructs from temporal logic are used, it is necessary to ensure that the semantics are compatible. The technical issue concerns true concurrency.

In CTL, every execution, corresponding to one path through the tree, must admit a *total* ordering of events. This is in conflict with the idea of modeling events by a system of constraints because in the constraint model it is possible for two events to occur exactly simultaneously. For very straightforward reasons, this technical discrepency is immaterial for the systems under consideration.

Let $<$ be the binary causal relation on the time of occurrence of events. There are three categories of events in the partially ordered model of time: causally independent, causally dependent and mutually causally dependent.

If two events, α and β are causally independent, $\neg\alpha < \beta$ and $\neg\beta < \alpha$, then there are three possible executions: $< \alpha, \beta >$, $< \beta, \alpha >$ and $\alpha|\beta$. $\alpha|\beta$ denotes true concurrency of the events α and β. Suppose that the ordering of the events α and β may affect the state reached. This implies that α and β must both affect some common process p. This can only occur if α and β synchronize in the timeline of p so they are causally dependent. The case where $\alpha|\beta$ is ruled out by the same argument because synchronization in a single process implies a total ordering of events. This behavior is easily captured by the branching time semantics. The argument may be extended to encompass sets of causally independent events.

If two events, α and β are causally dependent, then they are totally ordered, which is easily captured by the branching time semantics.

Finally, if events α and β are mutually causally dependent, that is, they must occur simultaneously, then in the branching time semantics, the events must be treated as a single event with a shared label. This might create difficulties if the causal constraints indicate that events must be concurrent, as in a causal loop. This

is impossible though because a loop including α requires an event in process a that precedes α to be caused by α and all events preceding α in a strictly precede it.

2.4.2 Messages and events

The proposed model, in the context of a LTS has an alphabet Σ consisting generally of tuples of integers. These tuples have special structure however, associated with their semantics. The terms "event", "action" and "message" refer to syntactic constructs rather than the particular events, actions or messages of an execution. The term "event instance" will be used to refer to an actual occurring event.

Events are composed of an action and a message. The action describes what the event does. It may be a communication or a synchronization point in the process. The message embodies the information that is associated with the event.

The following are the defined action symbols representing different kinds of events.

$+$ directed message send.

$-$ message receipt.

$\star$ message broadcast.

$\triangleleft$ state precedence.

$\triangleright$ state succession.

A message is a set of symbols. Each symbol in the message represents some data field that is included in the message. For example, $\sigma = \{s, d, f_1, \ldots, f_m\}$ is a directed message.

An event is formally a tuple (a, V), written aV, where a is one of the action symbols and V is a message. For events with $+$ actions, the message must be directed and include at least an element for both source and destination. For $\star$ actions, the message is broadcast so it must include the source but not necessarily the destination. Finally, for $\triangleleft$ or $\triangleright$ actions, the message must be a set containing a single symbol representing the identity of the process such as $\{p\}$. For example, $\triangleleft\{p\}$ is an event and if σ is a directed message then $+\sigma$ and $-\sigma$ are both events.

2.4.3 Role

Roles describe implicitly the state space and the behavior of processes in the protocol. In a general sense, this extended role definition combines actions of TLA with the behavioral model of Strands. The TLA action is used to describe the change in state variables, which the Strand model does not have, while Strands are used as a partially ordered, goal driven model of execution.

Formally, a role R is a tuple $R = (X^i, X^f, M, \Phi(L))$.

X^i: set of state symbols representing values of the state variables prior to the execution of the role.

X^f: set of state symbols representing values of the state variables after the execution of the role.

M: a sequence of events. If an event with the action $\triangleright$ occurs in this sequence, then it can only be the first event and it must be of the form $\triangleright\{p\}$ where p is a participant symbol. Similarly, if $\triangleleft$ occurs, then it must be the last event and of the form $\triangleleft\{p\}$.

For some finite sets Σ^d of directed messages and Σ^b of broadcast messages, the rest of the events are of the form $+\sigma_1$, $\star\sigma_2$ or $-\sigma_3$ for some $\sigma_1 \in \Sigma^d$, $\sigma_2 \in \Sigma^b$ and $\sigma_3 \in \Sigma^d \cup \Sigma^b$. For example

$$\langle \triangleright\{p\}, +\sigma_1, \ldots, +\sigma_m, \triangleleft\{p\} \rangle$$

where $\sigma_1, \ldots, \sigma_m \in \Sigma^d$, is such a sequence.

Define the length of a role R, written $l(R)$ as the length of the sequence of events M. Also define the i^{th} event of role R, written $R[i]$, where $1 \leq i \leq l(R)$ as the i^{th} event in the sequence of events M.

Φ: a formula of first order logic with free variables from the set of symbols L.

L: the complete set of uniquely labeled symbols included in X^i, X^f and M. Let $M = \langle a_1 V_1, \ldots, a_m V_m \rangle$. Let the distinguishable message symbols $\Omega = \bigcup_{i=1}^{m} \{i\} \times V_i$. The following mappings define L.

g_i: $X^i \rightarrow L_i$ where g_i 1-1 onto.

g_f: $X^f \rightarrow L_f$ where g_f is 1-1 onto.

g_M: $\Omega \rightarrow L_M$ where g_M is onto but generally not 1-1. There is a particular element of L_M that would not have a well defined inverse mapping, though

the rest of the elements of L_M would. Let α denote this symbol in L_M. If $a_i = \triangleright$ or $a_i = \triangleleft$, then $V_i = \{p\}$ for some symbol p and $g_M(i, p) = \alpha$. If $a_i = +$ or $a_i = \star$ then V_i contains s where s is a symbol for the sender and $g_M(i, s) = \alpha$. Similarly, if $a_i = -$ and V_i is a directed messsage, then the V_i contains a symbol r corresponding to the recipient and $g_M(i, r) = \alpha$. Every other element of Ω maps to a distinct element of L_M that is not α.

All pairs of ranges from L_i, L_f, L_M should have empty intersections. Let g be the union of g_i, g_f, g_M. $L = L_i \cup L_f \cup L_M$, the range of g.

The shared symbol α represents the *id* of the participant, which should remain constant throughout. In constrast, fields associated with state variables and message field values are potentially different and therefore labeled distinctly.

2.4.3.1 Logical Theories for Φ

The description of roles has so far been syntactic. The actual logical theory of Φ will depend on the protocol being studied. For most protocols, this theory will be some reduct of number theory but it is possible to have other theories, such as the theory of real numbers. The formula $\Phi(L)$ is always decidable because it models the next state function of a protocol for which an effective decision procedure must exist.

It is possible that Φ is merely a set of formulas as with an integer program. For example

$$l_1 + l_1 + l_2 + l_2 + l_2 = 0$$

where $l_1, l_2 \in L$ is such a formula which is equivalent to

$$2l_1 + 3l_2 = 0$$

in terms of what relation it defines on l_1 and l_2. This is a subset of Presburger arithmetic, $\mathfrak{N}_A = (\mathbb{N}; 0, S, <, +)$ for which satisfiability is exactly the same thing as solving an integer program. While decidable, like Presburger arithmetic, no decision procedure is fast enough for very long formulas.

In the case of routing protocols such as AODV [57] and TORA [55], it suffices to consider the theory of the structure

$$\mathfrak{N}_L = (\mathbb{N}; 0, S, <).$$

This theory admits the elimination of quantifiers, which is stronger than decidability. Some examples of atomic formulas of this structure include

$$\begin{aligned} l_1 &= l_2, \\ l_1 &\leq l_2, \\ l_1 &< l_2, \\ l_1 &+k = l_2 \end{aligned}$$

where $l_1, l_2 \in L \cup \mathbb{Z}$, $k \in \mathbb{Z}$. The formula $l_1 \leq l_2$ is equivalent to the formula $l_1 < l_2 \vee l_1 = l_2$ and the formula $l_1 + k = l_2$ is merely a suggestive way of writing the formula $s^k l_1 = l_2$. It is possible to extend the language to include relations $+k =$ as defined. Observe that on the structure $\mathfrak{N}_L$, $(+k =) \subset (<) \subset (\leq)$ for any $k \in \mathbb{Z}$. The inclusion of these extra relational symbols does not change the expressiveness of the

language but they do facilitate the description of an efficient satisfiability procedure for terms of this language to be described in Section 2.4.12.

BDDs might also be a good choice for the representation Φ. In this representation, each variable represented in L is a set of Boolean variables, and Φ can then be expressed in terms of the characteristic function of a Boolean function on L, which may be easily represented by a BDD.

2.4.4 Protocols

Define a protocol as the tuple $P = (\Sigma, \Delta)$.

Σ: set of messages. $\Sigma = \Sigma^d \cup \Sigma^b \cup \{p\}$ where Σ^d is a set of directed messages, Σ^b is a set of broadcast messages and p is a symbol representing a participant.

Δ: set of roles of the protocol where for all $R \in \Delta$ where $R = (X^i, X^f, M, \Phi)$ the messages of events in M are all in Σ.

2.4.5 Strands

A *strand* is a prefix of an execution of a role. Formally, a strand is a tuple $\theta = (R, k, I)$, where

R: a role, $R = (X^i, X^f, M, \Phi(L))$.

k: a number $1 \leq k \leq l(R)$ giving the number of events of the role that this strand instantiates.

I: some set of instance symbols. I has an element for each symbol occurring in L such that there is a 1-1 onto mapping $h : L \rightarrow I$.

Each instance symbol is unique. For any two strands $\theta_1 = (R_1, k_1, I_1)$ and $\theta_2 = (R_2, k_2, I_2)$, $I_1 \cap I_2 = \phi$. Even though the strand represents only the execution of the first k events of the role, the set I still contains all the symbols associated with L, because it might not be possible to express all the applicable constraints from Φ otherwise.

$t = (\theta, i)$, where $\theta = (R, k, I)$ is a *node* of θ if $1 \leq i \leq k$, written $t \in \theta$. t refers to the execution of event $R[i]$.

2.4.6 Goal bindings

A goal is an event required to occur by a particular node in order for that node's event to occur. For a node $t = (\theta, i)$, where $\theta = (R, k, I)$, $1 \leq i \leq k$ and $R[i] = aV$, the goal is bV, where b is some event depending on the action a as described below.

$a = +$: No goals are associated with actions of this type.

$a = -$: Message receipt has as its goal the sending of the same message, either through a broadcast or a directed send, depending on the message type. If V is a directed message, then the goal is $+V$, if it is a broadcast message, then the goal is $\star V$. There is exactly one goal instance for nodes having this action.

$a = \triangleright$: The single goal of nodes having this kind of event is $\triangleleft\{p\}$.

$a = \triangleleft$: This type of node has no goals.

Binders are also events, and these will satisfy the goals described above. Nodes having certain actions will have binders, and for these nodes the binder is $R[i]$. The number of instances of the binder depends on the binder's action symbol.

$a = +$: Exactly one instance of the binder.

$a = \star$: The number of instances of this binder depends on the topology of the network.

$a = -$: Zero binder instances.

$a = \triangleright$: Zero binder instances.

$a = \triangleleft$: Exactly one instance of the binder.

The general idea is that in an execution, all the goals must be satisified by binders. Also, the binders with actions $+$ or $\star$ require goals to be bound to them, while binders with the action $\triangleleft$ do not.

For any goal and binder pair, the binder may bind the goal when the events are equal, meaning that the actions are identical and the sets of variables are identical.

When a binder of node t_1 binds a goal of node t_2, it can be written as $t_1 \rightarrow t_2$. In the special case where $t_1 = \triangleleft\{p\}$ and $t_2 = \triangleright\{p\}$ for some symbol p, this may also be written as $t_1 \prec t_2$. Since each strand may have at most one occurrence of an event with action $\triangleright$ and one occurrence of an event with action $\triangleleft$, there is no confusion in also writing $\theta_1 \prec \theta_2$ if for some $t_1 \in \theta_1$, $t_2 \in \theta_2$ $t_1 \prec t_2$.

Define the binary relation $\rightarrow$ on nodes to hold for pairs of nodes where $t_1 \rightarrow t_2$.

2.4.7 Causal relations

The causal relation C is a transitive binary relation on nodes. It is possible to interpret this relation as $\leq$ where the node symbols are interpreted as timestamps. C arises from the strands and goal binding as follows.

For a strand $\theta = (R, k, I)$, for each $i \in \{1, \ldots, k-1\}$ and for each $j = i + 1$ the nodes $t_i = (\theta, i)$ and $t_j = (\theta, j)$ satisfy

$$Ct_it_j \wedge \neg Ct_jt_i.$$

Each node of a strand strictly precedes the higher numbered nodes in the same strand.

Given nodes t_1 and t_2 where $t_1 \rightarrow t_2$,

$$Ct_1t_2 \wedge Ct_2t_1.$$

The event associated with node t_1 occurs concurrently with the event associated with node t_2.

A given set of strands and goal bindings between their nodes describes a causal relation. If this causal relation is infeasible by interpreting over $\leq$ and timestamps, then there is no way that this describes an actual execution of the protocol. For example, if the transitive closure of constraints given by the binding relation and the ordering of nodes in a strand includes both Ct_1t_2 and $\neg Ct_1t_2$ then no ordering of t_1 and t_2 can satisfy both constraints.

2.4.8 Constraint Program

Given a set of strands $\Theta = \{\theta_1, \ldots, \theta_n\}$ and a goal binding relation $\rightarrow$ on nodes occurring in Θ define an equivalence relation E on the instance symbols of each strand in Θ.

For each strand $\theta_i \in \Theta$, let $(R_i, k_i, I_i) = \theta_i$, $R_i = (X_i^i, X_i^f, M_i, \Phi_i(L_i))$, g_i be the labeling function defined in R_i and h_i be the instantiation function defined in θ_i. By definition, for each $i \neq j$ where $1 \leq i \leq n$ and $1 \leq j \leq n$, $I_i \cap I_j = \phi$. Let $U = \bigcup_{i=1}^{n} I_i$. E is an equivalence relation on U satisfying the following.

- For $t_i \rightarrow t_j$ the instance symbols associated with the messages of the two nodes are equivalent under E. Let $(\theta_i, i_i) = t_i$ and $(\theta_j, i_j) = t_j$, where $\theta_i, \theta_j \in \Theta$. Since $t_i \rightarrow t_j$, $R_i[i_i] = aV$ and $R_j[i_j] = bV$ for exactly the same set of symbols V, though a and b will differ. For all $v \in V$

$$E(h_i(g_i(i_i, v)), h_j(g_j(i_j, v))).$$

- If $t_i \prec t_j$, then the above constraints still apply, along with some additional recursively defined constraints.

  Let $X_0^u = X_j^i$, $\theta_{p0} = \theta_i$. The following hold for $k = 0$ and for all $k > 0$ where $X_k^u \neq \phi$ and $\theta_{pk} \prec \theta_{p(k-1)}$ for some $\theta_{pk} \in \Theta$.

$$E(h_{pk}(g_{pk}(x)), h_i(g_i(x))) \text{ for all } x \in X_k^u \cap X_{pk}^f$$

$$X_{k+1}^u = X_k^u \setminus X_{pk}^f$$

  This recursion terminates for any finite set of strands and binding relations with a feasible causal relation.

The equivalence relation E defines a partition on U. For each partition, choose a distinct unused symbol and let e be a mapping from U onto this set of new symbols. $g(x) = g(y)$ iff $E(x, y)$. Consider the set of constraints $\bigcup_{i=1}^{|\Theta|} \Phi_i(I_i)$. Clearly, this is a set of constraints only on U. Apply to each symbol from U occurring in this set of constraints the mapping e. The resulting set of constraints Γ must be feasible in order for the set of strands and binding relation to be a valid execution.

2.4.9 Semibundles

A semibundle is a tuple $(\Theta, \rightarrow)$.

Θ: set of strands.

$\rightarrow$: goal binding relation on nodes occurring in Θ.

For all semibundles, the semibundle's causal relation (Section 2.4.7) and constraint program (Section 2.4.8) must be feasible.

2.4.10 Bundles

A bundle is a semibundle where every goal is bound to a binder and every binder requiring a goal to be bound to it is bound.

Theorem 2.4.1. *[29] There is a bundle iff there is a protocol execution that is consistent with the bundle.*

2.4.11 Topologies

The topologies of interest are chains of non-penetrator participants surrounded by penetrators. The penetrators can emulate the behavior of non-penetrator nodes. Distinguish endpoint roles with constraints on the identity symbol p. Then there is no ambiguity about how many binders are necessary in message broadcasts.

2.4.12 Constraint program feasibility

In the case where the model for Φ, the transitions, is an integer program, it is possible to solve it as an integer program. However, integer programs are difficult to solve, and all algorithms are in their worst case exponential in difficulty with the number of variables. Fortunately, AODV and TORA do not require the full expressiveness of integer prorgrams. They can actually be described completely on the structure $\mathfrak{N}_L = (\leq, s, 0)$ because they only make use of variable assignment, comparison and incrementing by a constant [57][55].

The constraint program arising from $\mathfrak{N}_L$ is an integer program and can be solved as such. There is a polynomial time algorithm to determine whether or not a diagonal constraint program is feasible.

Consider the atomic formulas of the first order language L_1 having nonlogical symbols $\{s, 0\}$, where s is the unary successor function and 0 is a constant symbol. All atomic formulas of this language have one of the following forms

$$s^n v_1 = s^m v_2$$

$$s^n v_1 = s^m 0$$

where $n, m \geq 0$ and s^n represents a string of n s symbols. Any formula is equivalent to one of the following normal forms

$$v_1 = s^n v_2$$

$$v_1 = s^n 0$$

$$s^n v_1 = 0$$

where $n \geq 0$.

Let $\Gamma \subset Atfm_{L_1}$ (where $Atfm_{L_1}$ denotes the set of all atomic formulas of L_1) be finite and let each formula in Γ be in normal form. Let ϕ be the formula formed by the conjunction over Γ. Let ψ be ϕ existentially quantified over all variables in $Fv(\phi)$. The satisfiability of ψ on the structure $\mathfrak{A} = (\mathbb{Z}, s)$ can be determined by a graphical method.

Let $G = (V, E)$ be the directed graph described as follows. $V = Fv(\phi) \cup \{0\}$. Define $\Gamma \subset Atfm_{L_1}$ where $|\Gamma| < \omega$ and $Fv(\Gamma) \subseteq \{x_1, \ldots, x_n\}$ for some fixed, finite n. Define the directed graph $G = (V, E)$ where $V = \{x_1, \ldots, x_n\}$, and the edges

$$E = \{(x, y) \in V \times V : +1 = xy \in \Gamma\}.$$

Theorem 2.4.2. *Γ is realized in $\mathfrak{A} = (\mathbb{Z}, +1 =)$ iff every simple, undirected loop in G contains an equal number of upstream and downstream edges.*

Proof. The goal is to show that Γ is feasible implies that every loop contains an equal number of upstream and downstream edges. It suffices to show the contrapositive. If there are an unequal number of $+1 =$ and $= 1+$ edges in a particular loop, then

that loop is infeasible, since for any node x in the loop, it says $x + k = x$ for some nonzero k, which is impossible.

The proof of the converse proceeds by constructing a solution. Assume without loss of generality that the graph is connected because disjoint partitions may be realized independently. Arbitrarily choose some $v \in V$ and some $c \in \mathbb{Z}$. Assign values to neighbors of v exactly as indicated by the edges. Continue in this manner by either breadth first or depth first search throughout the graph. The assignments will all be consistent by hypothesis. □

This result generalizes to encompass terms $+k =$ by introducing $k-1$ artificial variables and applying $+1 = k$ times.

Consider now the language L with nonlogical symbols $\{\leq, +1 =\}$, where both are binary relations. Define Γ as before, but with L instead of L_1. Define the graph $G = (V, E)$ with $V = \{x_1, \ldots, x_n\}$ as before, but the edges must include labels $\{\leq, +1 =\}$ in order to distinguish between $+1 =$ edges and $\leq$. Here,

$$E = \{(x, y, R) \in V \times V \times \{\leq, +1 =\} : Rxy \in \Gamma\}$$

Definition 2.4.1 (Simple Loop)**.** *A simple loop of a an edge labeled graph* $G = (V, E)$ *is a sequence of edges* $< (x_1, x_2, \lambda_1), (x_2, x_3, \lambda_2), \ldots, (x_{n-1}, x_n, \lambda_{n-1}) >$ *where*

$$\textit{for all } 1 \leq i \leq n-1 \ (x_i, x_i + 1, \lambda_i) \in E,$$

$$\textit{for all } 1 \leq i \leq n-1, i < j \leq n-1 \ x_i \neq x_j.$$

$$x_1 = x_n$$

Definition 2.4.2 (Direction Complemented Graph)**.** *Define the direction complemented graph G' of a graph G as $G' = (V, E')$ where*

$$E' = E \cup \{(y, x, = 1+) : (x, y, +1 =) \in E\}.$$

Theorem 2.4.3. *Γ is realized in $\mathfrak{A} = (\mathbb{Z}, \leq, +1 =)$ iff every simple loop of the direction complemented graph of G satisfies the following criteria.*

- *If the loop excludes edges labeled $\leq$, then the number of edges labeled $+1 =$ must equal the number of edges labeled $= 1+$.*
- *If the loop includes edges labeled $\leq$, then the number of edges labeled $= 1+$ must be greater than or equal to the number of edges labeled $+1 =$.*

Proof. It is easy to show necessity by contraposition. To show sufficiency, construct a feasible solution to the problem incrementally, showing that each increment preserves both criteria.

This problem is equivalent to the problem of assigning to each $\leq$ a value $s \geq 0$ such that the $\leq$ is interpreted as $+s =$. Let $e = (x, y, \leq) \in E'$. Formally, the graph after such a substitution has the set of edges

$$(E' \setminus \{e\}) \cup \{(x, y, +s =), (y, x, = s+)\}.$$

In this case, the problem is feasible if after assigning values to all of the $\leq$ edges, the first criterion still holds, as this is equivalent to the feasibility of the program in L_1.

It suffices to show that given that the criteria hold, it is possible to assign some s value to any $\leq$ edge in the graph such that after making the assignment,

the resulting graph still satisfies the criteria. Then a solution can be constructed by recursively assigning s values to the result of the previous substitution.

Replacing a $\leq$ labeled edge e may affect the criteria as follows.

