Analyzing MANET Jamming Strategies

by

Eamon Millman
B.Eng., University of Victoria, 2007

A Thesis Submitted in Partial Fulfilment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Analyzing MANET Jamming Strategies

by

Eamon Millman
B.Eng., University of Victoria, 2007

Supervisory Committee

Dr. Stephen Neville, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Michael McGuire, Departmental Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Stephen Neville, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Michael McGuire, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Mobile Ad-hoc Wireless Networks (MANETs) present a new paradigm in which to realize a variety of communication technologies and services. The use of stochastic event-based simulation is a common approach to modelling MANET operations as part of the engineering process. To improve observations many simulations are often averaged together to produce estimations of MANET operation; however, to be statistically meaningful start-up transients must be removed, and only ergodic data averaged. These statistical issues of stationarity and ergodicity are often approached in an ad-hoc manner, if at all. This thesis presents a formal method to address these two statistical issues and applies it to the problem of quantifying MANET operation under different physical-layer jamming strategies. This demonstration illustrates the complex nature of MANET operation and the need for rigorous statistical analysis as part of the engineering process.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. S. Neville, for the opportunity to work on this challenging topic.

Michael Jarrett, for his generous feedback and constructive criticism.

Western Canada Research Grid, for providing the computing resources needed.

NSERC and the B.C. Government, for funding me with a Scholarship.

DEDICATION

To my loving wife, supportive parents, and understanding friends.

Chapter 1

Introduction

1.1 Mobile Ad hoc Networks

A mobile ad hoc network (MANET) is a collection of mobile autonomous hosts interconnected through wireless devices to form self-organizing wireless networks. In traditional access point (AP) wireless networks hosts communicate via stationary wireless devices. This allows centralized wired infrastructure to be extended within a local area through the addition of APs. In MANETs, hosts communicate in a non-centralized manner by relaying messages on-demand between each other rather than through APs. This enables MANETs to function in the absence of any infrastructure. However, the coverage area of a MANET depends on the number of hosts in the area, their motion, and their per-host communication range. The coverage area of MANETs versus AP networks is illustrated in Figure 1.1. MANETs can be integrated with AP networks to effectively extend the coverage into areas where it may be impractical or not cost effective to fully provide fixed infrastructure.

MANETs are of interest because they enable services to be delivered without the presence of AP networks. In particular, the fixed location and size of an AP's coverage area limits its ability to deal with urban environments. Commonly this is addressed by installing more APs to provide additional coverage areas linked through the wired infrastructure; however, this can become prohibitively costly due to physical or economic factors, (e.g., a lack of existing wired infrastructure or high density population areas). By operating over a dynamic topology and scaling on-demand MANETs are able to operate under conditions that APs are unable to. This presents

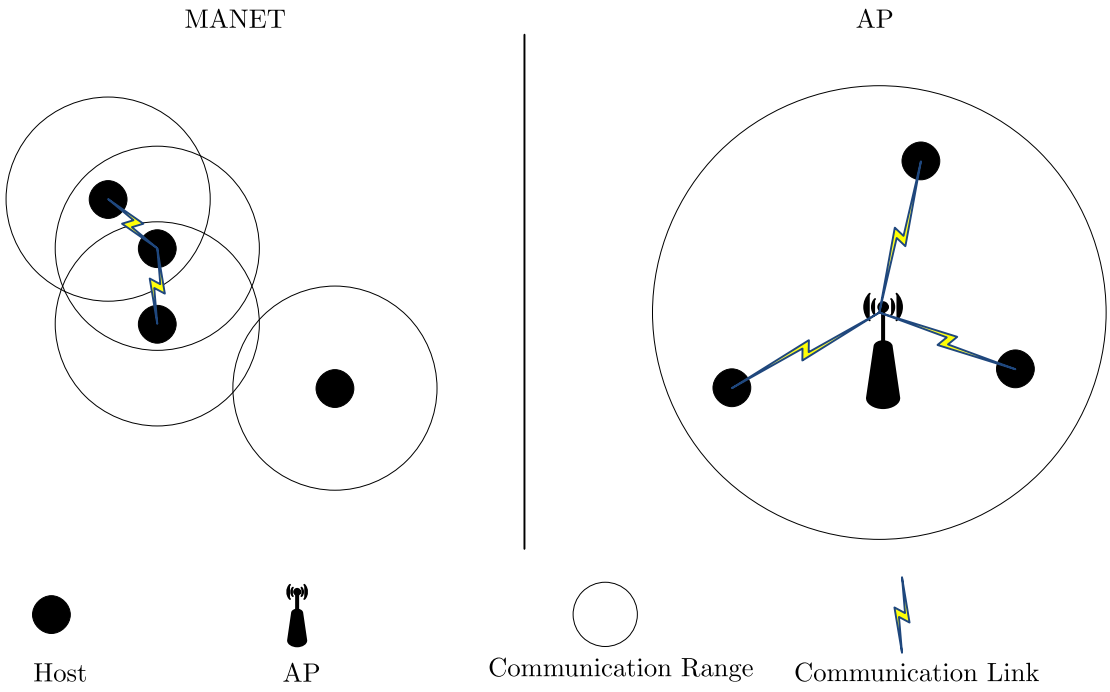


Figure 1.1: MANET and AP network coverage

a rich wireless network communication paradigm within which to address network communication problems not well serviced via standard AP-based approaches.

A number of deployments are already in use in a variety of domains including One Laptop Per Child [9] (OLPC) which leverages mesh networking, a variant of MANETs where the topology is less dynamic, in order to provide network communications in areas without infrastructure. Vehicular Ad-hoc Networks (VANETs), a high motion domain not well serviced by AP networks, are becoming more of a possibility through efforts to create driver assist safety systems in cars [21]. MANETs can also operate where infrastructure becomes non-functional due to natural disasters or social conflict.

In addition, the rapid growth of portable computing devices, (e.g., smart phones and tablets) offers a new ecosystem for MANET based services. Many of these devices contain multiple antennas and are capable of communicating over a variety of different protocols. For example: Cellular, Wi-fi, Bluetooth, etc.. This provides a rich ecosystem in which new MANET-based technologies and services can be realized.

1.2 MANET Operation

In order to deploy and provide services over MANETs the ability to engineer them to meet desired operational properties, (e.g., requirements with respect to availability, reliability, performance, etc.) becomes increasingly important. MANETs can fracture when hosts' movements cause disjoint groups to occur or when operational limitations cause a loss of communications paths between hosts. This degradation of MANET operations manifests itself as a reduction in network coverage, (i.e., availability issues), reduction of the ability to successfully communicate, (i.e., reliability issues) and increased latency or reduced capacity, (i.e., performance issues). MANETs formed by a sufficient numbers of hosts can provide more stable networks due to multiple communication routes being available between arbitrary hosts thereby mitigating these topological and operational issues.

These operational issues can surface in MANETs even if they are of sufficient density through either legitimate or malicious actions, with the former arising from the dynamic nature of MANET's topology. The primary interest of this work focuses on malicious actions. For publicly accessible MANETs there are a wide variety of possible attacks which can be performed by malicious users which require varying levels of knowledge to execute. These can be targeted at disrupting a specific service being delivered over the MANET, rendering the routing protocol non-functional, or jamming wireless communication paths. For non-public MANETs, service-level and protocol-level attacks may not be possible due to encryption or other implemented security measures.

Jamming, (i.e., disrupting the wireless communications through generation of interference) focuses on the physical-layer and can always be used against MANETs irrespective of their higher-layer security measures. This is of benefit to attackers as it provides a general purpose attack. Attack resistant MANETs first and foremost must be resistant to jamming attacks. The focus of this thesis is to observe how MANETs function under this simple attack.

1.3 Modelling

There are a number of approaches which can be applied to the problem of modelling jamming based MANET attacks, namely: a) Analytical Assessment, b) Simulation, c) Emulation, and d) Ad-Hoc testing. In general, this list is ordered by increasing fidelity

from left-to-right and decreasing controllability and per-experiment repeatability from right-to-left. While high fidelity is desirable when engineering real world systems it comes at an increased cost to repeatability and control which are tenets of the scientific method.

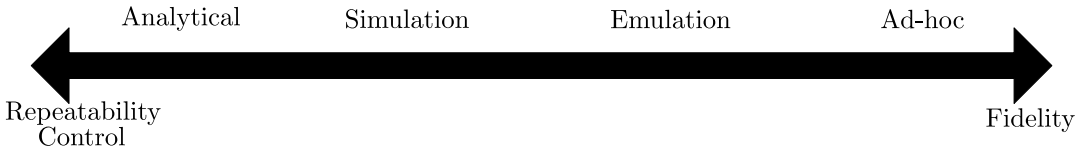


Figure 1.2: Modelling Methods

A general engineering rule of thumb in designing a solution is to move from left-to-right through these approaches. This is because solutions that are demonstrably incorrect in a preceding approach rarely, if ever, become correct in a proceeding approach. Most MANET research continues to focus on simulation-based research, and that is the approach followed in this thesis. The rationale for this choice is highlighted in the sub-sections below.

1.3.1 Analytical Assessment

Analytical assessment of MANETs preferably makes use of closed-form mathematical representations of the behaviours to be assessed. In order for the constructed model to be accurate the mathematics used must reflect the actual behaviour of the MANET. Hence, this approach is restricted to systems which are analytically tractable. Through making assumptions about the MANET's behaviour the complexity of the analytical model can be reduced. This provides tractability but limits fidelity; however, because the model is comprised of mathematical functions both repeatability and controllability are innate.

While limited to models that can be represented by tractable mathematic models, analytical assessment makes use of a number of different methods to deal with modelling problems related to MANETs and their protocols. Graph theory is often used to model MANETs when observing the performance of routing protocols [56, 35]. Other analytical models use derived equations to represent different features of the MANET operations [49]. Generally, such models focus on a narrow scope of the problem to achieve tractability, thereby making assumptions about the behaviour of the MANET.

A common assumption when using this approach to modelling is that the mathematical characteristics of the model will fully describe the MANET behaviours that are of interest. Due to the unpredictable nature of packet switching networks [50] MANETs can exhibit complex behaviours not represented by the model constructed. Moreover, the detail required to make observations about the packet-level behaviour of MANETs are often abstracted away when using analytical assessment because of their inherent complexity. This method would not provide sufficient fidelity because the focus of this work pertains to how the MANET operates in the presence of physical-layer jamming. In addition, sufficiently tractable analytical models of reasonably complex jamming scenarios are unlikely to exist.

1.3.2 Simulation

Simulation is another modelling approach which can be used to observe the operations of MANETs. As with analytical assessment the simulation created must closely approximate the real world MANET being examined [22]. This is because the aggregate behaviour of the simulation can be altered by the individual components. In order for the simulation to be accurate the level of fidelity must be high. This can potentially place resource constraints on the type of simulations that can be performed.

A common application of simulation is the study of packet-level MANET behaviours produced by defined protocols. This enables the evaluation of different transport-layer protocols [16], routing protocols [53], or even system performance in the presence of malicious users performing attacks against MANETs [3]. This extends the modelling to scenarios that are not analytically tractable. One drawback of the simulation method is the innate complexity of the models constructed when observing packet-level features of MANET operation.

This is partly mitigated by the availability of reusable simulation components within the MANET research community which enables rapid construction of complex models with tested implementations. This enables highly detailed and accurate models to be built without the expense of real world components. While using pre-existing simulation components can simplify creation of the simulation model, their implementation can limit the scale of the model, (e.g., number of hosts in the MANET or amount of network traffic) due to runtime or other resource constraints.

Simulation makes use of analytical representations of individual components to retain the level of repeatability and control required by the scientific method. Within

the MANET research community models are often built within a discrete-time event-based simulation engine. Such simulations are stochastic in nature and make use of one or more pseudo-random number streams to provide repeatability. The wide use of simulation and the availability of related tools promotes the use of this approach when modelling MANETs being attacked at the physical-layer.

1.3.3 Emulation

Unlike analytical assessment and simulation, which are approximations, emulation makes use of real world components within a controlled environment. Emulation therefore provides higher accuracy than the two previous methods since it does not use approximations of the real world components of the MANET. Discrepancies may still exist if the environment or equipment used in emulation do not reflect the real world deployed MANET. The use of a controlled environment helps to maintain the control and repeatability needed by the scientific method.

By combining real world implementations of communications protocols and computer hardware, various software tools can be used to construct testbeds. This enables actual MANET components to be tested under conditions which support the tenets of the scientific method. This is often done when observing the effects of real world wireless communications which are difficult to simulate with high fidelity [28]. This can be important when considering deployments where the environment has a noticeable impact on the wireless communications performed by the MANET, (e.g., environments involving subways, sky-scrapers, and moving vehicles).

While providing better fidelity than simulation or analytical analysis, emulation typically requires larger time and resource investments. Because emulation relies on testing real world implementations, existing tools can often be coupled to certain systems thus limiting reuse. It also becomes difficult to consider MANETs which have hundreds rather than tens of users as significantly more resources are required for larger scale models. This presents disadvantages when constructing a model to observe MANETs behaviours while under physical-layer jamming attacks. Also, in wireless, a core issue is sufficient per-experiment repeatability which is needed to deduce rigorous statistical information to assess MANET operations.

1.3.4 Ad-Hoc Testing

Ad-hoc testing is commonly used to observe the current operation of a real world MANET deployed in a specific environment. As such, this provides the highest level of fidelity and accurately represents the behaviour of the MANET. Unlike the previous approaches, ad-hoc, testing is not performed in a controlled environment and as such does not strictly satisfy the tenants of the scientific method. This is implied because the observations performed cannot be exactly reproduced nor can the physical properties of the MANET be held fixed due to the stochastic nature of the physical processes and the presence of external factors. Therefore, observations conducted may not be able to be meaningfully employed to predict the future behaviour of the MANET.

Due to this, ad-hoc testing is most commonly performed to stress test or troubleshoot services or protocols used in deployed MANETs. This is because the approach provides information about only a single instance of the process being observed, and lacks the repeatability and control needed when gathering statistical information. As such, the observations are limited in their scope of applicability to MANET operation.

While the highest level of fidelity is attainable through ad-hoc testing, a number of factors weigh negatively against this approach when considering the problem of modelling MANET operations in the presence of physical-layer jamming. In particular, the inability to achieve control and repeatability fails to satisfy the scientific method's tenets. Hence, although specific jamming experiments could be performed the generality of the observed MANET behaviours could not be assessed.

1.4 Existing Tools

Within the MANET community simulation is most often applied through the use of stochastic discrete-time event-based simulation engines combined with statistical analysis tools. Several software tools have been developed for this purpose and are in use to model MANET operations to examine a wide range of problems. More generally, these tools have been developed around communications networks and, as such, provide well used and trusted implementations of the complex protocols used in MANETs. Additional tools have been built around these simulators to perform observations of the MANET through statistical analysis.

Because models of this form are stochastic many simulations may need to be performed in order to make meaningful observations about issues, such as the impact of physical-layer attacks on MANET operations. This and other considerations are taken into account when evaluating the fitness of the existing tools, which can be grouped into two core categories: simulators and analysis tools.

1.4.1 Network-Level Simulators

A number of simulators have emerged and have become commonly used tools within the MANET research community, such as: NS-2 [23], OpNet [11], and OMNeT++ [47]. Each is well serviced and has advantages and disadvantages when applied to MANET research in general. Extensibility and scalability are primary factors when considering the appropriateness of each simulator to the specific problem of modelling MANET jamming.

NS-2

NS-2 is an open-source simulator that is well used within the network engineering community. The availability of source code to NS-2 allows it to be extended to a wide variety of problems related to computer networks, (i.e., those involving non-standard behaviours such as jamming wireless communications). Because it is well used there is a wealth of existing components available to provide correctness and to speed creation of the simulation required for the modelling MANET jamming scenarios.

A wide variety of problems relating to MANETs have been examined with the use of the NS-2 simulator. These include, but are not limited to: the number of users and their motion within the area to their impact on MANET operations [37], evaluation of existing and novel protocols [48, 30], and evaluation of MANET security [4]. This provides a wide range of existing observations against which the MANET jamming simulation can be compared for correctness.

While NS-2 is a popular simulator within the MANET community it suffers from a number of design decisions which hinder its extensibility and scalability when being applied to model physical-layer attacks against MANETs. In particular, the number of terminal hosts is limited due to resource constraints when examining how jamming impacts topologies where multiple routes are available [51, 25].

OpNet

OpNet is a closed-source commercial simulator that provides a powerful environment in which to observe MANETs. A major strength of this tool is the high level of accuracy and fidelity of the simulated physical-layer [11], and a wide range of existing validated simulation components. Although OpNet is closed-source, it provides support for the definition of new protocols and services which makes it a powerful network modelling and analysis tool. However, when attempting to simulate more complex behaviours, such as jamming attacks, the lack of available OpNet source code presents a significant barrier.

When modelling MANETs in isolation, it can be desirable to define rules about the operation of individual modules to ensure that pathological behaviours do not arise. This is most commonly performed by aborting the simulation being observed; however, when considering the presence of attackers, whose purpose is to cause the MANET to fail, such behaviours are innately what is of interest. Due to OpNet being closed-source, it restricts allowable behaviours of malicious users and their physical-layer attacks. As such, OpNet lacks the extensibility required when creating the MANET jamming simulation to evaluate different physical-layer attack strategies.

OMNeT++

OMNeT++ is rapidly gaining in popularity within the MANET community and, like NS-2, is open source. While not as well known or used as NS-2, it provides a highly extensible and scalable architecture [51] that supports a wide range of simulation modules and tools. Such resources are well serviced within the MANET community and are gaining in correctness as their usage increases[20]. Most recently with the release of version 4 of the OMNeT++ simulation engine a number of improvements have made it an effective network-level modelling tool.

Through the hierarchal definition of simulation modules via the Network Definition (NED) language, OMNeT++ allows for the dynamic creation of models. This enables existing simulations to be reused or altered with little effort by simply interchanging modules to provide the protocols and services of interest. New modules can also be defined using C++, and some models for NS-2 have even been ported for use with OMNeT++. This lends the established credibility of NS-2 to the OMNeT++ simulator because the same components can be used in each allowing for consistency to be observed [26, 6].

The modular design of OMNeT++ coupled with it being open-source provides an appropriate foundation on which to construct the MANET jamming simulation used to model physical-layer attacks. This is done by leveraging the INETMANET module library [7] to construct the basic MANET simulation which can then be extended to include attackers.

1.4.2 Analysis Tools

The stochastic nature of the simulation engines presented requires that statistical analysis be used if meaningful observations are to be made about the behaviour of different features of the MANET jamming model over time. That is to say, observations based on only a single simulation are unlikely to represent the general behaviour. In addition, it may be necessary to observe simulations over a long period of time to determine the full set of behaviours they may exhibit. Therefore, it is common to apply the Monte-Carlo method of statistical analysis which involves the observation of many simulations, (i.e., repetitions of the same configuration with different initial conditions). By averaging these observations together, over their ergodic modes, estimates can be produced which more accurately reflect the general case behaviour of the MANET jamming model.

Because it is not always possible to know a priori the number of simulations needed in order to make meaningful observations, manually assessing results and performing additional simulations is clearly impractical. A number of statistical analysis tools have been developed around this basic problem. These tools are considered below with respect to factors such as their ability to scale and the type of statistical analysis performed. These considerations are critical in determining the ability to conduct the statistical analysis at the scale needed. In particular, while these tools support parallel simulations, scalability of the statistical analysis is a concern because existing tools are not designed to make use of High Performance Computing (HPC) which limits the amount of data which can be collected.

When considering the applicability of each tool there are two core questions which it should address, namely: a) how the behaviour of the MANET changes over time, and b) how initial conditions impact the behaviour of a simulation. In statistics, the first of these pertains to the measured features' statistical stationarity, whereas the latter refers to their statistical ergodicity across Monte-Carlo runs. Assessing stationarity is important, for example, to identify start-up transients. While assessing

ergodicity is important for understanding whether a given MANET supports multiple modes of behaviour. By definition, statistical averaging, if it is to be meaningful, should only be done across ergodic data, as averaging across data drawn from different underlying distributions is of limited value.

Akaroa2

Akaroa2 [38] is a well known open-source analysis tool for use with both NS-2 and OMNeT++ which employs Multiple Replications In Parallel (MRIP) [38] to perform the simulations in a scalable manner. The statistical analysis is performed in-line with these parallel simulations and allows the tool to stop automatically once a sufficient amount of non-transient data is collected. Akaroa2 is designed to operate on a single multi-processor system or an interconnected set of systems to achieve scalability when performing Monte-Carlo runs of the simulation.

Akaroa2 uses a single master process to control the running of simulations across multiple hosts. Each host performs local analysis on the running simulation and reports back the results to the master process periodically. In this way the statistics calculated for each measured feature of the simulation are aggregated to a single process. The master process then compares the reported statistics against a researcher specified expected form. Once the aggregate data is within a specified tolerance the master process informs the hosts to terminate their simulations and the experiment is completed.

By coupling the statistical analysis with parallel simulation, scalability problems can arise when using Akaroa2. This is because the parallel simulations are performed concurrently and limited by the available computing resources. As such, it may not be possible even when utilizing cluster [17], grid [45], or cloud [29] based resources to achieve sufficient numbers of simulations. This coupling of analysis and simulation also means that trace data is not saved and any changes in the features measured or statistical analysis performed requires that the simulations be re-run. Furthermore, the centralized analysis presents potential issues with respect to the number of features which can be observed due to limitations, (e.g., memory, storage, and sockets) of the single computer system running the master process.

The statistical analysis performed also presents problems when stationarity and ergodicity are considered. In particular, Akaroa2 does not test if the statistics reported about each simulation are indeed governed by the same underlying distributions, a

requirement when properly performing statistical averaging. In addition, there is the assumption that the form of the aggregate statistics is known a priori which can present difficulties in identifying when sufficient data has been collected. As such, Akaroa2 does not adequately satisfy the requirements outlined for use in making observations about features of the MANET jamming model.

SimProcTC

SimProcTC [15] provides a completely open-source tool-chain based approach to the analysis problem designed to automate common tasks when performing OMNeT++ MANET simulations. Through Reliable Server Pooling (RSerPool) [14] the tool-chain can perform many simulations at once to achieve similar scalability as Akaroa2. SimProcTC also provides for the aggregation of each simulation's measured features. These are then analyzed by leveraging the GNU R statistics software package as part of the tool-chain. Results of the statistical analysis are then available to visualize using common methods, (e.g., GNU Plot graphing software, Octave mathematical software, etc.).

Some of the scalability problems of Akaroa2 are avoided by SimProcTC because it performs simulations independently of statistical analysis. This means that the number of simulations is not limited by the number of available processors. Yet there still exists a number of scalability concerns due to the use of the instrumentation included with OMNeT++ when observing many, (e.g., dozens) features across a large number of simulations. This is due to the native vector file format employed by OMNeT++ which make use of a single human readable file which is inefficient in terms of storage capacity. As such, this can constrain the number of measured features because the data must first be stored prior to conducting statistical analysis.

Questions related to testing for stationarity and ergodicity are not address by SimProcTC and it consists primarily of plotting basic statistics about each of the features observed. Unlike Akaroa2, there is no formal method used to determine when the initial transient behaviour has subsided leaving this task to be manually performed. This means that start-up transients are not removed from the data prior to statistical analysis. As such, SimProcTC lacks the statistical rigour needed to assess the behaviour of MANETs under different jamming scenarios.

STARS Framework

The STARS framework [34] was developed in conjunction with this work to address the shortcomings present in Akaroa2 and SimProcTC through the creation of an automation framework which facilitates a minimum of manual interaction. The framework uses a manager-worker architecture to control a set of computer resources used to process tasks, (e.g., perform simulations and their associated analysis) in a distributed manner by leveraging the MPI [18] standard. The automation support, provided through user defined scripts written in the Python[27] programming language, allows for customized sequences of tasks to be performed without manual intervention to support different research needs.

Unlike SimProcTC, the STARS framework provides instrumentation for use within the OMNeT++ engine to efficiently store the measurements collected from each simulation. This instrumentation provides considerable space savings by making use of a binary file format and data compression techniques. The statistical analysis, presented in Chapter 3, forms part of the framework and is implemented with MATLAB's Distributed Computing Toolbox[31]. This allows for distributed processing of calculations across multiple computers to mitigate resource constraints, (e.g., runtime, and per-node storage or memory limits).

Additionally, the statistical analysis conducted explicitly addresses testing for stationarity and ergodicity and is used to inform the feedback mechanism to control the number of simulations that the STARS framework performs for each experiment. This automation removes all but the interaction required for manually tuning of the MANET parameters and any problem specific portions of the experiment. As such, the STARS framework provides a scalable and extensible tool for performing and supporting the statistical analysis of stochastic MANET simulations.

1.5 Thesis Goals

MANET operation is formed by the aggregate behaviour of the hosts that comprise it as they exchange ratio signals. Jamming generates noise which impacts the ability for hosts to send and receive radio signals. This method of physical-layer attack on MANETs is illustrated in Figure 1.3 showing normal operations (left), and jamming effects (right).

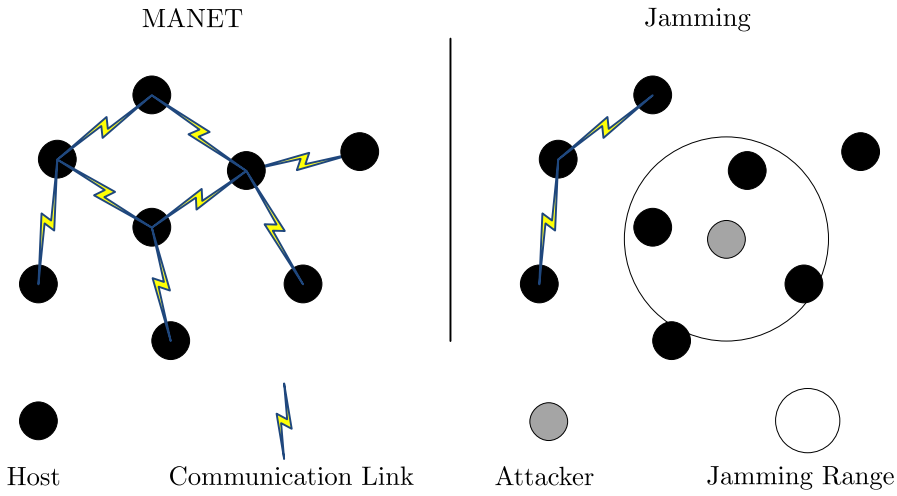


Figure 1.3: Normal MANET operation vs effects of jamming

The goal of this work is to quantify the impacts of different physical-layer jamming strategies on MANET operations. These strategies focus on how different jamming patterns compare with one another when range, motion, and the number of attackers are considered. Of interest is the impact each strategy has on normal MANET operations.

To meet these goals several needs must be addressed:

- A packet-level MANET simulation composed of: application, transport, network, link, physical, and motion layers.
- Attacker simulation capable of noise generation, motion, and multiple jammers.
- Observations, based on statistical analysis, which statistically quantify the MANET's range of behaviour informed by formal stationarity and ergodicity testing.
- Support for HPC facilities such as WestGrid to allow for observations to be made in a time practical manner.

1.6 Research Approach

The approach taken to analyze MANET jamming, and satisfy the goals of this work, is simulation using the OMNeT++ engine and the INETMANET library extended with custom modules. The statistical analysis uses custom MATLAB functions to

perform the needed formal statistical testing, (i.e., stationarity and ergodicity) as required to properly apply statistical averaging to the observations performed. These two components are leveraged using the STARS Framework in conjunction with the Hermes and Orcinus HPC facilities at WestGrid [10].

The MANET considered is comprised of autonomous hosts each constructed using modules which simulate various application, transport, network, link, physical, and mobility behaviours. The specific communication protocols used in the simulation of the MANET are UDP/IP (Section 2.2.4), DYMO (Section 2.2.5), and IEEE 802.11g (Section 2.2.6) chosen as described in their respective sections; however, different modules can be substituted into the simulation to change such things as the protocols used by the MANET.

Malicious users within the simulations perform four different jamming strategies: constant, random, reactive, and random reactive. The goal of each is to identify which is best able to disrupt the normal operations of the MANET in an efficient and difficult to detect way. Constant jamming employs a continually emitted radio signal to disrupt the MANET within communication range of the attacker; however, it is expensive in terms of power and is highly visible, relative to the other strategies. Random jamming attempts reduce the cost and visibility by emitting a radio signal for short periods of time. The reactive strategy is different from random in that the attacker emits a radio signal only when it observes MANET communications. This is intended to correlate the act of jamming with MANET activity to improve disruption and reduce visibility. Finally, random reactive is an extension of reactive jamming in which the attacker will randomly choose which MANET communications to jam.

Statistical analysis is performed on Monte-Carlo sets of experiment runs, (i.e., repetitions of the simulator for a given configuration using different initial conditions). The measured features on which the statistical analysis is performed are evaluated, individually and then across the set of Monte-Carlo runs. The start-up transients of a feature are formally removed by stationarity testing to detect steady-state behaviour. Common modes of feature behaviour are then identified by testing the Monte-Carlo runs found to exhibit stationarity for statistical similarity. Finally, averages can be reported using the ergodic data for the measured features of the MANET jamming simulator.

Experiments are constructed to observe how each of the four jamming strategies compare when being executed against the same MANET. In addition to performing these jamming strategies additional factors are considered, such as: the broadcast

power used when jamming, mimicking the motion of hosts in the MANET, and the presence of more than one malicious user. These variations are intended to expose how each strategy operates with respect to different configurations. In all cases the MANET observed is the same so that a baseline can be establish to compare each strategy, and their variations.

1.7 Glossary

Within this work, the terms below are used as defined:

- **Ergodic Mode** - A set of records for a given feature that were not observed to be statistically dissimilar on a pair-wise basis.
- **Ergodicity Graph** - A graph consisting of nodes, (i.e., all records for a given feature) connected by p-value weighted edges.
- **Record** - The data recorded for a given feature from a single run of the simulator.
- **Effective Jamming** - Disruption of MANET operation (availability, reliability, performance), or an increase in non-stationary and non-ergodic behaviour while minimizing the energy used (visibility.)

1.8 Thesis Organization

- **Chapter 2** presents the modifications and extensions made to OMNeT++ to allow it to be used to study jamming with MANETs.
- **Chapter 3** presents the statistical methodology used to analyze the collected MANET simulation data, inclusive of validating the methods developed.
- **Chapter 4** provides a discussion of physical-layer jamming strategies explored in this work, including the criteria by which the different jamming strategies are compared and how their impacts on the MANET behaviour is measured.
- **Chapter 5** evaluates the impact of each jamming strategy on the MANET operation and provides insight on the strategies relative performance.
- **Chapter 6** concludes the thesis and offers avenues of future work.

Chapter 2

MANET Jamming Simulator

This chapter introduces the event-based simulator upon which this work's MANET jamming experiments are conducted. The MANET operation is simulated within a fixed area, illustrated in Figure 2.1, and malicious users are introduced to execute the various jamming strategies. The MANET is comprised of many, (e.g., tens or hundreds) *terminal hosts* which interact autonomously to send data to one another as they move. *Attack hosts* are placed within the environment to observe the impacts of the different jamming strategies on MANET operations.

