Statistical Assessment of Peer-to-Peer Botnet Features

by

Teghan Godkin
B.Eng., University of Victoria, 2010

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

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Statistical Assessment of Peer-to-Peer Botnet Features

by

Teghan Godkin
B.Eng., University of Victoria, 2010

Supervisory Committee

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

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

Supervisory Committee

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

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

ABSTRACT

Botnets are collections of compromised machines which are controlled by a remotely located adversary. Botnets are of significant interest to cybersecurity researchers as they are a core mechanism that allows adversarial groups to gain control over large scale computing resources. Recent botnets have become increasingly complex, relying on Peer-to-Peer (P2P) protocols for botnet command and control (C&C). In this work, a packet-level simulation of a Kademlia-based P2P botnet is used in conjunction with a statistical analysis framework to investigate how measured botnet features change over time and across an ensemble of simulations. The simulation results include non-stationary and non-ergodic behaviours illustrating the complex nature of botnet operation and highlighting 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. W. Neville for the opportunity to work on this challenging and interesting topic and for his constructive feedback, encouragement, and patience.

NSERC, the B.C. Government, and the UVic Faculty of Graduate Studies for providing my Scholarship funding.

Adam Verigin for his contributions in developing the simulator used in this work.

DEDICATION

To my parents, who continue to provide constant support and encouragement.
Thank you for believing in me.

Chapter 1

Introduction

1.1 Botnets

A bot is a computer running malicious software that allows the computer to be controlled by a remotely located adversary. A botnet is a network of these compromised machines controlled by one or more botmasters (human operators) via a Command and Control (C&C) channel, which is used to distribute commands to individual machines. Once the bots have received their commands they largely act as autonomous and independent agents. Hence, C&C is a defining characteristic of botnets and much botnet literature has focused on study of C&C structure rather than post-command bot actions. Botnets have been used for a range of nefarious activities such as sending spam emails, phishing, information harvesting, and distributed denial of service (DDos) attacks [3–5]. Botnets are of significant interest to cybersecurity researchers as they are a core mechanism that allows adversarial groups to gain control over large scale computing resources. Botnets are considered a global security threat and continue to be an area of active research as per DARPA's recent broad agency announcement BAA-11-02 [6].

1.2 Botnet Command & Control

In early botnets, Command and Control (C&C) communications were delivered from a centralized control server to all of the bots, as shown in Figure 1.1. In these botnets, Internet Relay Chat (IRC) protocol [7] was commonly used for botnet C&C [4, 5]. This type of botnet architecture is efficient since bots receive commands very quickly, however the centralized communications channel is a single point of failure for the botnet. Thus, botnets which relied on this type of C&C structure proved relatively easy for the defensive community to address [4, 8].

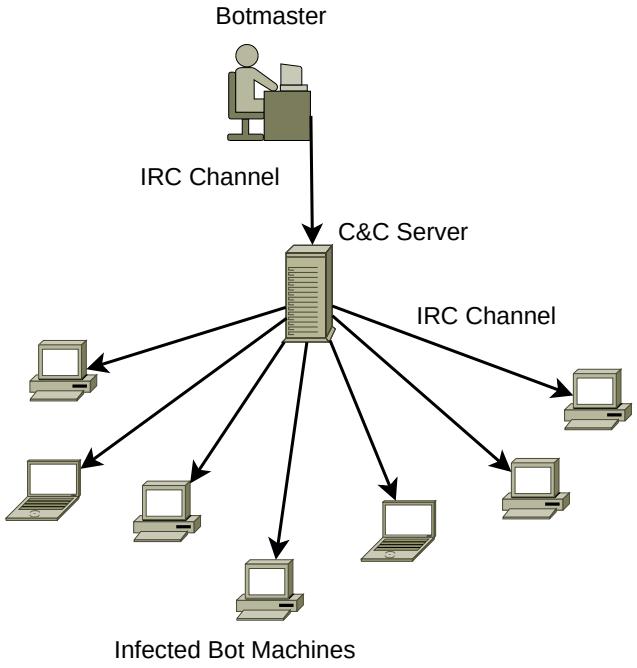


Figure 1.1: Centralized botnet architecture

The fact that centralized botnets are easily mitigated by disrupting their C&C channel prompted a shift to using Peer-to-Peer (P2P) protocols for botnet C&C communication [9]. As shown in Figure 1.2, in P2P protocols, bot computers typically act as both a client and a server (i.e. hosts send requests for information and also respond to requests for information) to communicate with peer machines so that no centralized C&C channel exists. This means that a botmaster can send commands or updates to the botnet via any of the infected bots and, through these bots the new commands can be propagated through the network, resulting in a botnet much

harder to address. As discussed in [3, 10–12], a number of recent botnets use P2P protocols for C&C.

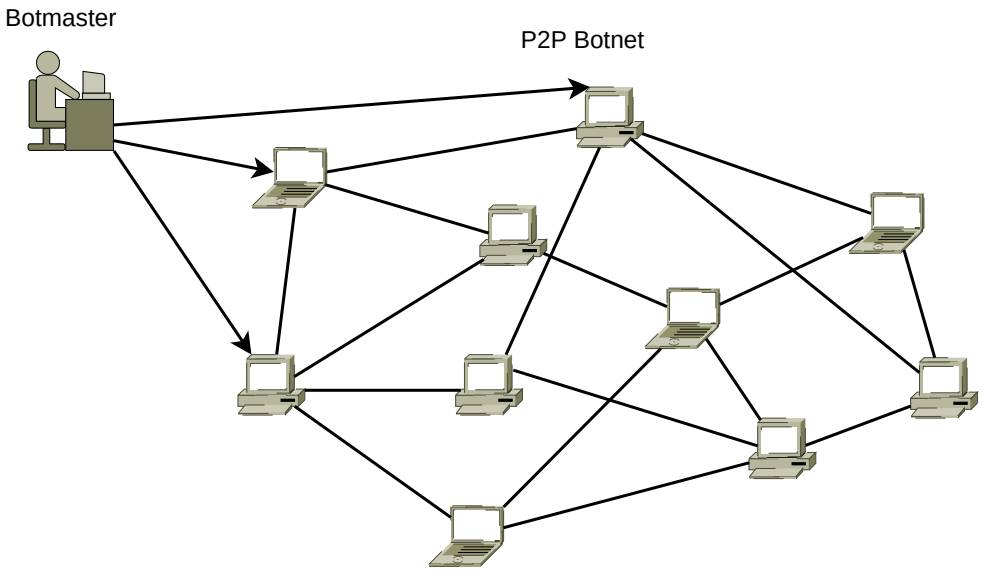


Figure 1.2: P2P botnet architecture

1.2.1 P2P Overlay Networks for Botnet C&C

The term overlay network is used to describe a network which is built on top of another existing network [13]. P2P protocols provide a convenient mechanism for creating overlays on top of existing networks such as the Internet, as illustrated in Figure 1.3. Host machines connected to the Internet that participate in a P2P network (malicious or otherwise) are part of an overlay network.

There are a number of characteristics of overlay networks that should be be defined before proceeding further. In this work, *robustness* will refer to the ability of the overlay network to recover from random failures, or hosts leaving the overlay network. The connectivity of the overlay network in terms of how many connections to other hosts in the overlay network are present, will be described by the term *diffuseness*. For example, in a highly diffuse overlay network, hosts will be loosely connected, meaning that the average number of connections to other hosts will be low. *Resilience* will describe the ability of the overlay network to recover from deliberate attempts by the defenders or defensive community to disrupt or take down the overlay network, such

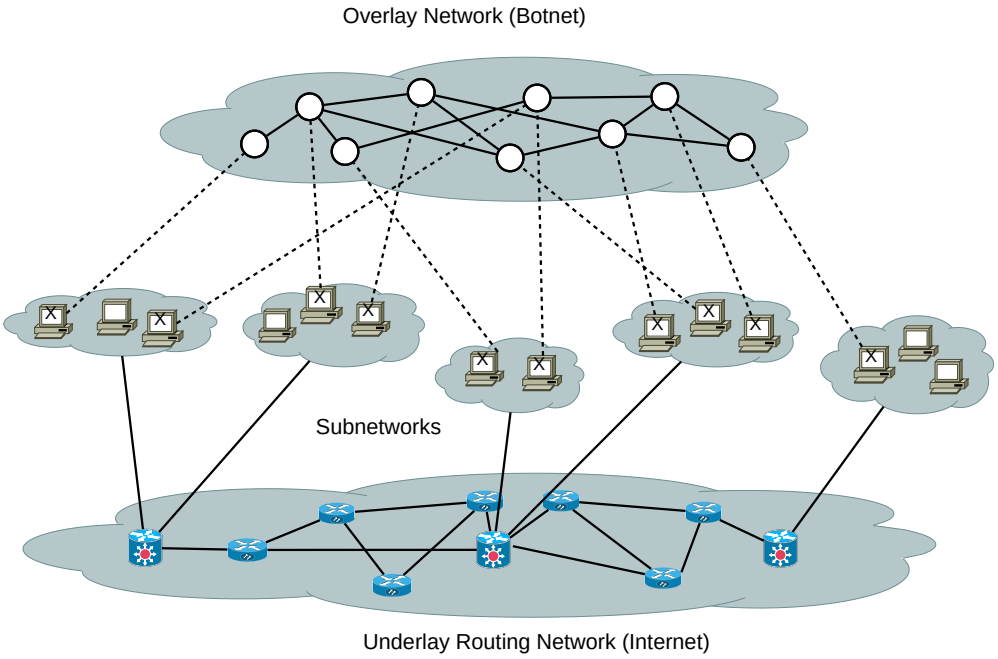


Figure 1.3: P2P overlay network

as a large number of hosts being removed. These definitions are in accordance with the terminology used in [14].

While relying on P2P communication protocols generally results in a fairly robust and diffuse overlay network, it is worth noting that not all P2P protocols are suitable for botnet C&C. Protocols such as Gnutella [15], which have been successfully used in P2P file sharing networks, are not as well suited to botnet C&C as they produce an overlay network structure with a few highly connected nodes which process much of the overlay network traffic. This type of overlay follows a Barabási-Albert network model [16] which is quite robust, but less resilient than an Erdös-Rényi model [17]. These properties are less than ideal for botnet C&C, as communications can be significantly disrupted by targeted disinfection strategies (i.e. the informed removal of overlay nodes) [14, 18].

Other P2P protocols such as Kademlia [19], produce more diffuse overlay network structures and result in more resilient overlay networks. The Storm botnet and it's variants are examples which used Kademlia as a base for P2P C&C communications [3, 9]. The diffuse and highly resilient nature of these overlay networks makes them much more difficult to detect and mitigate than the early centralized botnets.

1.3 Approaches to Botnet Detection

There is strong interest in mitigating the threat posed by botnets. A logical first step in botnet mitigation is botnet detection and, as such, a number of different approaches have have been considered. Namely: i) host-based detection, ii) network-level detection, and iii) graph theory based detection. This section describes the main focus of these approaches and discusses recent works in these areas.

1.3.1 Host-Based Detection

A seemingly straightforward approach is to detect malware running on host computers that participate in botnets (botcode) and remove it. However, as Symantec reported 403 million new variants of malware in 2011 [20], a 41% increase over the number of unique variants reported in 2010 [20], this approach is more challenging than it might appear. Contributing factors to the vast number of unique malware variants are: i) the increased use of advanced techniques such as polymorphism, which vary the internal structure of the the malware while its overall function remains the same [20], and ii) the ease with which attack toolkits supporting a robust set of features for creating new malware and setting up attacks are available in the underground cybercrime marketplace [20]. This complicates the matter of malware detection. A common detection approach known as misuse detection relies on pattern matching techniques and, in the worst case, would require a signature to match against for each new malware variant [21, 22].

Another approach to malware detection is known as anomaly detection, and is based on identifying deviations from normal behaviour. This approach can be more flexible than misuse detection as it is capable of detecting malicious behaviour that falls outside the profile of normal use, however, there is an inherent assumption that a profile of normal system behaviour can be developed that accurately reflects system use [21, 22]. This is especially problematic in non-trivial environments involving rich ecosystems of user behaviours, machine operating systems, and applications, where the normal behaviour of the system or role of a particular user may evolve over time.

Compounding the problem of anomaly detection, is the constant evolution of emerging malware threats. Thus, both the profile of normal behaviour for the de-

fended system, as well as set of incoming malware threats to defend against are likely to change over time. The problem of anomaly detection is complex, even in the case where the normal system behaviour remains consistent, thus it is worthwhile to to consider other approaches to botnet defense.

1.3.2 Network-level Detection

A second approach to defending against botnets is to track malicious activities associated with botnets and identify compromised machines based on this. This type of detection is typically largely based on network traffic monitoring, and intrusion detection and intrusion prevention systems (IDS and IPS, respectively) have been developed for this purpose [22], in addition to the considerable body of research investigating this approach.

Snort is one example of a widely used intrusion detection and prevention tool [23, 24]. Snort was used as a basis for the BotHunter tool developed in [8]. One of the major issues with systems such as Snort, is that they generate a vast number of alerts when operating on any reasonably sized network. This introduces a new problem of how to effectively sort through and respond to all of the generated alerts.

Looking more specifically at recent botnet research, a number of works have addressed network level botnet detection. The previously mentioned BotHunter [8] tool used Snort alerts to trigger correlation between additional information sources and gain a clearer understanding of network events which have triggered alerts. A different approach is used in [25], where the authors used flow based network traffic (Netflow) and DNS Metadata to look for hosts which are generating spam emails, working with the assumption that many of the spamming hosts are likely to be participating in a botnet. The work presented in [26] is also Netflow based and develops a system to passively classify behaviour of network hosts based on connection patterns.

Unfortunately, botnet activity can be highly varied and may be similar to normal network traffic, making botnet detection an arduous task [27]. While many different network level approaches have been developed for botnet defense, they tend to focus on a small subset of behaviours characteristic of a particular botnet. Malware authors and botnet operators have a strong motivation to avoid detection, and may adapt the

parameters of their botnet such that it's activities are stealthy and difficult to detect, or evolve over time [27]. In part, this work seeks to investigate the dynamic nature of botnet behaviour under changing operational parameters.

1.3.3 Graph Theory Based Detection

A third approach is to study the C&C structures used in botnets. While approaches to botnet C&C may vary across different protocols and botnet architectures, understanding botnet C&C provides insight into the overall structure and operation of the botnet, which can be used to develop effective ways to disrupt or disable the botnet [3, 4, 14]. By viewing hosts in the botnet as nodes, and connections between bots as edges, the botnet forms a random graph which can be described and studied using graph theory terminology. In addition, using this abstraction of the botnet overlay allows structural properties of botnet graphs to be quantitatively assessed via graph theory measures.

Focusing specifically on detecting botnet C&C communication of stealthy botnets, [28] used flow clustering and statistical fingerprints to identify hosts running P2P applications and determine which of these hosts are likely botnet participants. Nagaraja et al. [29] proposed methods for Internet Service Providers (ISPs) to detect P2P C&C structures based on conducting random walks of traffic to identify P2P topology subgraphs with fast mixing rates. This work is related to [30], which suggests that detection at a higher level (such as at the ISP) is more effective than using local approaches for P2P botnet detection. An additional graph analysis based detection tool is presented in [31] that uses linkage analysis and clustering techniques in conjunction with the PageRank algorithm to detect P2P communications. This work is extended to utilize MapReduce and cloud computing resources in [32].

1.4 Approaches to Botnet Analysis and Defense

Beyond botnet detection, considerable research efforts have been devoted to botnet analysis and defense. These works seek to investigate structural properties of botnets, evaluate defensive strategies, and predict new trends in future botnets. To this end,

three fundamental approaches have been employed in the study of botnets. Namely: i) graph theory based approaches, ii) simulation based approaches, and iii) in-the-wild observational studies. This section describes the main focus for each of these approaches and discusses prominent research works.