(1) Loops having e as the only $\leq$ labeled edge become loops containing only $+1 =$ and $= 1+$ labeled edges. These resulting loops must satisfy the first criterion.

(2) Loops containing e along with other $\leq$ labeled edges must still satisfy the second criterion after the substutition occurs.

(3) It is possible that the substitution creates loops that did not previously exist because it adds the reverse edge $(y, x, = s+)$. The resulting loop must contain at least one $\leq$ labeled edge where the direction differed from the removed $\leq$ labeled edge so the second criterion applies.

These effects constrain the value of s to be substituted. If the criteria hold prior to the substitution, these constraints are feasible.

Lemma 2.4.4. *The set of edges in any finite loop is the union of the sets of edges of some number of simple loops.*

Proof. Recursively decompose the arbitrary finite loop into smaller and smaller loops until they are simple. The only difference between a simple loop and an arbitrary loop is that simple loops contain only one repeated vertex, the beginning and end. The arbitrary loop may revisit multiple vertices.

If the loop is not simple, then there is some vertex, call it v, that is not the beginning or end but is traversed twice. The path between the two occurrences of

the same vertex is a loop. The path starting from the beginning and up to the first instance v skipping the v- to-v loop then continuing to the end is another loop. If either of these resulting loops is not simple, then recursively reapply the process. The recursion terminates because each stage of the recursion monotonically decreases the size of the loops and the initial loop is finite. □

Corollary 2.4.5. *If every simple loop of a graph satisfies the criteria, then every arbitrary finite loop satisfies the criteria.*

Proof. Both criteria are closed under summation, and every simple loop satisfies the criteria so every arbitrary loop that can be decomposed into simple loops also satisfies the criteria. □

If any loop l containing e has effect (1) above, then s will be equal to the number of edges labeled $= 1+$ minus the number of edges labeled $+1 =$. $s \geq 0$ holds because of the second criterion. It is easy to verify that this substitution does not invalidate any of the criteria for other loops containing e by considering the fact that for any loop containing e, the same loop not including e but traversing the remainder of l must still satisfy both criteria. It is necessary to apply the corollary here in case the loop described here is not a simple loop.

Otherwise, only effects (2) and (3) apply. Effect (2) places an upper bound on s, while (3) places a lower bound. For every loop where effect (2) applies, the value s must be less than or equal to the number of edges labeled $= 1+$ minus the number of edges labeled $+1 =$. There is some loop for which this difference is minimal, though always greater than or equal to zero by the second criterion. This is the

upper bound on s.

Let $(x, y, \leq) = e$. When effect (3) applies, there is a simple path from x to y that contains at least one $\leq$ labeled edge. s plus the number of $= 1+$ labeled edges minus the number of $+1 =$ labeled edges along this simple path must be greater than or equal to zero. In other words s must be greater than or equal to the number of $+1 =$ labeled edges minus the number of $= 1+$ labeled edges.

The simple path of effect (3) runs parallel to and in the same orientation as e. Then there is a loop consisting of the simple path and the loops having effect (2). These loops must satisfy the criteria by the corollary, which guarantees the lower bound is less than or equal to the upper bound. □

This result can be used to reason about $<$ and $=$ constraints. For $<$ constraints such as $x < y$, introduce an artificial variable a_0 and represent the constraint as the pair of constraints $a_0 + 1 = y$ and $x \leq a_0$. For $=$ constraints, such as $x = y$, remove the constraint and substitute throughout the rest of the constraints x every time y appears.

2.4.13 Algorithm for checking feasibility

An adaptation of the Floyd-Warshall all pairs shortest paths algorithm [8] determines the feasibility of the constraint program of Section 2.4.12 in polynomial time.

Define costs in the network as tuples $(a, b) \in \{0, 1\} \times \mathbb{Z}$ as follows, depending

on the edge label λ.

$$(1,0) : \text{when} \quad \lambda = \quad \leq$$

$$(0,-1) : \text{when} \quad \lambda = \quad +1 =$$

$$(0,1) : \text{when} \quad \lambda = \quad = 1+$$

The rest of this discussion will assume that the labels of the edges are costs as given above.

Let (a_1, b_1), (a_2, b_2) be any two costs. Define the sum of the costs $(a_1, b_1) + (a_2, b_2)$ as

$$(a_1 \vee a_2, b_1 + b_2)$$

where $\vee$ is logical or and $+$ is addition on the integers.

If x, y are two vertices in the graph and the sum of the cost along some path from x to y is (a, b), it represents the following relation between x and y.

$$x = b + y : \text{when} \quad a = 0$$

$$x \leq b + y : \text{when} \quad a = 1$$

The algorithm requires the comparison of pairs of costs. For pairs of costs c_1, c_2, define the comparison as the subset relation: $c_1 \leq c_2$ iff $c_1 \subseteq c_2$. The subset relation is not well defined for all pairs of costs, namely those that have empty intersections. The intersection of two costs $(a_1, b_1) \cap (a_2, b_2)$ is given below.

- (a_1, b_1) if $a_1 = 0, a_2 = 0, b_1 = b_2$
- (a_1, b_1) if $a_1 = 0, a_2 = 1, b_1 \leq b_2$

- (a_2, b_2) if $a_1 = 1, a_2 = 0, b_1 \geq b_2$
- (a_1, b_1) if $a_1 = 1, a_2 = 1, b_1 \leq b_2$
- (a_2, b_2) if $a_1 = 1, a_2 = 1, b_1 \geq b_2$
- ϕ otherwise

Note that the following are equivalent.

- The intersection is nonempty.
- One of the costs is a subset of the other cost.
- The intersection is equal to one of the two costs.

Also note that the intersection is commutative.

The next lemma helps to establish that in systems consistent with the loop criteria, the algorithm only compares pairs of costs where one is a subset of another.

Lemma 2.4.6. *Let x, y be any two variables of the system and let p_1, p_2 be any two paths in the graph from x to y. If the system satisfies the loop criteria, then the sums of the costs along p_1, $c_1 = (a_1, b_1)$ and p_2, $c_2 = (a_2, b_2)$ are such at least one of $c_1 \subseteq c_2$ or $c_2 \subseteq c_1$ hold.*

Proof. It suffices to show by contraposition that if $c_1 \not\subseteq c_2$ and $c_2 \not\subseteq c_1$ then the system does not satisfy the loop criteria. Assume $c_1 \not\subseteq c_2$ and $c_2 \not\subseteq c_1$ which is equivalent to $c_1 \cap c_2 = \phi$. There are two cases to consider depending on the values of a_1 and a_2.

- If $a_1 = 0$, $a_2 = 0$ then $c_1 \cap c_2 = \phi$ iff $b_1 \neq b_2$. Since a_1 and a_2 are sums of 0, each edge of both paths p_1 and p_2 must have a cost of the form $(0, b)$. Then there is a path from y to x that has a cost of $(0, -b_1)$. This forms a loop with p_2 and the number of $+1$ = edges is not equal to the number of $= 1+$ edges violating the loop crtieria as required.

- In the case where $a_1 \neq a_2$, either $a_1 = 0, a_2 = 1, b_1 > b_2$ or $a_1 = 1, a_2 = 0, b_1 < b_2$. Consider the first of these two cases where $a_1 = 0$. As in the previous argument, there is a reverse path from y to x of cost $(0, -b_1)$. Then this forms a loop with the path having cost $(1, b_2)$. Since $b_1 > b_2$, this path violates the second loop criterion. The other case is completely symmetrical with this case.

□

The overall search of the semi-bundle space maintains some feasible set of strands and instantiates roles and binders, thus adding vertices to a graph that is already known to be feasible. The algorithm should be efficient in the sense that it should reuse information about what is known to be feasible, so it proceeds by adding vertices one by one and testing their feasibility.

Let $G = (V, E)$ be the edge complemented graph of some set of constraints where $V = \{v_1, \ldots, v_N\}$. Let $D^1_{1,1} = 0$. Define D^n for $n > 1$ recursively in terms of D^{n-1} as follows.

First define $d_{n,x}$ and $d_{x,n}$ for all $x \in \{v_1, \ldots, v_{n-1}\}$. Let

$$\{(a_1, b_1), \ldots, (a_k, b_k)\} \text{ where } k \geq 0$$

be an enumeration of the set of costs

$$\{(a, b) : (v_n, v_x, (a, b)) \in E\}.$$

In this enumeration, $k = 0$ iff there is no edge from v_n to v_x, in which case define $d_{n,x} = \infty$. A cost of ∞ represents the universal relation: the entire set $\mathbb{Z} \times \mathbb{Z}$. Otherwise

$$d_{n,x} = (a_1, b_1) \cap \ldots \cap (a_k, b_k) \text{ for some } k \geq 1.$$

Define the cost $d_{x,n}$ as previously, but swapping the positions of v_x and v_n. Finally let $d_{n,n} = (0, 0)$. After computing these minimal costs, the rest of the edges and costs may be disregarded because they represent constraints that are consequences of the minimal constraint.

If any of the distances $d_{n,x}$ or $d_{x,n}$ are ϕ, then by contraposition of the lemma, the loop criteria fail and the system is infeasible. It is only necessary to continue if all the distances are nonempty.

For all pairs of vertices $v_x \neq v_y$ where $x, y \in \{1, \ldots, n-1\}$, check that the cost $(a, b) = d_{n,x} + D^{n-1}_{x,y} + d_{y,n}$ satisfies one of the following.

- $(a, b) = \infty$
- $a = 0$ and $b = 0$
- $a = 1$ and $b \geq 0$

Otherwise the system is inconsistent with the loop criteria.

Once the consistency has been established by the above checks, define D^n.

$$
\begin{aligned}
D^n_{x,x} &= (0,0) \ \ \forall x \in [1,n] \\
D^n_{n,x} &= \bigcap_{v_y \in \mathcal{N}(n)} d_{n,y} + D^{n-1}_{i,x} \ \ \forall x \in [1,n) \\
D^n_{x,n} &= \bigcap_{v_y : n \in \mathcal{N}(v_y)} D^{n-1}_{x,y} + d_{y,n} \ \ \forall x \in [1,n) \\
D^n_{x,y} &= D^{n-1}_{x,y} \cap (D^n_{x,n} + D^n_{n,y}) \ \ \forall x \neq y \in [1,n)
\end{aligned}
$$

Theorem 2.4.7. *The algorithm succeeds iff the set of constraints are consistent with the loop criteria.*

Proof. The proof proceeds by induction on the number of vertices. For the base step where there is a single vertex, the result is trivially true because there are no loops and the system is always feasible. Inductively hypothesize that the result holds for $n-1$ vertices and show that the result holds for n vertices.

Determining the initial distance estimates $d_{x,n}$ and $d_{n,x}$ and verifying that they are nonempty checks that loops of length less than or equal to 2 are consistent with the loop criteria. The second stage of the algorithm checks that every loop of size greater than 2 is consistent with the loop criteria. It suffices to show these two propositions independently.

First, estimating the distances $d_{n,x}$ and $d_{x,n}$ for $1 \leq x \leq n-1$ succeeds iff the loop criteria hold for loops with length less than or equal to 2. Prove the left to right implication by contraposition. It suffices to show that if either the first loop criterion or the second loop criterion are violated, then the construction yields ϕ as one of the distances.

- When the first loop criterion is violated it means that for some vertex $v_x \in \{v_1, \ldots, v_{n-1}\}$ $(v_n, v_x, (0, b_1)), (v_x, v_n, (0, -b_2)) \in E$ and $b_1 \neq b_2$. Since it is an edge complemented graph, the $(v_n, x_n, (0, b_2)) \in E$ and $(v_n, v_x, (0, b_1)) \cap (v_n, v_x, (0, b_2)) = \phi$.

- When the second loop criterion is violated, it means that for some $b > 0$ either

$$(v_n, v_x, (1, 0)), (v_x, v_n, (0, b) \in E$$

$$\text{or} \quad (v_x, v_n, (1, 0)), (v_n, v_x, (0, -b)) \in E$$

  Consider just the first here, since the two cases are symmetrical. $(v_n, v_x, (0, b)) \in E$ if $(v_x, v_n, (0, -b)) \in E$ because it is a complemented graph. Then $(v_n, v_x, (0, b)) \cap (v_n, v_x, (1, 0)) = \phi$ as required.

Next show that given that the loop criteria hold, the algorithm succeeds. Let $(v_n, v_x, (a_1, b_1)), (v_n, v_x, (a_2, b_2))$ be two edges in E. Consider the definition of the $\cap$ operation between two costs and observe that when both a_1 and a_2 are both 1, the intersection is always nonempty. Therefore

$$(a_1, b_1) \cap (a_2, b_2) = \phi \text{ implies } (a_1 = 0 \text{ or } a_2 = 0).$$

In the case where $a_1 = 0$ and $a_2 = 0$, $(a_1, b_1) \cap (a_2, b_2) \neq \phi$ by the first loop criteria considering the complement link $(v_x, v_n, (a_1, -b_1)) \in E$. If $a_1 \neq a_2$, assuming without loss of generality that $a_1 = 0, a_2 = 1$ as the cases are symmetrical, consider the complement link $(v_x, v_n, (0, -b_1))$. The second loop criterion ensures that $-b_1 + b_2 \geq 0$ or $b_1 \leq b_2$. Hence $(a_1, b_1) \cap (a_2, b_2) \neq \phi$ as required.

Since every pair of costs of parallel edges must have nonempty intersections and $a \cap b = a$ or $a \cap b = b$ given that $a \cap b \neq \phi$ the initial construction of the distances $d_{n,x}$ and $d_{x,n}$ succeeds iff the loop criteria hold. Now show that the same result holds for all loops of length greater than or equal to 2.

Lemma 2.4.8. *Assume that the graph considering only the vertices* $\{v_1, \ldots, v_{n-1}\}$ *and the edges between these vertices is consistent with the loop criteria. On this graph, define* $v_x, v_y \in V$ *with* $x, y \in \{1, \ldots, n-1\}$ *and* (a^*, b^*) *the least, in the sense of* $\cap$*, cost path from* v_x *to* v_y*. Add the vertex* v_n *and edges to the graph such that there are edges from* v_y *to* v_n *and from* v_n *to* v_x*. In other words, for all simple paths from* v_x *to* v_y *there is a simple loop including the vertices* v_y, v_n, v_x*. Let* (a_1, b_1) *be the sum of the costs of the edges from* v_y *to* v_n *and* v_n *to* v_x*. If the total cost of the loop* $(a, b) = (a^*, b^*) + (a_1, b_1)$ *satisfies the check* (a, b) *or* $(a = 0$ *and* $b = 0)$ *or* $(a = 1$ *and* $b \geq 0)$*, then every simple loop including the path* v_y, v_n, v_x *is consistent with the loop criteria.*

Proof. Every simple loop including the path v_y, v_n, v_x consists of v_y, v_n, v_x and another path, p. Since the loop is simple, p does not traverse v_n and hence traverses only vertices in $\{v_1, \ldots, v_{n-1}\}$. By the previous lemma, and the fact that the subgraph containing only vertices $\{v_1, \ldots, v_{n-1}\}$ is consistent with the loop criteria, the minimum (a^*, b^*) is nonempty.

By commutativity of $\cap$, letting (a_p, b_p) denote the sum of the costs of edges in the path p, $(a_p, b_p) \cap (a^*, b^*) = (a^*, b^*)$.

First let (a_n, b_n) be the sum of the costs of the edges in the path v_y, v_n, v_x.

There are several cases to consider depending on the values of a^* and a_p,

$a^* = 0, a_p = 0$ The only case when this can result in a nonempty intersection is if $b^* = b_p$. The check of the loop criteria succeeds for (a_p, b_p) since it does for $(a^*, b^*) = (a_p, b_p)$.

$a^* = 0, a_p = 1$ From the definition of $\cap$, this can be the case only when $b* \leq b_p$. There are two subcases to consider depending on the value of a_n.

When $a_n = 1$, the loop criterion requires that $b_n + b_p \geq 0$. The fact that $b_n + b^* \geq 0$ (by hypothesis) and $b* \leq b_p$ guarantees this to be true.

When $a_n = 0$, the loop criterion requires again that $b_n + b_p \geq 0$. The same reasons as above guarantee this to hold.

$a^* = 1, a_p = 0$ This case is impossible because no matter what the assignments of b^* and b_p, it is never the case that $(1, b^*) \subseteq (0, b_p)$.

$a^* = 1, a_p = 1$ By the minimality of (a^*, b^*), this can only occur when $b^* \leq b_p$. The loop criteria require only that $b_p + b_n \geq 0$. This fact is guaranteed by the fact that $b^* + b_n \geq 0$ and $b^* \leq b_p$.

□

For loops of size greater than or equal to 2, the left to right implication should be easy to show by contraposition. To show the right to left implication, the test is exactly the situation in the above lemma where the minimum costs are supplied by the Floyd-Warshall algorithm. The minimum costs exist because of the first lemma. □

The above algorithm has a running time equivalent to the Floyd-Warshall algorithm which is $O(N^3)$, and it is easy to see how the instantiation based searching can be applied because of the incremental statement of the algorithm. This algorithm statement includes some redundancy that does not need to be implemented but simplifies the development of the proof. It is also easy to extend the above algorithm to take into account edges labeled $=$ which have cost $(0, 0)$ and edges $= k+$ labeled with arbitrary $k \geq 1$.

2.5 Search procedure

The search procedure follows directly from the search procedure in Athena [64]. For example, a disruption attack can be modeled by instantiating a pair of participants with a path between them. Then instantiate a node with the goal term where one of the participants no longer has the other one in its routing table. Additionally, instantiate a set of nodes corresponding to the initial conditions of the system. Continually bind unbound goals in all possible ways by instantiating roles and binding to existing nodes in the system until all goals are bound. If a bundle is eventually discovered, a corresponding execution exists and the property has been disproved. On the other hand, verifying the property requires that all branches of the backwards reachability search converge. In Athena, this was claimed to be undecidable in general because each instantiation of a role potentially creates new unbound goals (though in Athena the possible instantiations are in fact bounded making the procedure decidable still, which is not the case here). It is possible to

force convergence by assuming a bound on the length of executions.

2.5.1 Protocol specification language

The messages are specified in a message file with a very simple syntax: the message name, followed by the names of the message fields where each message is separated by a new line.

While roles are described as sequences of events and a set of atomic formulas taken in conjunction to constraining the values of state variables and event message fields, the actual specification language allows for the description of roles in terms of atomic formulas joined by the logical connectives and some programmatic connectives. The logical connectives supported are the usual boolean operations $\wedge, \vee, \neg$ denoting and, or and not respectively. For convenience, the operator $\Rightarrow$ represents the programmatic construct if-then-else.

The interpretation of the programmatic connective "$\Rightarrow$" differs fundamentally from logical implication. It denotes that if the first argument is true, then the latter argument must also be true. However, when the first argument is false, the second argument is not executed. In the case where $\Rightarrow$ has three arguments, the third argument represents the else clause and is invoked when the first condition evaluates to false. In practice, the $\Rightarrow$ separates into two distinct roles, one in the case where the condition evaluates true, and the other when the condition evaluates false. This can lead to a multiplication of roles because a formula may contain multiple instances of $\Rightarrow$ terms where each possible combination forms a distinct

role.

After the $\Rightarrow$ construct is removed from the formulas by separation it into its components, the remaining formula consists of atomic formulas joined by $\wedge, \vee, \neg$. Converting this to disjunctive normal form (DNF) then taking each individual conjunct yields the roles. Negations appearing on the atomic formulas can be incorporated into the formulas themselves by reversing the inequality or equalities. If the reversal is of an equality, the role splits into two, one where the equality is substituted with $>$ and one with $<$.

2.6 Implementation

The implementation is mixed Ruby (an object-oriented scripting language) [66] and C. Ruby has many desirable features in a language: iterators, blocks and closures, built-in regular expressions and garbage collection. It also has a clean interface to native C. The core of the search engine is in native C for better performance and the rest of the implementation, of which a large portion is a parser for the language, is in Ruby.

2.7 Results

The test scenario comprises four participants, one of which is an intruder. The scenario intializes the network to a state where the hop counts to the destination are the actual distances and the route is valid. This state represents R in the formalization given previously. A simple way to invalidate this condition is by specifying

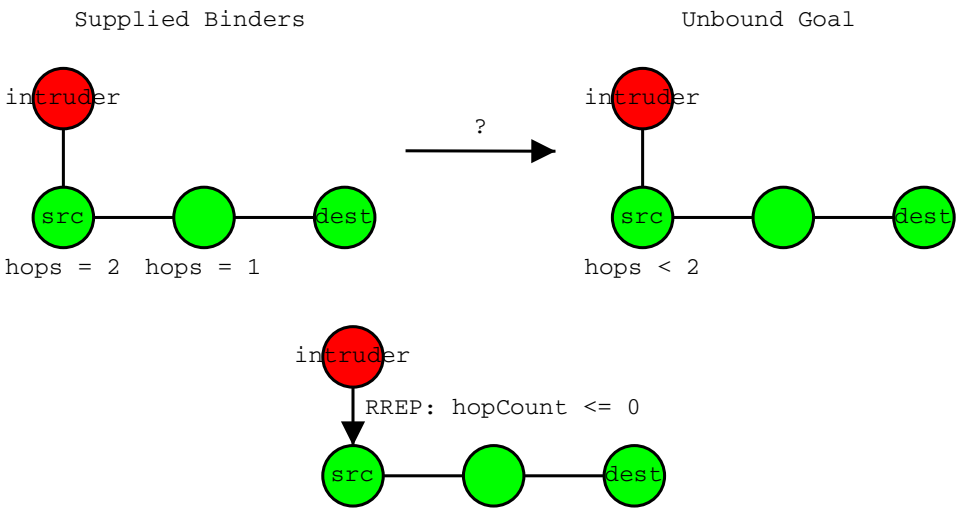

Figure 2.2: *Example scenario.*

that one of the hop counts is less than its actual distance to the destination. This goal state is an element of $\neg R$.

The goal binding search procedure examines possible executions that can lead to the goal state being reached and discovers that it is possible to reach such a state by a forged RREP emitted by the intruder. While this might be obvious to an analyst examining the protocol, this example demonstrates that it is possible to automate such reasoning.

