CARNEGIE MELLON UNIVERSITY

BEHAVIORAL MODELING OF BOTNET POPULATIONS VIEWED THROUGH INTERNET PROTOCOL ADDRESS SPACE

A DISSERTATION SUBMITTED TO THE GRADUATE SCHOOL IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree of

DOCTOR OF PHILOSOPHY
In
STATISTICS

by

RHIANNON LISA WEAVER

Department of Statistics
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

May, 2012

Thanks to all the faculty, staff, friends, family, and co-workers who helped me along the way.

This endeavor is dedicated in memory of

Colleen Kilker

whose passion for simplicity, precision, and pure mathematics, ignited my own.

Abstract

A botnet is a collection of computers infected by a shared set of malicious software, that maintain communications to a single human administrator or small organized group. Botnets are indirectly observable populations; cyber-analysts often measure a botnet's threat in terms of its size, but size is derived from a count of the observable network touchpoints through which infected machines communicate. Activity is often a count of packets or connection attempts, representing logins to command and control servers, spam messages sent, peer-to-peer communications, or other discrete network behavior. Front line analysts use sandbox testing of a botnet's malicious software to discover signatures for detecting an infected computer and shutting it down, but there is less focus on modeling the botnet population as a collection of machines obscured by the kaleidoscope view of Internet Protocol (IP) address space. This research presents a Bayesian model for generic modeling of a botnet due to its observable activity across a network. A generation-allocation model is proposed, that separates observable network activity at time t into the counts y_t generated by the malicious software, and the network's allocation of these counts among available IP addresses. As a first step, the framework outlines how to develop a directly observable behavioral model informed by sandbox tests and day-to-day user activity, and then how to use this model as a basis for population estimation in settings using proxies or Network Address Translation (NAT) in which only the aggregate sum of all machine activity is observed. The model is explored via a case study using the Conficker-C botnet that emerged in March of 2009.

Contents

Chapter 1

Introduction

This document outlines a Bayesian methodological framework for modeling the behavior of machines in a botnet given their contact points through observed network addresses, and provides a case study based on the behavior of the Conficker-C botnet that emerged in March of 2009. The analysis begins by formulating a probability model for the observable behavior of a single infected machine, informed by traditional defensive malware analytic techniques. The single-machine model is then used together with knowledge about common network resource allocation policies and typical user behavior to simulate the population as it is viewed through the lens of IP address space. Because botnets share the same or very similar code bases, the model of a single host can generalize to the population with only a few well-studied examples. Because most malware tries to stay hidden in the normal activity of the network, an infected machine will generally not influence either user or network administration and allocation policies. The population model is thus split into two parts: generation of malicious activity when machines are turned on and active, and allocation of malicious activity from the infected machine to the IP address that it will use to send its message out across the Internet.

Section 1.1 introduces botnets and motivates the reasons for studying botnet population size. Section 1.2 discusses the motivations for the generation and allocation model. Chapter 2 presents the generation and allocation model as a graph of unobserved machines linked to observed network touchpoints, and outlines broadly the estimation strategies to employ. Chapter 3 introduces the case study for the Conficker botnet, and describes the data set that was used in the case study. Chapter 4 presents the generation model for

the C variant of the Conficker worm and analyzes a directly observable population. Chapter 5 extends the generation model to an allocation model for a non-informative network–one where the allocation model does not provide any additional information about the population size or the machine-specific behavior–and analyzes several networks that can be considered as non-informative. Chapter 6 discusses the results and presents some preliminary thoughts for moving forward with simple informative network structures.

1.1 Botnets and size estimation

When it comes to measuring the threat of a botnet, the consensus among researchers and cyber-analysts is that size matters. Botnets–collections of infected computers that maintain communications to a single human administrator or small organized group–continue to be a relevant vehicle for malicious activity in the cyber landscape, causing damage through Distributed Denial of Service (DDoS) attacks, distributed large-scale sending of spam email, or other co-ordinated activities such as click-fraud and brute force password-cracking. While worry about catastrophic large-scale botnet attacks has lessened, businesses still listed targeted and sophisticated botnet-driven DDoS attacks as the number one concern for security coming into 2010 (Worth, 2010), and use of botnet-driven DDoS attacks is common against political, human rights, and independent media sites (Zuckerman et al., 2010), as evidenced by the recent DDoS attacks perpetrated by the Anonymous group against businesses refusing to support Wikileaks, and the 2011 attacks on the Livejournal infrastructure, theorized to be politically motivated as an attempt to silence Russian political bloggers on the site (Shuster, 2011). Botnets are a commodity in underground economies; they can be rented or sold among malicious actors for use and profit.

But what do we mean when we talk about botnet's "size", and why do we want to measure it? Intuitively, the larger the botnet network is, the more "zombie machines" are available at a controller's behest for causing damage. In popular literature such as blogs and security reports, the size of a botnet is mainly measured by the number of infected machines. This metric is generally what is used in the underground communities, and it has the advantage of an easy interpretation for communicating the scope of the botnet's threat to laypersons. An infected computer is something that can be visualized, catalogued and tabulated. Tracking the number of infections over time can yield information about clean-up measures such as patching systems against new infections, blacklists, or takedowns. Security analysts rely on population estimates to prioritize

threats and to measure the efficacy of clean-up strategies. But the uncertainty associated with estimating machine infections from network communication points is not well understood, and differences among published numbers are often substantial enough to leave analysts unsure of which numbers are realistic or trustworthy. Furthermore, population metrics inherently treat all individuals as equals, but it may be advantageous to target critical assets preferentially, or to use distribution of activity across observed network touchpoints vs. inferred machines to determine the type of countermeasures that would be most effective.

Botnets are feared as an attack vehicle specifically because of their ability to distribute large scale attacks widely across a large number of assets. A staple of defense against a large-scale attack from a botnet is to blacklist the botnet's network Internet Protocol (IP) addresses: to add them to a list of firewall rules so that any communication attempt from these addresses is dropped automatically without wasting more critical network and system resources. But a botnet is dynamic: if a central controlling machine is seized, a botnet's administrator can often stand up a new machine in its place, or propagate instructions through sideline peer-to-peer channels. This gives an attacker both flexibility to avoid network-level countermeasures, and resilience to move, grow and recruit new machines when faced with take-downs, even of central controllers (Leydon, 2010). Botnets also provide a layer of anonymity for their controllers, routing activity through unsuspecting third parties and acting as a buffer between malicious actors and malicious acts. In some cases, it is helpful just to inform network administrators of the extent of infection on their network, and a useful way to prioritize whom to contact is in terms of the extent of infection among devices on the network. A last resort strategy for combatting botnets is infiltration and take-down of a botnet's central command structure; this kind of effort can be difficult to co-ordinate among global law enforcement communities.

Several factors cloud the landscape when it comes to counting the number of infected machines in a botnet population. Unlike wildlife studies, where individual animals are directly observable, individual machines in a botnet are often visible only by their behavior. Physically, infected machines can be located anywhere in the world, but the malicious applications running on these machines need to communicate over the Internet–with each other, with a central controller, or with potential targets–in order for the botnet to survive as a co-ordinated large-scale threat. But behavior viewed through an IP address is not the same as directly observing an infected machine. Indirect observation also makes it difficult to track individual infections over time. And while infected machines are an intuitive measure of botnet size, a botnet's threat

is also related to the distribution of activity among its assets.

Enumeration of individual machines usually requires the ability to infiltrate the botnet and observe it "from the inside," as was done for example with botnets such as Storm (Holz et al., 2008), Torpig (Stone-Gross et al., 2009), and Waledac (Kang et al., 2009; Nunnery et al., 2010). Web-based working groups, that combine front-line mentality with both industry and academic contributions, are a staple of botnet research and collaboration that often drive botnet infiltration and monitoring. Well-established wide-range groups such as the SANS Internet Storm Center (2001) and the Shadow Server Foundation (2004) publish web-based reports on bots and botnets that they currently track. More recently, botnet-specific working groups have arisen, for example the Conficker Working Group (2009) and the Mariposa Working Group (2009). While researchers and working groups may have the resources to monitor a botnet from the inside, this avenue is often not available or is limited by legal restrictions for corporations or other security providers tasked with mitigating botnet threats.

The metric used for enumeration when individuals are indirectly observed is a count of IP addresses. Although botnets do produce behavior that can be seen at the network level, mobile devices as well as network policies such as Network Address Translation (NAT) and Dynamic Host Configuration Protocol (DHCP) complicate the relationship between machines and IP addresses. Security researchers and working groups often do not quantify the effect of measuring IP addresses on a size estimate meant to represent individual infections. For example, when the Mariposa botnet was seized in March 2010, Panda security officials reported that the botnet was associated with 13 million unique IP addresses. As reported in PC World online, "It's hard to pinpoint the exact size of the botnet from that number, but it appears to be one of the world's largest. Researchers studying the notorious Conficker botnet have linked it to half as many IP addresses."(McMillan, 2010). The Conficker working group also reports population size in terms of unique IP addresses, with a caveat that the number of machines may be anywhere from "25% to 75%" of the cited IP address counts [1], but the methods used to reach this conclusion are unclear.

Additionally, botnets are open populations; they can grow from new infections, churn through address space, or shrink over time due to cleanups. This leaves several alternatives for counting size, with different interpretations. Cumulative counts of all unique individuals (either IP addresses or machines) over time can

[1] See "Population Numbers" at `http://www.confickerworkinggroup.org/wiki/pmwiki.php/ANY/InfectionTracking`

give an indication of the spread or impact of a botnet over its history, but may not accurately represent the strength of a botnet at any particular point in time during its lifecycle. Indirect observation of individuals through IP addresses also makes tracking cleanup efforts more difficult, as it can be hard to distinguish between one long-lived mobile infection, and many short-lived static infections.

Rajab et al. (2007) provide the current standard for size measurement of botnets in the security research community. They define two different metrics for population size, implicitly measured across infected machines: a footprint size, that is a cumulative count of all infected individuals observed from the botnet's life, and active size, which measures the number of machines active in any given period of time. They focus on measurement when individuals can be directly observed, and outline several methods that have been successful in the past for enumerating individual infections. They do not quantify the effect of measuring IP addresses instead of infected machines on perceived botnet size according to either its footprint or active size measurements. Formally, individual botnets can be conceived as dynamic distributions of activity over the respective space of individuals (machines or IP addresses). The importance of activity-based measurement of botnets is well known, and security companies such as Sophos already measure overall botnet activity by the volume of spam output across a large number of recipients, but they do not track individual botnets (Vaugh-Nichols, 2010, parag. 3).

Estimating machine counts separately from IP address allocation can help guide defenders on both the takedown and blacklisting fronts. Understanding where the largest infections lie can help watchdog groups and service providers prioritize cleanups, both in terms of notifying administrators and in co-ordinating takedowns with law enforcement. Additionally, understanding a botnet's local neighborhoods–its relative size and spread among global assets–can help defenders to determine where to blacklist, what proportion of activity the blacklist contains, and how wide the blacklist should be to block machines that use ephemeral connections.

1.2 Developing and informing population models of network phenomena

1.2.1 Statistical population estimation

Traditional statistical population estimation is based on capture-recapture models and their extensions to a wide class of generalized linear models (Fienberg et al., 1999) that account for heterogeneity among individuals and capture methods. Past study of botnet populations has drawn on these methods, but has treated IP addresses as the directly observable individuals of interest, without generally accounting for the heterogeneity that arises from different numbers of machines residing behind those addresses. Observation based on network behavior, as occurs with botnet activity, is similar to counting animals based on tracks, song recording, photos (in which animals may or may not be confounded), or other indirect methods as opposed to directly tagging each individual.

Botnet behavior is quantifiable as a count or rate measured over each time period, and is visible up to aggregation. In that way, the model of observable behavior per time period is similar to a convolution of the underlying unobserved machines. The result is the need for a model where births, deaths, and heterogeneity are interpreted similarly to the traditional capture-recapture literature, but that also includes a hidden layer that explains the link between machines visible up to aggregation through their network touchpoints. The allocation of behavior among unobserved individuals is similar to the broad framework of finite mixture models, where observed behavior is attributed to an underlying set of groups with an unknown dimension.

Traditional capture-recapture methods

Capture-recapture population estimation is so named for its genesis in wildlife population studies. The simplest form, dating to Petersen (Petersen, 1896), requires repeated independent samples ($J = 2$) of a closed, homogenous population with the ability to mark and identify individuals. No births or deaths can occur between samplings, and the probability of sampling individual i is constant across all individuals. Table 1.1 shows an example. Capture profiles $\{y_{ij} : j = 1, 2\}$ for individuals are summarized as counts $m_{\mathbf{j}}$ in a 2×2 table, with the entry m_{00} equal to the (unknown) number of individuals not observed in either sample.

Let $+$ indicate a row or column summation. Peterson's estimate of the total population $n = N + m_{00}$ is

		In Sample 1?	
		No	Yes
In Sample 2?	No	m_{00}	m_{01}
	Yes	m_{10}	m_{11}

Table 1.1: Arrangement of counts for a 2-way simple capture-recapture study into a 2x2 table. The cell labeled m_{00} is the (unobserved) number of individuals that were not captured in either sample.

based on the fact that $\hat{p} = \frac{m_{11}}{m_{1+}}$ is an estimate of the proportion of marked individuals in the population, and thus as a plug-in estimate, $m_{+1} = \hat{p}n$. This leads to an estimate of the total population as:

$$\hat{n} = \frac{m_{+1}m_{1+}}{m_{11}}.$$

The natural extension to multiple independent samples ($J > 2$) for closed, homogenous populations followed in the early 20th century (Schnabel, 1938). For dependence among samples, Fienberg (Fienberg, 1972, 1980) introduced the use of the log-linear model, which was echoed by Cormack (1989). Let $\pi_j(\ell_i)$ be the probability of observing a capture profile $\ell_i, \ell_i \in [0, 1]$, associated with an individual i and a sampling period $j, j \in \{1, \ldots, J\}$. For J independent samples in a homogenous population, the probability of a full binary capture profile $\boldsymbol{\ell}_i = (\ell_{i1}, \ldots, \ell_{iJ})$ does not depend on the individual i, and can be written as a product of capture probabilities associated with each sampling period:

$$p(\boldsymbol{\ell}_i) = \prod_{j=1}^{J} \pi_j(\ell_{ij})$$

Aggregating across individuals, the expected cell count μ_ℓ of the 2^J-way table of then number of individuals m_ℓ with capture profile $\ell = \{\ell_i, \ldots, \ell_j\}$ can be written as a linear combination of parameter values for each sampling period on the log scale;

$$\log \mu_\ell = \sum_{j=1}^{J} u_j(\ell_j)$$

This model can be seen as a multiplicative model of main effects. Accounting for dependence among sampling periods equates to adding interaction terms into the log-linear model for the cell counts. This can account for situations where the probability of capture for one period is linked to the probability of capture

on other periods. For example in wildlife studies, an individual may become "trap shy" if caught in the first sampling period, and may be more difficult to catch in subsequent sampling periods. In the case of botnets and other Internet phenomena, sampling for spam emails and phishing scams is often done by collecting information from spam filters associated with many different email accounts, each with a different address. Dependence among sampling periods can equate to the fact that botnets tend to use a compromised email address as a seed, and then attempt to infect all of the addresses in that account's contact list. Lists of potential contacts are also bought and sold on the black market, inducing a dependency in the probability that a single email scam shows up in several different email in-boxes.

Multiple recapture is most widely used as a method to model more than one sampling period across an open population, or to model a closed population that is surveyed by multiple samples at the same time, for example in analyzing list data. Applications to list data first arose in the mid-20th century in the context of human health studies, where researchers had access to records from multiple institutions for tracking individuals. Sekar and Deming (1949) used multiple lists to track birth and death rates as well as registration. Multiple ordered sampling methods such as dual-system estimation and post-enumeration survey have a history of use in the US Census dating back to the 1940s for example with Shapiro (1949).

Heterogeneity of individuals is typically modeled as covariate information in a general GLM framework when groups or distinguishing features are known. Sekar and Deming demonstrate that population estimates based on aggregations across correlated sub-strata are biased toward an undercount of the true population. Seber (1982) provides general results describing bias as a function of correlation among capture probabilities of individuals, with the intuition that correlation of capture probabilities would generally be positive and would lead to consistent underestimation of populations, a particular form of Yule's association paradox (Yule, 1903). Kadane et al. (1999) formalize this argument by providing necessary and sufficient conditions under which this is the case.

In the case of heterogeneity due to unobserved differences, latent class or latent trait models are used. Darroch et al. (1993) present a multiple-recapture approach to heterogeneity of capture in the Census, using ordered repeated random samples to generate multiple views. They introduce the latent trait Rasch model (Rasch, 1980) to incorporate multiplicative heterogeneity in individuals and lists. Fienberg et al. (1999) explored the link between multiple list capture models, the Rasch model for heterogeneity of individuals,

and a hierarchical Bayesian formulation of the Rasch model that explicitly calculates posterior distributions for individual parameters θ_i and list parameters β_j.

Extensions of capture-recapture methods to open populations generally focus on experiments where the sampling periods themselves are ordered in discrete time intervals, and assuming births, deaths and migrations occur between samplings, for example in Schwarz and Arnason (1996). Dupuis and Schwarz (2007) present a model for population estimation in open, heterogeneous populations when heterogeneity can be modeled as a series of observable, nominal classes. In this case, posterior inference is based on multinomial models arising from categorical heterogeneous classes and the binary response associated with marked or unmarked animals at each sampling period.

Capture-recapture models have direct applications to the study of networking and Internet characterization. Bradlow and Schmittlein (2000) apply a Bayesian model of heterogeneous catchability in a closed population to explore the relative performance of six search engines, as well as to obtain population estimates of web pages characterized by key words. Briand et al. (2000) evaluate the use of capture-recapture models for estimating the number of errors or bugs in software applications. Chan and Hamdi (2003) use capture-recapture methods to estimate the extent of total network resources in queue management schemes for routers. They apply both a homogenous capture-recapture model and a heterogeneous model based on the Jackknife estimator (Burnham and Overton, 1978), that treats individual capture probabilities as nuisance parameters. In both cases, repeated samplings are modeled as independent draws over a closed population. Mane et al. (2005) use a capture-recapture method based on random walk sampling to estimate the number of nodes in a closed peer-to-peer network, assuming homogeneity among nodes, with motivations toward the study of open networks.

Indirect observation and behavioral modeling

In traditional capture-recapture models, the individuals of interest and the heterogeneity in capture probabilities are treated as directly observable. Heterogeneity in botnet behavior often arises on several scales. Within a single machine, behavior changes over time due to variations in continuous, latent behavioral traits such as scan rates. Though machines can be recorded as simply marked or unmarked in any sampling period, this marking loses information regarding the stochastic behavior that is used to identify individuals.

An individual y_i is often observed because it is performing some action $x_{it} \sim f$, where f is a pdf or pmf for the sampling period t that affects the probability of "capture", and x_{it} is correlated across sampling periods $t, t+1, t+2$, and so forth. This data admits a sparser multinomial model than the Jolly-Seber model described by Dupuis and Schwarz, due to a more complex set of observable categories, measured over a much shorter scale than traditional capture-recapture studies. And although infected machines share the same code base, network resources and availability can also cause individual differences in activity rates and therefore capture probabilities, akin to a random intercept model.

A model for population estimation of infected machines in a botnet also needs to adapt the traditional capture-recapture models to address indirect observations, particularly when behavior cannot be reliably linked to a single machine. Wildlife-based capture-recapture models do not generally address ambiguity in the identification of distinct animals, which arises when studying machines through a lens of IP addresses. Indirect observation in wildlife populations is associated mostly with photography and with natural signs of habitation. Photography is used for large animals who can be identified as individuals based on visual characteristics, such as patterns of stripes or scales, which allows for the direct application of traditional mark-recapture methods (Karanth and Nichols, 1998) or Bayesian modeling of uniquely identified individuals using multiple sources of information (Gopalaswamy et al., 2012). Acoustic detection focuses on determining presence or absence of a group or species as opposed to modeling the volume and pattern associated with one individual. For example, Armitage and Ober (2010) use machine learning techniques to detect the presence or absence of different species of bats based on acoustic recordings of their calls, but they do not attempt to link these calls to individuals for the purposes of population density estimation.

Heinemeyer et al. (2008) address the use of indirect observation through natural signs of habitation, in the form of animal tracks and scats. In traditional wildlife studies, these signs are used to determine prevalence of species in geographic areas; wildlife research has focused on using DNA markers to link these indirect signs back to individuals in order to use them in mark-recapture population studies. When surveys are standardized to control for outside sources of variation, the abundance of natural sign indices correlates to known population density and other indirect indices of population abundance. Additional data collection is required to inform the behavioral links between the indirect signs and the population. Capture-recapture models are augmented by methods such as line-transect photographic sampling, that helps link the density of

tracks to the number of animals that produced them. These indirect surveys require careful validation of the additional modeling assumptions in order to produce valid estimates. Hierarchical models and techniques are widely used in the community to account for spatial characteristics that lead to heterogeneity in capture-recapture models (Royle and Gardner, 2011), but they do not seem to be widely applied for building a stochastic model of variable behavior to represent an indirectly observed individual.

Links to finite mixture models

When the individuals are identifiable only by a behavioral pattern, estimating the size of the underlying population can be similar to estimating the number of underlying groups in a finite mixture model (McLachlan and Peel, 2000), where instead of having weighted sums across groups, one views the absolute sum of behavior aggregated from many identical (or nearly identical) individuals.

Suppose a population $x_1, \ldots, x_n$ consists of H groups of generative probability models, that cannot be observed directly, and suppose group h has density function $f_h(x; \theta_h)$. Denote the prior proportion of group h in the population as π_h, such that $\sum_h \pi_h = 1$. The distribution of an individual x, averaging across the unseen group assignments, has density function:

$$f(x_i) = \sum_{h=1}^{H} \pi_h f_h(x_i; \theta_h)$$

Estimation of mixture models is facilitated by data augmentation; the unobserved class assignments are explicitly modeled as draws of H-dimensional indicator vectors Z_i from a Multinomial$(\pi_1, \ldots, \pi_h)$ distribution. Estimation of individual group parameters θ_h is straightforward in the situation where Z_i is observed. Data augmentation can be used in the context of the Expectation and Maximization (EM) algorithm to find maximum likelihood estimates or posterior modes (Demptser and Rubin, 1977), or within a Markov Chain Monte Carlo simulation as a way to simplify complete conditional distributions (Tanner and Wong, 1987).

In the case of indirect observation of network phenomena, the H individuals are assumed independent, each generating a count y_h from an identical generation model $f(y_h; \theta)$. A set of J network interfaces are known and observed, and the observed count x_j is a sum of counts from one or more unseen individuals from the population of H total individuals. In this case a J-dimensional indicator vector W_h is drawn that

assigns y_h to a network interface, and

$$x_j = \sum_{h=1}^{H} W_{hj} y_j$$

The marginal distribution of the count x_j is an H-fold convolution g representing the sum of y_h for the individuals for whom $W_{hj} = 1$. When the count generations are Poisson-distributed, the conditional distribution of $\{y_h; W_{hj} = 1\}$ given the observed value x_j has the form of a multinomial (possibly with correlated mean structure) across the values $q = 0$ to $q = x_j$, where the probability π_{qj} is proportional to the density function $f(q; \theta)$, and arrangements $y_1, \ldots, y_h$ have positive probability only if $\sum_h y_h = x_j$.

Data augmentation methods can be used for assignment of aggregated data among a collection of individuals while preserving the summation; for example Park (2011) use a Bayesian method to distribute choice data among aggregated survey results. Richardson and Green (1997) present a Bayesian method for exploring the group structure in a mixture model within a Markov Chain Monte Carlo simulation that includes merging and splitting of adjacent groups. A similar method can be used for exploring the space of individuals and behavior, where individuals can be merged or split based on the relative likelihood of proposed arrangements of counts y_h among individuals. In the application to finite mixture models, groups are identifiable due to different centroids. In the case of individuals, identifiability of individuals is not as important as obtaining information about the population size. The allocation of counts y_h among individuals H is a facilitating data augmentation step for exploring the posterior distribution of H based on simpler forms of its complete conditional distribution. A chain may relabel unobserved individuals exchangeably, but the information regarding the number of individuals comes from a well-informed behavioral model for a single host.

1.2.2 Informing a behavioral model

Applying traditional mark-recapture models to machines, as opposed to IP addresses, requires a model that links behavior by address to behavior by machine. On the other hand, models for directly counting IP addresses include heterogeneity among individuals that is informed by the behavior of a single machine; for example, an address shared in parallel by 100 infected computers is observed if at least one of its underlying hosts is observed, whereas an address leased serially to a network of one infected machine and 99 clean

machines is observed only if it is currently allocated to the infected machine, and the machine's behavior is observed. For either case–measuring heterogeneous IP addresses or directly measuring machines–the natural place to start modeling is with the behavior of a single infected machine. Without profiling information for typical machine behavior as compared to typical network behavior, it is nearly impossible to distinguish machines from their network touchpoints.

There are two ways that network defenders learn about malware: reverse engineering and dynamic analysis. Reverse engineering focuses on learning about the static code that infects computers, akin to performing a dissection in order to learn about the inner workings of the virus. Most malware code is encrypted and obfuscated, and the process of reverse engineering unpacks and pulls apart the binary executable code, finding the starting pointers and functions that define the code's architecture. Dynamic analysis focuses on setting up virtual testing environments, often called *sandboxes*, in which to run the malicious code and to see how it attempts to compromise the system and to communicate with the outside world. Both kinds of analyses produce very detailed accounts of how the malware works in order to develop signatures for detecting infections on a host level. But because network security is more focused on front line operations, these detailed descriptions are rarely used as the basis of stochastic behavioral models applied to large populations.

Both static and dynamic analysis focus on detection of an infection on an active, compromised machine. Behavioral profiles built from these analyses do not take into account the day-to-day activity and individual differences that arise in active botnets due to user behavior or network administration policies. However, day to day activity profiles are built in industry tools that specialize in Network Behavioral Anomaly Detection (NBAD) for baselining normal network activity [2]. With information from both, it is useful to split a population model into a generation model and an allocation model. The generation model describes the information gained from reverse engineering and sandbox testing, as well as possibly incorporating a model of active vs. inactive periods due to the user profile of a machine–for instance, a work desktop may only be turned on during business hours, while an email server would be shut off only for rare maintenance periods, and a home computer may only be turned on once per week. The allocation model is informed by typical network administration policies and historical baseline and user population studies. Both kinds of studies can provide the strong prior information needed to tease apart machines from network allocations.

[2] examples include McAfee's Network Threat Behavior Analysis, Juniper STRM, Plixer Scrutinizer, Enterasys Dragon, and Sourcefire 3D, among others.

Chapter 2

Modeling Generic Machine Populations by Countable Behaviors in a Network

Viewing a population of machines through a network telescope is like looking at an image through a kaleidoscope: individuals can seem distorted, masked, or fractured.

This chapter outlines the basic data structures, estimation strategy and model diagnostics for inference about the behavior of machines as observed through the lens of IP addresses. Section 2.1 defines some terminology for network protocols and Internet address space that are used throughout this document. Section 2.2 introduces a graph structure framework for modeling network allocations as a combination of traffic generation and allocation processes, and describes several forms of sub-models that arise in real networks. Section 2.3 presents an overview of Bayesian estimation and Markov Chain Monte Carlo (MCMC) strategies that can be used for estimation of the unknowns in the graph-based models. Section 2.4 presents a general strategy for application of MCMC to the generation and allocation models.

2.1 Network protocols

A set of well-known instructions and rules that defines how software applications communicate with each other is called a protocol. Network protocols are the space of directly observable behavior from infected machines in a botnet. Whether outsiders can detect control channel or peer-to-peer communications between

infected machines, or observe large-scale attacks from the botnet to an outside target, such as DDoS attacks or the sending of bulk spam email, the behavior is catalogued by development of a signature based on network protocols.

Similar to most non-malicious applications, malicious programs mainly communicate using either the Transmission Control Protocol (TCP), or the User Datagram Protocol (UDP). These protocols all also rely on Internet Protocol version 4 (IPv4), that defines the recognizable IP address for network communication. In the remainder of this document, any reference to IP addresses is implicitly associated with IPv4, and not with the definition of IP addresses associated with the newly emerging protocol Internet Protocol version 6 (IPv6).

An IP address is a 32-bit integer that identifies a machine to others in a network, for the duration of a connection. IP addresses are generally represented in the dotted decimal format of four octets, which are unsigned 8-bit integers ranging from 0 to 255 (for example, "192.168.12.2"). In any network connection, the initiator (or source) specifies its own IP address, and specifies an intended target (or destination) IP address with which to establish a connection.

Both TCP and UDP protocols also require that a port is associated with both the source and destination IP addresses in a network connection. A port is an integer between 0 and 65535 that is used as a channel to disambiguate individual connections between the same communicating hosts.

Often, network traffic monitors cannot identify infected machines directly, but they can identify IP addresses, protocols and ports through which infected machines are communicating. As with people and "snail mail" addresses, the map between machines and IP addresses is not one-to-one. Device mobility can arise from the physical change in location; users switch connectivity of devices such as smart phones or laptops among different network resources over the course of a day. In addition, two widely adopted network administration practices complicate the relationship between IP addresses and individual machines even in static network settings:

- Network Address Translation (NAT, many-to-one): A network of many machines is configured to access the Internet through a single connection point with one external-facing IP address, often called a gateway or proxy. Gateway traffic is also often shuffled among two or more IP addresses over time to balance bandwidth across several assets, a technique known as load-balancing.