1.4.1 Graph Theory Based Approaches

As illustrated in Figure 1.3, the communication graph created by P2P overlay networks forms a random graph that can be studied using the extensive body of knowledge available in the field of graph theory. Much of the research in this area studies graph structures of the P2P overlay networks [14, 18], which may be compared to theoretical graph models such as Barabási-Albert [16] or Erdös-Rényi [17]. Measures may be used to quantify properties of the overlay network related to shortest paths in the graph, connectivity between different hosts in the graph, as well diffuseness, resilience and robustness as previously mentioned. Some works focusing specifically on graph theory approaches to botnet defense have been previously described in Section 1.3.3.

In [18], the authors identify likely structural forms for botnets, and propose measures for botnet effectiveness, efficiency and robustness. Using Barabási-Albert or Erdös-Rényi models, it was found that the Erdös-Rényi random graphs are highly resilient, and that the Barabási-Albert random graphs are quite robust to random removals, but are susceptible to targeted attacks. The previous work is extended in [14] and it was found that structured P2P protocols such as Overnet (a Kademlia based protocol) can achieve more resilience against targeted attacks than Erdös-Rényi random graphs, which are known to be highly resilient. In related works, [33, 34], Davis et al. looked at mitigation techniques using graph theory based approaches.

While these works provide valuable insights about structural properties of the overlay networks, they innately abstract out the details of the underlying routing network such as number of hops traveled, timing delays, and link capacity, which can influence properties such as response time and availability of hosts in real networks.

1.4.2 Simulation Based Approaches

Simulation is a complementary approach to botnet research that provides a powerful tool for studying botnet behaviour under controlled conditions. The ability to run controlled experiments enables researchers to isolate different aspects of botnet behaviour, and study them under varying conditions. Additionally, large numbers of experiments may be run to gain insight into the statistical nature of botnet behaviour.

A multitude of different simulation models have been developed to gain a better understanding of botnet behaviour and evaluate the effectiveness of potential botnet defense and disinfection strategies. In contrast to real-world network defense, where if the initial defensive efforts are not completely successful the botnet operator may be able to regain control of the botnet [35], simulation can be used to evaluate multiple defensive strategies against a particular network to evaluate the effectiveness of different approaches [33, 34].

Simulation has also been used to evaluate potential architectures and protocols for future botnets, such as in [36]. It must be noted that the study of future botnets is not intended to assist botnet developers, but rather to aid in preparing the defensive community should such developments occur. Within simulation based study, three main modeling approaches that have been used to improve understanding of botnets: epidemiological models, game theory models, and network level models. It is worth noting that in some cases, multiple modeling approaches may be combined to produce a richer model.

1.4.2.1 Epidemiological Models

Compartmental epidemiological models derived from the study of disease spread have been used to model botnet propagation and lifecyle. In these models, the botnet under study is considered the disease, and the network it spreads on (the Internet) is the population among which the disease spreads. The model compartments are used to classify the state of host machines in the network as Susceptible, Infected, or Recovered (in a SIR model). Dagon et al. use an SIR model and an SIR-based diurnal model to investigate how time zones and geographic location impact botnet propagation [37]. The work of [38] also incorporates epidemiological modeling and

uses a SIRS model where hosts may be re-infected after recovery, though this work primarily focuses on game theory modeling so it is described below.

1.4.2.2 Game Theory Models

Game theory models have been used to model strategy decisions in botnet behaviour and botnet defense in recent works. As mentioned above, Song et al. [38] couple a population dynamics model with an evolutionary game model to study interactions between botnets. This work explored the possibility for botnets to increase their survivability using co-operative approaches (i.e. allowing an infected host to become infected by another separate botnet), and found that botnets could increase their chances of survival by allowing multiple infections on the same host machines. A separate work looked at interaction and decision making between botnet operators and network defenders and used game theory models to evaluate optimal operational strategies based on minimizing the cost for each party involved [39].

1.4.2.3 Network Level Models

The authors of [40] develop a model to study factors impacting botnet growth for a network that is loosely based on the Storm P2P botnet. The work of Davis et al. described above [14] is also simulation based, but uses graph theory models and techniques for analysis. In [41], the authors develop a simulation based botnet testbed to evaluate monitoring and mitigation strategies for P2P botnets using graph theory based measures. This work is extended in [42], where the authors extend the botnet testbed and use it to evaluate infrastructure level detection techniques.

1.4.3 Observational Studies

Observational approaches to botnet research have been widely used, but tend to focus on reverse engineering botnet malware, observing in-the-wild botnet behaviour and botnet measurement. Botnet measurement is particularly interesting, since a vast range of results have been reported for studies on the same botnet [3, 43, 44]. For example, consider the Kademlia based P2P botnet Storm. In November 2007, [43]

estimated the size of the botnet as 230,000 hosts for data collected over a 24-hour period of activity, however in [44] it was estimated that the botnet consisted of 1-2 million hosts. Interestingly, in the measurement studies performed by the authors of [3], are again different, and the authors note that while crawling the botnet network to obtain measurements, they were able to observe the activities of other researchers, presumably performing similar measurement studies.

In [45] the authors argue that it is necessary to quantify any measurement results presented, to provide context for how these results were obtained and caveats of the measurement approaches. One obvious caveat of conducting measurement studies of large scale botnets is that botnets of significant notoriety will almost certainly be of interest to other researchers and there is no way to ensure that measurements studies will not contain artifacts such as inflated numbers due to the research activities of others.

1.4.4 Limitations of Prior Works

While valuable information can be gained about malware behaviour and botnet activities, there are considerable limitations to measuring or quantifying behaviours of a botnet in the wild. In addition to the issues described above, it is relatively straightforward for the botnet operator to change tunable parameters of the botnet, such as increasing or decreasing how often bots receive commands or updates. While it may be possible to observe different behaviours resulting from these changes, using observational studies it is not possible to determine the underlying cause of these behavioural changes or infer how botmasters may utilize such changes to avoid detection.

A common approach in botnet research and defensive efforts has been to focus on behaviours of a specific instance of a particular botnet. This innately assumes that one instance of a botnet is representative of general behaviour and, to the best of the author's knowledge, there are no works available which seek to evaluate whether or not this assumption is likely to hold.

1.5 Thesis Goals

The goal of this work is to investigate and quantify, in a statistically rigorous manner, the range of behaviours displayed by P2P botnets to assess how generalizable botnet results may be. Formal definitions of stationarity and ergodicity are provided in Chapter 2. In other words, this work seeks to whether measured botnet features can reasonably be modeled as stationary and ergodic processes.

This is important as many existing P2P botnet defensive strategies and countermeasures simply presume stationarity and ergodicity hold. If they do not and, particularly, if the expression of non-stationary and non-ergodic behaviours is under the botmaster's control, then defending against P2P botnets would become a substantially harder engineering challenge. This work proceeds via packet level network simulation for two main reasons: 1) many instances of the same botnet with varying initial conditions (i.e. Monte-Carlo experiments) are needed to ensure statistical rigour, and 2) simulation provides strong support for experiment control and repeatability.

This work seeks to investigate how botnet behaviour changes over time, and how initial conditions and manipulation of tunable parameters may or may not impact the simulation results. A primary motivation for this work stems from the fact that a large portion of recent botnet research innately assumes that a single botnet instance is representative of general behaviour. Hence, this work seeks to evaluate whether assumptions of stationarity and ergodicity are likely to hold by studying the impact on botnet behaviour of:

1. Conducting a set of Monte-Carlo botnet experiments.
2. Changing tunable parameters of the botnet overlay network.
3. Changing the size and configuration of the underlay network.

1.6 Outline

The remainder of this work is organized as follows:

- **Chapter 2** discusses existing tools and statistical background and provides an outline of how simulations will be conducted.
- **Chapter 3** contains a detailed description of the botnet simulator and statistical testing framework used in this work.
- **Chapter 4** covers experimental setup and assumptions made, the generated results and their analysis.
- **Chapter 5** provides a discussion of results and summary of contributions, and concludes with suggested directions for future research.

Chapter 2

Existing Tools and Statistical Background

In order to assess the statistical nature of botnet behaviours, it is necessary to consider both how statistical behaviours change over time, (stationarity) and across an ensemble of botnet processes (ergodicity). The observational research approaches described in Chapter 1 are not suitable as they are typically restricted to a single in-the-wild botnet instance, thus the ergodicity of botnet features cannot be evaluated. Additionally, the initial conditions of the botnet may impact the nature of the observed behaviours and observational approaches lack the control and repeatability tenets of the scientific method.

In contrast to observational approaches, simulation based approaches support observation of behavioural changes over time (within a simulation run), and across multiple instances of a botnet process (i.e. running multiple simulations with varying initial conditions), allowing the stationarity and ergodicity of botnet features to be evaluated. In addition, simulation provides strong support for the scientific method's control and repeatability tenets. For these reasons, the remainder of this work will rely on simulation.

Within simulation based study, a commonly used technique is Monte Carlo experiments. This involves repeating the same simulation configuration, with varying initial conditions (i.e. different random seeds), and averaging over the results. To ensure meaningful results, it is important to verify that the measured features follow

the same underlying distribution prior to applying this averaging. It can be uninformative and misleading to average results which are drawn from different underlying distributions. As a trivial example, consider an experiment that is repeated twice generating a Gaussian (Normal) distributed feature with N($\mu_1 = 0$, and $\sigma_1 = 1$) for the first experiment, while in the second experiment, the same feature is distributed according to N($\mu_2 = 6$, and $\sigma_2 = 1$). In this case, averaging across the experiment data produces an averaged mean of $\bar{\mu} = 3$ even though the value $x >= 3$ in the first experiment and $x <= 3$ in the second experiment only occur 0.05% of the time. The innate problem is that the combined data across these experiments is non-ergodic (i.e. the experiment runs generate statistically distinct distributions for different random seeds).

A number of different simulation approaches were discussed in the previous chapter, however this work will focus on simulating botnet C&C communication as this is a defining feature of botnets. This work is partially motivated by Birkhoff's Ergodic Theorem [46,47], which describes requirements of ergodicity and clearly indicates that ergodicity is not a universal property.

2.1 Existing Simulation Tools

As this work proceeds via simulation, appropriate tools must be selected. A number existing tools for network simulation and simulation analysis are available and are discussed below.

2.1.1 Network Simulation Tools

A number of different tools are available including: i) NS-2 [48], ii) PrimeSSF [49], iii) Möbius [50], and iv) OMNeT++ [51]. In selecting a tool for this work, extensibility and scalability are primary selection criteria and will be discussed for each tool below.

2.1.1.1 NS-2

NS-2 is an open source network simulation tool, which provides support for wired and wireless based simulation over a variety of different protocols [48]. Because it is an open source tool, complete source code for the simulator is available. Thus, NS-2 naturally provides good extensibility. In addition, NS-2 provides a large number of tested implementations of network protocols which can be re-used in developed models. Unfortunately, NS-2 suffers from some early design decisions which result in poor performance of large scale models [52] and thus, is of limited use in larger scale simulations. A new tool, NS-3 [53] was developed to resolve the scalability issues of NS-2, and while this tool provides support for a growing number of tested built-in models it is not backwards compatible with NS-2 so development of tested implementations of different protocols and devices is ongoing.

2.1.1.2 PRIME SSF

PRIME SSF is a real-time immersive modeling environment that was developed at Florida International University and provides support for both simulation and emulation. Integrated in its design is support for parallel and distributed computing environments [49]. PRIME SSF was used in [41] to create a P2P botnet testbed for the structural analysis, monitoring and mitigation of Kademlia based botnets. This work was extended in [42] to incorporate infrastructure level details into the model.

Complete source code is available for PRIME SSF, so this project supports extensibility. However, while basic reference information is available, other tools such as NS-2 and OMNeT++ provide a wider range of implemented simulation modules (i.e. protocols and devices), with more guidance for new development.

PRIME SSF is well suited to developing very rich real-time simulations by incorporating real physical devices (emulation), which is a disadvantage in the context of this work. To ensure statistical rigour of results, a lighter weight simulation tool which can produce a large number of simulation runs in a time-practical manner is needed.

2.1.1.3 Möbius

Möbius is a modeling tool developed by researchers at the University of Illinois Urbana-Champaign [50]. It provides support for several modeling formalisms including queueing networks, petri-nets and stochastic activity networks (SANs). This tool was used in [40] to create a SANs model of the creation of a large botnet that was loosely based on the Storm botnet. In this work, the authors tracked the growth of a botnet over the time period of one week and investigated the effectiveness of different anti-malware approaches throughout the botnet growth cycle.

While Möbius provides flexible support for a number of different modeling approaches in addition to support for running in a distributed computing environment, it has several disadvantages. First, Möbius is designed for high level modeling and requires that the simulated process can be described as a series of state transitions, hence it is not suitable for a packet-level botnet model. Further to this, the state-transition modeling approach is based on Markovian models, which are inherently ergodic and thus not suitable for assessing ergodicity. Second, Möbius is only supported on a limited number of platforms: Windows XP/Vista, Mac OS 10.5/10.6, and Ubuntu Linux 8.10/9.04/9.10.

2.1.1.4 OMNeT++

OMNeT++ [51] is an open source simulation engine for creating network simulations. OMNeT++ has been widely used for traditional and non-traditional networking applications including wired and wireless networking research, mobile ad-hoc [54] networks, sensor networks [55], and P2P networks [56]. Complete source code is available for OMNeT++, so this tool also supports extensibility and, similar to NS-2, a large number of tested protocol and device implementations are available for use in developed models. In contrast to NS-2, OMNeT++ provides good support for scalability, thus making it more feasible to run large scale experiments [52].

Oversim is an OMNeT++ based project that provides support for simulating P2P overlay networks [56]. Oversim supports common P2P protocols including Kademlia, and provides some support for configurable underlay networks topologies. One disadvantage of Oversim is that there is currently no support for saving the state of a

network, which means that large networks must be generated from scratch for each simulation run. This removes the possibility of running multiple experiments from a particular saved experiment state, which is necessary for evaluating the stationarity and ergodicity of measured simulation features. At the time development for this work, the Oversim project did not appear to be actively maintained, although a new version has since been released. For these reasons, Oversim is not used in this work.

Although the Oversim project is not suitable for this work, OMNeT++ is a good choice for developing a packet-level botnet as it is scalable and extensible, and provides a wide range of tested implementations for protocols and devices. The remainder of this work will make use of a custom made OMNeT++ packet-level botnet model which is mentioned in section 2.2 and discussed in detail in Chapter 3.

2.1.2 Analysis Tools

Simulation analysis tools have been developed to manage and automate experiment runs, apply statistical tests, and further analyze results. For this work, an OMNeT++ compatible tool is required that provides experiment management over an existing 42 machine cluster of computing resources, and explicit support for stationary and ergodicity testing of measured simulation features. With respect to these criteria, several existing analysis tools: i) SimProcTC , ii) Akaroa and iii) STARS are described below.

2.1.2.1 SimProcTC

The Simulation Process Tool Chain, SimProcTC [57] is an open source tool chain for OMNeT++ based simulations. This tool utilizes GNU R statistical software for parameterization of simulation runs as well as visualization of results, and Reliable Server Pooling (RSerPool) for parallelization of simulation runs. SimProcTC does allow other analysis tools such as Microsoft Excel and GNU Octave to be used, however, the main disadvantage of SimProcTC is that it does not provide direct support for stationarity or ergodicity testing. If multiple simulation runs are present, SimProcTC takes the average of these results without applying statistical hypothesis

tests to determine whether the results follow the same underlying distribution - an explicit requirement of this work.

2.1.2.2 Akaroa

Akaroa [58] is another open source analysis tool which supports parallelization of simulations using Multiple Replications in Parallel (MRIP). Akaroa was designed to run on Unix multiprocessor systems, or networks of Unix workstations, and has been previously integrated with OMNeT++ and NS-2. Using the Akaroa framework, simulations are run in parallel, and results are sent to a management process which computes the estimated mean value collated over all runs. When sufficient data is available to satisfy user-specified confidence level requirements, simulations are stopped. It must be noted that computing confidence intervals requires that the underlying distribution of the measured feature is known. In general, the distribution of measured botnet features is not known in advance, thus the experiment management approach of Akaroa is not suitable for this work. In addition, although Akaroa provides good support for parallelization, statistical testing for stationarity and ergodicity is not directly supported.