2.8 Discussion

There is some limited success here in vulnerability analysis, in the sense of finding an attack according to the definition using correctness. However, there are a number of important limitations. One of the problems is that the above example is quite limited and also, it is not clear that correctness and liveness are the "best"

formal (logic) notions for identifying attacks on ad-hoc network routing protocols. In the next chapter, more general criteria based on a probabistic and timed model of ad-hoc network routing protocols are introduced.

Chapter 3

Probabilistic-Timed Formal Model

3.1 Introduction

The problem of studying guarantees in general for ad-hoc networks is that the underlying medium and usually the protocols themselves behave in a probabilistic fashion. For example, transmitting a packet in a wireless medium cannot be modeled as a deterministic process because depending on network conditions, there is a significant chance of packet loss. Also, wireless networking protocols often utilize random jitter as a collision avoidance mechanism. The most commonly used approach approach to automating the process of checking properties of protocols is finite state model checking. It is possible to model the nondeterminism using nondeterministic finite state machines, but it would be very sound because every possible combination of events must be explored. Many of these explored traces will consist of highly improbable scenarios where for example, a particular message is lost repeatedly. Ad-hoc networks require both probabilistic treatment and temporal analysis. Delay is an important performance metric in ad-hoc networks and guarantees on delay should be something that analysis can provide. Therefore automated analysis of performance guarantees for ad-hoc networks should be both probabistic and timed.

However, timed, probabilistic model checking limits the number of nodes that

can be analyzed. The inclusion of timing creates difficulties in the state space explosion problem, particularly if time is discretized. If time is treated symbolically, then the same fundamental problem manifests itself as an explosion in the number of constraints needed to describe the problem. These problems are insurmountable by formal analysis alone, however, the probabilistic nature of the problem provides an alternative means to address the problem.

Probabilistic model checking in the sense of counting paths satisfying LTL formulas is NP-hard, but not NP-complete[46]. Thus every problem in the class NP can be reduced polynomially to an instance of probabilistic model checking. However, an oracle that is capable of providing solutions to NP hard problems would not solve probabistic model checking instances. NP hard language membership problems can be defined by the existence of a nondeterministic Turing machine that reaches some accepting state in polynomial time if and only if the input is a member of the NP hard language. The problem of probabistic model checking not only requires the existence of this accepting state, but computing the probability of reaching an accepting state. This is an example of a $\sharp P$-complete, or counting problem (defined in [67]). Other examples of this problem include counting the number of satisfying truth assignments in propositional logic, counting the number of linear extensions of a partial order[11], and computing the volume of a convex polytope[25].

An interesting characterization of the class of $\sharp P$-complete problems is that while exact solutions are very difficult to obtain, for many instances, randomized approximations exist[41]. This means that arbitrary levels of precision can be achieved

via approximation by increasing the amount of computational resources devoted to the problem. For an intuitive example, consider the problem of counting the number of satisfying truth assignments in an instance of the propositional satisfiability problem[41]. Randomly choosing truth assignments and testing for rejection is an effective scheme for establishing a count[41] (in the sense of absolute error, not in the sense of error relative to the number of solutions which is a much harder problem). Monte-Carlo integration for convex polytopes is another example of an approximation technique for problems of this class[25]. The error reduces at a rate of $1/\sqrt{n}$, where n is the number of samples, although the initial constant factor can be important and is dependent on the problem instance[62]. The approach advocated in this thesis for the probabilistic model checking problem for ad-hoc network routing utilizes this property. The problem is then formulated within the framework of Monte-Carlo simulation. Thus, the work addressed in this chapter addresses the analysis of properties of ad-hoc routing algorithms using Monte-Carlo simulation.

3.1.1 Related Work

Formal analysis has been applied to a large number of communication protocols including routing (AODV[10]). However, there is no work yet on applying formal analysis towards understanding timed or probabilistic properties of routing algorithms. The work on AODV focused on correctness and loop-freeness and not performance. The most closely related work would be the formal analysis of the 802.11 MAC [43] protocol. It shares the property of being probabilistic and timed.

As with OLSR, there are no guaranteed time bounds, and only probabilistic ones may be established.

As stated in the introduction, the formal analysis of routing algorithms is heavily exacerbated by the state space explosion problem. A naive approach using standard tools alone is impossible. In the analysis of 802.11 [43], only two nodes were modeled and there were in excess of 5 million states examined. The objective in this work is to analyze at least a network for which routing properties can be explored. Existing model-checking tools are inappropriate for this task in their current form. The problem is primarily the orientation of the tools. The probabilistic model checker PRISM[42] is closest in orientation to this task. However, for this class of problems, $\sharp P$, model checking by brute force state enumeration is fundamentally flawed. As stated in the introduction, the correct framework for addressing these types of problems is Monte-Carlo simulation.

3.2 Review of OLSR

In this section we review OLSR based on a mathematical decomposition by components[70]. For details on the operation of OLSR, refer to the RFC [20]. Figure 3.1 shows the dependencies in the behavior of OLSR. It is constructed by an analysis of the OLSR protocol by tracing the data flow in the OLSR algorithm. The components described decompose readily due to the clean design of OLSR. The figure may be interpreted in a manner similar to Bayesian networks (see [40] for an introduction to Bayesian networks).

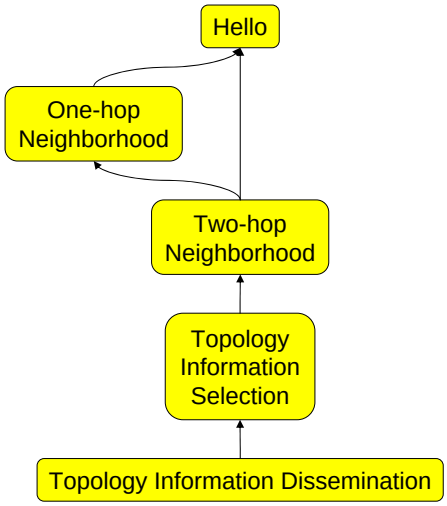

Figure 3.1: *OLSR Dependencies*

To see how this information can be used, associate with each node in the above figure a variable representing the state of the routing protocol associated with the component shown. For example, define the following random processes associated with the depicted components.

A : *Hello* process that generates hello messages having state space $\mathbf{A}$.

B : *One-hop Neighborhood* process of one-hop symmetric neighborhood discovery having state space $\mathbf{B}$.

C : *Two-hop Neighborhood* process of two-hop symmetric neighborhood discovery having state space $\mathbf{C}$.

D : *Topology Information Selection* process of generating (selecting) topological information to be distributed in topology control messages with state space $\mathbf{D}$. This includes selecting MPRs and selectors then notifying the MPRs.

E : *Topology Information Dissemination* process of disseminating to the network with topology control messages with state space **E**.

The state of these components at time k is given by $(a_k, b_k, c_k, d_k, e_k, f_k)$. Notationally, $x_{1:k}$ may stand for either $x_1 \wedge x_2 \wedge \ldots \wedge x_k$ or $(x_1, x_2, \ldots, x_k)$ depending on what the context requires. The following decomposition of the probability density function shows how the conditional independencies of Figure 3.1 can be applied.

$$P(a_{1:k} \wedge b_{1:k} \wedge c_{1:k} \wedge d_{1:k}) =$$

$$P(d_{1:k}|a_{1:k} \wedge c_{1:k})P(c_{1:k}|a_{1:k} \wedge b_{1:k})P(b_{1:k}|a_{1:k})P(a_{1:k})$$

The above decomposition is salient because each of the conditional probabilities can be computed naturally from the protocol behavior. Note the application of the conditional indepence property

$$P(d_{1:k}|a_{1:k} \wedge b_{1:k} \wedge c_{1:k}) = P(d_{1:k}|a_{1:k} \wedge c_{1:k})$$

which follows from the data dependencies shown in Figure 3.1 or in other words the causal dependencies of the protocol model induce these probabilistc dependencies, which reduce significantly the overall complexity of the probabilistic model. For example, the state of the two-hop symmetric neighborhood discovery process given by $c_{1:k}$ depends only on the state of the one-hop symmetric neighborhood discovery process given by $b_{1:k}$ and the hello process given by $a_{1:k}$.

The computation of $P(a_k \wedge b_k)$ follows below.

$$P(a_k \wedge b_k) = \sum_{a_{k-1} \in \mathbf{A}, b_{k-1} \in \mathbf{B}} P(a_k \wedge b_k \wedge a_{k-1} \wedge b_{k-1}) \tag{3.1}$$

$$= \sum_{a_{k-1}, b_{k-1}} P(a_k | b_k \wedge a_{k-1} \wedge b_{k-1}) P(b_k \wedge a_{k-1} \wedge b_{k-1}) \tag{3.2}$$

$$= \sum_{a_{k-1}, b_{k-1}} P(a_k | a_{k-1}) P(b_k | a_{k-1} \wedge b_{k-1}) P(a_{k-1} \wedge b_{k-1}) \tag{3.3}$$

Note that the step from (3.2) to (3.3) uses the conditional independence implied by Figure 3.1: $P(a_k | b_k \wedge a_{k-1} \wedge b_{k-1}) = P(a_k | a_{k-1})$. Observe that (3.3) is a recursive expression where it is assumed that the initial probabilities $P(a_1 \wedge b_1)$ are given.

This formulation provides a basis for decomposing OLSR into a number of distinct components for analysis. For example, given this causal dependency graph, it is possible to attempt to localize OLSR faults to invididual components of OLSR. This same structure will be useful when modifications are required because it will be possible to localize modifications to particular components.

3.2.1 Components of OLSR

OLSR can be divided into the following major components[70].

- Neighborhood Discovery
- Topology Information Dissemination (with two important sub-compononents, Topology Information Selection and Topology Information Dissemination)
- Pathfinding
- Data Forwarding

While *Data Forwarding* is an important component in routing, the OLSR standard [20] does not actually specify how packets are to be forwarded, only how routing tables are to be maintained.

3.2.1.1 Local Topology Maintenance

The first of these components, neighborhood discovery occurs via regular HELLO packets. The HELLO packets include a node identifier for the node performing the HELLO. This allows neighbors to become aware of the broadcasting node. It additionally broadcasts the identifier of each of its heard neighbors along with a flag indicating each neighbor as being asymmetrical, symmetrical or lost.

Asymmetrical: The broadcasting node has heard a neighbor, but the neighbor does not appear to hear it. Upon hearing a new neighbor initially, this is the state set if the neighbor does not indicate the receiver as a neighbor.

Symmetrical: A symmetrical link exists between the broadcasting node and this neighbor. This is set by a node A whenever it hears a neighbor node B broadcast the ID of A as either asymmetrical or symmetrical.

Lost: A link existed between the broadcasting node and a neighbor, but it has not heard any HELLO messages for 3 HELLO intervals.

The end goal of this neighbor discovery process, in OLSR, is that each node knows its two-hop symmetric neighborhood.

3.2.1.2 Information Dissemination

Nodes that are selected as MPRs will then flood information via TC (topology control) messages to the network. The TC messages include the identity of the MPR node and the identity of every node selecting it as an MPR, thus disseminating information about the symmetrical links between all MPRs and selectors throughout the network.

The flooding mechanism uses the MPR structure to reduce the overhead of flooding. Rather than having every node repeat a message, only nodes hearing a message from a symmetric neighbor selecting it as an MPR will forward the message. This system has one peculiarity: If a message is heard broadcast by a neighbor that does not select the node as an MPR, then heard again from a node that does select it as an MPR, then it will not forward the message. See Section 3.4.1 of the OLSR RFC [20] for details.

3.3 Protocol Specification by Extended Executable Model

OLSR will be specified formally by an extended executable model. An executable model is a formal specification, though it does include some specific implementation details. The execution of OLSR will have two sources of nondeterministic behavior. One is the random jitter that is part of the collision avoidance mechanism of OLSR and the other is the fact that packets that are transmitted have a non-zero probability of loss. In order to accommodate the analysis, both of these mechanisms will require slightly more careful treatment, entailing the extension of

normal executable models.

Ordinarily, in a simulation or executable model, the presence of branching behavior such as the random timer causes the simulation code to generate a random number from the correct distribution and inserts it into the simulation engine. This extended executable model requires not just the ability to sample from the execution space as above, but also to know the distribution of successor states from any given state. It is clear that given this distribution, the simulation engine can then either sample the distribution or perform state exploration from a given state. This can be achieved without detailed knowledge of the states themselves by the simulation engine.

3.3.1 Basic Formal Model

The basic formal model will be that of a discrete event simulation of a number of concurrent processes. Each process has its own internal state variables. Formally, let $P = (Q, E, \Delta)$ be a process.

Q: A state of the process P. It is fairly arbitrarily defined and may contain state variables or arbitrary internal data structures.

E: A set of events. These are fairly arbitrarily defined and may contain variables or internal data structures. Each event e has associated with it a time t_e. In the following let p denote any previous event and t_p the time at which it occurs.

- A specific absolute time value relative to 0.

- A specific relative time value to another preceding event. t_e may be defined as $t_e = t_p + D$ for any constant $D > 0$, generally implying a causal relationship between e and p.
- A random relative time value to another preceding event. t_e may be defined as $t_e \geq t_p + L$ and $t_e \leq t_p + U$ with $0 < L < U$. t_e occurs nondeterministically on the interval $[t_p + L, t_p + U]$.

Δ: A mapping $\Delta : Q \times E \rightarrow Q \times 2^E$. This mapping takes a state $q \in Q$, an event $e \in E$ and produces a successor state $q' \in Q$ and a set of new events $e' \subset E$. The new events in $\epsilon \in e'$ are constrained such that $t_\epsilon > t_e$ due to causality.

The state of a process is captured by the tuple (q, h, e, c).

q: A particular state that is an element of Q.

h: A subset of E of events that have already happened (history).

e: A subset of E of enabled events.

c: A set of causality constraints. These show up as constraints on the time values of events in e and h.

The nondeterministic time intervals are for now only uniform because OLSR uses uniform random jitter. The analysis that follows, particularly in section 3.3.3, assumes this fact, although other formulations are possible that are free from this assumption.

This is a minimalistic model that places most of the complexity in the process P. This basic model does not capture partial orders but can be extended to do so

if required.

3.3.2 Formal Timing Analysis

In OLSR, the random jitter is always based on a uniform distribution over a fixed interval. This fits within the framework of timed model checking of Alur and Dill[3]. The basic argument is presented below.

The following objects comprise the model.

$\Sigma = \{\phi, e_1, e_2, \ldots, e_n\}$: Set of events. ϕ denotes a special, constant zero event and can be aliased as e_0.

$\{0, t_1, t_2, \ldots, t_n\}$: Set of times for events in Σ. ϕ occurs at time 0, and for $i = 1 \ldots n$ event e_i occurs at time t_i.

C: Set of timed causal constraints having the form $t_i + d_{ij} \geq t_j$. For constraints involving ϕ, the following forms can be used: $0 + d_{0i} \geq t_i$ or $t_i + d_{i0} \geq 0$.

Given a candidate set of events Σ and constraints C, it is necessary to be able to determine whether or not an execution is feasible. The main result is that this corresponds exactly to the negative cycle detection problem[18].

Theorem 3.3.1. *Interpreting the d_{ij} quantities from the set C as distances between nodes i and j, the resulting distance matrix has no negative cycles if and only if there is an assignment of t values that satisfies all of the constraints.*

Proof. To show that having no negative cycles is necessary for feasibility assume

that for some $n_1, \ldots, n_k$ we have constraints from C that form a negative cycle:

$$d_{n_1 n_2} + d_{n_2 n_3} + \ldots + d_{n_{k-1} n_k} + d_{n_k n_1} < 0. \tag{3.4}$$

The equations associated with the above constraints are

$$x_{n_1} + d_{n_1 n_2} \geq x_{n_2}$$

$$x_{n_2} + d_{n_2 n_3} \geq x_{n_3}$$

$$\ldots$$

$$x_{n_{k-1}} + d_{n_{k-1} n_k} \geq x_{n_k}$$

$$x_{n_k} + d_{n_k n_1} \geq x_{n_1}.$$

These can be combined as

$$x_{n_1} + d_{n_1 n_2} + d_{n_2 n_3} + \ldots + d_{n_{k-1} n_k} + d_{n_k n_1} \geq x_{n_1}$$

$$d_{n_1 n_2} + d_{n_2 n_3} + \ldots + d_{n_{k-1} n_k} + d_{n_k n_1} \geq 0. \tag{3.5}$$

Note that (3.5) directly contradicts (3.4). This shows that negative cycles in the constraint system C result in infeasibility proving that feasibility implies no negative cycles.

Conversely, a constructive argument can be used to prove that having no negative cycles is sufficient for feasibility. Using each value d_{ij} from the constraint set C as the distance from node i to j, construct a shortest path matrix D with values D_{ij} for $i, j = 0, \ldots, n$. Entries D_{ii} are defined to be 0. By assumption, the values d_{ij} produce no negative cycles, so the distance matrix D is a well defined shortest path matrix. By definition of shortest paths, the resulting values must be

less than or equal to their original values: $D_{ij} \leq d_{ij}$. Interpreting this as constraints

$$t_i + d_{ij} \geq t_i + D_{ij} \geq t_j$$

shows that the new constraints in the shortest path matrix D are at least as strict as the original constraints in C, meaning that any feasible solution to the constraints specified by D will also be feasible solutions to C. Since the matrix D is a shortest path distance matrix, the following always holds:

$$D_{ij} + D_{jk} \geq D_{ik} \quad i, j, k = 0, \ldots, n. \tag{3.6}$$

Consider the point defined by

$$(0, -D_{10}, -D_{20}, \ldots, -D_{n0}).$$

It is possible to verify that this point satisfies all the constraints specified by the distances in the shortest path matrix. For constraints of the form $t_i + D_{i0} \geq \phi$, substitution results in

$$-D_{i0} + D_{i0} \geq 0$$

which obviously holds. For constraints of the form $\phi + D_{0i} \geq t_i$, substitution results in

$$0 + D_{0i} \geq -D_{i0}$$

which must hold since there are no negative cycles. Finally for constraints of the form $t_i + D_{ij} \geq t_j$, substitution yields:

$$-D_{i0} + D_{ij} \geq -D_{j0}$$

$$D_{ij} + D_{j0} \geq D_{i0}$$

which must hold because of (3.6). Thus the absence of negative cycles is sufficient for feasibility of a constaint set C. □

So a constraint set C is feasible if and only if it has no negative cycles. This means that feasibility of a particular partial order described by Σ and C can be easily checked by applying a negative cycle detection algorithm to the constraints, such as the Floyd-Warshall algorithm[21]. This is the basic mechanism for state space exploration of the model checker UPPAAL [7].

3.3.3 Probabilistic Timing Analysis

In Section 3.3.2, timing is treated as only a feasibility problem, but the time values generally come from a probability distribution. The time values of particular interest come from the uniform random jitter of OLSR and the probability of an execution of OLSR can be calculated. This enables probabilistic reasoning over the space of executions of the protocol.

3.3.3.1 Preliminaries

Before proceeding further, it is necessary to establish some basic facts about the shortest path matrix D as constructed in Section 3.3.2.

Theorem 3.3.2. *Assume that some feasible system (having no negative cycles) has shortest path matrix D. Then for all D_{ij} the following relation between the variables t_i and t_j holds*

$$t_i + D_{ij} \geq t_j. \tag{3.7}$$

Where D_{ij} is the shortest distance from i to j with distances between nodes provided by initial constraint differences d.

Proof. For each i, j pair D_{ij} being the shortest distance between them implies that for some $n_1, \ldots, n_k$:

$$d_{in_1} + d_{n_1n_2} + \ldots + d_{n_{k-1}n_k} + d_{n_kj} = D_{ij} \tag{3.8}$$

Each distance d_{xy} in the constraint set C, is defined by $t_x + d_{xy} \geq t_y$. A combination of the constraints associated with $d_{in_1}, d_{in_2}, \ldots, d_{n_{k-1}n_k}, d_{n_kj}$ yields

$$t_i + d_{in_1} + d_{n_1n_2} + \ldots + d_{n_{k-1}n_k} + d_{n_kj} \geq t_j, \tag{3.9}$$

which shows that (3.7) holds. □

Theorem 3.3.2 demonstrates that the the shortest path distance between nodes in the graph representation correspond directly to relationships between the corresponding variables that can be derived from the constraint set in the linear inequality set representation. The next result will demonstrate a special form of redundancy that allows for the elimination of variables, but first a small lemma will be needed.