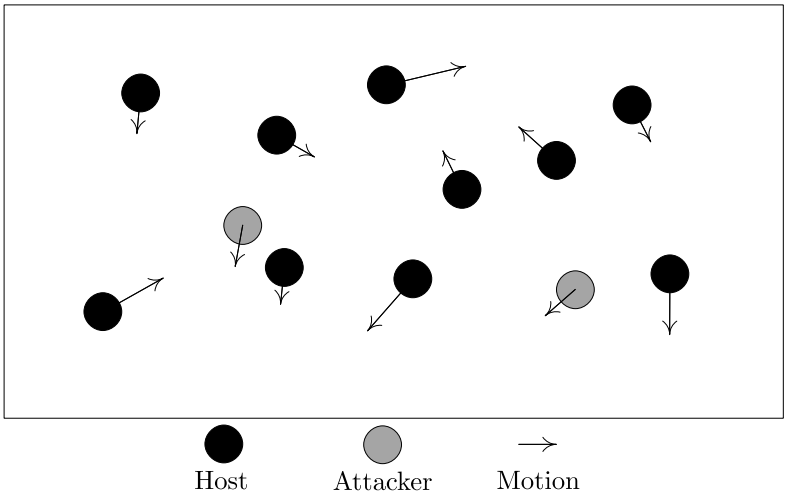


Figure 2.1: MANET Jamming Simulation Environment

The implemented MANET jamming model builds off of the OMNeT++ event-based simulator engine and leverages a number of existing modules from the INETMANET library. These pertain to standardized protocols, devices, and channel

physics to provide trusted implementations of complex modules required for simulation correctness. Additional extensions and modifications are made to support and study physical-layer jamming strategies.

2.1 OMNeT++ Simulator

OMNeT++ is a stochastic discrete-time event-based simulation engine commonly used to model wired and wireless networks. Simulations are constructed from interconnected modules which interact by sending messages to one another. These messages represent events which are scheduled based on the current state of the simulation and processed in time order. Randomness is supported through the Mersenne-Twister [32] pseudo-random number generator to allow for the exact recreation, or independent repetitions, (i.e., different initial conditions) of the simulation for the given configuration.

Simulation models, within OMNeT++, are declared as hierarchies of interconnected modules based on a root network definition (NED) file. Each module is implemented as a C++ class and uses its respective NED file to declare parameters or expose gates that act as sinks and sources for messages. Complex modules can also be declared in a NED file by defining the sub-modules to use, which in turn can be complex modules. This enables general purpose simulations to be declared and then customized by adding, removing, or swapping module implementations without needing to re-build the simulation.

The OMNeT++ engine has support for multiple random number streams which allow for the independent control of modules within a simulation. That is to say, modules can have their own, or share, a sequence of random numbers to isolate their actions from other modules. Individual sequences can then be held fixed or varied to control the initial conditions of the simulation. This enables repeatability and controllability within the simulation through the assignment of parameter values in a configuration file.

2.2 Terminal Hosts

The MANET simulation is formed through the interactions of terminal hosts communicating with one another as they move about the defined area. Host movement is autonomous and controlled through the use of a mobility model. To form the MANET

terminal hosts employ a number of common communications protocols, in this work: user datagram protocol (UDP), Internet protocol (IP), dynamic on-demand routing (DYMO) protocol, and the IEEE 802.11g wireless standard. While a number of protocols are available within the INETMANET library to choose from, those mentioned were chosen to suit the MANET jamming problem as discussed in the following sections.

The behaviour of each terminal host is controlled by an arrangement of software modules, shown in Figure 2.2, which mimics the structure of a simple wireless device. These modules can be classified *independent* or *dependent*. Those related to mobility and message generation are independent since their operation is determined by an independent random number stream. Independent modules act as inputs to the simulator. Modules related to communications protocols and wireless communications are considered dependent as they share the default random number stream. Dependent modules react to the inputs and simulate the MANET operations.

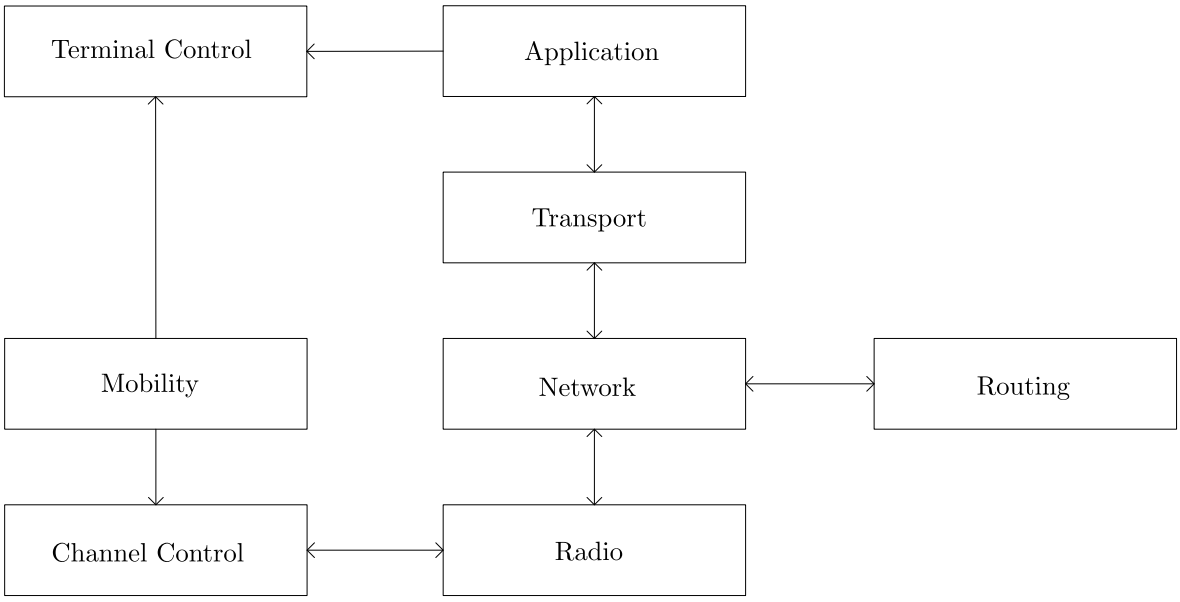


Figure 2.2: Software Module Structure & Interactions for Modelling Terminal Hosts

2.2.1 Terminal Mobility

By definition, terminal hosts move independently from one another within the defined simulated area. This movement was commonly modelled in earlier MANET research

via the random waypoint mobility model; however, it has been observed that this model does not converge to a uniform host distribution[55]. Instead, hosts tend to collect in the centre of the simulated area as the simulation advances. Uniformly distributed terminal hosts are desirable because it gives the minimum information assumption about the MANET. The hybrid model of [33] provides a near-uniform steady-state distribution of terminal hosts and, hence, is the mobility model used in this work.

More particularly, the simulated network area can be assumed to contain K terminal hosts with host k positioned at the Cartesian coordinates $\vec{p}_k = \langle x_k, y_k \rangle$, where $x_k \in [0...X_{max}]$ and $y_k \in [0...Y_{max}]$. This k^{th} host's motion is determined by an assigned waypoint $\vec{p'}_k = \langle x'_k, y'_k, s'_k \rangle$ which defines a destination location and speed $s'_k \in [S_{min}...S_{max}]$. When a new waypoint is needed it is randomly chosen from either the random waypoint or random walk mobility models.

These models differ in how they create new waypoints, as illustrated by Figure 2.3. The random waypoint chooses the new waypoint position $\langle x'_k, y'_k \rangle$ according to the uniform probability distributions p_x^{WP} and p_y^{WP}, given by

$$\begin{aligned} p_x^{WP}(x'_k) &\sim U(0, X_{max}) \\ p_y^{WP}(y'_k) &\sim U(0, Y_{max}) \end{aligned} \tag{2.1}$$

where $U(a, b)$ is the uniform distribution over the interval $[a, b]$. The random walk mobility model selects the waypoint location $\langle x''_k, y''_k \rangle$ by limiting the possible values of x''_k and y''_k to a uniform distribution defined over a maximum distance w from the host's current position, as p_x^{WK} and p_y^{WK}

$$p_x^{WK}(x''_k) = \begin{cases} \frac{1}{2w} & \text{for} \quad x_k \in [x_k - w, x_k + w] \quad \text{and} \quad x_k \in (w, X_{max} - w) \\ \frac{1}{2w}(w - x_k)\delta(x_k) + \frac{1}{2w} & \text{for} \quad x_k \in [0, x_k + w] \quad \text{and} \quad x_k \in [0, w] \\ \frac{1}{2w}(w - (X_{max} - x_k))\delta(X_{max} - x_k) + \frac{1}{2w} & \text{for} \quad x_k \in [x_k - w, X_{max}] \quad \text{and} \quad x_k \in [X_{max} - w, X_{max}] \end{cases}$$

$$p_y^{WK}(y''_k) = \begin{cases} \frac{1}{2w} & \text{for} \quad y_k \in [y_k - w, y_k + w] \quad \text{and} \quad y_k \in (w, Y_{max} - w) \\ \frac{1}{2w}(w - y_k)\delta(y_k) + \frac{1}{2w} & \text{for} \quad y_k \in [0, y_k + w] \quad \text{and} \quad y_k \in [0, w] \\ \frac{1}{2w}(w - (Y_{max} - y_k))\delta(Y_{max} - y_k) + \frac{1}{2w} & \text{for} \quad y_k \in [y_k - w, Y_{max}] \quad \text{and} \quad y_k \in [Y_{max} - w, Y_{max}] \end{cases} \tag{2.2}$$

Used alone random waypoint tends toward a non-uniform distribution of terminal hosts. This is because the random waypoint model converges toward large numbers of slow moving terminals clustering near the centre of the defined network area.

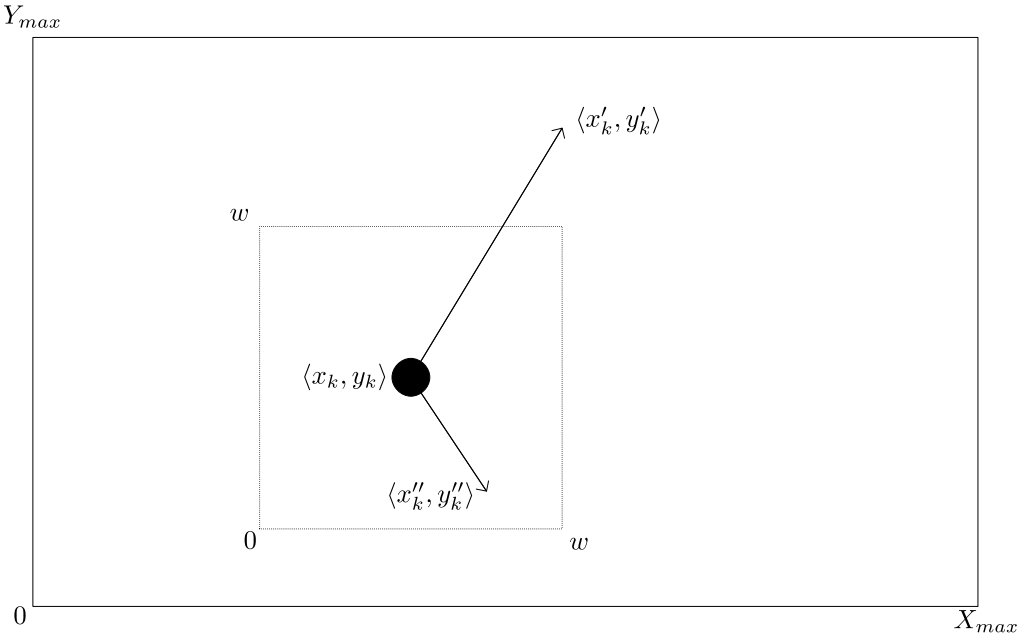


Figure 2.3: Waypoint Selection for Random Waypoint and Random Walk Mobility Models

This can result in a MANET which collapses to an expected diameter of one hop, (i.e., every terminal host is within communications range of all others). As a result, the simulation will will have movement model artifacts that will produce misleading observations. The hybrid model employed in this work avoids this problem as it can be shown to converge toward a near-uniform steady-state distribution [33].

The hybrid mobility model's algorithm, described in Algorithm (1), is implemented within the simulator by the *TerminalMobility* module. Each terminal host within the simulator contains an instance of this module which simulates motion based on the new waypoint chosen each time a prior waypoint is reached. Per-host movement is further defined in terms of a sequence of steps from the terminal host's location to the destination specified. The number of steps to reach the new waypoint from the current depends on the speed s'_k. From this, the number of steps to take in total S, as

$$S = \frac{d}{s'_k I} \tag{2.3}$$

where d is the Euclidean distance in meters between the waypoints and I is the seconds between steps. Each step performed moves the terminal host's position by $\vec{s}$,

$$\vec{s} = \frac{\langle x'_k - x_k, y'_k - y_k, 0 \rangle}{S} \tag{2.4}$$

Algorithm 1: Movement of Terminal Host k

```
p_k ← ⟨U(0, X_max), U(0, Y_max)⟩
p'_k ← ⟨x_k, y_k, 0⟩
s ← ⟨0, 0⟩
while simulation active do
    if p_k = p'_k then
        if U(0, 1) < randomChoice then
            p'_k ← ⟨U(0, X_max), U(0, Y_max), U(S_min, S_max)⟩
        else
            p'_k ← ⟨x_k + U(−w, w), y_k + U(−w, w), U(S_min, S_max)⟩
            p'_k ← ⟨max(0, min(x'_k, X_max)), max(0, min(y'_k, Y_max)), s_k⟩
        end
        s ← as per EQ. (2.3) and EQ. (2.4)
    else
        p_k ← s + p_k
        wait I seconds
    end
end
```

The behaviour of the *TerminalMobility* module is controlled by a number of per-experiment configuration parameters. These can be tuned to produce different mobility scenarios and those most commonly adjusted are listed in Table 2.1. For instance, by altering the *randomChoice* parameter the probability of choosing the random waypoint over the random walk can be set. Furthermore, the random walk mobility parameter w can be used to change the maximum distance a chosen waypoint can be from the current location of the terminal host. The *TerminalMobility* module is independent and makes use of random number stream 1 to allow movement patterns to be recreated. The full list of configurable parameters is provided in the module's NED file, see Appendix A.

Table 2.1: Tunable Terminal Mobility Module Parameters

Parameter	**Default Value**	**Description**
updateInterval	0.1 second	Time between a host's position updates, I
randomChoice	0.01	Probability $[0, 1]$ of using random waypoint instead of random walk model
w	15 meters	Random walk mobility parameter
s	5 m/s	Speed chosen for waypoints

2.2.2 Destination Selection

A MANET tries to provides one or more communication paths between any of the terminal hosts which comprise it; however, the terminal host movement discussed in Section 2.2.1 can impact how a MANET behaves. This is because terminal hosts can become disconnected due to changes in position over the simulated interval. This can result in pathological behaviours as the MANET fractures into disjoint groups, (i.e., multiple MANETs) of locally connected but globally disconnected groups. More commonly, terminal hosts at the edge of the MANET connect and disconnect more frequently than those near the MANET's centre. These edge terminal hosts will experience availability and reliability issues and this can introduce observation biases. By considering the terminal host level connectivity when selecting destinations, this bias can be mitigated.

Terminal hosts, within this work's simulations, statistically favour those it has a reasonable expectation of being connected to. That is to say, a terminal host will be more likely to send data within its own group than to a disconnected group. This is accomplished by tracking the relative position of each terminal host and creating a graph where edges denote that terminal hosts are within communication range. Within this graph, destinations can be randomly chosen from either the terminal host's local group, or alternatively from all possible terminal hosts.

The process of selecting message destinations is implemented in the simulator by the *TerminalControl* module which tracks the location of terminal hosts. It operates by updating the graph as terminal hosts move within the defined simulated area. In some cases, the addition or removal of an edge from the graph will not change the "connectedness" of terminal hosts within the group, (i.e., MANET). However, the

removal of edges can produce disjoint groups as one or more terminal hosts become disconnected. Therefore, each time an edge is removed from the graph a search is performed to determine if the two terminal hosts are still in the same group. This is because if a terminal host is connected to a terminal host within a group, it is also connected to the group.

This group membership test is performed through use of an A* search [5] which is guided by an evaluation function that is admissible ensuring that the shortest path will be found if one exists. The search runs until the goal , (i.e., destination terminal host) is reached, or all paths are exhausted. The evaluation function $\hat{e}(n) = \hat{g}(n) + \hat{h}'(n)$ is defined as the current path length $\hat{g}(n)$, in hops taken from the source to n, plus the estimated hops from n to the goal $\hat{h}'(n)$. The heuristic $\hat{h}'(n)$, is calculated as

$$\hat{h}'(n) = \frac{d(n,g)}{d_{hop}} \tag{2.5}$$

where $d(n,g) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2}$ is the Euclidean distance from terminal host n to the goal terminal host g and d_{hop} is the maximum one-hop distance in meters, (i.e., terminal host communication range). This means the estimated number of hops to the destination will always be equal to or less than the actual distance. Therefore the heuristic will be admissible. Algorithm (2) describes the search with respect to the starting node s, the goal node g, and the set of the current node's neighbours $N(n)$.

Algorithm 2: A* Search

$T_H \leftarrow \{s\}$
while $|T_H| > 0$ **do**
  $n \leftarrow first\,(T_H)$
  $T_H \leftarrow T_H - \{n\}$
  if $n = g$ **then**
    exit s is connected to g
  end
  $T_H \leftarrow T_H \bigcup N(n)$
  sort T_H by increasing $\hat{e}(n)$
end
exit s is not connected to g

The *TerminalControl* module tracks the current position of every terminal host and uses it to maintain the graph representing their "connectedness." Each time a terminal host changes its position, the *TerminalControl* module checks if any terminal hosts have entered or exited its connection distance. In addition to the graph, the module maintains a list of groups and which terminal hosts are in each group. When a destination is needed, the *TerminalControl* module randomly chooses to select one from either the list of terminal host in the local group, or from all terminal hosts. In both cases the destination is then selected uniformly from the list of possible choices.

The graph generated, and destinations provided, by the *TerminalControl* module is controlled by two configuration parameters. The first is the connection distance which depends on the number of terminal hosts and the size of the defined simulated area and should result in six to seven neighbours on average[36]. The second configuration parameter is the probability of randomly choosing destinations from all terminal hosts instead of from the local group. These are listed in in Table 2.2, while the full list of configuration parameters available from the module's NED file, see Appendix A. The *TerminalControl* module is dependent on both the motion of the terminal hosts and the network traffic they generate. However, the module makes use of random number stream 3 so the same destinations can be produced if motion and traffic are the same.

Table 2.2: Tunable Application Control Module Parameters

Parameter	**Default Value**	**Description**
connectionDistance	98m[1]	Maximum separation distance between two connected hosts, or d_{hop}
randomChoice	0.05	Probability of choosing a destination from all terminal hosts

2.2.3 Network Traffic Generation

The network traffic present in a MANET depends on the communications of individual terminal hosts that comprise it. A common method used to model network

[1]Combined with a host density of roughly 250 per km^2 the communication distance yields an average of 7 neighbours per terminal host , (i.e., $neighbours = \pi r^2 \frac{hosts}{area}$). This enables large numbers of alternate routes and provides resilience against jamming.

traffic is the On/Off model [54] which represents a single network traffic source, (i.e., application running on a terminal host). Terminal hosts are allowed to contain one or more of these sources which enables them to generate richer traffic patterns. Every source can be individually configured to produce a variety of different traffic patterns, such as: constant bitrate (CBR), Poisson, or Pareto.

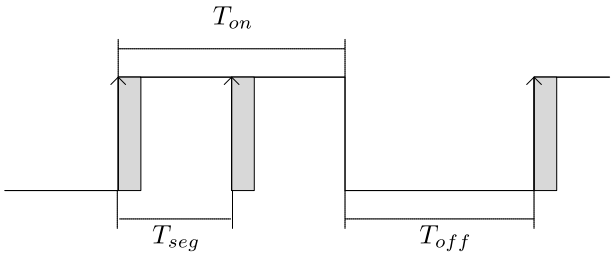


Figure 2.4: On/Off Network Traffic Source Model

The On/Off model, illustrated in Figure 2.4, functions by cycling between two operational states: the On state and the Off state. The On state is defined by a period of time T_{on} during which the source will issue data to the MANET for delivery. The source will send B_{msg} bytes of data over the period T_{on} in segments of maximum size B_{seg} bytes. The first segment will be issued at the start of the On state and each additional segment, if necessary, will be issued at an interval of T_{seg}, as per EQ. (2.6).

$$T_{seg} = \left\lceil \frac{B_{msg}}{B_{seg}} \right\rceil \frac{1}{T_{on}} \tag{2.6}$$

After the On state finishes, the Off state is entered and the source is idle for the duration T_{off}. Once the Off state is over the source once again enters the On state and the cycle is repeated for the duration of the simulation. The segmentation of data at the application-layer is done to avoid introducing observational biases from network-layer behaviours such as fragmentation and loss.

The application-level network traffic generated by a source depends on the duration of each state, the amount of data sent, and the segment size. The MANET behaviour will depend on the aggregate network traffic of every source. Depending on the properties desired a source can be configure to produce a number of different traffic patterns. For example constant bit rate (CBR) can be generated by choosing constant periods for the On and Off periods. More complex traffic can be generated

by randomly choosing On and Off durations based on distributions such as Exponential, or Pareto. This allows each source to be tuned such that it can reproduce CBR, Poisson, and Self-Similar traffic as needed to simulate the desired application traffic.

The *OnOffApp* module implements individual traffic sources in the MANET simulator. One or more of these modules can be instantiated per terminal host as application modules to send and receive data through the MANET via the simulated transport-layer. At the beginning of each On state, the source requests a destination from the *TerminalControl* module and then issues the first data segment to the MANET for delivery. If more than one segment is to be sent the source waits a period of T_{seg} before issuing the next. Data segments delivered by the MANET via the network-layer to the *OnOffApp* are discarded once received.

The source simulated by the *OnOffApp* module can be configured to produce a variety of traffic patterns. The configuration parameters most commonly tuned to accomplish this are given in Table 2.3. OMNeT++ allows for parameters to act as random variables, (i.e., take values described by a distribution) which enables more complex traffic patterns to be realised. The *OnOffApp* module is also independent and makes use of random number stream 2 to allow the application traffic pattern to be recreated. A full list of all configuration parameters for this module is available from the module's NED file, see Appendix A.

Table 2.3: Tunable Application Module Parameters

Parameter	**Default Value**	**Description**
onPeriod	8 second	CBR on period, T_{on}
onPeriod	exponential(7.5) seconds	Poisson on period, t_{on}
offPeriod	0 seconds	Duration of off period, T_{off}
requestBytes	2KB	Size of message to send during the on state, B_{msg}
segmentBytes	1KB	Maximum size of each segment, B_{seg}

2.2.4 Data Transport Over Internet Protocols

Terminal hosts make use of the user datagram protocol (UDP) [42] and the Internet protocol (IP) [40] to deliver messages. Although transport control protocol (TCP) [41] provides guaranteed delivery of data and congestion control, which are both

desirable in a lossy environment, (i.e., MANETs), it is more sensitive to the impacts of physical-layer jamming attacks than the more simple UDP.

TCP, unlike UDP, provides a number of network-level services such as guaranteed delivery. TCP accomplishes this by sending packets and waiting for verification from the destination that they were successfully delivered. If no acknowledgement is received within a certain period of time then the packet is sent again. As such, in an environment where packet loss is common, such as a MANET under physical-layer jamming, the use of TCP can result in a large increase in traffic due to either the loss of the acknowledgement or original packet causing re-transmission. This can lead to situations where the MANET becomes congested, causing the TCP protocol to exhibit worst case performance. Hence, it can be argued that assessing the effectiveness of jamming under UDP provides a lower bound on its effectiveness under TCP. Therefore, only the UDP transport protocol is considered in this work.

In order to deliver data from one terminal host to another each is assigned an unique IP version 4 (IPv4) address. Data issued by a terminal host's application module is first encapsulated as a UDP/IP packet, or packets if fragmentation is needed, and prefixed with a header prior to being issued to the MANET for delivery. The header contains the destination IPv4 address, the destination port, and the next-hop IPv4 address. The next-hop information is obtained from the host's routing table, as per Section 2.2.5, and is used by the network-layer to route the packet through the MANET to the destination terminal host.

The *UDP* and *IP* modules of the simulator implement their respective protocols. These two modules simulate the transport layer and part of the network layer respectively within each terminal host. These modules form core parts of the OMNeT++ network simulation tool; hence, their correctness has been well studied and assured by the wider OMNeT++ research community [20].

The *UDP* module, of each terminal host, functions to encapsulate data received from the application-layer prior to passing it on to the *IP* module. Alternatively, when the *UDP* module receives a packet from the *IP* module, decapsulates it, and delivers it to the specified *OnOffApp* module by port. This allows for the multiplexing and de-multiplexing support found in UDP required to run multiple applications on each terminal host.

A packet arriving at the *IP* module is handled depending on the destination address contained within the header. If the packet's destination is the current terminal host it is passed to the *UDP* module for delivery; otherwise, the packet is passed to the

wireless device module for transmission to its next hop address based on information contained within the terminal host's routing table. If no such information is available the *IP* module invokes the routing protocol to discover the next hop address and once found sends the packet to the wireless device.

Packets traversing the MANET can be dropped by the *IP* module due to a number of circumstances. The first of these occurs when the routing protocol is unable to determine the next-hop address, meaning that the route connecting the sender with the destination could not be discovered. The second reason for a packet to be dropped is when it exceeds the maximum hop count, (i.e., fails to reach its destination after being relayed by a predefined number of terminal hosts). A packet may also be lost due to data corruption detected when the checksum does not match. In all such cases the corresponding ICMP [43] error packet, (e.g., destination unreachable or time exceeded) is issued to the sender and delivered over the MANET.

The simulated *IP* module within each terminal host can be tuned to perform in different ways based on a number of configuration parameters. Those which relate to processing time and maximum hop counts are shown in Table 2.4. The *UDP* and *IP* modules are both dependent and use the default random number stream 0. Full lists of the parameters for the *UDP* and *IP* modules are available in their respective NED files, see Appendix A.

Table 2.4: Tunable IP Module Parameters

Parameter	**Default Value**	**Description**
procDelay	0 seconds	Time packet is delayed due to processing
timeToLive	32 hops	Maximum hops before unicast packet is discarded
multicastTimeToLive	32 hops	Maximum hops before multicast packet is discarded

2.2.5 Message Routing

Because MANETs innately have changing topologies, the route a packet will need to take to be delivered must be discovered before it can be sent. A number of solutions to the problem of finding routes through MANETs have been developed and fall into

different categories, such as: reactive, proactive, and hybrid routing strategies. The former includes dynamic MANET on-demand routing (DYMO) [12] and ad-hoc on-demand distance vector (AODV) routing [39] which exchange control packets between terminal hosts to identify routes as needed. Proactive protocols, such as optimized link state routing (OLSR) [13], on the other hand, elect certain terminal hosts to actively maintain and distribute route information throughout the MANET.

OLSR relies on the exchange of control packets at regular intervals to handle changes in topology that break routes. All terminal hosts emit a *HELLO* control packet to announce its participation in the MANET and listen for *TOPOLOGY* control packets to inform their routing decisions, (i.e., maintain the terminal host's routing table). These *TOPOLOGY* packets are issued by the elected multipoint relay (MPR) terminal hosts periodically. These selection of the MPR hosts is driven by the *HELLO* control packets in combination with a time-out. If the MPR is jammed, then multiple terminal hosts will have stale routing information until a new MPR is elected and the terminal hosts reconnect with the MANET.

Reactive protocols such as AODV and DYMO exchange control packets as needed. Both operate by flooding the MANET with route request (RREQ) packets and waiting for a route reply (RREP) or route error (RERR) packet in response. DYMO is an extension of AODV and of more interest to the MANET research community as it is less well studied than its predecessor, especially in the area of physical layer jamming attacks. Because DYMO is reactive, every terminal host in the MANET is able to gather its own routing information. OLSR's reliance on MPR terminal hosts makes it more susceptible to jamming and therefore DYMO is chosen for use in this work.

DYMO employs three types of control packets to discover routes and handle breaks within the MANET: RREQ, RREP, and RERR. When a terminal host has no next-hop information for a destination, it broadcast a RREQ control packet to every terminal host it is connected to. If a terminal hosts receives a RREQ packet and does not have the routing information sought it will re-broadcast the RREQ packet; otherwise, the terminal host will send a RREP packet to the terminal host which issued the RREQ packet along the path it arrived from.

As the RREQ, RREP, and RERR packets move through the MANET they carry with them the route each has taken, as illustrated in Figure 2.5. Upon reception of this propagated route information, the terminal host processes any local updates. The control packet's route information is then updated and forwarded as part of the control packet sent back to the MANET. In this way, DYMO builds all intermediate

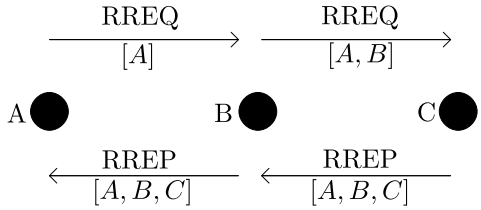


Figure 2.5: DYMO Route Propagation for A Requesting a Route to C

routes between the terminal hosts involved in each route discovery process. Whereas AODV constructs only the route between the sender and receiver terminal hosts. Moreover, this means that DYMO is continuously updating the routing information of each terminal host as control packets are exchanged.

The routing tables maintained by each host using DYMO can become out of date and result in failed communications. This can result from lack of use, terminal host motion, or the failure to successfully transmit a radio signal, as discussed in Section 2.2.6. If a terminal host detects that a route has broken, (i.e., it is unable to send packets to the next hop for any reason) it will emit a broadcast RERR control packet into the MANET to inform other terminal hosts. This RERR packet floods through the network and each terminal host receiving it remove routes which rely on the broken route.

The *DYMO* module used by simulator to implement the DYMO routing protocol is taken from the INETMANET library. This implementation, *DYMO-FAU* [46], is one of two available within the library and was chosen due to its implementation of a more recent draft of the DYMO standard, (i.e., it implements draft 10 while the other, *DYMOUM*, implements draft 5 and contains experimental code). A single instance of the *DYMO* module is contained within the network-layer of each terminal host.

The route discovery process is initiated for each packet that is passed to the *DYMO* module, by the *IP* module, when no-next hop information exists for the destination terminal host in the routing table. The *DYMO* module in turn issues, and receives, control packets to the MANET via the *IP* module to discover and maintain the local routing table. The packet associated with each route discovery is held at the *DYMO* module which initiated the route discovery until the process completes at which time the packet is returned to the *IP* module.

The *DYMO* module used in the simulator can be tuned by adjusting a number of configuration parameters. The most common of these are listed in Table 2.5 and

control how the protocol creates and handles the control packets. The *DYMO* module is a dependent module and makes use of the default random number stream 0. A full list of configuration parameters is available from the module's NED file, see Appendix A.