2.1.2.3 STARS

The Statistically Rigorous Simulation (STARS) framework [1, 59] was developed to address some of the limitations of the above tools with regard to explicit testing for stationarity and ergodicity. STARS is a parallel MPI-aware network simulation framework which provides automated support for statistically rigorous simulation based research. STARS is compatible with OMNeT++ based models, and supports configuration of Monte-Carlo experiments. Most importantly, STARS includes a distribution-free statistical analysis feedback loop with explicit testing for stationarity and ergodicity of measured features. For these reasons, STARS will be used as the experiment automation and analysis framework in the remainder of this work. A more detailed description of STARS is given in Section 3.3.

2.2 P2P Botnet Simulator

The previously described simulation tools provide support for a variety of useful features in botnet simulation. However, two main issues have largely not been addressed. First, when studying the statistical nature of large scale networks such as botnets it is necessary to be able to generate a large network (e.g. containing tens of thousands of hosts) and then save both the topological and state information of this network such that it may be reloaded for multiple experiments with different initial conditions or run-time parameters. With the exception of PrimeSSF, none of the above tools support this feature. Second, to evaluate the statistical nature of measured botnet features, direct support for stationarity and ergodicity testing is required. As none of the previously described tools provide a stand-alone solution suitable for this work, a custom made OMNeT++ based botnet simulation is used in conjunction with the STARS framework.

The OMNeT++ P2P botnet simulator is based on the well known Storm botnet which relies on the Kademlia P2P protocol for botnet C&C communications. The decision to focus on this particular botnet was made because it is well known and well studied, thus eliminating much of the uncertainty present when studying emergent botnets. In addition, previous research [14, 33] indicates that Kademlia based P2P botnets present an interesting research problem as they are fairly difficult to address. Both the botnet simulation model and the STARS framework are described in more detail in Chapter 3.

2.3 Stationarity and Ergodicity

When assessing statistical behaviour, this work will focus on two central concepts. The first is stationarity, and the second is ergodicity. The importance of these concepts stems from the fact that the simulation tools used in this work produce random data. That is to say, the simulations exist as discrete time random processes, and data produced must typically be analyzed using statistical techniques. Formal definitions of stationarity and ergodicity are provided in the following sections.

2.3.1 Probability Space

The concepts of stationarity and ergodicity rely on definitions from the field of measure theory. While probability and statistics are familiar topics in engineering, the more general field of measure theory may be less familiar. Hence, we begin by providing a definition from [60] of a probability space, first noting that a σ-field is a collection of subsets which is closed under countable unions, and contains the limits of all sequences of sets in the collection:

A probability space (Ω, B, μ) is a triple consisting of a sample space Ω, a σ-field B of measurable subsets of Ω, and a probability measure μ defined on the σ-field. That is, $\mu(A)$ assigns a real number to every member A of B such that the following conditions are satisfied:

Nonnegativity:

$$\mu(A) \geq 0, \text{all } A \in B, \tag{2.1}$$

Normalization:

$$\mu(\Omega) = 1, \tag{2.2}$$

Countable Additivity:

$$\begin{gathered}\text{if } A_i \in B, i = 1, 2, ... \text{ are disjoint, then} \\ \mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)\end{gathered} \tag{2.3}$$

Note that a set function μ which satisfies (2.1) and (2.3) but not necessarily (2.2) is called a *measure*, and the triple (Ω, B, μ) is called a *measure space*; in this work, only finite measures are considered. Further note that, for the purposes of this work, the terms measure and probability measure are used interchangeably; definitions are written using the more general measure notation in order to remain consistent with the reference material in [46, 47, 60, 61]. Additional details and more rigorous definitions

are provided in [60]. It is worth noting that a more familiar definition of probability may be written by replacing the symbol μ with P and defining a mapping from the sample space Ω to the interval $[0, 1]$ as in (2.4).

$$P : \Omega \rightarrow [0, 1] \tag{2.4}$$

From the definitions above, descriptions of random variables, and random processes may be developed. The text by Peebles [62] provides an excellent entry-level description of these concepts, while [60] approaches the material in greater depth from the perspective of measure theory and dynamical systems.

2.3.2 Dynamical System Representation of Random Processes

While Peebles [62] describes a random process as a family of random variables defined on a common probability space, dynamical systems offer a description which instead considers a single random variable together with a transformation defined on the underlying probability space. Resulting from this, outputs of the random process will be values of the random variable taken on transformed points in the original space [60].

A few additional words of explanation are offered prior to proceeding with dynamical systems representation. First, it is noted that a measurable function $f : \Omega \rightarrow \Re$ represents an *observable* of the system, that is a quantity that can be measured. The value of the observable $f(\omega)$ is the measurement of the observable f obtained when the system is in state ω. The notion of state often refers to the time-evolution of the system, but other definitions of state are possible. In this work, we are interested in both the time-evolution and ensemble evolution of simulation results. A transformation which represents the system evolution can be defined as $T : \Omega \rightarrow \Omega$. If $\omega \in \Omega$ is the initial state, $T(\omega)$ is state of the system after one state transition, and $f(T(\omega))$ is the value of the observable at state $T(\omega)$.

A dynamical system is a probability space (Ω, B, μ) together with a measurable transform $T : \Omega \rightarrow \Omega$. This definition can be written as the as a quadruple (Ω, B, μ, T), and this quadruple is known in the field of ergodic theory as a dynamical system.

It is worthwhile to note that measurability means that if $A \in B$ then $T^{-1}(A) = \{\omega : T(\omega) \in A\} \in B$, where $T^{-1}(A)$ is called the pre-image of A. An illustration is provided in Figure 2.1.

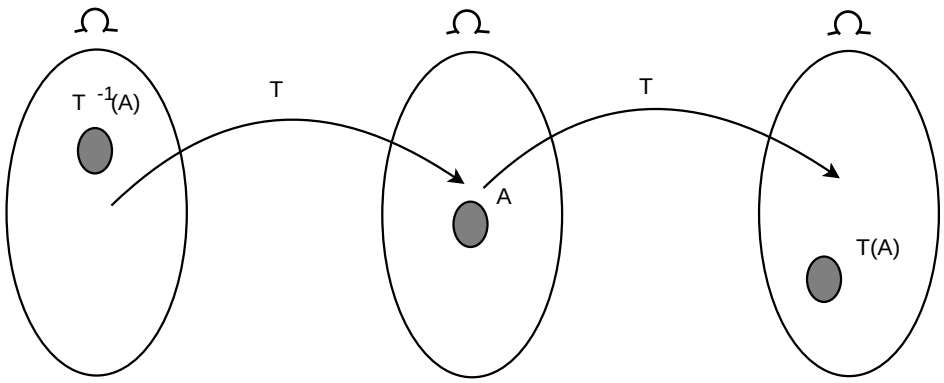


Figure 2.1: Dynamical systems random process representation

Let us return to our transformation T. Suppose that T is a one-to-one measurable transform mapping the points in the sample space Ω onto itself, and note that the composition of measurable functions is also measurable. Thus, the transformation T^n defined by $T^2 = T(T(\omega))$ and so on $T^n = T(T^{n-1}(\omega))$ is a measurable function for all integers n.

2.3.3 Measure Preserving Transformations

Next, recall the previously defined measure μ, which satisfies (2.1) and (2.3) but not necessarily (2.2). Now suppose we have our transformation $T : \Omega \to \Omega$. Using the definitions above, and those provided in [46, 61], we can say that T is *measure preserving* if (2.5) holds:

$$\mu(T^{-1}(A)) = \mu(A) \text{ for every measurable set A} \in B \tag{2.5}$$

2.3.4 Stationarity

Using the definitions developed above, and those provided in [60, 61] conditions for stationarity may now be written as:

1. Condition (2.5) is satisfied.
2. The transform $T : \Omega \to \Omega$ is one-to-one and onto
3. The transform $T : \Omega \to \Omega$ is a time shift.

2.3.5 Ergodicity

Again relying on above definitions and reference material from [46,47,60,61] conditions for ergodicity may be written as:

1. $T : \Omega \to \Omega$ is measure preserving as per (2.5)
2. The transform $T : \Omega \to \Omega$ is one-to-one and onto
3. Subsets A with $T^{-1}(A) = A$ satisfy $\mu(A) = 0 \ : or \ : 1$.

Further to this, we restrict our definitions of ergodicity in this work in accordance with Birkhoff's Ergodic Theorem [46, 47, 60] as provided below:

Theorem 1. *Let (Ω, B, μ) be a probability space, $T : \Omega \to \Omega$ a measure preserving ergodic transformation, and $f : \Omega \to \Re$ a real-valued function. Then:*

$$\lim_{N\to\infty} \sum_{N=1}^{n} f(T^n(x)) = \int f d\mu \ for \ \mu \ \text{-} \ almost \ every \ \omega \in \Omega \tag{2.6}$$

Using Birkhoff's theorem, we impose the constraint of stationarity as a pre-requisite to ergodicity. Hence, for the purposes of this work, non-stationary will imply non-ergodic.

2.3.6 Analysis of Measured Features

For the purposes of this work, events of interested are $A : x < X$, the empirical CDF of measured botnet features. That is to say we assess stationarity and ergodicity of measured botnet features by studying the empirical CDFs of these features. Additional details of the statistical testing process used in this work are provided in Chapter 3.

2.4 Contributions

To the best of the author's knowledge, previous research has not formally assessed the validity of ergodicity assumptions for measured botnet features. This work makes use of an OMNeT++ packet level Kademlia based botnet simulator and the STARS experiment automation and analysis framework to study the statistical nature of botnet behaviour. The stationarity and ergodicity of measured botnet features is assessed under the following scenarios:

1. Conducting a set of Monte-Carlo botnet experiments
2. Changing tunable parameters of the botnet overlay network
3. Changing the size and configuration of the underlay network

2.5 Summary

Past botnet research has looked at botnet C&C traffic primarily by studying the overlay network traffic, however the statistical nature of this traffic has not been formally studied. While observational approaches are not suitable for this type of research, simulation offers a controlled environment under which a large number of experiments may be run to produce substantial data, on which statistical analysis may be run. Multiple tools are available for this purpose, OMNeT++ was selected because it provides a flexible and scalable environment with tested implementations of protocols and devices available for re-use in new models. To ensure statistical rigour of simulation results, the OMNeT++ model is integrated with the STARS automation and analysis framework, which includes a statistical analysis feedback loop, and explicit testing for stationarity and ergodicity.

Chapter 3

P2P Botnet Simulator

This chapter describes the OMNeT++ based P2P botnet simulator used in this work, as well as the STARS statistical testing framework used to produce statistically rigorous results. The P2P botnet simulator includes underlay network infrastructure and an overlay network which runs Kademlia P2P overlay protocol for botnet C&C.

3.1 OMNeT++ Simulator

OMNeT++ is an open source discrete event network simulation tool that has been widely used for simulating wired and wireless communications, and other networking applications. OMNeT++ is object-oriented, hierarchical and modular, which enables high code re-use and simplifies the creation of new models.

Simulation models are comprised of multiple interconnected modules, and these modules may be either simple or compound. Modules are written in C++ programming language and provide implementation of active behaviours such as specific algorithms used the simulation. Simple modules are used at the lowest level of module hierarchy, whereas compound modules are comprised of multiple simple modules. Modules are interconnected via gates to form a network, and communications are achieved by sending messages between interconnected modules. Connection properties and configurations are described using the OMNeT++ Network Definition (NED) language to specify static and default model settings. Additional

configurations files are used to specify parameters of individual experiment runs. OMNeT++ supports exact replications and independent repetitions of simulations via the Mersenne Twister [63] pseudo-random number generator (PRNG).

OMNeT++ provides an eclipse-based GUI, and a command line environment for running simulations. The GUI is primarily used for debugging purposes, while the command line environment is preferable for batch execution of simulations. In this work, the command line environment was used so that simulations could be pushed to a cluster of servers running the STARS framework for batch execution.

A number of OMNeT++ component libraries are available which provide support for specific networking applications. Of particular note is the INET Framework [64], an OMNeT++ based simulation package which provides support for standard wired and wireless networking protocols. INET is used in this work to ensure correct implementation of protocols and devices in the underlying network.

3.2 P2P Botnet Simulator

The P2P botnet simulator used in this work is OMNeT++ based and provides a packet-level simulation covering both underlay and overlay network behaviour. The underlay network leverages modules from the INET component library pertaining to standard devices and protocols. Additionally, data from CAIDA's Skitter project [65] is used for generating an autonomous system (AS) level underlay network topology representative of real-world infrastructure, and a set of subnets which act as Internet Service Providers (ISPs), each being connected to one of the hosts in the underlay routing network topology. The overlay network relies on the Kademlia protocol for botnet C&C.

There are two modes of operation for the P2P botnet simulator, *Generation* and *Steady-state*. In generation mode, the botnet is grown to the desired size, at which point the simulation is stopped and topology and state information are archived for use in subsequent steady state simulations. In steady-state mode, an archived botnet topology is loaded and botnet birth and death processes are configured to maintain a steady state expected botnet size. All statistics from the botnet's operation are collected during steady-state operation.

3.2.1 Model Architecture

The underlay network consists of AS-level network topology and a set of subnets representing ISPs, as shown in Figure 3.1. Subnets are characterized by:

- maxBots: The maximum number of bots in a modeled subnet
- tBotCreation: The distribution describing the inter-arrival time of new bots in the subnet (i.e. birth rate)
- tbotRemoval: The distribution describing the inter-arrival of bots leaving the subnet (i.e. death rate)

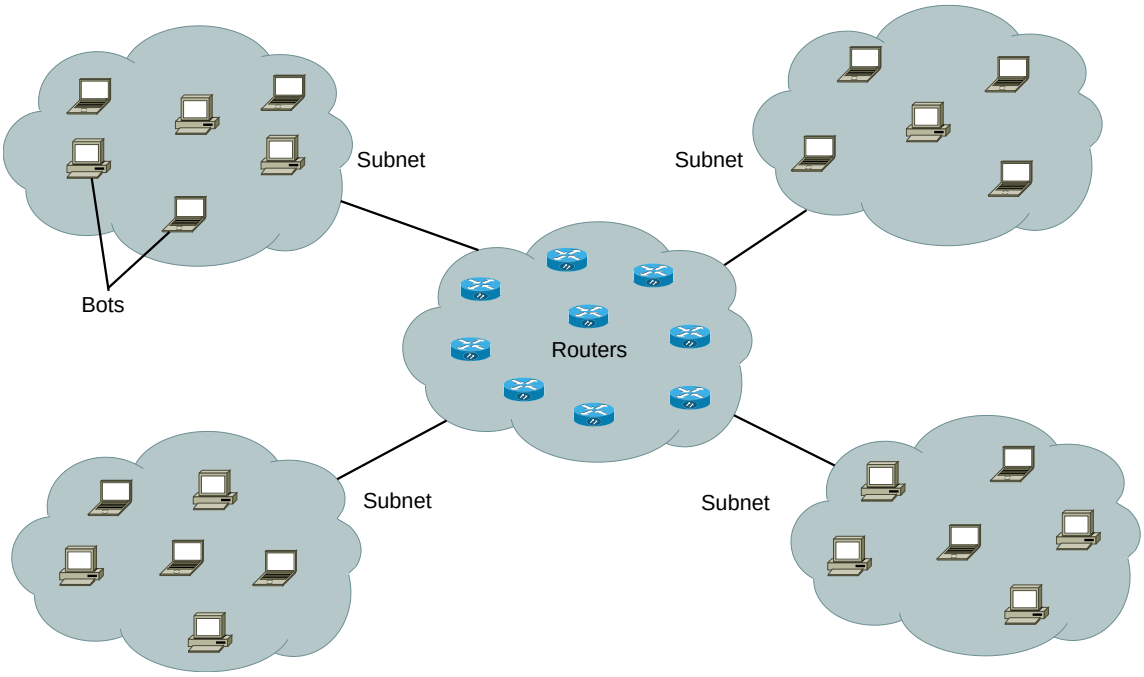


Figure 3.1: Underlay network

Two separate underlay configurations are considered in this work. The first consists of a small test underlay containing 20 routers and 10 subnetworks. The second is a larger topology based on a subset of CAIDA AS-adjacency data from the Skitter project. The Skitter project produces AS adjacencies for the global Internet, however incorporating the full dataset is prohibitively expensive in terms of memory requirements and compute time, hence a subset topology of this topology is used. The

topology subset is created by selecting several mid-size ASes and a subsampling of their peers and clients to form a topology of approximately 2000 routers. With this routing topology, 100 subnetworks are connected to the routers to form an underlay routing network similar to that of a corporation or government agency.