Lemma 3.3.3. *Assume that for two variables t_i, t_j, $D_{ij} + D_{ji} = 0$ or*

$$D_{ij} = -D_{ji}. \tag{3.10}$$

Then for all $k \neq i, j$, the following equalities hold:

$$D_{ik} = D_{ij} + D_{jk} \tag{3.11}$$

$$D_{ki} = D_{kj} + D_{ji} \tag{3.12}$$

$$D_{jk} = D_{ji} + D_{ik} \tag{3.13}$$

$$D_{kj} = D_{ki} + D_{ij}. \tag{3.14}$$

Proof. Assume that the hypothesis (3.10) holds. According to the theorem 3.3.2, the following applies

$$t_i + D_{ij} \geq t_j \tag{3.15}$$

$$t_j + D_{ji} \geq t_i. \tag{3.16}$$

Substituting (3.10) into (3.15) results in

$$t_i - D_{ji} \geq t_j$$

$$t_j + D_{ji} \leq t_i$$

which along with (3.16) shows that

$$t_j + D_{ji} = t_i, \tag{3.17}$$

and equivalently by (3.10)

$$t_i + D_{ij} = t_j. \tag{3.18}$$

Now consider any variable t_k such that $k \neq i, j$. The following hold because D is a

shortest path matrix:

$$D_{ik} \leq D_{ij} + D_{jk} \tag{3.19}$$

$$D_{ki} \leq D_{kj} + D_{ji} \tag{3.20}$$

$$D_{jk} \leq D_{ji} + D_{ik} \tag{3.21}$$

$$D_{kj} \leq D_{ki} + D_{ij} \tag{3.22}$$

Substituting (3.10) into (3.21) results in

$$D_{jk} \leq -D_{ij} + D_{ik}$$

$$D_{ik} \geq D_{ij} + D_{jk}. \tag{3.23}$$

Thus due to (3.19) and (3.23)

$$D_{ik} = D_{ij} + D_{jk}$$

holds (3.11). Similarly from (3.10) and (3.22)

$$D_{kj} \leq D_{ki} - D_{ji}$$

$$D_{ki} \geq D_{kj} + D_{ji}.$$

Therefore

$$D_{ki} = D_{kj} + D_{ji}$$

holds (3.12). By symmetry

$$D_{jk} = D_{ji} + D_{ik}$$

$$D_{kj} = D_{ki} + D_{ij}$$

both hold (3.13) (3.14). Thus each of the equalities of the consequent have been shown to hold. □

Theorem 3.3.4. *When* (3.10) *holds, that is*

$$D_{ij} = -D_{ji},$$

then either of the two variables t_i *or* t_j *(not simultaneously) may be removed from the shortest path matrix* D *while preserving all relationships between all other variables.*

Proof. Assume that the hypothesis (3.10) holds so therefore Lemma 3.3.3 applies. Suppose that variable t_j is removed. Then the structure does not change since the constraints

$$t_j + D_{jk} \geq t_k \tag{3.24}$$

$$\text{and} \quad t_k + D_{kj} \geq t_k \tag{3.25}$$

are implied by

$$t_i + D_{ik} \geq t_k \tag{3.26}$$

$$\text{and} \quad t_k + D_{ki} \geq t_i. \tag{3.27}$$

Substituting for t_i using (3.17) in the above results in

$$t_j + D_{ji} + D_{ik} \geq t_k \tag{3.28}$$

$$\text{and} \quad t_k + D_{ki} \geq t_j + D_{ji}. \tag{3.29}$$

By Lemma 3.3.3, (3.13) holds and applying it to (3.28) results in

$$t_j + D_{jk} \geq t_k$$

which is exactly (3.24). Applying first (3.10) then (3.14) (again using Lemma 3.3.3) into (3.29) results in

$$t_k + D_{kj} \geq t_j$$

which is exactly (3.25). Removing t_i instead of t_j follows similarly by symmetry. In summary, all variables that are related by $D_{ij} + D_{ji} = 0$ are structurally redundant.

□

3.3.3.2 Deriving the Timing Model from the Formal Model

The formal model will be used to generate instances of the timing model. The manner in which it does so is described here. Based on the results of Section 3.3.3.1 Σ may contain just a subset of the events from the formal model without losing any information. Construction of this Σ will be described below.

There will be an auxiliary data structure used to store information about redundant events and their predecessors. Each event will be stored as the tuple (e, p, d).

e: The event itself.

p: The parent event. It is assumed that p precedes e in time.

d: In the case of redundant events, the *fixed* distance $t_e - t_p$. In the case of irredundant events, the minimum distance between t_e and t_p corresponding to the L in $t_e \geq t_p + L$.

In the case of redundant events, the d value in the above satisfies $d = t_e - t_p$ or $t_p + d = t_e$ which can be rewritten in more familiar terms as

$$t_p + d \geq t_e$$

$$t_e - d \geq t_p.$$

The redundant events will be those that satisfy condition (3.10).

Recall from Section 3.3.1 that there were three given ways to define the time of an event. In the case where events are redundant, they are handled as follows.

1. *Absolute time in relation to* 0. These will be redundant with ϕ. A tuple will be created with (e, ϕ, D), where D is the absolute time.

2. *Fixed relative time to a past event.* These are always redundant. Let e be the current event, p be the past event and D be the distance $D = t_e - t_p$. There are two possibilities depending on whether p itself is a redundant event. In the case where it is not redundant, then the e is stored as the tuple (e, p, D). If p is redundant, then it has an associated tuple (p, p', D'). In this case, e gets stored with the tuple $(e, p', D' + D)$.

3. *Random relative time over a specified interval.* These can never be redundant, but they will be affected by redundant data. Let e be the random event, p be the predecessor event and let L and U be the constants in the constraints $t_e \geq t_p + L$ and $t_e \leq t_p + U$. e is added to Σ and t_e is added to the list of times. How the constraint set is modified depends on whether p is redundant.

   If p is an irredundant event, then the constraints are put into standard form and added to the constraint set as follows

   $$t_e - L \geq t_p$$

   $$t_p + U \geq t_e.$$

   In this case, the tuple is stored as (e, p, L).

Otherwise p is a redundant event. Let (p, p', D) be the tuple associated with p. Since $t_p = t_{p'} + D$, the correct modification of the constraint should be

$$t_e - (L + D) \geq t_{p'}$$

$$t_{p'} + U + D \geq t_e.$$

In this case, the tuple is stored as $(e, p', L + D)$.

Note that based on the rules, redundant tuples always indicate an irredundant tuple as its predecessor. Also observe that Σ will be comprised of ϕ and a single variable corresponding to each random variable in the execution.

Finally, the supplemental causal constraints must be translated and added to the constraint set. Assume supplemental constraints are given in the form $t_{e_1} + D \geq t_{e_2}$. e_1 and e_2 might be redundant events. In this case, a substitution is performed. Supposing that e_1 is redundant with tuple (e_1, p_1, d_1), a substitution is performed for t_{e_1} using $t_{e_1} = t_{p_1} + d_1$. Similarly for e_2. This will put the supplemental constraints in terms of irredundant variables.

The probability of a particular execution occurring is based on the distribution of the random variables. Define a new set of random variables $\{\tau_1, \ldots, \tau_n\}$ to correspond with the duration of the uniform random intervals. It is clear that the probability of a region in τ-space is the ratio of its volume to the overall rectangular region. However, the timing model is not formulated in terms of the lengths of the intervals, but rather their starting and ending times. It is possible to extract this information from the timing model. Let e_i be an event corresponding to the random

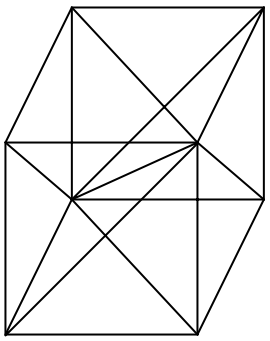

Figure 3.2: *Unit cube partitioned by coordinate order* - $6 = 3!$.

interval τ_i and let (e_i, p_i, d_i) be its associated tuple. Then

$$\tau_i = t_{e_i} - t_{p_i} - d_i.$$

This is true for each τ_i so it is possible to write down the expression in vector form $\tau = At - d$. In order to compute the volume in τ-space, it suffices to compute the volume in t space then multiply by $|Det(A)|$.

3.3.3.3 Volume Computation

The discussion above (Section 3.3.3.2) means is that it will be necessary to have an algorithm for computing the integral over convex polytopes described by the causal constraints. A discussion of the general characteristics of this problem follows.

The problem of exact volume computation is a known hard problem [25]. It is possible to use volume computation, for instance, to compute the number of linear extensions of a partial order [11]. A unit cube with number of dimensions N can

be partitioned into $N!$ regions based on the ordering of the coordinates. Figure 3.2 provides an example in $3D$. These partitions are all identical in volume by symmetry so each must have a volume of $1/N!$. The partial order constraints on the unit cube describe a convex polytope with volume exactly equal to the number of linear extensions divided by $N!$.

Since counting the number of linear extensions of a partial order is known to be $\sharp P$-complete [11] and is reducible to volume computation, volume computation must be at least $\sharp P$-hard. Monte-Carlo integration is a standard approximation scheme for these types of problems. What makes Monte-Carlo integration potentially slow is that it is based on a ratio to an initial estimate. Essentially, the technique requires knowing exactly the volume of a figure that encloses the region of interest. Then a uniform sampling of points is taken from the enclosing region, and membership within the volume of interest is evaluated. The ratio of members to nonmembers gives the ratio of the volume of the enclosed figure to the volume of the enclosing figure. It is clear that having an enclosing figure as close as possible to the figure of interest will produce the most efficient Monte-Carlo integration. If the figure encloses a space much larger than the volume of interest, then a vast majority of samples will be wasted on empty space. Thus the basis for any Monte-Carlo integration is still a good technique for exact volume computation. In this case, a projection of a lower dimensional computable volume into the higher dimensional space can serve as a starting point for Monte-Carlo sampling.

Lasserre's recursive decomposition of volumes is a possible choice as a starting point for this problem[47]. Lasserre's method is based on recursively decomposing

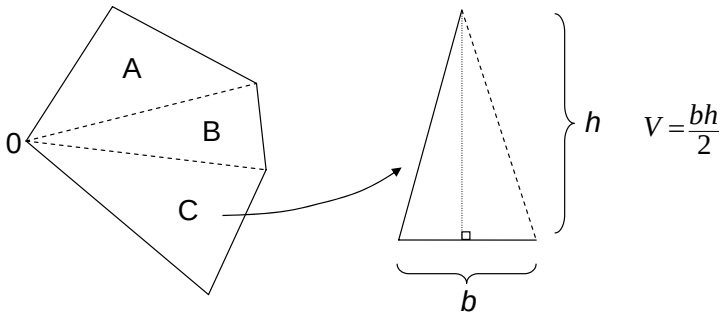

Figure 3.3: *Lasserre's algorithm example in 2D.*

convex polytopes into pyramids with a base that is one dimension less than the starting dimensionality. At one dimension, the volume of a polytope is just the length of an edge and therefore trivially computable. Figure 3.3 shows an example in two dimensions. The region is partitioned into sub-regions A,B and C, for which the volume is easier to evaluate.

The formula $V = bh/2$ generalizes for pyramids in higher dimensions to $V = bh/D$, where D is the number of dimensions, b is the volume of the base and h is the height of the pyramid. The main problem with this decomposition is that each dimensionality reduction step causes a fanout of the number of resulting polytopes. This is an exponential fanout that is endemic to problems that are NP-hard and unavoidable.

3.3.3.4 Adapting Lasserre's Algorithm to Temporal Structure

It is possible to adapt Lasserre's algorithm to the particular problem of interest here (routing protocols) and exploit the shortest path structure. Since every

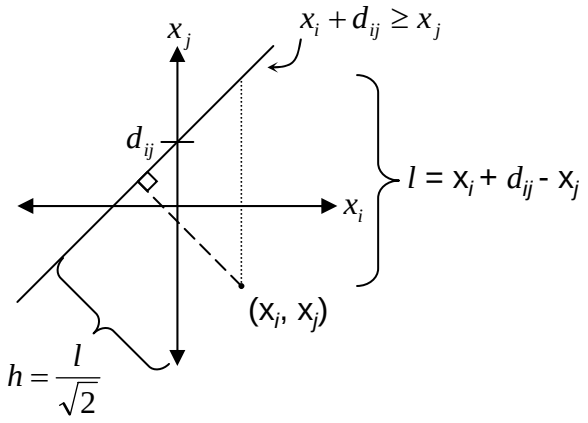

Figure 3.4: *Distance from point to line in constraint structure.*

constraint of the system has a corresponding weight within a shortest path matrix, manipulations of the constraints required for Lasserre's algorithm can be achieved by manipulating the shortest path matrix rather than the equations themselves. While doing so does not improve the overall complexity of the algorithm (it cannot because of the complexity class), it can reduce the constant factors significantly.

Lasserre's algorithm requires two basic recursive operations:

1. Measuring the distance from a particular face of a polytope to a specified point in order to establish the height of a pyramid.

2. Creating an object of dimensionality D-1 from the face of a polytope to recursively compute the volume of the base of a pyramid.

Since the constraints are all planes, the first of these operations is readily handled by the formula for finding the distance from a point to a plane. Every constraint in this temporal structure has the form $x_i + d_{ij} \geq x_j$. Let $\mathbf{x}$ be a point that satisfies

the constraint and assume that $(\mathbf{x}_i, \mathbf{x}_j)$ is its projection onto the i and j dimensions. As shown in Figure 3.4, the distance from the point $\mathbf{x}$ to the constraint associated with d_{ij} is given by $(\mathbf{x}_i - \mathbf{x}_j + d_{ij})/\sqrt{2}$.

The second operation requires taking a particular face of the polytope and examining its "volume." Before proceeding with the discussion of this operation, it is necessary to distinguish between constraints that are associated with faces of the polytope and constraints that are implied by other constraints. Figure 3.5 shows a polytope that has examples of both. Assume that initially

$$d_{ij} \geq d_{i0} + d_{0j}. \tag{3.30}$$

Combining the constraints $\phi + d_{0j} \geq x_j$ and $x_i + d_{i0} \geq \phi$ results in

$$x_i + d_{i0} + d_{0j} \geq x_j.$$

By assumption (3.30), this implies the constraint $x_i + d_{ij} \geq x_j$. When a constraint is implied by a combination of other constraints, it will be called an implied constraint.

Definition 3.3.1 (Implied Constraint)**.** *An implied constraint is a constraint that may be derived from a combination of other constraints. Such constraints are redundant.*

Theorem 3.3.5. *In the temporal graph structure, if the shortest path between two variables consists of multiple hops, then the constraint associated with the distance between those variables is an implied constraint and does not form a face of the polytope.*

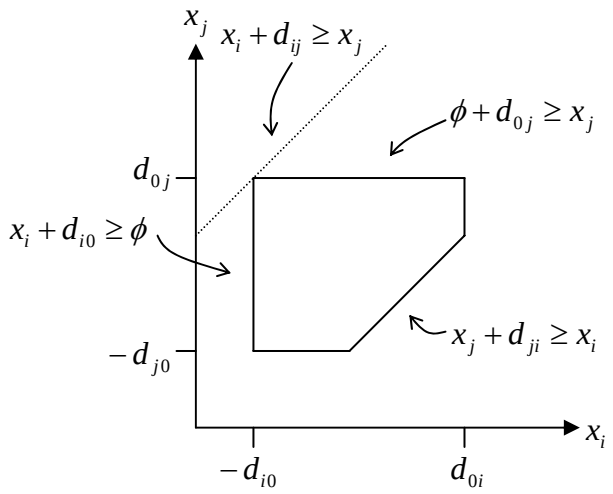

Figure 3.5: *The constraint $x_j + d_{ji} \geq x_i$ forms a face of the polytope, but $x_i + d_{ij} \geq x_j$ is degenerate, being implied by $\phi + d_{0j} \geq x_j$ and $x_i + d_{i0} \geq \phi$.*

Proof. Consider the constraint d_{ij}. Suppose that there exists $n_1, \ldots, n_k$ such that

$$d_{in_1} + d_{n_1n_2} + \ldots + d_{n_{k-1}n_k} + d_{n_kj} \leq d_{ij}. \tag{3.31}$$

$i, n_1, \ldots, n_k, j$ is a path shorter than the atomic path with distance d_{ij}. The distances $d_{in_1}, d_{n_1n_2}, \ldots, d_{n_{k-1}n_k}, d_{n_kj}$ are associated with the constraints

$$x_i + d_{in_1} \geq x_{n_1} \tag{3.32}$$

$$x_{n_1} + d_{n_1n_2} \geq x_{n_2} \tag{3.33}$$

$$\ldots \tag{3.34}$$

$$x_{n_{k-1}} + d_{n_{k-1}n_k} \geq x_{n_k} \tag{3.35}$$

$$x_{n_k} + d_{n_kj} \geq x_j \tag{3.36}$$

which can be combined as

$$x_i + d_{in_1} + d_{n_1n_2} + \ldots + d_{n_{k-1}n_k} + d_{n_kj} \geq x_j. \tag{3.37}$$

Due to (3.31), the equation above implies $x_i + d_{ij} \geq x_j$. $\square$

Let d_{ij} be the distance associated with constraint $x_i + d_{ij} \geq x_j$ and assume that this constraint is not implied. The figure formed by this face of the polytope is the intersection of the polytope with the constraint $x_i + d_{ij} = x_j$. Since the polytope already satisfies $x_i + d_{ij} \geq x_j$, this is equivalent to taking the intersection of the polytope with $x_i + d_{ij} \leq x_j$ or

$$x_j - d_{ij} \geq x_i. \tag{3.38}$$

It is possible to use this complementary constraint in order to perform the required intersection operation on the shortest path constraint structure. The following will describe the effects of adding constraint (3.38) to the system in terms of the shortest path distance matrix representation.

The immediate effect of adding constraint (3.38) is that the equality relation

$$x_i + d_{ij} = x_j \tag{3.39}$$

will hold. Assuming that initially, the constraints describe an N-dimensional polytope, constraining the original polytope to the plane (3.39) results in an $(N-1)$-dimensional polytope associated face of the polytope. After setting $d_{ji} = -d_{ij}$, it is the case that $d_{ij} + d_{ji} = 0$, so Theorem 3.3.4 applies and it is possible to completely remove either x_i or x_j without losing any information about the shape of the face assuming that the shortest path matrix is updated with the new constraint.

The constraints governing the selected polytope face are formed from constraints of the original polytope. There are two main cases to consider, the case

where the constraint from the original polytope is between variables other than x_i and x_j and the case where one of the variables is either x_i or x_j.

In the first of these cases, consider a constraint between variables x_a and x_b where $a, b \neq i, j$. Let

$$x_a + d_{ab} \geq x_b \tag{3.40}$$

be the constraint. Assume that it is known that (3.40) is not implied. No implied constraints are considered because they do not form polytope faces by Theorem 3.3.5. There are two possibilities. The first is that the constraint is not a face of the sub-polytope. This is the case when

$$d_{ab} \geq d_{aj} - d_{ij} + d_{ib}. \tag{3.41}$$

This condition tests if the d_{ab} constraint becomes implied when adding the complementary edge (setting $d_{ji} = -d_{ij}$). If condition (3.41) does not hold, then the constraint associated with d_{ab} remains non-implied in the sub-polytope. Note that updating $d'_{ab} = min(d_{ab}, d_{aj} - d_{ij} + d_{ib})$ is equivalent to performing a Floyd-Warshall update using the new edge $d_{ji} = -d_{ij}$ and performing this update on all pairs a, b suffices for updating the shortest path matrix.

Here, the second case, where one of the variables is x_i or x_j, will be considered. Let the other variable be called x_k with $k \neq i, j$. For reasons that will be explained later, it is not only necessary to keep track of the shape of the polytope face, but also exactly which original constraints the constraints in the polytope face correspond to. There are two possibilities to be considered separately.

- The constraint is associated with either d_{ik} or d_{jk}.

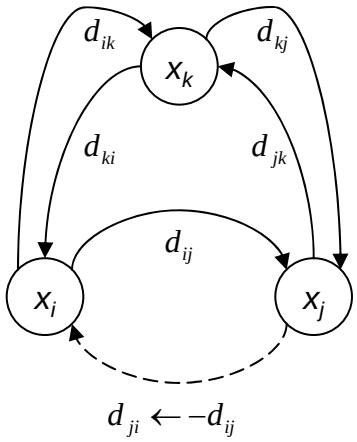

Figure 3.6: *Effect of assigning complementary distance* $d_{ji} \leftarrow -d_{ij}$

- The constraint is associated with either d_{ki} or d_{kj}.

The first of these cases corresponds graph-wise to constraints that are outgoing edges from x_i or x_j. As stated previously in this section, it is possible to remove one of the two variables without affecting the shortest path structure. It is assumed that x_i will be removed. The argument is similar if x_j is removed. It is necessary then to find the effect of adding the complementary edge on the distances and the polytope faces. The edge that is added, $d_{ji} \leftarrow -d_{ij}$, will generally create a path that is shorter than the existing path. Referring to Figure 3.6, observe the following

$$d_{ik} \leq d_{ij} + d_{jk} \tag{3.42}$$

$$d_{ik} - d_{ij} \leq d_{jk}. \tag{3.43}$$

(3.42) must hold because of the shortest path condition. (3.43) holds as a consequence. It means that after applying the complementary edge, the distance from x_j

to x_k becomes $d_{ik} - d_{ij}$ so the constraint becomes

$$x_j + d_{ik} - d_{ij} \geq x_k. \tag{3.44}$$

It is still necessary to determine what initial constraint this new constraint derives from.

There are two potential candidate constraints. The first is d_{ik}. This occurs whenever the inequality in (3.42) is strict

$$d_{ik} < d_{ij} + d_{jk}.$$

This is depicted in Figure 3.7. The constraint $x_i + d_{ik} \geq x_k$ is a constraint of the polytope face if it is not an implied constraint and the above condition is satisfied. It will be associated with the new constraint (3.44).

The second candidate is the constraint

$$x_j + d_{jk} \geq x_k. \tag{3.45}$$

If equality is achieved in (3.42)

$$d_{ik} = d_{ij} + d_{jk}$$

then $x_j + d_{jk} \geq x_k$ may be a constraint of the polytope face. This is depicted in Figure 3.3.3.4. If constraint (3.45) is not implied then it is associated with the new constraint (3.44).

If neither of the candidates satisfy the required conditions above, then the new constraint (3.44) is in fact an implied constraint.

The argument is similar in the case of the constraints associated with d_{ki} and d_{kj} so only the result will be described here. As before, it will be possible to remove

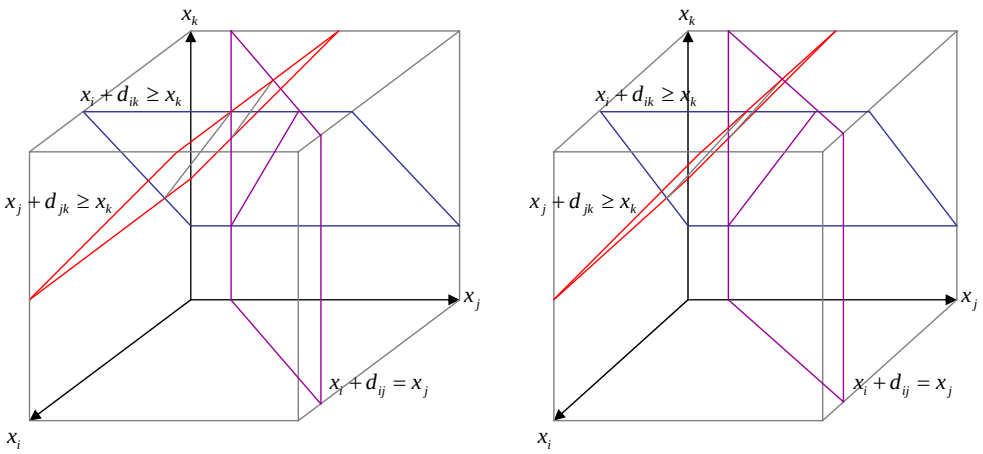

Figure 3.7: *3D cross-eye stereographic illustration: view left image in right eye and right image in left eye.* $d_{ik} < d_{ij} + d_{jk}$. *Consequently, the constraint* $x_i + d_{ik} \geq x_k$ *dominates* $x_j + d_{jk} \geq x_k$ *within the plane* $x_i + d_{ij} = x_j$. *In this case, the constraint within the polytope face,* $x_j + d_{ik} - d_{ij} \geq x_k$, *corresponds with the original constraint* $x_i + d_{ik} \geq x_k$. *Note that* $x_j + d_{jk} \geq x_k$ *is redundant within the plane* $x_i + d_{ij} = x_j$.