Table 2.5: Tunable DYMO Module Parameters

Parameter	**Default Value**	**Description**
MIN_HOPLIMIT	5 hops	Max hop distance for first RREQ control packet issued by the protocol during the discovery process
MAX_HOPLIMIT	10 hops	Max hop distance for subsequent RREQ control packets
RREQ_TRIES	3	Number of RREQ to issue before giving up
RREQ_WAIT_TIME	2 seconds	Time before route discovery fails and is re-tried

2.2.6 IEEE 802.11g Wireless Device

The terminal hosts which compose the MANET are interconnected at the physical-layer through wireless devices. This enables a terminal host to exchange data via radio signals with other terminal hosts in its communication range. IEEE 802.11g [1] is one of the most common wireless communication standards and it is often employed when modelling MANETs. IEEE 802.11g wireless devices, (i.e., radios) operate to fairly share access to the wireless channel within the communication range of each terminal host. It performs this through two mechanisms, namely: collision detection and avoidance. These form part of the medium access control (MAC) protocol used by IEEE 802.11g.

Two factors are taken into consideration by the radio of a terminal host when attempting to receive a signal sent by another terminal host: observed signal strength and signal-to-noise (SNR) ratio. The first varies inversely to distance from the source and if the observed strength of a radio signal is below a predefined sensitivity threshold the radio considers it noise. The second factor considered is the SNR which measures the ratio of the signal power observed over the noise observed. When the SNR falls below a predefined threshold, the reception is aborted due to loss of signal. This

implies that the actual communication range of a terminal host is depends not only on the power used to transmit the radio signal but also the time varying noise observed by the terminal host receiving the legible signal.

The collision detection employed by IEEE 802.11g drops incoming signals when their SNR is measured to be below the minimum value. This is because the signal, while still measurable, has a biterror rate too high for reception as a packet. The collision avoidance used by IEEE 802.11g dictates that if the radio of a terminal host observes a signal above the sensitivity threshold it will not attempt to broadcast its own signal. In such cases, a random back-off period is used to reschedule the signal transmission; however, this does not guarantee that collisions will not still happen.

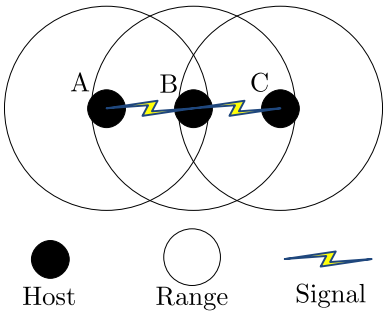


Figure 2.6: Illustration of The Hidden Neighbour Problem in MANETs

Collisions may still occur when using IEEE 802.11g wireless devices due to a scenario called the hidden neighbour problem [19]. This problem is illustrated by considering the three terminal hosts shown in Figure 2.6, where A is in range of B, B is in range of C, but A and C are not in range. If terminal host A broadcasts a signal to terminal host B, it will not be observed by terminal host C. Therefore, if terminal host C also broadcasts at the same time as A, terminal host B will observe both radio signals and experience a collision. In such cases due to the distance between terminal hosts A and C the collision avoidance mechanism will not prevent the collision, as neither can observe the other's broadcasts.

Within the simulator, the IEEE 802.11g wireless device used by each terminal host is implemented by the *WirelessDevice* module which contains three sub-modules: *Management*, *MAC*, and *Radio*. The first and third modules act as interfaces with the network-layer and physical-layer, respectively. The *MAC* module simulates the multiple access control (MAC) protocol employed by IEEE 802.11g to share the wireless channel.

Packets which are sent to the *WirelessDevice* module from the network-layer are first encapsulated inside data frames by the *Management* module and then queued prior to being passed onto the *MAC* module. Each frame is then issued to the *Radio* module by the *MAC* module where it is then converted into a radio signal and transmitted over the wireless channel. The *MAC* module does this based on the current state of the *Radio* module, (i.e., idle, sending, or receiving) according to the IEEE 802.11g standard.

Upon successful reception of a radio signal, the *Radio* module converts this signal into a data frame and passes it to the *MAC* module which in turn passes it to the *Management* module. The packet contained within the data frame is then decapsulated and sent to the *IP* module of the terminal host which received the radio signal.

Each of the sub-modules that comprise the *WirelessDevice* module expose configuration parameters which can be tuned to alter their behaviour. Those most often adjusted related to the radio sensitivity, minimum acceptable SNR, and datarate of the wireless device. These are listed below in Table 2.6 and Table 2.7 which pertain to the *MAC* and *Radio* modules respectively. The *WirelessDevice* module and its sub-modules are dependent modules and make use of the default random number stream 0. The full list of configuration parameters governing the operation of the sub-modules that comprise the *WirelessDevice* module is available in their respective NED files, see Appendix A.

Table 2.6: Tunable MAC Module Parameters

Parameter	**Default Value**	**Description**
bitrate	56.4Mbps	data frame transmission bitrate
basicBitrate	2.6Mbps	frame header transmission bitrate
opMode	2	1 - 802.11b legacy mode, 2 - 802.11g mode
mtu	1500Bytes	Maximum frame size in Bytes

2.2.7 Modelling The Wireless Communications Channel

Radio signals exchanged by terminal hosts propagate through the physical environment in which the MANET operates. However, the stochastic nature of physical

Table 2.7: Tunable Radio Module Parameters

Parameter	**Default Value**	**Description**
carrierFrequency	2.4GHz	IEEE 802.11 carrier frequency
transmitterPower	2mW	Power to use for data frame transmissions
thermalNoise	-110dBm	Ambient noise in network environment
snirThreshold	4	Minimum SNR below which broadcasts are considered lost
sensitivity	-90dBm	Minimum power below which broadcasts are ignored

processes characterizing the environment combined with real world deployment scenarios mean that the propagation of radio signals can be altered or blocked by objects. Such factors can influence the reliability and availability of the MANET. In order to isolate the impact of physical-layer jamming on MANET operations from environmental influences a simplified channel model is employed.

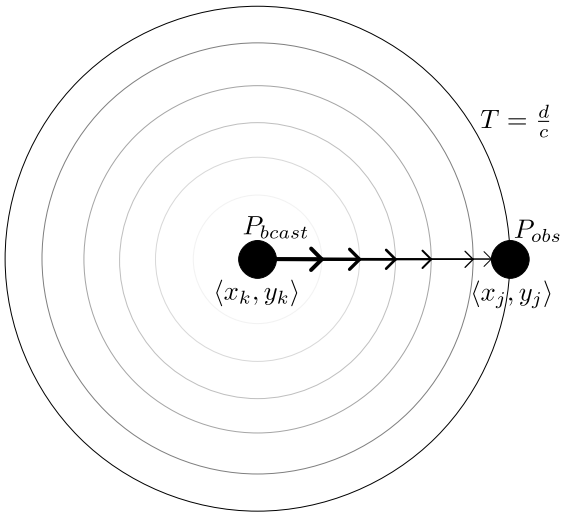


Figure 2.7: Radio Signal Propagation Over Time

Signal broadcasts propagate radially outward from a source, (i.e., a terminal host's radio) and are observed by one or more terminal hosts some distance away via an omni directional antenna. Figure 2.7 illustrates this by considering two terminal hosts located at $\langle x_k, y_k \rangle$ and $\langle x_j, y_j \rangle$ respectively. These hosts are separated by an Euclidean

distance of $d = \sqrt{(x_k - x_j)^2 + (y_k - y_j)^2}$ meters and the signal is characterized by two features. First is the signal wavelength $\lambda = \frac{c}{f}$, where c is the speed of light and f is the carrier frequency used by the IEEE 802.11g radio. Second is the broadcast power P_{bcast} in Watts (W) at which terminal hosts emit the radio signal. When either terminal host broadcasts a radio signal the other will observe it after a delay of $T = \frac{d}{c}$ seconds and at a power P_{obs}, as

$$P_{obs} = \frac{P_{bcast}\lambda^2}{16\pi^2 d^\alpha} \tag{2.7}$$

where $\alpha = 2$ is the path-loss coefficient which can be tuned to adjust how rapidly the signal power dissipates with respect to the distance. Once the signal strength falls below the noise threshold of *sat* decibels (dB) it is no longer possible to observe. The distance the signal can travel before it reaches this threshold is called the interference distance d_{intf}, defined by

$$d_{int} = \left(\frac{P_{bcast}\lambda^2}{16\pi^2 10^{\frac{sat}{10}}} \right)^{\frac{1}{\alpha}} \tag{2.8}$$

The channel model described above is implemented by the *ChannelControl* module of the simulator. The module used is taken from INETMANET and models the physical-layer of the MANET. The *ChannelControl* module is passed signals from the *Radio* module of the transmitting terminal host. Signals are then relayed to all terminal hosts within the interference distance. After the appropriate propagation delay the receiving terminal host's *Radio* module begins to receive the signal. The amount of time needed to complete the reception of the signal depends on the channel frequency and amount of data contained in the signal.

The *ChannelControl* module provides a number of tunable configuration parameters in order to adjust the channel model. The most commonly tuned pertain to the path-loss equation and are given in Table 2.8. The *ChannelControl* module is dependent and makes use of the default random number stream 0. A complete list of tunable configuration parameters is available from the module's NED file, see Appendix A.

Table 2.8: Tunable Channel Control Module Parameters

Parameter	**Default Value**	**Description**
pMax	0.1mW	Maximum transmission power of radio signal, P_{bst}
sat	-110dBm	Signals below this power level cannot be observed by the radio
alpha	2	Path-loss coefficient α

2.3 Attack Hosts

Malicious users within the simulator are represented by attack hosts which work to disrupt MANET operation through jamming. They perform this by broadcasting a radio signal in a variety of ways designed to exploit the collision detection and avoidance used by the IEEE 802.11g wireless devices used by the terminal hosts forming the MANET. To avoid detection, attack hosts also employ motion using the same mobility model as the terminal hosts.

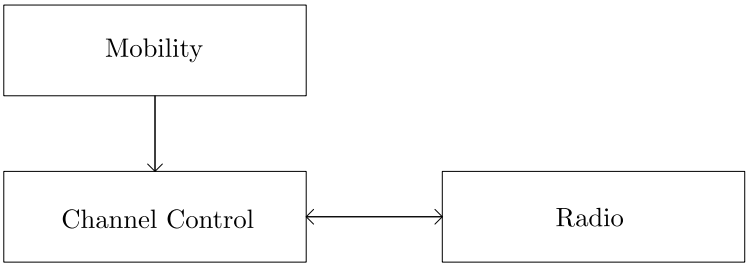


Figure 2.8: Software Module Structure & Interactions for Modelling Attack Hosts

Attack hosts function based on a collection of software modules, shown in Figure 2.8, which define the components necessary to enact physical-layer based jamming attacks. The first of these is a custom wireless device which allows for direct control of the radio in order to emit noise into the MANET's wireless communications channel. This is required in order to produce the different jamming strategies. The second component controls attack host movement through predefined motion paths. This enables attack host not only to mimic terminal host motion, as described in Section 2.2.1, but avoid pathological cases undesirable for jamming, (e.g., at the edge of the defined simulated area).

2.3.1 Physical-Layer Jamming

Physical-layer jamming strategies seek to disrupt MANET operation by exploiting the properties of the shared wireless communication channel. This can be done by broadcasting signals in different ways to generate common jamming strategies, namely: constant, random, and reactive [52]. A fourth strategy, called random reactive is also studied in this work and is an extension of reactive jamming.

The constant jamming strategy continually emits a radio signal to stop terminal hosts, within jamming range of the attack host, from exchanging radio signals over the channel. Undesirably, the constant emission of a radio signal is highly visible and power intensive. Random jamming mimics the On/Off source, described in Section 2.2.3, to randomly emit a signal. This reduces the power used and is less visible, but may or may not emit the signal when the MANET is actively communicating. Reactive jamming addresses this by only emitting a signal when activity is detected within the communication distance of the attack host. The random reactive strategy extends the reactive strategy by emitting a signal randomly when activity is detected. The four jamming strategies considered in this work impact MANET operations differently depending on the relative distances between terminal hosts and attack hosts, in three ways:

1. The collision detection mechanism in IEEE 802.11g will force terminal hosts to drop any incoming radio signals while jamming is active.

2. The collision avoidance mechanism in IEEE 802.11g will force terminal hosts to wait before attempting to broadcast radio signals, (i.e., 802.11g's random back-off will be applied).

3. The noise observed within the wireless channel will be increased thereby reducing the per-terminal communication ranges.

Terminal hosts within communication range of the attack host, as defined by Section 2.2.6, will be impacted by (1) and (2). Terminal hosts outside of the communication range but inside the interference distance of the attack host, given in Section 2.2.7, will be impacted by (3).

The wireless device utilized by each attack host is implemented within the simulator by the *AttackerRadio* module. There is one instance of this module in each attack host and it interacts with the MANET through the *ChannelControl* module used by

terminal hosts to exchange radio signals. The *AttackerRadio* module is independent and makes use of random number stream 4.

The configuration parameters listed in Table 2.9 encompass those needed to produce each of the four jamming strategies. The constant strategy is produced by setting the *mode* configuration parameter to 0 and specifying an *onPeriod* time proportional to the *updateInterval* configuration parameter of the mobility module, listed in Table 2.1. The random jamming strategy is produced by setting *mode* to 1 and assigning random variables to the *onPeriod* and *offPeriod* configuration parameters. Reactive and random reactive strategies are both produced by setting the *mode* to 2. Reactive is chosen by setting the probability of jamming, *Pjam*, to 1, while random reactive uses values in the range $(0, 1)$.

Table 2.9: Tunable Attacker Radio Module Parameters

Parameter	**Default Value**	**Description**
mode	0,1,2	Operate in 0) constant, 1) random, or 2) reactive mode
onPeriod	10ms	Emit a signal for this amount of time.
offPeriod	0s	Do not emit a signal for this period, mode (1) only.
Pjam	1	Probability of reactively emitting a radio signal, mode (2) only.

2.3.2 Pre-Defined Motion

Attack hosts within the simulated area employ motion to avoid detection by following a predefined path chosen to mimic the movement of terminal hosts. While this can be achieved through the use of the terminal hosts' hybrid mobility model, presented in Section 2.2.1, the manual selection of the path is done to avoid pathological cases. Furthermore, this work represents an initial assessment of the MANET jamming problem. As such, this approach is limited in that attack hosts cannot dynamically alter their movement based on MANET operation.

The simulated network area contains one or more attack hosts, with the ith positioned at $\langle x_i, y_i \rangle$ at time t_i. The path taken by attack hosts is described by a sequence

of waypoints listing locations and times when the attack host is to arrive. Each waypoint is defined as $\langle x'_i, y'_i, t'_i \rangle$, where $x'_i \in [0...X_{max}]$, $y'_i \in [0...Y_{max}]$, and $t'_i \in [0, t_{max}]$ is monotonically increasing.

Unlike terminal hosts, which move based on predefined speeds, attack hosts calculate their speed of travel according to the time between consecutive waypoint. This speed s'_i is calculated as

$$s'_i = \frac{|\langle x'_i - x_i, y'_i - y_i \rangle|}{t'_i - t_i} \tag{2.9}$$

where $0 \leq s'_i \leq S_{max}$. If the speed calculated exceeds a maximum S_{max} then the attack host will move toward the waypoint at S_{max} until time t'_i when a new waypoint will be issued. As such, the attack host may not reach the intended waypoint prior to proceeding to the next one.

By combining the location $\langle x'_i, y'_i \rangle$ and the calculated speed s'_i attack hosts create a new waypoint $\langle x'_i, y'_i, s'_i \rangle$. Thus attack hosts use the same step-based movement as terminal hosts, (i.e., defined by number of steps S, as per EQ. (2.3), and step $\vec{s}$ as per EQ. (2.4)).

The motion of attack hosts is implemented in the simulator by the *AttackerMobility* module. There is one instance of this module in each attack host and it is similar in operation to the *TerminalMobility* module presented in Section 2.2.1. The *AttackerMobility* module differs by obtaining new waypoints from a file rather than through a mobility model.

The *AttackerMobility* module exposes a number of configuration parameters which can be set to control the motion of each individual attack host. The most commonly adjusted, listed in Table 2.10, relate to the maximum speed at which the attack host can travel and the file containing the waypoints to consume. The *AttackerMobility* module is independent and makes use of random number stream 5. A full listing of configuration parameters for the *AttackerMobility* module can be found in the module's NED file, see Appendix A.

The *wpfile* configuration parameter identifies the relative or absolute location of the file containing waypoints which the *AttackerMobility* module consumes. The file is formatted as a set of three 64-bit IEEE floating point numbers stored sequentially in binary describing each waypoint $\langle x'_i, y'_i, t'_i \rangle$.

Table 2.10: Tunable Attack Host Mobility Module Parameters

Parameter	**Default Value**	**Description**
wpfile	"middle.wp"	file from which to read waypoints.
s	5m/s	maximum speed at which to travel.

2.4 Chapter Summary

This chapter introduced the event-based simulation to be used when conducting MANET jamming experiments. The model exists within a defined area populated by terminal hosts which form the MANET while the attack hosts jam at the physical-layer. The terminal hosts make use of UDP/IP, DYMO, and IEEE 802.11g wireless devices to simulate the packet-level MANET operation. The motion of the terminal hosts, their traffic patterns, and selected destinations can be independently varied for repeatability and controllability. Attack hosts can move about the defined area and execute different jamming strategies, namely: constant, random, reactive, random reactive. This represents the needed fidelity to simulate the MANET jamming problem.

Chapter 3

Statistical Analysis

The stochastic nature of the event-based simulator presented in Chapter 2 requires that formal statistical analysis be applied to make meaningful observations about the MANET's operation under different physical-layer jamming attacks. These observations are based on features, (e.g., packet delivery radio (PDR), throughput, etc.) which are any aspect of the simulation that can be quantitatively measured. Within the MANET research community a number of ad-hoc approaches to statistical analysis are employed[24, 44]. However, these generally produce two critical issues, namely: ad-hoc removal of known start-up transients, and averaging across non-ergodic data.

An example of the first issue is the selection of an initial period of time to drop before analyzing the features[24]. The goal of this is to discard any start-up transients present. Unfortunately, there is no guarantee that steady-state will have been reached by that time. Furthermore, there is no guarantee that the given feature will indeed ever reach a steady-state.

The second issue arises when averaging the records of a feature when applying the Monte-Carlo method to reduce the uncertainty of the calculated statistics[44]. However, the stochastic nature of the feature means there will be random variations observed in each record due to different initial conditions. In order to produce meaningful results averaging must only be performed across ergodic data sets. Because a record may contain transients, never reach steady-state, or exhibit different steady-state behaviour there is the possibility that it will be non-ergodic with other records.

To ensure that the observations made about the the MANET jamming simulation are statistically meaningful the analysis performed must formally assess the stochastic nature of each feature prior to averaging. In this work all features are first individually tested for stationarity. This is done to identify steady-state behaviour, and if present

remove the start-up transients. The empirical distribution governing the value of each feature is then calculated and ergodic sets across the Monte-Carlo runs are then identified.

3.1 Measuring Simulation Features

By measuring different features of the MANET jamming simulation observations can be drawn from the quantitative data collected. However, these measurements are stochastic in nature and require a formal statistical analysis approach. Features are characterized by both a random value and a random sampling period. This is because measurements occur based on the activity of the simulation and its initial conditions.

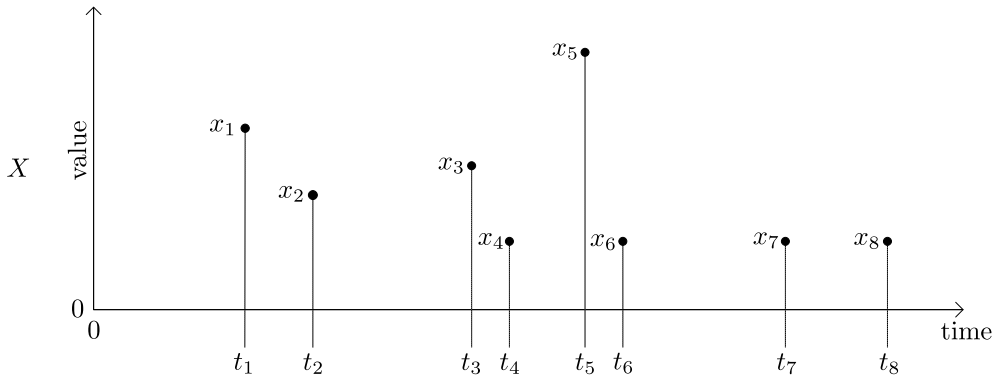


Figure 3.1: Measured Feature X

Within this work a measured feature of the MANET jamming simulation exists as a time-series random process. Furthermore, the measurements are discrete and assume their values at random intervals. The feature X, illustrated by Figure 3.1, is defined as the ordered set of J tuples, given by

$$X = \{\langle x_j, t_j \rangle \,|j = 1, \ldots, J\} \text{ where } t_j \geq t_{j-1} \tag{3.1}$$

where x_j is the value measured at time t_j over the interval $[0, T_{max}]$ of the observations. The values x_j taken by feature X are governed by an underlying distribution of unknown form, denoted as $P(x)$. While this work treats x_j as a scalar quantity the discussion that follows can be trivially extended to a vector $\vec{x}_j$. The feature X is further characterized by the time between measurements, (i.e., the sampling period

of the random process), $\tau = t_j - t_{j-1}$, where $t_0 = 0$, itself a random variable with an unknown distribution.

Due to the random sampling period any measurements of a feature made over different runs of the simulation, (i.e., not an exact recreation using the same configuration and initial conditions) cannot be directly compared. A method to mitigate this is to re-sample the feature, as

$$X(kT) = G[\{x_j \in X_n | t_j \in [kT, (k+1)T)\}] \tag{3.2}$$

where the constant sampling period $T >> \tau$ and $G[.]$ is an appropriately defined function, such as the sample mean. However, because the density of events with respect to time is of interest this method can only be applied when $max[\tau]$ is small when compared to $[0, T_{max}]$. As such, to compare one feature to another a different method must be applied. This work presents such a method through detecting steady-state behaviour and comparing the empirical distributions identified.

3.2 Detecting Steady-State Behaviour

In order to detect steady-state behaviour each feature is tested for stationarity. This process measures the statistical similarity, over time, of the underlying distribution which governs the values taken by a feature. Using this information start-up transients can be removed formally rather than in an ad-hoc manner. Moreover, this enables the estimated empirical underlying distribution to be calculated to enable comparisons between features.

Based upon the assumed presence of start-up transients in a given feature X the underlying distribution $P(x)$ is time-dependent, (i.e., $P_t(x)$ where $t \in [0, T_{max}]$). If the feature reaches steady-state then there exists a time $t^{SS} \in [0, T_{max}]$ after which the distribution can be considered time independent[1], (i.e., $P_t(x) = P(x)$). Hence, the feature X can be considered transient over $\left[0, t^{SS}\right)$ and stationary over the interval $\left[t^{SS}, T_{max}\right]$. The removal of start-up transients from the feature X can be accomplished through testing for stationarity to identify t^{SS} and discarding measurements prior to such time.

[1]If $P_t(x) = P(x)$ then all statistics calculated for $P_t(x)$ are also time independent.

3.2.1 Testing For Stationarity

Prior to removing the start-up transients a feature must first be tested for stationarity. This is done by testing the statistical similarity of the underlying distribution over time. Because this distribution is of an unknown form the statistical test employed must not rely on such assumptions, (e.g., Chi-squared goodness-of-fit test relies on a known analytical form of the distribution). Two tests which can be considered are the two sample Kolmogorov-Smirnov[8] test (KS-test) and the Anderson-Darling[8] test (AD-test). The AD-test assesses the cumulative difference between the two empirical distributions and ensures that all points are no more than ϵ apart. The KS-test considers only the maximum distance and tests to see of there is a point within the two empirical distributions where they differ by ϵ. Hence, the KS-test can pass when the AD-test fails, and the AD-test can pass where the KS-test fails. The KS-test is chosen for this work[2] and is given by

$$D = max\left[\left\{\left|\hat{P}^1(x) - \hat{P}^2(x)\right| \mid \text{for } \forall_x\right\}\right] \tag{3.3}$$

where $\hat{P}^1(x)$ and $\hat{P}^2(x)$ are two empirical distributions. The test statistics D indicates the largest distance between the two empirical distributions and is used to calculate the p-value of the test via the asymptotic Q-function. The test is considered to pass if the p-value is observed to be greater than the test significance α. If the test fails to reject the null hypothesis then the two samples are assumed to have been drawn from the same underlying distribution[3]. Conversely, if the two samples are observed to exhibit dissimilar behaviour then the null hypothesis is rejected. To be clear passing the KS-test only denotes that no evidence of dissimilarity was observed between the two empirical distributions.

Prior to testing, the feature X is split into the set of Q windows W_q of width W, as described in Section 3.2.2. The KS-test is then applied right-to-left over the windows, (i.e., $q = Q, \ldots, 1$) using the empirical distribution $\hat{P}_q(x)$ formed from each. Because comparing adjacent windows is not meaningful in a statistical sense[4] the empirical

[2]The implementation allows for substitution of the statistical test allowing for the AD-test, or some other suitable test, to be used in place of the KS-test.

[3]The KS-test does not give information as to the form of the underlying distribution. Rather it provides a test to determine if the two empirical distributions are drawn from the same, or different, distribution.

[4]Adjacent windows can be statistically similar to one another but dissimilar to previously tested windows, (i.e., $\{\hat{P}_Q(x), \hat{P}_{Q-1}(x)\}$ are similar, $\{\hat{P}_{Q-1}(x), \hat{P}_{Q-2}(x)\}$ are similar, but $\{\hat{P}_Q(x), \hat{P}_{Q-2}(x)\}$ are not).

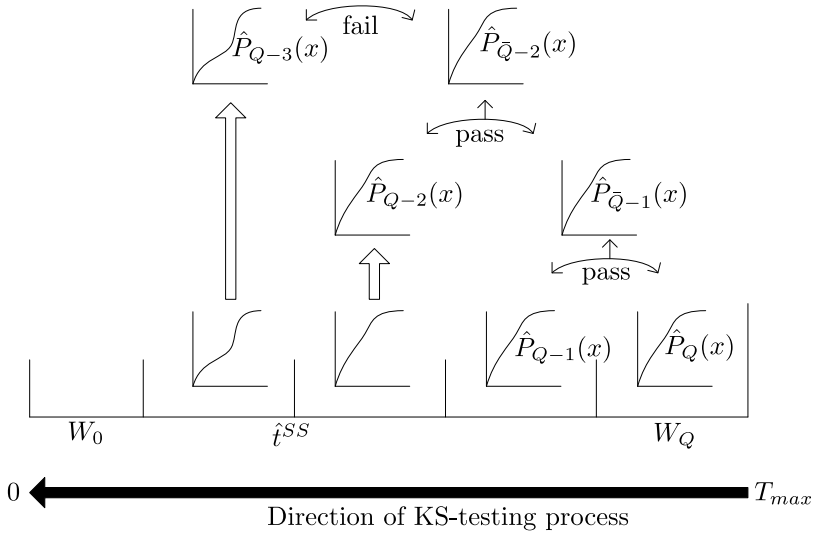


Figure 3.2: Right-to-left Stationarity Test

distribution of a window $\hat{P}_q(x)$ is compared to the average empirical distribution of all previously tested windows $\hat{P}_{\bar{q}}(x)$, given by

$$\hat{P}_{\bar{q}}(x) = \frac{1}{Q-q} \sum_{r=q+1}^{Q} \hat{P}_r(x) \tag{3.4}$$

where q is decreased each time the KS-test passes; Otherwise, the test for stationarity is stopped. This means that each time the KS-test passes a better estimate of the empirical underlying distribution is available as $\hat{P}_{q^-1}(x)$[8]. When the KS-test finishes and if $q < Q$, (i.e., at least two windows were similar) the estimated start time that steady-state was exhibited $\hat{t}^{SS} = T_{max} - Q^{SS}W$ is calculated from the number of statistically similar windows found $Q^{SS} = Q - q$.

It should be noted that the test applied, as per Algorithm (3), expressly does not denote a multiple hypothesis test. This is because each window is tested against the average of all windows to the right. That is to say $\hat{t}^{SS}$ only depends on a single application of the KS-test. Furthermore, this test operated under the assumption that $P(x)$ is constant within a window.

3.2.2 Selection of Window Width

The random sampling period of the feature means that the window width over which the feature exhibits stationarity is not known a priori. This means that the test for

Algorithm 3: Stationarity Test

$Q \leftarrow$number of windows
$q \leftarrow Q$
while $1 \leq q \leq Q$ **do**
  $q \leftarrow q - 1$
  $\hat{P}_q(x)$ from $\{x_j \in W_q\}$
  $\hat{P}_{\bar{q}}(x)$ from EQ. (3.4)
  $D \leftarrow$*KS-test*$\left(\hat{P}_q(x), \hat{P}_{\bar{q}}(x)\right)$
  if $\alpha \geq$*Q-function*(D) **then**
    $q \leftarrow q + 1$
    break
  end
end
$Q^{SS} = Q - q$

stationarity performed in Section 3.2.1 may fail at one width, but pass at another. Furthermore, the test may pass for more than one width and it is assumed that they will have same underlying statistical properties. As such, several factors must be considered when choosing the window width to use when calculating the averaged empirical underlying distribution for the feature.

Narrow windows which contain fewer measurements will have higher uncertainty in their empirical distributions. This implies that the selection of windows containing more measurements is preferable as the uncertainty in the empirical distributions being compared is less. This leads to the obvious extreme of selecting a window width sufficient to contain all steady-state values in just two windows. However, by selecting smaller windows many empirical distributions can be averaged to produce an estimate with lower variance[8] for the feature over the same interval. As such, a trade-off exists between the window width chosen and the number of windows to be averaged. Therefore, several different resolution levels are tested for stationarity prior to selecting the window width.

The process of determining which window width W_p to use involves testing the feature X at several resolution levels p as illustrated by Figure 3.3. The window width is a multiple of the maximum time between measurements, given as

$$W_p = 2^p \rho \cdot max\left[\tau\right] \text{ for } p = 0, \ldots \tag{3.5}$$

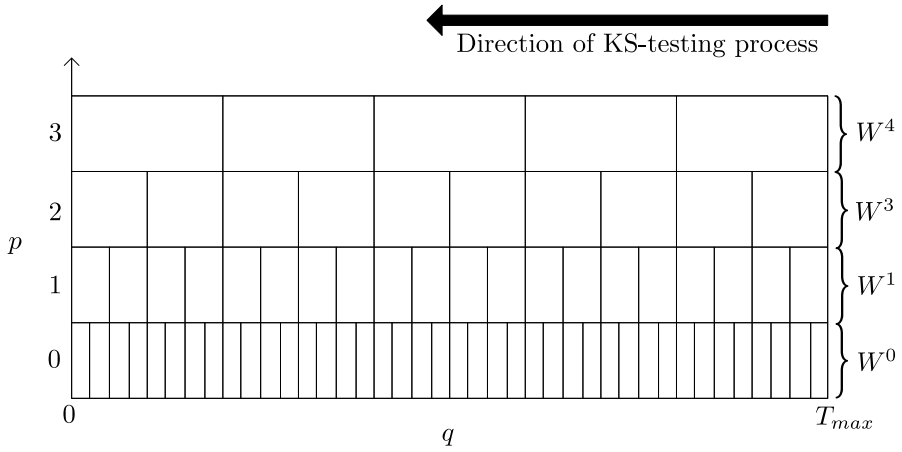


Figure 3.3: Expanding Window Search

where ρ is chosen to ensure windows contain a minimum number of measurements. Thus, as p increases the window width grows exponentially. At each resolution level the feature X is split into a set of non-overlapping windows $W^p = \left\{W_q^p | q = 0, \ldots, Q_p\right\}$ of equal width W_p, as per

$$W_q^p = \{\langle x_j, t_j\rangle \in X | t_j \in [t_{offset} + qW_p\ , t_{offset} + (q+1)W_p)\} \tag{3.6}$$

where the number of windows is $Q_p = \left\lceil \frac{W_p}{T_{max}} \right\rceil$ and the window offset $t_{offset} = T_{max} - (Q_p - 1)W_p$ is used to right-align the windows to T_{max}. This means that the left most window W_0^p will be narrower than W_p as needed. Because this may result in a window with high uncertainty it is excluded from the stationarity test and assumed to be part of the start-up transients. The highest resolution level is bound by the minimum number of windows $Q_p \geq Q_{min}$.