Each modeled subnet contains a networking module which is responsible for modeling transport-level behaviour and uses a message mapping module which is responsible for simulating intra-subnet routing and delays. Static routing is used for scalability reasons, and the underlay network consists only of wired hosts (i.e. no wireless hosts are included). Traffic in the network is sent via the connectionless User Datagram Protocol (UDP) protocol, to minimize complexity and ensure that statistical events of interest are not simply an artifact of connection oriented protocols or congestion control methods present in Transmission Control Protocol (TCP) traffic.

Kademlia P2P overlay protocol is modeled via a set of bot modules, each of which runs the Kademlia P2P overlay protocol. Attributes of the Kademlia protocol are given in Table 3.1.

Table 3.1: Kademlia protocol attributes

Attribute	**Description**
`k`	The maximum number of contacts stored in each k-bucket [Default: $k = 20$].
`alpha`	The degree of parallelism for iterative lookups [Default: $\alpha = 3$].
`keyLength`	The length, in bits, of the host IDs used to identify bots and of the keys used to identify values [Default: keyLength = 128].
`peerListSize`	The number of peers in the initial peer list sent to each newly created bot [Default: peerListSize = 200]
`tRefresh`	Interval between when a bot refreshes the contents of its k-buckets [Default: Refresh = 3600 seconds].
`tValueLookup`	Interval between when a bot attempts to look-up a value it does not have [Default: tValueLookup = 3600 seconds].
`tReplicate`	Interval between when a bot republishes every key,value pair it has found [Default: tReplicate = 3600 seconds].
`maxStorageLocations`	The number of locations in the botnet where each key,value pair is initially stored. [Default: maxStorageLocations = 10]

Within the simulation, several modules are used to implement the Kademlia P2P overlay, as illustrated in Figure 3.2. The Subnet module contains modules which implement the operational mechanics of each subnetwork. This covers both the overlay and the underlay. Specific to the overlay network, the Overlay Controller module is responsible for generating { key, value } pairs which will be searched, and monitoring the size of the botnet such that signals can be sent to archive the simulation when a botnet of sufficient size is generated. The Subnet Controller module manages and maintains each subnet, this includes creating and removing bots as per the specified birth and death rate processes keeping a list of active bots, and generating bootstrap messages for newly created bots. The P2P Host module contains the submodules which form the operational mechanics of an individual bot, namely implementation of the Kademlia protocol.

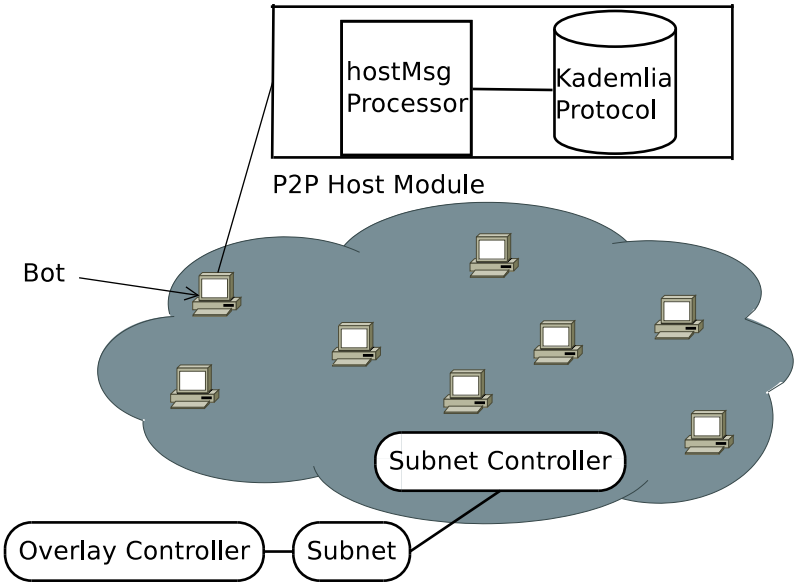


Figure 3.2: Overlay modules

3.2.2 Kademlia Overview

The overlay network protocol used for botnet C&C in this simulation is based on Kademlia [19]. An overview of the Kademlia protocol based on that of [66] is provided below. Kademlia is a Distributed Hash Table (DHT) based P2P network overlay

protocol that was not constructed to support botnets, but has been effectively used for C&C in botnets including Storm [3].

In order to form a botnet, the Kademlia protocol must be active within every bot that is part of the botnet. Host infection mechanisms are outside the scope of this work, so it is assumed that a sufficient number of hosts have become infected and are running the same Kademlia-based botcode such that a botnet that is connected in the graph-theory sense exists and is active.

The Kademlia protocol supports four main types of P2P messages which are sent between bots:

- **PING** is used to probe whether a particular peer is online
- **STORE** instructs a peer to store a key,value pair into the botnet for later retrieval
- **FINDNODE** takes a key as its argument and returns a <IP Address, UDP Port, Host ID >triplet for the peers that are closest to the sought host ID
- **FINDVALUE** is similar to the FINDNODE message, except in the case when a peer receives a STORE message for the key, it just returns the stored value.

All messages within Kademlia are routed using the botnet's P2P overlay. Hence, the IP addresses of infected hosts are not used for within-Kademlia routing operations. Instead, when new bots first join the botnet, they generate a random host ID which is used to identify that host to the botnet. The length of this ID is specified by the *keyLenth* attribute of Kademlia, and for the purposes of this work, the default key length of 128 bits is used.

The central feature of Kademlia is the management of peer-lists that are held within each bot. These peer-lists provide the mechanism by which message routing occurs within Kademlia. The size of the peer-list is specified by *peerListSize*, and is generally in the hundreds. When bots first join the botnet, they are given an initial seed list of botnet peers which they may contact to begin receiving commands and participating in the botnet. This process is known as bootstrapping.

Within Kademlia, peer-lists are not stored as a single continuous list. Instead, they are subdivided into sub-lists called k-buckets, where each k-bucket contains a

maximum of k entries, and the Kademlia attribute k is generally set to k= 20. All P2P messages contain the sending bot's 128-bit host ID, and the k-buckets for each host are updated in accordance with the XOR distance [19] between the sending and receiving bots host IDs. Hence, the 1-bucket contains contact information for active bots whose host ID has only a 1-bit difference from the receiving bot's host ID. The 2-bucket contains IDs for active bots with a difference of 2-bits, and so on. It is worth noting that when using a 128-bit ID length, it is highly likely that only the last few k-buckets within any of the bots will be fully used due to the fact that half of the address space will be covered be the last k-bucket, one quarter of the address space will be covered by the *k-1* bucket, and so on, as illustrated in Table 3.2.

Table 3.2: Kademlia address space of k-buckets

***k*-bucket Number**	**Address Space**
127	2^{127}
126	2^{126}
125	2^{125}
...	...
3	2^3
2	2^2
1	2^1
0	2^0

The contents of k-buckets for each bot are not static and will change as botnet operates and bots join or leave the botnet, as illustrated in Figure 3.3. When a message is received by a bot, the receiving bot calculates the k-bucket to which the sending bot belongs, using the XOR metric. The determined k-bucket is then checked to see if there is already an entry for the sending host. If there is already an entry for the sending host, this entry is moved to the top of the k-bucket. If there is no entry present for the sending host and the k-bucket has less than k entries, the sending host's contact information is added to the top of the k-bucket. Otherwise, if the k-bucket is full, a PING message is send to the host at the bottom of the k-bucket. if this bot does not respond, then its entry is dropped from the k-bucket and the new bot's information is added to the top of the k-bucket. If the PINGed host responds, it's entry is moved to the top of the k-bucket and the new host's information is discarded. This ensures that hosts which are active for long durations are kept in the botnet thus reducing churn.

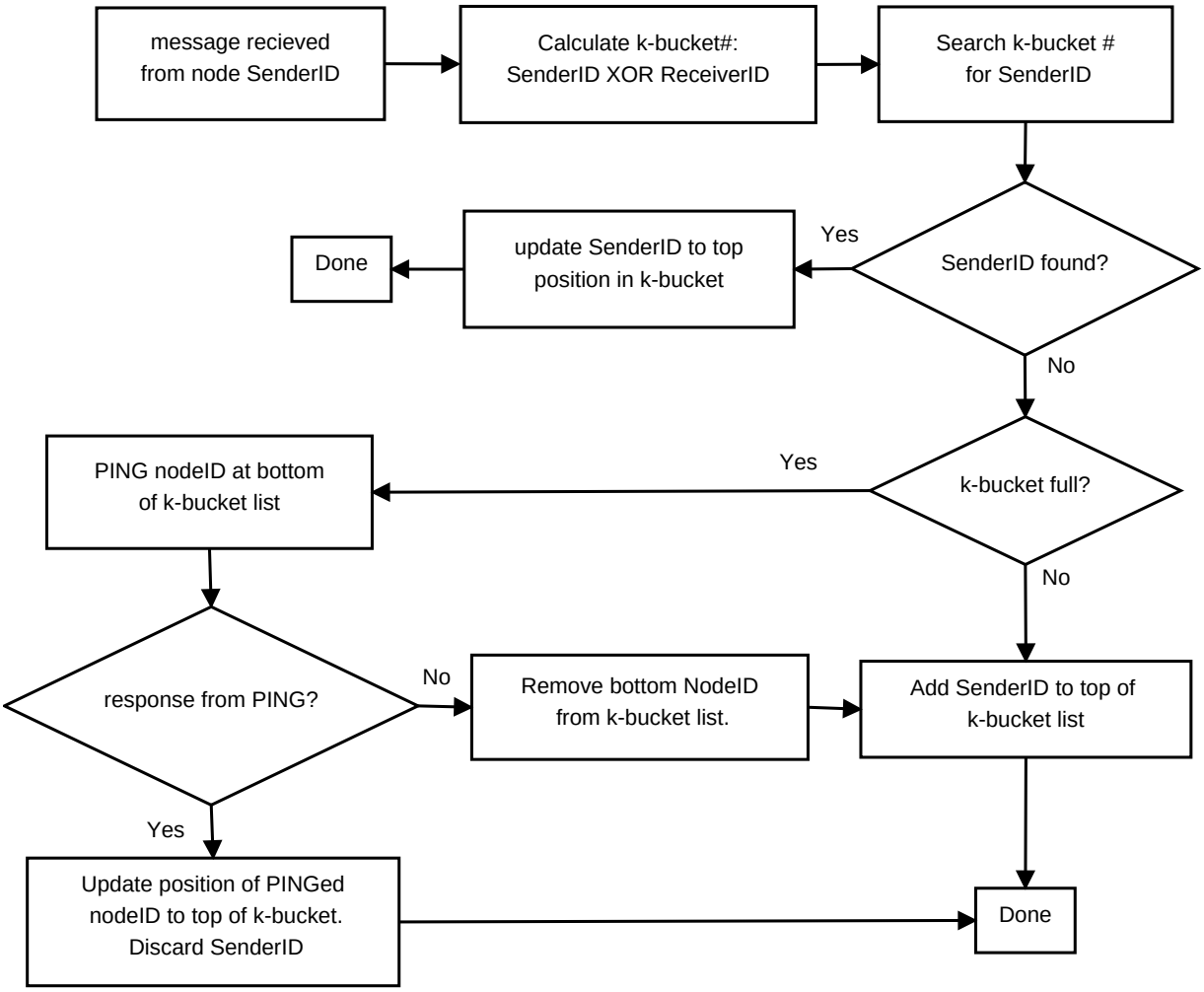


Figure 3.3: k-bucket update process

The primary function of Kademlia is to enable peers to receive answers to { key, value } queries made via the overlay. When a STORE message is sent into the Kademlia network, the { key, value }-pair-argument is stored in *maxStorageLocations* peers, whose IDs are closest to the key, where the key is also a 128-bit random ID. When a Kademlia host receives a FINDVALUE message, it is either one of these *maxStorageLocations*, in which case it responds with the requested value, or it is not. If it is not, then it checks it's k-bucket entries to determine whether it knows one of the *maxStorageLocations* hosts, in which case the FINDVALUE query may be passed directly to the closest host. If none of the *maxStorageLocations* hosts are known then α hosts are randomly selected from the closest k-bucket and the message is forwarded to them. In this work, the default value of $\alpha = 3$ is used.

Although Kademlia was not designed as a mechanism for botnet C&C, it is quite effective at supporting the botnet operator's end goal of recruiting bots to perform specified tasks. Kademlia has previously been used to facilitate pull-structured botnets, which are pre-configured request information at specified intervals. From a defensive perspective, this is distinct from a push-structured botnet where a botmaster sends commands out to the bots. In comparison to push structured botnets, pull structured botnets offer a higher degree of parallelism which in turn can reduce network traffic visibility.

This work simulates a pull structured botnet, where each bot is pre-configured to search for 32 different { key, value } pairs at regular intervals over a 24 hour period. Hence 32 independent botnets are effectively simulated per Monte-Carlo run, as each 32 key is associated with a different peer list. For the purposes of this work, the measured botnet data is assessed from the perspective of a defender, who would not have complete topology information about the peer-lists associated with the different keys, and thus would treat the entire botnet as a single process.

3.3 STARS: Statistical Testing Framework

The Statistically Rigorous Simulation Framework (STARS) [1, 59] is a parallel MPI-aware network simulation framework which provides support for statistically rigorous simulation-based research. STARS is comprised of the following components:

1. Simulation Resource Package
2. Experiment Automation Framework
3. Statistical Analysis Framework

The Simulation resource package is user-supplied, and model specific. In this work, the model used is an OMNeT++ based P2P botnet simulation. The resource package contains all of the information (libraries, model dependencies, etc) to run the simulation. In addition to this, a workfile must be supplied which outlines experiment configuration simulation parameters.

The experiment automation framework is implemented in python, and enables hands-free execution of experiments. In conjunction with the analysis framework described below, the automation framework initiates experiment runs, submits interim results to the analysis feedback loop for statistical testing, and stores results to a central location.

The statistical analysis framework is implemented in MATLAB [67] and tests measured features from the simulation for stationarity and ergodicity. Further details of the testing procedure are provided below.

3.3.1 Statistical Analysis

Within this work, measured botnet features collected over a simulation run take the following form:

$$X = \{< x_k, t_k > | k = 1, ..., K\} \text{ with } t_k > t_{k-1} \tag{3.1}$$

The values for each x_k are governed by an underlying distribution of unknown form, denoted $P(X)$. In addition to this, the measured feature X is further characterized by the time between measurements, $\tau = t_k - t_{k-1}$ with $t_0 = 0$. The sampling time τ is itself a random variable with an unknown distribution.

Due to the random sampling time of measured features, results from different simulation runs (i.e. using a different random seed) cannot be directly compared. One method of solving this problem is to re-sample the measured features using a constant sampling time. However, when the density of events with respect to time is also of interest, re-sampling can only be applied when τ is small compared to the total simulation time.

3.3.1.1 Testing for Stationarity

An alternative approach to re-sampling measured features to enable comparison is to test, using statistical hypothesis tests, the statistical similarity of values assumed by

the measured features over time. In doing this, empirical distributions of measured features can be calculated and their statistical similarity can be assessed. If no evidence is found to suggest that the measured features are statistically different, then the measured features may be compared.