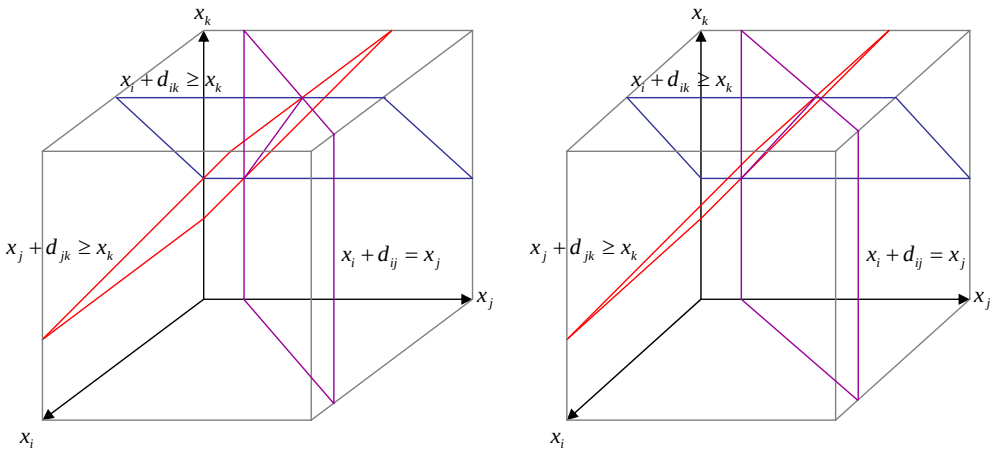

Figure 3.8: *3D cross-eye stereographic illustration: view left image in right eye and right image in left eye.* $x_i + d_{ik} \geq x_k$ *is implied by* $x_j + d_{jk} \geq x_k$ *and* $x_i + d_{ij} \geq x_j$*. In this case, the polytope face in the plane* $x_i + d_{ij} = x_j$ *has* $x_j + d_{jk} \geq x_k$ *as a constraint.* $x_i + d_{ik} \geq x_k$ *is implied and therefore degenerate.*

one of x_i or x_j and it is assumed that x_i will be removed. Referring to Figure 3.6, it is clear that the weights of incoming edges to x_j will never be affected by the addition of the complementary edge because the added edge is outgoing from x_j. It is still necessary to determine which of the edges (x_k, x_i) or (x_k, x_j) is associated with the resulting constraint $x_k + d_{kj} \geq x_j$. For reasons similar to the above, in the case of strict inequality,

$$d_{kj} < d_{ki} + d_{ij},$$

the associated edge is (x_k, x_j) if that edge is not implied. In the case of equality,

$$d_{kj} = d_{ki} + d_{ij},$$

the associated edge is (x_k, x_i) if that edge is not implied. If none of the above apply then no edge is associated with the resulting constraint.

There are a number of optimizations that apply to this computation. The most important of which is caching of sub-volumes. Each sub-volume that is formed is the result of taking the intersection of the original polytope with a set of its enclosing planar constraints. It turns out that this recursive process causes the same volumes to be generated again and again. In fact, each sub-volume is generated exactly $N!$ times where N is the number of constraints taken in intersection to produce the sub-volume. Since the volume of a sub-volume is invariant, it is possible to cache these values based on the set of planes used to produce them, reducing the amount of computation needed.

There is a trade-off to be considered in caching. The size of the cache grows quickly as a function of the number of dimensions. Performance of cache lookups tends to degrade as the cache gets larger and larger generally $O(N \log N)$, even if hash tables are used. Since volumes become easier to compute as the dimensionality decreases, and each level of the recursion decreases the dimensionality by one, at some point it becomes more efficient to compute a volume directly than to perform a cache lookup for that volume. In this particular implementation, that occurs at 3 dimensions. For any volume of 3 dimensions or less, it is more efficient to recompute the volume than it is to cache. The efficiency may be further improved by unrolling the computation of the lower dimensional volumes. Every 2-dimensional volume has the form depicted in Figure 3.3.3.4. The formula expressing its volume is

$$(d_{i0} + d_{0i})(d_{j0} + d_{0j}) - \frac{(d_{0j} - d_{ij} + d_{i0})^2}{2} + \frac{(d_{0i} - d_{ji} + d_{j0})^2}{2}.$$

An optimization that has to deal with the geometry of Lasserre's algorithm is

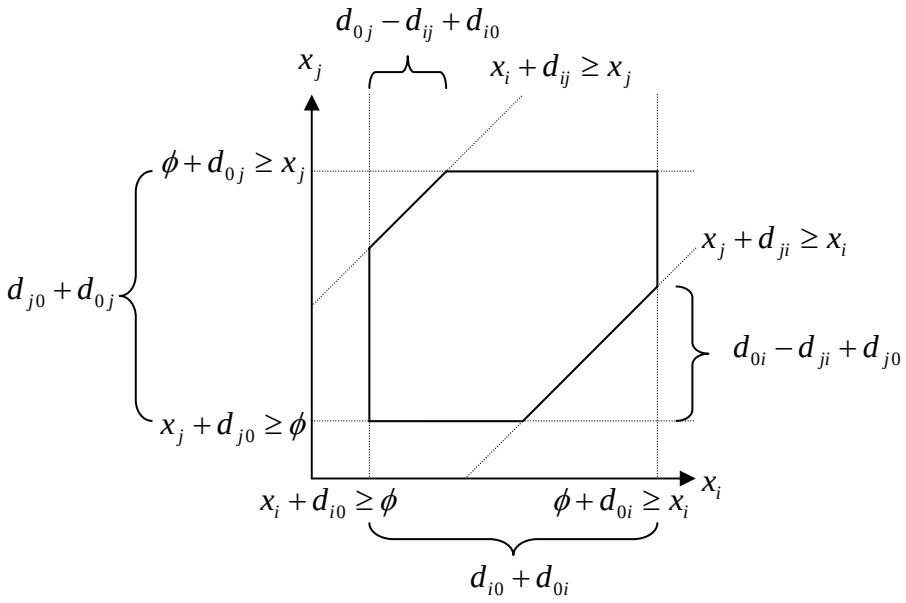

Figure 3.9: *A representative 2D volume is shown. Its volume (area) can be decomposed into a main rectangular region subtracting two triangular regions.*

shifting the anchor point to a vertex of the polytope. Faces adjacent to this vertex then have zero height and it is unnecessary to compute their volumes. This can be used to reduce the number of computations necessary. In this implementation, the anchor point is always chosen to be the intersection of all constraints of the form $x_i + d_{i0} \geq \phi$, which yields the point $(-d_{10}, -d_{20}, \ldots, -d_{n0})$ as the anchor (where n is the number of dimensions).

Execution times for a Java implementation based on these ideas is compared to Vinci[13], an optimized implementation of Lasserre's algorithm in C in Figure 3.3.3.4. The numbers are the average over 10 runs of randomly generated N-dimensional volumes. The Java times start out greater than the times for Vinci because the executions include JVM startup times and JIT compilation. Without

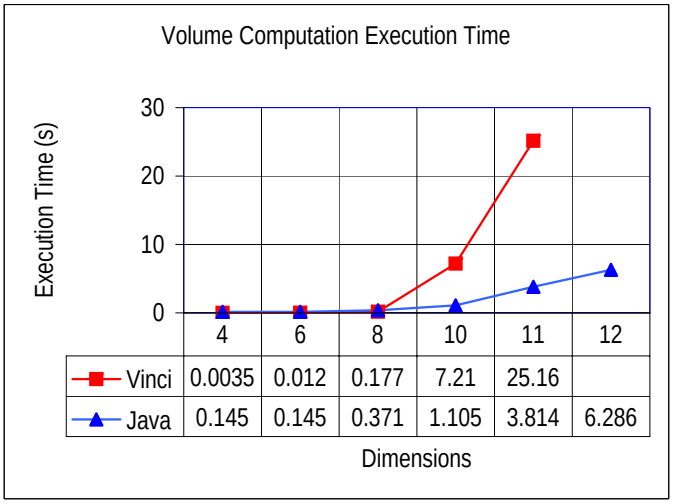

Figure 3.10: *Execution times vs number of dimensions. No data point given for 12 dimensions in Vinci because it crashes.*

the overhead, it is expected that the Java version would be faster. The execution times are provided for an AMD Athlon(tm) 64 X2 Dual Core Processor 4200+ on a machine with 2 GB of RAM running Linux.

The reason that this implementation is significantly faster than Vinci is probably due to the reduction in the branching achieved by eliminating implied constraints as shown in Section 3.3.3.4. Eliminating constraints directly reduces the number of branches needed at each stage of the recursion. This is due to the fact that constraint elimination in the shortest path structure has only quadratic complexity. In the general case which Vinci handles, constraint elimination would be equivalent to the problem of elimination of redundant inequalities from a set of linear inequalities, which is reducible to linear programming, which has a much greater complexity than the shortest-path manipulations used here.

Developing a parallelized version of this algorithm in Java was not fruitful, most likely because of the limitations of the JVM. A fine-grain parallelized version was written, but very little performance gain was realized. An experiment was performed to see how much overhead the JVM imposes on threaded programs. Running two non-interacting instances of the program in two threads in the same JVM is only 1.4 times faster than running two instances without threads. Running two instances of the JVM resulted in close to the expected 2 times performance boost.

3.3.3.5 Coupling Monte-Carlo Integration with Exact Computation

The volume computation problem can also be solved using Monte-Carlo integration. Monte-Carlo integration based on the rejection method may be used. Let R be a region for which it is possible to determine for any given point p whether or not $p \in R$. It is clear that for the constraint programs arising from the formal timing model that this is the case because it is possible to test any point p against each of the planar constraints.

The rejection method is based on taking a region R' such where $R \subseteq R'$. The region R', in addition to enclosing R must satisfy the following properties.

- The volume $V(R')$ of R is either given, or easily computed.
- There is a method for generating an arbitrary number of random samples $x_1, x_2, \ldots, x_n$ uniformly in R'.

Given these three conditions, it is possible to use the rejection method to integrate

over a volume. An estimate for the volume of region R, $\tilde{V}(R)$, can be computed as

$$\tilde{V}(R) = V(R')\frac{1}{n}\sum_{i=1}^{n}\begin{cases} 0 & : \quad x_i \notin R \\ 1 & : \quad x_i \in R \end{cases} \tag{3.46}$$

The convergence of this method dependends on the ratio $p = V(R)/V(R')$. Let c be the count of samples from R' in R. c is binomially distributed with paramters n and p so its mean is given by np and its standard deviation by $\sqrt{np(1-p)}$. The estimate of p given by c/n has expected value p and standard deviation $\sqrt{p(1-p)}/\sqrt{n}$. The standard deviation, which can be used as an indication of absolute magnitude of the error decreases with $1/\sqrt{n}$. In this case, it is also useful to consider the ratio of the error to the computed volume $(\sqrt{p(1-p)}/\sqrt{n})/p$ which equals

$$\sqrt{\frac{1-p}{np}}. \tag{3.47}$$

A plot of this function for $n = 1$ is shown in Figure 3.3.3.5. The relative error goes towards infinity as p goes to zero. What this means is that the algorithm converges more quickly if p is close to 1. In terms of the regions, this implies that performance is improved by choosing R' to cover as little volume as possible outside of R.

So in implementing Monte-Carlo integration, it is necessary to choose bounding regions $R' \supseteq R$ that are similar to R. However, it is necessary to be able to compute the volume of R' and also to be able to sample from R'. The problem is that R is typically a complex figure and to properly enclose R, it is necessary to use figures that are less complex than R itself, for example rectangular prisms. Rectangular prisms are easy to sample from and their volumes can be computed, however, there may be better choices for approximation for these partially ordered structures.

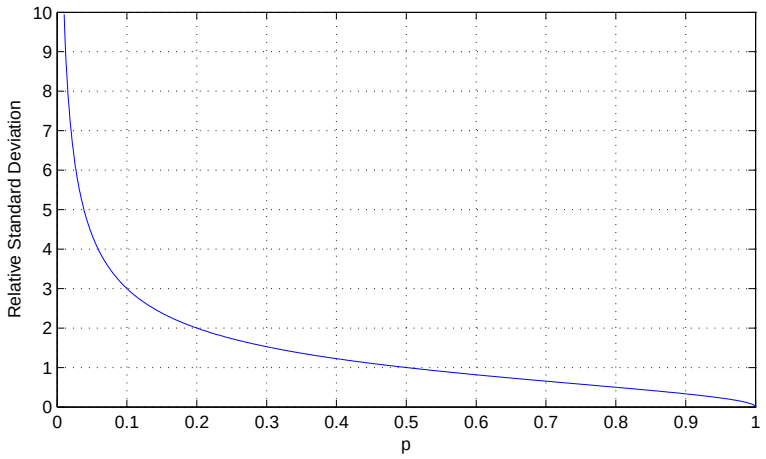

Figure 3.11: *Plot of the function* $\sqrt{\frac{1-p}{p}}$. *The relative error goes towards infinity as p approaches 0 and 0 as p approaches 1.*

The technique proposed here is to use a simplified version of the region R as the bounding region. More specifically, for a chosen subset of the variables, compute the volume exactly, and use rectangular constraints for the remaining variables for Monte-Carlo sampling. This hybrid approach will improve the convergence rate of Monte-Carlo sampling and also take advantage of exact methods for volume computation.

Ideally, the variables selected for exact computation are chosen so that the remaining variables, that are treated using rectangular Monte-Carlo sampling, are the ones best approximated by rectangular prisms. A heuristic is applied to rank the variables. Assuming that constraints of the form $x_i + d_{ij} \geq x_j$ where $i, j \neq 0$ do not interact with one another, it is possible to quickly compute the ratio of the volume of the actual figure to its enclosing rectangular prism. The variables are thereby ranked and selected in order of their ranking.

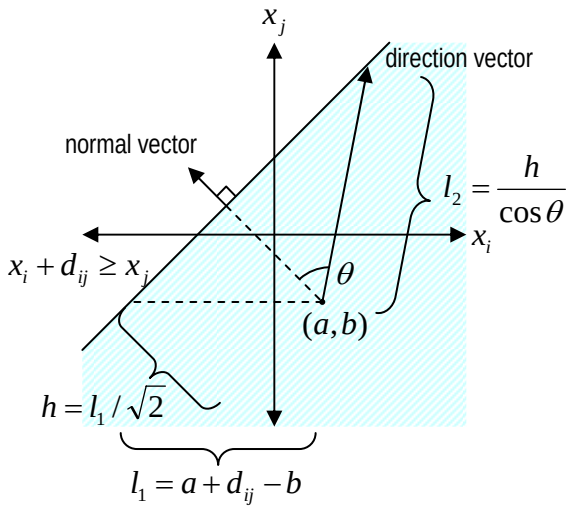

Figure 3.12: *Shows the computation of the distance from a given point* (a, b) *to the constraint* $x_i + d_{ij} \geq x_j$ *along a chosen direction vector.* $\cos(\theta)$ *can be obtained by taking the dot product of the direction vector with the normal vector.*

It is necessary to be able to sample from the region R'. The hit-and-run sampling algorithm can be used to sample uniformly from a convex body and is used for this purpose. The basic idea is to take the previous sample, choose a random chord over the body and then sample uniformly along the length of that chord. Executed repeatedly, this procedure should converge to the uniform distribution.

Choosing a random chord can be split into two parts, choosing a random direction then computing the resulting chord by intersection with the constraints forming the region. Choosing a random direction can be achieved by sampling from a multi-dimensional Gaussian distribution with zero covariance and unit variances then normalizing. Once the direction is chosen, it is still necessary to determine the length of the chord. This is achieved by testing the starting sample and direction

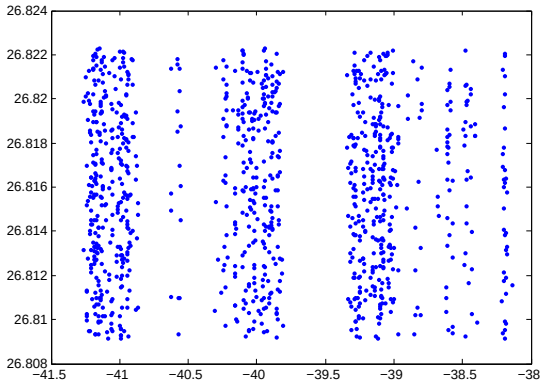

Figure 3.13: *Shows an example of hit and run sampling with poor convergence to the distribution. This should be uniform over a rectangle. The reason for this is that the vast majority of random chords are nearly vertical in this rather severe aspect ratio.*

against every planar constraint. The minimum length discovered is the correct one to use for chord length. The calculation for a particular constraint is shown in Figure 3.3.3.5. This must be done separately for the positive and negative directions to find the chord.

The problem with hit and run sampling is that it can converge arbitrarily slowly depending on the figure. In the test examples, its performance was generally poor, although the test generator was intentionally designed to produce pathological cases. Figure 3.3.3.5 illustrates an example of non-uniform sampling due to the configuration of a figure.

Nonetheless, it is possible to show that the hybrid Monte-Carlo integration that incorporates both exact volume computation and sampling outperforms Monte-Carlo integration using just rectangular sampling. This is illustrated Figure 3.3.3.5.

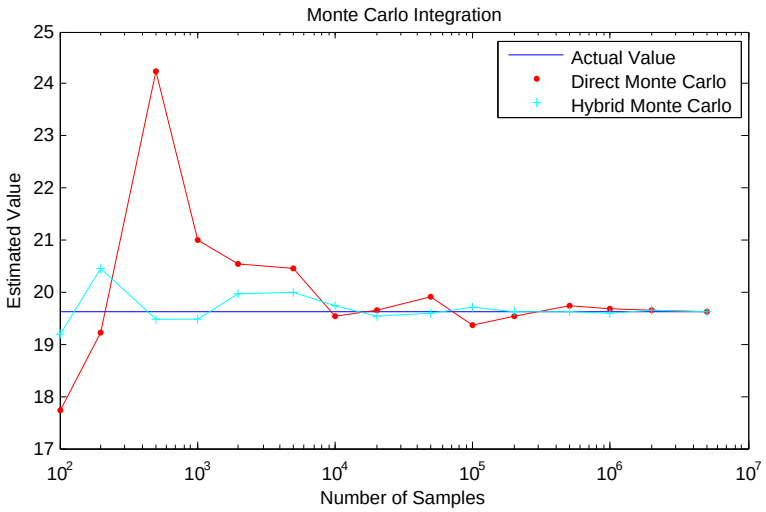

Figure 3.14: *Shows comparison of convergence of Monte-Carlo integration using plain rectangular rejection method and hybrid rejection method. The key difference is that the hybrid method results in a sampling region 1.4 times the size of the target region whereas the plain rectangular method results in a sampling region 7.5 times the size of the target. This example is for a 6-dimensional figure. Greater gains should be achievable in higher dimensions, but the hit-and-run sampling method works poorly there.*

In order to overcome the convergence issues with hit and run sampling it is possible to employ a sampling routine based on Lasserre's dimensional recursion. This sampling technique presupposes the ability to compute the volume of the region of interest using Lasserre's algorithm. In Lasserre's algorithm, a pivot point is selected and the main volume is partitioned into sub-volumes. The volume of each of the sub-volumes is known based on prior volume computation using Lasserre's algorithm. Given this, it is possible to assign a probability of drawing a sample from each of the sub-volumes. A sample from a given sub-volume can be computed by taking a sample from its base, and sampling uniformly along the ray from the base to the pivot point. Sampling from the base is a recursive application of this sampling algorithm.

3.4 Discussion

The problem with this approach is that in any network, there are a very large number of concurrent nodes. And while partial order reduction is compatible with this approach, broadcast messages used in ad-hoc networks necessitate synchronization points spanning large numbers of nodes reducing its effectiveness.

Sampling methods have been demonstrated here to be able to be able to extend the capabilities of formal methods. The next chapter will focus entirely on sampling methods within the context of an optimization framework.

Chapter 4

Trace-Based Model within an Optimization Framework : OLSR Example

4.1 Introduction

Adapting OLSR to meet changing requirements using components decomposes into two problems. The first of these problems is isolating protocol changes to particular components. The second problem is once the components that need to be changed are identified, implementing protocol modifications. In this chapter, it will be shown that components can facilitate re-evaluation of the overall system performance through re-use of models.

The new requirement will be to address a security problem in OLSR, namely malicious nodes that can arbitrarily ignore routing messages. These nodes are problematic from a security perspective if they can adversely affect routing performance because they are formally indistinguishable (in a non-probabilistic formal model) from nodes that are sporadically receiving packets due to packet loss. A model for an attacker of this form will have to be devised and tested against OLSR extensions that are meant to counteract the effects of such attacks. A significant portion of the work will be devoted to automating the construction of such an attacker in order to test the methodology.

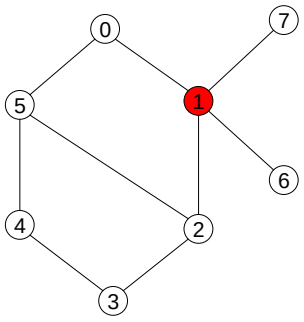

Figure 4.1: *Simple network with a malicious node.*

4.2 New Security Requirement

Figure 4.1 depicts a simple network with a malicious node 1. This node is permitted to ignore any message that it receives, emulating packet loss. The network has a topology where 1 will be given preference by an MPR selection algorithm [20] due to the two unique neighbors 6 and 7 that it possesses. Note that in this scenario every node other than 6 and 7 is connected by a pathway that excludes node 1. The longest of these pathways is the path from 0 to 3, which has pathlength of at least 3. This is the pathway that we will be observing for the performance metric. There may exist a direct means to compute the optimal policy for ignoring packets in this scenario, but it is not known offhand. Furthermore, if the routing protocol changes or the topology changes, the policy will need to change as well. By using policy iteration, an optimization technique, a good though not necessarily optimal controller can be automatically contructed. This can serve as a lower bound on the performance of a controller. Since the policy will be automatically determined, it will be easy to adapt it to multiple scenarios or different routing algorithms.