The test for stationarity is applied at each resolution level p considered after which the estimated start of steady-state $\hat{t}_p^{SS}$ and number of similar windows Q_p^{SS} are recorded. If a resolution level results in a steady-state period shorter than the minimum, $Q_p^{SS} W_p < T_{min}^{SS}$, it is considered to be transient and discarded. Otherwise, the p-value of the last KS-test to pass is recorded as D_p. Once all resolution levels are tested, the results of each are assessed to select the desired level p' at which steady-state was observed. This resolution level is then used to determine $\hat{t}^{SS} = \hat{t}_{p'}^{SS}$, $Q^{SS} = Q_{p'}^{SS}$, and $W = W_{p'}$ used when calculating the estimated empirical distribution.

The first step in selecting p' is to sort the resolution levels, in ascending order of $\hat{t}_p^{SS}$, and calculate the minimum value $\hat{t}_{min}^{SS} = min\left[\left\{\hat{t}_p^{SS} | \forall_p\right\}\right]$. The resolution level p

with the highest D_p is then chosen from $\forall_p\ \hat{t}_p^{SS} \in \left[\hat{t}_{min}^{SS}, \hat{t}_{min}^{SS} + \frac{T_{max}}{100}\right]$. This is done because the unknown form of $P(x)$ prevents confidence intervals from being employed to determine which window width to use when forming the estimated empirical distribution $\hat{P}(x)$ of X.

3.2.3 Removal of Start-Up Transients

With the start of steady-state behaviour identified, and selected window width, the start-up transients can be removed from a feature. This provides a formal method in place of existing ad-hoc approaches which cannot guarantee start-up transients are no longer present. Based upon the statistical analysis conducted it can be asserted that all measurements performed after the detected start of steady-state are drawn from the same underlying distribution, (i.e., no evidence has been observed to the contrary). In cases where no steady-state behaviour is detected the feature is considered to be entirely transient and discarded.

To remove start-up transients for the feature X all values x_j where $t_j < \hat{t}^{SS}$ are dropped. The resulting feature can then be split into the set of windows containing steady-state data $W^{p'} = \left\{W_q^{p'} | q = Q_{p'} - Q^{SS}, \ldots, Q_{p'}\right\}$ of width $W_{p'}$, as per EQ. (3.6). The empirical distribution $\hat{P}_q^{p'}(x)$ is then formed from each window $W_q^{p'}$ prior to calculating the estimated empirical distribution $\hat{P}(x)$, derived from EQ. (3.4) as

$$\hat{P}(x) = \frac{1}{Q_{p'} - Q^{pass} + 1} \sum_{q=Q^{SS}}^{Q_{p'}} \hat{P}_q^{p'}(x) \tag{3.7}$$

3.3 Statistical Averaging of Feature Records

When averaging feature records produced by the Monte-Carlo method, their ergodicity must first be assessed. This is due to the random variations in each record having the potential to exhibit dissimilar steady-state behaviour. Because of this a formal method is required to measure the ergodicity, of all pairs, of records to be averaged. In addition, more than one set of ergodic records can exist meaning that the feature is governed by different modes of behaviour. As such, to be statistically meaningful more than one average may be produced for a feature to describe its behaviour.

The records collected using the Monte-Carlo method are produced by performing independent runs of the simulation with the same configuration but different initial

conditions. This is achieved by the selection of different random seeds for one or more of the random number streams used by the simulator each time it is run. Within this context the feature X is defined as an ensemble $X = \{X_n | n = 1, \ldots, N\}$ of N records X_n, as per EQ. (3.1).

Only records X_n which were found to exhibit steady-state behaviour, as per Section 3.2, are considered when testing the feature X for ergodicity. Under this assertion, the test identifies which records X_n exhibit statistically similar steady-state behaviour based on their estimated empirical distribution $\hat{P}_n(x)$. Using this information pairs of records which are all statistically similar to one another can be grouped together to form sets of ergodic records X_n using a clustering algorithm. This enables different modes of behaviour for the feature X to be detected and ensures only ergodic records are averaged when calculating the averaged estimated empirical distribution.

3.3.1 Testing for Ergodicity

The test for ergodicity measures the statistical similarity of all pairs of records for the feature X prior to averaging. It is performed on a feature-by-feature basis and takes into consideration only records within the ensemble which exhibited stationarity. Continuing from the test for stationarity, presented in Section 3.2.1, each record is represented by an estimated empirical steady-state distribution. As such, the two-sample KS-test is again used to identify statistically similar pairs of records. Any record which did not exhibit stationarity is not included in the ergodicity testing as no steady-state behaviour was detected.

The ergodicity of feature X is measured by applying the KS-test to all combinations of record pairs $\forall n, m < X_n, X_m >$ where $n \neq m$, as per

$$e_{n,m} = max\left[\left\{\left|\hat{P}_n(x) - \hat{P}_m(x)\right| \mid \text{for } \forall_x\right\}\right] \tag{3.8}$$

where $e_{n,m}$ is the p-value returned by the KS-test. If X_n or X_m are non-stationary records then $e_{n,m} = 0$. It should be noted that this is expressly not a multiple hypothesis test as pairs of records are tested individually. Moreover, no claim is made as to the similarity of the pairs with respect to one another.

A graph $\mathcal{G}= \{X, E\}$ is then constructed using the feature ensemble X as the set of nodes and the set of p-values as edges $E = \{e_{n,m} | \forall_{n,m} \text{ where } n \neq m\}$. This ergodicity graph $\mathcal{G}$ is a fully connected bi-directional weighted graph which represents the statistical similarity between all records within the ensemble X. To expose different

modes of behaviour all edges which are weighted less than or equal to the KS-test's significance level, $e_{n,m} < \alpha$, are removed from the set E. This produces a graph where only statistically similar records share an edge.

3.3.2 Clustering Ergodic Records

Determining if there is more than one mode of behaviour governing a feature can be viewed as a clustering problem. The method employed to perform the clustering of records is based the ergodicity graph generated in Section 3.3.1 and is performed in three steps. The first of these finds all subsets of records from the feature ensemble that were not statistically dissimilar from one another, (i.e., the maximal cliques of the ergodicity graph). The second and third steps determine the greatest common subsets of records found within the list of maximal cliques.

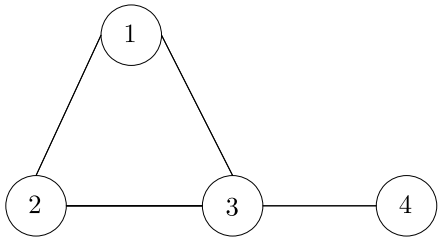


Figure 3.4: Graph With Five Cliques and Two Maximal Cliques

In graph theory, a clique is defined as a set of nodes which are completely connected, (i.e., every node shares an edge with all other nodes in the group)[2] while maximal cliques refer to the largest set of completely connected nodes within a graph and are a proper subset of the cliques. For example, the graph shown in Figure 3.4 contains five cliques: $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$, $\{3, 4\}$ and $\{1, 2, 3\}$. Of these, two are maximal cliques $\{1, 2, 3\}$ and $\{3, 4\}$ as no other nodes of the graph can be added without violating the all-connected condition. With respect to the ergodicity graph $\mathcal{G}$ each maximal clique defines the largest sets of records X_n which are all statistically similar to one another, (i.e., the sets of ergodic records for the feature X).

The Bron-Kerbosch (BK) algorithm, as per Algorithm (4), is used to find all maximal cliques within the ergodicity graph $\mathcal{G}$. The algorithm operates based on three input sets: the set of nodes currently in the maximal clique R, the set of possible nodes to test for membership in the maximal clique P, and the set of nodes to exclude from the maximal clique X. Using these three inputs the algorithm chooses

some node n to add to the maximal clique and then recursively iterates over all nodes in P which are not neighbours of n, $N(n)$.

Algorithm 4: Bron-Kerbosch With Pivoting

Input: $P = \emptyset$,$R = V$,$X = \emptyset$
Output: C
BronKerbosh(R, P, X)
if $P = \emptyset$ *and* $X = \emptyset$ **then**
  $C = C \bigcup \{R\}$
else
  choose node n from $P \bigcup X$ with most neighbours
  foreach *node* $v \in P - N(n)$ **do**
    BronKerbosh$(R \bigcup \{v\}, P \bigcap N(v), X \bigcap N(v))$
    $P = P - \{v\}$
    $X = X \bigcup \{v\}$
  end
end

The output of the BK algorithm is a set of maximal cliques $C = \{C_u|$ for $u = 1, ..., U\}$ found in the ergodicity graph $\mathcal{G}$, where there are U maximal cliques C_u. The worst case behaviour of the BK algorithm can result in $U = N^{\frac{3}{N}}$ maximal cliques being present. This occurs in graphs where all nodes share an edge with all other nodes but one. Due to the potential for a large numbers of maximal cliques to be present two additional steps are performed as part of the the clustering algorithm to reduce the number of ergodic modes. The first of these goes through all of the stationary records $X_n \in X$ and calculates the intersection of all maximal cliques of which X_n is a member. This produces the intermediate set of ergodic modes $M' = \{M_{b'}|b' = 1, \ldots, B'\}$ with B' modes $M_{b'}$, as per

$$M_{b'} = \bigcap_{\forall_u} \{C_u|X_n \in C_u\} \tag{3.9}$$

where $B' \leq N$ and all modes contain at least two records, (i.e., $\forall_{b'}\ |M_{b'}| \geq 2$). It should be noted that this can result in a record X_n being a member of two or more modes $M_{b'}$.

This can be avoided by removing all records contained in $M_{b'}$ from all cliques C_u before creating $M_{b'+1}$; however, doing so causes the modes $M_{b'}$ to be dependent on the order in which they are created. As such, there can exist two modes where one is a subset of the other, resulting in duplicate modes, (i.e., $M_{b'} \subseteq M_{d'}$ where $b' \neq d'$).

To resolve this the clustering algorithm performs a third step to produce a final set of ergodic modes $M = \{M_b | b = 1, \ldots, B\}$ by removing all modes which are subsets of others.

3.3.3 Averaging Ergodic Records

With the clustering process completed, statistical averaging can be performed for each of the ergodic sets of records, or modes, identified. As such, more than one average will be produced for a given feature when multiple modes of behaviour are detected. This is done to avoid discarding information prior to observing the behaviour of the feature. Two types of averages are produced for each ergodic mode, these are: the averaged estimated empirical distribution, and the averaged sampled time-series. The latter is provided so that characteristics of the feature's behaviour with respect to time can be observed.

The first average produced for each ergodic mode, $M_b \in M$, is the averaged estimated empirical distribution $\hat{P}^{\bar{b}}(x)$, as per

$$\hat{P}^{\bar{b}}(x) = \frac{1}{|M_b|} \sum_{n \in M_b} \hat{P}_n(x) \tag{3.10}$$

where $\hat{P}_n(x)$ is the estimated empirical distribution of each record X_n calculated from EQ. (3.7).

The second average produced for the ergodic mode M_b is the averaged sampled time-series. Because this average is time dependent, it is only valid over the common period of ergodic behaviour for all records of the mode $\hat{t}_b^{ES} = max\left[\left\{\hat{t}_n^{SS} | n \in M_b\right\}\right]$ which is the maximum start of steady-state for all records in the mode. Furthermore, because of their random sampling period τ_n, each record X_n is first re-sampled as $X_n(kT)$, via EQ. (3.2), with a user specified function $G[.]$ and the largest window width of the records $T = max\left[\{W_n | n \in M_b\}\right]$ as the common sampling period. The averaged sampled time-series $X^{\bar{b}}(kT)$ is then calculated, according to

$$X^{\bar{b}}(kT) = \frac{1}{|M_b|} \sum_{n \in M_b} X_n(kT) \text{ for } kT \in \left[\hat{t}_b^{ES}, T_{max}\right] \tag{3.11}$$

3.4 Composite Features

Several features of interest with respect to the MANET jamming simulation are defined in terms of other features, (e.g., PDR is the ratio of packets received to packets sent). As such, directly measuring these features from the simulation can be problematic or impractical. Moreover, as per Section 3.1 features measured have a random sampling period which prevents direct comparisons or combinations to be made between two or more features. To overcome this issue the component features are first re-sampled to have the same constant sampling period. The composite feature is then formed and tested for stationarity and ergodicity as described in Section 3.2 and Section 3.3.

The composite feature X is defined as the combination of two other features Y and Z. This definition can be trivially expanded to encompass three or more features. In order to combine the features while maintaining as many measurements as possible the constant sampling period $T = max\left[\{max\left[\tau\right] \mid \text{for } Y, Z\}\right]$ is taken as the largest time between any two measurements of the features involved. The component features are then re-sampled into $Y(kT)$ and $Z(kT)$ respectively, using EQ. (3.2) where the function $G[.]$ is feature specific, (e.g., PDR uses $G[.] = |\{x_j | t_j \in [kT, (k+1)T)\}|$ which counts the number of measurements in sampling period k). The features are then combined to form $X(kT)$, as per

$$X(kT) = G\left[Y(kT), Z(kT)\right] \tag{3.12}$$

where the function $G\left[.\right]$ is again feature specific, (e.g., PDR uses $G\left[.\right] = \frac{Y(kT)}{Z(kT)}$ to calculate the ratio at each interval k).

3.5 Validation of The Statistical Analysis

In order to have confidence that the statistical analysis presented in this work is implemented correctly, a number of tests were conducted. These tests are performed on a randomly generated feature crafted to exhibit two modes of behaviour. As such, the randomly generated records of the feature have a known analytical which allows the accuracy of the analysis results to be validated through application of statistical tests.

The statistical analysis is performed on a single feature composed of 500 randomly generated records. The feature is characterized by two modes of behaviour, a random

interval between measurements, and the duration of start-up transients. The modes differ in the distribution governing the values taken by a record and occur with equal probability: $P^1(x)$ which is uniform over $[0, 80]$ and $P^2(x)$ which is exponential with mean 5. The random time in seconds between each value measured for a given record is exponentially distributed with mean 0.1. Under both modes, the start-up transients in each record last for a random interval uniformly distributed over $[150, 450]$ of the observed $T_{max} = 1800$ seconds. The value of each record for the duration of the start-up transients is -5 which is outside the range of normal values. This is done to make the presence of transient data in the statistical analysis results clearly visible.

Performing the statistical analysis on a feature governed by a known form allows the accuracy of results to be quantified. This is done by applying the Chi-square goodness of fit test, with a 95% confidence interval. The tests applied focus on the measured characteritics of the feature which can be compaired to the known form of the records analyzed. Validation of the implemented analysis is achieved by showing the results are statistically similar to their expected form.

3.5.1 Steady-State Behaviour

The statistical analysis begins with detecting steady-state behaviour of the feature as described in Section 3.2. To validate this portion of the statistical analysis, the characteristic of interest is the detected start of steady-state behaviour $\hat{t}_n^{SS}$ for each record X_n in the ensemble X. Because all records are tested individually, the distribution for the start time of steady-state behaviour should be statistically similar to the expected form, (i.e., uniform over the interval $[150, 450]$ seconds).

From Figure 3.5 it can be observed that the empirical distribution has a minimum value of 157.735, a mean of 369.802, a median of 366.562, and a maximum of 1281.774 seconds. A total of 111 records fall outside of the expected range, all occurring after 450 seconds. While in general the records are observed to be to the right of the expected values this can be explained due to $\hat{t}_n^{SS}$ being an estimate of the actual t_n^{SS} for the feature X_n. More specifically, $\hat{t}_n^{SS}$ may be an over-estimate resulting from the process of selecting the resolution level with the highest statistical similarity in the rightmost window, as presented in Section 3.2.2. However, further examination of the outliers is needed to provide insight into their occurrence outside the expected range.

To further examine the behaviour of $\hat{t}_n^{SS}$ two records are considered: X_{165} which is the largest of the outliers, and X_{142} which is within the expected range. Of interest is

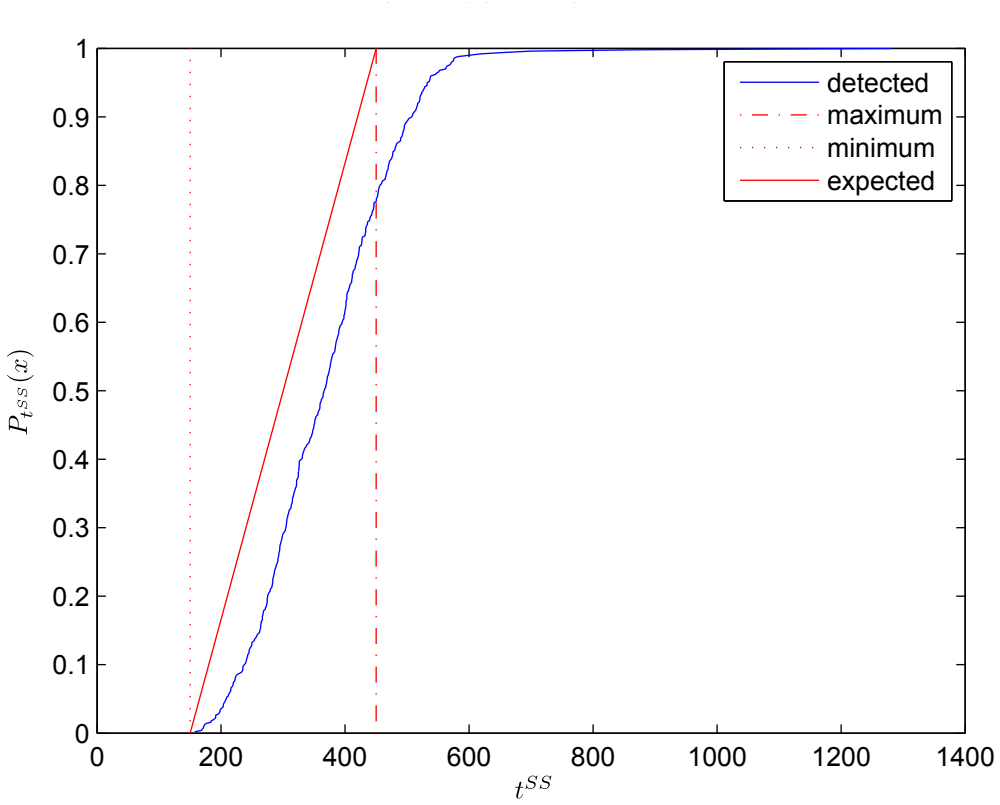


Figure 3.5: Empirical CDF for Start Time of Steady-State Behaviour

whether or not the selection of the resolution level played any part in the presence of outliers. To accomplish this a scatter plot of the start time of steady-state behaviour $\hat{t}^{SS}_{n,p}$ versus the statistical similarity $D_{n,p}$ for each resolution level is produced using trace information recorded by the statistical analysis. Figure 3.6 and Figure 3.7 show the trace information for X_{165} and X_{142} respectively with the selected resolution level circled in red. It can be observed that for X_{165} no resolution level was found where $\hat{t}^{SS}_{165,p} < 1242.658$ while the actual $t^{SS}_{165} = 422.144$ seconds. Record X_{142} is seen exhibiting a $\hat{t}^{SS}_{142} = 219.200$ which is relatively close to the actual $t^{SS}_{142} = 217.203$ seconds. From this, it can be concluded that the random nature of $x_j \in X_{165}$ was the cause of the observed behaviour.

Applying the Chi-Square goodness of fit test to the empirical distribution $\hat{P}_{\hat{t}^{SS}}(x)$ yields $\chi^2 = 1.4833 \times 10^5$ which fails to reject the null hypothesis within the 95% confidence interval for 495 degrees of freedom. As such, the start of steady-state behaviour detected by the statistical analysis within this work is considered to be accurate.

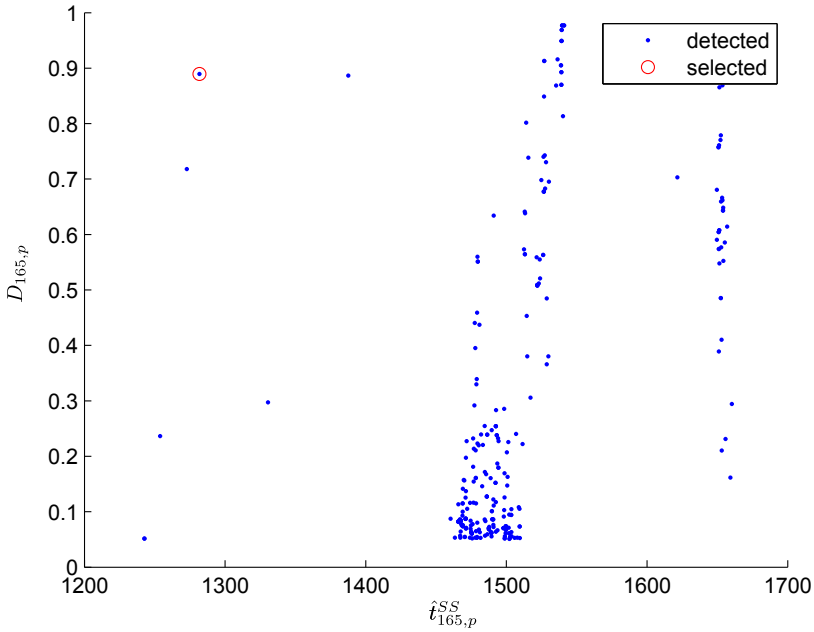


Figure 3.6: Resolution Levels for X_{165}

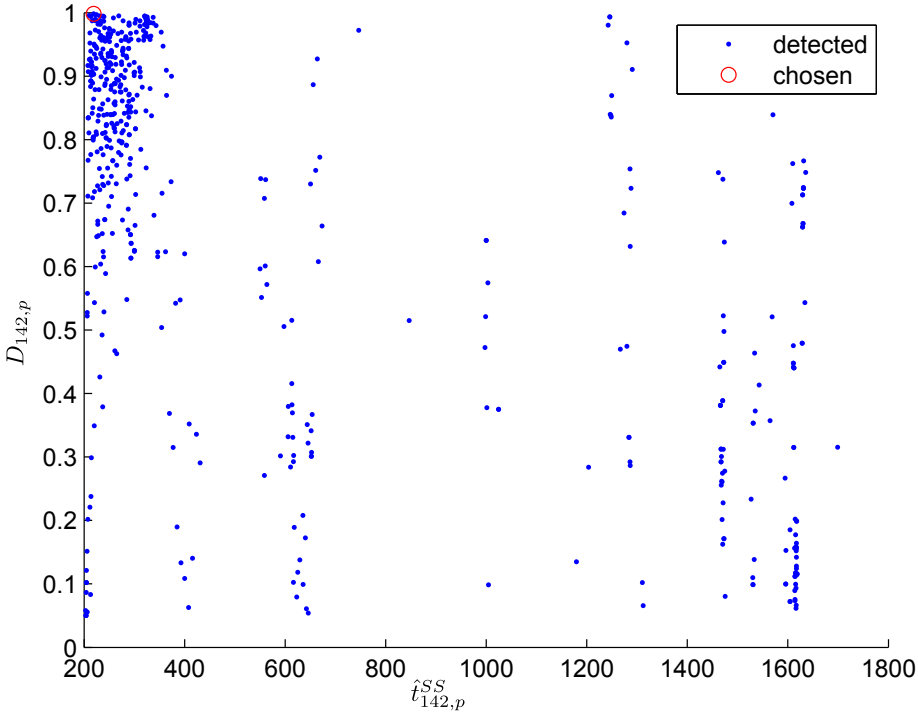


Figure 3.7: Resolution Levels for X_{142}

3.5.2 Modes of Behaviour

The second portion of statistical analysis performed focuses on the detection of ergodic groups of records within the feature ensemble as described in Section 3.3. To validate this portion of the statistical analysis the characteristics of each mode detected are considered against the known form, (i.e., 0.5 probability of being governed by $P^1(x)$ or $P^2(x)$).

Table 3.1: Stationarity and Ergodicity of Records

Record X_n			**Mode** M	
Non-Stationary	**Stationary**	**Non-Ergodic**	**Count**	**Largest**
$0, 3$	497	0	71	129

As detailed in Table 3.1, of the 500 records observed 497 exhibited stationarity while 3 were found to be transient. The pair-wise statistical similarity of each record X_n relative to all others can be observed from the contour plot Figure 3.8, where 1 is high similarity and 0.05 to 0 is dissimilar. The plot is read from left-to-right and top-to-bottom. The number of records with a statistical similarity greater than or equal to a given p-value to a record can be read from the contour lines' x-axis position. The top most record has the highest number of statistically similar records, and the bottom the least. This information suggests that the majority of records are statistically similar to 250 others. This supports the expectation that records are drawn from two different modes of behaviour with equal probability.

The ergodicity graph $\mathcal{G}$ created from the 497 stationary records X_n is seen to contain more than 1.748×10^5 maximal cliques which are reduced to 71 ergodic modes by the clustering algorithm. The modes detected have a maximum size of 129, a mean of 57, a median of 51, and a minimum of 8 records. If modes containing fewer than 40 records are excluded then there are a total of 51 modes which contain reasonable estimations of the underlying distributions $P^1(x)$ and $P^2(x)$ governing the values taken by feature X. Figure 3.9 shows the largest detected exponential and uniform mode, where M_1 contains 117 exponential records X_n and M_{11} contains 96 uniform records.

Of the 51 modes considered to be reasonable estimates of $P^1(x)$ and $P^2(x)$, 27 are observed to contain a small portion of transient data; however, this accounts for less than 0.63% of the distribution in the worst case. In all cases the 71 averaged

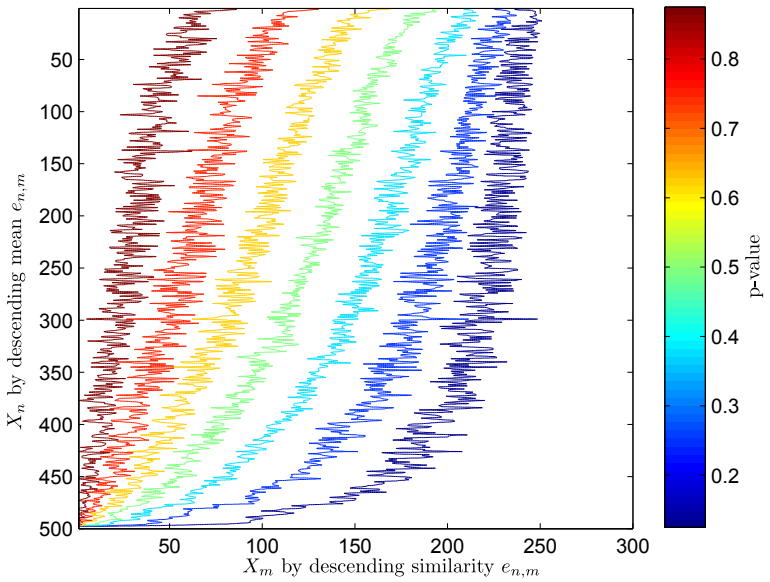


Figure 3.8: Statistical Similarity of Stationary Records in X

estimated empirical distributions are observed to pass the Chi-Square goodness of fit test with respect to their underlying distribution.

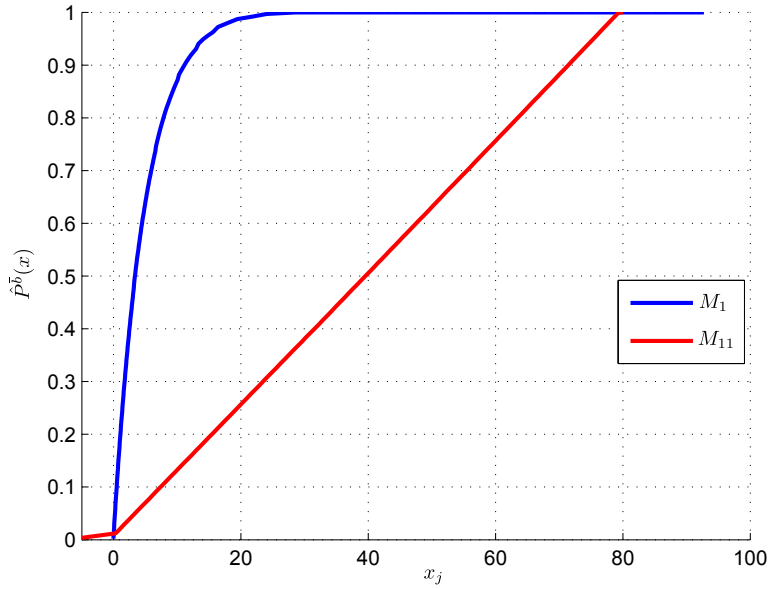


Figure 3.9: Detected Distributions for Modes M_1 and M_{11} of X

3.6 Chapter Summary

This chapter introduced the statistical analysis method to be used when assessing the MANET operations under different jamming strategies. In particular it focuses on two key points of analysis: a formalized method to remove start-up transients, and averaging over empirically ergodic data. As such, observations based upon periods of steady-state and ergodic modes of behaviour provide statistically meaningful information about the measured features. Furthermore, the implementation of the method is validated by testing outputs for a randomly generated feature of known form.

Chapter 4

Attacker Experiments

This chapter describes the simulation based experiments created to make an initial assessment of the MANET jamming problem. The four physical-layer attack strategies to be assessed are: constant, random, reactive, and random reactive jamming. Of interest is how each strategy behaves when configuration parameters of the attack host are varied under different scenarios, in particular: jamming range, motion, and multiple attackers. These three scenarios are considered versus MANET scale. This serves to exposes how the number of hosts forming the MANET impacts the effectiveness of the four jamming strategies under each scenario.

4.1 Method

To enable comparison between the results of experiments each is performed against a baseline MANET. That is to say the actions of the terminal hosts, (i.e., how each moves and generates network traffic) do not vary by experiment but only by repetition. In this work, two baseline MANET simulations, as per Chapter 2, are considered to assess the strategies effectiveness at different scales. The two MANETs, one large one small, share the same terminal host configuration parameters and vary only in the size of the defined area and number of terminal hosts within it.

With respect to both baseline MANETs, three scenarios are constructed to evaluate each of the strategies. The first scenario considers how the jamming range used by one motionless attack host influences the four strategies impact on MANET operations. The second scenario considers the use of motion by a single attack host and how it impacts the effectiveness of the strategies while decreasing detectability. The

last scenario uses multiple attack hosts employing motion to measure the impact of multiple jamming sources under each strategy. Variations of each scenario are performed to provide multiple data points from which to make comparisons about how each strategy behaves relative to the two baselines.