Because the distribution of measured features may be time dependent, and in general not known in advance, it is critical that any statistical tests used do not require assumptions regarding the underlying distribution (e.g. Chi-squared test requires an analytical for of the distribution to test against and is thus not suitable for this work). The STARS framework employs a two sample Kolmogorov-Smirnoff (KS) test [68] for this purpose. This test is used to assess whether two sets of sample data are likely to have been drawn from the same underlying distribution without requiring any expected form of the distribution to be specified. Note that the KS-test does not provide information about the form of the underlying distribution of the tested features, it only indicates that the tested features appear(do not appear) to come from the same unspecified underlying distribution when test passes(fails). The KS-test is given by:

$$D = max[\{P^1(x) - P^2(x)\}|\forall_x] \tag{3.2}$$

Where $P^1(x)$ and $P^2(x)$ are empirical distributions. The test statistic D indicates the largest difference between the tested distributions, and is used to calculate the p-value of the test via the asymptotic Q function. The calculated p-value may then be compared to a user-defined significance level α. If the p-value is found to be greater than the specified value of α, then the test passes, accepting the null-hypothesis that the features are drawn from the same distribution. Otherwise the test fails, rejecting the null-hypothesis. To be clear, passing the KS-test denotes only that no evidence of dissimilarity between two empirical distributions was observed at the specified confidence level α. For the purposes of this work, a standard value of $\alpha = 0.05$ is used.

Statistical tests for stationarity are conducted using a sliding window KS-test as illustrated in Figure 3.4. Before testing begins, the feature X is split into Q windows W_q. The window width is selected based on the maximum time between measurements, and is chosen to ensure that windows contain a minimum number

of measurements; additional details are provided in [1]. After splitting the feature into windows, empirical distributions are calculated for each window, and the KS-test is applied to adjacent windows. This is done in reverse time order, since it is assumed that if transient behaviour is present, it is more likely to appear at the start of the simulation, whereas stationary behaviour is most likely to occur later in the simulation. Note that when the KS-test passes, the empirical distributions are averaged for use in the subsequent tests. That is, the distribution of a window is compared to the average empirical distribution of all previously tested windows. This means that each time the KS-test is passed, a better estimate of the underlying distribution is obtained. Further note that this is distinct from multiple hypothesis testing since $t_{stationary}$ depends only on a single application of the KS-test.

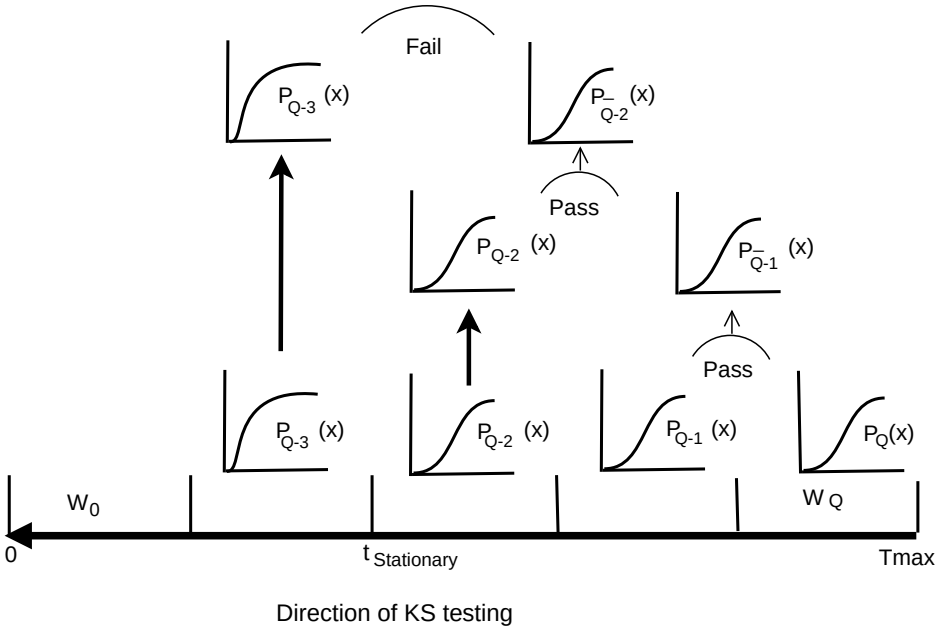


Figure 3.4: Sliding window KS-test process, as used in [1, 2].

Each measured feature is tested for stationarity separately since in general, the stationarity or non-stationarity of one measured feature, does not provide information about the stationarity or non-stationarity of other measured features from the same simulation. Additionally, the window width for each feature is not known a priori, so the STARS analysis framework tests multiple window widths to determine the most appropriate window size for each feature.

3.3.1.2 Testing for Ergodicity

Once stationary behaviour has been detected in measured features, these features may be tested for ergodicity in a similar manner to the stationarity testing. At the completion of stationarity testing, data from the stationary periods of each simulation run for each feature which passed the stationarity test, may be used to calculate the average empirical distribution. Once the average empirical distribution is available for a feature for each simulation run, testing for ergodicity can be done by again applying the two-sample KS-test used in stationarity testing. However, in testing for ergodic behaviour, pairwise testing is done across all simulation runs. By testing in this manner, it is possible to detect multiple modes of behaviour if they are present. Note that this application of pairwise KS-testing does not denote a multiple hypothesis test as pairs are tested separately and no claims are made regarding the pairwise similarity with respect to other pairs.

If stationary features pass the pairwise KS-tests for ergodicity, the features may be averaged and used in subsequent analysis. Note that measured features which do not pass the stationarity tests are not included in the ergodicity tests. For the purposes of this work, stationarity is an explicit prerequisite to ergodicity. Thus, non-stationary features are assumed to be non-ergodic for the purposes of this work.

Upon completion of ergodicity testing, STARS employs clustering techniques to detect multiple modes of behaviour. An ergodicity graph is formed by placing a weighted edge between all available samples. The graph is fully connected, and the edge weight is the corresponding p-value of the ergodicity KS-test. After creating this initial graph, edges are removed for which the edge weight (p-value) was less than the significance level ($\alpha = 0.05$) of the KS-test. In the remaining graph, only statistically similar samples will share an edge. At this point, repeated subsamples are removed, and the remaining maximal cliques are reported as the ergodic modes of the tested feature.

3.3.2 STARS and P2P Botnet Simulator Integration

While STARS provides generic support for stationarity and ergodicity testing, some additional work was required to integrate a new model such as the P2P botnet model

with the STARS framework. This mainly consisted of python development, which enables experiment automation and analysis. Some additional MATLAB development was also required, to support analysis of measured features. STARS provides a basic set of functionality which can be leveraged for analysis, but instrumentation specific to a particular model requires additional development work.

Finally, the model must be structured such that it can be packaged into an experiment resource archive that contains the simulation model and all relevant dependencies, such that the resource file can be sent to machines running STARS and can run simulations using information contained in the resource archive.

A workfile which is similar in form to the configuration files used in OMNeT++ is also needed and describes the experiment configurations such as how many simulation runs to conduct, how many Monte-Carlo runs of each parameter set should be performed, where to locate resource files, where to store results, per-run configurations, which features to collect data on, and which statistical analysis to run.

3.4 Summary

This chapter describes the tools used to run simulation experiments and generate results presented in this work. The central tool is an OMNeT++ based P2P Botnet simulator that simulates packet level behaviour of a Kademlia based P2P Botnet. This tool provides support for configurable underlay topologies, as well as the configuration of runtime parameters of the overlay protocol.

The P2P botnet simulator is used in conjunction with the STARS statistical testing framework, to ensure that statistically rigorous results are generated. In particular, the analysis component of STARS monitors the simulation output to detect periods stationary and ergodic behaviour and will run additional results if needed to ensure that sufficient samples of the measured features are available. Once simulation runs have been completed, additional analysis may be performed on the ergodic data sets.

Chapter 4

Experiments and Results

This chapter describes the setup of simulation based experiments used to assess the stationarity and ergodicity of measured botnet features, and the results of these experiments. The experiments are based on two different underlay routing network topologies, for which a set of operational botnet parameters is applied to create a parameter sweep for each of the two scenarios. In this way, both the impact of the underlying routing network, as well as the effect of operational botnet parameters on measured botnet features may be assessed.

4.1 Basic Setup

This section describes the basic setup and parameters settings used in experiments. These settings cover the common configuration of the operational parameters of the Kademlia protocol used for botnet C&C, as well as botnet parameters such as size and birth/death rate. These settings are common across both the small and realistic underlay experiments, and are listed in Tables 4.1 and 4.2

Within the simulation, two aspects are varied. First, two separate topologies are used for the underlay routing network. Second, a parameter sweep over 5 different values of key lookup interval (`tValueLookup`) is performed for each of the two underlay topologies. Note that hosts in the botnet are distributed across the subnetworks in the underlay topologies. That is, for the small underlay network, each subnetwork

Table 4.1: Fixed Kademlia protocol attributes

Attribute	Value
k	20
alpha	3
keyLength	128
peerListSize	200
tRefresh	3600
tReplicate	3600
maxStorageLocations	10

Table 4.2: Fixed botnet attributes in simulations

Attribute	Value
botnet size	20,000 hosts
birth rate	exponential(3600) seconds
death rate	exponential(3600) seconds
simulation time	24 hours

will contain approximately 2000 bots, and for the realistic underlay network each subnetwork will contain approximately 200 bots. A summary of variable attributes is given in Table 4.3.

Table 4.3: Attributes varied during simulation

Attribute	Value
Underlay routing network	Small(20 routers, 10 subnetworks), Realistic(2000 routers, 100 subnetworks)
tValueLookup	{300, 900, 1800, 3600, 7200} seconds

4.2 Small Underlay Network

For this experiment, an underlay routing network consisting of 20 routers and 10 subnetworks was used. Prior to running the simulations, a 20,000 node botnet overlay running on the specified 20 router 10 subnetwork underlay was generated using the generation mode of the P2P botnet simulator. This saved network was used as the starting point for subsequent steady state simulations.

As the steady state simulations all make use of the same network topology, they vary only by repetition. In this way, sensitivity to initial conditions and statistical behaviour of measured botnet features may be assessed. The basic setup described in Section 4.1 is used, and 30 simulation repeats are run across a set of 5 values of `tValueLookup`. These values are `tValueLookup` $= \{300, 900, 1800, 3600, 7200\}$ seconds resulting in 150 simulations for this experiment. Note that each bot is pre-configured to search for 32 different { key, value } pairs at regular intervals (i.e. `tValueLookup`) over a 24 hour period; hence 32 independent botnets are effectively simulated per Monte-Carlo run, as each of the 32 keys is associated with a different peer list. For the purposes of this work, the measured botnet data is assessed from the perspective of a defender, who would not have complete topology information about the peer-lists associated with the different keys, and thus would treat the entire botnet as a single process.

4.3 Realistic Underlay Network

For this experiment, an underlay network consisting of 2000 routers and 100 subnetworks was used. As discussed in Chapter 3, the underlay network is created using a subset of data from CAIDA's Skitter project [65]. Again, a 20,000 node botnet is created using the generation mode of the P2P botnet simulator and this topology is used in subsequent experiments. The basic setup described in Section 4.1 is used, and 30 simulation repeats are run across a set of 5 values of `tValueLookup`. These values are `tValueLookup` $= \{300, 900, 1800, 3600, 7200\}$ seconds, resulting in 150 simulation for this experiment. As in the small underlay experiment, bots are pre-configured to search for 32 different { key, value } pairs over a 24 hour period; hence 32 independent botnets are effectively simulated per Monte-Carlo run, as each of the 32 keys is associated with a different peer list. Again, the measured botnet data is assessed from the perspective of a defender, who would not have complete topology information about the peer-lists associated with the different keys, and thus would treat the entire botnet as a single process.

4.4 Practical Considerations

For each of the simulations, the required CPU time and storage required vary depending on the simulation parameters. These experiments were run on a cluster of 30 IBM Blade servers, each of which has an Intel Xeon dual core 3GHz processor and 8GB of RAM. Run times and storage requirements for each scenario are listed below in Tables 4.4, and 4.5 for the small and realistic underlay networks, respectively. Note that these numbers do not include the CPU time and storage required to run statistical analysis on the simulation results. Statistical analysis for each of the two experiments took approximately one week to run, and required an additional 280GB of storage space. The experiments were run using the STARS framework, which is MPI-aware and runs 2 simulation threads in parallel on each of the 30 machines in the cluster. This greatly reduced the total time required to complete the all the simulation runs, and allowed the experiments to be completed in a time practical manner. As can be seen from the tables below, it would have required several years to complete the simulations using only a single machine.

Table 4.4: Small underlay resource usage

Key lookup interval	**Runs**	**CPU time per run**	**CPU time total**	**Storage per run**	**Storage total**
300 seconds	30	5.25 days	158 days	340MB	10.2 GB
900 seconds	30	4.75 days	143 days	315MB	9.5GB
1800 seconds	30	4.25 days	128 days	280MB	8.4GB
3600 seconds	30	3.25 days	98 days	215MB	6.5GB
7200 seconds	30	2.25 days	68 days	160MB	4.8GB

Table 4.5: Realistic underlay resource usage

Key lookup interval	**Runs**	**CPU time per run**	**CPU time total**	**Storage per run**	**Storage total**
300 seconds	30	13.25 days	398 days	420MB	12.6GB
900 seconds	30	11.75 days	353 days	385MB	11.6GB
1800 seconds	30	10.25 days	308 days	340MB	10.2GB
3600 seconds	30	7.5 days	225 days	260MB	7.8GB
7200 seconds	30	5.5 days	165 days	190MB	5.7GB

4.5 Evaluation

This work seeks to assess the impact of varying initial conditions, using different underlay routing topologies, and modifying operational parameters of the botnet C&C protocol on measured botnet features. Measured botnet features are tested for stationarity and ergodicity in order to gain a better understanding of the statistical behaviour of the measured features.

4.5.1 Assessment of Measured Features

Analysis focuses on measuring features of the overlay traffic associated with a successful key-value pair retrieval, and four features are considered. The first feature is the duration of a successful lookup. This measurement records how long it has taken for a host to successfully retrieve a key-value pair from the overlay network. The second feature is the key-value pair index of the successful lookup. In this work, 32 different key-value pairs are searched for; the index is used as an identifier to determine which of the keys was searched for, or alternatively, which of the values was found. The third feature is the lookup interval. This measurement records the time interval between successful lookups. Last, is the subnet feature, which records the subnet number of the host, each time a host successfully retrieves a key-value pair from the overlay network.

Statistical stationarity and ergodicity tests were run on the measured features using the STARS framework, following the procedure outlined in Chapter 3. Results are reported for the small underlay and realistic underlay experiments in Sections 4.5.1.1 - 4.5.1.4 below. Note that the set of features used in this work was selected to demonstrate that interesting statistical behaviours can occur in measured botnet features, but is not intended to provide a complete assessment. It would be worthwhile to consider a wider range of operational parameters and a richer set of underlay and overlay topologies; these avenues for future work are discussed in section 5.2.

For each of the experiments, the number of non-stationary, stationary, and ergodic samples are reported. For samples with ergodic behaviour, the number of ergodic modes, as well as the mode sizes are also reported. Finally, estimated empirical distributions of the measured features are provided.