- Dynamic Host Configuration Protocol (DHCP, one-to-many): a machine leases an IP address temporarily through an Internet Service Provider (ISP), which often has a pool of available addresses. Depending on the network configuration, DHCP leases can be valid for hours or days. One-day leases are common. Ephemeral machines which spend significant time turned off or disconnected from the network are more likely to receive a different IP address upon re-connecting.

Networks of IP addresses are usually allocated in contiguous blocks to entities such as companies, countries, academic institutions or Internet service providers. These blocks can range widely in size. A $/N$ net block (read "slash N net block") is the collection of addresses obtained by fixing the first N bits of an IP address. Often, net blocks are sized by octets. A $/24$ net block, denoted for example by $192.168.12.0/24$, collects together the 256 IP addresses in the last octet range 0 through 255. A $/16$ net block, denoted for example by $192.168.0.0/16$, collects together 256^2 or 65536 IP addresses.

In the remainder of this document, the labels for observed data for $/24$ net blocks are reported only for the third octet; the first two octets are anonymized to "$100.100.$" to maintain privacy of the owners of infected networks.

2.2 Representing machines and network connections in a graph

In the general setup of a population study, H machines communicate through J network interfaces, for example IP addresses, over a period of time. Communications from machines are observed as a count of activity over discrete time periods $t = 1, \ldots, T$; at any point t, a machine i produces a count y_{it}. This count can also be subdivided into counts that are divided up among network interfaces. Let w_{ijt} be the count at time t originating from machine i and sent through network interface j, such that

$$y_{it} = \sum_{j=1}^{J} w_{ijt}.$$

Each network interface j also has an associated aggregated count x_{jt}, where

$$x_{jt} = \sum_{i=1}^{H} w_{ijt}.$$

The number of IP addresses used to transmit y_{it} in the time period t is given by

$$J_{it} = \sum_{j=0}^{J} 1_{\{w_{ijt}>0\}}$$

The collection $\mathbf{W}$ of host-to-network counts $\{w_{itj} > 0; i = 1 \dots H, t = 1 \dots T, j = 1 \dots J\}$ can be envisioned as a weighted graph linking individual machine activity to network interfaces over time. Inference about the system comes from deriving a model $Pr(w_{ijt})$ for $w_{itj} \in \mathbf{W}$. In practice in botnet studies, J and $x_{jt}, t = 1 \dots T$ are observed quantities, while H, y_{it} and w_{ijt} are unobserved but informed by static analysis and network administration policies. Define a machine i and network j as *associated* over the time period $t = 1 \dots T$ if j exists as a possible communication point for i when it is active. For any network j and machine i that are not associated, $Pr(w_{ijt} > 0) = 0$ for all t.

2.2.1 Generation and allocation

The graph is modeled as a "generate and allocate" system, where the individual machine generates a count y, and then y is allocated among the set of J potential interfaces. The generation can be modeled by a suitable discrete, non-negative distribution, and the allocation can be modeled as splitting y multinomially among possible interfaces with some suitable probability vector. For a machine i and time t, let $\mathbf{w}_{it}$ be the collection $\{w_{ijt} : j = 1 \dots J\}$, and let $\mathbf{W}_{1,..t-1}$ be the history of the system up to time $t-1$, with a set of system parameters θ. The joint likelihood for a single count y_{it} and its associated allocations $\mathbf{w}_{it}$ is written as the product of the probability mass functions for generation of y_{it} and the allocation conditional on y_{it}:

$$\begin{aligned} Pr(y_{it}, \mathbf{w}_{it}|\mathbf{W}_{1,\dots,t-1}, \theta) &= Pr(y_{it}|\mathbf{W}_{1,\dots,t-1}, \theta) Pr(\mathbf{w}_{it}|y_{it}, \mathbf{W}_{1,\dots,t-1}, \theta) \\ &= f(y_{it}|\mathbf{W}_{1,\dots,t-1}, \theta) \frac{y_{it}!}{\prod_{j=1}^{J} w_{ijt}!} \prod_{j} [g_{ijt}(\mathbf{W}_{1,\dots,t-1}, \theta)]^{w_{ijt}}. \end{aligned} \quad (2.1)$$

Here, $f(\cdot)$ is the density function of the generating process for y_{it}, and the sub-counts enter into the likelihood through the Multinomial probability density function, parameterized by a total equal to y_{it} and a collection of J allocation probabilities. Like a traditional generalized linear model framework (McCullagh and Nelder, 1989), the function $g(\cdot)$ is a suitable polytomous link function of the graph history and system parameters, that describes the probability of allocating one instance of activity to each network touchpoint.

When the time window t is small, it is likely that $g_{ijt}(\mathbf{W}_{1,\ldots,t-1}, \theta) \approx 1$ for a single network interface j.

Botnets may employ peer-to-peer communications for transferring orders, but in executing orders for large-scale scanning or spamming, machines can generally be modeled as acting independently from each other, conditional on some shared resources in local networks. This independence assumption suggests that the generation process for a single machine can depend on past values from that machine, but usually does not depend on counts from other machines, or on allocation of counts once they are generated. Using this simplification, the joint likelihood of a graph $\mathbf{W}$ of machine allocations for a collection $\mathbf{x} = \{x_{jt} : j = 1 \ldots J; t = 1 \ldots T\}$ of observed network counts generated by independent machine activity can be written as the product of the machine-specific terms given in Equation 2.1:

$$\ell(\mathbf{W}, \theta; \mathbf{x}) = \prod_{t=1}^{T} \prod_{i=1}^{H} f(y_{it} | \mathbf{W}_{1,\ldots,t-1}, \theta) \frac{y_{it}!}{\prod_j w_{ijt}!} \prod_{j=1}^{J} [g_{ijt}(\mathbf{W}_{1,\ldots,t-1}, \theta)]^{w_{ijt}}. \tag{2.2}$$

In a population study, the joint posterior distribution $p(H, y, \mathbf{W}, \theta \mid \mathbf{x})$ is proportional to the product of Equation 2.2 and the prior distributions for H and θ:

$$p(\mathbf{W}, \theta \mid \mathbf{x}) \propto \pi(H)\pi(\theta)\ell(\mathbf{W}, \theta; \mathbf{x}) \tag{2.3}$$

The assumption of independent priors for H and θ implies that, in terms of the generation model, behavior of an individual in the botnet is not linked with the overall size of the botnet. In other words, the individual is generally not aware of the whole. Botnet propagation relies on infecting new computers with static software, and it is this static software that drives the behavior of individuals. As the botnet grows, large behavioral changes are typically phased in via software updates from the botmaster, similar to new "version releases" of non-malicious software, but these changes would follow more of a punctuated equilibrium model, that would describe the motivations and goals of the botmaster. For the allocation model, the independence of H and θ assumes that allocation of activity among network touchpoints does not change with the size of the botnet. This is generally true, as most networks housing infected machines are unaware that the machines exist. As a botnet grows in activity or number of machines, administrators may recognize problems and move to balance resources or to clean up infected machines, but this process is related more to modeling of take-downs and machine deaths than *a priori* ideas about the overall ability of a network to

be infected depending on its allocation policies.

Equation 2.3 makes use of conditionally independent machine activity as well as independent prior assumptions on the population size and the system parameters. Section 2.2.2 examines reasonable simplifications of this general model.

2.2.2 Reducing the full model

Network connectivity is often transparent to the end user. As long as an active network interface is available, user behavior or malicious software activity does not change its characteristics. For example, most non-administrative users who connect to their ISP, university, or corporate network in order to browse to `www.google.com` are not cognizant of whether their Hypertext Transfer Protocol (HTTP) sessions are sent directly from their client IP address, or are first rerouted through an HTTP proxy server. Changing the HTTP proxy IP address also would not affect the look or feel of a client web session. But IP address allocation can follow a user's physical location throughout the day, if they connect a mobile device like a laptop to different networks such as a home network, work or university network, or a wireless hotspot.

Allocation of resources within an autonomous network is generally independent of machine activity except for two cases:

- Load balancing: traffic is switched to a new proxy (and possibly IP address) when the bandwidth for the current proxy exceeds some threshold. In these case, the network policy depends on the overall volume of traffic going though the network touchpoint.

- Machine shut-down/start-up in DHCP pools: When a machine's DHCP lease expires, it is typical for a DHCP server to reassign the machine its previous IP address if that address is available in the pool. When the number of machines competing for addresses exceeds the number of available addresses, it is more likely that a machine's IP address is unavailable, if that machine has been turned off for the expiration of its lease. Thus machines that start up after long periods of shut-down time (longer than the DHCP lease time of the network) may be more likely to be reassigned arbitrary addresses in the pool, and thus appear to "travel" around IP space.

These relationships can be modeled explicitly, but they are rarely directly observable to network telescopes. Covariates, such as time of day, may sufficiently capture the information from these interactions

between generation and allocation. For simplification, it is assumed that the allocation process depends only on the counts y generated from the machines, and not on the parameters for the generating process. In this case, system parameters can be divided into $\theta = (\xi, \phi)$, where ξ are the generation parameters and ϕ are the allocation parameters.

Even in the case where botnets all share a common set of malware (and thus behaviors for the generating process), network configurations and speeds, as well as differing machine hardware and software capabilities can lead to individual differences in the generating process $f(y_{it})$. In modeling, the prior $\pi(\theta)$ can be thought of as a population distribution, possibly dependent on another layer of hyperparameters κ, which generates a machine specific parameter $\xi_i \sim \pi(\xi \mid \kappa)$, so that the generation process can be written as $f(y_{it}|\xi_i, y_{i1} \dots y_{it-1})$, and a network specific parameter ϕ_j so that the allocation functions can be written as $g_{ijt}(\mathbf{W}_{1,\dots,t-1}, \phi)$, where $\mathbf{W}_{1,\dots,t-1}$ is the history of all connections w_{ij} up to time $t-1$.

Substituting these constraints into the generation function $f(\cdot)$ from Equation 2.2, the reduced model posterior distribution has the form:

$$\begin{aligned} p(\mathbf{W}, \theta \mid \mathbf{x}) &\propto \pi(H)\pi(\kappa)\pi(\phi \mid \kappa) \\ &\times \prod_{i=1}^{H} \pi(\xi_i \mid \kappa) \prod_{t=1}^{T} f(y_{it}|\xi_i, y_{i1}, \dots, y_{it-1}) \frac{y_{it}!}{\prod_j w_{ijt}!} \prod_j [g_{ijt}(\mathbf{W}_{1,\dots,t-1}, \phi_j)]^{w_{ijt}} \end{aligned} \tag{2.4}$$

Under this format, the dependency between generation and allocation at time t relies only on the count y_t. This is a reasonable simplification for many network phenomena, that also simplifies the estimation scheme for the graph. Section 2.2.3 introduces activity profiles for the generation process in this framework. Section 2.2.4 presents some common allocation methods in increasing order of complexity. Section 2.4 outlines the general estimation strategy for model parameters using Markov Chain Monte Carlo (MCMC) methods.

2.2.3 Generation models for open populations

In order to obtain both footprint and active size measurements of a botnet over its history, it is necessary to keep track of machine availability. Malware rarely controls the physical aspects of a computer, such as whether it is turned off or on, and information about malware behavior is naturally predicated on the

machine having an active network connection and being turned on. It is useful to model the activity profile and generation processes separately because the information informing priors on single-host behavior is often gained via sandbox testing, which provides data on the behavior of the infection but not on the day-to-day activity patterns of an infected machine. The prevalence of various network activity profiles in a collection of infections depends on both the distribution of typical activity profiles across the Internet, and on malware propagation techniques, targets, and software requirements that might bias a population to favor certain profiles over others.

For open populations, at time t any machine in the population can be considered as belonging to one of four discrete states:

1. Not yet infected (*);
2. Infected and active;
3. Infected and turned off;
4. Patched and cleaned of infection (†).

A "birth" in the population occurs when a machine transitions from the uninfected state (*) to one of infected and active. A "death" occurs when a machine transitions from either infection state to patched and cleaned of infection (†). In between the birth and death states, an infected machine can be either switched on or switched off. A machine that is switched off at time t has $Pr(y_t = 0) = 1$, while a machine that is switched on follows the generation model $f(y)$.

2.2.4 Some specific allocation models

This section introduces allocation models in increasing order of complexity required for updating the unobserved population size H and the edge links w_{ijt}. The order is:

1. One to one allocation;
2. Non-informative network allocation;
3. Independent informative allocation;
4. Dependent informative allocation.

One to one allocation: In the very simplest case, a machine i is statically allocated to a single IP address j, with $g_{ijt}(\cdot) = 1$ for all t, and $g_{ikt}(\cdot) = 0$ for all $k \neq j$ and all t. This allocation structure defines a directly observable population. Essentially, H is known and is equal to J, and $w_{ijt} = y_{it} = x_{jt}$. This is the trivial case where the generation parameters ξ and network population parameters κ can be updated through traditional MCMC methods. Often, small sub-populations of a large botnet can be labeled as one-to-one allocations *a priori* with high probability, and the resulting posterior inference on ξ can help to inform the generation model "in the wild" (including individual differences arising from network heterogeneity) as opposed to solely using sandbox tests.

Non-informative network allocation: A non-informative network allocation is one for which the allocation function $g_{ijt}(\cdot) \propto c$, an equally likely constant value across all network touchpoints j for any generated activity $\mathbf{W}_i$ at any time t. In this case the conditional distribution of H and y_{it} given ϕ, w, and x_{jt} depends only on $\sum_j x_{jt}$. An example of a non-informative network allocation arising in practice is a persistent "many to one" allocation scheme, for example a NAT address that is known never to change. A population can also be comprised of collections of non-informative networks based on strict boundaries, for example a collection of static NATs where it is known that machines never migrate between addresses. Another example (that does not generally arise in real world situations) is a DHCP pool where every address is equally likely to be assigned to any host at any time. In these cases, the only information available for inference on H is through the generation model and the behavioral profiles of host availability, and any conditional distributions involving H or w depend only on the generation model $f(y_{it}|\xi_i, y_{i1}, \ldots, y_{it-1})$.

Under non-informative network allocations, the Levy central limit theorem can be used in the simplest case (or with an appropriately marginalized generation model) to obtain active size estimates for each sampling period without requiring simulations of graph edges w (Weaver, 2010b, for example). But it takes more work to understand the footprint size and to track the life of individual machines.

Independent informative network allocation: In this case, the count allocations are informed by both sides of the generation and allocation model. There is some kind of movement distribution that describes how likely it is for a host to switch IP addresses given its prior allocations. A count of active IP addresses is often available, and the population estimate is informed by the fact that with DHCP pools, for example, it

is comparatively unlikely for a machine to split its allocation across two IP addresses within an hour, or for more than one machine to use a single IP address within an hour.

Dependent informative network allocation: This is the case when allocations among multiple machines switch at once, for example a switch *en masse* of all hosts behind one gateway or proxy to another as a result of load-balancing.

2.3 Posterior inference and Markov Chain Monte Carlo estimation

2.3.1 Overview

Suppose data y exists that informs a parametric model with a set of parameters θ. Let $\ell(\theta; y)$ be the likelihood of the data when θ is fixed, and describe the prior beliefs of θ by a distribution $\pi(\theta)$. The parameter θ can be multi-dimensional, and it can also include unobserved quantities from different levels of a hierarchical model. The functional form of the likelihood $\ell(\theta; y)$ and the parameter θ together comprise a set of unobserved but theorized quantities, that can be interpreted as anything from a useful way to explain data variations and to assess confidence and make predictions, to an interpretable generative model of the processes by which y was created (Cox, 1990). The data y consists of directly observable quantities.

Bayesian inference is based on understanding the posterior distribution of the parameters θ in light of the prior beliefs $\pi(\theta)$ and the likelihood $\ell(\theta; y)$. Call this distribution $p(\theta \mid y)$. By Bayes rule, the posterior distribution is defined as

$$p(\theta \mid y) \quad = \quad \frac{\pi(\theta)\ell(\theta; y)}{\int_\theta \pi(\theta)\ell(\theta; y)} \tag{2.5}$$

The posterior distribution can be used to make joint, conditional or marginal probability statements about components of θ, to produce point estimates–such as means, modes, or quantiles–that have well-understood large-sample properties, and to produce credible intervals in order to describe uncertainty.

The denominator integral in Equation 2.5 is a function of y alone, and is known as the marginal distribution of the data, or $m(y)$. In addition to being interpretable as a relative measure of fit for the model (Kass and Raftery, 1995), $m(y)$ is the normalizing constant for the posterior distribution, as it ensures that

the distribution integrates to 1. In some simple cases, the integral can be computed analytically, and the posterior distribution has a closed form. When the normalizing constant cannot be computed analytically, there are several options for pursuing posterior inference:

- Numeric integration: approximate $m(y)$ by evaluating the numerator of Equation 2.5 on a grid of values of θ (more useful when θ is low-dimensional)
- Approximation: posterior densities or posterior expectations can be approximated using closed form distributions, for example the Laplace (Tierney and Kadane, 1986) or Saddlepoint (Daniels, 1954) approximations based on Normal distributions.
- Monte Carlo sampling: bypass the need to calculate $m(y)$ by instead obtaining a sample $\theta^1 \dots \theta^M$ drawn from $p(\theta \mid y)$ using a method that does not require normalization. Approximate the expected value integral $E[g(\theta)] = \int g(\theta)p(\theta \mid y)$ of any function $g(\theta)$ using the monte carlo estimate:

$$\widehat{E[g(\theta)]} = \frac{1}{M}\sum_{m=1}^{M} g(\theta_m)$$

  Quantiles, modes and other functions of $p(\theta \mid y)$ can also be approximated by their sample quantities, using resampling techniques (for example bootstrapping) to estimate variability.

Monte Carlo methods require use of a distribution $q(\theta)$ that is easy to sample from and in some way "close" to the target distribution $p(\theta \mid y)$. Random deviates $\theta_1 \dots \theta_M \sim q(\theta)$ are generated and then further transformed to represent draws from the target distribution. The transformations can be iterative or non-iterative. Examples of non-iterative Monte Carlo methods that do not require calculation of $m(y)$ are rejection sampling (also called the acceptance-rejection method or simply the rejection method), which requires only the calculation of the posterior mode up to a constant C to ensure that $Cq(\theta) \geq m(y)p(\theta \mid y)$ for all θ ($q(\theta)$ covers the target distribution), or importance sampling, which uses ratios of the sampling and target distribution. Importance sampling does not require that $q(\theta)$ cover the target distribution, but importance sampling estimates can suffer from high variability if the importance ratios $[q(\theta)]/[m(y)p(\theta \mid y)]$ are very large.

Markov Chain Monte Carlo (MCMC) is an iterative method that is useful for obtaining samples from high-dimensional parameter spaces. In MCMC, sampling from the target distribution $p(\theta \mid y)$ is obtained

by constructing a Markov chain with a stationary distribution equal to $p(\theta \mid y)$. A distribution π over a set of states is stationary for a transition matrix P if it satisfies the equation $\pi = \pi P$, that is, the distribution $\pi(X_n \in A)$ does not depend on n.

For MCMC sampling, an initial state θ^0 is set, and successive sampling of the next value θ^{m+1} is performed conditional only on the most recent value θ^m. The chain is let run until the influence of the initial starting state is negligible and $\theta_m, \theta_{m+1}, \ldots$ are drawn from the stationary distribution. MCMC uses Metropolis-Hastings (M-H) sampling to achieve the correct form of the stationary distribution. The general M-H step uses a proposal distribution $g(\cdot \mid \theta^m)$, based on the current state θ^m of the chain, to produce a candidate value θ^*. The candidate is accepted as θ^{m+1} with probability

$$r \quad = \quad \min\left(1, \frac{\ell(\theta^*; y)\pi(\theta^*)g(\theta^m \mid \theta^*)}{\ell(\theta^m; y)\pi(\theta^m)g(\theta^* \mid \theta^m)}\right)$$

If the candidate is rejected, then θ^{m+1} is set equal to θ^m. It is important to choose a proposal distribution that allows for efficient exploration of the posterior distribution; if candidates are chosen too far from the current value, the chain may spend many iterations rejecting candidates before it moves to a new value. On the other hand, if candidates are chosen too close to the current value, the chain will accept nearly everything but may take a long time to explore the space. An efficient M-H chain should show good mixing, with the proportion of accepted candidates generally around 0.60 to 0.70.

2.3.2 Detailed balance and reversible jumps

Suppose a Markov chain θ^n at step n has pdf $\pi(\theta)$, such that for any set A, $Pr(\theta^n \in A) = \int_A \pi(\theta)d\theta$, and time independent transition distribution $P(\theta' \mid \theta) = Pr(\theta^{n+1} = \theta' \mid \theta^n = \theta)$. The Markov chain is *reversible* if for any two states θ' and θ, the property holds that

$$\pi(\theta')P(\theta \mid \theta') \quad = \quad \pi(\theta)P(\theta' \mid \theta).$$

When the chain is reversible, the distribution $\pi(\theta)$ is invariant. In order for the MCMC chain to be reversible, it must maintain this detailed balance. Often the distribution $P(\theta' \mid \theta)$ is written as a transition kernel $p(\theta, \theta')$ that maps θ to θ'. To help recall the directions (as this notation is parsed in the opposite direction

of conditional probability statements), I use the kernel notation of $p(\theta \rightarrow \theta')$. In MCMC, this kernel is comprised of the product of the jumping distribution $g(\theta \rightarrow \theta')$ and the acceptance ratio $\alpha(\theta \rightarrow \theta')$.

By using well-crafted proposal distributions, MCMC chains built with Metropolis-Hastings steps are reversible with respect to the posterior distribution $p(\theta \mid y)$, which ensures that $p(\theta \mid y)$ is the invariant distribution of the chain. But when the dimensionality and format of the parameter space θ is one of the uncertain quantities in the model, it is not immediately clear how to construct a proposal distribution that links parameters in one space to interpretably "close" parameters in another, while also maintaining detailed balance. This change in dimensionality arises for example when the model is a mixture distribution with an unknown number of components, which requires a proposal distribution that can accommodate dimensional jumps (for example, adding a new group and its parameters, or removing a group). If standard proposal distributions, such as Normal distributions centered at the current value of the parameter vector, are used, the MCMC chain becomes reducible amongst the sub-chains corresponding to different sets of dimensionality. On the other hand, parameters should maintain deterministic interpretability across dimensions; when two groups are merged, the mean value of the new group should be close to the average of the current means weighted by the prior group proportions.

The solution is to augment the proposal distributions to include proposing new steps that link these different spaces in interpretable or useful ways. But in order to keep the posterior distribution $p(\theta \mid y)$ as the invariant distribution of the chain, detailed balance should be maintained. To achieve this, Green (1995) outlines a method called Reversible Jump Markov Chain Monte Carlo (RJMCMC).

Suppose a model space $\mathcal{S}$ can be partitioned into sub-spaces $\mathcal{S}_k, k = 1, \cdots, K$, possibly with different dimensions or interpretations of the parameters θ_k. To traverse the entirety of $\mathcal{S}$, different types of proposal moves are required for exploring each $\mathcal{S}_k$ and for traveling between sub-spaces $\mathcal{S}_k$ and $\mathcal{S}_j$, $k \neq j$. Let $g_m(\theta \rightarrow \theta')$ be a move type that proposes to take the state θ to θ', that is accepted with probability $\alpha_m(\theta \rightarrow \theta')$. It can be shown that for detailed balance to hold between arbitrary states θ and θ', it is sufficient to prove that it holds for each move type, that is:

$$p(\theta \mid y) g_m(\theta \rightarrow \theta') \alpha_m(\theta \rightarrow \theta') \quad = \quad p(\theta' \mid y) g_m(\theta' \rightarrow \theta) \alpha_m(\theta' \rightarrow \theta),$$

for each θ, θ', and m. Traditional Metropolis-Hastings steps can be used for within-subspace moves, and

it suffices to provide cross-subspace moves that maintain detailed balance across any transformations that are used to map θ to θ' when they reside in different subspaces. This is done via "dimension matching": using a joint, common dominating measure across the subspaces that puts positive probability only on valid moves in both directions. For example if $\mathcal{S}_1 = \mathcal{R}^1$ and $\mathcal{S}_2 = \mathcal{R}^2$, with parameters θ and (θ_1, θ_2), a typical cross-subspace jump from $\mathcal{S}_2$ to $\mathcal{S}_1$ at step n may be the function $\theta^{n+1} = h(\theta_1, \theta_2) = \frac{1}{2}(\theta_1^n + \theta_2^n)$. Thus for the move to maintain detailed balance, the reverse jump from (θ_1, θ_2) in $\mathcal{S}_2$ to θ in $\mathcal{S}_1$ needs to put positive probability only on the set $\{(\theta_1, \theta_2) : \frac{1}{2}(\theta_1 + \theta_2) = \theta\}$. This is achieved, for example, by generating a random variable u with distribution $g(u) = U(0, 1)$ and setting $(\theta_1^n, \theta_2^n) = h^{-1}(\theta, u) = (2\theta u, 2\theta(1 - u))$.

The acceptance ratios still depend on the proposal distributions g in their respective spaces, but must be multiplied by the determinant of the Jacobian of the variable transformation h that takes $(\theta', u1)$ to $(\theta, u2)$ (and vice versa) in either direction, in order to account for the ratio of the two densities. In general, when variables u_1 and u_2 are used to match dimensions between $\mathcal{S}_k$ and $\mathcal{S}_j$ with proposal distributions $q_k(u_1)$ and $q_j(u_2)$ that map θ to θ', detailed balance is maintained with the acceptance ratio

$$\alpha_m(\theta \to \theta') = \min\left\{1, \frac{p(\theta' \mid y) g_k(u_1)}{p(\theta \mid y) g_j(u_2)} \left| \frac{\partial h(\theta', u_1)}{\partial(\theta, u_2)} \right| \right\}.$$

The reciprocal move is used for the jump from $\mathcal{S}_j$ to $\mathcal{S}_k$. The jumps are designed in tandem, so that the reverse jump from θ' to θ uses the function h^{-1}. In practice, the dimension of one of u_1 or u_2 is 0 when parameters are shared across models using deterministic transformations. However, when parameters are not shared, the dimension matching can allow for more degrees of freedom. For example, a sampler jumping between a merged state with parameter θ and two split states with parameters θ_1, θ_2 can match dimensions across a space $(\theta, u) \to (\theta_1, \theta_2)$ with only a single degree of freedom, or it can match dimensions across a space $(\theta, u_1, u_2) \to (u, \theta_1, \theta_2)$. The first case can be used to preserve exact transformation between θ and θ', while the second case can be used to relax the constraint, for example by enforcing the map between h and h^{-1} in expected value.

2.3.3 Blocking and sweep strategies

MCMC is useful for high-dimensional and variable-dimensional problems because Metropolis-Hastings steps can be applied to individual components, to lower-dimensional sub-blocks, or to different config-

urations of θ when it is multidimensional, without changing the limiting distribution of the chain. If $\theta = (\theta_1, \ldots, \theta_K)$ is the vector-valued parameter, the *full conditional distribution* (Gilks et al., 1996, ch 1.4.1) of the $k - th$ component, denoted by $\pi(\theta_k \mid \theta_{-k}, y)$, is the distribution obtained by treating $\theta_{-k} = (\theta_1, ...\theta_{k-1}, \theta_{k+1}, \ldots, \theta_K)$ as constants and normalizing the quantity $\ell(\theta; y)\pi(\theta)$ as a function of θ_k alone. When the full conditional distribution is used as the proposal distribution $g(\theta_k \mid \theta_{-k}, y)$, then the ratios in Equation 2.6 cancel and the acceptance ratio is always equal to 1. This is known as a Gibbs step.

The term *data augmentation* usually refers to the practice of expanding a distribution $\pi(\theta \mid y)$ that is costly to evaluate by a set of unobserved values η such that $\pi(y, \eta \mid \theta)$ is easy to write down (Tanner and Wong, 1987). Traditionally, the unobserved values η are strictly a device for ease of computation, but the method is unchanged if these values represent quantities of interest. Data augmentation arises for example in mixture models where the class assignments are unknown. Taking expectations of the expanded likelihood across unobserved variables and iteratively refining maximum likelihood or a posteriori estimates produces the Expectation and Maximization (EM) algorithm, while in Markov Chain Monte Carlo sampling, the unknown values η enter in to the chain as another hierarchy of parameters to simulate using full conditional distributions, and the expanded form of the likelihood makes each Metropolis or Gibbs step easier to compute. The expanded full conditional distributions $\pi(\eta \mid \theta^m, y)$ and $\pi(\theta \mid \eta^m, y)$ may also lack closed form expressions, but they are often easier to sample from than the unexpanded conditional distributions, and they can facilitate smaller-dimensional steps and better mixing in a chain.

Updating sub-components of a k-dimensional parameter θ can be done either deterministically or randomly. In deterministic updating, one round of component-wise or block-wise updating of $\theta_1 \ldots \theta_k$ is considered as a single "sweep" of the Gibbs or Metropolis-within-Gibbs algorithm, and each sweep is counted as an iteration of the Markov Chain. Detailed balance holds for each individual component step, if not the whole transition of θ to θ' considered as a single iteration. Implementing deterministic sweeps on different component-wise reversible move types does not invalidate the theory of the stationary or limiting distribution of the chain, as long as all move types and states are guaranteed to be visited infinitely often (Roberts, 1996).