For each experiment conducted, 75 repetitions are performed resulting in more than five thousand individual simulations. These simulations are instrumented to generate the feature records needed to asses the four strategies. Additionally, over thirty different features are measured for each simulation. This presents an estimated eight CPU years of processing time, consuming four terabytes of storage, to perform both simulation and statistical analysis. Furthermore, CPU time and storage needs are expected to vary depending on the behaviour of the simulation or feature data being analyzed. As such, the computing resources of two WestGrid facilities are leveraged: Hermes to run simulations and Orcinus to perform the statistical analysis on each experiment. This is done due to the current storage limitations of the Orcinus cluster.

4.2 Baseline MANET

By exactly reproducing the same underlying MANET across different experiments the baseline can be used to compare results. In this work, two baseline MANETs are considered: one large and one small. The large MANET is simulated as 360 terminal hosts within an area of $1,500$ by $1,000$ meters, while the small MANET is comprised of 50 terminal hosts within a 600 by 300 meter area. This is the only differences between the two baseline simulations. Both employ the same terminal host configuration, including mobility and network traffic generation. The different scales serve to expose how the MANET operations change in response to the jamming attacks.

Motion of terminal hosts in both baselines is configured, as per Section 2.2.1, such that they move at a constant speed of 5 meters per second and select the random waypoint model over the random walk with probability 0.01. The small MANET has an average of 278 terminal hosts per square kilometre while the large MANET has a terminal host density of 244. In both cases at no time is the MANET, (i.e., the largest connected group within the area) comprised of fewer than 80% of the terminal hosts. This ensures that the MANET does not encounter pathelogical topologies which would render it non-operational. Moreover, this ensures that each terminal

host has on average 7 neighbours which provides the connectedness needed to support multiple routes.

Terminal hosts are configured, as per Section 2.2.3, to have a single network traffic source, (i.e., simulated application-level traffic). The On/Off source is configured to generate messages at a Pareto distributed interval, with parameters $a = 3$ and $b = 5$. At the start of each interval a destination is selected, as per Section 2.2.2, with a 0.05 probability of choosing from all terminal hosts instead the local group. The terminal hosts then issue $2KB$ of data in two $1KB$ segments to the network-layer for delivery to the destination. As such, the aggregate network traffic of the small MANET will be much lower than that of the large MANET.

The IEEE 802.11g wireless device employed by the terminal hosts to exchange radio signals are configured, as per Section 2.2.6, to have a communication range of approximately 98 meters. The radio signals are transmitted at a frequency of 2.4 gigahertz using 0.1 milliwatts of power. They are received with a sensitivity level of -90dBm and a minimum signal-to-noise ratio of 4. The signals propagate through the wireless channel, from Section 2.2.7, with a path-loss coefficient $\alpha = 2$.

4.3 Attack Scenarios

The four attack strategies to be considered are assessed under three scenarios for each of the baseline MANETs. The goal of which is to identify if different parameters controlling the attack hosts benefit or hinder the strategies. The parameters selected are chosen to increase disruption from jamming or reduce detectability of the attack, these are: jamming range, movement, and number of attack hosts. The three scenarios encompass the experiments created to observe how variation in the configuration parameters influence the each strategy's impact on baseline MANET operations at different scale. The attack hosts are configured as shown in Table 4.1[1] for each of the four strategies. Unless otherwise specified the attack hosts use the same configruation paramters as the terminal hosts of the two baselines. Full configuration details for all experiments are provided in Section A.3.

[1] *onPeriod* values are taken based on the mean broadcast duration measured for terminal hosts of the two baseline MANETs.

Table 4.1: Jamming Strategy Attack Host Radio Configuration

Strategy	***mode***	***onPeriod***	***offPeriod***	***Pjam***
Constant	0	50ms	–	–
Random	1	$U\,(38, 56)\,\mu$s	$U\,(0, 50)$ms	–
Reactive	2	46μs	–	1
Random Reactive	2	46μs	–	0.5

4.3.1 Jamming Range

The purpose of the jamming range scenario is to assess how the power used by attack hosts influences each strategy used against the baseline MANETs, (e.g., doubling the jamming range will increase the number of effected terminal hosts geometrically). However, jamming range is dependent on transmission power which is highly visible to detection. As such, it is also desirable to consider if lowering the jamming range will impact the overall effectiveness of the strategy.

Four experiments for each baseline MANET are constructed to provide an initial assessment of the impact jamming range has on the strategies. The attack hosts in each experiment are configured, as per Section 2.3.1, to use 0.0253, 0.1, 0.227, and 0.404mW respectively. This gives give the attack host a jamming range of 50, 100, 150, and 200 meters respectively.

In each experiment conducted against the small and large baseline MANET the single attack host is positioned at $\langle 300, 150\rangle$ and $\langle 750, 500\rangle$ respectively. Using the averaged 261 terminal hosts per square kilometre, of both baseline MANETs, the number of terminal hosts expected to be impacted by the attack host can be calculated, as shown in Table 4.2.

Table 4.2: Transmission Power Levels for Attack Hosts

Range (m)	**Power** (mW)	**Area** (km^2)	**Terminal Hosts**
50	0.0253	0.0079	2
100	0.1	0.0314	8
150	0.227	0.0707	18
200	0.404	0.1257	33

4.3.2 Movement

The movement scenario is used to assess how using the same mobility model as the MANET influences the strategies considered. More specifically, attack hosts move randomly in the same way as terminal hosts while performing the different jamming strategies to avoiding detection and impact different areas of the MANET. However, due to the random nature of the terminal host motion the attack hosts make use of predefined waypoints to avoid any pathological movement, (e.g., jamming the edge of the MANET will reduce the number of effected terminal hosts).

Two experiments for each baseline MANET are created to evaluate how employing the same movement patterns as terminal hosts impact each strategy. The attack host in each experiment is configured, as per Section 2.3.2, to follow a predefined path. In particular, the paths considered are randomly generated, according to Section 2.2.1, and uses the default configuration parameters given in Table 2.1. The generated paths are then manually evaluated for desirable characteristics, namely: the path covers a large area over the simulated interval and avoids positions near the edge of the area. For example Figure 4.1 provides a visualization of one of the paths selected for the attack host to follow for the large baseline.

4.3.3 Multiple Attackers

The multiple attacker scenario considers how the four strategies impact the baseline MANETs when two or more attack hosts are considered. That is to say, multiple attack hosts move randomly about the simulated area while performing the jamming strategies. By spreading out, larger areas of the MANET are effected reducing the number of available routes between terminal hosts. As such, it has the potential to disconnect a larger number of terminals from one another.

Three experiments are created for each baseline to evaluate the impact of multiple attack hosts on MANET operations. The number of attack hosts is configured to be 2, 3, and 4 and their motion is again randomly generated, as in Section 4.3.2. The paths chosen depend on the number of attack hosts and positions them away from each other to minimize overlap and spread their impact over a larger area of the MANET. Figure 4.2 illustrates the small baseline MANET two attack host experiment while Figure 4.3 shows how their relative positions change over time.

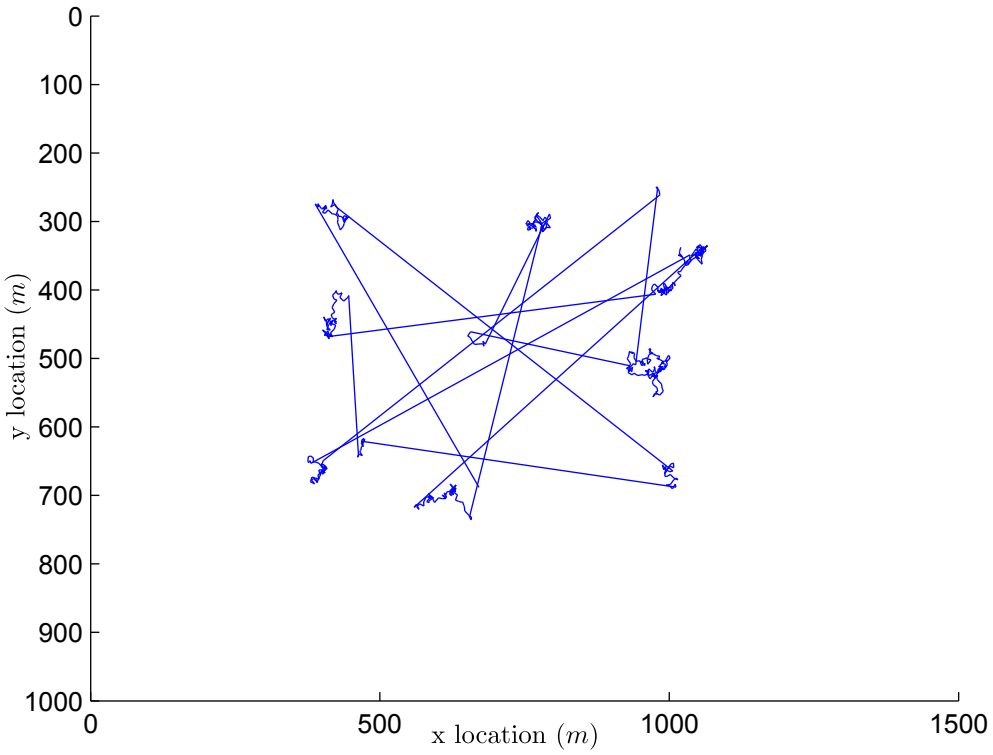


Figure 4.1: Attack Host Path for Large MANET

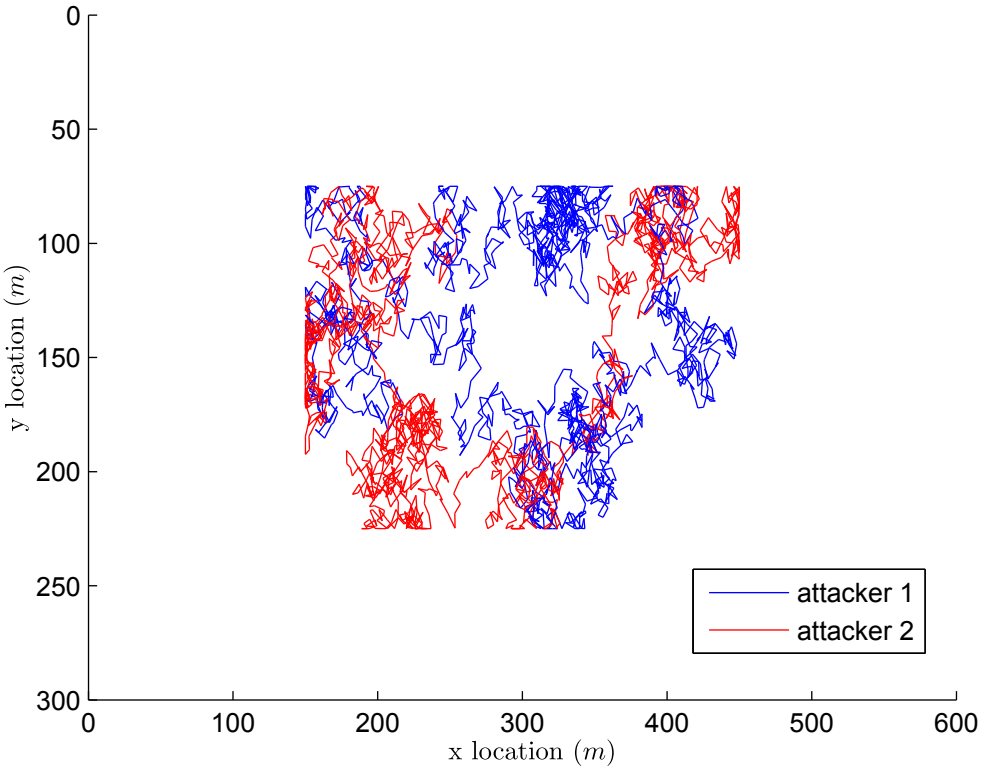


Figure 4.2: Paths Followed by Two Attack Hosts

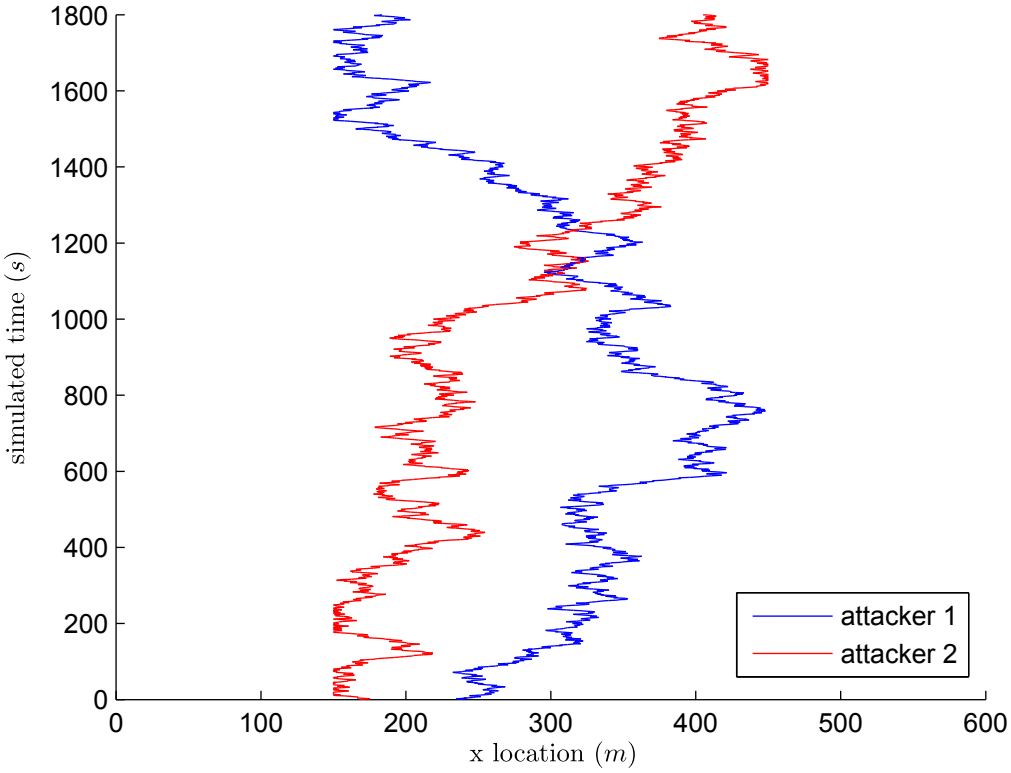


Figure 4.3: X Position of Two Attack Host vs Time

4.4 Evaluation Criteria

The four strategies are evaluated based on the features observed through statistical analysis of their respective experiments. To avoid undue complexity when assessing experiments a set of six primary features are considered, which are commonly reported in MANET research. These six comprise only a portion of the more than thirty tested by the statistical analysis. The remaining features serve to provide additional insight as to the per-experiment behaviour if needed. This enables a detailed view of various application, network, routing, and radio behaviours to be observed. The six primary features are:

1. Route Availability (X^{RA}) - the ratio of requested routes found in the routing table of the issuing terminal host
2. Packet Delivery Ratio (X^{PDR}) - the ratio of routable packets delivered to the destination terminal host
3. Normalized Routing Overhead (X^{NRO}) - the ratio of sent control, (i.e., routing) packets to received data packets in the MANET
4. Delivery Delay (X^{DD}) - total amount of time taken to deliver a message from source to destination terminal host, measured at the application-layer
5. Hops Travelled (X^{HT}) - number of terminal hosts used to relay the packet from source to destination through the MANET, including the destination
6. MANET Energy Usage (X^{MEU}) - amount of power in $\frac{mW}{s}$, measured from the transmission power and broadcast duration of all terminal hosts

The statistical analysis is performed using a KS-test significance level $\alpha = 0.05$ for both the stationarity and ergodicity tests. Furthermore, when detecting steady-state behaviour the minimum number of window $Q_{min} = 5$ and the minimum period is $T^{SS}_{min} = 100$ seconds. The assessment of each strategy is based upon the results of this analysis and focus on the largest ergodic mode reported. The number of records in the largest ergodic mode detected for each feature, as per Section 3.3, has a direct influence on the significance of the calculated averaged estimated empirical distribution. That is to say, an observation based on a mode containing fewer than

10 records will not be meaningful, while one comprised of more than 40 will provide a reasonable estimate of the feature behaviour[1].

[1]The number of records needed to produce a meaningful estimate is dependent on the form of the distribution. In general the more complex the distribution is the more records are required.

Chapter 5

Evaluation of Attack Strategies

This chapter presents an evaluation of the four physical-layer jamming strategies, namely: constant, random, reactive, and random reactive jamming. The evaluation is made based on the results of the statistical analysis performed and first considers the two baseline MANETs. The four strategies are then evaluated across the three scenarios individually to assess how they impact each of the baseline MANETs. When multiple modes of behaviour are reported by the statistical analysis for a single feature the evaluation will focus on the largest ergodic mode found. In total, more than thirty different features are available to observe; however, the evaluation focuses on six primary features, as presented in Section 4.4. The full set of results is provided in Appendix B for all experiments performed.

5.1 Experiment Resource Usage

The 5550 simulations performed as part of the experiments conducted in this work were processed using the Hermes cluster. The system is housed at the University of Victoria and consists of 84 nodes tuned for serial jobs. Each node consists of eight 2.67GHz Intel Xeon X5550 processors, 24GB of physical memory, and a shared 192TB file system. The Hermes cluster shares infrastructure with the Nestor cluster, providing a further 288 nodes tuned for parallel jobs. Table 5.1 and Table 5.2 shows the per-simulation CPU time and storage usage for small and large baseline MANETs respectively.

The data produced by the simulations was then moved to the Orcinus cluster where the statistical analysis of the 74 experiments was performed. The cluster is

Table 5.1: Small Baseline Simulation Resource Usage

Strategy	Simulations	CPU Time per	Storage per
Baseline	75	7 minutes	82MB
Constant	675	3 minutes	14MB
Random	675	7 minutes	97MB
Reactive	675	7 minutes	86MB
Random Reactive	675	7 minutes	86MB

Table 5.2: Large Baseline Simulation Resource Usage

Strategy	Simulations	CPU Time per	Storage per
Baseline	75	16 hours	1.7GB
Constant	675	1 hour	413MB
Random	675	16 hours	1.7GB
Reactive	675	16 hours	1.5GB
Random Reactive	675	16 hours	1.5GB

housed at the University of British Columbia and consists of 384 nodes tuned for parallel jobs. Each node consists of eight 3.0GHz Intel Xeon E5450 processors, 16GB physical memory, and a shared 6.4TB file system. The statistical analysis for the small and large baselines was performed using 5 and 10 processors respectively. Table 5.3 and Table 5.4 give the per-experiment CPU time and storage[1] requirements. The results of the statistical analysis for each experiment are on average 20MB.

Table 5.3: Small Baseline Experiment Resource Usage

Strategy	Experiments	CPU Time per	Storage per
Baseline	1	5 hour	4GB
Constant	9	1 hour	340MB
Random	9	5 hour	5GB
Reactive	9	5 hour	3GB
Random Reactive	9	5 hour	4GB

[1]The storage used by the statistical analysis is in addition to the simulation storage needs; however, it is temporary data used to promote scalability by reducing duplicated calculations and provide recovery in case of failure.

Table 5.4: Large Baseline Experiment Resource Usage

Strategy	**Experiments**	**CPU Time per**	**Storage per**
Baseline	1	60 hours	65GB
Constant	9	10 hours	1.5GB
Random	9	60 hours	64GB
Reactive	9	60 hours	61GB
Random Reactive	9	60 hours	62GB

In total, over 4 CPU years of processing and nearly 6TB of storage were required to produce the results. The majority of CPU time and storage needs were the result of the simulations performed, primarily those conducted using the large baseline. In order to perform the needed simulations in a time-efficient manner the ability to run many concurrently is required. This increases the storage resource requirements and is the motivating factor in using the Hermes cluster due to the limited storage space on Orcinus[2].

The scalability of the statistical analysis is limited to 64 processes by the number of MATLAB distributed computing engine (MDCE) licenses available on Orcinus. Hence, the maximum theoretical speedup is 64. The efficiency of the implementation, (i.e., the speedup gained through parallelization divided by the number of processors used) depends on a range of factors. Within the MDCE an experiment using P processors will have a maximum efficiency of $1 - \frac{1}{P}$ because one processor is reserved as a manager and remains idle for the majority of the statistical analysis[3]. This promotes the use of more processors to improve efficiency; however, the benefit of increasing the number of processors is limited by the number of simulations performed as part of the experiment, (i.e., records), and the number of features measured. The statistical analysis first distributes the detection of steady-state behaviour for each feature record and then distributes the mode detection for each feature. As such, when conducting the statistical analysis the number of processors used is chosen for speedup rather than efficiency.

[2]The Orcinus cluster received a substantial hardware upgrade in both the number of processors and storage capacity shortly after performing the experiments needed for this work.

[3]This is the result of using the *parfor* feature of the MDCE. While the *pmode* feature allows for more efficiency it is not well suited to distributed calculations which vary in runtime because the processors operate in lock-step.

5.2 Baseline MANET Operation

The small baseline MANET was observed to have between 36 and 69 records exhibit stationarity for each of the six primary features. The least stationary feature was X^{NRO} while the most was X^{RA}. The non-stationary records were found to be entirely transient in all but one case where there was insufficient data to test, again for the X^{NRO} feature. Of the stationary records several were non-ergodic. All six features exhibited more than 10 modes of behaviour. Just four of the primary features were seen to have a largest mode of more than 10 records, these being: X^{RA}, X^{PDR}, X^{DD}, and X^{MEU}.

The features of the large baseline MANET were observed to have more records be non-stationary than those of the small baseline with between 27 and 60 records exhibiting steady-state behaviour. All non-stationary records were transient in nature, while a handful of those exhibiting stationarity were non-ergodic. Just two features, X^{DD} and X^{MEU}, were observed to have a largest ergodic mode comprised of more than 10 records.

The four figures below show the averaged estimated empirical distributions of the features with 10 or more records in their largest mode. Full details are available from Table B.2.

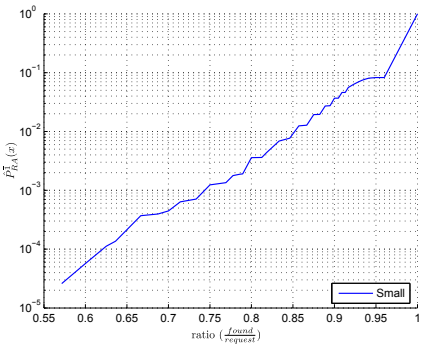


Figure 5.1: Baseline X_{RA}

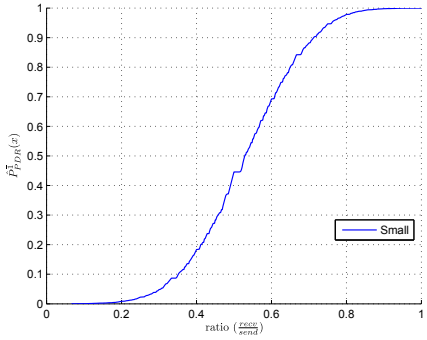


Figure 5.2: Baseline X^{PDR}

From Figure 5.3 it can be observed that both the small and large baseline MANETs behave as expected. That is to say, the large MANET has a longer delivery delay, on average, than that of the small MANET; however, the minimum, median, and maximum delays are similar. Furthermore, it can be seen that in both cases the form of the averaged estimated empirical distributions are similar.

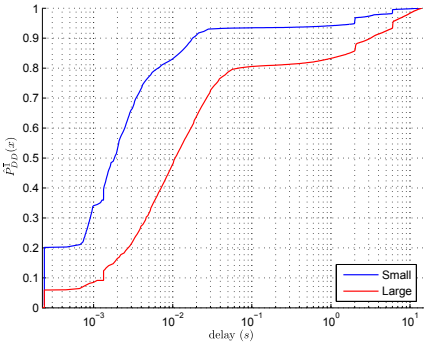


Figure 5.3: Baseline X^{DD}

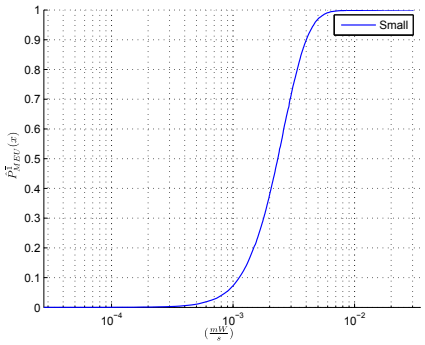


Figure 5.4: Baseline X^{MEU}

5.3 Constant Jamming

The features of the small MANET were observed to become almost completely non-stationary across all three scenarios under the constant jamming strategy. More specifically, only the X^{DD}, X^{HT}, and X^{MEU} features produced enough data to be tested for stationarity. The 50m jamming range experiment was the only one in which the X^{DD} and X^{HT} features were observed to have an ergodic mode comprised of 10 or more records.

The large MANET was observed to produce more stationary records than the small MANET for the six features considered under the constant jamming strategy. The jamming range scenario was observed to have more than 10 records in the largest mode for the X^{DD} and X^{HT} features. While the motion scenario was seen to have mostly transient features for the six considered and no modes larger than 10 records. The multiple attacker scenario was observed to have more than 10 records in the largest ergodic mode of the X^{HT} feature for the 2, 3 and 4 attacker experiments.

The three figures below show the averaged estimated empirical distributions detected for the features having largest modes of 10 or more records. The full results for the six features are available in Section B.2 for all three scenarios performed against the small and large baseline MANETs.

Figure 5.5 illustrates the impact of the 50m jamming range experiment relative to the small baseline MANET's X^{DD} feature. It can be observed that the minimum is seen to increase while the mean, median, and maximum delay decrease relative to the small baseline MANET. However, the significance of the impact is limited due to it being based on only 10 records.

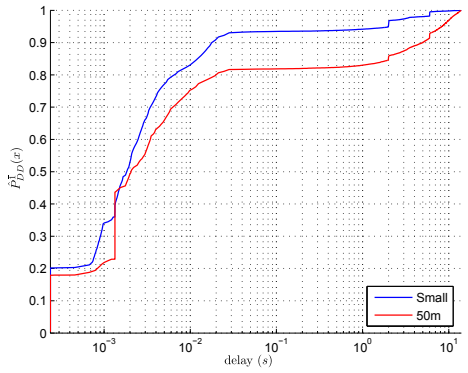


Figure 5.5: Constant vs Small X^{DD}

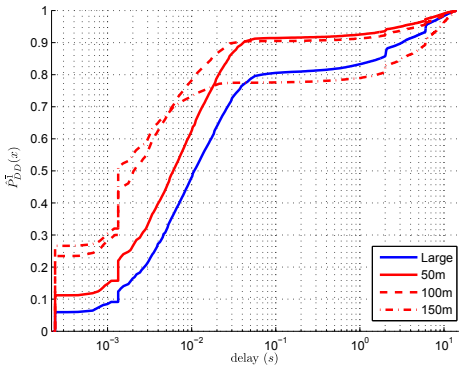
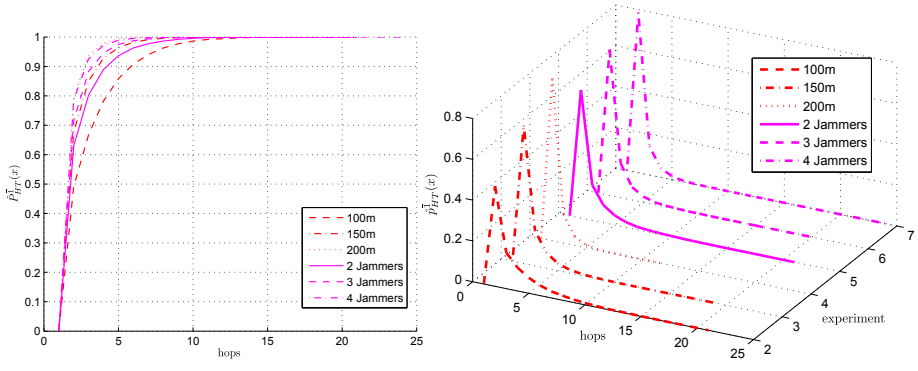


Figure 5.6: Constant vs Large X^{DD}

The impact of the constant jamming power scenario on the X^{DD} feature of the large baseline MANET for 50m, 100m, and 200m is shown in Figure 5.6. Under the 50m and 200m jamming range experiments the minimum, mean, median, and maximum delay decreased relative to the large baseline MANET. While the 100m experiment saw the minimum increase and the mean, median, and maximum decrease. Overall, the mean delay for the jamming range scenario experiments were observed to decrease relative to the range while the median increased. This behaviour is contrary to the impact observed on the small baseline. To understand this, the feature X^{HT} shows that the number of hops taken by delivered packets decreases with jamming range.



(a) CDF (b) PDF

Figure 5.7: Constant vs Large X^{HT}

Figure 5.7 shows the X^{HT} feature measured from the large MANET for both the jamming range and multiple attacker scenarios under the constant jamming strategy. The six experiments are seen to exhibit the same minimum value for the X^{HT} feature.

In general the mean, median, and maximum value decreases when range or number of attackers is increased. The lowest impact is caused by the 100m jamming range followed by 2 attackers, 150m range, 3 attackers, 4 attackers, and finally the largest impact is from the 200m jamming range. Both the 200m jamming range and 4 attacker experiments were seen to have a largest ergodic mode containing 40 or more records making them reasonable estimates.

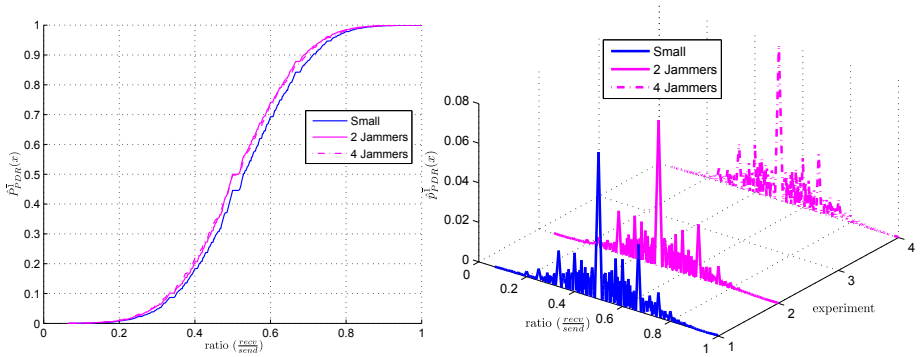
5.4 Random Jamming

The small MANET under the three random jamming scenarios is observed to exhibit slightly higher numbers of stationary records than the baseline for each of the six primary features. That being said, the number of records which did not contain enough data to be tested for stationarity increased slightly with the majority being related to the X^{NRO} feature. Furthermore, the number of non-ergodic features also increased for all three scenarios. The number of records in the largest ergodic mode for each of the features was also seen to increase relative to the baseline. Only the X^{MEU} feature for the jamming range scenario experiments failed to contain 10 or more records in the largest ergodic mode.