4.5.1.1 Lookup Duration Evaluation

For the small underlay network, the lookup duration feature was found to have 2 stationary records out of 150 samples. Both of the stationary samples were produced by the 300 second key lookup interval. A summary of the statistical evaluation results is provided in Table 4.6. The stationary samples were found to be ergodic, but it must be noted that the ergodic mode contains only about 1% of the total samples from the small underlay experiment. The estimated empirical distribution of this feature is provided in Figures 4.1a and 4.1b. For the realistic underlay network, this feature

Table 4.6: Small underlay lookup duration

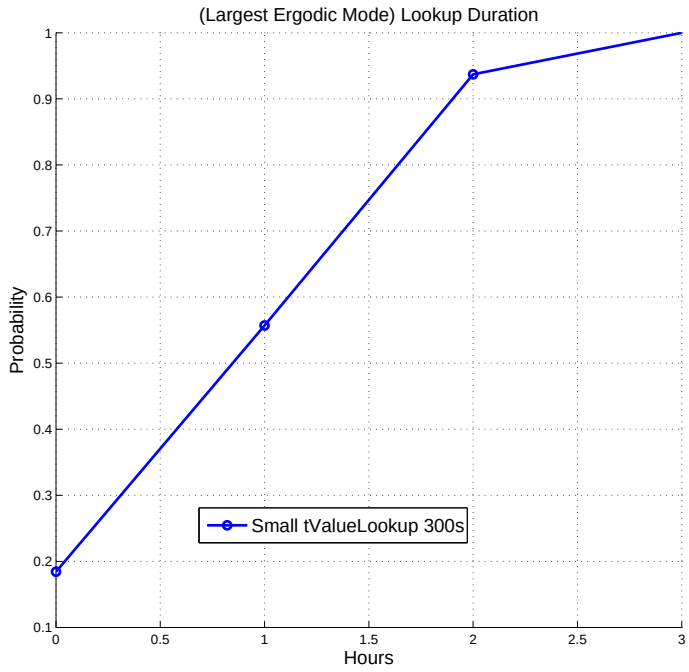
Key Lookup interval	**Non-stationary**	**Stationary**	**Ergodic**	**Modes**	**Mode sizes**
300 seconds	28	2	2	1	{2}
900 seconds	30	0	0	-[1]	-
1800 seconds	30	0	0	-	-
3600 seconds	30	0	0	-	-
7200 seconds	30	0	0	-	-

Table 4.7: Realistic underlay lookup duration

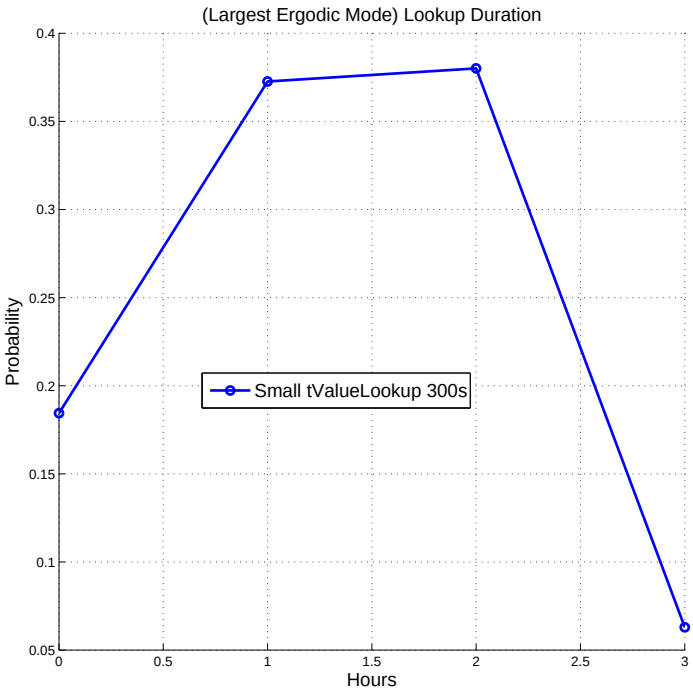
Key Lookup interval	**Non-stationary**	**Stationary**	**Ergodic**	**Modes**	**Mode sizes**
300 seconds	7	23	23	3	{18,4,2}
900 seconds	1	29	29	4	{11,10,9,6}
1800 seconds	0	30	27	7	{11,5,4,4,3,2,2}
3600 seconds	30	0	0	-	-
7200 seconds	30	0	0	-	-

had 82 stationary samples and 68 non-stationary samples. All key lookup intervals except for 1800 seconds produced stationary samples. Of the stationary samples, 4 were non-ergodic, and 14 modes of behaviour were recorded for the ergodic samples. The statistical evaluation results are provided in Table 4.7. The largest ergodic modes for this feature contained more than 10 samples. The estimated empirical distributions for this feature are provided in Figures 4.2a and 4.2b.

[1]Denotes no entry.

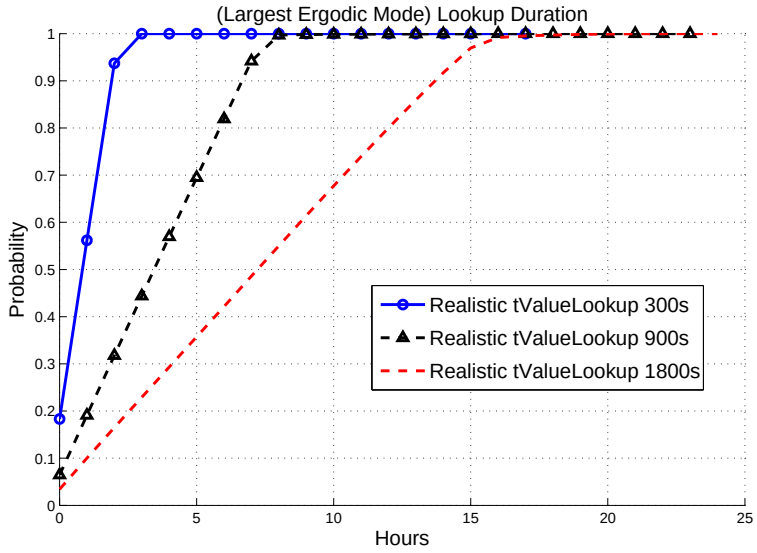


(a) CDF of lookup duration

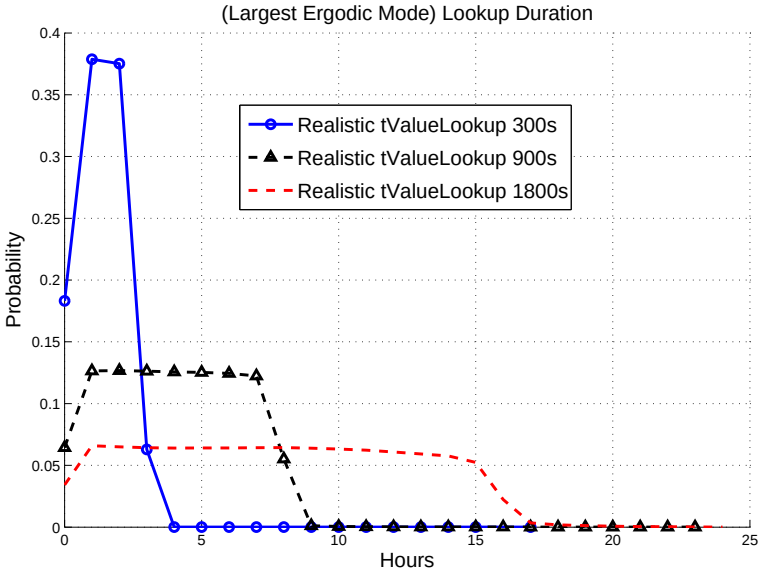


(b) PDF of lookup duration

Figure 4.1: Estimated empirical distribution for small underlay lookup duration



(a) CDF of lookup duration



(b) PDF of lookup duration

Figure 4.2: Estimated empirical distribution for realistic underlay lookup duration

4.5.1.2 Lookup Interval Evaluation

For the small underlay network, the lookup interval feature was found to have 63 stationary samples. A summary of the statistical evaluation results is provided in

Table 4.8. Note that the 900 and 1800 second key lookup intervals did not produce any stationary samples. Of the stationary samples generated, all were found to be ergodic, and 8 modes of behaviour were observed. For the 3600 and 7200 second key lookup intervals, the largest ergodic modes contained more than 10 samples. For the 300 second key lookup time, the largest ergodic mode contained only 3 samples. The estimated empirical distribution of this feature is provided in Figures 4.3a and 4.3b.

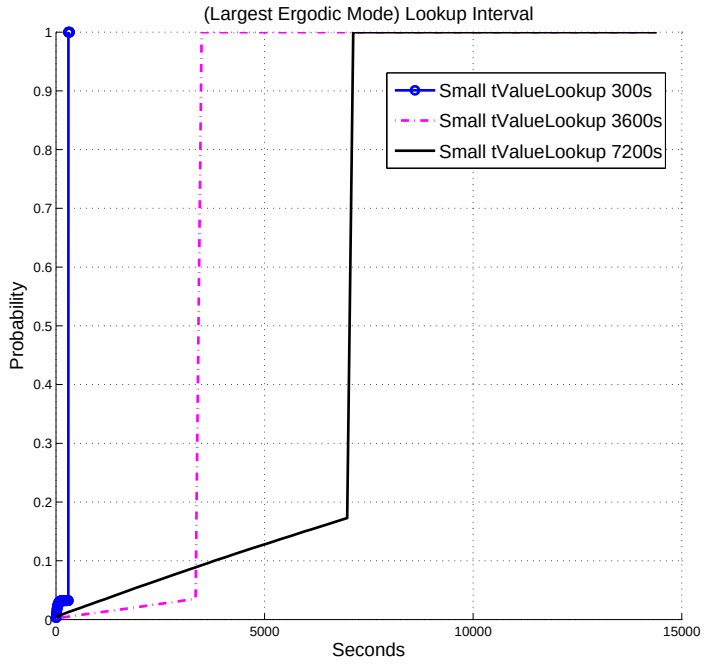
For the realistic underlay network, this feature was found to have 149 stationary samples, and only 1 non-stationary sample. All of the stationary samples were ergodic, and 20 modes of behaviour were recorded for the ergodic samples. The statistical evaluation results are provided in Table 4.9. The largest ergodic modes for this feature all contained more than 10 samples. The estimated empirical distributions for this feature are provided in Figures 4.4a and 4.4b.

Table 4.8: Small underlay lookup interval

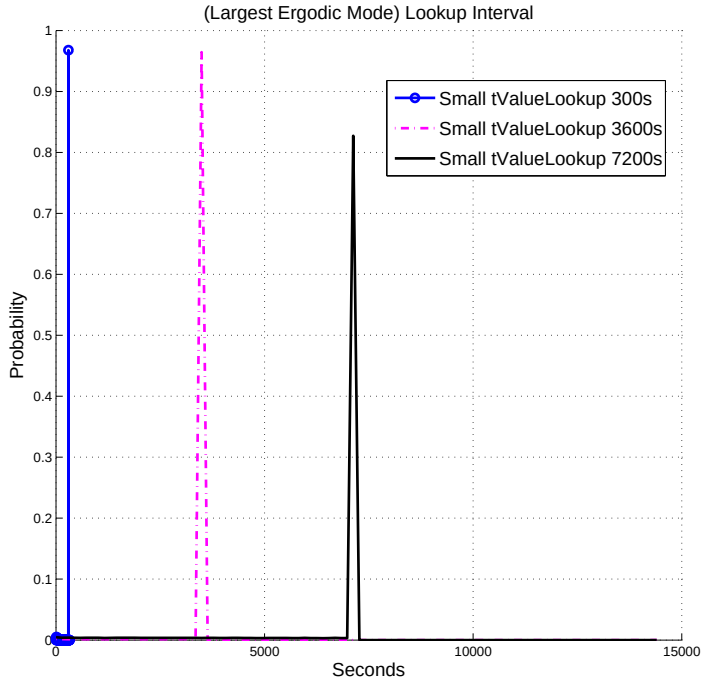
Key Lookup interval	**Non-stationary**	**Stationary**	**Ergodic**	**Modes**	**Mode sizes**
300 seconds	27	3	3	1	{3}
900 seconds	30	0	0	-	-
1800 seconds	30	0	0	-	-
3600 seconds	0	30	30	3	{16,15,7}
7200 seconds	0	30	30	4	{18,14,11,2}

Table 4.9: Realistic underlay lookup interval

Key Lookup interval	**Non-stationary**	**Stationary**	**Ergodic**	**Modes**	**Mode sizes**
300 seconds	0	30	30	1	{30}
900 seconds	0	30	30	5	{18,17,16,15,13}
1800 seconds	0	30	30	3	{24,14,11}
3600 seconds	0	30	30	3	{21,12,8}
7200 seconds	1	29	29	8	{14,11,6,6,4,4,3,3}

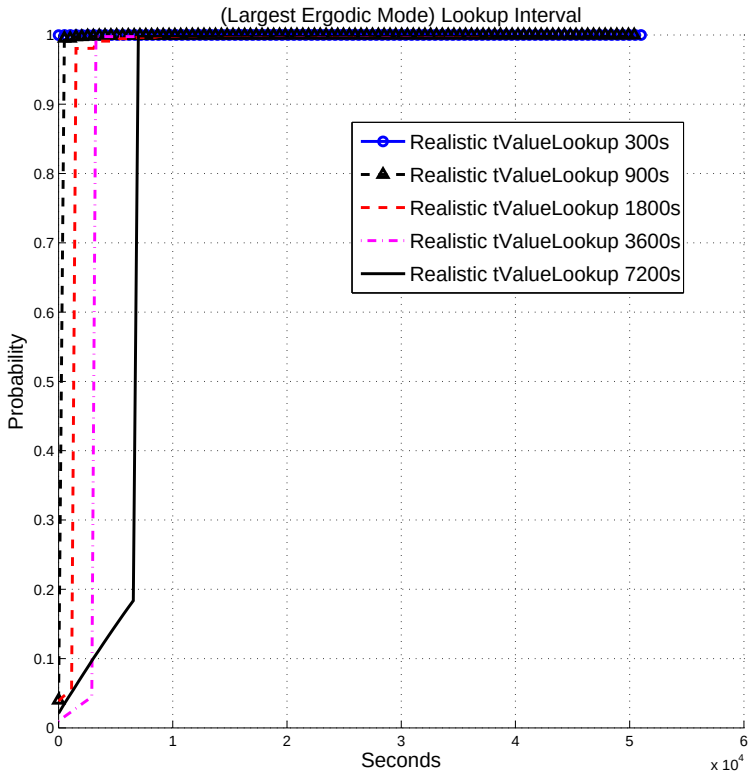


(a) CDF of lookup interval

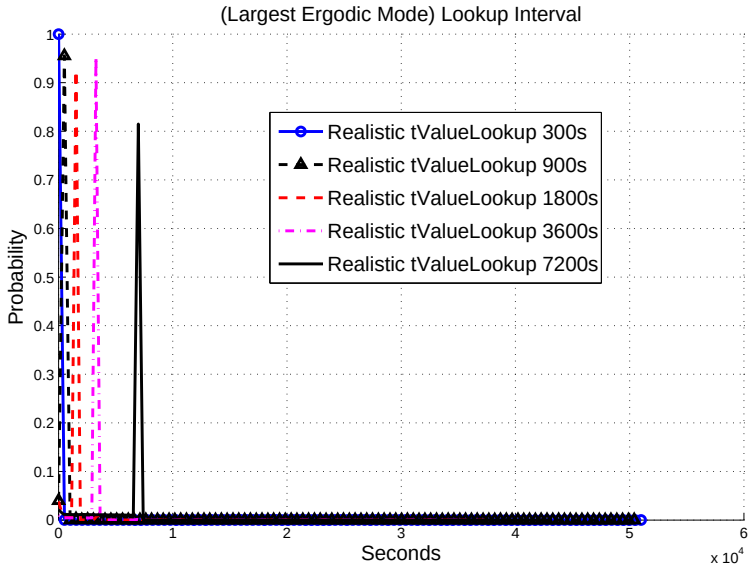


(b) PDF of lookup interval

Figure 4.3: Estimated empirical distribution for small underlay lookup interval



(a) CDF of lookup interval



(b) PDF of lookup interval

Figure 4.4: Estimated empirical distribution for realistic underlay lookup interval

4.5.1.3 Key-Value Pair Index Evaluation

For the small underlay network, the key-value pair index feature was found to have 105 stationary samples. A summary of the statistical evaluation results is provided in Table 4.10. Of the stationary samples, 12 samples were non-ergodic. The remaining stationary samples were ergodic, and displayed 17 modes of behaviour. For the 300, 900 and 1800 second key lookup intervals, the largest ergodic modes contained less than 10 samples. For the 3600 and 7200 second key lookup intervals, the largest ergodic modes contained more than 25 samples. The estimated empirical distribution of this feature is provided in Figures 4.5a and 4.5b.