In random updating, each different component-wise step is considered as a move type. One iteration of the chain consists of selecting a move type according to a discrete probability distribution, and then

performing the steps necessary for that move type. This version assures that all iterations (where an iteration is considered as a single-component move that is selected among move types) are completely reversible, as long as the move selection probabilities are incorporated into the detailed balance equations. If selection probabilities are uniform over all move types, then they will cancel from the Metropolis-Hastings ratio. The only difficulty may be encountering states for which only subsets of move types are available. But when the number of move types is small, and the computational cost of proposing a move type and then immediately rejecting it in states where all moves are not available, uniform selection can still be used to good effect.

Sub-chains of deterministic sub-components can also be updated within move types. As before with the simple Gibbs sampler, moving deterministically between sub-component updating steps or iterations of them does not change the limiting or stationary distribution away from $p(\theta \mid y)$. However, the choice of how to select between deterministic sweeps vs. random move types can affect empirical observable mixing in the chain, as well as the theoretical convergence rates. Amit and Grenander (1991) discuss the differences in convergence between deterministic vs. random sweep strategies for Normal models, and Levine (2005) shows that deterministic sweeps are a special case of random scans.

2.4 Updating strategy for the graph

MCMC methods can take advantage of the decomposition of the posterior into generation and allocation parts conditional on the edge links w_{ijt}. There are several types of updating steps that can be implemented either as Gibbs or Metropolis-Hastings steps within the chain:

1. Update ξ_i for a single machine;
2. Update the machine and network population parameters κ;
3. Re-allocate machine-to-address counts w conditional on observed counts x and allocation parameters for an observed network touchpoint at a single hour;
4. Update H by merging or splitting machines;
5. Update H by adding or deleting an empty (off) machine associated to one network;
6. Associate or disassociate machines among networks.

7. Update the network allocation parameters ϕ for a single network or collection of network touchpoints;

Directly observable populations in one-to-one network allocations require steps 1 and 2. In addition to 1 and 2, non-informative network allocation requires steps 3, 4 and 5 to disambiguate machines from the sum total of all observed network activity. Finally, for informative networks, steps 6 and 7 are also required in order to fully explore the posterior distribution. Steps 1, 2, 3, and 6 can be implemented using traditional Metropolis-Hastings or Gibbs steps. Steps 4 and 5 require reversible jump steps in order to match the new dimensions. Depending on how parameters are associated with network touchpoints, step 7 can also require a reversible jump step.

For directly observable populations, steps 1 and 2 can be implemented as a fixed sweep; update each machine in turn with a pre-set number of iterations, and then update the population parameters. For both non-informative and informative network allocation, it is less evident how to perform a sweep. The number of machines is subject to change, as are (possibly) the groupings, association and parameters among network touchpoints. A uniform, random selection among the possible move types is thus proposed for each iteration. Once the move type is selected, deterministic sweeps or multiple sub-move iterations can be chosen in order to aid mixing. Using this strategy, it is necessary only to check that within-move detailed balance is maintained, and that the chain maintains its irreducibility.

Conditional on choosing step 1, a machine i is selected at random from the current population, and the machine-specific parameters ξ_i are updated in a series of one or more sweeps of the dimensions needed. For step 2, all hyperparameters are updated in one or more sweeps of the dimension needed. For step 3, a network or association of networks is chosen at random, and for one or more hours in that network, counts are rearranged among the machines. Since associations/disassociations are performed only in step 6, step 3 can be augmented to choose only networks with more than one machine associated to them, without having to change the detailed balance of the sub-step.

For step 4, a coin flip is used to choose a merge or a split. Then a machine is chosen at random, and if the merge step is chosen, another machine that is connected via network association is chosen to pair with the first chosen machine, and the merge/split step is proposed using a reversible jump step. For step 5, a coin flip is used to determine whether to add or delete a machine. If addition is chosen, a network is sampled at random and an empty machine associated with all touchpoints within that network is proposed. If deletion

is chosen, an empty machine is sampled at random and is proposed to be deleted.

Step 6 and 7, the updating of associations among machines to networks, can be implemented on a network-by-network basis with reversible jumps. Two unassociated networks can be chosen for merging together or a single network can be chosen to split apart, with reversible jumps needed in order to accommodate any shared network parameters such as DHCP lease rates. All machines that reside behind the proposed collection of IP addresses now carry associations with all of those addresses, whether or not the machine currently directs traffic to each individual IP address. Because IP addresses are often assigned to institutions in contiguous blocks, the likelihood of network allocation can also depend on IP address distance between the two blocks.

Allowing a specific step for association/disassociation makes it easier to maintain detailed balance in the count rearranging steps. To facilitate the reversible jumps in the network merging step, any network selected for splitting should have at least two strongly connected components in its allocation of counts among machines and IP addresses (that is, its graph of machine-to-IP address assignments for traffic must be decomposable into at least two unconnected sub-graphs).

Chapter 3

Case Study: The Conficker Botnet

3.1 History

Conficker is a self-propagating software worm that was initially released in the autumn of 2008. Also known as Downup, Downadup, and Kido, the worm targeted a specific vulnerability in the Remote Procedure Call (RPC) software package, that allowed it to force the victim host to connect to external machines, download malicious content, and execute arbitrary code. Conficker was one of a new vanguard of botnets that moved from Internet Relay Chat (IRC) protocols to the Domain Name Service (DNS) protocol as their main method of communication. Though other, earlier botnets such as Storm (Stover et al., 2007) were using similar technology, Conficker implemented it on a larger, more "elegant" scale, as Rendon (2011) explains in the Lessons Learned document published from the Conficker Working Group. Due to its domain-based control structure, encrypted communications channels, and robust communications including peer-to-peer capability, Conficker emerged as one of the largest and most resilient botnets of its time.

Five different variations of Conficker were released between November, 2008 and April, 2009. Each variant adapted the botnet's ability to evade detection and maintain communication in the face of defensive countermeasures:

- Conficker-A: Released on November 21, 2008, this was the initial version of the code. It exploited the MS08-067 vulnerability in the RPC service, and focused on spreading infections through computers connected via local intranets. Corporations that did not install patches were especially hard hit, giving

rise to many infected computers residing behind network gateways and proxies. Conficker-A used many different methods for propagation and for obfuscation, including limiting its use of network resources so that users would not notice a slow-down in computer performance. Conficker-A used a static set of 250 domain names as the control channel for receiving instructions.

- Conficker-B: Released on December 29th, 2008, this variant used much of the same code as Conficker-A, but updated its list of control channel domain names to exclude those that had been registered and taken over by researchers (see more in Section 3.2), and to include 100 new domains. It also updated the malware's encryption, making it more difficult to reverse engineer. The quick updating in light of countermeasures indicated that the writers of the botnet were closely following developments in the cybersecurity community.

- Conficker-B++: (Designated C by Microsoft) Released February 16, 2009, this variant shared almost all of its code with Conficker-B, but included some new protocols and the beginnings of a peer-to-peer method for communication among infected machines, obviating the need for machines to check in to the static domains in order to receive instructions.

- Conficker-C: (Designated D by Microsoft) Released in late February, 2009, this was a major redesign of the code that increased the number of domains used in control channels from a static 350 to a dynamic list of $50,000$ that changed from day to day. This variant also introduced peer-to-peer scanning and other security countermeasures to prevent computers from being cleaned of an infection and rebooting.

- Conficker-E: Released April 7, 2009, this variant was an update to computers infected with Conficker-C. It downloaded and installed a malicious code package called Waledac, that presents users with a fake anti-virus software scam. This variant was programmed to delete itself and revert back to Conficker-C on May 3, and may have been a result of the Conficker botnet owners renting the botnet out to a third party criminal organization.

There has been much speculation as to the botnet's origins and to its overall purpose. An April 1, 2009 date that was discovered hard-coded into the Conficker-C software led to some speculation that Conficker would initiate some sort of activity on that date, such as a high profile DDoS. But despite its threat capability

the botnet has remained passive and generally silent since its inception. Though the Conficker-C variant has ended activity as of February 2010, the A and B variants continue to show activity with roughly 3.5 million unique IP addresses still attempting to check in daily to their control channels as of Fall 2011.

3.2 Behavior

Two ways that infected hosts maintained contact in the Conficker botnet were domain check-ins and Peer to Peer (P2P) communication. Both methods were potentially visible to an external observer, although monitoring domain check-ins required more active and expensive collection techniques than monitoring P2P communication passively from a large network.

3.2.1 Domain check-ins

Infected machines were programmed to "check in" with a control channel at regular intervals. For Conficker-A and Conficker-B, machines checked in every 3 hours, while for Conficker-C the interval increased to every 24 hours. An observer wishing to curtail Conficker infections could register all known domains used by the botnet as Command and Control (C&C) channels, and link each domain to a monitored IP address, or "sinkhole", thus throttling the botnet's ability to communicate and to retrieve instructions. Rather than specifying a single IP address or pool of addresses that could be easily blacklisted by network administrators, Conficker-C used domain names as the basis for control channels. Domain names (for example "www.google.com") can be registered and tied to an IP address on short notice, and they can be cheaply discarded after one use. The Conficker-C software upgraded from using a static list of 250 domain names as the control channels to using a deterministic algorithm to generate $50,000$ unique domain names per day. To check in, each infected host would randomly select 500 domain names from the $50,000$ possibilities, and contact them by initiating a web connection. To propagate instructions, the botnet controllers could register only a few of these domains and leave instructions; the P2P capability ensured that, even if only a fraction of machines successfully updated from a C&C check-in, they could propagate these instructions to each other across peer lists.

Researchers reverse-engineered the Domain Generation Algorithm (DGA) for Conficker-C's domains, and were able to pre-emptively publish the $50,000$ control channel domain names daily. As opposed to the

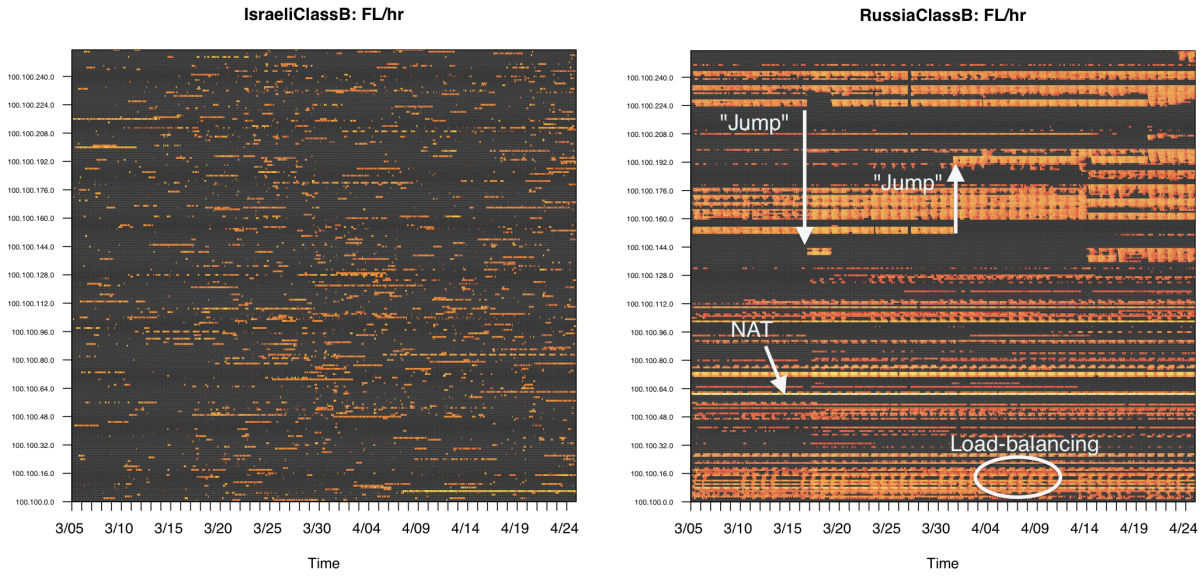

Figure 3.1: UDP Connection attempts per hour originating from a /16 net block allocated to an Israeli ISP (left) and a Russian Telecom company (right). Each row represents one of the 256 /24 net blocks that comprise the /16 network. Time ranges from March 5 through April 24, 2009. Color intensity (black to bright yellow) is a log-scale measure of the number of connection attempts originating from each $/24$ in the picture, directed at a large monitored network.

early variants where it was easy to register and sinkhole 250 domains from a static list, cutting off the central controller for the new strain would involve registering and sinkholing a new set of $50,000$ domains daily, which was not feasible. However, the monitored sinkholes were used for data collection. Though only a fraction of active machines would contact any one particular monitored sinkhole, inference on population size followed from the hypergeometric model used in the selection of 500 possibilities from the set of $50,000$. Several researchers on various security blogs used this method (some incorrectly substituting a binomial model for the hypergeometric) to estimate the size of the botnet. However, while web requests gave a bit more information than just an IP address (for example the browser used, and a "q-count" detailing possibly the machine's successful attempts at infecting its peers), it was still not possible to link individual check-ins to individual machines.

3.2.2 Peer-to-peer communication

Each infected machine kept a list of up to 2048 peers that it occasionally connected with, looking for software updates or propagation of instructions from a control channel. But an infected Conficker-C host also had a method for bootstrapping new addresses into its peer list, in case its own list was lost, cleaned, or corrupted.

Whenever an infected host was turned on, with one or more open Internet connections, it surreptitiously used those connections to continuously scan the Internet. The host determined connection ports using an algorithm based on the source IP address and date, which was cracked by several independent researchers (Porras et al., 2009a, eg.). As a result, Conficker-C P2P requests could be identified with high reliability in large-scale network summary information, with no need to inspect the content of messages.

For the outside observer, Conficker-C hosts were visible (up to IP addresses) when they chose to connect to an IP address that the observer monitored. Because the population of Conficker-C hosts was quite large after its inception, and each host generated many requests per hour, an observer monitoring even a relatively small network had a good chance of seeing some randomly generated P2P requests.

An example of observed passive scan data by network is displayed in Figure 3.1. Color intensity (black to bright yellow) in both figures represents the number of connection requests per hour sent from Conficker-C infected machines in a $/16$ net block, to uninfected machines located on a large monitored network. Each row in the figure corresponds to a $/24$ net block, with brighter yellow colors indicating higher numbers of connection requests (on the log scale) initiated from that net block.

The leftmost figure shows a network allocated to an Israeli ISP. The $/24$ blocks in this network show patterns that might be expected with single-machine activity per net block. Intensity is uniformly low-scale, with daily trends evident in some blocks, and little correlation of activity between rows. Comparing these patterns to the figure on the right, which is allocated to a Russian Telecom company, the effects of administration policies such as NAT, load-balancing, and DHCP become more evident. Intensity varies among blocks, with the highest belonging to a single net block ($94.25.61.0/24$) that appears to be acting as a proxy for a comparatively large number of infected hosts. Daily patterns are also evident as users come on- and off-line, but the effect seems to be correlated across multiple adjacent blocks, evidence of DHCP leasing or load-balancing. There are also several places where large blocks of activity appear to "jump"–

scans cease abruptly in one location and appear abruptly in another–possibly due to load-balancing or other administrative decisions.

Figure 3.2 provides a clear illustration of the difficulties that arise when trying to measure the size of the botnet by a count of IP addresses. The connection attempts (flows) per hour from Conficker-C infected machines are displayed for IP addresses in a $/17$ network assigned to a Ukrainian Telecom company. Again, each row represents a $/24$ net block (256 IP addresses), with intensity represented by the heat colors on the log scale. Aggregates of total IP addresses per hour, total $/24$ net blocks per hour, and total connection attempts per hour are shown in the underlying plots. Over the 51-day period, a total of 26830 of the possible 32768 IP addresses in the $/17$ were observed sending Conficker-C infected traffic, with an average of 104 IP addresses active per hour.

The network allocation plot shows what appears to be a random allocation of activity across the entire range of available addresses from March 3rd through to April 1st. Starting abruptly on April 1st, a drastic change in allocation occurs as the 8192 IP addresses in the upper $/18$ go almost completely dark for 11 days, and a descending daily pattern is implemented across the lower $/18$ net block that appears to activate half as many $/24$ net blocks per hour at peak activity times, in a matched sequential pattern. Despite the concentration of activity to only a few $/24$ net blocks during that period, the IP address plot shows that the number of active IP addresses per hour actually increases over the interval. The random pattern is re-implemented for another 11 days starting on April 12, and then on April 23, all activity is moved to the 1024 IP addresses in the lowest $/21$ on the network for 24 hours.

Counting the number of active IP addresses alone, the botnet appears to grow drastically on April 1. On the other hand, counting the number of active $/24$s, the botnet appears to *shrink* on this interval. But the rate of connection attempts per hour, from active infected machines communicating through these addresses, tells a different story. Behaviorally, despite all of this re-allocation and re-arranging, the total number of connection attempts per hour remains fairly constant. The bottom time series plot shows what appears to be only a gradual decline in active rates across the entire 51-day window. Using a "ballpark" estimate of 4 connection attempts per hour per active machine, and assuming that infections are long-lived, it is possible that the infection activity across over 26000 IP addresses on this $/17$ network could be a wide footprint for as few as 200 infected machines being slowly taken offline and cleaned.

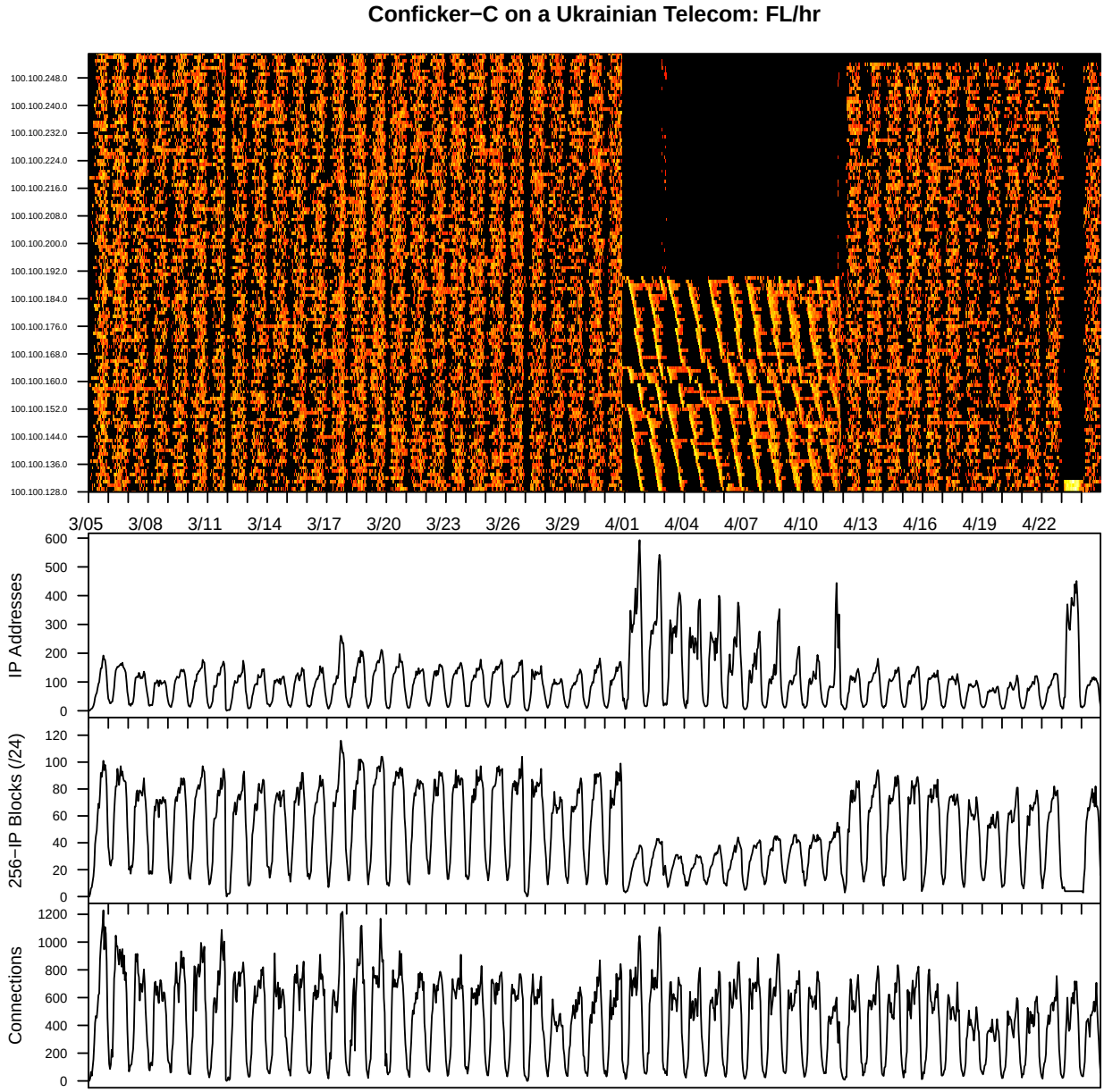

Figure 3.2: UDP Connection attempts per hour originating from a $/17$ net block allocated to a Ukrainian Telecom company. Each row represents one of the 128 $/24$ net blocks that comprise the network. Time ranges from March 5 through April 24, 2009. Color intensity (black to bright yellow) is a log-scale measure of the number of connection attempts originating from each $/24$ in the picture, directed at a large monitored network. The bottom three time series plots are IP addresses active per hour, $/24$ blocks per hour, and Connection attempts per hour.

3.3 Observing a single infected machine

Suppose an observer monitors a proportion δ of destination IP addresses over a period where Conficker-C hosts are active. In one hour t, a single infected machine actively scanning at rate λ_t initiates M_t random UDP connection requests from a source IP address, of which y_t are directed at destinations within the monitored network. The quantity M_t is modeled as a Poisson process:

$$M_t \sim \text{Poisson}(\lambda_t)$$

This a reasonable model for small-packet scanning activity programmed at regular intervals. Network traffic is often cited as "bursty" in behavior, but Paxson and Floyd (1995) note that this self-similarity is more common in packet inter-arrival times once connections have been established, as opposed to multiple connection requests. Published experiments with the Conficker-C malware in controlled settings (Porras et al., 2009b) show relatively smooth scanning rates, within both 30-minute and 6-hour time frames.

The Poisson count M_t is assumed to be conditionally independent from M_{t-1} given its rate λ_t. The conditional distribution $\pi(y_t \mid M_t, \delta)$ is Binomial(M_t, δ), which yields a Poisson marginal model for y_t in terms of λ_t and δ:

$$y_t \sim \text{Poisson}(\lambda_t \delta). \tag{3.1}$$

The proportion δ is measured empirically, and the scan rates are unknown; without loss of generality, the rate λ_t can be modeled as an observed hit rate (subsuming the parameter δ).

In September 2009, Porras et al. (2009c) provided a de-obfuscated reverse engineering of the Conficker-C P2P binary image as it appeared on March 5, 2009. When initiated, the P2P module spawns a UDP scanning process for each valid network connection discovered, in order to bootstrap a peer list of up to 2048 peers. Up to 32 processes can run simultaneously. Each process alternates between a 5-second sleep cycle and a scan phase where it randomly generates a list of up to 100 IP addresses to contact. At each selection, the machine chooses an IP address from its list of n peers with probability equal to

$$Pr(\text{existing peer chosen} \mid n) = \left(1000 - \left\lfloor \frac{950n}{2048} \right\rfloor\right)^{-1}. \tag{3.2}$$

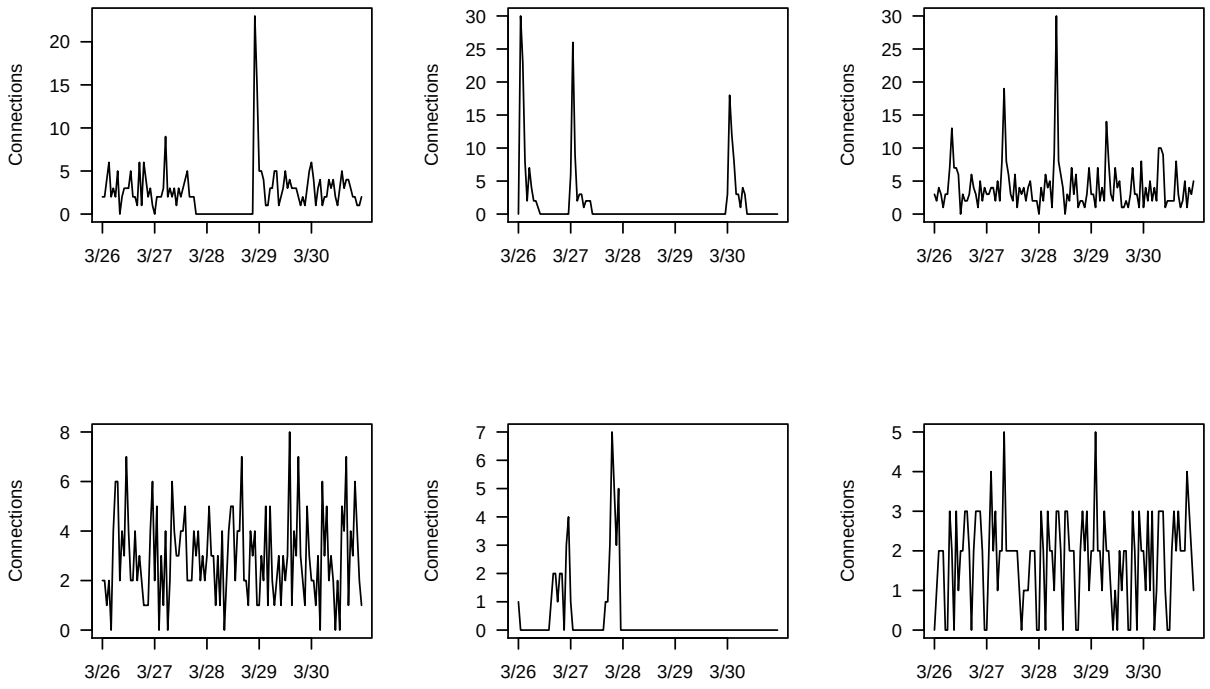

Figure 3.3: Connection attempts per hour over a 5-day period, from six different IP addresses.

The speed at which UDP connection requests are sent out over a connection depends on the hardware and network capabilities of the infected machine, as well as the amount of bandwidth and drop percentage of the network. The P2P protocol has a maximum of 1200 scanning connection attempts per minute, but observed accounts of Conficker-C P2P scan activity cite lower numbers. Porras et al. (2009b) performed a sandbox test of an infected Conficker-C host with a single network interface and observed scanning rates starting at approximately 1000 to 2000 IP addresses per 5 minute interval, and decreasing over the first two hours of activity to a steady rate of approximately 200 IP addresses per 5 minutes. Figure 3.3 shows the number of connection requests per hour sent from six different Conficker-C infected IP addresses, to uninfected machines located in a large monitored network over a five-day period. Several of the series display spikes of activity. It is speculated that each of these time series plots represents a single machine; empirically, low overall scan rates in these net blocks suggest they are not NAT gateways. Each address also resides in an "isolated" section of otherwise dormant (non-scanning) IP addresses that do not appear to be DHCP pools.

Based on the information from the reverse engineering and sandbox tests, two reasons for spikes can be postulated:

- Clearing of peer lists: The proportion of random vs. peer-directed connection attempts depends on the size of the machine's peer list. When a peer list is lost or cleaned (for example after a machine is rebooted), the rate of random scanning will increase due to the effects of Equation 3.2 and gradually decrease as the machine rebuilds its peer list.

- User-driven network connections: Reboots and other user behavior can initiate multiple network connections (for example, requesting IP addresses from servers, downloading Windows updates, or opening web browsing sessions), that cause a multiplicative increase in scanning rates due to the malicious Conficker-C software initiating multiple concurrent scanning threads.

3.4 Data collection

This work focuses on the modeling of the Conficker-C UDP scan rate, so the main data set used in analysis is hourly counts of P2P communication requests sent into a large private network. Sinkhole data was also available from earlier variants of the worm, Conficker-A and Conficker-B, prior to the switch to the $50,000$ new daily infections. Because Conficker-C was installed on platforms already infected with the earlier variants, these data sets had many overlapping IP addresses. For this analysis, the sinkhole data was used only to vet the passive scan observations as true positives given the network signature.

3.4.1 Passive scan data

The monitored network used in this study is a large private network of approximately $21,000$ $/24$ net blocks, or 0.15% of IPv4 address space. The network is instrumented with a set of sensors running the System for Internet-Level Knowledge (SiLK) [1] tool set. SiLK is an open-source capability for collecting and analyzing network flow data, which provides a high-level summary of network communications without inspecting or storing the content of messages. A flow record is a summary of the time, ports, protocols, packet header information and bytes transferred between two IP addresses, one designated as the source and

[1] `http://tools.netsa.cert.org/silk`

one as the destination of a communication. Because the network signature for Conficker-C P2P scanning was a deterministic function of the IP address, the source port and the timestamp, it was possible to detect UDP connection requests in SiLK flows. From the period of March 5th through April 24th, 2009, inbound Conficker-C UDP P2P connection attempts were collected using a SiLK plug-in tool (Faber, 2009) based on the reverse engineering performed by Porras et al. The plug-in related each connection attempt to a single flow record, and so a count of connection attempts could be proxied by a count of flow records that were flagged by the Conficker-C signature.