The large MANET was observed to exhibit similar numbers of stationary records under the random jamming strategy as the baseline for the six features considered. The non-stationary records were found to be transient in nature in all but one case, where too little data was present to test. As with the small MANET the number of non-ergodic records was higher relative to the baseline under the random jamming strategy. Additionally, the number of records in the largest ergodic mode for each feature was higher as well. The X^{DD} and X^{HT} features were seen to have more than 10 records in the majority of experiments.

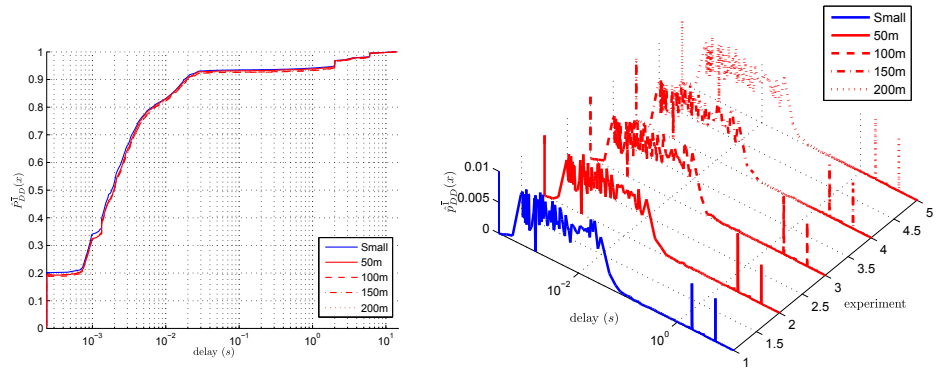
The three figures below show the averaged estimated empirical distributions detected for the X^{PDR} and X^{DD} features which can be compared relative to the baseline MANETs. Full results for the random jamming strategy experiments are in Section B.3 for all three scenarios.

Figure 5.8 shows the largest impact observed for the random jamming strategy on the small baseline MANET. The minimum, mean, and median X^{PDR} was observed to decrease under the 2 and 4 attacker experiments relative to the baseline. However, their median impact relative to one another was less pronounced. The 2, 3, and 4 attacker experiments had a largest mode of 17, 17, and 21 records respectively.



(a) CDF (b) PDF

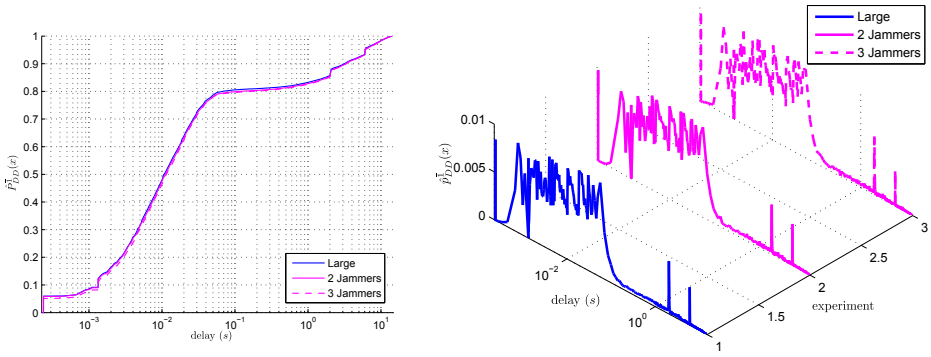
Figure 5.8: Random vs Small X^{PDR}



(a) CDF (b) PDF

Figure 5.9: Random vs Small X^{DD}

Figure 5.9 and Figure 5.10 show the impact of the random jamming strategy on the small and large baseline X^{DD} features respectively. From this it can be observed that in both cases the mean increased slightly relative to the baseline, while the median decreased marginally. Overall, due to the low number of records in the largest modes, between 10 and 13, the results are not significantly different from the baseline.



(a) CDF (b) PDF

Figure 5.10: Random vs Large X^{DD}

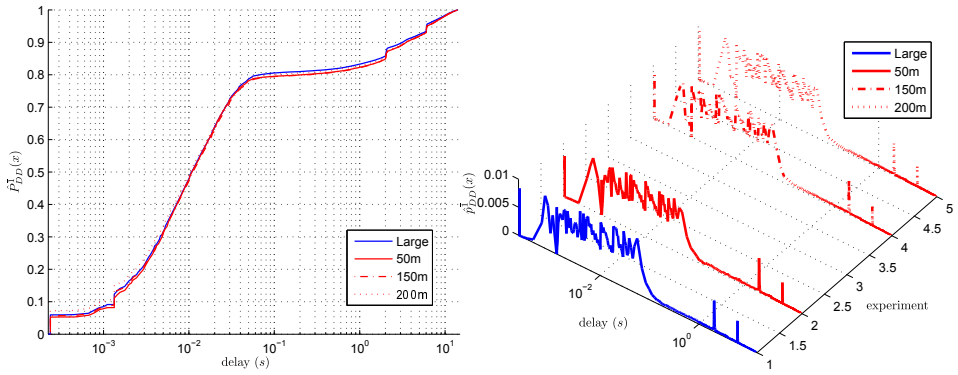
5.5 Reactive Jamming

The small MANET under the three reactive jamming scenarios was observed to exhibit only a small number of stationary records. The X^{RA}, X^{PDR}, and X^{NRO} features showed the most number of records with insufficient data to test. From the 9 experiments considered in only two instances was a feature observed to have 22 stationary records, the largest number. Of the records which were stationary half, if not all, were seen to be non-ergodic. Overall, no experiment produced a feature with 10 or more records in the largest ergodic mode.

The large MANET was observed to exhibit a slightly lower number of stationary records in comparison to the baseline under the three reactive jamming scenarios. In all cases the records found to be non-stationary had sufficient data to be tested but were transient in nature. The number of stationary records which were found to be non-ergodic slightly increased in comparison to the baseline. Under the jamming range scenario only the X^{DD} and X^{HT} features were observed to have 10 or more records in their largest ergodic mode. For the motion scenario the X^{HT} feature was the only one to have at least 10 records in the largest ergodic mode. While the multiple attacker scenario produced no features with largest ergodic mode of sufficient size.

The two figures below show the averaged estimated empirical distributions for the X^{DD} and X^{HT} features. Full results for the six features across all three scenarios is available from Section B.4.

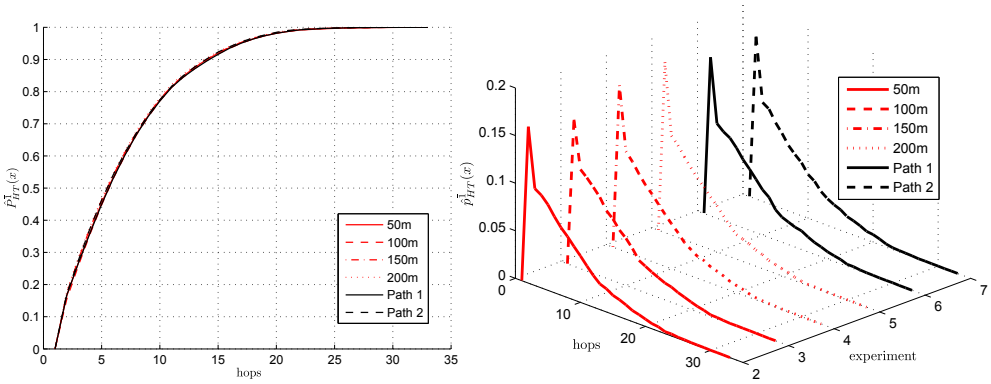
Figure 5.11 shows the reactive jamming impact on the X^{DD} feature of the large baseline for the three jamming range scenario experiments containing sufficient data. For the 50, 150, and 200m jamming range experiments the minimum delay was ob-



(a) CDF (b) PDF

Figure 5.11: Reactive vs Large X^{DD}

served to decrease slightly. The mean delay was observed to increase in the first two instances and decrease in the third, with the median behaving inversely. The maximum delay was observed to decrease for the 50m and 200m experiments, and increase for the 150m range. Overall, the differences were minimal and due to the low number of records in the ergodic modes reported the behaviour is not significantly different from the baseline.



(a) CDF (b) PDF

Figure 5.12: Reactive vs Large X^{HT}

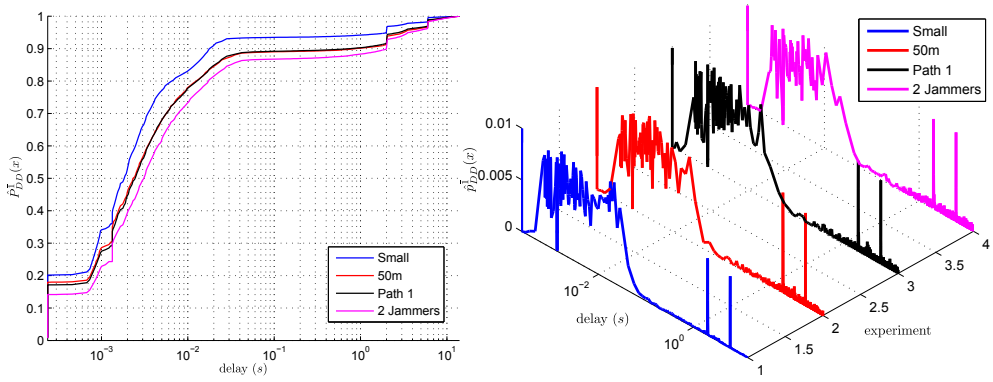
The effect of varying jamming range and the presence of mobility on the X^{HT} feature of the large MANET is shown in Figure 5.12. In all cases the minimum, median, and maximum hops taken did not change. The mean hops taken were seen to increase as jamming range changed from 50m to 100m, and then decrease for 150m and 200m experiments. The two paths of motion chosen were seen to result in a mean hops taken slightly lower than the 100m and 150m jamming range experiments

respectively. Overall, there was no significant impact on the X^{HT} feature under the power and motion scenarios for reactive jamming.

5.6 Random Reactive Jamming

The small MANET under the influence of the three random reactive jamming scenarios is seen to exhibit slightly fewer stationary records per feature than the baseline. The majority of non-stationary records are due to transient behaviour while a small number, primarily in the multiple attacker scenario, have insufficient data available to test. In general, there are more non-ergodic records when compared to the baseline. The jamming range scenario is observed produce largest ergodic modes of 10 or more records for all but X^{MEU} for at least one of the four experiments.

The large MANET was observed to exhibit slightly higher numbers of stationary records under the influence of the three random reactive jamming scenarios in comparison to the baseline. In general, the non-stationary records were found to be transient, and none lacked sufficient data to test. The number of non-ergodic records was seen to be both above and below that of the baseline. Overall, the number of features which exhibited 10 or more records in their largest ergodic mode was greater than the baseline, with X^{DD} and X^{HT} being the most common.

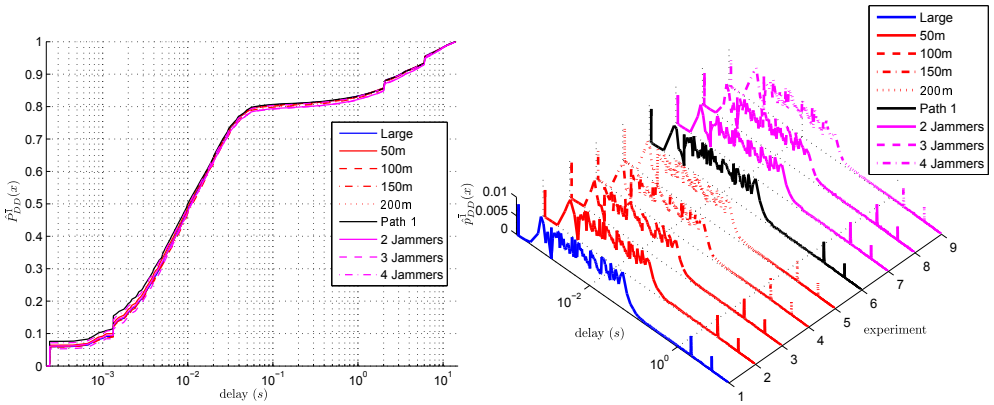


(a) CDF (b) PDF

Figure 5.13: Random Reactive vs Small X^{DD}

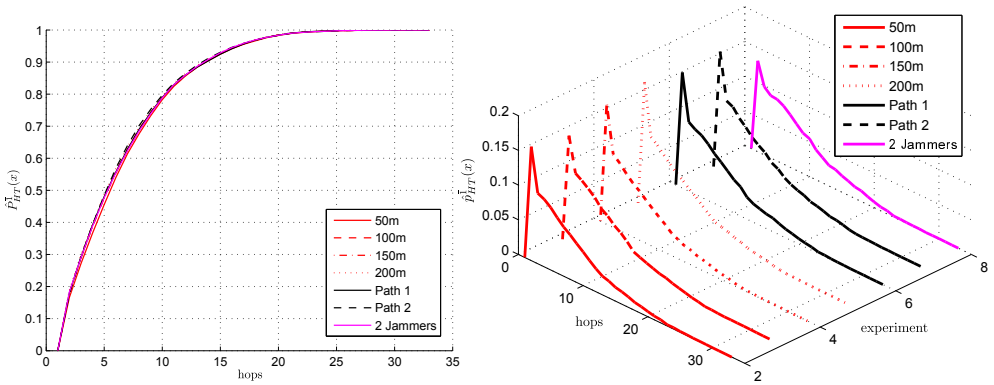
Figure 5.13 shows the impact of the three scenarios on the X^{DD} feature of the small MANET under random reactive jamming. One experiment from each of the three scenarios were observed to have a largest ergodic mode of 10 or more records.

In general, it can be seen that increasing the number of attackers caused a larger increase in the delay than a reduced jamming range or having a moving attack host.



(a) CDF (b) PDF

Figure 5.14: Random Reactive vs Large X^{DD}



(a) CDF (b) PDF

Figure 5.15: Random Reactive vs Large X^{HT}

Figure 5.14 and Figure 5.15 show the observed behaviour of the X^{DD} and X^{HT} features respectively for the large MANET under the random reactive jamming strategy. Overall, there is minimal impact across the two features with X^{DD} showing the greatest change. Due to the low number of records, no more than 17, the differences observed are not significant.

5.7 Summary of Strategy Evaluation

This chapter presented an evaluation of the results produced by the statistical analysis of a variety of MANET jamming experiments. Of the six primary feature considered delivery delay and hops taken were observed to be the most ergodic, having their largest modes comprised of 10 or more records in almost all experiments. The least ergodic features were observed to be the three composite features, namely: route availability, packet delivery ratio, and normalized routing overhead. In general, the impact of the four jamming strategies was to reduce the level of stationarity or increase the amount of non-ergodic records. Furthermore, a number of reported modes did not differ significantly from the baseline MANETs which further re-enforces the notion that physical layer jamming pushes MANETs into non-stationary and non-ergodic modes of behaviour.

The largest disruption to MANET operations observed was from the constant jamming strategy while the least was from the random jamming strategy. The reactive and random reactive strategies appear to have a different impact on the baseline MANETs, when compared to the other two. That is, the level of non-stationarity and non-ergodicity in comparison to the baselines increased. From this it can be concluded that while the constant jamming strategy is the most disruptive to MANET operations is it may undesirable due to the amount of energy required which increases the attack host's visibility. The random jamming strategy, while using less energy, was not seen as effective due to its inability to target MANET activity. The reactive jamming strategy was able to reduce energy usage while maintaining an observable impact by timing the attack host's actions to the activity of the MANET. The random reactive jamming strategy was observed to offer even lower energy usage at the cost of a reduction in impact on MANET operations. As such, the reactive and random reactive jamming strategies can be considered the most effective when minimizing the visibility of the attack.

The increase in jamming induced non-stationary behaviours is interesting as it suggests that jamming could be used to push the MANET into non-stationary behaviours. This implies that to be resilient against physical-layer jamming attacks adaptive control strategies for the various MANET features of operational importance should be considered in the engineering process. This is counter to the current design methods applied to MANET routing protocols which focus primarily on improving performance characteristics of the network.

Chapter 6

Conclusions

With the growing number of portable computing devices, (e.g., smart phones and tablets), MANET based communications networks are becoming more of a reality. As such, in order to deploy and provide MANET based services the ability to engineering them to meet desired properties is critical. In particular, how issues such as the reliability, availability, and performance of MANETs must be understood. Moreover, it is important to understand how the actions of malicious users impact MANET operations so that services delivered can be engineered to be resilient against inevitable attack. Physical-layer jamming is one such attack that can be easily performed by malicious users to impact the MANET with a minimum of knowledge. Therefore, understanding how MANETs of different scale are impacted by physical-layer jamming is important to engineering robust solutions.

The assessment of MANET operations under the different physical-layer jamming strategies is accomplished through the creation of a discrete-time event-based simulation. More specifically, the OMNeT++ simulator engine and INETMANET module library are combined with custom modules to provide a highly detailed and configurable model within which variations of the MANET jamming problem are modelled. The MANET is formed by a collection of autonomous terminal hosts comprised of application, transport, network, routing, link, physical, and mobility components. Attack hosts capable of performing constant, random, reactive, and random reactive jamming strategies are introduced into the simulation environment to model different physical-layer attacks on the MANET. The implemented MANET jamming simulation is instrumented to collect measurements relating to the most commonly reported features of operation, namely: route availability, packet delivery ratio, normalized routing overhead, delivery delay, hops taken, and MANET energy usage.

The statistical analysis presented in this work is successful in providing quantitative data about the behaviour of these commonly measured features. Moreover, the results of the statistical analysis provide empirical evidence as to the importance of addressing the stated critical issues present when using less formal methods, specifically: ad-hoc removal of start-up transients, and averaging over non-ergodic Monte-Carlo runs. These issues are addressed through the formal application of stationarity and ergodicity testing of the measured features of the MANET jamming simulation. Prior to conducting the experiments related to the MANET jamming problem the implementation of the statistical analysis is validated. This is done using randomly generated data which are observed to be statistically similar to their expected form within a 95% confidence interval, as measured by the Chi-square goodness of fit test.

The results of the statistical analysis applied to the MANET jamming experiments show that a range of behaviours are present in the features measured. In particular, the increase in non-stationary and non-ergodic behaviours suggests that the design of jamming resistant MANET protocols presents additional complexities which are not well studied. Hence, rigorous statistical testing can be seen as a requirement in the engineering process as it exposes the various sets of behaviours that jamming induces in the MANET. Furthermore, is is concluded that the use of informal methods is likely to produce misleading information about the behaviour of the features measured and result in sub-optimal design decisions when engineering MANETs to be resilient against physical-layer jamming.

6.1 Future Work

The complexity of behaviour exposed through the statistical analysis presented in this work highlights several avenues of future work. The first of these is suggested by the relatively small size of the ergodic modes detected for each of the features measured, (i.e., many of the experiments conducted exhibited ergodic modes with fewer than 10 records). Thus, it is desirable to examine the behaviour of MANET operations using more than 75 records per experiment, potentially in the range of 500 or more. The second avenue for future work is suggested by the presence of multiple modes of behaviour which presents considerable complexity when drawing comparisons between different experiments. As such, a formalized method of comparison between experiments that takes into account all modes of sufficient size will assist in better understanding the behaviour of MANETs. The final avenue of future work pertains

to the statistical analysis performed. More specifically, the features measured in this work are the aggregate of multiple sources. Thus, the ability to assess the behaviour of these individual sources can provide insight into the transient behaviour of many of the features measured.

Bibliography

[1] Ieee standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements part ii: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11g-2003 (Amendment to IEEE Std 802.11, 1999 Edn. (Reaff 2003) as amended by IEEE Stds 802.11a-1999, 802.11b-1999, 802.11b-1999/Cor 1-2001, and 802.11d-2001)*, pages i –67, 2003.

[2] A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1-3):564 – 568, 2008.

[3] M. Abdelhafez, G. Riley, R.G. Cole, and Nam Phamdo. Modeling and simulations of tcp manet worms. In *Principles of Advanced and Distributed Simulation, 2007. PADS '07. 21st International Workshop on*, pages 123 –130, june 2007.

[4] L.A. Al-Sulaiman and H. Abdel-Wahab. Cacman: a framework for efficient and highly available ca services in manets. In *Computers and Communications, 2005. ISCC 2005. Proceedings. 10th IEEE Symposium on*, pages 10 – 15, june 2005.

[5] Thomas Dead James Allen and Yiannis Aloimonos. *Artificial Intelligence Theory and Practice*. Addison-Wesley Publishing Company, Menlo Park, CA, USA, 1995.

[6] T.R. Andel and A. Yasinsac. On the credibility of manet simulations. *Computer*, 39(7):48 –54, july 2006.

[7] Ahmed Ayadi, Patrick Maille, and David Ros. Improving distributed tcp caching for wireless sensor networks. In *Ad Hoc Networking Workshop (Med-Hoc-Net), 2010 The 9th IFIP Annual Mediterranean*, pages 1 –6, june 2010.

[8] J.S. Bendat and A. G. Piersol. Random data: Analysis and measurement procedures, 3rd edition. Number 3. Wiley-Interscience, 2000.

[9] Suzanne Fox Buchele. Using olpc laptop technology as a computer science case study teaching tool. *J. Comput. Small Coll.*, 24:130–136, April 2009.

[10] Compute Canada. List of available computing resources at compute canada. [Online] `https://computecanada.org/page/4/EN`, Oct 2011. Last Accessed: Oct 2011.

[11] David Cavin, Yoav Sasson, and André Schiper. On the accuracy of manet simulators. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, POMC '02, pages 38–43, New York, NY, USA, 2002. ACM.

[12] I. Chakeres, CenGen, C. Perkins, and Wichorus. Dynamic manet on-demand (dymo) routing. [Online] `http://tools.ietf.org/html/draft-ietf-manet-dymo-21`, January 2011. Last Accessed: Oct 2011.

[13] T. Clausen Ed., P. Jacquet Ed., and Project Hipercom INRIA. Dynamic manet on-demand (dymo) routing. [Online] `http://www.ietf.org/rfc/rfc3626.txt`, October 2003. Last Accessed: Oct 2011.

[14] T. Dreibholz and E.P. Rathgeb. Rserpool - providing highly available services using unreliable servers. In *Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on*, pages 396 – 403, aug.-3 sept. 2005.

[15] Thomas Dreibholz, Xing Zhou, and Erwin Paul Rathgeb. A powerful tool-chain for setup, distributed processing, analysis and debugging of omnet++ simulations. In *Proceedings of the 1st ACM/ICST International Workshop on OMNeT++*, 1st ACM/ICST, Marseille/France, 2008. ACM.

[16] H. ElArag and M. Bassiouni. Simulation of transport protocols over wireless communication networks. In *Simulation Conference Proceedings, 2000. Winter*, volume 2, pages 1235 –1241 vol.2, 2000.

[17] Zhe Fan, Feng Qiu, Arie Kaufman, and Suzanne Yoakum-Stover. Gpu cluster for high performance computing. In *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, SC '04, pages 47–, Washington, DC, USA, 2004. IEEE Computer Society.

[18] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett,

Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[19] K. Ghaboosi, M. Latva-aho, and Yang Xiao. Receiver blocking problem in mobile ad hoc networks: Challenges and solutions. In *Wireless Days, 2008. WD '08. 1st IFIP*, pages 1 –5, nov. 2008.

[20] J.D. Gibbs and H.S. Sarjoughian. Assessing the impact of a modeling tool and its support for verification and validation. In *Performance Evaluation of Computer Telecommunication Systems, 2009. SPECTS 2009. International Symposium on*, volume 41, pages 73 –80, july 2009.

[21] Jason J. Haas and Yih-Chun Hu. Communication requirements for crash avoidance. In *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*, VANET '10, pages 1–10, New York, NY, USA, 2010. ACM.

[22] J.M.M. Kamal, M.S. Hasan, A.L. Carrington, and Hongnian Yu. Lessons learned from real manet experiments and simulation-based evaluation of udp and tcp. In *Computer and Information Technology (ICCIT), 2010 13th International Conference on*, pages 511 –515, dec. 2010.

[23] Dongkyun Kim, J.J. Garcia-Luna-Aceves, K. Obraczka, J.-C. Cano, and P. Manzoni. Routing mechanisms for mobile ad hoc networks based on the energy drain rate. *Mobile Computing, IEEE Transactions on*, 2(2):161 – 173, april-june 2003.

[24] G. Koltsidas, S. Karapantazis, G. Theodoridis, and F.-N. Pavlidou. A detailed study of dynamic manet on-demand multipath routing for mobile ad hoc networks. In *Wireless and Optical Communications Networks, 2007. WOCN '07. IFIP International Conference on*, pages 1 –5, july 2007.

[25] M. Korkalainen, M. Sallinen, N. Karkkainen, and P. Tukeva. Survey of wireless sensor networks simulation tools for demanding applications. In *Networking and Services, 2009. ICNS '09. Fifth International Conference on*, pages 102 –106, april 2009.

[26] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. Manet simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9:50–61, October 2005.

[27] Python Programming Language. Python v2.7.2 documentation. [Online] `http://docs.python.org/release/2.7.2/`, Oct 2011. Last Accessed: Oct 2011.

[28] J. Lei, R. Yates, L. Greenstein, and Hang Liu. Mapping link snrs of real-world wireless networks onto an indoor testbed. *Wireless Communications, IEEE Transactions on*, 8(1):157 –165, jan. 2009.

[29] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, and Thomas Sandholm. What's inside the cloud? an architectural map of the cloud landscape. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, CLOUD '09, pages 23–31, Washington, DC, USA, 2009. IEEE Computer Society.

[30] Yao-Nan Lien and Yi-Fan Yu. Hop-by-hop tcp over manet. In *Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE*, pages 1150 –1155, dec. 2008.

[31] MathWorks. Matlab distributed computing server. [Online] `http://www.mathworks.com/help/toolbox/mdce/`, Oct 2011. Last Accessed: Oct 2011.

[32] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8:3–30, January 1998.

[33] Michael McGuire. Stationary distributions of random walk mobility models for wireless ad hoc networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '05, pages 90–98, Urbana-Champaign, IL, USA, 2005. ACM.

[34] E. Millman, D. Arora, and S.W. Neville. Stars: A framework for statistically rigorous simulation-based network research. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 733 –739, march 2011.

[35] H. Moustafa, H. Labiod, and P. Godlewski. A reactive random graph (rrg) model for multicast routing in manets. In *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, volume 5, pages 5 pp. –2724, dec. 2005.

[36] A. Munjal, T. Camp, and W.C. Navidi. Constructing rigorous manet simulation scenarios with realistic mobility. In *Wireless Conference (EW), 2010 European*, pages 817 –824, april 2010.

[37] K. Patel, J. DeDourek, and P. Pochec. Investigation of channel formation in a manet. In *Systems and Networks Communications (ICSNC), 2010 Fifth International Conference on*, pages 229 –231, aug. 2010.

[38] K. Pawlikowski, V.W.C. Yau, and D. McNickle. Distributed stochastic discrete-event simulation in parallel time streams. In *Simulation Conference Proceedings, 1994. Winter*, pages 723 – 730, dec. 1994.

[39] C. Perkins, Nokia Research Center, E. Belding-Royer, University of California Santa Barbara, S. Das, and University of Cincinnati. Ad hoc on-demand distance vector (aodv) routing. [Online] `http://tools.ietf.org/html/rfc3561`, July 2003. Last Accessed: Oct 2011.

[40] J. Postel. Dod standard internet protocol. [Online] `http://www.ietf.org/rfc/rfc760.txt`, jan 1980. Obsoleted by RFC 791, updated by RFC 777, Last Accessed: Oct 2011.

[41] J. Postel. DoD standard Transmission Control Protocol. [Online] `http://www.ietf.org/rfc/rfc761.txt`, January 1980. Last Accessed: Oct 2011.

[42] J. Postel. User Datagram Protocol. [Online] `http://www.ietf.org/rfc/rfc768.txt`, August 1980. Last Accessed: Oct 2011.

[43] J. Postel. Internet control message protocol. [Online] `http://www.ietf.org/rfc/rfc792.txt`, September 1981.

[44] M.A. Rahman, M.S. Azad, F. Anwar, and A.H. Abdalla. A simulation based performance analysis of reactive routing protocols in wireless mesh networks. In *Future Networks, 2009 International Conference on*, pages 268 –272, March 2009.

[45] Michael Schwind, Oliver Hinz, and Roman Beck. A cost-based multi-unit resource auction for service-oriented grid computing. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, GRID '07, pages 137–144, Washington, DC, USA, 2007. IEEE Computer Society.

[46] Christoph Sommer, Isabel Dietrich, and Falko Dressler. A Simulation Model of DYMO for Ad Hoc Routing in OMNeT++. In *1st ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008): 1st ACM/ICST International Workshop on OMNeT++ (OMNeT++ 2008)*, Marseille, France, March 2008. ACM.

[47] Christoph Sommer, Isabel Dietrich, and Falko Dressler. Simulation of ad hoc routing protocols using omnet++. *Mob. Netw. Appl.*, 15:786–801, December 2010.

[48] A. Suresh and K. Duraiswamy. Scalable instant way point routing protocol for manet. In *Computing Communication and Networking Technologies (ICCCNT), 2010 International Conference on*, pages 1 –8, july 2010.

[49] M. Varvello, I. Rimac, U. Lee, L. Greenwald, and V. Hilt. On the design of content-centric manets. In *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, pages 1 –8, jan. 2011.

[50] W. Willinger and V. Paxson. Where mathematics meets the internet. *Notices of the American Mathematical Society*, 45(8):961–970, August 1998.

[51] Xiaodong Xian, Weiren Shi, and He Huang. Comparison of omnet++ and other simulator for wsn simulation. In *Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on*, pages 1439 –1443, june 2008.

[52] Wenyuan Xu, Ke Ma, W. Trappe, and Yanyong Zhang. Jamming sensor networks: attack and defense strategies. *Network, IEEE*, 20(3):41 – 47, may-june 2006.

[53] Fan Ya-qin, Fan Wen-yong, and Wang Lin-zhu. Opnet-based network of manet routing protocols dsr computer simulation. In *Information Engineering (ICIE), 2010 WASE International Conference on*, volume 4, pages 46 –49, aug. 2010.

[54] Lu Ying, Kang Feng-ju, Zhong Lian-jiong, and Wang Zhi-Guang. A self-similar traffic generation method and application in the simulation of mobile ad-hoc network. In *Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS International Colloquium on*, volume 4, pages 229 –233, aug. 2009.

[55] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1312 – 1321 vol.2, march-3 april 2003.

[56] Fang Zhe and Wang Wan-liang. Research on unidirectional link routing algorithm for mobile ad hoc networks. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 5, pages V5–287 – V5–290, aug. 2010.

Appendix A

OMNeT++ Simulator Model

This appendix contains a listing of all major simulation modules and experiment configuration files used in this work.