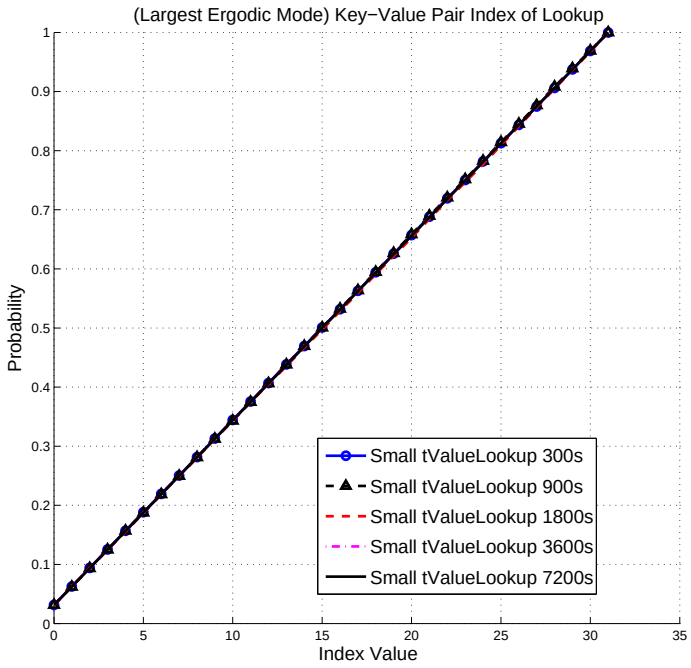
For the realistic underlay network, all 150 samples of this feature were found to be stationary. All of the stationary samples were ergodic, and 14 modes of behaviour were observed. The statistical evaluation results are provided in Table 4.11. The largest ergodic modes for this feature all contained more than 20 samples. The estimated empirical distributions for this feature are provided in Figures 4.6a and 4.6b.

Table 4.10: Small underlay key-value pair index

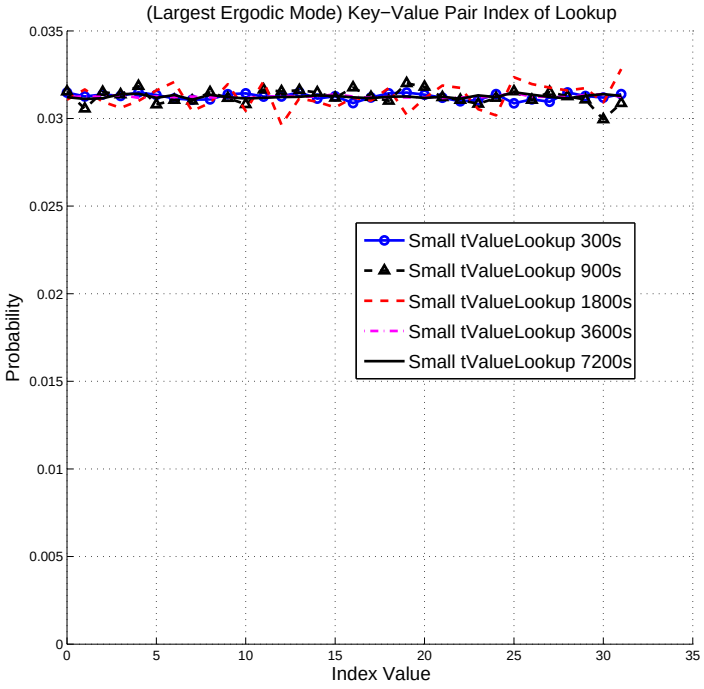
Key Lookup interval	**Non-stationary**	**Stationary**	**Ergodic**	**Modes**	**Mode sizes**
300 seconds	11	19	19	6	{7,7,5,5,4,2}
900 seconds	14	16	13	7	{3,3,3,3,2,2,2}
1800 seconds	20	10	2	1	{2}
3600 seconds	0	30	30	1	{30}
7200 seconds	0	30	29	2	{28,28}

Table 4.11: Realistic underlay key-value pair index

Key Lookup interval	**Non-stationary**	**Stationary**	**Ergodic**	**Modes**	**Mode sizes**
300 seconds	0	30	30	1	{30}
900 seconds	0	30	30	4	{27,25,25,25}
1800 seconds	0	30	30	6	{22,17,16,16,3,2}
3600 seconds	0	30	30	1	{30}
7200 seconds	0	30	30	2	{29,29}

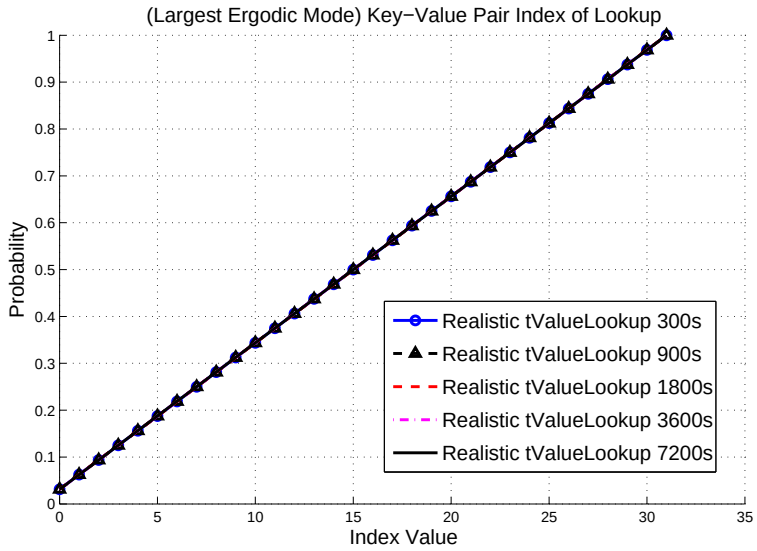


(a) CDF of key-value pair index

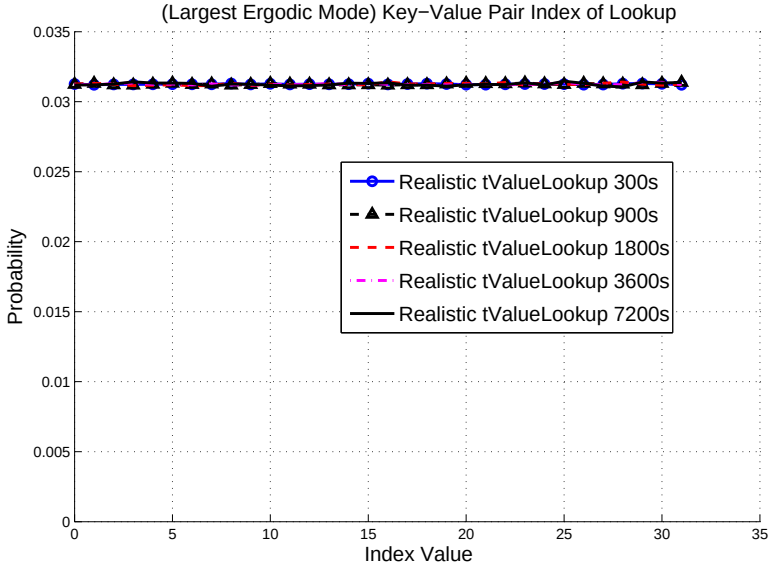


(b) PDF of key-value pair index

Figure 4.5: Estimated empirical distribution for small underlay key-value pair index



(a) CDF of key-value pair index



(b) PDF of key-value pair index

Figure 4.6: Estimated empirical distribution for realistic underlay key-value pair index

4.5.1.4 Subnet Number of Lookup

For the small underlay network, the subnet feature was found to have 60 stationary samples. A summary of the statistical evaluation results by parameter is provided in

Table 4.12. Note that stationary samples were only produced for the 3600 and 7200 second key lookup intervals. The remaining key lookup intervals did not produce any stationary samples. Of the stationary samples, all were found to be ergodic, and 4 modes of behaviour were observed. The largest ergodic modes for both the 3600 and 7200 second key lookup intervals contained more than 25 samples. The estimated empirical distribution of this feature is provided in Figures 4.7a and 4.7b.

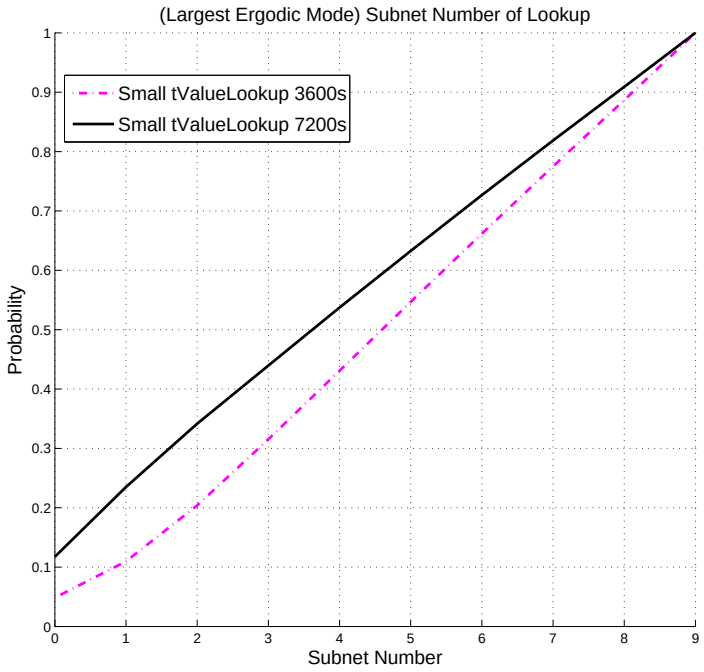
For the realistic underlay network, 139 samples were found to be stationary. Of the stationary samples, 24 were found to be non-ergodic; all of the stationary but non-ergodic samples were produced by the 1800 second key lookup interval. The remaining stationary samples were found to be ergodic, and 9 modes of behaviour were observed. A summary of the statistical evaluation results by parameter are provided in Table 4.13. The largest ergodic modes for this feature contained only 3 samples for the 1800 second key lookup interval, but contained at least 15 samples for the 3600 and 7200 second key lookup intervals. The estimated empirical distributions for this feature are provided in Figures 4.8a and 4.8b.

Table 4.12: Small underlay subnet number of key-value pair

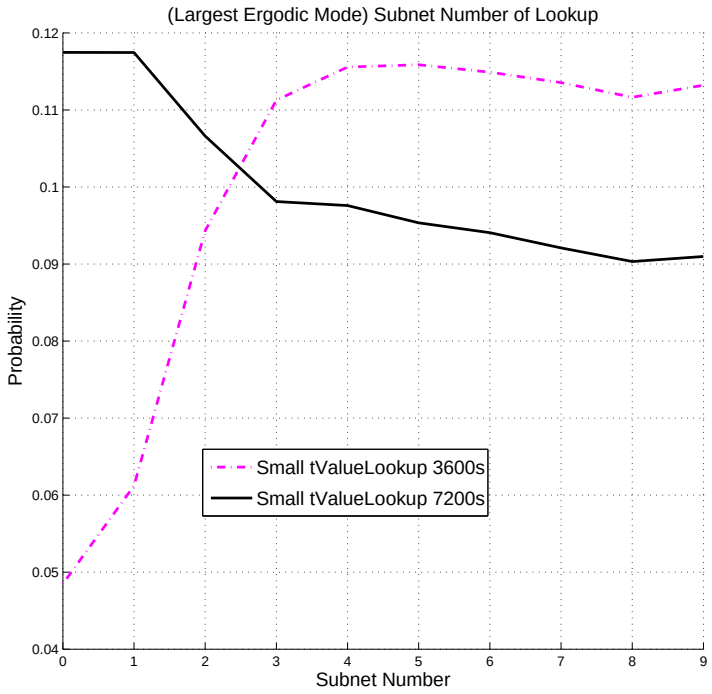
Key Lookup interval	**Non-stationary**	**Stationary**	**Ergodic**	**Modes**	**Mode sizes**
300 seconds	30	0	0	-	-
900 seconds	30	0	0	-	-
1800 seconds	30	0	0	-	-
3600 seconds	0	30	30	2	{28,12}
7200 seconds	0	30	30	2	{26,25}

Table 4.13: Realistic underlay subnet number of key-value pair

Key Lookup interval	**Non-stationary**	**Stationary**	**Ergodic**	**Modes**	**Mode sizes**
300 seconds	9	21	0	-	-
900 seconds	0	30	0	-	-
1800 seconds	0	30	6	3	{2,2,2}
3600 seconds	2	28	28	3	{20,18,18}
7200 seconds	0	30	30	3	{15,13,13}

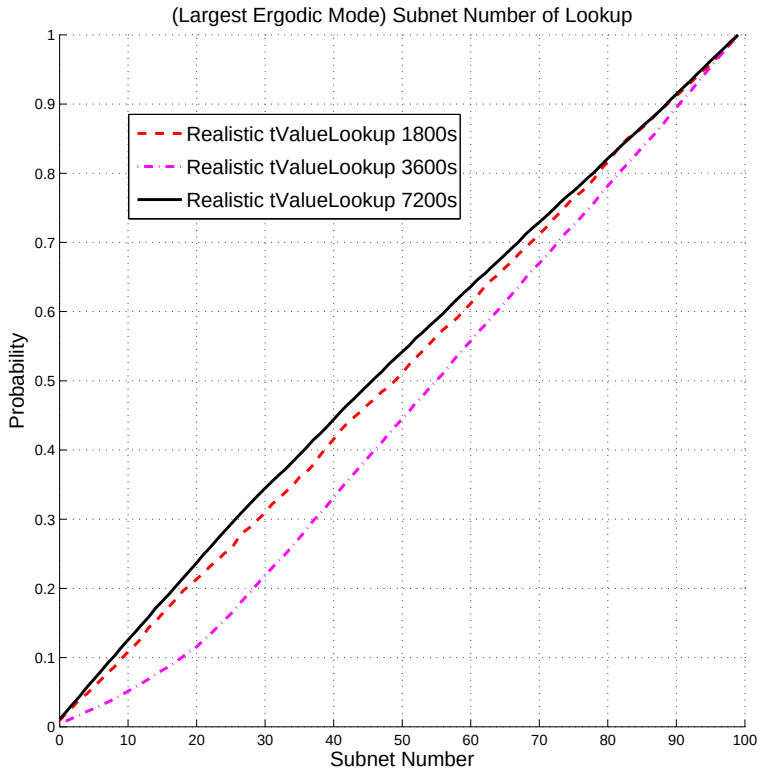


(a) CDF of subnet number

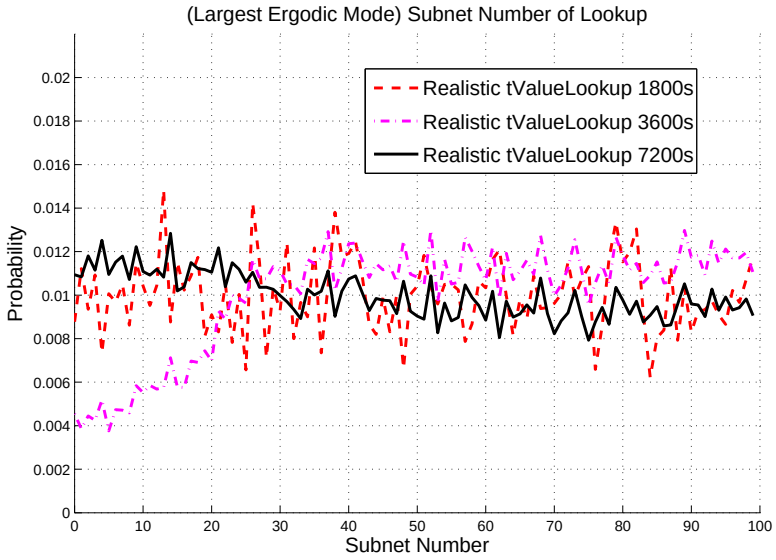


(b) PDF of subnet number

Figure 4.7: Estimated empirical distribution for small underlay subnet number



(a) CDF of subnet number



(b) PDF of subnet number

Figure 4.8: Estimated empirical distribution for realistic underlay subnet number

4.6 Summary of Evaluation

This chapter presented an outline of experiments conducted, as well as the results and analysis of the obtained experiment data. For the small underlay network, the key-value pair index feature was observed to be the most ergodic measured feature, with all key value lookup intervals producing ergodic modes of behaviour. The largest ergodic modes were observed for the 3600 and 7200 second key lookup intervals, with more than 25 samples in each of these modes. The least ergodic feature for the small underlay network was the lookup duration. Only 2 out of 150 samples were stationary. These 2 samples were also ergodic, but the ergodic mode of behaviour for this feature accounts for only about 1% of the sample data for this feature.