The malicious node 1 will require some information on which to base its decision to either ignore or accept a message. It is assumed that the malicious node can determine for each neighbor (nodes 0 and 2, since nodes 6 and 7 should never be ignored) its neighbor state for 1 and its timeout value for 1. The neighbor state is provided in the HELLO advertisements from 1 and the timeout value for 1 can be roughly inferred by the last time that 1 has sent a HELLO message.

4.2.1 Policy Iteration Formulation

The malicious node will have a decision function

$$p = \pi(\theta; v)$$

where θ is a set of observations, v is a set of parameters for the policy and $0 \leq p \leq 1$ is the decion. The decision here is whether or not to ignore a given packet and y will be interpreted as the probability of ignoring a packet. Since the particular decision maker or parameters may depend on the topology or versions of the routing protocol using different extensions, it is not trivial to determine the specific form of π to use.

This malicious node will act in the context of a simulation with a distribution $f(x; v)$. The dependency of f on the policy π is implicit and accounts for all of the dependencies of f on v. x represents a simulation trace. Let $S(x)$ be a score function that maps simulation traces x to values in $\mathbf{R}$. In the problem stated in Section 4.2, the score is the connectivity which can easily be evaluated given a simulation trace. The optimization problem can be stated as the following:

$$\max_{v} E_{x \sim f(x;v)}[S(x)]. \tag{4.1}$$

The distribution $f(x; v)$ has no closed form description, however samples can be drawn from this distribution by *simulation*. Each simulation trace is a sample from the distribution.

4.2.1.1 Cross-Entropy Optimization

The cross-entropy method of optimization[63] is a heuristic method for optimization that is amenable to being used in cases where the objective function must be estimated by running simulations. For a tutorial on this method and how it can be applied to rare-event simulation see [23]. The method is reviewed here as it applies to determining a policy for the attacker. While the cross-entropy method is originally defined in terms of rare event simulation, it serves as a useful heuristic for simulation-based optimization.

Problem (4.1) is complex because the cost function must be estimated for v by performing simulation. Since the simulation is a nondeterministic process, many simulations are needed to make a cost evaluation for any choice of v. Many optimization problems can be made simpler by exploiting problem specific structure. However, since this framework is being proposed as a means to evaluate many different versions of a routing protocol, it is undesirable to impose structure on the problem since different versions could have very different structures. Nevertheless the cross-entropy method provides an optimization heuristic that can be applied to this problem. The following is a derivation of the heuristic.

Let μ be the initial parameter for the system such that the cost is given by

$$E_{x\sim f(x;\mu)}[S(x)]. \tag{4.2}$$

Let γ be some threshold value and define the indicator function

$$I_{\{S(x)\geq\gamma\}} = \begin{cases} 1 & S(x) \geq \gamma \\ 0 & S(x) < \gamma \end{cases}. \tag{4.3}$$

Assume in the following that γ is chosen such that strict inequality holds in the following

$$1 > \int I_{\{S(x)\geq\gamma\}} f(x;\mu)dx > 0. \tag{4.4}$$

This means that γ is such that there is a nonzero probability that $S(x) \geq \gamma$ and a nonzero probability that $S(x) < \gamma$.

Define a new density function $g_\gamma(x;\mu)$ in terms of $f(x;\mu)$ that admits only x values that pass the indicator function:

$$g_\gamma(x;u) = \frac{I_{\{S(x)\geq\gamma\}} f(x;\mu)}{\int I_{\{S(x)\geq\gamma\}} f(x;\mu)dx}. \tag{4.5}$$

Define also the complementary density function $h_\gamma(x;\mu)$ that selects only x values for which $S(x) < \gamma$:

$$h_\gamma(x;u) = \frac{I_{\{S(x)<\gamma\}} f(x;\mu)}{\int I_{\{S(x)<\gamma\}} f(x;\mu)dx}.$$

The following holds because every value that is averaged over is greater than or equal to γ :

$$E_{x\sim g_\gamma(x;\mu)}[S(x)] \geq \gamma. \tag{4.6}$$

Similarly:

$$E_{x\sim h_\gamma(x;\mu)}[S(x)] < \gamma. \tag{4.7}$$

Thus:

$$E_{x\sim g_\gamma(x;\mu)}[S(x)] > E_{x\sim h_\gamma(x;\mu)}[S(x)]. \tag{4.8}$$

The above expression can be expanded as

$$\frac{\int S(x) I_{\{S(x)<\gamma\}} f(x;\mu)dx}{\int I_{\{S(x)<\gamma\}} f(x;\mu)dx} < E_{x\sim g_\gamma(x;\mu)}[S(x)]. \tag{4.9}$$

Moving the term from the right over and multiplying through by the denominator which is nonzero due to (4.4) there results:

$$\int S(x) I_{\{S(x)<\gamma\}} f(x;\mu)dx - E_{x\sim g_\gamma(x;\mu)}[S(x)] \int I_{\{S(x)<\gamma\}} f(x;\mu)dx < 0.$$

Adding $E_{x\sim g_\gamma(x;u)}[S(x)]$ to both sides:

$$\begin{aligned}\int S(x) I_{\{S(x)<\gamma\}} f(x;\mu)dx + E_{x\sim g_\gamma(x;\mu)}[S(x)](1 - \int I_{\{S(x)<\gamma\}} f(x;\mu)dx) \\ < E_{x\sim g_\gamma(x;\mu)}[S(x)]\end{aligned}$$

Noting that $\int f(x;\mu)dx = 1$ and

$$\int f(x;\mu)dx - \int I_{\{S(x)<\gamma\}} f(x;\mu)dx = \int I_{\{S(x)\geq\gamma\}} f(x;\mu)dx$$

allows the reduction

$$\begin{aligned}\int S(x) I_{\{S(x)<\gamma\}} f(x;\mu)dx + E_{x\sim g_\gamma(x;\mu)}[S(x)] \int I_{\{S(x)\geq\gamma\}} f(x;\mu)dx \\ < E_{x\sim g_\gamma(x;\mu)}[S(x)]\end{aligned}.$$

Now expanding the $E_{x\sim g_\gamma(x;\mu)}[S(x)]$ on the left results in:

$$\begin{aligned}\int S(x) I_{\{S(x)<\gamma\}} f(x;\mu)dx + \frac{\int S(x) I_{\{S(x)\geq\gamma\}} f(x;\mu)dx}{\int I_{\{S(x)\geq\gamma\}} f(x;\mu)dx} \int I_{\{S(x)\geq\gamma\}} f(x;\mu)dx \\ < E_{x\sim g_\gamma(x;\mu)}[S(x)]\end{aligned}$$

$$\int S(x) I_{\{S(x)<\gamma\}} f(x;\mu)dx + \int S(x) I_{\{S(x)\geq\gamma\}} f(x;\mu)dx < E_{x\sim g_\gamma(x;\mu)}[S(x)].$$

Applying the fact that

$$\int S(x) I_{\{S(x)<\gamma\}} f(x;\mu)dx + \int S(x) I_{\{S(x)\geq\gamma\}} f(x;\mu)dx = \int S(x) f(x;\mu)dx$$

to the above results in:

$$\int S(x) f(x;\mu)dx < E_{x\sim g_\gamma(x;\mu)}[S(x)].$$

Which is an equivalent way to state *the key cross-entropy method optimization heuristic*:

$$E_{x\sim f(x;\mu)}[S(x)] < E_{x\sim g_\gamma(x;\mu)}[S(x)]. \tag{4.10}$$

What this expression states is that using the distribution $g_\gamma(x;\mu)$ defined in (4.5), *increases* the expected score over the nominal distribution $f(x;\mu)$.

This can be thought of as providing in some way a stochastic gradient for the distribution to follow in updating the parameter v in $f(x;v)$. The mechanism for doing this depends on minimizing the difference between $g_\gamma(x;\mu)$ and the updated distribution $f(x;v)$. To quantify the distance between two distributions, the *cross-entropy* or *Kullback-Leibler* divergence is used as defined below.

$$\begin{aligned} D(g_\gamma(x;\mu), f(x;v)) &= E_{x\sim g_\gamma(x;\mu)}[\log \frac{g_\gamma(x;\mu)}{f(x;v)}] \\ &= \int \log \frac{g_\gamma(x;\mu)}{f(x;v)} g_\gamma(x;\mu)dx \\ &= \int g_\gamma(x;\mu) \log g_\gamma(x;\mu)dx - \int g_\gamma(x;\mu) log f(x;v)dx \end{aligned} \tag{4.11}$$

Minimizing this distance with respect to the parameter v is equivalent to the fol-

lowing maximization:

$$\begin{aligned}&\max_v \int g_\gamma(x;\mu) log f(x;v) dx \\ =&\max_v \int \frac{I_{\{S(x)\geq\gamma\}} f(x;\mu)}{\int I_{\{S(x)\geq\gamma\}} f(x;\mu) dx} log f(x;v) dx\end{aligned} \tag{4.12}$$

Since it is only v that is interesting, one may instead ignore the normalization term $\int I_{\{S(x)\geq\gamma\}} f(x;\mu) dx$ and just solve:

$$\begin{aligned}&\max_v \int I_{\{S(x)\geq\gamma\}} f(x;\mu) log f(x;v) dx \\ =&\max_v E_{x\sim f(x;\mu)}[I_{\{S(x)\geq\gamma\}} log f(x;v)].\end{aligned} \tag{4.13}$$

The above expression can be estimated in sample form. Let each $x[n]$ for $n = 1 \ldots N$ be a sample from the distribution $f(x;\mu)$. Then an approximate form for (4.13) is given by:

$$\begin{aligned}&\max_v \frac{1}{N} \sum_{n=1}^{N} I_{\{S(x[n])\geq\gamma\}} log f(x[n];v) \\ =&\frac{1}{N} \max_v \sum_{n\in\{n:S(x[n])\geq\gamma\}} log f(x[n];v) \\ =&\frac{1}{N} \max_v \log \prod_{n\in\{n:S(x[n])\geq\gamma\}} f(x[n];v)\end{aligned}$$

Since log is monotonically increasing, the same v value will maximize

$$\max_v \prod_{n\in\{n:S(x[n])\geq\gamma\}} f(x[n];v). \tag{4.14}$$

This is immediately recognizable as the maximum likelihood estimate for the v parameters given the elite samples $x[n]$ satisfying $S(x[n]) \geq \gamma$. In summary, the cross-entropy method states prescribes the following procedure for selecting a parameter v that increases the score over parameter μ.

1. Generate a sample $x[n]$ for $n = 1 \ldots N$ using distribution $f(x; \mu)$ where μ is the initial parameter.

2. Select a parameter γ such that

$$\int I_{\{S(x) \geq \gamma\}} f(x; \mu) dx \quad \text{and} \quad \int I_{\{S(x) < \gamma\}} f(x; \mu) dx$$

   are both nonempty.

3. Determine the maximum subset

$$Y = \{y[1], \ldots, y[N']\} \subset \{x[1], \ldots, x[N]\}$$

   satisfying $S(y[n]) \geq \gamma$.

4. Determine the new parameters v using maximum-likelihood estimation for generating the samples Y.

Note that the selection of γ in steps 2 and 3 above is usually implemented by setting a percentile threshold for selecting elite samples.

A phenomenon observed in the implementation of this method is that elite sampling causes instability in the algorithm. The problem is that the elite exemplar vectors completely determine the result of the algorithm. It is sometimes the case (see Section 4.3.1) that the population can be decomposed into distinct, overlapping groups. In these cases, the group with the higher score variance may be selected preferentially over the group with the higher mean score because they occupy more of the highest ranking positions. Consider the Gaussian distributions shown in Figure 4.2.1.1. The problem is that the elite samples will come predominantly from the

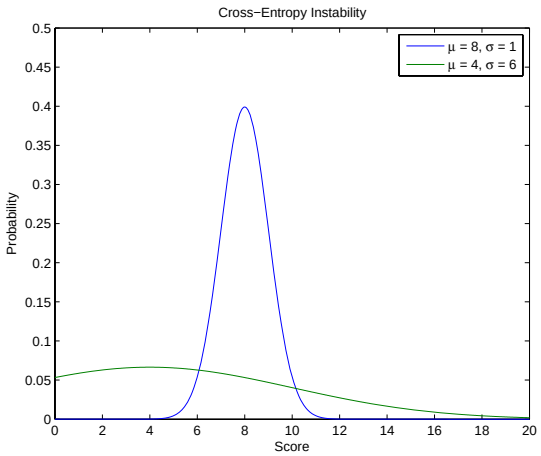

Figure 4.2: *Elite sampling instability. Elite samples from a distribution that is a composite of these two will be dominated by the distribution with the lower mean but higher variance.*

distribution with the lower mean but higher variance. So as the algorithm converges towards higher scores, the properties of the elite samples will become more and more like the distribution with the lower mean. At some point, the scores will fall back to the distribution with the lower mean. The distribution with the higher mean will not be a stable equilibrium point.

One way to ameliorate this behavior is to reduce the rigidity of the elite sampling. Any mechanism for choosing samples that satisfies (4.10) can be used. One possibility is to use a score-weighted sample rather than a purely elite sample. Another option is to keep some of the previously selected samples and combining those samples with high scoring samples in the next training epoch. Numerous other heuristic schemes adhering to the principle in (4.10) exist[63].

A remarkable property of the cross-entropy method is that it gives mathe-

matical justification to a very simple algorithm that can be applied in instances where very little is known about the structure of the cost function. It should be noted that for many distributions, explicit formulas exist for performing the maximum-likelihood parameter estimates. That is usually considered a strength of the cross-entropy optimization method, but it will not be possible here in the attack generation problem. Instead, the maximum-likelihood parameter estimate will be treated as a sub-optimization problem.

4.2.1.2 Multi-Layer Perceptrons

In Section 4.2.1.1, the cross-entropy method was reviewed as a technique for solving the overall optimization problem. However, step 4 of the algorithm does not state explicitly how the maximum-likelihood parameter estimate should be performed. In this section, a function class for the decision-maker π and a means for updating π according to the elite samples Y is provided.

As in Section 4.2.1.1, the main problem may be stated as

$$\max_{v} E_{x\sim f(x;v)}[S(x)].$$

This is dependent on v through a decision-maker

$$p = \pi(\theta; v).$$

The decision-maker π acts multiple times in each simulation trace x in the following manner. As the simulation proceeds, the malicious node will be presented with a series of decision points. At each decision point i, a number of observations θ_i will

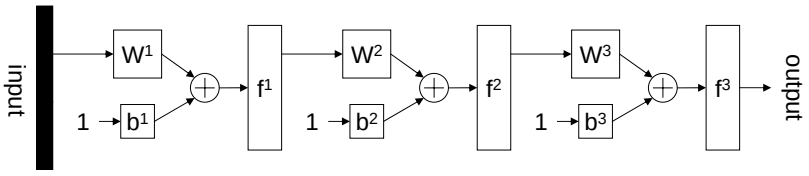

Figure 4.3: *Multi-layer perceptron.*

be presented to the decision-maker and it will choose a probability p_i with which to accept or ignore a particular packet. The decision-maker requires no additional inputs and is assumed to be parameterized by a vector of parameters v.

One candidate family of functions for the decision-maker π is the multi-layer perceptron (MLP) shown in Figure 4.2.1.2. Given a two layer MLP, the first layer having a sigmoid transfer function and the second layer having a linear transfer function, it has been shown that almost any function can be approximated to arbitrary accuracy [37] using a sufficient number of units in the first layer. Therefore, the form of the solution in this problem is chosen to be a two-layer multilayer perceptron.

The parameters for the MLP will be optimized within the training framework provided by the cross-entropy method described in Section 4.2.1.1. The training data will consist of a set of simulation runs Y. It is assumed for simplicity that the decisions are all independent of one another and extract a set of training pairs (θ_i, d_i) for $i = 1 \ldots N$ from all of the simulation runs where θ_i is the input to the decision-maker and $d_i \in \{0, 1\}$ is the decision made for that input in that simulation. N is the total number of decisions made over each of the simulation runs in Y. The objective of the optimization will be to perform a maximum-likelihood estimate of

the parameters v that are most likely to produce the set of training pairs (θ_i, d_i) for $i = 1 \ldots N$.

The following argument can be used to show that the optimal parameter selection in this problem may be accomplished by minimizing the sum of the error squared. Let p be the probability that $d = 1$ (with $1 - p$ as the probability that $d = 0$), for some choice of θ. Then

$$\begin{aligned} & E[(d - \pi(\theta; v))^2] \\ =& E[d^2] - 2E[d]\pi(\theta; v) + E[\pi(\theta; v)^2]. \end{aligned}$$

Substituting $E[d] = (p)(1) + (1 - p)(0) = p$ and $E[d^2] = (p)(1^2) + (1 - p)(0^2) = p$ then simplifying yields

$$p - 2p\pi(\theta; v) + \pi(\theta; v)^2.$$

The value of $\pi(\theta; v)$ that minimizes the above expression can be found by taking the derivative with respect to $\pi(\theta; v)$ and setting it to zero.

$$\begin{aligned} -2p + 2\pi(\theta; v) &= 0 \\ \pi(\theta; v) &= p \end{aligned}$$

What this shows is that it is reasonable to minimize the square error with the $\{0, 1\}$ valued decision variables d_i in

$$\min_v \sum_{i=1}^{N} (d_i - \pi(\theta_i; v))^2 \tag{4.15}$$

and interpret the resulting values of $\pi(\theta; v)$ as probabilities in Bernoulli trials. What remains to be shown is how to perform the minimization in (4.15).

There are well known techniques for finding parameters for MLPs such as the v parameters in $\pi(\theta; v)$. A good description of the Matlab Neural Networks Toolbox is given in [33]. The formulas for performing Levenberg-Marquardt optimization (described in [61]) on multi-layer perceptrons will be reviewed here. A few definitions will be required, refer also to Figure 4.2.1.2.

N : number of training samples.

L : number of layers in the perceptron.

D^k : the length of vector after stage k for $k = 1 \ldots L$.

$(x[n], y[n])$: training samples for $n = 1 \ldots N$. $x[n]$ denotes the input and $y[n]$ denotes the output. (These correspond with the (θ_i, d_i) pairs described previously.)

The equations that govern the perceptron may be written as follows. Initially define

$$x^0[n] = x[n] \tag{4.16}$$

where $x[n]$ is the input to the perceptron corresponding to training output $y[n]$. The rest of the behavior may be defined recursively for $l = 1 \ldots L$, $m = 1 \ldots D^l$:

$$\begin{aligned} x^l_m[n] &= f^l([(W^l)^T x^{l-1}[n] + b^l]_m) \\ &= f^l((W^l_{*m})^T x^{l-1}[n] + b^l_m). \end{aligned} \tag{4.17}$$

The notation W_{*m} denotes the m^{th} column of W. The squared error for input pair $(x[n], y[n])$ can thus be written

$$(x^L[n] - y[n])^T (x^L[n] - y[n]).$$

The goal is to find W^l, b^l for $l = 1 \dots L$ that minimizes the sum of the training error

$$\sum_{n=1}^{N} (x^L[n] - y[n])^T (x^L[n] - y[n]).$$

It is convenient to define the above in terms of an aggregated, homogenized parameter vector v. Let v be a vector of the parameters, such as

$$v = \left[W^1_{1*}, \dots, W^1_{D^0 *}, (b^1)^T, \dots, W^L_{1*}, \dots, W^L_{D^{L-1}*}, (b^L)^T \right]^T. \tag{4.18}$$

The particular ordering of the parameters does not matter as long as the mapping between the parameter and its position in v is accounted for. Then it is possible to define

$$F(v) = \sum_{n=1}^{N} (x^L[n] - y[n])^T (x^L[n] - y[n]). \tag{4.19}$$

where the dependency of $x^L[n]$ on v is implicit and it is clear that $y[n]$ does not depend on v at all.

The problem is to minimize $F(v)$ with respect to the parameter vector v. This is a nonlinear optimization which can be troublesome to solve. One simple approach towards solving this problem is to use steepest gradient descent. In this method, the parameters are updated according to

$$\Delta v = -\alpha \nabla_v F(v) \tag{4.20}$$

where $\alpha > 0$ is the learning rate. The idea is to move the parameters in the direction that most quickly decreases the error. This approach is called *backpropagation of error* (described in [33]).

The gradient $\nabla_v F(v)$ can be computed from the definition of $F(v)$.

$$\begin{aligned}\nabla_v F(v) &= \nabla_v \sum_{n=1}^{N} (x^L[n] - y[n])^T (x^L[n] - y[n]) \\ &= \nabla_v \sum_{n=1}^{N} \sum_{m=1}^{D^L} x_m^L[n]^2 - 2x_m^L[n] y_m[n] + y_m[n]^2 \\ &= 2 \sum_{n=1}^{N} \sum_{m=1}^{D^L} (x_m^L[n] - y_m[n]) \nabla_v x_m^L[n] \end{aligned} \tag{4.21}$$

In (4.21) above, the error term $(x_m^L[n] - y_m[n])$ can be computed given $(x[n], y[n])$ from (4.16) and (4.17). The other term $\nabla_v x_m^L[n]$ will require some further derivation. The expression $\nabla_v x_m^L[n]$ is actually the per training input sensitivity of the m^{th} output of the perceptron on parameters v. The terms of v are the weights W_{ij}^l and bias offsets b_j^l for each layer $l = 1 \ldots L$.