A.1 Network Definition

```
package network;

import inet.world.ChannelControl;
import inet.networklayer.autorouting.FlatNetworkConfigurator;
import manet.nodes.Terminal;
import manet.nodes.Attacker;
import manet.world.TerminalControl;
import manet.instrument.Instruments;

network manet
{
    parameters:
        int numHosts = default(360);
        int numAtkrs = default(0);
        double playgroundSizeX = default(1500);
        double playgroundSizeY = default(1000);
    submodules:
        host[numHosts]: Terminal {
            parameters:
                id = index;
                @display("i=device/pocketpc_vs;r=,,#707070;p=70,117");
        }
        atkr[numAtkrs]: Attacker {
                parameters:
                        id = index + numHosts;
                        @display("i=device/palm2_vs;r=,,#707070;p=70,117");
        }
        channelcontrol: ChannelControl {
            parameters:
                playgroundSizeX = playgroundSizeX;
                playgroundSizeY = playgroundSizeY;
                @display("p=31,31;i=misc/sun");
        }
        terminalcontrol: TerminalControl {
            @display("p=31,31;i=misc/sun");
        }
        configurator: FlatNetworkConfigurator {
            parameters:
                networkAddress = "145.236.0.0";
                netmask = "255.255.0.0";
                @display("p=77,31;i=block/cogwheel_s");
        }
        logger: Instruments {
            @display("p=177,31");
        }
    connections allowunconnected:
}
```

A.1.1 Terminal

```
package manet.nodes;

import inet.applications.udpapp.UDPApp;
import inet.base.NotificationBoard;
import inet.mobility.BasicMobility;
import manet.linklayer.ieee80211.TerminalIeee80211NicAdhoc;
import inet.networklayer.common.InterfaceTable;
import inet.networklayer.ipv4.RoutingTable;
import manet.nodes.inet.NetworkLayerGlobalArp_instrumented;
import inet.nodes.inet.NetworkLayerGlobalArp;
import inet.transport.udp.UDP;
import inet.networklayer.manetrouting.ManetRouting;

module Terminal
{
    parameters:
        @node();
        int id = default(-1);
        int numUdpApps = default(0);
        string udpAppType = default("OnOffApp");
        string routingFile = default("");
        string mobilityType = default("TerminalMobility");
        @display("i=device/pocketpc_s");
    gates:
        input radioIn @directIn;
    submodules:
        notificationBoard: NotificationBoard {
            parameters:
                @display("p=60,70");
        }
        interfaceTable: InterfaceTable {
            parameters:
                @display("p=60,150");
        }
        routingTable: RoutingTable {
            parameters:
                IPForward = true;
                routerId = "";
                routingFile = routingFile;
                @display("p=60,230");
        }
        udpApp[numUdpApps]: <> like UDPApp {
            parameters:
                id = id;
                @display("p=272,67");
        }
        udp: UDP {
            parameters:
                @display("p=272,154");
        }
        networkLayer: NetworkLayerGlobalArp_instrumented {
            parameters:
                id = id;
                proxyARP = false;
                globalARP = true;
                @display("p=248,247;q=queue");
            gates:
                ifIn[1];
                ifOut[1];
        }
        manetrouting: ManetRouting{
            parameters:
            @display ("p=153,247");
        }
        wlan: TerminalIeee80211NicAdhoc {
            parameters:
                id = id;
                @display("p=248,349;q=queue");
        }
        mobility: <mobilityType> like BasicMobility {
            parameters:
                id = id;
                @display("p=149,307");
        }
    connections allowunconnected:
        for i=0..numUdpApps-1 {
            udpApp[i].udpOut --> udp.appIn++;
            udpApp[i].udpIn <-- udp.appOut++;
        }
        udp.ipOut --> networkLayer.udpIn;
        udp.ipIn <-- networkLayer.udpOut;
        networkLayer.MANETOut --> manetrouting.from_ip;
        networkLayer.MANETIn <-- manetrouting.to_ip;
        radioIn --> wlan.radioIn;
        wlan.uppergateOut --> networkLayer.ifIn[0];
        wlan.uppergateIn <-- networkLayer.ifOut[0];
}
```

A.1.2 TerminalMobility

```
package manet.mobility;

simple TerminalMobility like inet.mobility.BasicMobility
{
    parameters:
        bool debug = default(false);
        int id = default(-1);
        int rng = default(1);
        double x @unit("m") = default(-1m);
        double y @unit("m") = default(-1m);
        double w @unit("m") = default(15m);
        double randomChoice = default(0.01);
        double updateInterval @unit("s") = default(0.1s);
        double startTime @unit("s") = default(0s);
        bool registerControl = default(true);
        volatile double speed = default(5);
        volatile double waitTime @unit("s") = default(0s);
        @display("i=block/cogwheel_s");
}
```

A.1.3 TerminalControl

```
package manet.world;

simple TerminalControl
{
    parameters:
        int rng = default(3);
        double connectionDistance @unit("m") = default(30m);
        double randomChoice = default(0.05);
        double positionInterval @unit("s") = default(20s);
        double startTime @unit("s") = default(0s);
        @display("i=misc/sun");
        bool trackManet = default(true);
        bool findDestination = default(true);
}
```

A.1.4 OnOffApp

```
package manet.applications.udpapp;

simple OnOffApp like inet.applications.udpapp.UDPApp
{
    parameters:
        int id = default(-1);
        int rng = default(2);
        double startTime @unit("s") = default(-1s);
        double stopTime @unit("s") = default(0s);
        int localPort = default(1234);
        int destPort = default(1234);
        volatile double requestOn @unit("s") = default(0s);
        volatile double requestOff @unit("s") = default(0s);
        volatile int requestBytes @unit("Byte") = default(1KB);
        int segmentBytes @unit("Byte") = default(1472B);
        bool enableTransport = default(true);
    gates:
        input udpIn;
        output udpOut;
}
```

A.1.5 TerminalIeee80211NicAdhoc

```
package manet.linklayer.ieee80211;

import inet.linklayer.ieee80211.mac.Ieee80211NewMac;
import inet.linklayer.ieee80211.mgmt.Ieee80211MgmtAdhoc;
import inet.linklayer.radio.Radio;

module TerminalIeee80211NicAdhoc
{
    parameters:
        int id = default(-1);
        string radioType = default("TerminalIeee80211gRadio");
        @display("i=block/ifcard");
    gates:
        input uppergateIn; // to upper layers
        output uppergateOut; // from upper layers
        input radioIn; // to receive AirFrames
```

```
    submodules:
        mgmt: Ieee80211MgmtAdhoc {
            parameters:
                @display("p=96,69;q=wlanDataQueue");
        }
        mac: Ieee80211NewMac {
            parameters:
                queueModule = "mgmt";
                @display("p=96,155");
        }
        radio: <radioType> like Radio {
            parameters:
                id = id;
                @display("p=96,240");
        }
    connections allowunconnected:
        radioIn --> radio.radioIn;
        radio.uppergateIn <-- mac.lowergateOut;
        radio.uppergateOut --> mac.lowergateIn;

        mac.uppergateOut --> mgmt.macIn;
        mac.uppergateIn <-- mgmt.macOut;

        mgmt.uppergateOut --> uppergateOut;
        mgmt.uppergateIn <-- uppergateIn;

}
```

A.1.6 TerminalIeee80211gRadio

```
package manet.linklayer.radio;

import inet.linklayer.radio.Ieee80211NewRadio;

simple TerminalIeee80211gRadio extends Ieee80211NewRadio
{
    parameters:
        @class(TerminalIeee80211gRadio);
        int id = default(-1);
}
```

A.1.7 Attacker

```
package manet.nodes;

import inet.base.NotificationBoard;
import inet.mobility.BasicMobility;
import manet.linklayer.ieee80211.TerminalIeee80211NicAdhoc;
import manet.nodes.inet.analysisNetworkLayerGlobalArp;

module Attacker
{
    parameters:
        @node();
        int id = default(-1);
        string mobilityType = default("AttackerMobility");
        @display("i=device/pocketpc_s");
    gates:
        input radioIn @directIn;
    submodules:
        notificationBoard: NotificationBoard {
            parameters:
                @display("p=60,70");
        }
        wlan: TerminalIeee80211NicAdhoc {
            parameters:
                id = id;
                radioType = "AttackerIeee80211gRadio";
                @display("p=248,349;q=queue");
        }
        mobility: <mobilityType> like BasicMobility {
            parameters:
                id = id;
                registerControl = false;
                @display("p=149,307");
        }
    connections allowunconnected:
        radioIn --> wlan.radioIn;
}
```

A.1.8 AttackerIeee80211gRadio

```
package manet.linklayer.radio;

import inet.linklayer.radio.Ieee80211NewRadio;

simple AttackerIeee80211gRadio extends Ieee80211NewRadio
{
    parameters:
        @class(AttackerIeee80211gRadio);
        int id = default(-1);
        int mode = default(2);
        int rng = default(5);
        double Pjam = default(1);
        double onPeriod @unit("s") = default(1us);
        double offPeriod @unit("s") = default(0s);
        bool passThrough = default(false);
}
```

A.1.9 AttackerMobility

```
package manet.mobility;

simple AttackerMobility like inet.mobility.BasicMobility
{
    parameters:
        bool debug = default(false);
        int id = default(-1);
        int rng = default(1);
        double x @unit("m") = default(-1m);
        double y @unit("m") = default(-1m);
        double w @unit("m") = default(15m);
        double randomChoice = default(0.01);
        double updateInterval @unit("s") = default(0.1s);
        double startTime @unit("s") = default(0s);
        bool registerControl = default(true);
        volatile double speed = default(5);
        volatile double waitTime @unit("s") = default(0s);
        string wpfile = default("");
        @display("i=block/cogwheel_s");
}
```

A.1.10 Instruments

```
package manet.instrument;

import experiment.instrument.Instrument;

module Instruments
{
submodules:
        appsend: Instrument {
                prefix = "appsend";
                format = "8f,2i,2i,1i,2i";
        }
        apprecv: Instrument {
                prefix = "apprecv";
                format = "8f,4f";
        }
        appiat: Instrument {
                prefix = "appiat";
                format = "8f,2i,8f,8f";
        }
        ipappsend: Instrument {
                prefix = "ipappsend";
                format = "8f,4f,2i,1i,1i";
        }
        ipapprecv: Instrument {
                prefix = "ipapprecv";
                format = "8f,4f,2i,1i,1i";
        }
        iprecv: Instrument {
                prefix = "iprecv";
                format = "8f,4f,2i,1i,1i";
        }
        ipsend: Instrument {
                prefix = "ipsend";
                format = "8f,2i,1i";
        }
        iproute: Instrument {
                prefix = "iproute";
                format = "8f,8f,1i";
        }
        hostpos: Instrument {
                prefix = "hostpos";
                format = "8f,2i,4f,4f,4f";
```

```
        }
        hostwp: Instrument {
                prefix = "hostwp";
                format = "8f,2i,4f,4f,4f";
        }
        framesend: Instrument {
                prefix = "framesend";
                format = "8f,2i,4f,4f,4f,4f,1i";
        }
        framerecv: Instrument {
                prefix = "framerecv";
                format = "8f,2i,4f,4f,4f,1i";
        }
        framelost: Instrument {
                prefix = "framelost";
                format = "8f,2i,4f,4f,4f,1i";
        }
        framejam: Instrument {
                prefix = "framejam";
                format = "8f,2i";
        }
        manetgroup: Instrument {
                prefix = "manetgroup";
                format = "8f,2i,2i";
        }
        manetlink: Instrument {
                prefix = "manetlink";
                format = "8f,8f";
        }
        connections allowunconnected:
}
```

A.1.11 Instrument

```
package experiment.instrument;

simple Instrument
{
    parameters:
        string prefix = default("unknown");
        string path = default("results/");
        string format = default("");
        int compress = default(1);
        int maxBytes @unit("B") = default(0B);
        bool enabled = default(false);
        bool readable = default(false);
        double startTime @unit("s") = default(0s);

    gates:
        input in;
}
```

A.2 INETMANET Modules

The module definitions contained in the following sections are copied directly from the INETMANET project source code. As such, they are copyright of their respective authors and licensed under the terms laid out in the original source files. The project is available from its website located at *http://inet.omnetpp.org/*.

A.2.1 UDP

```
// copied from INETMANET project src/transport/udp/UDP.ned
package inet.transport.udp;

simple UDP
{
    parameters:
        @display("i=block/transport");
    gates:
        input appIn[] @labels(UDPControlInfo/down);
        input ipIn @labels(UDPPacket,IPControlInfo/up);
        input ipv6In @labels(UDPPacket,IPv6ControlInfo/up);
        output appOut[] @labels(UDPControlInfo/up);
        output ipOut @labels(UDPPacket,IPControlInfo/down);
        output ipv6Out @labels(UDPPacket,IPv6ControlInfo/down);
}
```

A.2.2 IP

```
// copied from INETMANET project src/networklayer/ipv4/IP.ned
package inet.networklayer.ipv4;

simple IP
{
    parameters:
        double procDelay @unit("s") = default(0s);
        int timeToLive = default(32);
        int multicastTimeToLive;
        string protocolMapping;
        double fragmentTimeout @unit("s") = default(60s);
        bool forceBroadcast = default(false);
        @display("i=block/routing");
    gates:
        input transportIn[] @labels(IPControlInfo/down,TCPSegment,UDPPacket);
        output transportOut[] @labels(IPControlInfo/up,TCPSegment,UDPPacket);
        input queueIn[] @labels(IPDatagram);
        output queueOut @labels(IPDatagram);
}
```

A.2.3 DYMO

```
// copied from INETMANET project src/networklayer/manetrouting/DYMOFAU.ned
package inet.networklayer.manetrouting;

import inet.networklayer.manetrouting.BaseRouting;

simple DYMO extends BaseRouting
{
    parameters:
        @class(DYMO);
        bool coreDebug = default(false);
        int RESPONSIBLE_ADDRESSES_PREFIX = default(-1);
        double ROUTE_AGE_MIN_TIMEOUT @unit("s") = default(1s);
        double ROUTE_AGE_MAX_TIMEOUT @unit("s") = default(60s);
        double ROUTE_NEW_TIMEOUT @unit("s") = default(5s);
        double ROUTE_USED_TIMEOUT @unit("s") = default(5s);
        double ROUTE_DELETE_TIMEOUT @unit("s") = default(10s);
        int MIN_HOPLIMIT = default(5);
        int MAX_HOPLIMIT = default(10);
        double RREQ_RATE_LIMIT = default(10);
        int RREQ_BURST_LIMIT = default(3);
        double RREQ_WAIT_TIME @unit("s") = default(2s);
        int RREQ_TRIES = default(3);
        int BUFFER_SIZE_PACKETS = default(50);
        int BUFFER_SIZE_BYTES @unit("B") = default(75000B);
    gates:
        input from_ip;
        output to_ip;
}
```

A.2.4 ChannelControl

```
// copied from INETMANET project src/world/ChannelControl.ned
package inet.world;

simple ChannelControl
{
    parameters:
        bool coreDebug = default(false);
        double playgroundSizeX = default(600);
        double playgroundSizeY = default(400);
        double pMax @unit("mW") = default(20mW);
        double sat @unit("dBm") = default(-110dBm);
        double alpha = default(2);
        double carrierFrequency @unit("Hz") = default(2.4GHz);
        int numChannels = default(1);
        string propagationModel = default("PathLossReceptionModel")
        @enum("","PathLossReceptionModel","TwoRayGroundModel","RiceModel",
                "RayleighModel","NakagamiModel","FreeSpaceModel",
                "LogNormalShadowingModel");
        @display("i=misc/sun");
        @labels(node);
}
```

A.3 Experiment Configurations

```
[General]
network = networks.manet
sim-time-limit = 5 min
repeat = 75
num-rngs = 6
seed-set = ${repetition}
cmdenv-express-mode = true
cmdenv-status-frequency = 5min
**.vector-recording = false
**.scalar-recording = false
**.cmdenv-ev-output = false
**.debug = false
**.coreDebug = false
**.channelNumber = 0
**.routingFile = ""
**.ip.procDelay = 10us
**.ip.timeToLive = 32
**.arp.retryTimeout = 1s
**.arp.retryCount = 3
**.arp.cacheTimeout = 100s
**.fixFSM = true
**.wlan.radio.berTableFile="model/per_table_80211g_Trivellato.dat"
**.wlan.mac.bitrate = 54Mbps
**.wlan.radio.bitrate = 54Mbps
**.wlan.radio.sensitivity = -90dBm
**.broadCastDelay=uniform(0s,0.005s)
*.terminalcontrol.connectionDistance = 98m
*.terminalcontrol.positionInterval = 0.1s
*.channelcontrol.pMax = 0.1mW
**.transmitterPower = 0.1mW
*.host[*].numUdpApps = 1
*.host[*].udpApp[*].type-name = "OnOffApp"
*.host[*].udpApp[*].startTime = uniform( 0, 16, 2 )*1s
*.host[*].mobility.startTime = uniform( 0, 0.1, 1 )*1s

[Config Network]
include dymo.ini
*.logger.*.enabled = true
*.logger.*.compress = 2
*.logger.*.maxBytes = 60MB
*.host[*].udpApp[*].requestOn = pareto_shifted( 3, 5, 0, 2 )*1s
*.host[*].udpApp[*].requestOff = 0s
*.host[*].udpApp[*].requestBytes = 2KB
*.host[*].udpApp[*].segmentBytes = 1KB
*.host[*].mobility.speed = 5

[Config Manet]
extends = Network
*.numHosts = 360
*.playgroundSizeX = 1500
*.playgroundSizeY = 1000

[Config ManetSmall]
extends = Network
*.numHosts = 50
*.playgroundSizeX = 600
*.playgroundSizeY = 300
```

A.3.1 Baseline

```
include manet.ini

[Config SmallBase]
extends = ManetSmall
sim-time-limit = 30 min
*.logger.*.path = "/global/scratch/<username>/.../${configname}/sim/r${runnumber}"

[Config Base]
extends = Manet
sim-time-limit = 30 min
*.logger.*.path = "/global/scratch/<username>/.../${configname}/sim/r${runnumber}"
```

A.3.2 Constant Jamming

```
include base.ini

[Config SmallAttack0]
extends = SmallBase
*.numAtkrs = 1
*.atkr[0].mobility.wpfile = "data/middle.wp"
*.atkr[*].mobility.speed = 5
*.atkr[*].**.mode = 0
*.atkr[*].**.Pjam = 1
*.atkr[*].**.onPeriod = 50ms
```

```
[Config SmallAttack0Power]
extends = SmallAttack0
*.atkr[0].**.transmitterPower = ${ PWR = 0.0253, 0.227, 0.404 }*1mW
*.channelcontrol.pMax = ${ 0.1, 0.227, 0.404 ! PWR }*1mW

[Config SmallAttack0Motion]
extends = SmallAttack0
*.atkr[0].mobility.wpfile = ${"data/motion2.wp","data/motion5.wp" }

[Config SmallAttack0Multiple]
extends = SmallAttack0
*.numAtkrs = ${ a = 2, 3, 4 }
*.atkr[0].mobility.wpfile =
        ${"data/motion26.wp","data/motion7.wp", "data/motion3.wp" ! a  }
*.atkr[1].mobility.wpfile =
        ${"data/motion39.wp","data/motion10.wp", "data/motion9.wp" ! a }
*.atkr[2].mobility.wpfile =
        ${"", "data/motion22.wp", "data/motion21.wp" ! a }
*.atkr[3].mobility.wpfile = ${"","", "data/motion27.wp" ! a }

[Config Attack0]
extends = Base
*.numAtkrs = 1
*.atkr[0].mobility.wpfile = "data/middle_large.wp"
*.atkr[*].mobility.speed = 5
*.atkr[*].**.mode = 0
*.atkr[*].**.Pjam = 1
*.atkr[*].**.onPeriod = 50ms

[Config Attack0Power]
extends = Attack0
*.atkr[*].**.transmitterPower = ${ PWR = 0.0253, 0.227, 0.404 }*1mW
*.channelcontrol.pMax = ${ 0.1, 0.227, 0.404 ! PWR }*1mW

[Config Attack0Motion]
extends = Attack0
*.atkr[0].mobility.wpfile = ${"data/motion11_large.wp","data/motion36_large.wp" }

[Config Attack0Multiple]
extends = Attack0
*.numAtkrs = ${ a = 2, 3, 4 }
*.atkr[0].mobility.wpfile =
        ${"data/motion1_large.wp","data/motion6_large.wp",
        "data/motion5_large.wp" ! a  }
*.atkr[1].mobility.wpfile =
        ${"data/motion10_large.wp","data/motion18_large.wp",
        "data/motion17_large.wp" ! a }
*.atkr[2].mobility.wpfile =
        ${"", "data/motion40_large.wp",
        "data/motion28_large.wp" ! a }
*.atkr[3].mobility.wpfile = ${"","", "data/motion38_large.wp" ! a }
```

A.3.3 Random Jamming

```
include base.ini

[Config SmallAttack1]
extends = SmallBase
*.numAtkrs = 1
*.atkr[0].mobility.wpfile = "data/middle.wp"
*.atkr[*].mobility.speed = 5
*.atkr[*].**.mode = 1
*.atkr[*].**.Pjam = 1
*.atkr[*].**.onPeriod = uniform( 38, 56, 5 )*1us
*.atkr[*].**.offPeriod = uniform( 0, 1, 5 )*50ms

[Config SmallAttack1Power]
extends = SmallAttack1
*.atkr[0].**.transmitterPower = ${ PWR = 0.0253, 0.227, 0.404 }*1mW
*.channelcontrol.pMax = ${ 0.1, 0.227, 0.404 ! PWR }*1mW

[Config SmallAttack1Motion]
extends = SmallAttack1
*.atkr[0].mobility.wpfile = ${"data/motion2.wp","data/motion5.wp" }

[Config SmallAttack1Multiple]
extends = SmallAttack1
*.numAtkrs = ${ a = 2, 3, 4 }
*.atkr[0].mobility.wpfile =
        ${"data/motion26.wp","data/motion7.wp", "data/motion3.wp" ! a  }
*.atkr[1].mobility.wpfile =
        ${"data/motion39.wp","data/motion10.wp", "data/motion9.wp" ! a }
*.atkr[2].mobility.wpfile =
        ${"", "data/motion22.wp", "data/motion21.wp" ! a }
*.atkr[3].mobility.wpfile = ${"","", "data/motion27.wp" ! a }
```

```
[Config Attack1]
extends = Base
*.numAtkrs = 1
*.atkr[0].mobility.wpfile = "data/middle_large.wp"
*.atkr[*].mobility.speed = 5
*.atkr[*].**.mode = 1
*.atkr[*].**.Pjam = 1
*.atkr[*].**.onPeriod = uniform( 38, 56, 5 )*1us
*.atkr[*].**.offPeriod = uniform( 0, 1, 5 )*50ms

[Config Attack1Power]
extends = Attack1
*.atkr[*].**.transmitterPower = ${ PWR = 0.0253, 0.227, 0.404 }*1mW
*.channelcontrol.pMax = ${ 0.1, 0.227, 0.404 ! PWR }*1mW

[Config Attack1Motion]
extends = Attack1
*.atkr[0].mobility.wpfile = ${"data/motion11_large.wp","data/motion36_large.wp" }

[Config Attack1Multiple]
extends = Attack1
*.numAtkrs = ${ a = 2, 3, 4 }
*.atkr[0].mobility.wpfile =
        ${"data/motion1_large.wp","data/motion6_large.wp",
        "data/motion5_large.wp" ! a  }
*.atkr[1].mobility.wpfile =
        ${"data/motion10_large.wp","data/motion18_large.wp",
        "data/motion17_large.wp" ! a }
*.atkr[2].mobility.wpfile =
        ${"", "data/motion40_large.wp", "data/motion28_large.wp" ! a }
*.atkr[3].mobility.wpfile = ${"","", "data/motion38_large.wp" ! a }
```

A.3.4 Reactive Jamming

```
include base.ini

[Config SmallAttack2]
extends = SmallBase
*.numAtkrs = 1
*.atkr[0].mobility.wpfile = "data/middle.wp"
*.atkr[*].mobility.speed = 5
*.atkr[*].**.mode = 2
*.atkr[*].**.Pjam = 1
# key for the mean on period
*.atkr[*].**.onPeriod = 47us

[Config SmallAttack2Power]
extends = SmallAttack2
*.atkr[0].**.transmitterPower = ${ PWR = 0.0253, 0.227, 0.404 }*1mW
*.channelcontrol.pMax = ${ 0.1, 0.227, 0.404 ! PWR }*1mW

[Config SmallAttack2Motion]
extends = SmallAttack2
*.atkr[0].mobility.wpfile = ${"data/motion2.wp","data/motion5.wp" }

[Config SmallAttack2Multiple]
extends = SmallAttack2
*.numAtkrs = ${ a = 2, 3, 4 }
*.atkr[0].mobility.wpfile =
        ${"data/motion26.wp","data/motion7.wp", "data/motion3.wp" ! a  }
*.atkr[1].mobility.wpfile =
        ${"data/motion39.wp","data/motion10.wp", "data/motion9.wp" ! a }
*.atkr[2].mobility.wpfile =
        ${"", "data/motion22.wp", "data/motion21.wp" ! a }
*.atkr[3].mobility.wpfile = ${"","", "data/motion27.wp" ! a }

[Config Attack2]
extends = Base
*.numAtkrs = 1
*.atkr[0].mobility.wpfile = "data/middle_large.wp"
*.atkr[*].mobility.speed = 5
*.atkr[*].**.mode = 2
*.atkr[*].**.Pjam = 1
# key for the mean on period
*.atkr[*].**.onPeriod = 47us

[Config Attack2Power]
extends = Attack2
*.atkr[*].**.transmitterPower = ${ PWR = 0.0253, 0.227, 0.404 }*1mW
*.channelcontrol.pMax = ${ 0.1, 0.227, 0.404 ! PWR }*1mW

[Config Attack2Motion]
extends = Attack2
*.atkr[0].mobility.wpfile = ${"data/motion11_large.wp","data/motion36_large.wp" }
```

```
[Config Attack2Multiple]
extends = Attack2
*.numAtkrs = ${ a = 2, 3, 4 }
*.atkr[0].mobility.wpfile =
        ${"data/motion1_large.wp","data/motion6_large.wp",
        "data/motion5_large.wp" ! a  }
*.atkr[1].mobility.wpfile =
        ${"data/motion10_large.wp","data/motion18_large.wp",
        "data/motion17_large.wp" ! a }
*.atkr[2].mobility.wpfile =
        ${"", "data/motion40_large.wp", "data/motion28_large.wp" ! a }
*.atkr[3].mobility.wpfile = ${"","", "data/motion38_large.wp" ! a }
```

A.3.5 Random Reactive Jamming

```
include base.ini

[Config SmallAttack3]
extends = SmallBase
*.numAtkrs = 1
*.atkr[0].mobility.wpfile = "data/middle.wp"
*.atkr[*].mobility.speed = 5
*.atkr[*].**.mode = 2
*.atkr[*].**.Pjam = 0.5
# key for the mean on period
*.atkr[*].**.onPeriod = 47us

[Config SmallAttack3Power]
extends = SmallAttack3
*.atkr[0].**.transmitterPower = ${ PWR = 0.0253, 0.227, 0.404 }*1mW
*.channelcontrol.pMax = ${ 0.1, 0.227, 0.404 ! PWR }*1mW

[Config SmallAttack3Motion]
extends = SmallAttack3
*.atkr[0].mobility.wpfile = ${"data/motion2.wp","data/motion5.wp" }

[Config SmallAttack3Multiple]
extends = SmallAttack3
*.numAtkrs = ${ a = 2, 3, 4 }
*.atkr[0].mobility.wpfile =
        ${"data/motion26.wp","data/motion7.wp", "data/motion3.wp" ! a  }
*.atkr[1].mobility.wpfile =
        ${"data/motion39.wp","data/motion10.wp", "data/motion9.wp" ! a }
*.atkr[2].mobility.wpfile =
        ${"", "data/motion22.wp", "data/motion21.wp" ! a }
*.atkr[3].mobility.wpfile = ${"","", "data/motion27.wp" ! a }

[Config Attack3]
extends = Base
*.numAtkrs = 1
*.atkr[0].mobility.wpfile = "data/middle_large.wp"
*.atkr[*].mobility.speed = 5
*.atkr[*].**.mode = 2
*.atkr[*].**.Pjam = 0.5
# key for the mean on period
*.atkr[*].**.onPeriod = 47us

[Config Attack3Power]
extends = Attack3
*.atkr[*].**.transmitterPower = ${ PWR = 0.0253, 0.227, 0.404 }*1mW
*.channelcontrol.pMax = ${ 0.1, 0.227, 0.404 ! PWR }*1mW

[Config Attack3Motion]
extends = Attack3
*.atkr[0].mobility.wpfile = ${"data/motion11_large.wp","data/motion36_large.wp" }

[Config Attack3Multiple]
extends = Attack3
*.numAtkrs = ${ a = 2, 3, 4 }
*.atkr[0].mobility.wpfile =
        ${"data/motion1_large.wp","data/motion6_large.wp",
        "data/motion5_large.wp" ! a  }
*.atkr[1].mobility.wpfile =
        ${"data/motion10_large.wp","data/motion18_large.wp",
        "data/motion17_large.wp" ! a }
*.atkr[2].mobility.wpfile =
        ${"", "data/motion40_large.wp", "data/motion28_large.wp" ! a }
*.atkr[3].mobility.wpfile = ${"","", "data/motion38_large.wp" ! a }
```

Appendix B

Experiment Results

B.1 Baseline MANET

Table B.1: Baseline Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	69 27	$0, 6$ $0, 48$	8 3	15 12	17 5
X^{PDR}	41 27	$0, 34$ $0, 48$	1 5	9 11	11 3
X^{NRO}	36 34	$1, 38$ $0, 41$	3 8	12 9	8 7
X^{DD}	46 49	$0, 29$ $0, 26$	1 1	14 18	14 13
X^{HT}	58 50	$0, 17$ $0, 25$	2 1	27 18	9 9
X^{MEU}	59 60	$0, 16$ $0, 15$	4 11	20 20	15 7

Table B.2: Baseline Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	5.714286×10^{-1} –	9.781971×10^{-1} –	8.666667×10^{-1} –	1.000000 –
X^{PDR}	6.666667×10^{-2} –	5.286939×10^{-1} –	5.370879×10^{-1} –	1.000000 –
X^{NRO}	– –	– –	– –	– –
X^{DD}	2.380036×10^{-4} 2.247629×10^{-4}	1.832914×10^{-3} 1.072372×10^{-2}	5.932240 6.114430	1.399081×10^{1} 1.410451×10^{1}
X^{HT}	– –	– –	– –	– –
X^{MEU}	2.772945×10^{-5} –	2.332295×10^{-3} –	1.477410×10^{-2} –	3.059016×10^{-2} –

B.2 Constant Jamming

B.2.1 Range Scenario

Table B.3: Small Constant Range Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{PDR}	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{NRO}	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{DD}	22	0, 53	0	6	10
	0	53, 22	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{HT}	54	0, 21	0	8	25
	0	53, 22	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{MEU}	2	0, 73	2	–	–
	10	0, 65	4	3	2
	15	0, 60	2	6	5
	7	0, 68	5	1	2

Table B.4: Small Constant Range Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	2.380031×10^{-4}	2.054463×10^{-3}	6.013426	1.398440×10^{1}
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{HT}	1.000000	1.819484	7.000000	1.200000×10^{1}
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–

Table B.5: Large Constant Range Ergodicity

	Sample X_n			Mode M_X	
Feature	Stationary	Non-Stationary	Non-Ergodic	Count	Largest
X^{RA}	8	0, 67	3	3	2
	10	0, 65	1	3	4
	1	36, 38	1	–	–
	0	75, 0	0	–	–
X^{PDR}	10	0, 65	0	5	3
	0	67, 8	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{NRO}	27	0, 48	2	10	6
	22	3, 50	6	8	4
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{DD}	44	0, 31	4	14	11
	26	0, 49	0	4	14
	23	0, 52	1	9	7
	29	0, 46	1	6	12
X^{HT}	47	0, 28	5	16	7
	52	0, 23	0	19	11
	59	0, 16	0	11	14
	71	0, 4	0	4	43
X^{MEU}	25	0, 50	2	11	5
	10	0, 65	5	3	2
	15	0, 60	7	3	4
	28	0, 47	4	9	6