The large network telescope, coupled with typical UDP scan rates on the order of 1000 per hour or more, resulted in a good chance that an active host would be observed with at least one flow per hour into the monitored network. A total of 33.6 million unique IP addresses were observed performing Conficker-C P2P scans over the 2-month window. Hourly flow counts were collected, and aggregated by $/24$ net block to reduce the sparsity and size of the data set, as well as to account for ephemeral DHCP leases allocated over a small number of addresses. A total of 1.1 million unique $/24$ net blocks were observed performing Conficker-C P2P scans. A manual examination of the destination IP addresses by source IP address revealed a few anomalies in terms of repeated attempts from a single source IP to the same destination IP, but the vast majority of the connection attempts from single IP addresses each went to unique destination IP addresses within the hour.

The raw data set is represented as two matrices, $\mathbf{Y}$ and $\mathbf{A}$, each of 1.1 million rows (one for each unique $/24$ that showed activity) by 1224 columns (one per hour), where y_{it} is the number of requests seen from net block i at time t, and a_{it} is the number of distinct IP addresses (from 0 to 256) seen communicating from net block i at time t. Figure 3.4 summarizes a random sample of 1000 net blocks from the larger set and displays the rate of flows per active IP address per hour for each net block across the observation period. Color intensity (black to bright yellow) is a log-scale measure of the number of connection attempts originating from each $/24$ in the picture, directed at destinations within the monitored network. Daily activity patterns can be seen in some traces, as well as the effect of NAT in several bright yellow traces across the spectrum. Figure 3.5 shows the count of active IP addresses observed per hour for the duration of the data collection period. The Conficker-C update appeared to take a tiered approach, with a small number of infected machines updating on March 3rd, and then propagating the new instructions to the rest of the

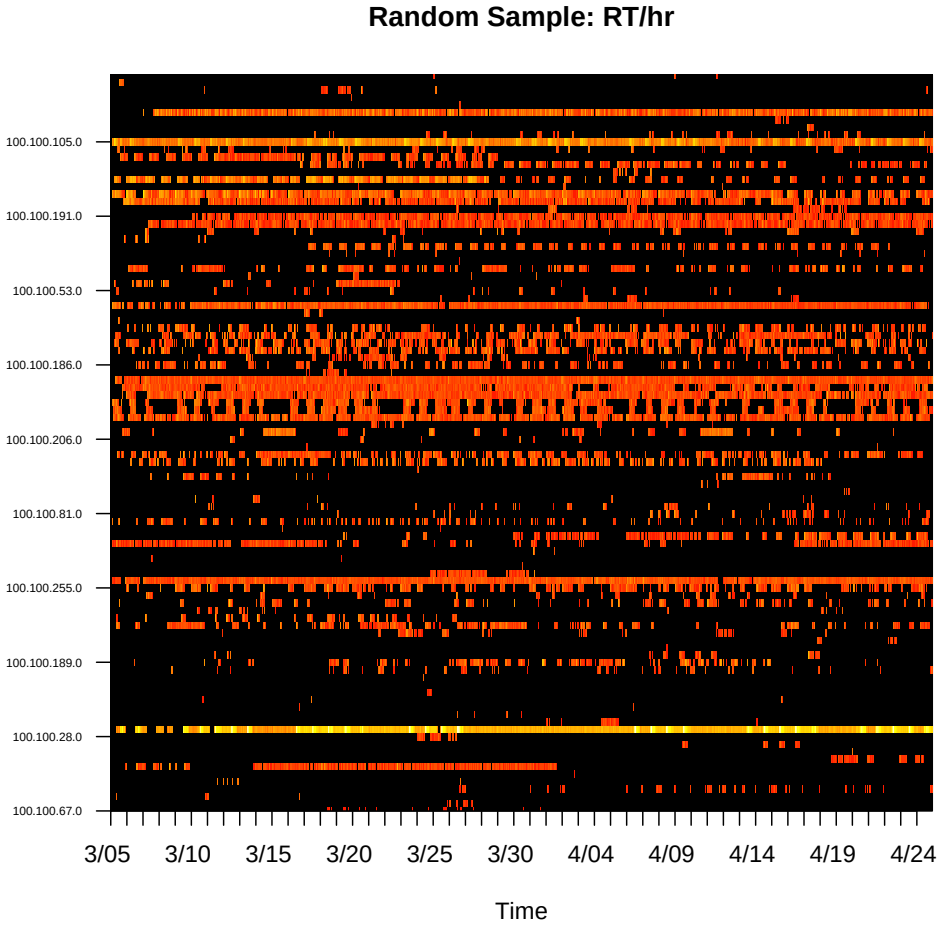

Figure 3.4: Observed scan rate (Flows/IP/Hour) for a random sample of 100 $/24$ net blocks from the population. Color intensity (black to bright yellow) is a log-scale measure of the number of connection attempts originating from each $/24$ in the picture, directed at a large monitored network. A suspected NAT device is evident as a bright yellow line on the plot.

botnet on or near the 17th of March. After the March 17th spike, the counts appear to decline to a level baseline toward the end of April. This decline is most likely due to clean-ups; the P2P scanning allowed for a robust communication channel, but it also created network "chatter" that was more easily noticeable to administrators. Figure 3.6 shows plots of the average number of connection attempts per IP address per active hour (left), and of the live range (right), for each $/24$ network. Connection attempts are measured with observed flows; for each hour with a non-zero count of activity, the number of flows seen for that hour is divided by the number of active IP addresses seen for that hour. The average value for each net block

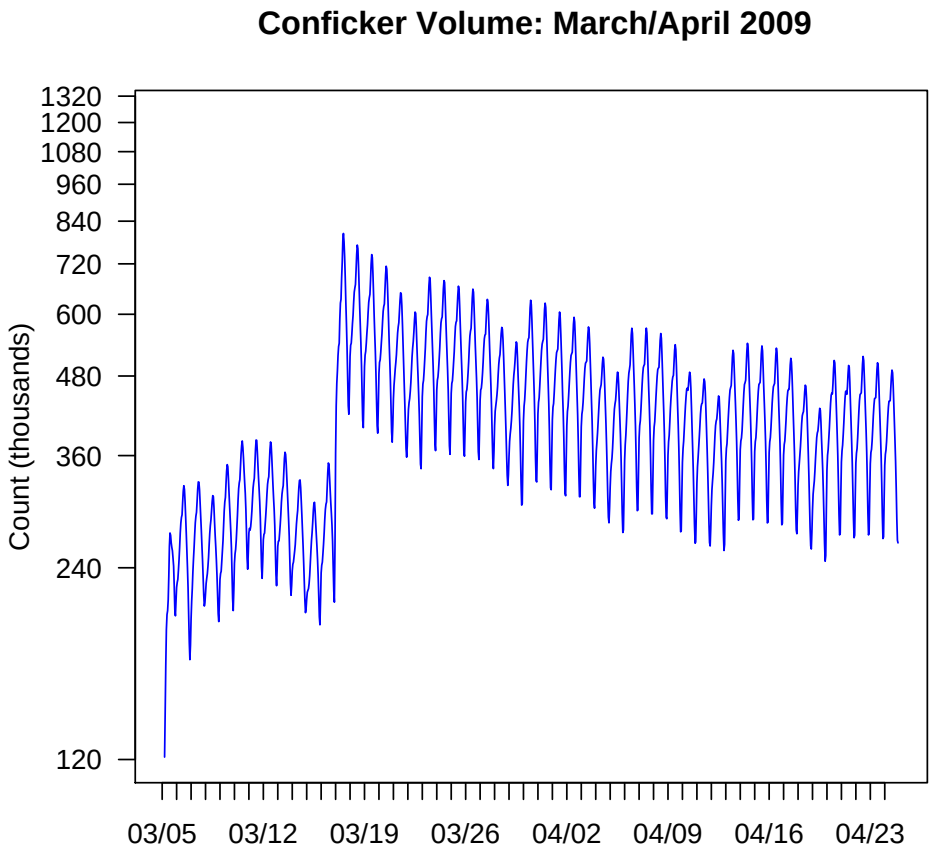

Figure 3.5: Active IP addresses per hour across the observation window.

is calculated across all of the net block's active hours. The plot shows a log-log scale histogram of these averages, rounded to the nearest integer. Most net blocks have average values of less than 10 (mean $= 4.67$, median $= 4.25$). Based roughly on the sandbox analysis by Porras et. al, a computer that spikes in activity every 24 hours, and decays back down to the rate of approximately 200 connection attempts per 5 minute interval, would send out roughly $89,000$ connection requests per day. A network telescope of 0.015% of IPv4 space would see an average rate of approximately 5.5 connections per active hour (though this also includes hours where the machine is active but yields a count of 0), which appears reasonably consistent with the mode of the data. However, the long tail of this rate distribution also provides evidence for the presence of NAT devices and proxies, with some rates of over 500 to 1000 per hour. A machine that does

not have typical spike activity would tend to have an average rate of approximately 3.6 connections per active hour. Smaller average rates (1 to 2 per hour) may also be indicative of non-spiking activity or of more ephemeral use of IP addresses: the address could be used in smaller increments than a single hour, or a small number of observed active hours could also yield a higher variability in the low end of the spectrum of observed counts.

The live range (right plot) is calculated as the difference of the last active hour and the first active hour as seen in the observation window. The y-axis is a log-scale plot of the count of net blocks that have live range of the value x on the horizontal axis. The saw-toothed pattern is a periodicity of 24 hours. The red points highlight live ranges of 24 hours, 168 hours (one week), and 920 hours (approximately 38 days, the length of the observation window starting from the March 17th spike in activity). The saw tooth pattern persists for net blocks that have observed start and end times cushioned at least one week from the start and end of the observation window, so this pattern does not appear to be an effect of the constrained collection window. The regularity of the 24 hour pattern suggests that activity within net blocks tends to initiate and cease at the same hour of the day. This suggests, possibly, that some net blocks (or IP addresses within blocks) are assigned or re-arranged according to very structured schedules, evidence of widespread DHCP pools and leases on the scale of days. A total of 38295 (3.5%) net blocks were observed active for only one single hour in the data set. Figure 3.7 shows a plot of these ephemeral net blocks over time. An anomalous spike occurs just prior to the general March 17th surge. On average, each hour of the data set saw 31 net blocks both initiate and end all activity, with 96% of hours admitting fewer than 60 of these highly ephemeral blocks. On March 15th, 12:00:00 UTC time, a total of 284 net blocks both initiated and ended all activity within the same hour, mostly across the countries of the United States, Japan, France, Great Britain and Canada. This activity took place approximately 32 hours prior to the surge in scan rates. According to the Conficker Working Group (2009), March 15th, 2009 coincided with the activation of the malicious domain "rmpezrx.org", and allowed for "many hosts" to update to the C variant via DNS [2]. But it is unclear why this large number of seemingly unrelated network blocks appeared and disappeared so rapidly from observation via passive UDP scans, unless they were somehow blocked or taken down almost immediately by their owners.

[2] `http://www.confickerworkinggroup.org/wiki/pmwiki.php/ANY/Timeline`

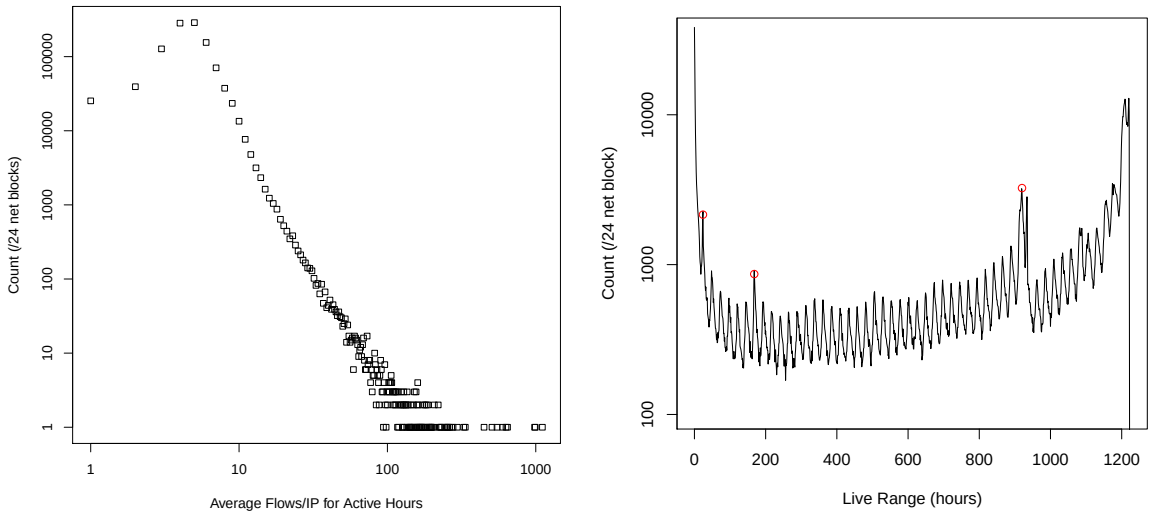

Figure 3.6: Log-log plot of average connection attempts per IP address per active hour for each $/24$ net block (left), and log histogram of the live range of last active hour observed minus first active hour observed (right) for the data set. The long tail in the left plot is evidence of NATs, while the 24-hour periodic component of the right plot is evidence of structured DHCP leases.

3.4.2 Geolocation and alignment of behavioral rate spikes

Geolocation information for each net block was obtained using a database of country codes associated with IP addresses (MaxMind, 2010). Each net block was also assigned roughly to a time zone based on its country code, with 1% of blocks remaining unassigned due to satellite locations or unavailable country codes. Table 3.1 shows a breakdown of the top 20 countries appearing in the data set, by both IP address, and by $/24$ net block. The countries that appear in only one of the two lists are marked by a (*). Large telecom companies in Eastern Europe and Southeastern Asia appear to contribute many more IP addresses than Europe and North America, but due to allocation policies and possibly to widespread DHCP use, the activity is spread across fewer $/24$s. On the other hand, Japan, the U.S. and Western Europe appear to have sparser allocations of infected IP addresses by net block. Weaver (2010a) examines these results in detail. Widespread Conficker-C activity occurs on a $/15$ net block belonging to Ukraine telecom, and on telecom companies located in Indonesia and Viet Nam.

To capture the tendency of machines to spike in their scan rates at regular 24-hour intervals, a periodic alignment value t^* was also estimated for each $/24$ net block in the data set, using flow quantity percentiles

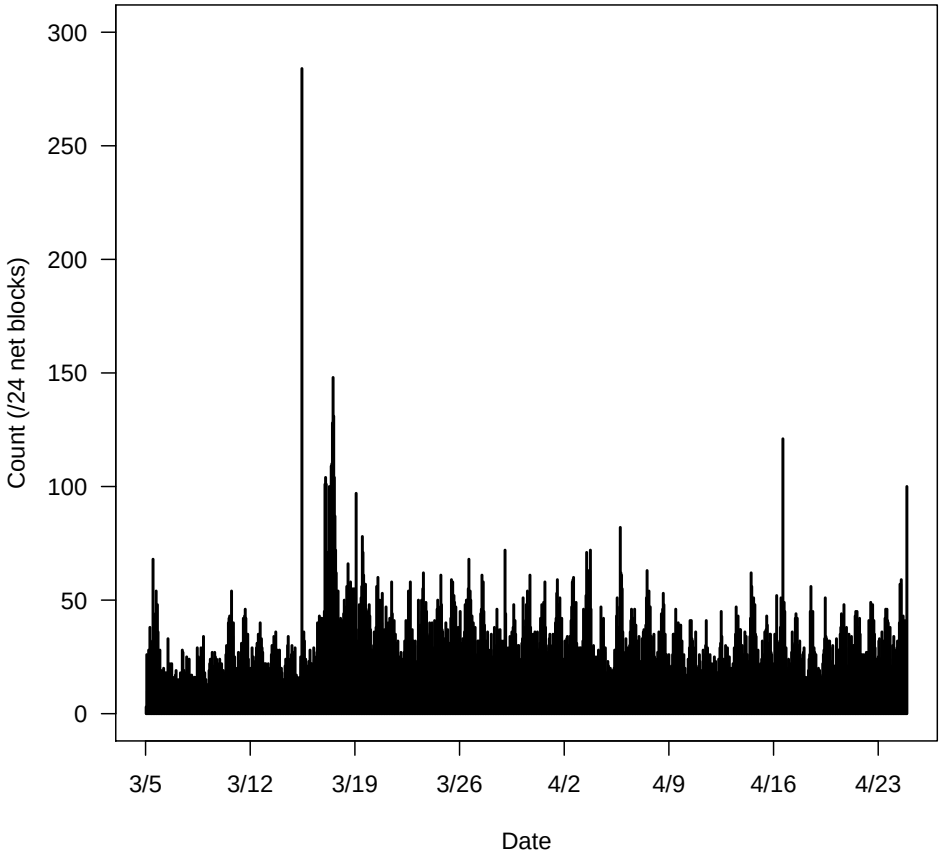

Figure 3.7: Count of "ephemeral" $/24$ net blocks by date. An ephemeral block is a block that both initiates and ceases all scanning activity (from the viewpoint of the 2-month observation window) within the same hour.

by hour of day. The value of t^* is an offset that normalizes the hour of the day such that the periodic spikes, if they exist, are aligned with hour 5 out of the hours $0 - 23$ of the day. Figure 3.8 shows an example of the calculation for four different net blocks. Alignment is calculated using the median, 75th, 85th, or 90th percentiles of the flow counts for each hour of the day, or the maximum value observed if all other quantiles are equal to 0 across the observation window. The maximum value for each hour is plotted in red to indicate that it is a more highly variable estimate of trend. The spike hour is taken as the hour with the maximum value of the lowest level quantile for which there is at least one hour with a non-zero quantile value. For example, if the median value of the count across all hours between the first and last observed active time

	By IP Address				By /24 net block			
Rank	Country	#IPs	%IPs	Cumul.%	Country	#/24s	%/24s	Cumul.%
1	China	10317667	26.81	26.81	China	202425	17.82	17.82
2	Brazil	4012459	10.43	37.23	*United States	102614	9.04	26.86
3	Russia	2920027	7.59	44.82	Germany	89046	7.84	34.70
4	India	1742319	4.53	49.35	South Korea	65623	5.78	40.48
5	Italy	1729420	4.49	53.84	*Japan	55669	4.90	45.38
6	Taiwan	1185604	3.08	56.92	Brazil	47936	4.22	49.60
7	Argentina	1125793	2.93	59.85	*France	46468	4.09	53.69
8	Germany	1077345	2.80	62.65	*United Kingdom	44325	3.90	57.60
9	*Viet Nam	1037676	2.70	65.34	Italy	41256	3.63	61.23
10	*Thailand	891329	2.32	67.66	Russia	38709	3.41	64.64
11	*Ukraine	835716	2.17	69.83	Spain	38207	3.36	68.00
12	South Korea	701623	1.82	71.65	Mexico	24902	2.19	70.19
13	Spain	645119	1.68	73.33	Taiwan	22447	1.98	72.17
14	*Malaysia	638925	1.66	74.99	Turkey	18390	1.62	73.79
15	Turkey	567354	1.47	76.46	*Australia	17036	1.50	75.29
16	Romania	565670	1.47	77.93	India	16757	1.48	76.76
17	Mexico	540947	1.41	79.34	*Poland	16624	1.46	78.23
18	*Indonesia	516460	1.34	80.68	Argentina	13604	1.20	79.43
19	*Chile	488220	1.27	81.95	*Canada	11931	1.05	80.48
20	*Philippines	478989	1.24	83.19	Romania	11561	1.02	81.49

Table 3.1: Top 20 Country Codes per IP address and per $/24$ net block (address $x.x.x.0$ as proxy for the block). (*) highlights countries which appear only in one list. Large telecom companies in Eastern Europe and Southeastern Asia appear to contribute many more IP addresses but spread across fewer $/24$s. On the other hand, Japan, the U.S. and Western Europe appear to have sparser allocations of infected IP addresses.

is 0, then the spike hour is chosen from 75th percentile. If the 75th percentile is also equally 0 across all hours, then the 85th is examined, etc. This reduces the noise of, for example, a very occasional spike at an off hour that may overwhelm the overarching trend. For example in the top left, the median value was used to align the tendency for spikes; even though some of the higher quantiles indicate ephemeral spikes, the largest consistent trend is captured by the blue line. Not all machines exhibit regular spikes, for example the top two net blocks show very prominent trends, whereas the bottom two net blocks seem to have a more even distribution. A single-machine model should be flexible enough to parameterize both cases. In some cases, the periodicity appears to be over or under 24 hours. However, the alignment only takes into account the 24-hour case.

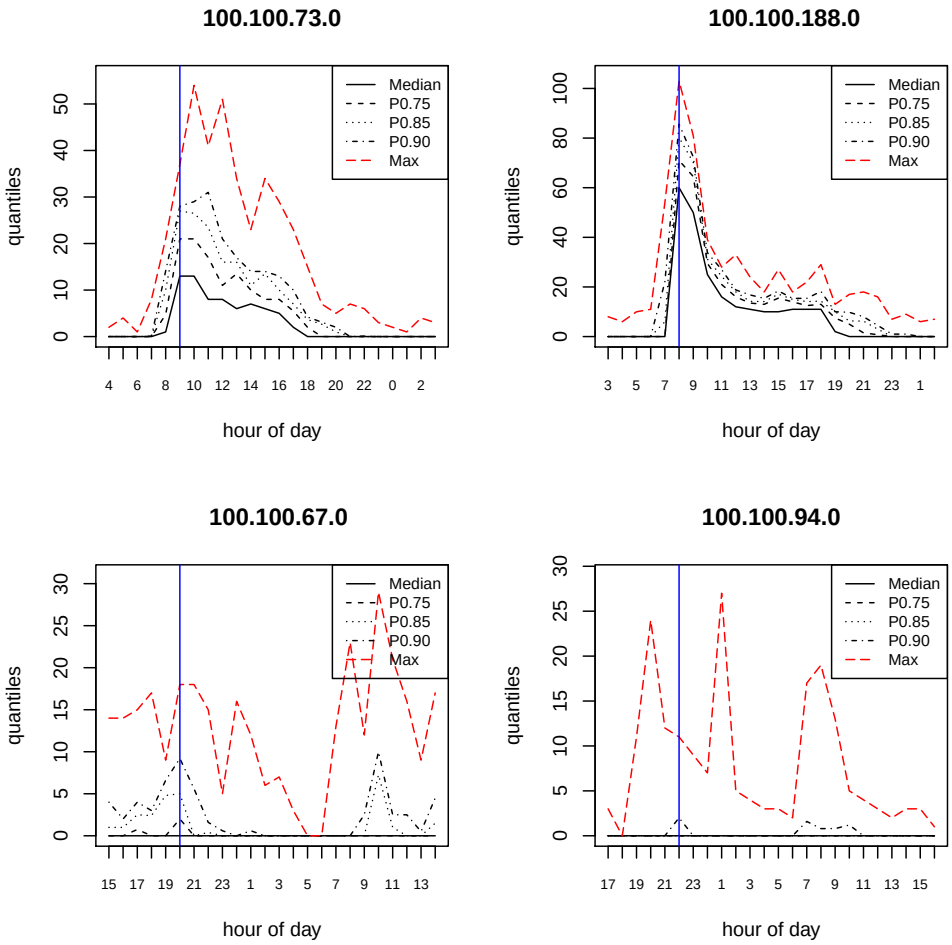

Figure 3.8: Alignment values for spike tendency in a 24-hour period. The maximum value for each hour is plotted in red to indicate that it is a more highly variable estimate of trend. Not all machines exhibit regular spikes, for example the top two net blocks show very prominent trends, whereas the bottom two net blocks seem to have a more even distribution of activity. A single-machine model should be flexible enough to parameterize both cases.

Chapter 4

Inference for a Directly Observable Population

A directly observable population is one for which machines are linked one-to-one with IP addresses, or for which communications can be observed on a machine-by-machine basis in some other way. For observational network telescopes studying botnets, this is rarely the case. However, in some cases, researchers and other network defenders have infiltrated botnet C&C channels or have studied individual infections on their own internal networks. Obviously, it is easy to obtain population estimates when machines are directly observable. More importantly, directly observable populations can be used to inform the generation model $f(y)$ and–in some cases–its interaction with daily user activity patterns that may not be evident through sandbox analysis alone.

In a directly observable population, the number of machines H is fixed equal to J, and the counts are visible such that $w_{ijt} = y_{it} = x_{jt}$ with probability 1 for all t. The allocation model is trivial, and the graph $\mathbf{W}$ is thus the collection $y_{11}, \ldots, y_{HT}$ of all counts generated from infected machines throughout the observation window. The prior $\pi(H = J) = 1$ is also trivial, and the posterior distribution in equation 2.5 has the form:

$$p(\kappa, \xi \mid \mathbf{W}) \quad \propto \quad \pi(\kappa) \prod_{i=1}^{H} \pi(\xi_i \mid \kappa) \prod_{t=1}^{T} f(y_{it} | \xi_i, y_{i1}, \ldots, y_{it-1}) \tag{4.1}$$

Inference for the generation-allocation model reduces to inference for an independent sample of the single-host generation model, with one added hierarchical level comprising the population hyperparameters. The remainder of this chapter develops this model for the Conficker-C botnet and presents results for a subset of $/24$s in the collected data that could be verified via external information as probable examples of individual machines.

4.1 Generation model for the Conficker-C UDP scan rate

This section develops the form of the generation function $f(y)$ for observed counts from a single machine infected by Conficker-C. The malware is observed in this study through its random scanning process. The observed count y at each hour t is the number of times that an infected host sends a UDP connection attempt to one of the machines that the observer monitors.

4.1.1 Hourly scan rate

For a single machine i, the generation function $f(y_{it}|\xi_i, y_{i1}, \ldots, y_{it-1})$ is modeled as a Poisson process with a time-dependent mean λ_{it}. In this section the host index i is suppressed for ease of notation. The overall observed scan rate λ_t at time t is parameterized by a baseline active rate q that represents the stable rate of connection attempts sent by a machine when it is turned on, a parameter $\omega > 1$ that multiplicatively increases the baseline rate during a spike, and a parameter α that controls the geometric decay of spike rates back to the baseline. Let $\theta = \{q, \omega, \alpha\}$. The generation function is given by:

$$\begin{aligned} f(y_t) &\sim \text{Pois}(\lambda_t(\theta, \eta_t)) \\ \eta_t &\sim \pi(\eta_{t-1}) \end{aligned}$$

where η_t is a discrete active state that denotes the tendency and frequency of the hosts to spike in rate due to the malware's programming. The priors for the baseline, spike and decay rates are informed by the sandbox analysis and by the size of the monitored network used for the study. The activity profile of a machine is a mixture of this generation model with a dormant (zero) rate associated with an inactive state. Inactive states (where the computer is turned off or disconnected) can also be modeled by the Markov progression;

but patterns of inactivity tend to be more evenly centered around the mean. Section 4.1.2 discusses the modeling decisions that arise from this interpretation.

Denote by $\mathcal{O}$, $\mathcal{S}$, and $\mathcal{D}$ the set of hours at which the machine is off, spiking, or decaying, respectively. The rate of observed connection attempts λ_t is equal to 0 for all $t \in \mathcal{O}$. Given $t \in \mathcal{S}$, define the spike rate as

$$\lambda_t = q\omega, \quad \omega > 1 \tag{4.2}$$

For an hour $t \in \mathcal{D}$, define

$$\lambda_t = q + \max[0, \alpha(\lambda_{t-1} - q)], \quad 0 < \alpha < 1 \tag{4.3}$$

The overall observed scan rate $\lambda_t^{\mathcal{D}}$ for decay states can also be expressed in terms of the most recent state change. If a decay state directly follows an off state ($t + 1 \in \mathcal{D} \mid t \in \mathcal{O}$), then the rate is equal to the baseline rate q until the machine either spikes or is turned off. For any decay state that follows k steps after a spike state with no intervening off states ($t + k \in \mathcal{D} \mid t \in \mathcal{S}, t + 1, \ldots, t + k - 1 \in \mathcal{D}$) , the rate can be expressed as

$$\lambda_t = q(1 + \alpha^k(\omega - 1)) \tag{4.4}$$

A spike state corresponds to $k = 0$, resulting in the rate $q\omega$ as described in Equation 4.2.

4.1.2 Transitions between off/spike/decay states

Depending on the configuration and use of the infected hosts, rate spikes can appear at relatively regular intervals (for example, approximately every 24 hours, on weekdays only), at sporadic intervals corresponding to rare or unscheduled user events, or as a mixture of both scheduled and unscheduled activity. Rate spikes are also more likely to follow periods of inactivity, regardless of the hour at which a machine turns on.