The individual gradient terms $\frac{\partial}{\partial v_\gamma} x_m^L[n]$ in the case where v_γ corresponds with $W_{ij}^{l_2}$ are given by the recursion below for $L \geq l_1 > l_2 \geq 1$.

$$\frac{\partial x^{l_1}[n]}{\partial W_{ij}^{l_2}} = \frac{\partial x^{l_1}[n]}{\partial x^{l_1-1}[n]} \frac{\partial x^{l_1-1}[n]}{\partial W_{ij}^{l_2}}. \tag{4.22}$$

Similarly in the case where v_γ corresponds with $b_i^{l_2}$,

$$\frac{\partial x^{l_1}[n]}{\partial b_i^{l_2}} = \frac{\partial x^{l_1}[n]}{\partial x^{l_1-1}[n]} \frac{\partial x^{l_1-1}[n]}{\partial b_i^{l_2}}. \tag{4.23}$$

To complete these recursions, it's necessary to compute the Jacobians $\frac{\partial x^{l_1}[n]}{\partial x^{l_1-1}[n]}$ and also evaluate the cases where $l_1 = l_2$. First, the Jacobian calculation: the following proceeds from equation (4.17) for $l = 1 \ldots L$.

$$\begin{aligned} x_m^l[n] &= f^l((W_{*m}^l)^T x^{l-1}[n] + b_m^l) \\ \frac{\partial x_m^l[n]}{\partial x_i^{l-1}[n]} &= \dot{f}^l((W_{*m}^l)^T x^{l-1}[n] + b_m^l) W_{im}^l \\ \nabla_{x^{l-1}[n]} x_m^l[n] &= \dot{f}^l((W_{*m}^l)^T x^{l-1}[n] + b_m^l) W_{*m}^l \end{aligned} \tag{4.24}$$

It's possible to express the Jacobian in terms of the gradient expression above (4.24).

For $l = 1 \dots L$:

$$\frac{\partial x^l[n]}{\partial x^{l-1}[n]} = \begin{bmatrix} \nabla_{x^{l-1}[n]} x_1^l[n] \\ \vdots \\ \nabla_{x^{l-1}[n]} x_{D^l}^l[n] \end{bmatrix} \tag{4.25}$$

Now that the Jacobian is computed, all that remains to solve equations (4.22) and (4.23) is to evaluate the case where $l_1 = l_2$.

$$\begin{aligned} \frac{\partial x_m^l[n]}{\partial W_{ij}^l} &= \frac{\partial}{\partial W_{ij}^l} f^l((W_{*m}^l)^T x^{l-1}[n] + b_m^l) \\ &= \begin{cases} \dot{f}^l((W_{*m}^l)^T x^{l-1}[n] + b_m^l) x_i^{l-1}[n] & \text{if } j = m \\ 0 & \text{if } j \neq m \end{cases} \end{aligned} \tag{4.26}$$

The expression above can be rewritten in vector form as

$$\frac{\partial x^l[n]}{\partial W_{ij}^l} = \dot{f}^l((W_{*j}^l)^T x^{l-1}[n] + b_j^l) x_i^{l-1}[n] \cdot \mathbf{e}_j \tag{4.27}$$

where $\mathbf{e}_j$ denotes the j^{th} unit vector.

Similarly, it is necessary complete the evaluation of equation (4.23) under the condition $l_1 = l_2$.

$$\begin{aligned} \frac{\partial x_m^l[n]}{\partial b_i^l} &= \frac{\partial}{\partial b_i^l} f^l((W_{*m}^l)^T x^{l-1}[n] + b_m^l) \\ &= \begin{cases} \dot{f}^l((W_{*m}^l)^T x^{l-1}[n] + b_m^l) & \text{if } i = m \\ 0 & \text{if } i \neq m \end{cases} \end{aligned} \tag{4.28}$$

The above expression can also be written in vector form as

$$\frac{\partial x^l[n]}{\partial b_i^l} = \dot{f}^l((W_{*i}^l)^T x^{l-1}[n] + b_i^l) \cdot \mathbf{e}_i \tag{4.29}$$

In summary, to find $\nabla_v x_m^L[n]$, which is required for the gradient $\nabla_v F(v)$ (4.21):

- Compute all the $x^l[k]$ values for $l = 1 \ldots L$ using equations (4.16) and (4.17).
- Use equations (4.24) and (4.25) to compute the Jacobian matrices $\frac{\partial x^l[n]}{\partial x^{l-1}[n]}$ for $l = 2 \ldots L$.
- Use equations (4.27) and (4.29) along with the Jacobian matrices from the last step to solve the recursions (4.22) and (4.23) for the entries of $\nabla_v x_m^L[n]$.

This completes the description of the backpropagation algorithm. The backpropagation algorithm suffers from slow convergence in areas where the gradient is small. Also, it is not clear how to choose a learning rate α. The Levenberg-Marquardt [61] algorithm addresses both of these issues by using an estimate of the second order derivatives of the error function.

In the Levenberg-Marquardt algorithm, an approximate second order Taylor series expansion around the current parameter set is constructed. A local extremal point can then be searched for by setting the derivative equal to zero in this approximation. The second order Taylor series expansion of $F(v)$ is written as

$$F(v + \Delta v) \approx \frac{1}{2}\Delta v^T \nabla_v^2 F(v) \Delta v + \nabla_v F(v)^T \Delta v + F(v). \tag{4.30}$$

To evaluate the above expression, it is necessary to know both the gradient and the Hessian of $F(v)$. The gradient $\nabla_v F(v)$ is the same as the one used in the backpropagation algorithm (4.21) and can be computed the same way. The Hessian requires additional evaluation. Note that the individual i^{th} entries of this gradient vector may be written as

$$\nabla_v F(v)_i = 2 \sum_{n=1}^{N} \sum_{m=1}^{D^L} (x_m^L[n] - y_m[n]) \frac{\partial}{\partial v_i} x_m^L[n] \tag{4.31}$$

Evaluating the Hessian $\nabla_v^2 F(v)$ proceeds from the above equation as follows:

$$\begin{aligned}
\nabla_v^2 F(v)_{ij} &= \frac{\partial}{\partial v_j} \nabla_v F(v)_i \\
&= \frac{\partial}{\partial v_j} \left\{ 2 \sum_{n=1}^{N} \sum_{m=1}^{D^L} (x_m^L[n] - y_m[n]) \frac{\partial}{\partial v_i} x_m^L[n] \right\} \\
&= 2 \sum_{n=1}^{N} \sum_{m=1}^{D^L} \frac{\partial}{\partial v_i} x_m^L[n] \frac{\partial}{\partial v_j} x_m^L[n] + (x_m^L[n] - y_m[n]) \frac{\partial^2}{\partial v_i \partial v_j} x_m^L[n] \qquad (4.32)
\end{aligned}$$

In the last line above, the second order expression

$$(x_m^L[n] - y_m[n]) \frac{\partial^2}{\partial v_i \partial v_j} x_m^L[n]$$

is approximated as zero because the error terms $x_m^L[n] - y_m[n]$ will tend to cancel each other out in the sum and the second order derivatives are assumed to be relatively small. This results in the approximate Hessian

$$\nabla_v^2 F(v)_{ij} \approx 2 \sum_{n=1}^{N} \sum_{m=1}^{D^L} \frac{\partial}{\partial v_i} x_m^L[n] \frac{\partial}{\partial v_j} x_m^L[n].$$

It is apparent that this can be rewritten in matrix form as

$$\nabla_v^2 F(v) \approx 2 \sum_{n=1}^{N} \sum_{m=1}^{D^L} (\nabla_v x_m^L[n])(\nabla_v x_m^L[n])^T. \qquad (4.33)$$

Note that the above expression for the Hessian depends on $\nabla_v x_m^L[n]$ which is already computed by the backpropagation algorithm to generate the gradient $\nabla_v F(v)$. Now to find the update, optimize the quadratic expression (4.30) by setting its gradient to zero.

$$\begin{aligned}
\nabla_{\Delta v} F(v + \Delta v) &\approx \nabla_{\Delta v} \frac{1}{2} \Delta v^T \nabla_v^2 F(v) \Delta v + \nabla_v F(v)^T \Delta v + F(v) \\
&\approx \nabla_v^2 F(v) \Delta v + \nabla_v F(v) \qquad (4.34)
\end{aligned}$$

It is assumed that the second order derivatives are continuous so the Hessian is symmetric. Setting the gradient expression in (4.34) to 0 results in

$$\nabla_v^2 F(v)\Delta v + \nabla_v F(v) = 0$$

$$\Delta v = -(\nabla_v^2 F(v))^{-1}\nabla_v F(v). \tag{4.35}$$

The update rule (4.35) is not always going to be optimal because the error surface is not generally quadratic. Therefore it is possible that the update rule increases the error. The innovation introduced by Levenberg was to blend this update rule with the steepest descent algorithm by introducing a damping term μ as follows:

$$\Delta v = -(\nabla_v^2 F(v) + \mu I)^{-1}\nabla_v F(v) \tag{4.36}$$

For large values of μ this rule approaches gradient descent with a small learning rate, and for small values of μ behaves just like a Newton's method update. To apply this in training a network, the following procedure is suggested. A scaling factor α is selected having a value of about 10.

1. Start initially with a small value of μ.

2. Calculate the update Δv using the selected μ value and update the parameters accordingly.

3. If the update increases the error, revert to the previous set of parameters, multiply μ by α and go back to step 2.

4. If the update decreases the error, keep the new set of parameters.

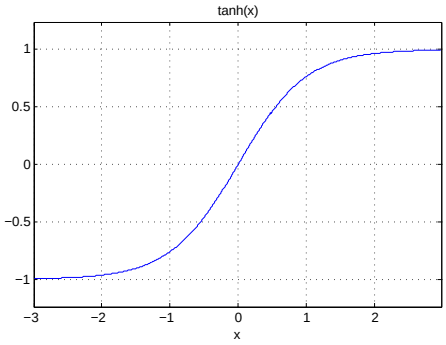

Figure 4.4: *Transfer function* $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$.

5. If error criteria are not yet satisfied, keeping the new set of parameters, divide μ by α and repeat from step 2.

This concludes the description of the Levenberg-Marquardt algorithm for MLP training.

There remains an issue of the initialization of the values of the parameters for the MLP. In this problem, there are two layers, the first layer being sigmoid and the second layer being linear. There is a geometrical interpretation that will aid in establishing initial parameters.

From equation (4.17), each neuron m for $m = 1 \dots D^1$ in layer 1 is governed by the equation

$$x_m^1[n] = f^1((W_{*m}^1)^T x^0[n] + b_m^l).$$

Choices of f^1 will be a sigmoidal function, usually something like *tanh* as shown in Figure 4.4. Observe that this function is symmetrical, roughly linear close to the origin and saturates to $1, -1$ as values increase in distance from the origin. The

argument of the transfer function

$$a_m^1(x) = (W_{*m}^1)^T x + b_m^l \tag{4.37}$$

may be interpreted as a scaled distance of the input argument x to a particular plane described by

$$\begin{aligned} a_m^1(x) &= 0 \\ W_{*m}^1 \cdot x + b_m^1 &= 0. \end{aligned} \tag{4.38}$$

This plane has normal direction $(W_{*m}^1)^T$ (the m^{th} column of W^1). Since the shortest distance between a point and a plane is the perpendicular distance, the distance between this plane and the origin can be computed as the length of $|l| = \sqrt{l \cdot l}$ such that l is parallel to the normal vector and resides in the plane.

$$\begin{aligned} (W_{*m}^1) \cdot l + b_m^1 &= 0 \\ |W_{*m}^1||l| \cos(\phi) + b_m^1 &= 0 \\ |l| &= \frac{-b_m^1}{\cos(\phi)|W_{*m}^1|} \end{aligned}$$

A positive value of $-b_m^1 |W_{*m}^1|^{-1}$ indicates $\phi = 0$ and a negative value indicates $\phi = \pi$. The placement of the plane can be on either side of the origin along the normal vector depending on ϕ. From this it is possible to define the point in the plane closest to the origin as

$$p_m^0 = \frac{-b_m^1 W_{*m}^1}{W_{*m}^1 \cdot W_{*m}^1}. \tag{4.39}$$

The distance $d_m^1(x)$ of x to plane (4.38) can be computed by taking the distance along the normal from x to any point in the plane. It is convenient to just use p_m^0

as define above.

$$\begin{aligned} d_m^1(x) &= (x - p_m^0) \cdot \frac{W_{*m}^1}{|W_{*m}^1|} \\ &= x \cdot \frac{W_{*m}^1}{|W_{*m}^1|} - \frac{-b_m^1 W_{*m}^1}{W_{*m}^1 \cdot W_{*m}^1} \cdot \frac{W_{*m}^1}{|W_{*m}^1|} \\ &= x \cdot \frac{W_{*m}^1}{|W_{*m}^1|} + \frac{-b_m^1}{|W_{*m}^1|} \end{aligned}$$

Comparing equation (4.37) to the expression above, observe that $d_m^1(x) = \frac{a_m^1(x)}{|W_{*m}^1|}$ or

$$a_m^1(x) = d_m^1(x)|W_{*m}^1|. \tag{4.40}$$

This means that by choosing the magnitude of W_{*m}^1, it is possible to scale the distance to the plane.

A number of facts have been established giving a geometric interpretation to the action of neurons in the input layer. Each neuron in the input layer defines a plane with normal direction $W_{*m}^1|W_{*m}^1|^{-1}$. The distance of this plane from the origin along the normal is given by $-b_m^1|W_{*m}^1|^{-1}$. By selecting a normal direction and an appropriately scaled offset b_m^1 it is possible to select any arbitrary plane. As shown in equation (4.40) the argument received by the transfer function will be the distance to this plane multiplied by $|W_{*m}^1|$. So in addition to being able to select an arbitrary plane, it is also possible to choose a scaling factor for the distance from that plane.

To apply this to initializing the neurons in a two-layer network the following algorithm is used. First choose a saturation threshold t for the transfer function (a value beyond which values are considered saturated, for *tanh* a value of $t = 3$ might be appropriate). Choose also a moderate value $0 < r < 1$, the ratio of training

inputs that will not saturate a particular neuron.

1. The mean and variance of each of the input parameters is computed.

2. For each neuron in the first layer, randomly sample from the multi-variate Gaussian distribution a point with the parameters determined in step 1.

3. For each neuron in the first layer, define a plane passing through the selected point such that the selected point is closest to the origin.

4. For each neuron in the first layer, compute the distances from each of the training points to the defined plane. Find the training input that is at rank rN in distance from the plane. Set the scaling factor for that neuron according to Equation (4.40) so that the distance of the chosen training input times the scaling factor is equal to the threshold t.

Following this procedure will ensure that each initialized neuron will not be operating in a completely saturated fashion with respect to the data and places the separating planes fairly regularly over the dataset.

In order to initialize the second layer, which is purely linear, it suffices to perform a least-squares fitting using the outputs of the first layer and the corresponding training outputs. Due to linearity, the least-squares fitting can be solved exactly. Let $x^1[n]$ be the outputs from the initialized first layer corresponding to inputs $x^0[n]$ and therefore outputs $y[n]$. It is clear that the neurons are all independent of each other, so it is possible to perform the minimization for each neuron independently. The minimization can be solved by setting the derivatives with respect to the weights

and the offset equal to zero. Define for $m = 1 \dots D^2$

$$\begin{aligned} F_m &= \sum_{n=1}^{N} ((W^2_{*m})^T x^1[n] + b_m - y[n]_m)^2 \\ &= \sum_{n=1}^{N} \left((\sum_{k=1}^{D^1} W^2_{km} x^1[n]_k) + b_m - y[n]_m) \right)^2 . \end{aligned} \tag{4.41}$$

Then the optimization for neuron m proceeds as follows.

$$\begin{aligned} \frac{\partial F_m}{\partial W_{im}} &= \frac{\partial}{\partial W_{im}} \sum_{n=1}^{N} \left((\sum_{k=1}^{D^1} W^2_{km} x^1[n]_k) + b_m - y[n]_m \right)^2 \\ &= 2 \sum_{n=1}^{N} \left((\sum_{k=1}^{D^1} W^2_{km} x^1[n]_k) + b_m - y[n]_m \right) x^1[n]_i \end{aligned} \tag{4.42}$$

Taking the derivative with respect to b_m yields:

$$\begin{aligned} \frac{\partial F_m}{\partial b_m} &= \frac{\partial}{\partial b_m} \sum_{n=1}^{N} \left((\sum_{k=1}^{D^1} W^2_{km} x^1[n]_k) + b_m - y[n]_m \right)^2 \\ &= 2 \sum_{n=1}^{N} \left((\sum_{k=1}^{D^1} W^2_{km} x^1[n]_k) + b_m - y[n]_m \right) \end{aligned} \tag{4.43}$$

Setting the derivatives equal to zero from equations (4.42) for $i = 1 \dots D^1$ and (4.43) results in $D^1 + 1$ equations in the $D^1 + 1$ variables $\{W^2_{1m} \dots W^2_{D^1 m}, b_m\}$. The solution to this set of equations gives the optimal least squares values for the weights and offset of neuron m. Performing this process for $m = 1 \dots D^2$ completes the initialization for the second layer of the multi-layer perceptron.

It is a good idea to initialize multiple times using different random configurations because the Leverberg-Marquardt and the steepest descent algorithms are both local optimizers. Having multiple initial points decreases the likelihood of ending in an especially bad local optimum.

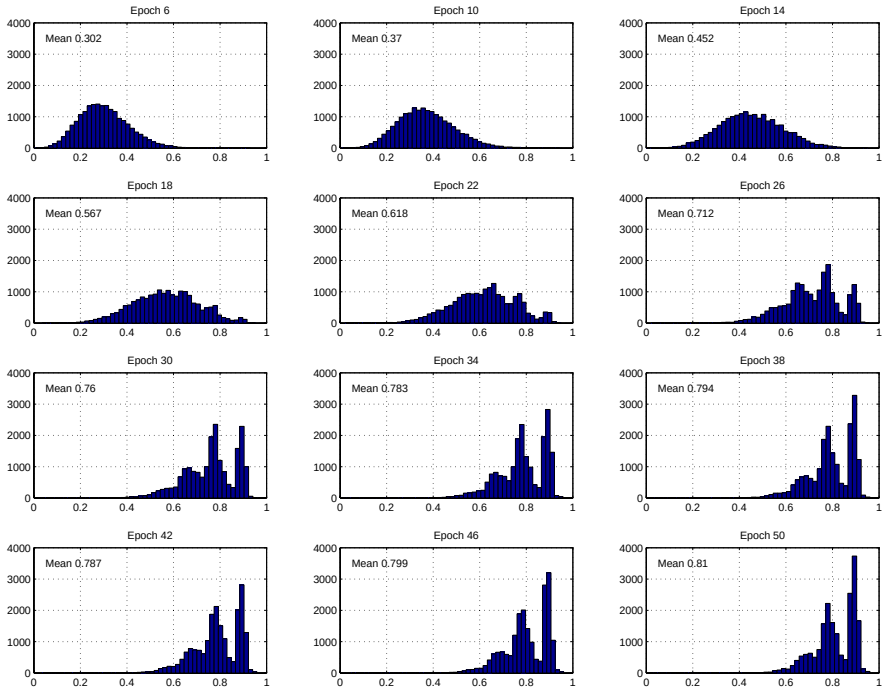

Figure 4.5: *Fraction of time Disconnected vs Training Epoch. In this training scenario, the malicious node may ignore any packet. The random initial training data for Epoch 0 begins with a mean value of 0.2.*

4.3 Initial Experiments

A sequence of experiments was performed using the described scenario and training for an attacker from the class described above. The first of these scenarios allows for an attacker that can arbitrarily ignore any HELLO or TOPOLOGY CONTROL message directed at it. The score is given by the amount of time that the nodes 0 and 3 are disconnected as a fraction of the total running time.

4.3.1 Experiment 1

Figure 4.3.1 depicts the results of training using the above described algorithm where any packets may be ignored. The initial training epoch uses a Bernoulli decision maker with $p = 0.5$ for every possible decision. Each training epoch represents 20000 1 minute long (in simulation-time) trials using a ten-neuron decision maker. The training algorithm is unable to discover the optimal law (in the mean sense) for the scenario, probabaly due to the elite sampling heuristic being used. There are a number of distinct peaks in the histogram corresponding to two different strategies employed by the decision-maker (here the attacker).

The first strategy is interference with hello messages. Refer to back to Figure 4.1 for the scenario topology. Node 1 may cause node 2's MPR selection to oscillate between itself and node 5. This makes the topology unstable and causes node 0 to be disconnected from node 3. This strategy increases the variance over the core strategy of ignoring only `TC` messages.

The second strategy, and the dominant one, is intereference with the `TC` messages. In fact, by examining the resulting neural network, we see that the output is close to zero over the entire input space. The output weight vector for the `TC` probability is given by

$$[2.0, 3.6, 1.5, -0.5, 2.6, -2.2, -1.3, -3.1, -10.0, 4.3] \times 10^{-}3$$

with offset $-6.958 \times 10^{-}5$. Since the outputs of stage 1 are bounded between -1 and 1, the output values will be very close to 0. This means that the decision maker has decided to ignore all `TC` messages. Examining the topology, we see that if this

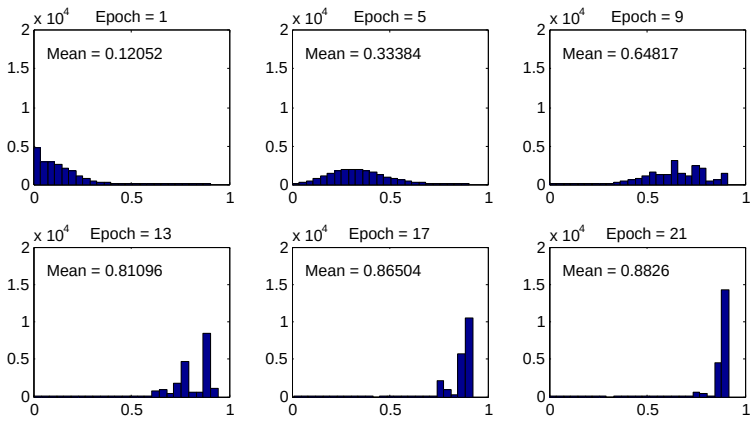

Figure 4.6: *Fraction of time Disconnected vs Training Epoch. In this training scenario, the malicious node may only ignore* `TC` *packets and must accept all* `HELLO` *messages.*

is indeed an effective strategy, causing node 0 to be unable to receive `TC` messages from node 5, resulting in disconnection. The network initially starts in a state where everything is connected so it takes approximately one `TC_HOLD_INTERVAL` of time for the connection to be lost.