Table B.6: Large Constant Range Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	2.269501×10^{-4}	6.031249×10^{-3}	5.958300	1.403573×10^{1}
	2.380010×10^{-4}	2.043233×10^{-3}	5.967641	1.399180×10^{1}
	–	–	–	–
	2.156092×10^{-4}	1.336134×10^{-3}	6.068761	1.401533×10^{1}
X^{HT}	–	–	–	–
	1.000000	2.044112	1.000000×10^{1}	2.100000×10^{1}
	1.000000	1.737456	8.000000	1.900000×10^{1}
	1.000000	1.627005	7.000000	1.200000×10^{1}
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–

B.2.2 Motion Scenario

Table B.7: Small Constant Motion Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	0 0	75, 0 75, 0	0 0	– –	– –
X^{PDR}	0 0	75, 0 75, 0	0 0	– –	– –
X^{NRO}	0 0	75, 0 75, 0	0 0	– –	– –
X^{DD}	0 0	4, 71 10, 65	0 0	– –	– –
X^{HT}	0 1	4, 71 10, 64	0 1	– –	– –
X^{MEU}	3 5	0, 72 0, 70	1 5	1 –	2 –

Table B.8: Small Constant Motion Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	– –	– –	– –	– –
X^{PDR}	– –	– –	– –	– –
X^{NRO}	– –	– –	– –	– –
X^{DD}	– –	– –	– –	– –
X^{HT}	– –	– –	– –	– –
X^{MEU}	– –	– –	– –	– –

Table B.9: Large Constant Motion Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	0 0	0, 75 0, 75	0 0	– –	– –
X^{PDR}	0 0	41, 34 24, 51	0 0	– –	– –
X^{NRO}	1 0	2, 72 0, 75	1 0	– –	– –
X^{DD}	5 21	0, 70 0, 54	1 2	2 4	2 9
X^{HT}	3 27	0, 72 0, 48	3 4	– 6	– 8
X^{MEU}	0 3	0, 75 0, 72	0 1	– 1	– 2

Table B.10: Large Constant Motion Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	– –	– –	– –	– –
X^{PDR}	– –	– –	– –	– –
X^{NRO}	– –	– –	– –	– –
X^{DD}	– –	– –	– –	– –
X^{HT}	– –	– –	– –	– –
X^{MEU}	– –	– –	– –	– –

B.2.3 Multiple Attacker Scenario

Table B.11: Small Constant Multiple Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{PDR}	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{NRO}	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{DD}	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{HT}	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{MEU}	5	0, 70	2	2	2
	1	0, 74	1	–	–
	1	7, 67	1	–	–

Table B.12: Small Constant Multiple Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{HT}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–

Table B.13: Large Constant Multiple Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	0	11, 64	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{PDR}	0	75, 0	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{NRO}	0	62, 13	0	–	–
	0	75, 0	0	–	–
	0	75, 0	0	–	–
X^{DD}	19	0, 56	2	6	7
	26	0, 49	1	9	7
	22	0, 53	0	6	7
X^{HT}	45	0, 30	0	6	16
	65	0, 10	0	6	24
	51	0, 24	0	2	40
X^{MEU}	0	0, 75	0	–	–
	14	0, 61	1	7	4
	32	0, 43	5	8	10

Table B.14: Large Constant Multiple Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	– – –	– – –	– – –	– – –
X^{PDR}	– – –	– – –	– – –	– – –
X^{NRO}	– – –	– – –	– – –	– – –
X^{DD}	– – –	– – –	– – –	– – –
X^{HT}	1.000000 1.000000 1.000000	1.794022 1.697707 1.635596	1.000000×10^{1} 8.000000 8.000000	2.100000×10^{1} 2.000000×10^{1} 2.400000×10^{1}
X^{MEU}	– – 2.943733×10^{-4}	– – 1.158328×10^{-3}	– – 2.708904×10^{-3}	– – 5.123434×10^{-3}

B.3 Random Jamming

B.3.1 Range Scenario

Table B.15: Small Random Range Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	70	0, 5	7	16	19
	65	0, 10	9	14	18
	70	0, 5	7	14	23
	69	0, 6	12	7	26
X^{PDR}	47	1, 27	2	10	14
	43	0, 32	5	13	16
	48	1, 26	1	15	10
	44	0, 31	7	12	9
X^{NRO}	41	1, 33	1	13	12
	35	0, 40	3	9	11
	40	3, 32	1	13	8
	42	3, 30	2	11	12
X^{DD}	48	0, 27	5	16	12
	46	0, 29	2	18	10
	46	0, 29	2	18	10
	52	0, 23	4	22	10
X^{HT}	59	0, 16	2	25	14
	55	0, 20	2	22	13
	55	0, 20	2	21	11
	55	0, 20	2	16	12
X^{MEU}	57	0, 18	18	14	9
	59	0, 16	17	17	5
	55	0, 20	12	22	7
	54	0, 21	16	13	8

Table B.16: Small Random Range Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	5.714286×10^{-1}	9.781475×10^{-1}	8.708333×10^{-1}	1.000000
	5.000000×10^{-1}	9.781079×10^{-1}	8.786765×10^{-1}	1.000000
	5.000000×10^{-1}	9.771862×10^{-1}	8.708333×10^{-1}	1.000000
	5.000000×10^{-1}	9.770710×10^{-1}	8.823529×10^{-1}	1.000000
X^{PDR}	5.882353×10^{-2}	5.291444×10^{-1}	5.303309×10^{-1}	1.000000
	4.545455×10^{-2}	5.319262×10^{-1}	5.384615×10^{-1}	1.000000
	5.882353×10^{-2}	5.321712×10^{-1}	5.536398×10^{-1}	1.000000
	–	–	–	–
X^{NRO}	1.521739×10^{-1}	5.866760	1.394647×10^{2}	3.111176×10^{2}
	7.575758×10^{-2}	5.805403	3.286043×10^{2}	7.492857×10^{2}
	–	–	–	–
	2.500000×10^{-2}	5.873102	3.009516×10^{2}	6.878571×10^{2}
X^{DD}	2.380014×10^{-4}	1.954116×10^{-3}	5.872166	1.389829×10^{1}
	2.380036×10^{-4}	1.985543×10^{-3}	5.935168	1.396478×10^{1}
	2.380036×10^{-4}	1.987636×10^{-3}	5.929262	1.400029×10^{1}
	2.380014×10^{-4}	1.951629×10^{-3}	5.895037	1.398553×10^{1}
X^{HT}	1.000000	3.154070	1.000000×10^{1}	3.000000×10^{1}
	1.000000	3.123936	1.000000×10^{1}	2.800000×10^{1}
	1.000000	2.938583	8.000000	1.500000×10^{1}
	1.000000	3.139935	1.000000×10^{1}	3.100000×10^{1}
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–

Table B.17: Large Random Range Ergodicity

	Sample X_n			Mode M_X	
Feature	Stationary	Non-Stationary	Non-Ergodic	Count	Largest
X^{RA}	28	0, 47	5	10	4
	25	0, 50	5	8	4
	27	0, 48	5	8	7
	24	0, 51	2	8	5
X^{PDR}	31	0, 44	2	11	9
	36	0, 39	5	12	6
	26	0, 49	3	8	8
	27	0, 48	4	9	5
X^{NRO}	33	0, 42	4	9	6
	30	0, 45	4	11	4
	31	0, 44	6	8	6
	26	0, 49	14	4	5
X^{DD}	50	0, 25	2	19	8
	47	0, 28	1	19	13
	51	0, 24	2	17	12
	54	0, 21	0	18	13
X^{HT}	57	0, 18	2	19	13
	56	0, 19	2	22	12
	52	0, 23	1	16	12
	55	0, 20	4	20	9
X^{MEU}	58	0, 17	11	16	7
	59	0, 16	15	17	6
	60	0, 15	18	15	6
	60	0, 15	7	23	5

Table B.18: Large Random Range Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	–	–	–	–
	2.263787×10^{-4}	1.049276×10^{-2}	6.160166	1.404798×10^{1}
	2.349850×10^{-4}	1.089352×10^{-2}	6.123552	1.409302×10^{1}
	2.380015×10^{-4}	1.131539×10^{-2}	6.105547	1.405158×10^{1}
X^{HT}	1.000000	5.087927	1.700000×10^{1}	3.300000×10^{1}
	1.000000	5.386525	1.700000×10^{1}	3.300000×10^{1}
	1.000000	5.141431	1.700000×10^{1}	3.300000×10^{1}
	–	–	–	–
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–

B.3.2 Motion Scenario

Table B.19: Small Random Motion Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	66 62	0, 9 0, 13	6 6	13 10	27 27
X^{PDR}	39 42	0, 36 0, 33	1 2	16 13	8 12
X^{NRO}	35 46	2, 38 0, 29	0 2	9 11	13 14
X^{DD}	51 48	0, 24 0, 27	2 5	19 17	14 12
X^{HT}	52 53	0, 23 0, 22	2 1	27 18	7 11
X^{MEU}	50 58	0, 25 0, 17	7 16	15 18	11 9

Table B.20: Small Random Motion Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	3.750000×10^{-1} 3.333333×10^{-1}	9.780063×10^{-1} 9.760279×10^{-1}	8.619048×10^{-1} 8.619048×10^{-1}	1.000000 1.000000
X^{PDR}	– 5.263158×10^{-2}	– 5.255704×10^{-1}	– 5.370879×10^{-1}	– 1.000000
X^{NRO}	7.317073×10^{-2} 5.882353×10^{-2}	5.842101 5.819685	4.236356×10^{2} 1.997780×10^{2}	9.594444×10^{2} 4.453636×10^{2}
X^{DD}	2.380009×10^{-4} 2.380014×10^{-4}	1.872908×10^{-3} 2.018324×10^{-3}	5.920012 5.890966	1.399498×10^{1} 1.395931×10^{1}
X^{HT}	– 1.000000	– 3.239050	– 9.000000	– 1.700000×10^{1}
X^{MEU}	5.460022×10^{-5} –	2.373419×10^{-3} –	1.339395×10^{-2} –	2.700278×10^{-2} –

Table B.21: Large Random Motion Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	29 26	0, 46 0, 49	10 7	9 8	3 4
X^{PDR}	30 30	0, 45 0, 45	6 4	10 8	6 6
X^{NRO}	30 38	0, 45 0, 37	3 7	10 14	7 5
X^{DD}	57 57	0, 18 0, 18	2 0	20 24	12 16
X^{HT}	54 52	0, 21 0, 23	0 2	16 15	11 13
X^{MEU}	60 55	0, 15 0, 20	7 5	19 21	6 7

Table B.22: Large Random Motion Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	– –	– –	– –	– –
X^{PDR}	– –	– –	– –	– –
X^{NRO}	– –	– –	– –	– –
X^{DD}	2.380018×10^{-4} 2.380020×10^{-4}	1.048255×10^{-2} 1.095017×10^{-2}	6.211945 6.125372	1.418222×10^{1} 1.406484×10^{1}
X^{HT}	1.000000 1.000000	5.150456 5.103951	1.700000×10^{1} 1.700000×10^{1}	3.300000×10^{1} 3.300000×10^{1}
X^{MEU}	– –	– –	– –	– –

B.3.3 Multiple Attackers Scenario

Table B.23: Small Random Multiple Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	69	0, 6	14	12	17
	66	0, 9	8	16	17
	65	1, 9	10	15	21
X^{PDR}	42	1, 32	4	12	13
	37	0, 38	3	11	8
	41	1, 33	11	7	10
X^{NRO}	38	1, 36	7	14	7
	32	2, 41	0	11	9
	39	1, 35	1	10	21
X^{DD}	47	0, 28	3	19	13
	47	0, 28	8	10	12
	51	1, 23	5	18	13
X^{HT}	56	0, 19	1	21	15
	57	0, 18	0	21	16
	49	1, 25	3	17	13
X^{MEU}	56	0, 19	20	13	10
	55	0, 20	18	12	7
	54	0, 21	13	17	7

Table B.24: Small Random Multiple Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	3.333333×10^{-1}	9.759564×10^{-1}	8.619048×10^{-1}	1.000000
	5.000000×10^{-1}	9.761435×10^{-1}	8.786765×10^{-1}	1.000000
	3.333333×10^{-1}	9.760567×10^{-1}	8.786765×10^{-1}	1.000000
X^{PDR}	6.250000×10^{-2}	5.066486×10^{-1}	5.312500×10^{-1}	1.000000
	–	–	–	–
	5.263158×10^{-2}	5.190570×10^{-1}	5.333333×10^{-1}	1.000000
X^{NRO}	–	–	–	–
	–	–	–	–
	3.030303×10^{-2}	6.013329	1.088123×10^{2}	2.360000×10^{2}
X^{DD}	2.286476×10^{-4}	1.923940×10^{-3}	5.892458	1.399602×10^{1}
	2.380019×10^{-4}	1.969450×10^{-3}	5.903517	1.393950×10^{1}
	2.206550×10^{-4}	2.046237×10^{-3}	5.918101	1.402363×10^{1}
X^{HT}	1.000000	3.198830	1.000000×10^{1}	2.400000×10^{1}
	1.000000	3.053124	9.000000	1.700000×10^{1}
	1.000000	3.189260	9.000000	3.000000×10^{1}
X^{MEU}	5.717311×10^{-5}	2.358956×10^{-3}	9.480915×10^{-3}	1.890466×10^{-2}
	–	–	–	–
	–	–	–	–

Table B.25: Large Random Multiple Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	25	0, 50	3	9	5
	32	0, 43	4	11	5
	21	0, 54	3	8	4
X^{PDR}	32	0, 43	2	11	7
	36	0, 39	9	12	5
	31	1, 43	9	7	8
X^{NRO}	30	0, 45	6	10	5
	29	0, 46	5	11	5
	28	0, 47	10	8	4
X^{DD}	46	0, 29	0	17	11
	55	0, 20	8	16	11
	52	0, 23	1	28	9
X^{HT}	53	0, 22	3	17	10
	55	0, 20	3	24	11
	53	0, 22	3	25	10
X^{MEU}	60	0, 15	13	21	6
	62	0, 13	12	18	6
	58	0, 17	12	19	6

Table B.26: Large Random Multiple Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	2.380020×10^{-4}	1.113762×10^{-2}	6.135997	1.404088×10^{1}
	2.300304×10^{-4}	1.149360×10^{-2}	6.107732	1.404046×10^{1}
	–	–	–	–
X^{HT}	1.000000	5.616759	1.700000×10^{1}	3.300000×10^{1}
	1.000000	5.179094	1.700000×10^{1}	3.300000×10^{1}
	1.000000	5.159478	1.700000×10^{1}	3.300000×10^{1}
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–

B.4 Reactive Jamming

B.4.1 Range Scenario

Table B.27: Small Reactive Range Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	0	0, 75	0	–	–
	0	0, 75	0	–	–
	0	0, 75	0	–	–
	0	0, 75	0	–	–
X^{PDR}	3	12, 60	1	1	2
	4	16, 55	0	2	2
	5	17, 53	0	2	3
	3	13, 59	1	1	2
X^{NRO}	4	5, 66	4	–	–
	3	6, 66	3	–	–
	1	5, 69	1	–	–
	3	5, 67	1	1	2
X^{DD}	10	0, 65	1	3	4
	6	0, 69	2	2	3
	8	0, 67	5	1	3
	11	0, 64	0	5	4
X^{HT}	22	0, 53	1	8	7
	16	0, 59	1	5	6
	15	0, 60	0	4	7
	14	0, 61	2	4	7
X^{MEU}	14	0, 61	2	5	3
	14	0, 61	2	5	3
	9	0, 66	0	4	4
	8	0, 67	0	4	2

Table B.28: Small Reactive Range Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{HT}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–

Table B.29: Large Reactive Range Ergodicity

	Sample X_n			Mode M_X	
Feature	Stationary	Non-Stationary	Non-Ergodic	Count	Largest
X^{RA}	19	0, 56	2	7	5
	17	0, 58	1	8	3
	21	0, 54	7	5	5
	17	0, 58	5	5	3
X^{PDR}	33	0, 42	5	10	6
	38	0, 37	2	13	7
	34	0, 41	9	10	4
	36	0, 39	16	12	3
X^{NRO}	33	0, 42	5	12	5
	34	0, 41	2	10	6
	35	0, 40	5	17	3
	30	0, 45	9	8	5
X^{DD}	51	0, 24	0	25	10
	48	0, 27	4	21	7
	52	0, 23	0	20	15
	48	0, 27	4	16	10
X^{HT}	49	0, 26	0	12	12
	48	0, 27	1	15	13
	56	0, 19	2	13	13
	49	0, 26	0	18	11
X^{MEU}	51	0, 24	10	17	6
	49	0, 26	9	17	7
	55	0, 20	8	22	5
	53	0, 22	14	15	4

Table B.30: Large Reactive Range Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	2.380020×10^{-4}	1.086180×10^{-2}	6.092419	1.403752×10^{1}
	–	–	–	–
	2.380016×10^{-4}	1.109656×10^{-2}	6.165978	1.415808×10^{1}
	2.370042×10^{-4}	1.028683×10^{-2}	6.146301	1.409671×10^{1}
X^{HT}	1.000000	5.574252	1.700000×10^{1}	3.300000×10^{1}
	1.000000	5.635096	1.700000×10^{1}	3.300000×10^{1}
	1.000000	5.482795	1.700000×10^{1}	3.300000×10^{1}
	1.000000	5.415199	1.700000×10^{1}	3.300000×10^{1}
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–

B.4.2 Motion Scenario

Table B.31: Small Reactive Motion Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	1 1	0, 74 1, 73	1 1	– –	– –
X^{PDR}	8 13	7, 60 5, 57	0 0	2 4	5 9
X^{NRO}	5 6	2, 68 1, 68	3 2	1 2	2 2
X^{DD}	7 12	0, 68 0, 63	1 2	3 3	3 5
X^{HT}	18 18	0, 57 0, 57	1 0	8 6	6 9
X^{MEU}	22 10	0, 53 0, 65	7 1	5 4	5 4

Table B.32: Small Reactive Motion Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	– –	– –	– –	– –
X^{PDR}	– –	– –	– –	– –
X^{NRO}	– –	– –	– –	– –
X^{DD}	– –	– –	– –	– –
X^{HT}	– –	– –	– –	– –
X^{MEU}	– –	– –	– –	– –

Table B.33: Large Reactive Motion Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	18 22	0, 57 0, 53	4 4	7 7	3 4
X^{PDR}	29 25	0, 46 0, 50	3 5	8 6	9 8
X^{NRO}	32 31	0, 43 0, 44	8 4	11 12	4 8
X^{DD}	53 48	0, 22 0, 27	0 5	26 18	8 9
X^{HT}	54 54	0, 21 0, 21	0 1	19 16	10 13
X^{MEU}	58 52	0, 17 0, 23	9 6	16 18	8 5

Table B.34: Large Reactive Motion Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	– –	– –	– –	– –
X^{PDR}	– –	– –	– –	– –
X^{NRO}	– –	– –	– –	– –
X^{DD}	– –	– –	– –	– –
X^{HT}	1.000000 1.000000	5.619795 5.471025	1.700000×10^1 1.700000×10^1	3.300000×10^1 3.300000×10^1
X^{MEU}	– –	– –	– –	– –

B.4.3 Multiple Attackers Scenario

Table B.35: Small Reactive Multiple Ergodicity

	Sample X_n			Mode M_X	
Feature	Stationary	Non-Stationary	Non-Ergodic	Count	Largest
X^{RA}	0 0 0	$1, 74$ $11, 64$ $34, 41$	0 0 0	– – –	– – –
X^{PDR}	0 0 0	$74, 1$ $75, 0$ $75, 0$	0 0 0	– – –	– – –
X^{NRO}	0 0 0	$33, 42$ $71, 4$ $75, 0$	0 0 0	– – –	– – –
X^{DD}	6 6 4	$0, 69$ $0, 69$ $0, 71$	1 2 1	1 2 2	5 2 2
X^{HT}	12 5 17	$0, 63$ $0, 70$ $0, 58$	0 1 0	6 2 5	4 3 8
X^{MEU}	3 1 3	$0, 72$ $0, 74$ $0, 72$	0 1 3	1 – –	3 – –

Table B.36: Small Reactive Multiple Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{HT}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–

Table B.37: Large Reactive Multiple Ergodicity

	Sample X_n			Mode M_X	
Feature	Stationary	Non-Stationary	Non-Ergodic	Count	Largest
X^{RA}	14	0, 61	4	5	3
	10	0, 65	5	3	3
	5	0, 70	3	1	2
X^{PDR}	30	0, 45	3	10	6
	32	0, 43	1	10	7
	20	0, 55	3	8	6
X^{NRO}	27	0, 48	9	8	4
	33	0, 42	5	11	8
	28	0, 47	6	10	4
X^{DD}	46	0, 29	0	20	9
	39	0, 36	2	22	6
	35	0, 40	7	8	9
X^{HT}	47	0, 28	0	16	9
	51	0, 24	1	18	8
	41	0, 34	1	17	7
X^{MEU}	45	0, 30	11	14	5
	43	0, 32	12	13	5
	43	0, 32	11	15	6

Table B.38: Large Reactive Multiple Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{HT}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–

B.5 Random Reactive Jamming

B.5.1 Range Scenario

Table B.39: Small Random Reactive Range Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	40	0, 35	9	10	8
	34	0, 41	6	13	4
	32	0, 43	3	9	11
	41	0, 34	10	12	6
X^{PDR}	43	0, 32	4	17	7
	38	0, 37	1	12	11
	46	0, 29	4	12	10
	47	1, 27	0	11	12
X^{NRO}	44	1, 30	2	11	10
	43	0, 32	1	14	10
	42	1, 32	1	18	8
	45	0, 30	1	13	12
X^{DD}	48	0, 27	0	22	11
	54	0, 21	2	23	9
	47	0, 28	4	20	8
	47	0, 28	9	20	8
X^{HT}	53	0, 22	1	20	11
	59	0, 16	1	16	19
	53	0, 22	1	20	16
	59	0, 16	3	16	12
X^{MEU}	58	0, 17	6	24	7
	55	0, 20	9	15	8
	50	0, 25	11	18	6
	54	0, 21	15	18	5

Table B.40: Small Random Reactive Range Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	– – 4.000000×10^{-1} – –	– – 9.726388×10^{-1} – –	– – 8.571429×10^{-1} – –	– – 1.000000 – –
X^{PDR}	4.347826×10^{-2} 5.882353×10^{-2} 5.882353×10^{-2}	5.266876×10^{-1} 5.187253×10^{-1} 5.186330×10^{-1}	5.491935×10^{-1} 5.370879×10^{-1} 5.278638×10^{-1}	1.000000 1.000000 1.000000
X^{NRO}	3.571429×10^{-2} 3.448276×10^{-2} – 2.321429×10^{-1}	5.850296 5.500303 – 5.613018	2.559764×10^{2} 6.099506×10^{1} – 7.737532×10^{2}	5.804000×10^{2} 1.296000×10^{2} – 1.776400×10^{3}
X^{DD}	2.380026×10^{-4} – – –	2.324883×10^{-3} – – –	6.012373 – – –	1.399822×10^{1} – – –
X^{HT}	1.000000 1.000000 1.000000 1.000000	3.151804 3.111886 3.122308 3.187453	9.000000 1.000000×10^{1} 1.000000×10^{1} 9.000000	1.600000×10^{1} 3.300000×10^{1} 2.600000×10^{1} 1.600000×10^{1}
X^{MEU}	– – – –	– – – –	– – – –	– – – –

Table B.41: Large Random Reactive Range Ergodicity

	Sample X_n			Mode M_X	
Feature	Stationary	Non-Stationary	Non-Ergodic	Count	Largest
X^{RA}	18	0, 57	2	7	3
	33	0, 42	5	11	7
	21	0, 54	7	6	4
	29	0, 46	4	7	8
X^{PDR}	33	0, 42	3	11	9
	27	0, 48	1	9	7
	32	0, 43	3	13	6
	29	0, 46	1	9	8
X^{NRO}	31	0, 44	6	12	5
	31	0, 44	7	9	8
	31	0, 44	6	10	6
	38	0, 37	10	12	5
X^{DD}	50	0, 25	2	17	17
	54	0, 21	1	24	12
	54	0, 21	0	20	14
	57	0, 18	3	22	16
X^{HT}	56	0, 19	4	16	10
	52	0, 23	1	17	10
	54	0, 21	2	16	16
	54	0, 21	1	17	12
X^{MEU}	59	0, 16	3	26	6
	59	0, 16	8	24	6
	59	0, 16	11	19	7
	63	0, 12	16	18	6

Table B.42: Large Random Reactive Range Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	2.237345×10^{-4}	1.043996×10^{-2}	6.164140	1.413765×10^{1}
	2.380056×10^{-4}	1.096958×10^{-2}	6.030648	1.405714×10^{1}
	2.380048×10^{-4}	1.136793×10^{-2}	6.156365	1.407139×10^{1}
	2.380015×10^{-4}	1.099299×10^{-2}	6.125237	1.406453×10^{1}
X^{HT}	1.000000	5.526442	1.700000×10^{1}	3.300000×10^{1}
	1.000000	5.250739	1.700000×10^{1}	3.300000×10^{1}
	1.000000	5.311078	1.700000×10^{1}	3.300000×10^{1}
	1.000000	5.151268	1.700000×10^{1}	3.300000×10^{1}
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–
	–	–	–	–

B.5.2 Motion Scenario

Table B.43: Small Random Reactive Motion Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	40 35	0, 35 0, 40	11 7	11 14	5 4
X^{PDR}	48 44	1, 26 1, 30	5 6	17 17	14 7
X^{NRO}	50 47	0, 25 1, 27	5 6	14 10	9 14
X^{DD}	42 42	0, 33 0, 33	0 6	11 16	15 9
X^{HT}	54 53	0, 21 0, 22	1 0	20 19	10 14
X^{MEU}	56 60	0, 19 0, 15	8 19	15 12	14 7

Table B.44: Small Random Reactive Motion Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	– –	– –	– –	– –
X^{PDR}	5.555556×10^{-2} –	5.263981×10^{-1} –	5.416667×10^{-1} –	1.000000 –
X^{NRO}	– 3.703704×10^{-2}	– 5.500047	– 1.413619×10^{2}	– 3.162000×10^{2}
X^{DD}	2.380043×10^{-4} –	2.449717×10^{-3} –	6.008713 –	1.397344×10^{1} –
X^{HT}	1.000000 1.000000	3.149421 3.198337	1.000000×10^{1} 9.000000	1.900000×10^{1} 1.700000×10^{1}
X^{MEU}	2.710667×10^{-5} –	2.384366×10^{-3} –	1.307621×10^{-2} –	2.641558×10^{-2} –

Table B.45: Large Random Reactive Motion Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	26 22	$0, 49$ $0, 53$	8 3	8 8	5 5
X^{PDR}	28 25	$0, 47$ $0, 50$	5 5	8 8	7 7
X^{NRO}	30 34	$0, 45$ $0, 41$	0 4	10 13	8 6
X^{DD}	48 50	$0, 27$ $0, 25$	0 1	18 21	11 9
X^{HT}	45 56	$0, 30$ $0, 19$	0 1	16 18	10 12
X^{MEU}	62 64	$0, 13$ $0, 11$	6 7	19 20	10 7

Table B.46: Large Random Reactive Motion Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	– –	– –	– –	– –
X^{PDR}	– –	– –	– –	– –
X^{NRO}	– –	– –	– –	– –
X^{DD}	2.380022×10^{-4} –	9.981608×10^{-3} –	6.144999 –	1.407758×10^{1} –
X^{HT}	1.000000 1.000000	5.254318 5.183728	1.700000×10^{1} 1.700000×10^{1}	3.300000×10^{1} 3.300000×10^{1}
X^{MEU}	1.300577×10^{-2} –	3.821596×10^{-2} –	3.890046×10^{-2} –	6.479515×10^{-2} –

B.5.3 Multiple Attackers Scenario

Table B.47: Small Random Reactive Multiple Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	22	0, 53	5	10	4
	9	0, 66	3	3	2
	5	0, 70	3	1	2
X^{PDR}	38	2, 35	1	7	14
	29	0, 46	7	10	4
	18	4, 53	0	4	9
X^{NRO}	26	0, 49	0	12	9
	26	0, 49	5	9	5
	6	3, 66	1	2	4
X^{DD}	44	0, 31	0	15	13
	41	0, 34	6	15	9
	40	0, 35	3	21	8
X^{HT}	53	0, 22	0	19	19
	48	0, 27	1	13	13
	44	0, 31	1	12	12
X^{MEU}	48	0, 27	12	11	9
	51	0, 24	5	17	8
	43	0, 32	5	11	7

Table B.48: Small Random Reactive Multiple Largest Mode Results

Feature	**Min**	**Mean**	**Median**	**Max**
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	5.882353×10^{-2}	5.271522×10^{-1}	5.416667×10^{-1}	1.000000
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	2.380049×10^{-4}	3.090478×10^{-3}	6.054450	1.398225×10^{1}
	–	–	–	–
	–	–	–	–
X^{HT}	1.000000	3.008773	1.000000×10^{1}	1.800000×10^{1}
	1.000000	2.873199	9.000000	1.700000×10^{1}
	1.000000	2.717904	9.000000	1.700000×10^{1}
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–

Table B.49: Large Random Reactive Multiple Ergodicity

	Sample X_n			Mode M_X	
Feature	**Stationary**	**Non-Stationary**	**Non-Ergodic**	**Count**	**Largest**
X^{RA}	23	0, 52	4	9	5
	21	0, 54	9	4	4
	25	0, 50	3	10	4
X^{PDR}	31	0, 44	2	14	6
	31	0, 44	1	12	7
	38	0, 37	9	11	6
X^{NRO}	34	0, 41	1	16	5
	37	0, 38	6	12	8
	33	0, 42	2	8	7
X^{DD}	57	0, 18	3	22	11
	52	0, 23	6	15	12
	51	0, 24	2	19	11
X^{HT}	56	0, 19	3	20	10
	54	0, 21	3	20	9
	53	0, 22	1	20	9
X^{MEU}	51	0, 24	7	20	7
	56	0, 19	11	19	6
	61	0, 14	6	19	6

Table B.50: Large Random Reactive Multiple Largest Mode Results

Feature	Min	Mean	Median	Max
X^{RA}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{PDR}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{NRO}	–	–	–	–
	–	–	–	–
	–	–	–	–
X^{DD}	2.380056×10^{-4}	1.097220×10^{-2}	6.140724	1.405169×10^{1}
	2.172643×10^{-4}	1.113955×10^{-2}	6.123661	1.407710×10^{1}
	2.370102×10^{-4}	1.054407×10^{-2}	6.143149	1.405724×10^{1}
X^{HT}	1.000000	5.296557	1.700000×10^{1}	3.300000×10^{1}
	–	–	–	–
	–	–	–	–
X^{MEU}	–	–	–	–
	–	–	–	–
	–	–	–	–