A discrete 3×3 transition matrix is used to characterize the state changes (spiking, decaying, or off) from hour to hour. Let $a \in \{o, s, d\}$ be an index representing the state of activity at time t. To capture these varying levels of periodicity on a 24-hour scale from host to host, the transition probabilities from each state

are defined with a periodic component that controls for amplitude with a scaling parameter $\nu_a > 0$ and a baseline parameter $0 \leq \rho_a \leq 1$, as well as an aperiodic choice parameter $0 \leq \gamma_a \leq 1$. Let t^* be the centered time index ($t^* = t - d$) such that a value of 0 aligns the periodicity with the desired hour of the day, for example, aligning troughs with midnight and peaks with noon. Define $p(\rho_a, \nu_a, t^*)$ as a 24-hour periodic probability function parameterized by ρ_a and ν_a relative to t^*:

$$p(\rho_a, \nu_a, t^*) = \frac{\rho_a}{\nu_a + 2} \left[\sin(2\pi t^*/24) + \nu_a + 1\right]$$

Figure 4.1 shows some examples of $p(\rho_a, \nu_a, t^*)$ across a 24-hour window for various values of ρ_a and ν_a. The value ρ_a represents the maximum or peak probability achieved across the 24-hour window, while the scaling parameter ν_a controls the strength of the sinusoidal pattern across the 24-hour period. The function $p(\rho_a, \nu_a, t^*)$ is a sinusoidal pattern that can be used to model transitions between two discrete states. For parameterizing transition probabilities $\pi_{\cdot|a}(t) = Pr(\eta_{t+1} \in \{o, s, d\} \mid \eta_t = a)$ between three states relative to the 24-hour period, the sinusoidal probability function is multiplied by the aperiodic choice parameter γ_a. This parameter is used to model state transitions that are most common across the observations. For example, a spike state is very rarely followed by another spike state, irrespective of the time at which the spike occurs. This constraint can be modeled into the probability structure $\pi_{\cdot|s}(t)$ as follows:

$$\begin{aligned}
\pi_{o|s}(t) &= (1 - \gamma_s)(1 - p(\rho_s, \nu_s, t^*)) \\
\pi_{s|s}(t) &= \gamma_s \\
\pi_{d|s}(t) &= (1 - \gamma_s)(p(\rho_s, \nu_s, t^*))
\end{aligned}$$

The transition probability model can be further informed by a strong prior on γ_s that penalizes large values.

Decay states tend to occur in long sequences as a machine stays turned on and idle. The model for decay states uses an aperiodic estimate of $\pi_{d|d}(t)$ and a periodic component to model the conditional probability of switching to a spike or off state given a transition away from a decay state. This is modeled in the probability

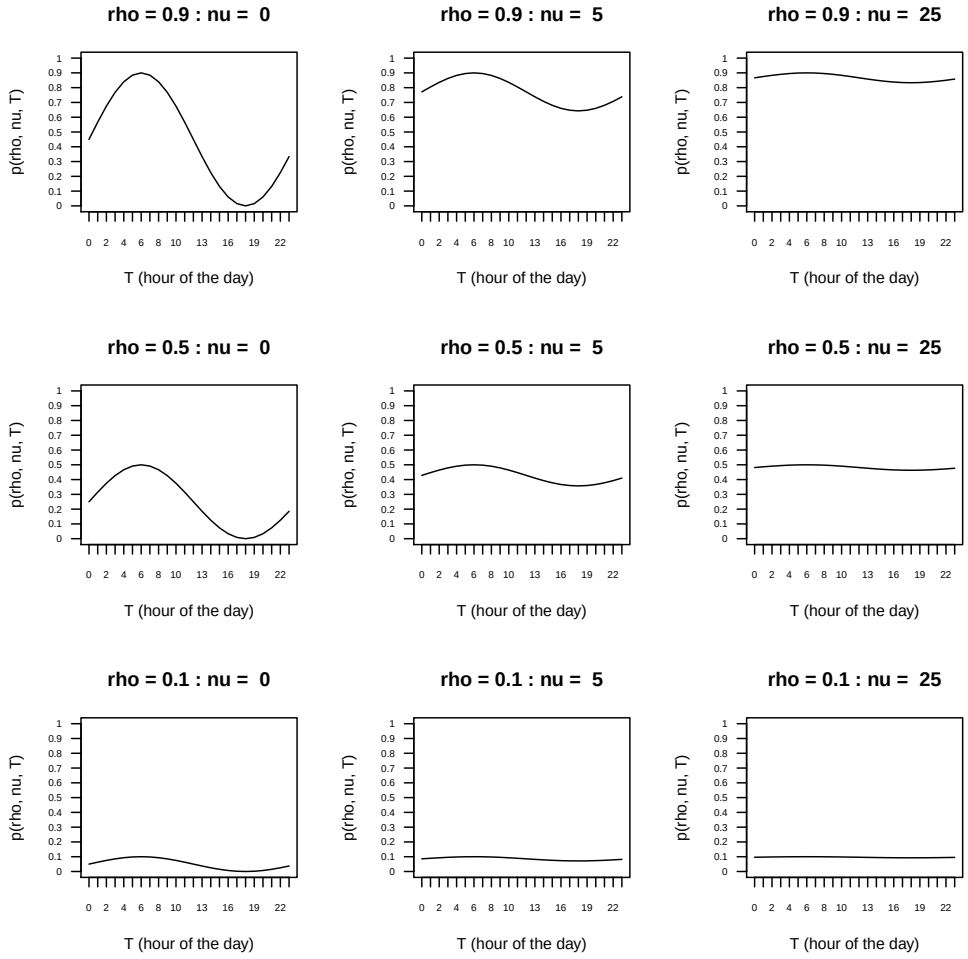

Figure 4.1: 24-hour profiles of $p(\rho_a, \nu_a, t^*)$ for different values of ρ and ν.

structure as:

$$\begin{aligned}
\pi_{o|d}(t) &= (1 - p(\rho_d, \nu_d, t^*))(1 - \gamma_d) \\
\pi_{s|d}(t) &= p(\rho_d, \nu_d, t^*)(1 - \gamma_d) \\
\pi_{d|d}(t) &= \gamma_d
\end{aligned}$$

Figure 4.2 shows an example of the transition probabilities from these parameterizations. The left plot shows transitions from spike states for a machine that has few spike-to-spike transitions and a moderate propensity to turn off in late hours after a spike. The right plot shows transitions from decay states for a machine whose

runs of decay states follow a geometric distribution with success rate of approximately $2/3$, and that follows a strong conditional periodic trend ($\rho = 0.81$) when transitioning away from a decay state.

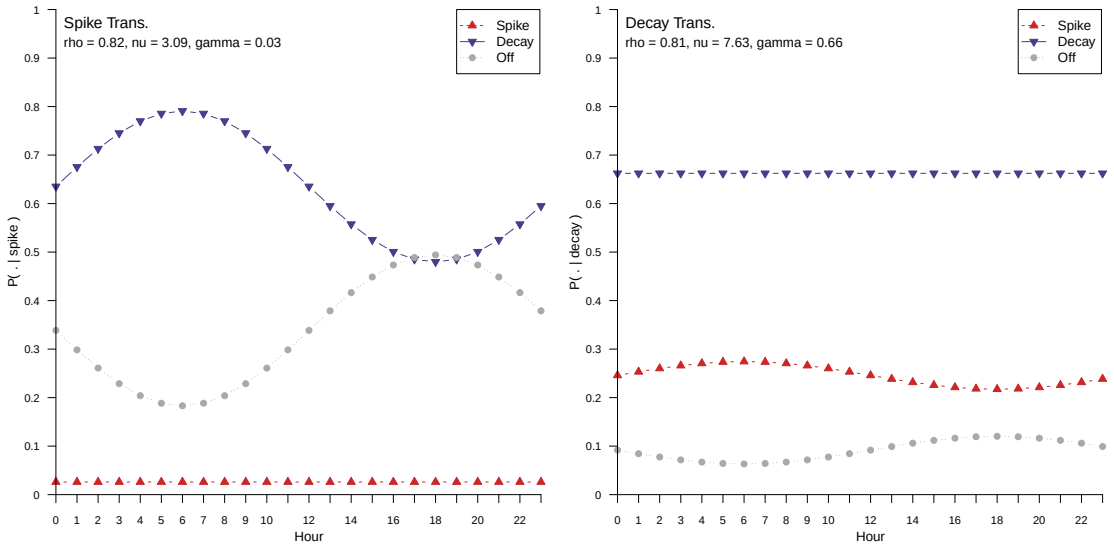

Figure 4.2: Transition probability profile for a machine transitioning out of a spike state (left), and transitioning out of a decay state (right).

The periods of inactivity for a machine are tied not to the specific malware that resides on it, but on its typical use. For example a desktop computer in a corporate network will most likely be turned on during business hours and turned off on nights and weekends. Conversely, a computer in a university laboratory may be turned off only for sporadic scheduled maintenance, but show definite patterns of higher use during daytime hours. Other home desktops or laptops may show much more sporadic periods of activity among long stretches (days or weeks) of inactivity. In most cases, the time spent turned off does not really follow a Markov interpretation, as in for example a geometric decay. So, instead of parameterizing an off-to-off transition $\pi_{o|o}(t)$ using the three-state transition matrix parameterization, contiguous blocks of off-to-off state transitions are treated as realizations from a Poisson(λ_o) distribution. This leads to a more symmetric distribution around the mean λ_o than a geometric interpretation. The periodic function $p(\rho_o, \nu_o, t^*)$ is then used to determine the probability of the first state seen after a run of off states concludes. Let c_t be the number of consecutive off-to-off transition states following an initial transition to an off state at time t. The

model for off-state transitions is:

$$\begin{aligned} c_t &\sim \text{Pois}(\lambda_o) \\ \pi_{o|s}(t) &= 1 - p(\rho_o, \nu_o, t^*) \\ \pi_{o|d}(t) &= p(\rho_o, \nu_o, t^*) \end{aligned}$$

These parameterizations are flexible for capturing individual differences in the patterns of behavior across machines. As part of the estimation process, the parameterizations themselves could also be estimated, in terms of how to combine the conditional periodic and aperiodic probabilities to create the transition probabilities. However in the initial single-host model, the parameterizations were fixed.

4.2 Birth process

Prior information regarding birth times (as opposed to start times) comes from security experts. The UDP scanning signature for Conficker-C was not observed prior to March 3rd, 2009. An analysis of the P2P bootstrapping technique indicated that new software updates could propagate throughout the botnet in approximately 24 hours. The Conficker-C update appeared to take a tiered approach, with a small number of infected machines updating on March 3rd, and then propagating the new instructions to the rest of the botnet on or near the 17th of March. Conficker-C was known to propagate only from infected Conficker-A and Conficker-B hosts, and security experts estimated that the population of those infections was dropping due to clean-ups. So in effect the Conficker-C worm had two major growth spurts; the initial trial test on the 3rd of March and the resulting botnet-wide communication on the 17th, with generally a 24-hour window representing saturation of instructions amongst the botnet. However, the 24-hour estimate does not take into account the user activity of turning off machines infected with Conficker-A and Conficker-B prior to either of the large-scale software updates, and leaving these machines turned off for a long period of time. These machines would obtain updates as they "woke up" and checked in, and would represent a longer tail in births than the 24-hour window.

Figure 4.3 shows the distribution of the first observed activity by $/24$ for the data set. The start times are augmented to incorporate new machines starting at later times within a network block, but only for the

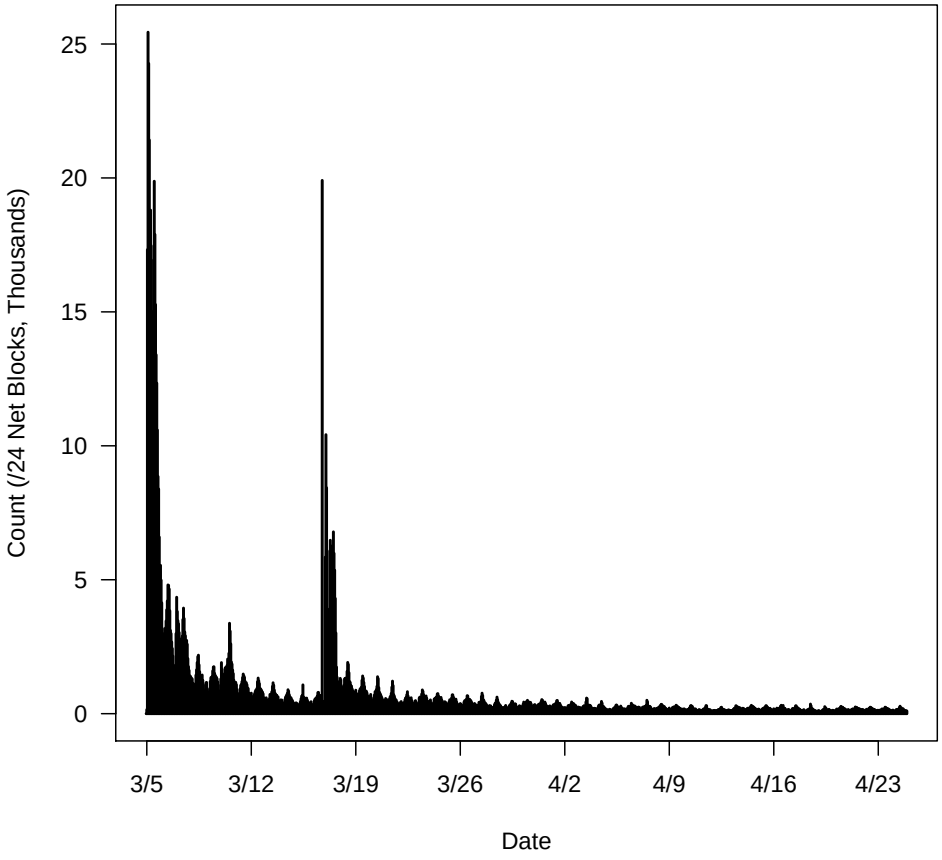

Figure 4.3: Augmented start times for $/24$ net blocks in the Conficker-C population.

possibility of an increased activity period occurring on March 17th, 2009. $28,752$ network blocks with an empirical start time initiated prior to March 17th also showed an increase in the mean active scan rate after March 17th of 4 or more hits per hour. In the figure, the start times for these net blocks were considered a split case between the early start date and the March 17th surge, weighted by the ratio of pre-17th to post-17th flow averages per hour. This allocation contributes to the large spike seen on the 17th. The pattern clearly shows two main initializations of activity, with surges spread across a few days, and trailing tails showing 24-hour periodicity.

Because the Conficker-C update took place across machines already infected with earlier variants of the virus, the distinction between "infection date" and "turn-on date" for these machines is hard to draw. For

simplification, we choose to model two discrete "birth" times for a machine, either the initial March 5th midnight (UTC) observation hour, or March 17th (actually March 16th, 19:00:00 UTC) with equal prior probability, and to model any delays in observations after those dates as off states representing an infected but inactive machine.

4.3 Death process and immunity

Although clean-ups do lead to use of fewer IP addresses in the population, a virus death due to clean-up of an infected machine is different than an observed "end time" for a net block in the data set. Infected machines can be turned off for long periods of time only to resurface when they are turned back on, or they can switch between several IP addresses due to DHCP practices. Furthermore, an infected machine is generally going to be turned off for some amount of time before it is cleaned of infection. Because the cleanup process after a machine's final shutdown is not observable (nor is it of interest in this study), we will define a "death" as the last active hour the infected machine admits before being taken offline for clean-up.

Figure 4.4 shows the distribution of the last observed activity by $/24$ for the data set. This figure excludes the final 48 hours of the observation window, as toward the end of the window, the last active hour becomes less indicative of a possible clean-up time and more indicative of ephemeral activity such as daily activity patterns, that would generally persist beyond the last time point of record. The plot shows that even at the $/24$ level, net blocks disappear regularly from view, indicating some measure of transience or decay in the population. The spike in end dates occurring prior to the March 17th surge is due to the anomalous initiation of 284 ephemeral net blocks during that hour, as discussed in Section 3.4.

Figure 4.5 shows the decay by week of $\hat{H}$, a naive estimate of the number of active hosts per hour (Weaver, 2010b), for the weeks after the March 17th surge. Each boxplot shows the distribution of the ratio $\hat{H}_{t+168}/\hat{H}_t$ for the 168 hours comprising the week. A value below 1.0 indicates geometric decay. The drop-off in activity appears most pronounced for the first week following the surge, and then levels out toward a more gradual decline. Although the P2P scanning was implemented to help strengthen the botnet's communication and resilience, it also made infected machines more visible to administrators upon its initiation. This suggests that the hazard of discovery for Conficker-C infections increased after the surge, and that machines would be in in danger of discovery while they were turned on. The computer's "death

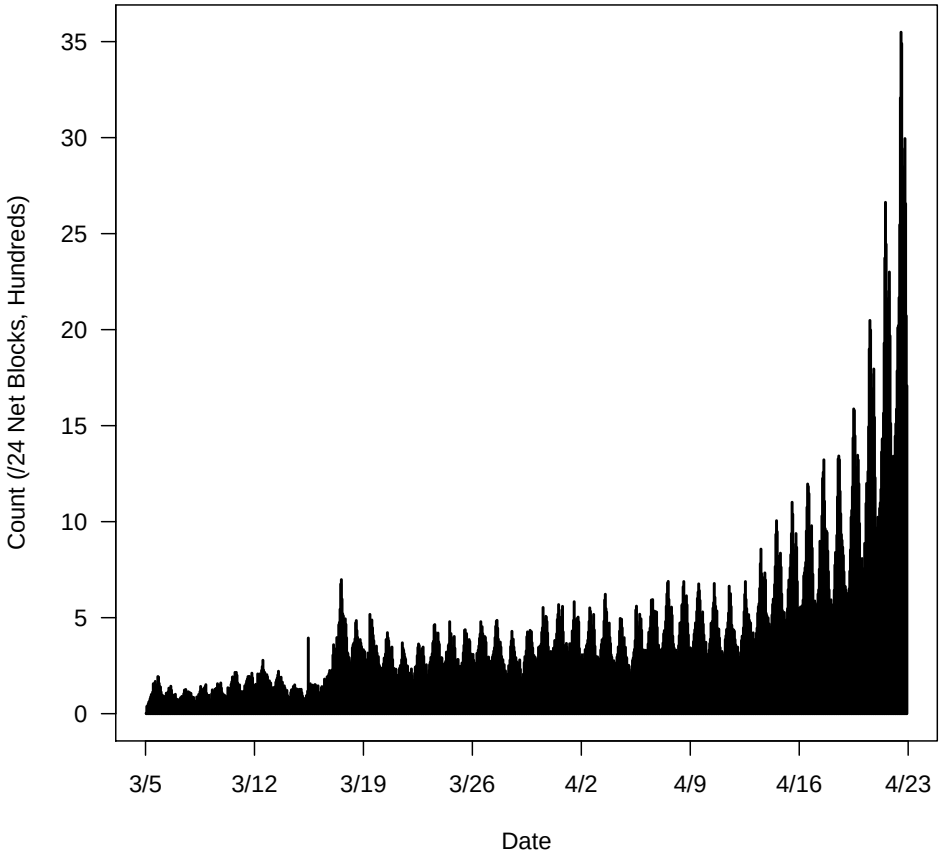

Figure 4.4: Last observed activity by $/24$ net block. This plot excludes net blocks with activity in the final 48 hours of the study. 52% of net blocks had a last active hour prior to the last two days of the observation window, and appear on the chart.

date" $t^\dagger$ can be set as either $t^\dagger = T + 1$, indicating that the infection survives at least up through the observation window of the study, or as the last active state: $t^\dagger = \max_t(\eta_t \in \{s, d\})$, indicating that the machine is turned off and cleaned at that time within the study.

A single survival rate for all machines is a simplification of the discovery risk among the population. In particular, most botnets have a core component of very persistent machines that are in little to no danger of being cleaned of infection. Computers running older operating systems or pirated software, poorly administered network computers, or little-used home computers can all be late adopters to patching against infections or cleaning infections when they occur, if either of these mitigations is used at all. To account for

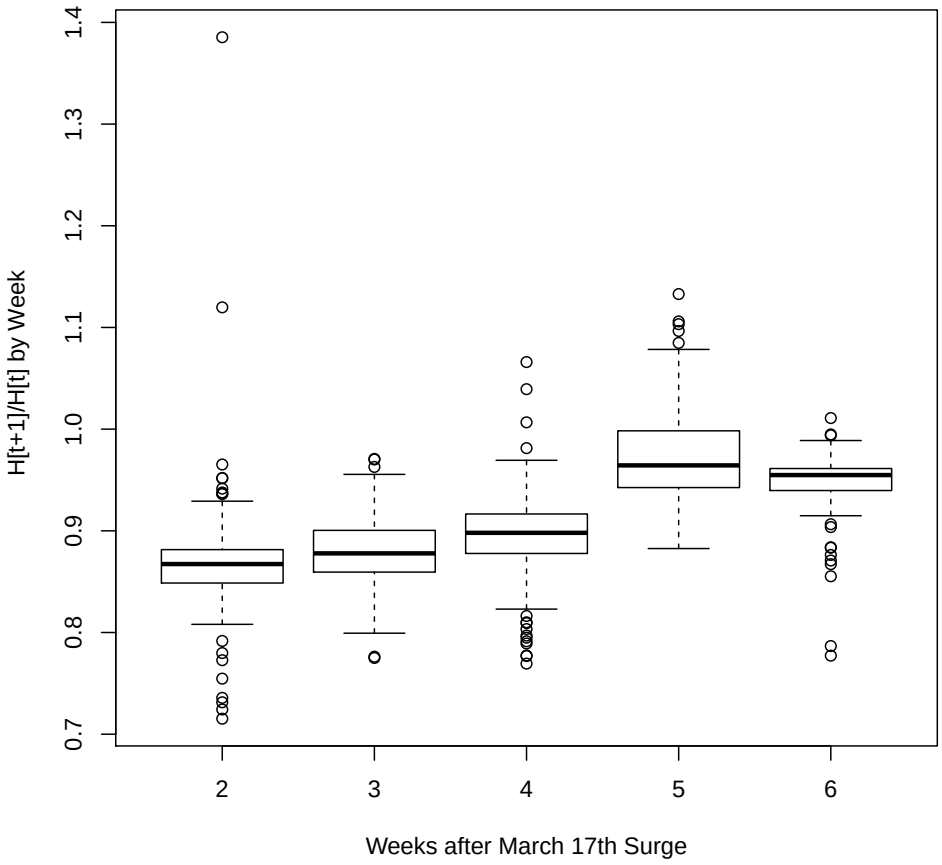

Figure 4.5: Decay rates by week (periodicity = 168 hours) for a preliminary estimate (Weaver, 2010b) of active hosts per hour. Values below 1.0 indicate a decline in population.

these resilient infections, a sub-population of the botnet is modeled as immune from discovery and remediation. Similar to the model introduced by Mariano and Kadane (2001), let $\epsilon_i = 1$ if machine i is immune from discovery, and $\epsilon_i = 0$ if it is not, where

$$\epsilon_i \quad \sim \quad \text{Ber}(\zeta).$$

Conditional on $\epsilon_i = 1$, transitions between states for machine i include only the 3×3 matrix described in Section 4.1.2. For $\epsilon_i = 0$, a new absorbing state representing cleanup is added to the transition model. Any transition probability from an active state to a non-cleanup state, $\pi_{\cdot|s}(t)$ or $\pi_{\cdot|d}(t)$, is multiplied by a survival

probability ϕ_t that describes the risk of being discovered and shut off for remediation in that hour, with a transition probability of $(1 - \phi_{t^\dagger})$ for transitioning to the cleanup state at the death date $t^\dagger$. For a preliminary treating of cleanups, we set $\phi_t = \phi$, a constant for all hours. Survival rates may also be correlated within related networks.

4.4 Summary of single-host parameters

Parameter		Description	Def. Prior	MCMC	Proposal dist.
θ	q	Baseline active scan rate	$\Gamma(1.5, 3.0)$	Gibbs	$\Gamma(\cdot, \cdot)$
	ω	Multiplier of baseline rate during a spike	$\beta^*(10, 50, 1, 24)$	M-H	$N^*(\cdot, 0.70)$
	α	Geometric decay rate per hour from spike to baseline	$\beta(10, 15)$	M-H	$N^*(\cdot, 0.05)$
ψ	λ_o	Mean off-to-off state transition run (unused if γ_o is used instead)	$\beta^*(1, 1, 6, 1000)$	M-H	$N^*(\cdot, 3.0)$
	ρ_o	Maximal periodic probability for off-state transitions	$\beta(1, 1)$	M-H	$N^*(\cdot, 0.15)$
	ν_o	Periodic scaling parameter for off-state transitions	$\Gamma(1, 2)$	M-H	$N^*(\cdot, 0.25)$
	γ_o	Aperiodic parameter for off-state transitions (unused if λ_o is used instead)	$\beta(1, 1)$	Gibbs	$\beta(\cdot, \cdot)$
	ρ_s	Maximal periodic probability for spike-state transitions	$\beta(1, 1)$	M-H	$N^*(\cdot, 0.15)$
	ν_s	Periodic scaling parameter for spike-state transitions	$\Gamma(1, 1)$	M-H	$N^*(\cdot, 0.25)$
	γ_s	Aperiodic parameter for spike-state transitions	$\beta(1, 100)$	Gibbs	$\beta(\cdot, \cdot)$
	ρ_d	Maximal periodic probability for decay-state transitions	$\beta(1, 1)$	M-H	$N^*(\cdot, 0.15)$
	ν_d	Periodic scaling parameter for decay-state transitions	$\Gamma(1, 2)$	M-H	$N^*(\cdot, 0.5)$
	γ_d	Aperiodic parameter for decay-state transitions	$\beta(1, 1)$	Gibbs	$\beta(\cdot, \cdot)$
	ϵ	Immune ($= 1$) vs. non-immune ($= 0$) indicator	Ber(ζ)	Gibbs	Discrete
$\boldsymbol{\eta}$	η_t	Active state (off, spike, decay) at time t	Discrete	M-H	Discrete
	t^β	Initial infection date	Discrete	Gibbs	Discrete
	$t^\dagger$	Cleanup (death) date	Discrete	Gibbs	Discrete
	ϕ_t	Survival probability per active hour for non-immune machines	$\beta(2, 1)$	Gibbs	$\beta(\cdot, \cdot)$
	ζ	Population proportion of immune machines	$\beta(2, 4)$	Gibbs	$\beta(\cdot, \cdot)$

Table 4.1: Summary of parameters, priors and MCMC steps for the single-host model. A $(\cdot)$ indicates that the proposal value depends on the current state of the chain. The function $\beta^*(\alpha, \beta, i, j)$ is a Beta(α, β) distribution scaled to the range $[i, j]$. The function $N^*(\cdot, \sigma^2)$ is a normal distribution centered at the current value, truncated appropriately to the values with non-zero prior probability.

Table 4.1 provides a summary of the parameters included in the single-host model, as well as the population parameters for immunity and survival rates. Prior distributions listed are those used in the MCMC documented in this chapter. They will be discussed in more detail in section 4.5.2. The type of MCMC step, Metropolis-Hastings or Gibbs, is also listed for each parameter, as well as proposal distributions for each parameter. These will be discussed further in section 4.5.3.

4.5 Estimation

Posterior distributions for the generation parameters of a single host can be estimated on a machine-by-machine basis, using Markov Chain Monte Carlo techniques designed for data augmentation (Tanner and Wong, 1987). Let η_t be the state of the machine at time $t : \eta_t \in \{o, s, d\}$, and define $\boldsymbol{\eta} = \{\eta_0, \ldots, \eta_T, t^\beta, t^\dagger\}$. Let y_t be the observed count at time t, and define $\boldsymbol{y} = \{y_0, \ldots, y_T\}$. Let $\psi = (\rho_{\{o,s,d\}}, \nu_{\{o,s,d\}}, \gamma_{\{o,s,d\}}, \lambda_o, \epsilon)$ be the collection of the transition probability parameters and $\theta = (q, \omega, \alpha)$ be the collection of rate parameters. An updating strategy iterates between the following steps:

1. Update parameters ψ from the complete conditional distribution $p(\psi \mid \boldsymbol{\eta})$ using a combination of Gibbs sampling and Metropolis-Hastings sampling.

2. Update parameters θ from the complete conditional distribution $p(\theta \mid \boldsymbol{\eta}, \boldsymbol{y})$ using a combination of Gibbs sampling and Metropolis-Hastings sampling.

3. Update a randomly chosen state $\eta_{t_0} \in \{\eta_{t^\beta}, \ldots, \eta_{t^\dagger}\}$ from the distribution $\pi(\eta_{t_0} \mid \boldsymbol{\eta}, \psi, \theta, \boldsymbol{y})$, using Metropolis-Hastings sampling.

4. Update birth state, death state and immunity designation jointly from the complete conditional distribution $\pi(t^\beta, t^\dagger, \epsilon \mid \boldsymbol{\eta}, \theta, \psi)$ using Gibbs sampling.

Steps (3) and (4) can be considered a data augmentation step, where the states facilitate the parameter updating described in steps (1), and (2). The relative frequency of steps chosen in each sequence can be changed to facilitate mixing in the chain. Sections 4.5.1, 4.5.2, and 4.5.3 describe the details of these steps. Section 4.5.4 expands the estimation strategy to include population parameters for a collection of machines where H is known and the relationship between machines and network touchpoints is exactly one-to-one.

4.5.1 Full likelihood

For machines with no off-to-off state transitions, the likelihood $\ell(\psi, \theta, \boldsymbol{\eta}; \boldsymbol{y})$ can be written as the product of an initial probability $p(\eta_0)$, state-to-state transition probabilities, and Poisson count probabilities conditional

on the rates defined for each underlying state:

$$\ell(\psi, \theta, \boldsymbol{\eta}; \boldsymbol{y}, \boldsymbol{\phi}, \epsilon) \quad = \quad p(\eta_{t^\beta}) \prod_{t=t^\beta+1}^{t^\dagger} p(\eta_t \mid \eta_{t-1}, \psi, \phi_t, \epsilon) \prod_{t=t^\beta}^{t^\dagger} p(y_t \mid \eta_t, \theta) \tag{4.5}$$

Let $\mathcal{O}$, $\mathcal{D}$, and $\mathcal{S}$ be the set of hours t in off, decay, or spike states, respectively. Let $\mathcal{T}_{s_2|s_1}$ be the set of hours t that transition to state s_2 at $t+1$ from state s_1 at t, when at least one of s_1 or s_2 is active. Let $\mathcal{T}_{o|o}$ be the set of times t that start each string of consecutive off states, and let c_t be the count of off-to-off transition states following time $t \in \mathcal{T}_{o|o}$. Define the following sets of lag states relative to spike and decay states:

$\mathcal{L}_B$: all decay times t that follow an off state with no intervening spike states. At any hour $t \in \mathcal{L}_B$, the infected machine broadcasts at the baseline rate q.