Figure 4.3.1 shows what happens when only the ability to ignore `TC` messages is enabled. The result is as expected: the decision-maker converges rapidly to accepting no `TC` messages whatsoever. In Section 4.3.2, the case where only `HELLO` messages may be ignored will be discussed. The performance is actually better in the mean when the ability to ignore `HELLO` messages is disabled. A possible explanation for the reason why this is not the discovered strategy comes from the elite sampling heuristic. While the performance of the `TC` only strategy is better in the mean, it has less variance than strategies that include ignoring `HELLO` messages. This is due to the fact that the `TC_HOLD_INTERVAL` is longer than the `NEIGHBOR_HOLD_INTERVAL`.

It takes time roughly equal to TC_HOLD_INTERVAL for ignoring TC messages to cause a loss of connectivity, whereas it takes only time equal to NEIGHBOR_HOLD_INTERVAL to cause a loss of connectivity due to ignoring HELLO messages.

Ignoring HELLO messages causes a temporary disruption in connectivity because OLSR responds by rerouting traffic through an alternate path. The side effect is that the effectiveness of ignoring TC messages by the malicious node is nullified. This reduces the mean scores achievable by ignoring HELLO messages. However, on occasion, ignoring HELLO messages is beneficial because it is possible to cause disconnection through HELLO timeout, quickly reconnect and then ignore TC messages. This may produce slightly higher scores than ignoring TC messages alone because it is possible to disconnect the network in the initial TC_HOLD_INTERVAL. Figure 4.7 shows the average HELLO decision surface. There is a high probability of accepting a HELLO message when a timeout is about to occur, and a low probability when it is far from timeout. This posture maximizes the amount of control that the malicious node may exhibit over the state of its neighbor node by keeping its neighbor nodes as close as possible to the timeout state.

4.3.2 Experiment 2

In this experiment, the malicious node is allowed only to ignore HELLO messages of OLSR, but is otherwise the same as Experiment 1. Figure 4.3.2 shows the result of training. The controller here is assumed to have access to the following information (which is all locally available at the malicious node).

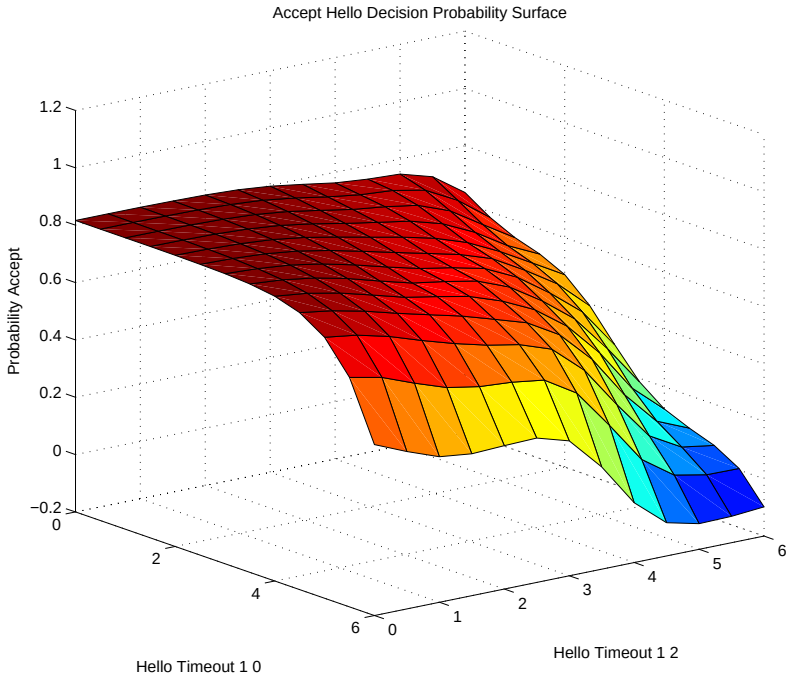

Figure 4.7: *Probability of ignoring* `HELLO` *messages vs timeout values. Values are averaged over response to* `HELLO` *messages from nodes* 0 *and* 2*, and all possible combinations of values of* `NEIGHBOR_TYPE`*.*

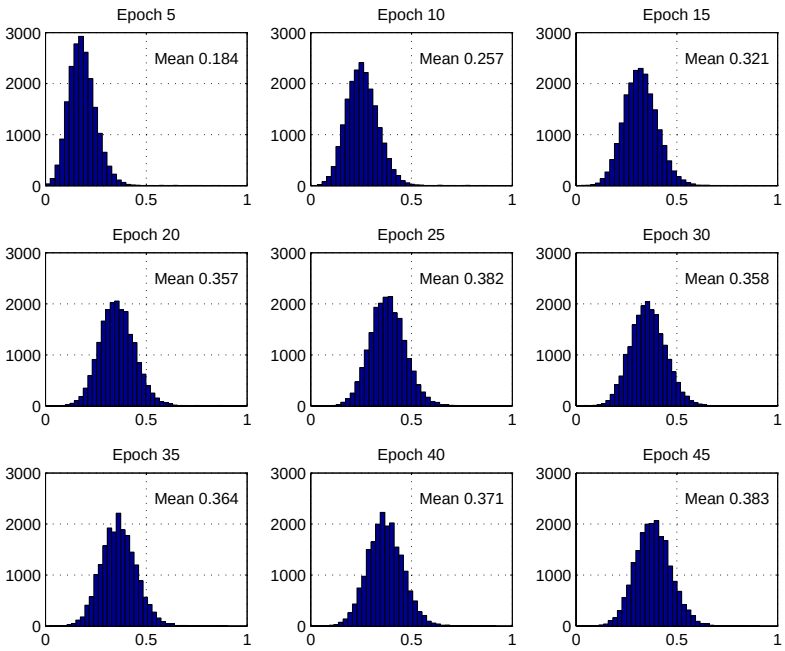

Figure 4.8: *Fraction of time Disconnected vs Training Epoch. In this scenario, the malicious node may only ignore* `HELLO` *messages.*

- The hello timeout value the malicious node keeps on its neighbors.
- The link state that the malicious node assigns to its neighbors.

These pieces of information seem insufficient for attacking the network. The results can be improved by using the following pieces of information. The effect is shown in Figure 4.3.2.

- The difference between the current time and the last time a hello message was sent by the malicious node.
- The recorded value of the neighbor state that the malicious node last time sent a HELLO message.

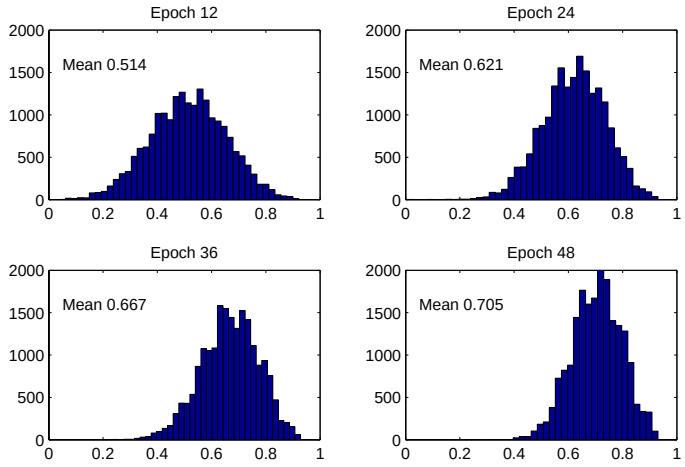

Figure 4.9: *Fraction of time Disconnected vs Training Epoch. In this scenario, the malicious node may only ignore* `HELLO` *messages. This scenario differs from the previous in that different inputs used for the decision maker.*

This is an effective attack that can be achieved by only ignoring HELLO messages.

4.3.3 Experiment 3

In this experiment, the same conditions are used as in the second experiment in Section 4.3.2, but a countermeasure described below is put in place. First, `NEIGH_HOLD_TIME` is set to zero. Second, k additional states are introduced to the OLSR state machine. These states modify the behavior in the following manner. Before processing any `HELLO` message, at least k messages in a row must have been received without any timeouts occurring. This in combination with the first modification selects strongly against links that are lossy.

Figure 4.3.3 shows the results of training an attacker on a version of the protocol that contains this countermeasure with a value of $k = 2$. The attacker is much

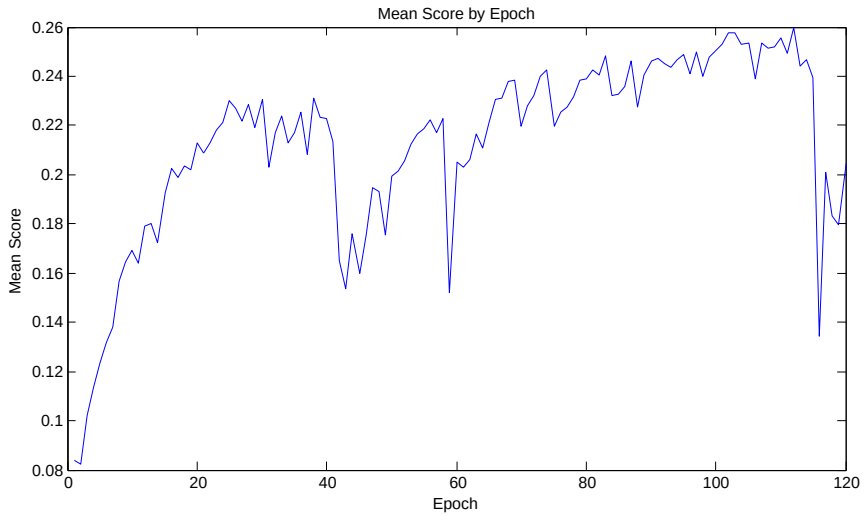

Figure 4.10: *Mean Fraction of time Disconnected vs Training Epoch. In this scenario, the malicious node may only ignore* `HELLO` *messages. OLSR has been modified to contain countermeasures to attackers that mysteriously drop packets.*

less effective in this scenario. The instability shown in the training algorithm was discussed in Section 4.2.1.1.

4.4 Discussion

A technique for using only trace-based information for optimizing an attacker is presented, demonstrating that it is possible to disrupt connectivity in a bi-connected network running OLSR using only the ability to selectively ignore certain packets. In this case, the metric for the attack is defined by the effect that the attack has on the performance of the protocol over time. A number of attacks are found, and it is shown that this method is also applicable to checking instances where attack countermeasures are in place. The fact that this is a trace-based method implies

that it will be impossible to use this to prove that a protocol is free from possible attacks.

Chapter 5

Discussion and Future Work

This dissertation presents a number of different ways to explore the state space of an ad-hoc routing protocol in the presence of an attacker. The first of these methods, presented in Chapter 2 is timed formal analysis.

By taking advantage of symbolic variable representation and partial order reduction, it is possible to explore the behavior of an ad-hoc routing protocol in the presence of a malicious node. By specifying a set of safety properties, a simple attack on the network on routing protocol AODV was demonstrated by automating the search for a violation of this set of properties. However difficulties were encountered in finding attacks once the models became more realistic or complex.

With the introduction of a fully timed-probabilistic model that is more representative of the fine grained behavior of ad-hoc routing protocols, many of the problems with formal methods become apparent. In particular, state space explosion is a problem and it is compounded by the difficulty of evaluation of the probabilities of trajectories through the state space. The evaluation of probabilistic, partially-ordered trajectories described in Chapter 3 requires the calculation of the volume of a particular type of polytope. This calculation is in the complexity class $\sharp - P$ complete. The best known algorithm for its exact computation is Lasserre's recursive decomposition. Using the particular geometry of the partial order struc-

ture, the difference bound matrix, it was shown how the complexity of the algorithm could be reduced, however not into a class other than $\sharp - P$ complete. This meant that it was not possible to apply this method towards the problem of automated attack discovery in general.

Another shortcoming of formal analysis is that even if a behavior violating a safety or a liveness condition is discovered, it is not clear that the violation is an attack. For example, if an attack occurs, the routing algorithm may have a mechanism for isolating the area affected by the attack or may be able to detect the attack and respond to it. If the routing algorithm is able to mitigate the effects of an attack, then it is hard to say formally whether the degradation in performance constitutes an attack.

Other instances, such as the attack introduced in Section 4.2 are also hard to distinguish formally from normal behavior because it merely emulates packet loss. It may be possible to develop a formal specification on the packet loss characteristics of the links in the network, but this does not appear to be a sound direction.

Instead of specifying attacks by correctness or liveness properties, Chapter 4 uses a performance metric as the criterion for discovering attacks. Since for the attack discovery problem it suffices to show that an attack exists, simulation of particular scenarios is adequate for measuring performance. The technique presented shows that it is possible to use trace based methods to find attacks and evaluate their effects.

It is possible to view trace-based methods in relation to formal methods. In formal methods, the goal is an enumeration or accounting of all possible traces of the

system. In realistically sized systems, this will not be possible. A collection of traces, being a result of a trace-based search routine is essentially a set of samples from the formal search space. If the collected traces satisfy a particular property, then existence of that property within the formal universe of possibilities is established. In large problems with timing and lengthy, highly branching traces, any computer will be overwhelmed by an enumeration task and sampling may be the only recourse available for providing a quick analysis of the problem.

The idea of using a combination of formal methods and trace based methods could not be applied specifically here to the problem of automated vulnerability discovery in ad-hoc routing protocols, but such techniques are likely applicable to problems that are more formally tractable. In Chapter 3, it was demonstrated how exact, symbolic, timing analysis could be linked with sampling methods to provide more rapidly converging approximations than sampling methods alone. State space reduction techniques such as partial order reduction and symbolic representations may be applied towards reducing the complexity of optimization problems or rare event simulation involving systems with concurrent states.

Other protocols may be tested as well as other topological configurations with many other possible sets of parameters. Additionally, testing under dynamically changing topology and testing under a large range of topologies in larger problem instances are still unaddressed problems.

Bibliography

[1] Gergely Acs, Levente Buttyan, and Istvan Vajda. Modelling adversaries and security objectives for routing protocols in wireless sensor networks. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2006.

[2] Rajeev Alur. Timed automata. In *Computer Aided Verification*, pages 8–22, 1999.

[3] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking for probabilistic real-time systems (extended abstract). In *Automata, Languages and Programming*, pages 115–126, 1991.

[4] Shah an Yang and John S. Baras. Tora: Verification, proofs and model checking. In *Proceedings of WiOpt 03: Modeling and Optimization in Mobile, AdHoc and Wireless Networks*, 2003.

[5] Anonymous. Low cost and portable gps jammer. http://www.phrack-dont-give-a-shit-about-dmca.org/phrack/60/p60-0x0d.txt, December 2002. Phrack 60.

[6] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *Proceedings of the ACM workshop on Wireless security*, pages 21–30. ACM Press, 2002.

[7] Johan Bengtsson, Kim Guldstrand Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. UPPAAL - a tool suite for automatic verification of real-time systems. In *Hybrid Systems*, pages 232–243, 1995.

[8] Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice-Hall, Inc., 1992.

[9] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Two Volume Set*. Athena Scientific, 2001.

[10] Karthikeyan Bhargavan, Davor Obradovic, and Carl A. Gunter. Formal verification of standards for distance vector routing protocols. *Journal of the ACM (JACM)*, 49(4):538–576, 2002.

[11] Graham Brightwell and Peter Winkler. Counting linear extensions is #p-complete. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 175–181, New York, NY, USA, 1991. ACM Press.

[12] Randal E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. Technical Report CS-92-160, Carnegie Mellon University, School of Computer Science, July 1992.

[13] B. Büeler, A. Enge, and K. Fukuda. Exact volume computation for Polytopes: a practical study. In *Polytopes - Combinatorics and Computation*, volume 29 of *DMV Seminar*, pages 131–154. Birkhäuser Verlag, 2000. PRO 00.12.

[14] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[15] Olaf Burkart, Didier Caucal, Faron Moller, and Bernhard Steffen. Verification on Infinite Structures. In A. Ponse J. Bergstra and S. Smolka, editors, *Handbook of Process Algebra*. Elsevier, North-Holland, 2001.

[16] L. Buttyan and I. Vajda. Towards provable security for ad hoc routing protocols, 2004.

[17] V. Chandru and J. N. Hooker. *Optimization Methods for Logical Inference*. Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 1999.

[18] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, 1999.

[19] Edmund M. Clarke and Jeannette M. Wing. Formal methods: state of the art and future directions. *ACM Computing Surveys (CSUR)*, 28(4):626–643, 1996.

[20] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr) rfc 3626, 2003.

[21] T. H. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms*. MIT Press, 1990.

[22] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge Mathematical Textbooks, 1990.

[23] P. de Boer, D. Kroese, S. Mannor, and R. Rubinstein. A tutorial on the cross-entropy method, 2004.

[24] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[25] M. E. Dyer and A. M. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.*, 17(5):967–974, 1988.

[26] E. Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of theoretical computer science (vol. B): formal models and semantics*, pages 995–1072. MIT Press, 1990.

[27] E. Allen Emerson. Model checking and the mu-calculus. In Neil Immerman and Phokion G. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, chapter 6. American Mathematical Society, 1997.

[28] Javier Esparza, David Hansel, Peter Rossmanith, and Stefan Schwoon. Efficient algorithms for model checking pushdown systems. In *Computer Aided Verification*, pages 232–247, 2000.

[29] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2,3):191–230, 1999.

[30] Jeanne Ferrante and Charles W. Rackoff. *The Computational Complexity of Logical Theories*. Lecutres Notes in Mathematics. Springer Verlag, 1979.

[31] Li Gong. A security risk of depending on synchronized clocks. *Operating Systems Review*, 26(1):49–53, 1992.

[32] J. D. Guttman. Filtering postures: Local enforcement for global policies. In *Proceedings of the 1997 Conference on Security and Privacy*, pages 120–129, Los Alamitos, May 4–7 1997. IEEE Press.

[33] Martin T. Hagan, Howard B. Demuth, and Mark Beale. *Neural network design*. PWS Publishing Co., Boston, MA, USA, 1996.

[34] Ahmed Helmy, Deborah Estrin, and Sandeep K. S. Gupta. Fault-oriented test generation for multicast routing protocol design. In *FORTE*, pages 93–109, 1998.

[35] Ahmed Helmy, Deborah Estrin, and Sandeep K. S. Gupta. Systematic testing of multicast routing protocols: Analysis of forward and backward search techniques. In *Proceedings of IEEE ICCCN*, October 2000.

[36] Gerard J. Holzmann. The model checker spin. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

[37] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[38] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks, 2002.

[39] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks, 2003.

[40] Finn V. Jensen, F.V. V. Jensen, and F. V. Jensen. *Introduction to Bayesian Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[41] R. M. Karp, M. Luby, and N. Madras. Monte-carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448, 1989.

[42] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM: Probabilistic symbolic model checker. In *Computer Performance Evaluation / TOOLS*, pages 200–204, 2002.

[43] Marta Z. Kwiatkowska, Gethin Norman, and Jeremy Sproston. Probabilistic model checking of the ieee 802.11 wireless local area network protocol. In *PAPM-PROBMIV*, pages 169–187, 2002.

[44] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.

[45] L. Lamport. Verification and specification of concurrent programs. *Lecture Notes in Computer Science*, 803:347–374, 1994.

[46] Richard Lassaigne and Sylvain Peyronnet. Probabilistic verification and approximation. In *Proceedings of the 12th Workshop on Logic, Language, Information and Computation (WoLLIC 2005)*, pages 101–114, Amsterdam, Netherlands, 2006. Elsevier.

[47] Jean B. Lasserre. An analytical expression and an algorithm for the volume of a convex polyhedron in rn. In *Journal of Optimization Theory and Applications*, volume 39, pages 363–377. Springer, 1983.

[48] Zohar Manna and Amir Pnueli. Adequate proof principles for invariance and liveness properties of concurrent programs. *Science of Computer Programming*, 4(3):257–290, December 1984.

[49] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[50] Catherine Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[51] R. Milner, J. Parrow, and J. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40,41–77, September 1992.

[52] Vishal Mittal and Giovanni Vigna. Sensor-based intrusion detection for intra-domain distance-vector routing. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 127–137. ACM Press, 2002.

[53] D. Obradovic. *Formal Analysis of Routing Protocols*. PhD thesis, University of Pennsylvania, 2001.

[54] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, San Antonio, TX, January 2002.

[55] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM (3)*, pages 1405–1413, 1997.

[56] Joachim Parrow. An introduction to the π-calculus. In Bergstra, Ponse, and Smolka, editors, *Handbook of Process Algebra*, pages 479–543. Elsevier, 2001.

[57] C. Perkins. Ad-hoc on-demand distance vector routing, 1997.

[58] Radia Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, MIT Laboratory for Computer Science, Cambridge, Massachusetts, 1988.

[59] Adrian Perrig, Ran Canetti, J.D. Tygar, and Dawn Song. The tesla broadcast authentication protocol. RSA CryptoBytes, 2002. vol. 5.

[60] Adrian Perrig and Dawn Song. A first step towards the automatic generation of security protocols. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS '00)*, pages 73–83, San Diego, CA, February 2000. Internet Society.

[61] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[62] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer, July 2005.

[63] Reuven Y. Rubinstein, Dirk P. Kroese, and Reuven Y. Rubenstein. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2004.

[64] Dawn Song, Sergey Berezin, and Adrian Perrig. Athena: A new efficient automatic checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW '99)*, pages 192–202, Washington - Brussels - Tokyo, June 1999. IEEE.

[65] James P. G. Sterbenz, Rajesh Krishnan, Regina Rosales Hain, Alden W. Jackson, David Levin, Ram Ramanathan, and John Zao. Survivable mobile wireless networks: issues, challenges, and research directions. In *Proceedings of the ACM workshop on Wireless security*, pages 31–40. ACM Press, 2002.

[66] David Thomas and Andrew Hunt. *Programming Ruby: the pragmatic programmer's guide*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[67] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[68] Shahan Yang. Modeling strands with predicate logic. Unpublished Work at University of Maryland.

[69] Shahan Yang and John S. Baras. Modeling vulnerabilities of ad hoc routing protocols. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 12–20, New York, NY, USA, 2003. ACM.

[70] Shahan Yang and John S. Baras. Causal decomposition of olsr into components and applications to performance analysis. Technical report, Institute for Systems Research and Center for Hybrid Networks, University of Maryland, January 2007.

[71] Manel Guerrero Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the ACM workshop on Wireless security*, pages 1–10. ACM Press, 2002.

[72] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.