For the realistic underlay network, the most ergodic feature again was found to be the key-value pair index. All 150 of the samples were found to be stationary, and all of the stationary samples were also found to be ergodic measured feature, with 14 modes of behaviour observed. The largest modes of behaviour for this feature all contained at least 20 samples. The least ergodic features for this experiment were the lookup duration and subnet features. For the lookup duration feature, 82 of the 150 samples displayed stationary behaviour. Of the stationary samples, 4 samples were found to be non-ergodic while the remaining stationary samples were found to be ergodic with 14 modes of behaviour observed; the largest ergodic modes all contained at least 10 samples. For the subnet feature, 139 stationary samples were observed and 24 of the stationary samples were found to be non-ergodic. The remaining stationary samples were found to be ergodic, and 9 modes of behaviour were observed. With the exception of the 300 second key value lookup interval, the largest ergodic modes for the subnet feature contained at least 15 samples.

It is interesting to note that the different key value lookup interval parameter settings resulted in distinct statistical behaviours for some of the measured features, and that varying initial conditions (simulation runs with different random seeds) also impacted the statistical behaviour of the measured features.

A general trend was observed in the small underlay network experiment that at first appears somewhat counter-intuitive. For the small underlay network, measured features became more stationary and more ergodic as the key lookup interval was increased. The volume of overlay traffic generated is strongly influenced by the key

lookup intervals, and for the small network, observed features contained more stationary and ergodic samples when key lookup intervals were large, and thus the traffic volumes in the network were reduced. In contrast to this, the measured features became less ergodic in the small network when the key lookup interval was small, and the traffic volumes were larger. One possible explanation is that the underlay routing network was overwhelmed by the high traffic volumes generated by the small key lookup intervals, resulting in dropped packets and less stationary and ergodic behaviour present in the measured features.

In contrast to the small network experiments, the realistic underlay experiment did not display the same trends. For this experiment, the measured features generally became less stationary and ergodic as the key lookup interval increased. In this case, the underlay routing network is significantly larger (i.e. 2000 routers as opposed to 20 routers), hence the high traffic volumes generated by small key lookup intervals would have been less likely to overwhelm the underlying routing network, and hosts in the overlay network are likely to be close in a network sense to other overlay hosts. What is most noteworthy about the trends observed in these experiments, is that results indicate that both the underlay routing network and the operational parameters of the botnet can influence the statistical behaviours of measured overlay network features.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Botnets remain a serious security threat, and as such, considerable research efforts will continue to be directed towards understanding and mitigating this threat. Many of the currently available research works are based on observation or simulation and innately assume that observed botnet behaviour is generalizable, i.e. that botnets can be reasonably modeled as stationary and ergodic processes. The fact that botnets remain a persistent and recurring security threat suggests that they are complex processes, and that assumptions of ergodicity may not be warranted.

This work is motivated by theorems from the field of ergodic theory, which discuss stationarity and ergodicity and clearly state that these are not universal properties. Hence, there are no known theory reasons to assume that botnets MUST be stationary or ergodic. As the statistical nature of observed botnet behaviours have not been well addressed in existing research, this work provides a preliminary assessment by investigating the range of behaviours displayed over time, and across an ensemble of Monte-Carlo experiments.

Assessment of measured botnet features is conducted using a custom-built OMNeT++ packet level simulation of a Kademlia based P2P botnet, in conjunction with a distribution-free statistical testing framework called STARS. Simulation results clearly indicate that measured botnet features can exhibit both non-stationary

and non-ergodic behaviours. In addition to the non-stationary and non-ergodic behaviours observed, the features which were found to contain ergodic samples, often contained multiple modes of behaviour.

This work has served to illustrate that complex behaviours are indeed present measured botnet features and, as such, it is important that rigorous statistical analysis be conducted on collected data as part of the engineering process.

5.2 Future Work

Although a reasonably large set of 300 simulations was conducted for this work, several of the measured features produced only a very small number of ergodic samples. In light of this, it would be useful to conduct a larger set of experiments to better understand the behaviours displayed by the measured features and the mechanisms which may lead to non-stationary and non-ergodic behaviours. It would be desirable to have ergodic modes with a minimum of 50 records, potentially up to several hundred records per ergodic mode if resources permitted. In order to do this, it would likely be necessary to utilize high performance computing (HPC) resources to ensure simulations were completed in a time practical manner.

With regard to exploring the mechanisms which may lead to non-stationary and non-ergodic behaviour, it would be interesting to study a richer set of operational parameters (i.e. running simulations with a wider selection of values for `tValueLookup`), and investigate other operational parameters of the botnet and the Kademlia protocol such as the peer list sizes, the birth and death processes for bots joining and leaving the botnet, and the number of seeding hosts initially storing key-value pairs.

In addition to exploring a wider range of operational parameters, it would be valuable to investigate the influence of total botnet size, and distribution of bots across the underlay network on measured overlay features. Further to this, a final avenue for future work would be to study a wider range of overlay and underlay topologies. In particular, for the Kademlia-based botnet used in this work, studying the structural properties of subgraphs within the botnet associated with the 32 key-value pairs may provide insight as to whether these graph structures could also influence the statistical properties of measured features of the overlay network.

Bibliography

[1] E. Millman, D. Arora, and S. 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*, March 2011, pp. 733 –739.

[2] E. Millman, "Analyzing manet jamming strategies," Master's thesis, Department of Electrical and Computer Engineering, University of Victoria, 2011.

[3] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm," in *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*. Berkeley, CA, USA: USENIX Association, April 2008, pp. 9:1–9:9. [Online]. Available: http://dl.acm.org/citation.cfm?id=1387709.1387718

[4] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: understanding, detecting, and disrupting botnets," in *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*. Berkeley, CA, USA: USENIX Association, July 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251282.1251288

[5] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, Sept. 2006.

[6] E. Rhyne, "Federal R&D landscape and DHS S&T," 2011. [Online]. Available: https://buildsecurityin.us-cert.gov/swa/wg_presentations_dec2011/Rhyne-RAndD_AndDHS_SAndT.pdf

[7] J. Oikarinen and D. Reed, "Internet relay chat protocol request for comments (rfc 1459)," 1993. [Online]. Available: http://tools.ietf.org/html/rfc1459

[8] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: detecting malware infection through ids-driven dialog correlation," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, Aug. 2007, pp. 12:1–12:16. [Online]. Available: http://dl.acm.org/citation.cfm?id=1362903.1362915

[9] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: overview and case study," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA: USENIX Association, April 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1323128.1323129

[10] J. Calvet, C. Davis, and P.-M. Bureau, "Malware authors don't learn, and that's good!" in *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, Oct. 2009, pp. 88 –97.

[11] P. Porras, S. Hassan, and V. Yegneswaran, "An analysis of conficker c," SRI International, Tech. Rep., 2009.

[12] D. Dittrich and S. Dietrich, "P2P as botnet command and control: A deeper insight," in *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, Oct. 2008, pp. 41 –48.

[13] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys Tutorials, IEEE*, vol. 7, no. 2, pp. 72 – 93, 2005.

[14] C. Davis, S. Neville, J. Fernandez, J.-M. Robert, and J. McHugh, "Structured peer-to-peer overlay networks: Ideal botnets command and control infrastructures?" in *Computer Security - ESORICS 2008*. Springer Berlin / Heidelberg, Oct. 2008, vol. 5283, pp. 461–480. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88313-5_30

[15] P. Kirk, "Gnutella protocol development," 2003. [Online]. Available: http://rfc-gnutella.sourceforge.net/

[16] A.-L. Barabsi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999. [Online]. Available: http://www.sciencemag.org/content/286/5439/509.abstract

[17] P. Erdös and A. Rényi, "On random graphs I," *Publicationes Mathematicae Debrecen*, vol. 6, p. 290, 1959.

[18] D. Dagon, G. Gu, C. Lee, and W. Lee, "A taxonomy of botnet structures," in *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, Dec. 2007, pp. 325 –339.

[19] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*. London, UK, UK: Springer-Verlag, 2002, pp. 53–65. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687801

[20] Symantec, "Internet security threat report," 2011. [Online]. Available: http://www.symantec.com/threatreport/

[21] B. Mukherjee, L. Heberlein, and K. Levitt, "Network intrusion detection," *Network, IEEE*, vol. 8, no. 3, pp. 26 –41, May-June 1994.

[22] K. A. Scarfone and P. M. Mell, "Sp 800-94. guide to intrusion detection and prevention systems (idps)," Gaithersburg, MD, United States, Tech. Rep., 2007.

[23] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*, ser. LISA '99. Berkeley, CA, USA: USENIX Association, Nov. 1999, pp. 229–238. [Online]. Available: http://dl.acm.org/citation.cfm?id=1039834.1039864

[24] "Snort." [Online]. Available: http://www.snort.org/

[25] W. K. Ehrlich, A. Karasaridis, D. Liu, and D. Hoeflin, "Detection of spam hosts and spam bots using network flow traffic modeling," in *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*. Berkeley, CA, USA: USENIX Association, April 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855686.1855693

[26] J. McHugh, R. McLeod, and V. Nagaonkar, "Passive network forensics: behavioural classification of network hosts based on connection patterns,"

SIGOPS Oper. Syst. Rev., vol. 42, no. 3, pp. 99–111, April 2008. [Online]. Available: http://doi.acm.org/10.1145/1368506.1368520

[27] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128612003568

[28] J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, and X. Luo, "Detecting stealthy P2P botnets using statistical traffic fingerprints," in *Dependable Systems Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, June 2011, pp. 121 –132.

[29] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "Botgrep: finding P2P bots with structured graph analysis," in *Proceedings of the 19th USENIX conference on Security*. Berkeley, CA, USA: USENIX Association, Aug. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1929820.1929830

[30] M. Jelasity and V. Bilicki, "Towards automated detection of peer-to-peer botnets: on the limits of local approaches," in *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*. Berkeley, CA, USA: USENIX Association, April 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855676.1855679

[31] J. Franois, S. Wang, R. State, and T. Engel, "Bottrack: Tracking botnets using netflow and pagerank," in *NETWORKING 2011*. Springer Berlin / Heidelberg, 2011, vol. 6640, pp. 1–14. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20757-0_1

[32] J. Francois, S. Wang, W. Bronzi, R. State, and T. Engel, "Botcloud: Detecting botnets using mapreduce," in *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, Dec 2011, pp. 1 –6.

[33] C. Davis, J. Fernandez, S. Neville, and J. McHugh, "Sybil attacks as a mitigation strategy against the storm botnet," in *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, Oct. 2008, pp. 32 –40.

[34] C. Davis, J. Fernandez, and S. Neville, "Optimising sybil attacks against P2P-based botnets," in *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, Oct. 2009, pp. 78 –87.

[35] D. Dittrich, "So you want to take over a botnet," in *Proceedings of the 5th USENIX conference on Large-Scale Exploits and Emergent Threats*. Berkeley, CA, USA: USENIX Association, April 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228340.2228349

[36] P. Wang, S. Sparks, and C. Zou, "An advanced hybrid peer-to-peer botnet," *Dependable and Secure Computing, IEEE Transactions on*, vol. 7, no. 2, pp. 113 –127, April-June 2010.

[37] D. Dagon, C. Zou, and W. Lee, "Modeling botnet propagation using time zones," in *Proceedings of the 13 th Network and Distributed System Security Symposium NDSS*, Feb. 2006.

[38] L.-P. Song, Z. Jin, and G.-Q. Sun, "Modeling and analyzing of botnet interactions," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 2, pp. 347 – 358, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437110008381

[39] A. Bensoussan, M. Kantarcioglu, and S. Hoe, "A game-theoretical approach for finding optimal strategies in a botnet defense model," in *Proceedings of the First international conference on Decision and game theory for security*, Nov. 2010.

[40] E. Van Ruitenbeek and W. Sanders, "Modeling peer-to-peer botnets," in *Quantitative Evaluation of Systems, 2008. QEST '08. Fifth International Conference on*, Sept. 2008, pp. 307 –316.

[41] D. Ha, G. Yan, S. Eidenbenz, and H. Ngo, "On the effectiveness of structural detection and defense against P2P-based botnets," in *Dependable Systems Networks, 2009. DSN '09. IEEE/IFIP International Conference on*, July 2009, pp. 297 –306.

[42] Y. Zeng, G. Yan, S. Eidenbenz, and K. Shin, "Measuring the effectiveness of infrastructure-level detection of large-scale botnets," in *Quality of Service (IWQoS), 2011 IEEE 19th International Workshop on*, June 2011, pp. 1 –9.

[43] K.J.Higgins, "The world's biggest botnet," 2007. [Online]. Available: http://www.darkreading.com

[44] S. Gaudin, "Storm worm botnet more powerful than top supercomputers," 2007. [Online]. Available: http://www.informationweek.com/news/201804528

[45] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA: USENIX Association, April 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1323128.1323133

[46] J. Barnes and L. Koss, "The ergodic theory carnival," *Mathematics Magazine*, vol. 83, no. 3, pp. 180–190, 2010. [Online]. Available: http://www.jstor.org/stable/10.4169/002557010X494823

[47] C. Ulcigrai, "Birkhoff ergodic theorem and applications," 2010. [Online]. Available: www.maths.bristol.ac.uk/ maxcu/Birkhoff.pdf

[48] "The network simulator ns-2." [Online]. Available: http://nsnam.isi.edu/nsnam/

[49] "Prime ssf." [Online]. Available: https://www.primessf.net/bin/view/Public

[50] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. Doyle, W. Sanders, and P. Webster, "The mobius modeling tool," in *Petri Nets and Performance Models, 2001. Proceedings. 9th International Workshop on*, Sept. 2001, pp. 241–250.

[51] A. Vargas, "Omnet++." [Online]. Available: http://omnetpp.org

[52] E. Weingartner, H. vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *Communications, 2009. ICC '09. IEEE International Conference on*, June 2009, pp. 1 –5.

[53] "The network simulator ns-3." [Online]. Available: http://www.nsnam.org/

[54] A. Vargas, "Inetmanet." [Online]. Available: https://github.com/inetmanet/

[55] "Castalia." [Online]. Available: http://castalia.npc.nicta.com.au/

[56] I. Baumgart, B. Heep, and S. Krause, "Oversim: A flexible overlay network simulation framework," in *IEEE Global Internet Symposium, 2007*. IEEE, May 2007, pp. 79–84.

[57] T. Dreibholz, E. P. Rathgeb, and X. Zhou, "Simproctc: the design and realization of a powerful tool-chain for omnet++ simulations," in *Proceedings of*

the 2nd International Conference on Simulation Tools and Techniques. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), March 2009, pp. 75:1–75:8. [Online]. Available: http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5517

[58] S. Sroka and H. Karl, "Using akaroa2 with omnet++," in *Proceedings of the 2nd International OMNeT++ Workshop, pp 43-50*, Jan. 2002.

[59] E. Millman, "Stars framework source code," 2011. [Online]. Available: https://github.com/emillman/STARS

[60] R. Gray, *Probabilty, Random Processes, and Ergodic Properties*, 1st ed. Springer-Verlag, 2009.

[61] K. Petersen, *Ergodic Theory*, 1st ed. Cambridge University PressSpringer-Verlag, 1989.

[62] P. Peebles, *Probability, Random Variables, and Random Signal Principles*, 4th ed. McGraw-Hill, 2001.

[63] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, Jan 1998. [Online]. Available: http://doi.acm.org/10.1145/272991.272995

[64] "Inet." [Online]. Available: http://inet.omnetpp.org/

[65] B. Huffaker, Y. Hyun, D. Andersen, and K. Claffy, "Skitter as links dataset. December 21, 2007 to January 31, 2008." [Online]. Available: http://www.caida.org/data/active/skitter_aslinks_dataset.xml

[66] D. Arora, T. Godkin, A. Verigin, and S. Neville, "Assessing trade-offs between stealthiness and node recruitment rates in peer-to-peer botnets," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 Seventh International Conference on*, Nov. 2012, pp. 148 –155.

[67] Mathworks, "Matlab." [Online]. Available: http://www.mathworks.com/

[68] J.-Y. Le Boudec, *Performance Evaluation of Computer and Communication Systems*. EPFL Press, Lausanne, Switzerland, 2010.