$\mathcal{L}_k, k = 0, 1, 2, ...$: all states that follow k steps after a spike state, with $k = 0$ indicating spike states themselves. At any hour $t \in \mathcal{L}_k$, the infected machine broadcasts at the rate $q(1 + \alpha^k(\omega - 1))$.

Let $T^\delta_\beta = t^\dagger - t^\beta + 1$ be the total machine lifetime, and let $\phi_{T+1} = 0$ to indicate a machine that has not been cleaned as of the latest hour in the data set. Collecting like terms for immunity, survivability, and hourly scan rates, as well as incorporating off-to-off transitions, the full likelihood can then be expressed as the following product:

$$\begin{aligned}
\ell(\psi, \theta, \boldsymbol{\eta}, \boldsymbol{\phi}, \zeta; \boldsymbol{y}) \quad &= \quad \zeta^\epsilon \left[(1-\zeta)(1-\phi_{t^\dagger}) \prod_{t \in \mathcal{S} \cup \mathcal{D}} \phi_t \right]^{1-\epsilon} \\
&\times \quad \prod_{t \in \mathcal{T}_{o|o}} \frac{e^{-\lambda_o} \lambda_o^{c_t}}{c_t!} \\
&\times \quad \left[\prod_{s_1, s_2 \in \{o,s,d\} - \{s1=s2=o\}} \prod_{t \in \mathcal{T}_{s_1|s_2}} \pi_{s_1|s_2}(t) \right] \\
&\times \quad \left[\prod_{t \in \mathcal{O}} 1_{\{y_t = 0\}} \prod_{t \in \mathcal{L}_B} \frac{e^{-q} q^{y_t}}{y_t!} \prod_{k=0}^{T^\delta_\beta} \prod_{t \in \mathcal{L}_k} \frac{e^{q(1+\alpha^k(\omega-1))} \left[q(1+\alpha^k(\omega-1)) \right]^{y_t}}{y_t!} \right]
\end{aligned} \tag{4.6}$$

The first term is the additional survival rates multiplied into the transition probabilities to incorporate the cleanup state for non-immune machines. The second term is the Poisson likelihood for each string of consecutive off-to-off state transitions. The third term is the product of 3-state transition probabilities

for each collection of observed state-to-state transitions that are not off-to-off transitions. Recall that each transition probability is a function of ψ. The final term is the same product of Poisson probabilities that was expressed as a time series in Equation 4.5, but collected together into subsets of hours that have equal means as a function of the lag k from a spike or off state. The form of Equation 4.6 is useful for collecting like terms in the derivation of complete conditional distributions for system parameters.

4.5.2 Priors

Table 4.1 provides a summary of the prior distributions used for parameters in the single host model. When possible, prior hyperparameters are informed by previous knowledge, and parametric forms are chosen with the physical limits of the parameter in mind as well as ease of functional form for conditional distributions.

Distribution	Hyperparameters	p.d.f. $\pi(\varphi \mid \cdot)$	$E(\varphi)$
Gamma (Γ)	$k > 0; s > 0$	$\frac{1}{s^k \Gamma(k)} \varphi^{k-1} e^{-\varphi/s}$	ks
Beta (β)	$a > 0; b > 0$	$\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \varphi^{a-1}(1-\varphi)^{b-1} 1_{\{0 \le \varphi \le 1\}}$	$\frac{a}{a+b}$
Scaled Beta (β^*)	$a > 0; b > 0$ $-\infty < l < u < \infty$	$\frac{\Gamma(a+b)}{(u-l)\Gamma(a)\Gamma(b)} \left(\frac{\varphi-l}{u-l}\right)^{a-1} \left(1 - \frac{\varphi-l}{u-l}\right)^{b-1} 1_{\{l \le \varphi \le u\}}$	$\frac{ua+lb}{a+b}$

Table 4.2: Summary of parametric prior distributions and hyperparameters

In addition to general finite discrete distributions, the model makes use of three parametric families for prior distributions: the Beta (β) distribution parameterized by success shape a and failure shape b, the Gamma (Γ) distribution parameterized by shape parameter k and scale parameter s, and the scaled Beta (β^*) distribution which uses success and failure shapes a and b along with a set of scaling bounds, l and u. For a model parameter φ, any relevant associated hyperparameter is designated using a subscript, for example, k_q and s_q to designate the shape and scale parameters for a Gamma distribution used as a prior for the parameter q. Table 4.2 summarizes the functional forms of these parametric distributions in terms of their hyperparameters. The Gamma distribution $\Gamma(k, s)$ is differentiated from the analytic Gamma function $\Gamma(x)$ by specification of the two hyperparameters.

Prior distributions for the rate parameters θ are informed by the results of sandbox testing and network

telescope parameters. The baseline rate $q > 0$ uses a Gamma$(k_q = 1.5, s_q = 3.0)$ distribution, yielding a prior mean of 4.5 with reasonable deviations allowing for different network speeds and uncertainty about the size of the monitored network for the study.

The spike rate ω was chosen to reflect both the results of the sandbox tests, and the dependence of UDP spikes on the number of open network connections. Information collected from the sandbox tests indicated that rate spikes between two and ten times the baseline rate were common, with the possibility of much larger spikes occurring due to opening of many multiple network connections. A hard limit for the number of simultaneous scanning threads from the reverse engineering led to setting a strict maximum value m_ω of the mean spike multiplier. The maximum number of allowable open threads is 32 in the botnet's source code, although most day-to-day user activity rarely initiate this many threads at once. In initial MCMC runs the maximum value was set to 24, and preliminary estimates based on EM for a test set of probable individual machines showed no initial spike values greater than 16 (see next section). The spike rate was also restricted to take values greater than 1 for interpretability. A Scaled Beta$(a_\omega = 10, b_\omega = 50, l_\omega = 1, u_\omega = 24)$ is used to model this distribution.

The prior on the geometric decay rate α uses a Beta$(a_\alpha = 10, b_\alpha = 15)$ distribution in order to emphasize the empirical observation of decay to baseline occurring generally within 2 to 6 hours after a spike occurs. Low values for α also help to differentiate between possible ambiguity in the model between spike and decay states, that can occur if decay rates are slow enough ($\alpha \approx 1$) to admit long strings of high values. Figure 4.6 shows density plots of these three distributions.

Prior distributions for the transition probability parameters reflect the uncertainty about the distribution of different kinds of day-to-day activity among the population of infected hosts. Uniform priors (Beta or Scaled Beta distributions with $a = b = 1$) are chosen for the average off-to-off state transition run λ_o, the maximal periodic probability ρ for each of off, spike and decay states, and the aperiodic transition probability γ_d for decay states, to allow for a range of activity types from sporadic activity, to workday patterns, to machines that are rarely turned off. Periodic scaling parameters ν for off and decay states are modeled with a Gamma$(k_{\nu_o} = k_{\nu_d} = 1, s_{\nu_o} = s_{\nu_d} = 2)$, where the scale parameter 2 is also the value of the prior mean. These priors are strongly informative due to the fact that the parametric form of $p(\rho_a, \nu_a, t^*)$ flattens out appreciably as ν increases, with changes in values greater than 10 causing little

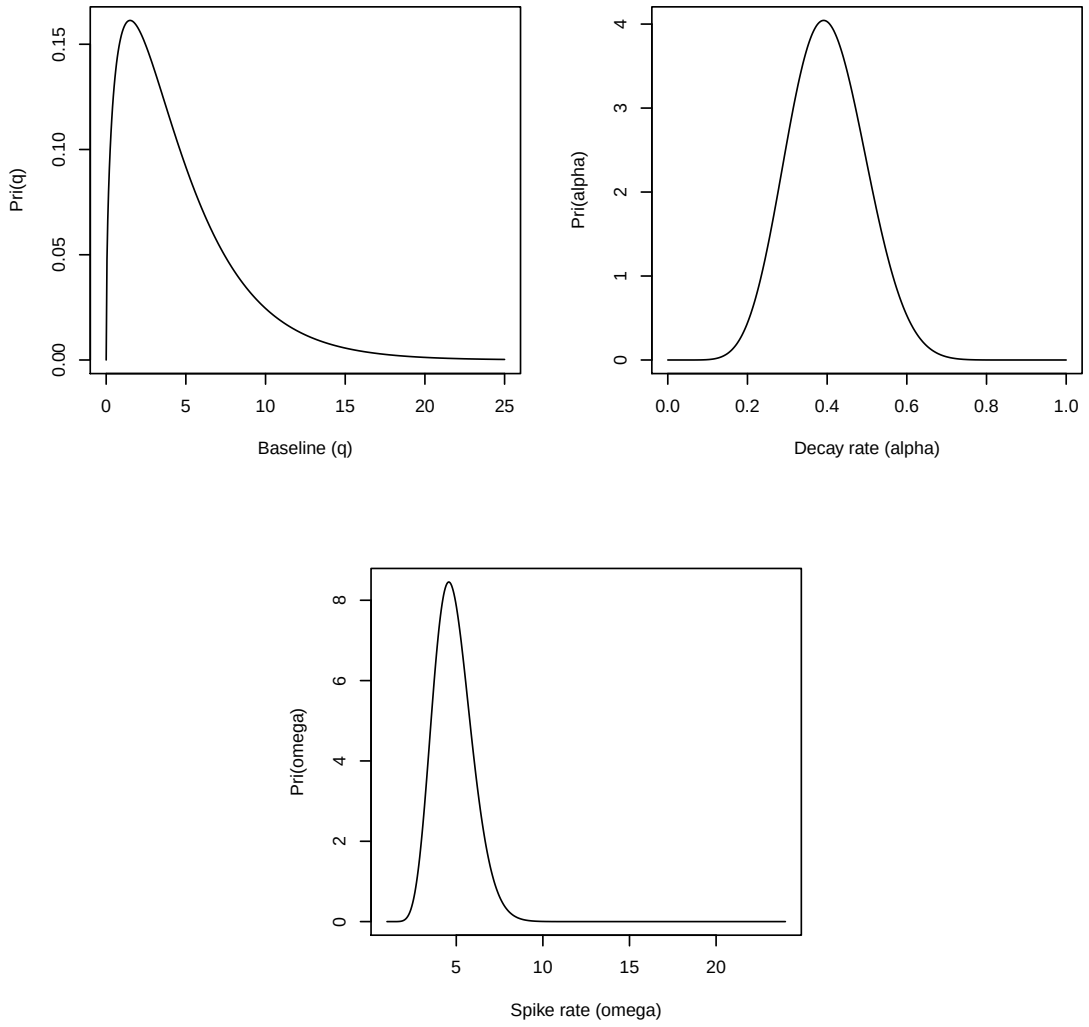

Figure 4.6: Prior distributions for the Poisson rate parameters q, ω, and α.

change to the functional shape of the transition probability. The prior for ν for spike states is chosen as a Gamma$(k_{\nu_s} = 1, s_{\nu_s} = 1)$, which focuses slightly more weight on values that admit a larger 24-hour time-dependent curve when transitioning away from a spike to either a decay or off state. This parameterization should also help to coax the underlying states of aperiodic baseline machines toward progressions of decay states as opposed to spike states.

The exception to these non-informative priors is the prior for the aperiodic parameter for spike to spike state transitions, γ_s, which is modeled as a Beta$(a_{\gamma_s} = 1, b_{\gamma_s} = 100)$ in order to keep the interpretability

of spike states as events that rarely occur in sequence. Setting spike-to-spike transitions as rare events disambiguates (to some extent) the modes in likelihood that can occur between a baseline value of b with many decay states in a row, as opposed to a baseline value of b/c and a spike multiplier of c with many spike states in a row, although this ambiguity does still appear in the test data for machines that have very few hours with non-zero counts.

For discrete state values, η_0, t^β and $t^\dagger$, discrete uniform priors are used. For η_0, equal probability among off, decay or spike states is assumed. The birthdate is assumed uniformly likely as either the start time $t = 0$ or the start of the March 17th surge at $t = 284$. A priori, death dates are assumed equally likely among the 1225 possibilities (including "survived beyond time $T = 1224$"). This could use improvement in future models, particularly to include the first few weeks past the March 17th surge as higher likelihoods for cleanup of a machine. Prior values for the population parameters ϕ_t and ζ are discussed in Section 4.5.4.

4.5.3 Conditional distributions and sampling strategies

Let $\varphi \in \{\psi, \theta, \boldsymbol{\eta}, \boldsymbol{\phi}, \zeta\}$ be a parameter in the model, and let $\{\psi, \theta, \boldsymbol{\eta}, \boldsymbol{\phi}, \zeta\}_{|\varphi}$ be the vector of all parameters except for φ. The complete conditional distribution of φ has the general form:

$$\pi(\varphi \mid \{\psi, \theta, \boldsymbol{\eta}, \boldsymbol{\phi}, \zeta\}_{|\varphi}, \boldsymbol{y}) \quad = \quad \frac{\pi(\varphi)\ell(\varphi, \{\psi, \theta, \boldsymbol{\eta}, \boldsymbol{\phi}, \zeta\}_{|\varphi}; \boldsymbol{y})}{\int_\varphi \pi(\varphi)\ell(\varphi, \{\psi, \theta, \boldsymbol{\eta}, \boldsymbol{\phi}, \zeta\}_{|\varphi}; \boldsymbol{y})}, \tag{4.7}$$

where all parameter values in Equation 4.6 except for φ are considered fixed constants. Any multiplicative terms not containing φ in the numerator product can also be pulled out from the denominator integral as constants and thus canceled out of the equation. The collection of prior and likelihood terms containing φ in the numerator of Equation 4.7 are called the *kernel* of the complete conditional distribution of φ. When the functional form of this kernel is recognizable as a parametric family (or the denominator integral can be computed analytically via other means), a new value φ_{m+1} can be sampled at $m + 1$-st iteration of the Markov Chain using a Gibbs step dependent on the current state of all other parameters. A *conjugate* conditional distribution has the same parametric form as the prior distribution. When the form of $\pi(\varphi \mid \cdot)$ is the same as that of $\pi(\varphi)$, then the calculation of this complete conditional distribution reduces to describing updated values for the prior hyperparameters, as a function of the current hyperparameter, the m-th state of parameters and the observed data $\boldsymbol{y}$. For conjugate Gibbs steps, denote by h^* the updated value of the prior

hyperparameter h.

When the kernel cannot be solved analytically, a Metropolis-Hastings step is used instead of a Gibbs step. The Metropolis-Hastings step generates a candidate value φ^c from a proposal distribution $g(\varphi \mid \varphi_m)$, and then sets $\varphi_{m+1} = \varphi^c$ with probability r:

$$r = \min\left\{1, \frac{\pi(\varphi^c)\ell(\varphi^c, \{\psi, \theta, \boldsymbol{\eta}, \boldsymbol{\phi}, \zeta\}_{|\varphi}; \boldsymbol{y})}{\pi(\varphi_m)\ell(\varphi_m, \{\psi, \theta, \boldsymbol{\eta}, \boldsymbol{\phi}, \zeta\}_{|\varphi}; \boldsymbol{y})} \frac{g(\varphi_m \mid \varphi^c)}{g(\varphi^c \mid \varphi_m)}\right\} \tag{4.8}$$

If φ^c is rejected, then the step concludes by setting $\varphi_{m+1} = \varphi_m$.

For Metropolis-Hastings steps for continuous parameters, a truncated Normal distribution is used for proposals. If $l < \varphi < u$ for a parameter φ, with a current iteration φ_m in the Markov chain, then the proposal distribution $g(\varphi^c \mid \varphi_m)$ with variance σ_φ^2 is

$$g(\varphi^c \mid \varphi_m, \sigma_\varphi^2, l, u) = \begin{cases} \left[\Phi\left(\frac{u-\varphi_m}{\sigma_\varphi}\right) - \Phi\left(\frac{l-\varphi_m}{\sigma_\varphi}\right)\right]^{-1} \frac{1}{\sqrt{2\pi}\sigma_\varphi} \exp\left[\frac{(\varphi^c-\varphi_m)^2}{2\sigma_\varphi^2}\right], & l < \varphi^c < u \\ 0, & \text{otherwise} \end{cases}$$

Here, $\Phi(\cdot)$ is the CDF of the Standard Normal distribution.

Updating θ

In this section, denote by $N_{\mathcal{A}}$ the cardinality of a set $\mathcal{A}$, and note that for the set $\mathcal{L}_k$ from Equation 4.6 (including the baseline set $\mathcal{L}_B$where $k = \infty$),

$$\prod_{t \in \mathcal{L}_k} \frac{e^{q(1+\alpha^k(\omega-1))}\left[q(1+\alpha^k(\omega-1))\right]^{y_t}}{y_t!} = \frac{\exp\left[-N_{\mathcal{L}_k} q(1+\alpha^k(\omega-1))\right]\left[q(1+\alpha^k(\omega-1))\right]^{\sum_{t\in\mathcal{L}_k} y_t}}{\prod_{t\in\mathcal{L}_k} y_t!},$$

where the denominator term is a function of the observed data alone and thus cancels in the kernel for any parameter φ. The range of positive probability for each parameter is specified in bold in the description; conditional distributions have value 0 outside the specified parameter ranges.

For $q > 0$: The kernel for the baseline rate q has the following form:

$$\pi(q \mid \boldsymbol{\eta}, \boldsymbol{y}, \theta_{|q}, k_q, s_q) \quad \propto \quad q^{(k_q - 1 + \sum_{T_\beta^\delta} y_t)} \exp\left[-\frac{q}{s_q}\left[(T_\beta^\delta - N_{\mathcal{O}}) + \sum_{k=0}^{T_\beta^\delta} N_{\mathcal{L}_k}\alpha^k(\omega - 1)\right]\right]$$

This is a conjugate conditional distribution; it has the functional form of a Gamma distribution with parameters

$$\begin{aligned} k_q^* &= k_q - 1 + \sum_{T_\beta^\delta} y_t; \\ s_q^* &= s_q\left[(T_\beta^\delta - N_{\mathcal{O}}) + \sum_{k=0}^{T_\beta^\delta} N_{\mathcal{L}_k}\alpha^k(\omega - 1)\right]^{-1}. \end{aligned}$$

Updated values of q can be sampled using a Gibbs step with the Gamma(k_q^*, s_q^*) distribution.

For $l_\omega < \omega < u_\omega$: The kernel for the spike rate ω has the following form:

$$\begin{aligned} &\pi(\omega \mid \boldsymbol{\eta}, \boldsymbol{y}, \theta_{|\omega}, l_\omega, u_\omega, a_\omega, b_\omega) \\ &\qquad \propto \quad \left(\tfrac{\omega - l_\omega}{u_\omega}\right)^{a_\omega - 1}\left(\tfrac{l_\omega - \omega}{u_\omega}\right)^{b_\omega - 1} \exp\left[-\omega q \sum_{k=0}^{T_\beta^\delta} N_{\mathcal{L}_k}\alpha^k\right] \prod_{k=0}^{T_\beta^\delta}\left(1 + \alpha^k(\omega - 1)\right)^{\sum_{t \in \mathcal{L}_k} y_t}. \end{aligned}$$

This distribution can be sampled using a Metropolis-Hastings step with a truncated Normal proposal in the interval $[l_\omega, u_\omega]$.

For $0 < \alpha < 1$: The kernel for the decay rate α has the following form:

$$\pi(\alpha \mid \boldsymbol{\eta}, \boldsymbol{y}, \theta_{|\alpha}, a_\alpha, b_\alpha) \quad \propto \quad \alpha^{a_\alpha - 1}(1 - \alpha)^{b_\alpha - 1} \exp\left[-q(\omega - 1)\sum_{k=0}^{T_\beta^\delta} N_{\mathcal{L}_k}\alpha^k\right] \prod_{k=0}^{T_\beta^\delta}\Big(1 + \alpha^k(\omega - 1)\Big)^{\sum_{t \in \mathcal{L}_k} y_t}.$$

This distribution can be sampled using a Metropolis-Hastings step with a truncated Normal proposal in the interval $[0, 1]$.

Updating ψ

For conditional distributions of the transition probability parameters ρ, ν, and γ, it is useful to consider a set of hours comprising "success" vs. "failure" transition states for each parameter, relative to the format of the transition probabilities $\pi_{\cdot|\cdot}(t)$. For the function $g(\varphi)$ equal to either the identity function or the periodic transition function $p(\rho_a, \nu_a, t^*)$, a transition $\pi_{s_1|s_2}(t)$ at hour t is a "success" transition for a parameter φ if φ enters into the probability as $g(\varphi)$ alone, and it is a "failure" transition if φ enters into the probability through a $(1 - g(\varphi))$ term instead. Denote by $\mathcal{S}_{\varphi,a}$ and $\mathcal{F}_{\varphi,a}$ the set of hours comprising success transitions and failure transitions for φ_a from state a, respectively. As an example, Section 4.1.2 outlines the transitions away from a spike state as:

$$\begin{aligned}
\pi_{o|s}(t) &= (1-\gamma_s)(1-p(\rho_s, \nu_s, t^*)) \\
\pi_{s|s}(t) &= \gamma_s \\
\pi_{d|s}(t) &= (1-\gamma_s)(p(\rho_s, \nu_s, t^*))
\end{aligned}$$

Using this parameterization and the set notation $\mathcal{T}_{\cdot|s}$ from Section 4.5.1, the set of success transitions for ρ_s is the set $\mathcal{S}_{\rho,s} = \mathcal{T}_{d|s}$, and the set of failure transitions for ρ_s is the set $\mathcal{F}_{\rho,s} = \mathcal{T}_{o|s}$. A similar argument shows that these are the success and failure sets for ν_s as well. The set of success transitions for γ_s is the set $\mathcal{S}_{\gamma,s} = \mathcal{T}_{s|s}$, and the set of failure transitions for γ_s is the set $\mathcal{F}_{\gamma,s} = \mathcal{T}_{o|s} \cup \mathcal{T}_{d|s}$.

For $0 < \rho_a < 1$: Using a Beta(a_{ρ_a}, b_{ρ_a}) prior distribution, the kernel for the periodic amplitude ρ_a for a state $a \in \{o, s, d\}$ has the following form:

$$\pi(\rho_a \mid \boldsymbol{\eta}, \psi_{|\rho_a}, a_{\rho_a}, b_{\rho_a}) \propto (\rho_a)^{a_{\rho_a}-1+N_{\mathcal{S}_{\rho,a}}} (1-\rho_a)^{b_{\rho_a}-1} \prod_{t \in \mathcal{F}_{\rho,a}} \left(1 - \frac{\rho_a}{\nu_a + 2}\left[\sin(2\pi t^*/24) + \nu_a + 1\right]\right).$$

This distribution can be sampled using a Metropolis-Hastings step in the Markov chain. Note that as $\nu_a \to \infty$, this conditional distribution approaches a conjugate Beta distribution with parameters $a^*_{\rho_a} = a_{\rho_a} + N_{\mathcal{S}_{\rho,a}}$ and $b^*_{\rho_a} = b_{\rho_a} + N_{\mathcal{F}_{\rho,a}}$. This distribution can also be sampled using a rejection method based on this Beta distribution.

For $\nu_a > 0$: The kernel of the periodic baseline ν_a for state $a \in \{o, s, d\}$ has the following form:

$$\begin{aligned}\pi(\nu_a \mid \boldsymbol{\eta}, \psi_{|\nu_a}, k_{\nu_a}, s_{\nu_a}) &\propto (\nu_a)^{k_{\nu_a}-1} \exp\left[\frac{\nu_a}{s_{\nu_a}}\right] \\ &\times \prod_{t \in \mathcal{S}_{\nu,a}} \left(\frac{\rho_a}{\nu_a + 2}\left[\sin(2\pi t^*/24) + \nu_a + 1\right]\right) \\ &\times \prod_{t \in \mathcal{F}_{\nu,a}} \left(1 - \frac{\rho_a}{\nu_a + 2}\left[\sin(2\pi t^*/24) + \nu_a + 1\right]\right)\end{aligned}$$

This distribution can be sampled using a Metropolis-Hastings step with a truncated Normal proposal distribution.

For $0 < \gamma_a < 1$: The kernel of the aperiodic transition parameter γ_a for $a \in \{o, s, d\}$ has the following form:

$$\pi(\gamma_a \mid \boldsymbol{\eta}, \psi_{|\gamma_a}, a_{\gamma_a}, b_{\gamma_a}) \propto (\gamma_a)^{a_{\gamma_a}-1+N_{\mathcal{S}_{\gamma,a}}} (1-\gamma_a)^{b_{\gamma_a}-1+N_{\mathcal{F}_{\gamma,a}}}$$

This is a conjugate conditional distribution; it has the functional form of a Beta distribution with parameters

$$\begin{aligned}a^*_{\gamma_a} &= a_{\gamma_a} + N_{\mathcal{S}_{\gamma,a}} \\ b^*_{\gamma_a} &= b_{\gamma_a} + N_{\mathcal{F}_{\gamma,a}}\end{aligned}$$

Updated values of γ_a can be sampled using a Gibbs step with the Beta($a^*_{\gamma_a}, b^*_{\gamma_a}$) distribution.

For $l_{\lambda_o} < \lambda_o < u_{\lambda_o}$: Using the scaled Beta($a_{\lambda_o} b_{\lambda_o}, l_{\lambda_o}, u_{\lambda_o}$) prior distribution, the kernel for the mean number of hours spent in consecutive off states, λ_o, has the following form:

$$\pi(\lambda_o \mid \boldsymbol{\eta}, \boldsymbol{y}, \psi_{|\lambda_o}, l_{\lambda_o}, u_{\lambda_o}) \propto \exp\left[-N_{\mathcal{T}_{o|o}} \lambda_o\right] \lambda_o^{a_{\lambda_o}-1+\sum_{t \in \mathcal{T}_{o|o}} c_t} (1-\lambda_o)^{b_{\lambda_o}-1}$$

When $b_{\lambda_o} = 1$, as in the case for example with a Uniform($l_{\lambda_o}, u_{\lambda_o}$) prior, the complete conditional distribution $\pi(\lambda_o \mid \cdot)$ is the kernel of a Gamma distribution restricted to the interval $[l_{\lambda_o}, u_{\lambda_o}]$. The parameters

are

$$\begin{aligned} k_{\lambda_o} &= a_{\lambda_o} + \sum_{t \in \mathcal{T}_{o|o}} c_t \\ s_{\lambda_o} &= [N_{\mathcal{T}_{o|o}}]^{-1} \end{aligned}$$

It can be sampled in the chain using either a Metropolis-Hastings step, or a Gibbs step using for example the rejection method on an unrestricted Gamma distribution with the same parameters. The only difference between using a Gibbs step with a rejection method, and using a Metropolis-Hastings step with the same Gamma distribution as a proposal, is that the Gibbs step would repeatedly draw proposal values until it obtained a value that was accepted, whereas each Metropolis-Hastings step would perform only one draw and accept or reject it.

Sampling of ϵ is discussed in the section addressing updating birth and death times.

Updating η_t

It is useful to consider the likelihood as its time series progression from Equation 4.5 for updating states, as the conditional distribution of a state at hour t depends on ψ, θ, and also on subsections of $\boldsymbol{\eta}_{|\eta_t}$. Let $t^\beta \leq \eta_{t_0} \leq t^\dagger$ be a state selected to be updated. Because of the geometric link between spike and decay states, changing the state at time t_0 also changes the mean scan rates in neighboring hours. Define u_0 and v_0 as the interval affected by a change of state at time t_0. The definition of this interval depends on the value of the observed count at the chosen state.

For $y_{t_0} > 0$**:** When there is no possibility of transitioning to an off state, the interval of affected states is defined as:

$$\begin{aligned} u_0 &= \max_t[(t^\beta, \{t^\beta \leq t < t_0 : \eta_t \in \{o, s\}\})] \\ v_0 &= \min_t[(t^\dagger, \{t_0 < t \leq t^\dagger : \eta_t \in \{o, s\}\})] \end{aligned} \tag{4.9}$$

When $y_t > 0$, the conditional path likelihood $[p(t, a; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{y})]_u^v$ is defined by the product:

$$[p(t, a; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{y})]_u^v = \pi_{a|\eta_{t-1}}(t-1)\pi_{\eta_{t+1}|a}(t)p(y_t \mid \eta_t = a, \theta) \prod_{\ell=u, \ell\neq t}^{v} p(y_\ell \mid \eta_\ell, \theta)$$

This is the product of the 3×3 transition probabilities to and from state η_t, multiplied by the product of the likelihood of the data given the scan rate parameters for the entire block between u and v.

For $y_{t_0} = 0$: When there is a possibility to transition to or from an off state, the interval of affected states depends on the values at $t_0 - 1$ and $t_0 + 1$. if $\eta_{t_0-1} \neq o$, then u_0 is defined as in Equation 4.10. If $\eta_{t_0} - 1 = o$ then,

$$u_0 \quad = \quad \max_t[(t^\beta, \{t^\beta \leq t < t_0 : \eta_t \in \{d, s\}\})]$$

Similarly, if $\eta_{t_0+1} \neq o$, then v_0 is defined as in Equation 4.10, but if $\eta_{t_0+1} = o$ then

$$v_0 \quad = \quad \min_t[(t^\dagger, \{t_0 < t \leq t^\dagger : \eta_t \in \{d, s\}\})]$$

These new definitions find the start or end of the run of off states that abut t_0, as opposed to the start or end of the run of decay states. When $y_{t_0} = 0$, the conditional path likelihood $[p(t, a; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{\phi}, \epsilon, \boldsymbol{y})]_u^v$ is defined by the product:

$$\begin{aligned}
[p(t, a; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{\phi}, \epsilon, \boldsymbol{y})]_u^v \quad &= \quad \prod_{t \in \mathcal{T}_{o|o}, u \leq t \leq v} \frac{e^{-\lambda_o}\lambda_o^{c_t}}{c_t!} \\
&\times \quad [(\phi_{t-1})^{(1-\epsilon)1_{\{\eta_{t-1}\neq o\}}}\pi_{a|\eta_{t-1}}(t-1)]^{1_{\{(\eta_{t-1}, a)\neq(o,o)\}}} \\
&\times \quad [(\phi_{t+1})^{(1-\epsilon)1_{\{a\neq o\}}}\pi_{\eta_{t+1}|a}(t)]^{1_{\{(a,\eta_{t+1})\neq(o,o)\}}} \\
&\times \quad p(y_t \mid \eta_t = a, \theta) \prod_{\ell=u, \ell\neq t}^{v} p(y_\ell \mid \eta_\ell, \theta)
\end{aligned} \tag{4.10}$$

This product accounts for the transition probabilities that may change within the affected interval due to addition or subtraction of η_{t_0} from a consecutive run of off states.

The kernel of the complete conditional distribution of η_{t_0} is a discrete distribution where the probability

of $\eta_{t_0} = a$ is proportional to the appropriate conditional path likelihood $[p(t_0, a; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{y})]_{u_0}^{v_0}$. The full conditional distribution is normalized by dividing by the sum of conditional path likelihoods for $a \in \{o, s, d\}$:

$$\pi(\eta_{t_0} = a \mid \boldsymbol{\eta}, \theta, \psi, \boldsymbol{y}) = \frac{[p(t_0, a; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{y})]_{u_0}^{v_0}}{\sum_{x \in \{o,s,d\}} [p(t_0, x; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{y})]_{u_0}^{v_0}} \tag{4.11}$$

When $y_{t_0} > 0$, the probability that $\eta_0 = o$ is 0 due to the indicator function $1_{y_t=0}$ for off states in the path likelihood.

This distribution can easily be normalized to create a Gibbs step, but a Metropolis-Hastings step is used in order to avoid proposing the same state as the current iteration. Suppose $\eta_{t_0 m} = a_0$. When $y_{t_0} > 0$, the proposal distribution places probability 1 on the remaining alternative state $\eta^* = a_1$, and the acceptance ratio r reduces to the likelihood ratio:

$$r = \frac{[p(t_0, \eta^*; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{y})]_{u_0}^{v_0}}{[p(t, \eta_{t_0 m}; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{y})]_{u_0}^{v_0}}$$

For $y_{t_0} = 0$, at iteration m a proposal $\eta^* = a_1$ is chosen from the two alternative states a_1 and a_2 to the current state η_{0m}, according to the re-normalized conditional probabilities in Equation 4.11, and accepted with probability:

$$r = \frac{[p(t_0, \eta^*; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{\phi}, \epsilon, \boldsymbol{y})]_{u_0}^{v_0} + [p(t_0, a_2; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{\phi}, \epsilon, \boldsymbol{y})]_{u_0}^{v_0}}{[p(t, \eta_{t_0 m}; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{\phi}, \epsilon, \boldsymbol{y})]_{u_0}^{v_0} + [p(t_0, a_2; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{\phi}, \epsilon, \boldsymbol{y})]_{u_0}^{v_0}}$$

If the last non-off state is proposed to change to an off state, and it is also the death date $t^\dagger$ for a non-immune machine, then the proposal step includes the effect of changing the death date to the next earliest observed non-off state.

Updating t^β, $t^\dagger$, and ϵ

With the path $\boldsymbol{\eta}$ considered fixed, let t_ℓ be the latest non-off state in $\boldsymbol{\eta}$. there are six possible states for the collection of discrete parameters $(t^\beta, t^\dagger, \epsilon)$. These are described in Table 4.5.3.

For switching a birthdate from 284 to 0, all newly entered states between 0 and 284 are assigned to an off state. For switching a death date from t_ℓ to 1224, all newly entered states between t_ℓ and 1224 are assigned

t^β	$t^\dagger$	ϵ	Description
0	1224	1	Immune, infected throughout the entire observation period.
0	1224	0	Not immune, infected throughout the entire observation period.
284	1224	1	Immune, infected from March 17th through the end of the observation period.
284	1224	0	Not immune, infected from March 17th through the end of the observation period
0	t_ℓ	0	Not immune, infected from March 3rd and cleaned on the last observed non-off state.
284	t_ℓ	0	Not immune, infected from March 17th and cleaned on the last observed non-off state.

Table 4.3: Candidate states for $(t^\beta, t^\dagger, \epsilon)$ when $\boldsymbol{\eta}$ is fixed.

to an off state. The complete conditional distribution $\pi((t^\beta, t^\dagger, \epsilon) \mid \boldsymbol{\eta}, \boldsymbol{\phi}, \zeta, \psi)$ has the following form:

$$\begin{aligned} &\pi((t^\beta, t^\dagger, \epsilon) \mid \boldsymbol{\eta}, \boldsymbol{\phi}, \zeta, \psi) \qquad (4.12) \\ &\quad \propto \quad \zeta^\epsilon \left[(1-\zeta)(1-\phi_{t^\dagger}) \textstyle\prod_{t \in \mathcal{S} \cup \mathcal{D}} \phi_t\right]^{1-\epsilon} \frac{e^{-2\lambda_o} \lambda_o^{c_{\min(t \in \mathcal{T}_{o|o})} + c_{\max(t \in \mathcal{T}_{o|o})}}}{c_{\min(t \in \mathcal{T}_{o|o})}! c_{\max(t \in \mathcal{T}_{o|o})}!} \textstyle\prod_{t \in \mathcal{O}} 1_{y_t = 0} \end{aligned}$$

This distribution is normalized by summing across the six possibilities listed in Table 4.5.3, and is updated using a Gibbs step.

4.5.4 Population hyperparameters

The single-host model generalizes to populations of independent machines where the number of machines H is known and the counts y_{it} are directly observable, for example within a network where a one-to-one relationship exists between machines and IP addresses. To generalize the estimation for the population, the updating strategy is:

- Repeat steps (1)-(4) above for machines $i = 1, \ldots, H$, with appropriate repetitions of steps to ensure mixing.

- Update population parameters ϕ and ζ, conditional on the current states and parameters of all machines in the population, using Gibbs sampling.

Using a Beta(a_{ϕ_t}, b_{ϕ_t}) prior, the complete conditional distribution of ϕ_t has the following form:

$$\pi(\phi_t \mid \boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_H) \quad \propto \quad (\phi_t)^{a_{\phi_t}-1}(1-\phi_t)^{b_{\phi_t}-1} \prod_{\{i:\eta_{it}\in\mathcal{S}\cup\mathcal{D}, t_i^\dagger \neq t\}} (\phi_t)^{1-\epsilon_i} \prod_{\{i:t_i^\dagger = t\}} (1-\phi_t)^{1-\epsilon_i}$$

The first product term is taken across machines that are not turned off or cleaned at time t, and a survival rate ϕ_t is multiplied to the likelihood for all non-immune machines. The second product is taken across machines that have a cleanup date of t, in which case $(1-\phi_t)$ is multiplied into the likelihood. For any $t < T$, $t_i^\dagger = t$ only if the machine is not immune. This distribution is a conjugate conditional distribution; it has the functional form of a Beta distribution with parameters

$$\begin{aligned} a^*_{\phi_t} &= a_{\phi_t} + \sum_{i=1}^{H} (1-\epsilon_i) 1_{\{\eta_{it}\in\mathcal{S}\cup\mathcal{D}\}} 1_{\{t_i^\dagger \neq t\}} \\ b^*_{\phi_t} &= b_{\phi_t} + \sum_{i=1}^{H} (1-\epsilon_i) 1_{\{t_i^\dagger = t\}} \end{aligned}$$

Updated values of ϕ_t can be sampled using a Gibbs step with the Beta$(a^*_{\phi_t}, b^*_{\phi_t})$ distribution.

Using a Beta(a_ζ, b_ζ) prior, the complete conditional distribution of the immune rate ζ has the following form:

$$\pi(\zeta \mid \boldsymbol{\eta}_1 \ldots \boldsymbol{\eta}_H) \quad \propto \quad (\zeta)^{a_\zeta - 1 + \sum_i \epsilon_i} (1-\zeta)^{b_\zeta - 1 + H - \sum_i \epsilon_i}$$

This is a conjugate conditional distribution; it has the functional form of a Beta distribution with parameters

$$\begin{aligned} a^*_\zeta &= a_\zeta + \sum_i \epsilon_i \\ b^*_\zeta &= b_\zeta + H - \sum_i \epsilon_i \end{aligned}$$

Updated values of ζ can be sampled using a Gibbs step with the Beta(a^*_ζ, b^*_ζ) distribution.

4.6 Analysis and Results

4.6.1 Finding "single host" data

The characteristics of individual machines are nuisance parameters for the overall population estimates when a one-to-one association does not exist between machines and IP addresses. But population estimates depend on an accurate model of active behavior based both on sandbox testing, and on how the activity varies among individuals "in the wild". Validating the single-host model is contingent on finding $/24$s in the data set that likely map consistently to a single machine. To do this, $/24$s from the Conficker-C data set were compared to a data set obtained from the Waledac botnet in 2009. Waledac is a malicious program that used infected hosts in the Conficker botnet to propagate fake anti-virus software.

Researchers infiltrated the Waledac Command and Control channel for an 18-day period in December of 2009, and they were able to tie unique identifiers of machines to IP addresses through which the machines checked in to a central Command and Control server. This allowed the researchers to explicitly model the IP-machine dynamic in the botnet.

Data on Waledac was collected from December 4th through 22nd, 2009. Of the $1,091,013$ $/24$s appearing in the Conficker-C data set, $180,347$ of them (16.5%) also appeared in the Waledac data set. Many of the links in the Waledac graph were "singleton pairs", that is, a machine ID and IP address pair that exclusively appeared together and nowhere else. A list of 870 "persistent singleton" IP-machine pairs was compiled from the set of singleton pairs in Waledac that overlapped the Conficker-C data set. These persistent singletons appeared consistently throughout at least 7 days of the 18-day window of data collection for Waledac. These persistent singletons comprised 837 $/24$ net blocks, with 804 isolated net blocks containing only a single IP address in the Waledac data.

This set of 804 isolated persistent singletons in Waledac was used as the starting place to search for $/24$ net blocks with high probability of a one-to-one ratio to machines in the Conficker-C data set. In the Conficker-C data set, the set was further reduced to 179 $/24$s for which a maximum of one unique IP address was recorded for Conficker-C data in the two-month window. This set included two obvious NATs with average hit rates of over 100 connections per hour, which were removed. A further 23 machines with fewer than 8 hours of activity across the 51-day window were also excluded, resulting in a set of 154 $/24$s

with evidence of single-machine activity in Waledac. Figure 4.7 shows a graph of the scan hits per hour originating from this set (Note that the blocks are not contiguous but are plotted in IP address order). Color intensity (black to bright yellow) is a log-scale measure of the number of connection attempts originating from each $/24$ in the picture, directed at destinations within the monitored network.

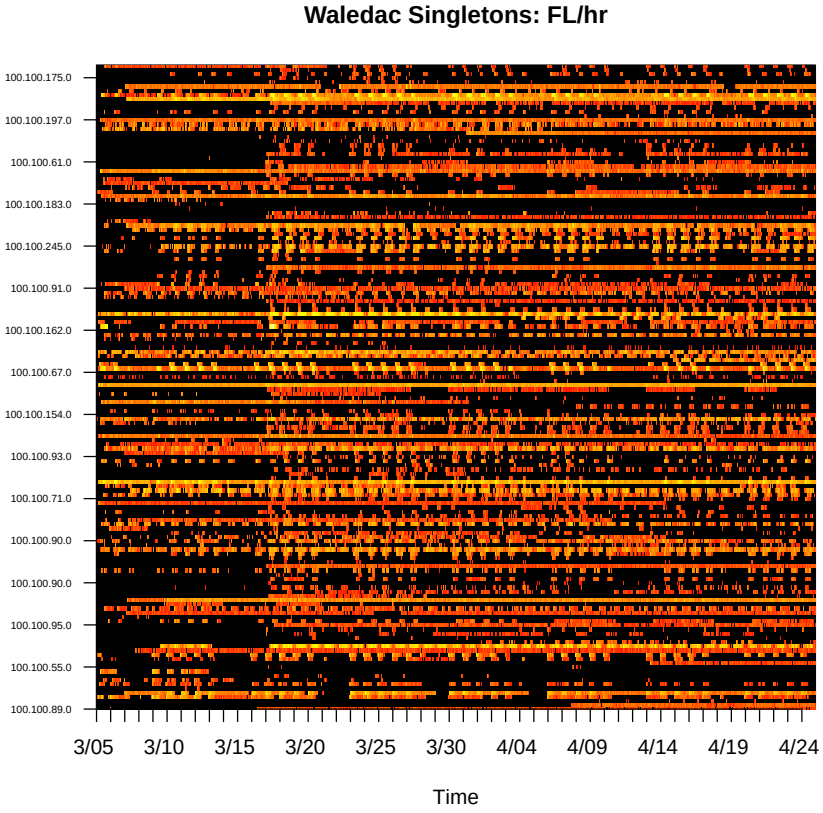

Figure 4.7: Conficker-C UDP scan hits per hour from a set of 154 $/24$ net blocks marked as persistent singletons (1 IP address to 1 machine) in an observational study of the Waledac botnet. Color intensity (black to bright yellow) is a log-scale measure of the number of connection attempts originating from each $/24$ in the picture, directed at a large monitored network.

For this analysis, the value of t^* estimated independently for each net block (see Section 3.4.2) was used as a fixed measurement of periodic alignment, as opposed to being updated within the MCMC structure.

4.6.2 Modeling decisions and starting values

Parameter		Description	Start S_1	Start S_2	Start S_3
θ	q	Baseline active scan rate	4.0	9.0	by machine
	ω	Multiplier of baseline rate during a spike	3.0	2.0	by machine
	α	Geometric decay rate per hour from spike to baseline	0.5	0.9	by machine
ψ	λ_o	Mean off-to-off state transition run	by machine	by machine	by machine
	ρ_o	Maximal periodic probability for off-state transitions	0.8	0.5	by machine
	ν_o	Periodic scaling parameter for off-state transitions	0.5	0.5	by machine
	γ_o	Aperiodic parameter for off-state transitions	unused	unused	unused
	ρ_s	Maximal periodic probability for spike-state transitions	0.8	0.5	by machine
	ν_s	Periodic scaling parameter for spike-state transitions	0.5	0.8	by machine
	γ_s	Aperiodic parameter for spike-state transitions	0.5	0.5	by machine
	ρ_d	Maximal periodic probability for decay-state transitions	0.8	0.5	by machine
	ν_d	Periodic scaling parameter for decay-state transitions	0.5	0.5	by machine
	γ_d	Aperiodic parameter for decay-state transitions	0.9	0.5	by machine
	ϵ	Immune ($= 1$) vs. non-immune ($= 0$) indicator	Stochastic	Stochastic	Stochastic
$\boldsymbol{\eta}$	η_t	Active state (off, spike, decay) at time t	by machine	by machine	by machine
	t^β	Initial infection date	0	0	0
	$t^\dagger$	Cleanup (death) date	1224	1224	1224
	ϕ	Survival probability per active hour for non-immune machines	Stochastic	Stochastic	Stochastic
	ζ	Population proportion of immune machines	Stochastic	Stochastic	Stochastic

Table 4.4: Three schemes for starting values for parameters in the single-host generation model. Stochastic or machine-dependent starting values are described further in the text.

Each IP address in the set of Waledac-informed singletons was considered as the complete record of a single machine for the duration of the study. The graph was initialized and fixed to match one machine per net block for the 154 observed /24s. For simplicity, the survival rate ϕ_t was set such that $\phi_t = \phi$ for all t.

Starting values for the non-state system parameters are shows in Table 4.6.2. Three different sets of starting values were used for the system parameters, in order to assess the effect of starting values on chain

convergence. The population parameters ζ^0 and ϕ^0 were initialized by a random draw from the prior. The initial immunity state for each machine was determined by a coin flip using the starting value ζ^0.

For starting values S_1 and S_2, the state vector η_t and off rate λ_o were initialized for each machine as follows:

1. Let $P_{95,1+}$ be the 95-th percentile of the counts at hours $\{t : y_t > 0\}$. Set $\eta_t^0 = s$ for the set $\{t : y_t \geq P_{95,1+}\}$.

2. if $y_0 = 0$, set $\eta_0^0 = o$, and assume that the machine was turned off for the three hours prior to the start of the observation window.

3. Set $\eta_t^0 = o$ for the set $\{t : y_t = 0 \text{ and } y_{t-1} = y_{t-2} = y_{t-3} = 0\}$.

4. Set $\eta_t^0 = d$ for all remaining hours.

5. Let C_0 be the number of strings of consecutive off states. Initialize λ_o as :

$$\lambda_o = \frac{|\{t : \eta_t^0 = o\}|}{C_0}.$$

For the third set of starting values, S_3, estimates were set on a machine-by-machine basis using a Poisson mixture analysis, with starting values of q and ω set empirically using the two groups. Given the set of active hours described by the starting scheme for S_1 and S_2, a further refinement to the starting values of both spike vs. decay states, as well as q, ω and the transition probability parameters was set as follows.

1. Use the spike vs. decay states from the 95-th percentile analysis from S_1 and S_2, and the respective sample averages for each group, as starting points in a 2-parameter Poisson mixture model,

$$p(y_t) = \pi_0 f(y_t; \lambda_0) + \pi_1 f(y_t; \lambda_1),$$

where $f(y; \lambda)$ is the Poisson density of x with mean λ, Z_t is the group assignment (0 or 1) of y_t, and $\lambda_0 < \lambda_1$. Refine these estimates using E-M until convergence.

2. Compare the BIC of the 2-parameter model to the BIC of a single parameter model, assuming all

active values broadcast at a shared baseline rate q. Select starting values according to the model chosen.

3. If the 2-parameter model is chosen:

   - Set $q = \lambda_0$, $\omega = \lambda_1/\lambda_0$, and $\alpha = 0.65$.
   - Set $\eta_t = s$ if $Pr(Z_t = 1|y_t) \geq 0.75$ (this tries to account for decay values that may still seem like higher spikes).
   - Keep the starting values for ψ as in the starting values from S_2.

4. If the 1-parameter model with mean λ is chosen:

   - Set $q = \lambda$, $\omega = 6.0$, and $\alpha = 0.25$.
   - Set $\gamma_s = 0.001$ and $\gamma_d = |\{t : \eta_t \neq o\}|/1224$.
   - Flatten the starting values for the periodicity by setting $\nu_o = \nu_d = 15.0$.
   - Keep other starting values for ψ as described in scheme S_2.

Figure 4.8 shows a plot of the EM-based initial starting values (S_3) for q and ω for the 154 IP addresses in the sample set. Red indicates machines for which the 1-parameter model was chosen over the 2-parameter model (13 machines). Despite the existence of a few outliers with baseline values > 20, most of the initial parameters show consistency with the result of the sandbox testing and the prior distributions for single hosts.

4.6.3 MCMC estimation details

MCMC chains were run for $30,000$ iterations using the fixed starting states from S_1, and for $30,000$ iterations using the starting values based on EM refinement. Each iteration consisted of the following steps:

1. For each of the 154 machines:

   - Perform one updating step (Metropolis-Hastings or Gibbs) in sequence for each system parameter in $(\theta, \psi_{|\{t^\beta, t^\dagger, \epsilon\}})$;

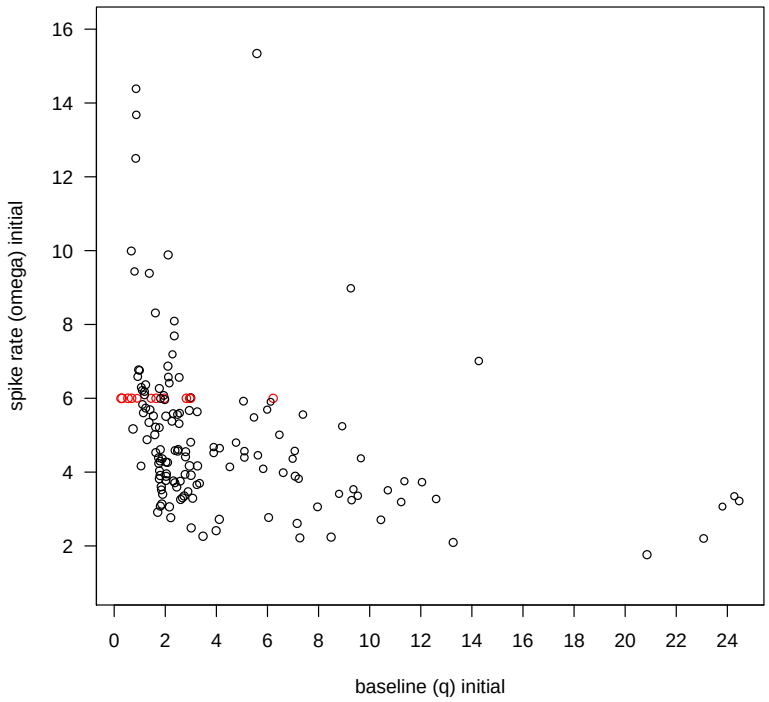

Figure 4.8: Starting states for q and ω informed by fitting a time-independent Poisson mixture model to the active states determined using the S_3 starting value algorithm.

- Perform updating steps in sequence for 100 states chosen randomly from between the current birth and death dates of the machine;
- Perform a joint updating step to sample a valid configuration of birth date, death date, and immunity status for the machine.

2. Update the survival rate ϕ for the population

3. Update the immunity rate ζ for the population.

All chains were thinned by a value of 5 for ease of storage: at the end of every fifth full iteration of steps (1) through (3), the values of all states and all parameters for each machine, as well as the population parameters, were written to disk. The first 500 thinned iterations were removed from each parameter chain as burn-in. The remaining data points were used to calculate posterior means. The state iterations were used to build an empirical marginal distribution of states for each hour t in the sample, in order to analyze the

estimation results. For continuous parameters, Geweke's test of equality of means for the beginning and end of the chain was used as a rough assessment of convergence within each chain. Summaries are shown for the S_1 starting states, with discussions of comparisons to the EM-based S_3 starting states.

The marginal distribution of states at each time t does not preserve the order of the states visited during iterations of the chain. In order to assess mixing, the distribution of runs of consecutive states in the chain was summarized for each time t for each machine. Let M be the total number of iterations of the chain after removing burn-in. For the discrete-valued state at time t, the number of runs of consecutive states, R_t is recorded to obtain an understanding of how the marginal proportions are distributed across the length of the chain. This measure is coupled with the Kullback-Liebler divergence of the marginal distribution with the distribution of states the first 50% of the chain and the last 25% of the chain, to measure stationarity. A well-mixing chain should have a relatively large number of runs R_t with respect to the marginal proportions, and should show only small differences in the K-L divergence from the beginning vs. the end of the chain.

4.6.4 Population distribution of parameters

The distribution of the individual differences in θ and ψ can help to inform the population distribution hyperparameters for analysis of networks where H is unknown. Figure 4.9 shows the posterior mean values for q and ω in the population. As compared to the starting values, the baseline rates are generally lower and more clustered between 0 and 10. A surprising number of machines (44) showed evidence of q vs. ω confusion. These machines show up as a cluster at the top right corner of the graph. They had very low q values (< 1), and ω values near the hard coded upper limit of 24. Another 5 machines show baseline rates $q > 10$, which is possibly an indication of more than one host residing behind the IP address.

Figure 4.10 shows density plots of the posterior distribution of the common survival rate ϕ and the immune rate ζ from the population. The modeling of a single survival rate does not appear to be flexible enough to capture the difference between immune and non-immune machines. The posterior means of the death dates of all but 3 machines were equal to 1224, and the posterior means of all birth dates were set at 0, indicating that the timelines were rarely set to shorter intervals than the entire range. A more flexible model for survival rates with time can help alleviate this problem.

Acceptance rates for the state proposals (aggregated across all 1224 states) were universally low for all

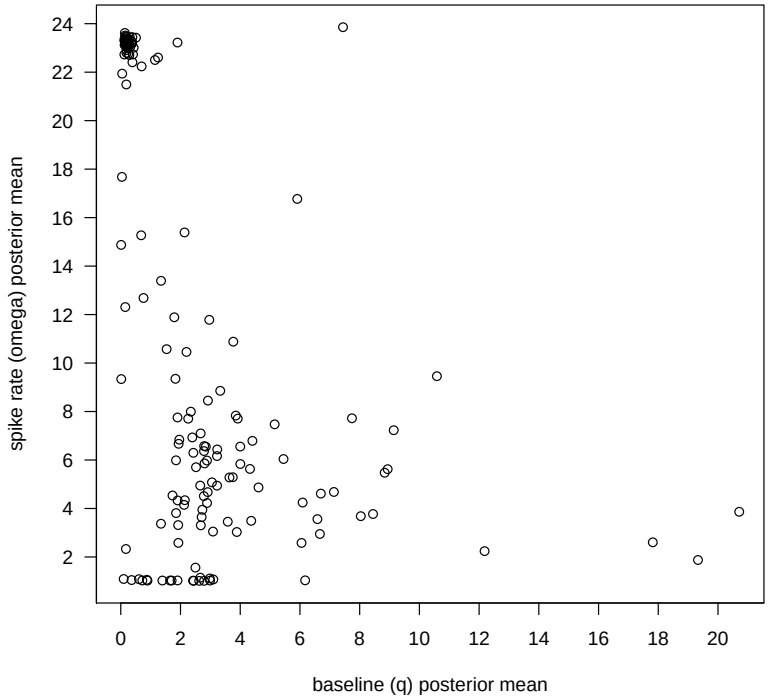

Figure 4.9: Posterior means of q and ω in the population. Baseline rates are more contained within the range of 1 to 10 than the rates that were used as starting values. But there are a large number of machines (44) in the upper left-hand corner that exhibit some confusion between spike and decay states, yielding low baseline rates coupled with high spike rates to model generally sparse observations.

machines ($\sim 2\%$ of all proposals were accepted in many cases). In some cases this was due to obvious spikes or to long strings of low counts clearly indicating decay states. Many states with 0 counts were not changed throughout the model iterations. The median number of runs per chain for chains with an observed count of 0 was 35 runs per chain. For each discrete observation size $y_0 > 0$, the median number of runs per hour across all hours with $y_t = y_0$ and all machines was generally less than 5 runs per chain, but the distribution and average values (see Figure 4.11) show decent mixing of switches between states for middle-sized counts. Very small and very large counts show less mixing between states, but these counts are also *a priori* well-informed as either spikes or small baselines. On the other hand, changing the proposal method for updating states from one hour at a time to updating runs of states at once, may aid mixing in some chains. Profiles of states over time are further examined in Section 4.6.5.

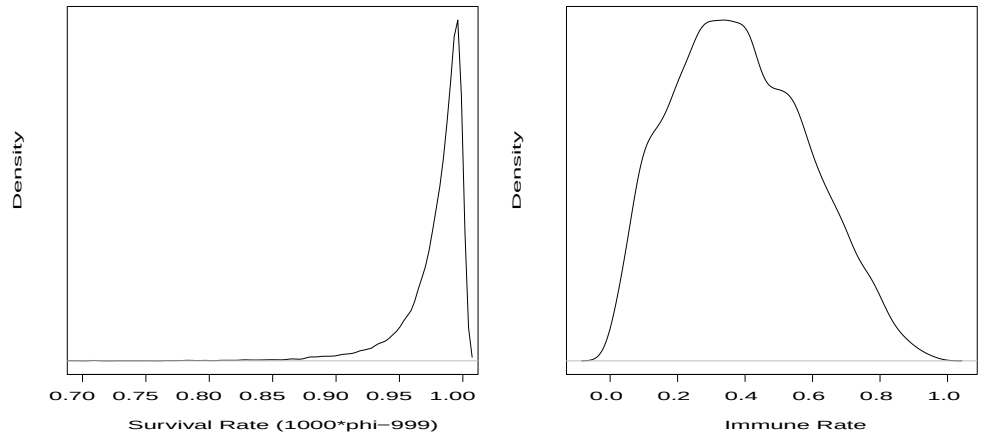

Figure 4.10: Posterior density plots of the common survival rate ϕ and the immune rate ζ in the population. The survival rate is very high, and thus the difference between immune and non-immune is difficult to detect. The result is uncertainty about the immune rate. The posterior mean of the death date $t^{\dagger}$ was near 1224 for all machines, indicating that very few timelines were listed as cleaned at the last non-off state. Adapting the survival rate to depend on t and including λ_o in a joint update may alleviate this problem.

Many machines showed evidence of non-convergence from the Geweke test. The model is an attempt at a flexible structure to fit varying profiles of activity, with reasonable defaults for proposal values, but it does not appear to have provided a good fit for all machines in the Waledac data set. This could be due to differing behavior profiles in some machines, or to the inclusion of NAT and DHCP networks in the vetted data set. Figure 4.12 shows a plot of the number of non-zero counts in the machine vs. the number of continuous parameters flagged as showing convergence issues based on the Geweke test of means for the first 50% of the chain after burn-in vs. the last 10% of the chain. The machines that show more problems tend to be the ones that have fewer observed non-zero counts. This could be indicative of a difficulty with the transition probabilities and the underlying states.

A total of 105 machines with greater than 48 hours of non-zero activity had 4 or fewer chains flagged as non-convergent by the Geweke test. Out of these 94 machines, 19 displayed evidence of very low baseline rates coupled with large spike rates, and were flagged as possible examples of model mis-fit or non-identifiability of parameters. Machines with baseline rates less than 0.25 or spike rates greater than 20 were set aside, leaving a total of 82 out of the 154 machines in the data set with persistent activity for more than 48 hours, few flagged convergence issues, and no obvious identifiability problems from the S_2 starting state.

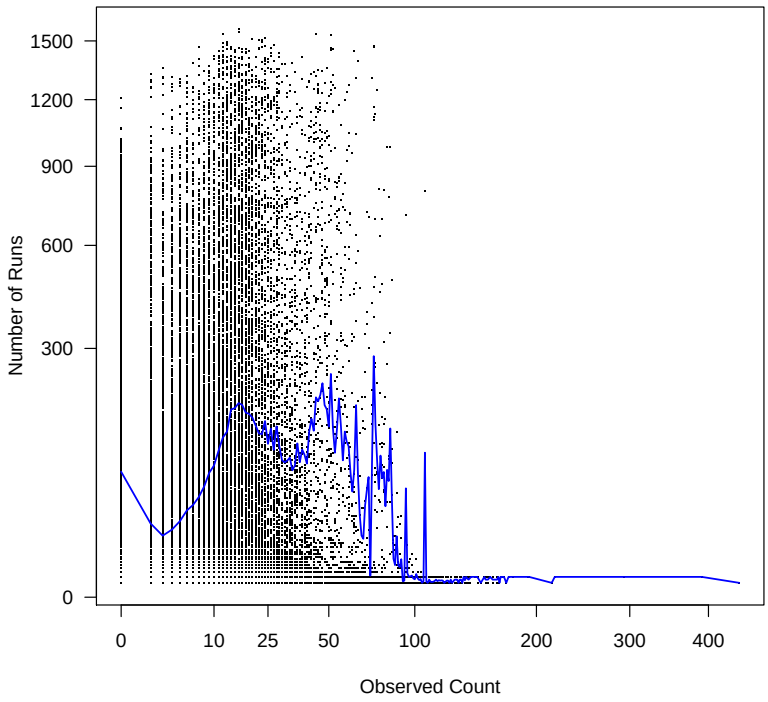

Figure 4.11: Plot of the number of runs R_t (switches from one state to another) vs. the observed count $y_t$for all hours for all 154 machines. Axes are on the $x^{1/2}$ and $y^{1/2}$ scale. The blue line shows the average of R_t for each observed count value $y = y_0$. Very small or very large counts on average show fewer fluctuations in the chain than middle values $(10 - 100)$, which can switch more often between spike states and early decay states.

A total of 10 of these machines displayed some noticeable changes in posterior means between the S_1 and S_3 starting states, mostly consistent with the difficulty in estimating the baseline rate q in the presence of a large decay rate α, when the machine is turned off for non-workday periods (Machine 50, discussed in the next section, shows an example of this phenomenon). The KL-divergence of state distributions for these machines (beginning vs. total, and end vs. total) showed some signs of poorer mixing for the beginnings of the chains, but in general were well behaved for the end of the chain vs. the total.

Figure 4.14 shows plots of the posterior means for S_2 (x axis) vs. S_3 (y axis) for these machines for the 12 continuous parameters in the model. The center line is $y = x$, and the blue dashed lines represent the boundary of points for the other 72 machines in the set. Each of the machines displaying outliers is marked by its ID label. Blue lines represent ± 2.5 standard deviations beyond the typical variability in the population

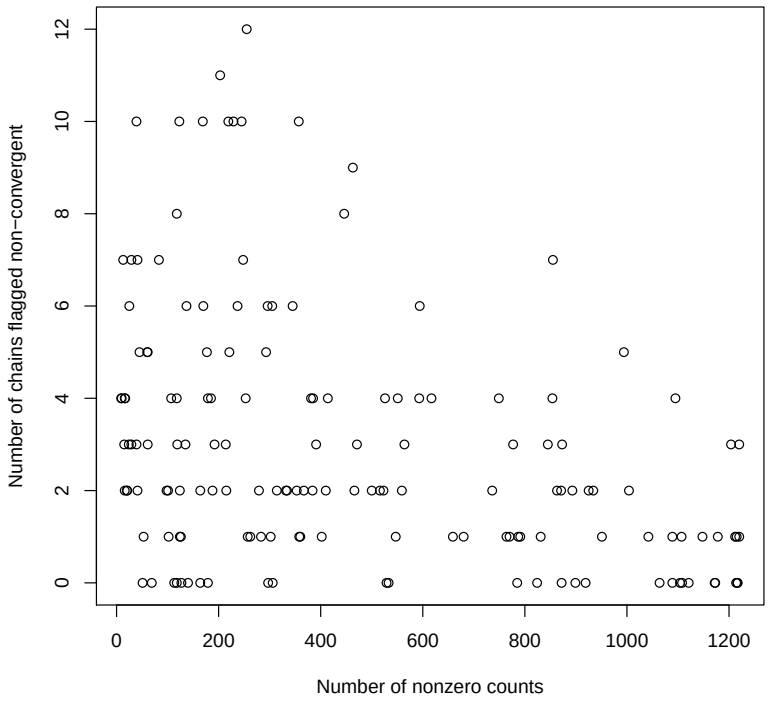

Figure 4.12: Plot of the number of non-zero counts vs. the number of chains flagged with convergence difficulties using a simple Geweke test of means of the first 50% of the chain post burn-in vs. the last 10% of the chain. 104 machines had 3 or fewer parameter chains flagged with convergence problems at a significance level of 0.01. The chains that show large-scale problems tend to have fewer observed non-zero counts.

along the $y = x$ line.

Table 4.6.4 summarizes the distribution of posterior means of continuous parameters for these machines, and Figure 4.13 shows marginal density plots. Visual diagnostics were used to examine individual machines further, in order to assess the results of the Geweke test and to examine behavioral profiles of the machines that exhibited problems.

Parameter		Min.	Q1	Median	Mean	Q3	Max.	Avg. σ
θ	q	0.36	2.45	2.97	4.39	4.56	20.71	0.11
	ω	1.02	3.39	5.01	5.22	6.76	16.77	0.83
	α	0.22	0.40	0.53	0.54	0.64	0.92	0.04
ψ	λ_o	6.01	8.89	17.02	29.18	31.80	151.32	2.06
	ρ_o	0.05	0.32	0.47	0.50	0.65	0.91	0.12
	ν_o	0.71	1.88	2.58	3.01	3.92	7.49	2.17
	ρ_s	0.04	0.87	0.96	0.84	0.98	0.99	0.07
	ν_s	0.33	2.38	6.23	5.97	8.88	14.15	1.62
	γ_s	0.01	0.01	0.04	0.13	0.25	0.52	0.03
	ρ_d	0.05	0.47	0.69	0.66	0.92	0.99	0.10
	ν_d	0.13	2.22	4.70	6.40	10.58	20.93	2.27
	γ_d	0.62	0.88	0.95	0.92	0.98	0.99	0.02

Table 4.5: Summaries of population posterior means for 82 postulated machines in the Waledac informed data set. Summaries are restricted to the set of machines with at least 48 hours of non-zero activity (median 767), with few indications of non-convergence or model non-identifiability. The last column is the average standard deviation of the posterior distribution of the parameter among the 82 machines.

4.6.5 Assessing individual machines

Output analysis graphs for each machine were created using R, for visual model checking. A dashboard visualization was created using a time series plot of the observed counts for each machine, along with four types of diagnostic plots using the thinned MCMC iterations that were not discarded as burn-in:

1. A posterior predictive plot summarizing simulated data for each time point;

2. A barplot summary of the state information for each time point;

3. Transition probability profiles of the parameterized state transitions;

4. MCMC sparkline plots for each of the continuous parameters in the model.

Posterior predictive plot: The posterior predictive plot is a goodness of fit diagnostic for a chain that compares the observed data y to an empirical predictive distribution $y_1^* \dots y_M^*$ generated from the model. At each iteration m of the MCMC, the parameter values θ_m are used to simulate y_m^*. Assuming that burn-in iterations have been discarded, a posterior predictive p-value $\tilde{p}_y$ for the goodness of fit of the observation to

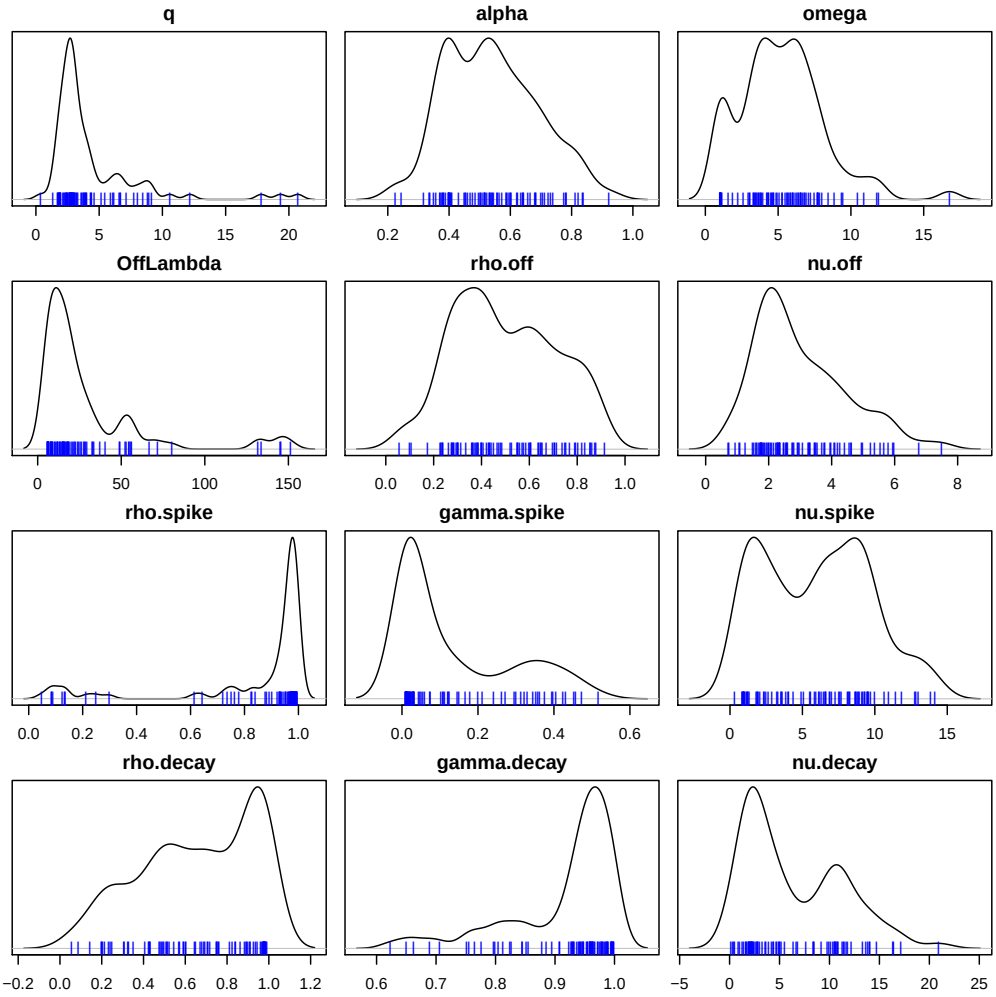

Figure 4.13: Density plots of parameter posterior means highlighting marginal individual differences in a population of 82 IP addresses believed to represent individual machines.

the model is given by

$$\tilde{p}_y \quad = \quad \frac{1}{m} \sum_{m=1}^{M} 1_{\{y \leq y_m^*\}}$$

Generally, observed values that fall within likely empirical percentiles of the predictive distribution are an indication of good fit, while values falling far outside the empirical percentiles are an indication of a poor fit.

Full predictive simulation for the Waledac data set would require saving the joint information about both the 1224-dimensional state vector $\boldsymbol{\eta}_m$ and the generation parameters θ_m at each iteration. The posterior

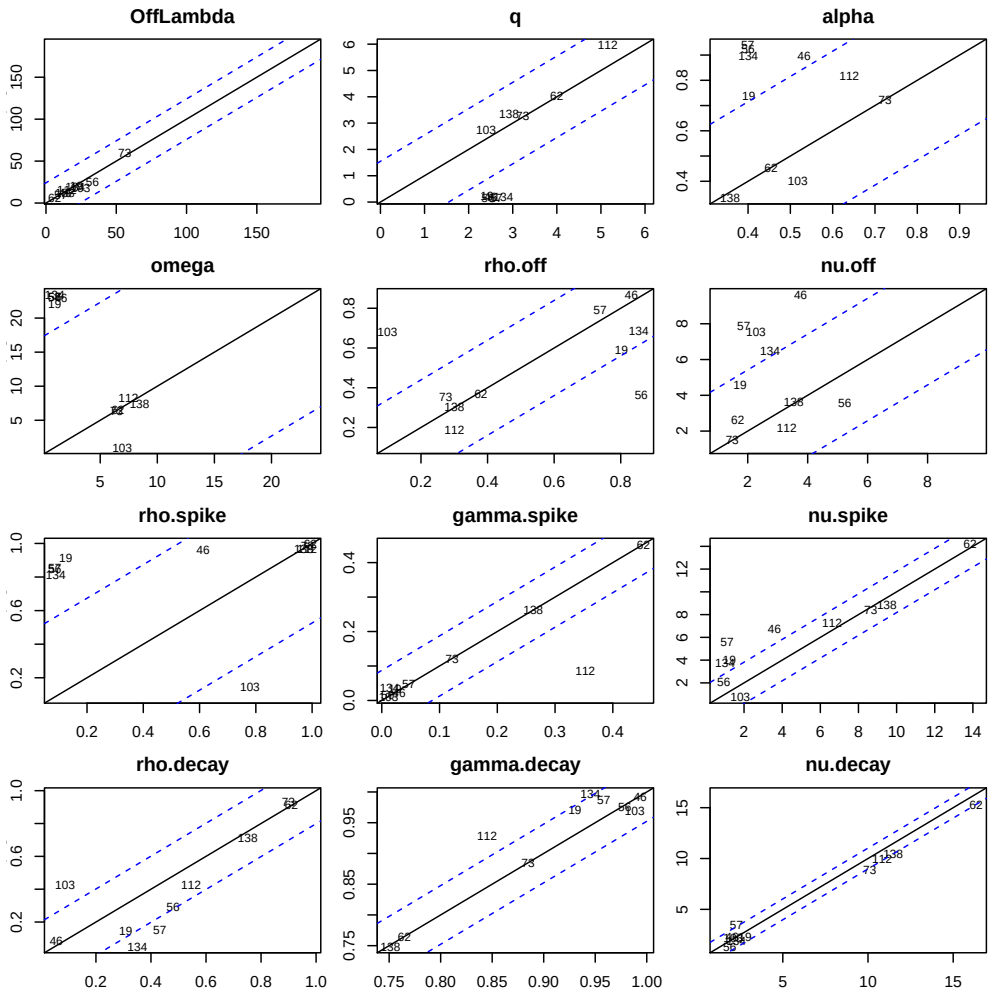

Figure 4.14: Comparison of posterior means for two different sets of starting values: S_1 (x axis) and S_3 (y axis), for 82 IP addresses believed to represent individual machines. Blue lines represent ± 2.5 standard deviations beyond the typical variability in the population along the $y = x$ line. ID values for machines showing deviations of > 2.5 in at least one parameter (10 machines) are marked on the graph.

predictive plot was adapted slightly because the individual state information was not saved at this granularity for the output analysis. Two methods for generating an approximate posterior predictive distribution y^*_{mt} using θ_m in conjunction with a set of states $\boldsymbol{\eta}^*_m$ were considered:

1. Parametric generation: At each iteration m, use the transition parameters ψ_m to generate states $\boldsymbol{\eta}^*_m$;

2. Empirical generation: At each iteration m, use the marginal empirical distribution of the state values to generate a set of states $\boldsymbol{\eta}^*_m$ independently for each hour. For each hour t, let π_{ot}, π_{st} and π_{dt} be the proportion of MCMC iterations spent in off, spike or decay states, respectively. A state η^*_{mt} is chosen

according to these discrete probabilities.

The second method ignores the correlation between states, but is easier to compute than the first method. The resulting intervals are generally more conservative than the intervals obtained using method 1; for example, parametric state generation can result in simulation of a spike state even when $\pi_{st} = 0$ in the empirical distribution. Method 2 can be used to check the suitability of the Poisson distribution, as opposed to checking the goodness of fit of the parametric transition model, which can be assessed in a separate analysis.

An example of a posterior predictive plot using empirical state generation for a machine in the Waledac data set (Machine 34) is shown in Figure 4.15. The observed counts are shown as points, while the gray bars indicate the posterior predictive interval given by the percentiles labeled on the graph. A blue line shows the mean of the simulated counts $y_1^* \dots y_m^*$ for each hour. Points colored in red indicate an observation falling outside of the predictive interval.

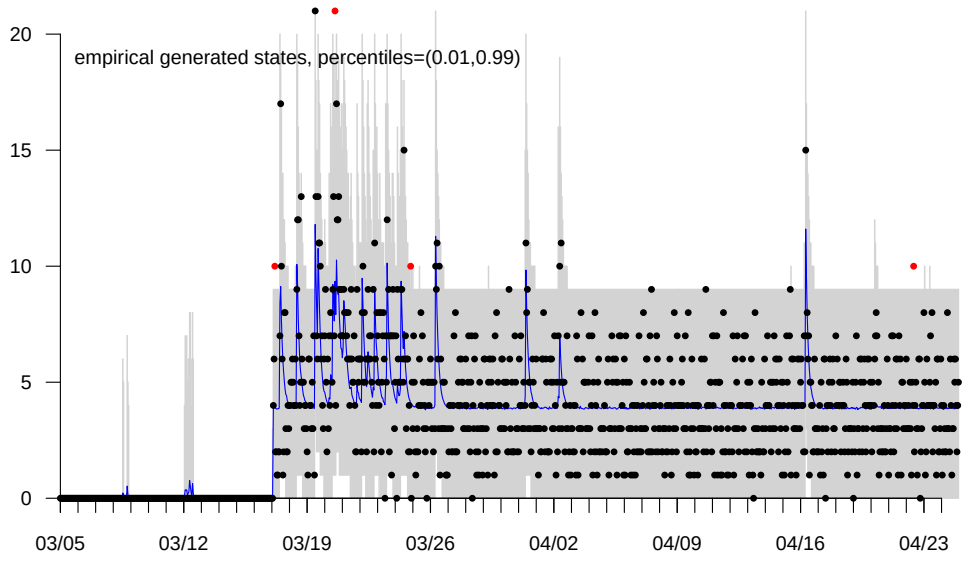

Figure 4.15: A posterior predictive plot for Machine 34 in the Waledac data set. Gray shading indicates the interval between percentiles 0.01 and 0.99 in the simulated counts. Observations are shown as points, with red indicating an observation falling outside of the predictive interval. The blue line indicates the mean of the simulated observations at each hour.

Barplot summary: The barplot summary graph is a visual display of the state information for each hour. Each hour t admits a discrete distribution π_{ot}, π_{st} and π_{dt} (such that $\sum_a \pi_{at} = 1$), representing the propor-

tion of times that hour t was set to each state in the chain. The barplot summary displays a color-coded, stacked plot of this discrete distribution for each hour. An example, again for Machine 34, is shown in Figure 4.16. Red, blue, and gray colors indicate the proportion of spike, decay, and off states seen for each hour, stacked in that order from 0 to 1. Ordering the distributions this way presents an intuitive picture of the activity profile for each hour. Hours with zero observed activity appear as a grey background on the graph, while hours with non-zero counts appear blue, and indicate the propensity of spike activity in the hour with the red line.

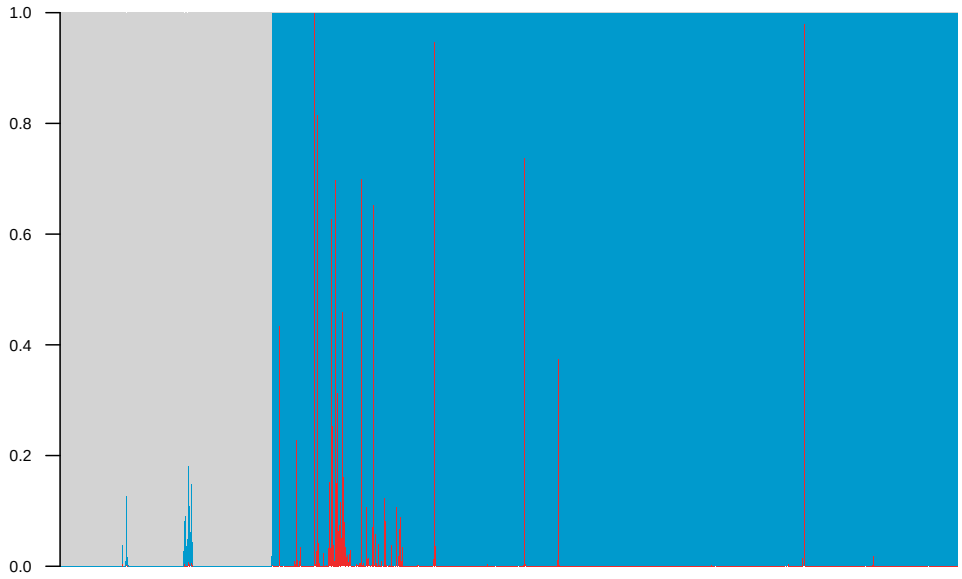

Figure 4.16: A barplot summary of the empirical state distribution for Machine 34 in the Waledac data set. Distributions of states for each hour are displayed as vertical stacks of probabilities that sum to 1. Red, blue, and gray colors indicate the proportion of spike, decay, and off states seen for each hour, stacked in that order from 0 to 1.

Transition probability profile: The transition probability profile plot is a version of the periodic curves shown in Figure 4.1 in Section 4.1.2. The probability profile is a graphical display of the 24-hour profile for state-to-state transition probabilities $\pi_{a|\cdot}(t)$ from state a to off, spike or decay states (or just to spike or decay states, if $a = o$). The profile is computed using the posterior mean estimates for ν_a, ρ_a, and γ_a. Red upward-facing triangles represent transitions to spike states, blue downward-facing triangles to decay states, and gray circles to off states. Transitions are connected hour-to-hour by dashed lines to highlight the effect of the periodic component $p(\rho_a, \nu_a, t^*)$. Vertically per hour, each set of three probabilities sums to 1. When

state by state information is available, the empirical distribution of transitions, calculated from the MCMC output, can also be plotted to gauge goodness of fit of the periodic transition model to the constraints of the observed data.

Figure 4.17 shows a transition probability profile for Machine 34 as it transitions away from a spike state. Empirical transition probabilities are shown as color-coded X points on the graph. These indicate that the spike state favors transition to a decay state more often (and transition to off states less often) than is suggested by the parametric profile evaluated at the posterior mean.

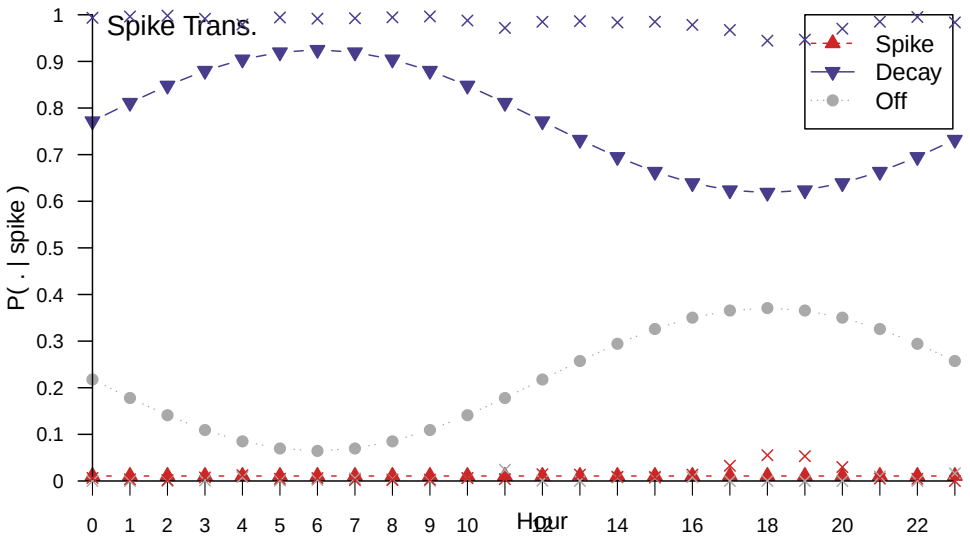

Figure 4.17: A transition probability profile for transitions from a spike state to either off (gray), decay (blue), or spike (red). The parametric curve, generated using the posterior mean of the transition probabilities ψ, is shown in solid colors. Empirical transition probabilities calculated from the simulated state values in the MCMC are shown as color-coded X points on the graph. These indicate that the spike state favors transition to a decay state more often (and transition to off states less often) than is suggested by the parametric profile evaluated at the posterior mean.

MCMC sparkline: The MCMC sparkline plot provides a visual summary of the output and iterations for any continuous parameter in the chain. The design attempts to remove clutter from the graphs so that they may be stacked for comparison and analyzed on a relatively small visual scale. Figure 4.18 shows two examples; the top plot from Machine 34 is a stable parameter q that appears to have converged, while the bottom plot from Machine 111 is a parameter q that shows signs of instability. The sparkline has two components: a time series plot of the iterations on the left, and a density plot of the posterior distribution

on the right, both using the same scale. To show scale, the minimum and maximum values of the observed iterations are marked as red lines bracketing the density plot. The posterior mean value is labeled on the density plot and represented as a dotted blue line that marks both the iteration plot and the density plot. Iterations of the chain are shown as a gray background, with a black line indicating trends over time using a lowess smoother. Acceptance ratios are also labeled (in the case of the baseline rate q, the acceptance ratio is equal to 1 because it is a Gibbs step).

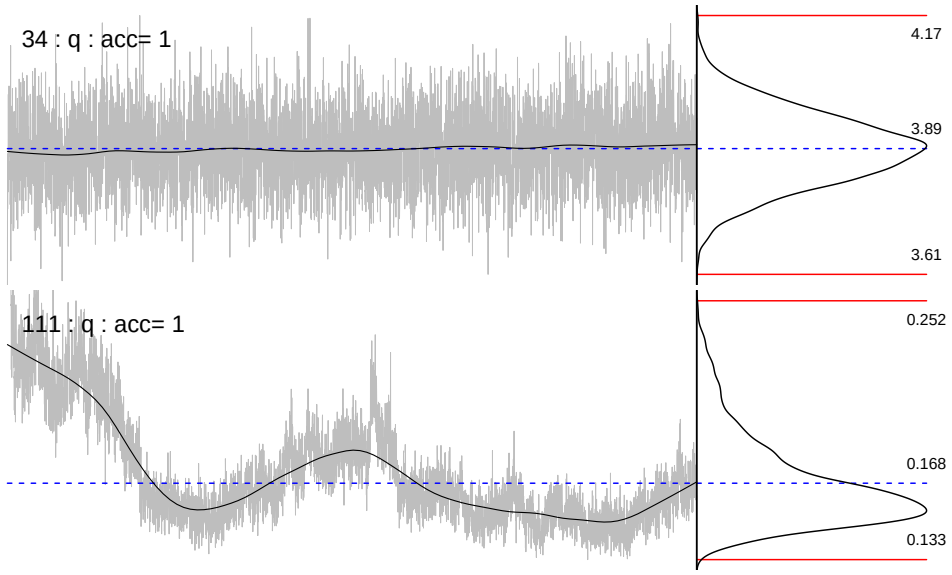

Figure 4.18: MCMC sparklines for the baseline rate parameter q from two different machines, shown at a large scale to highlight the features of the graph. Machine 34 (top) shows a well behaved chain with little pattern to the iterations over time, and a symmetric, unimodal posterior distribution with mean equal to 3.89. A slight upward trend of the estimate is seen in the smoothed iterations. Machine 111 shows obvious signs of instability and non-convergence.

Dashboard visualization: The dashboard visualization is a summary of the MCMC output for each machine, using the components described above in conjunction with a time series plot of the observed data. Figure 4.19 shows the layout of the full visualization for Machine 34, on which the model appears to be stable and generally well-behaved. There is an obvious change to the off state mean λ_o, resulting in a bimodal posterior distribution that nonetheless does not appear to have much of an effect on the rest of the parameters. From the barplot summary, it is likely that the model started out with all "off" states prior to March 17th, then suggested several active decay states to break up the run of consecutive off states. All activity for

Machine 34 begins after the late birth date of March 17th, but the model does not appear to suggest the later birth date, a sign of poor mixing for the birth, death and immunity updating step in the chain.

Machine 34 also appears to show a slight uptick in its baseline rate activity during the March 17th surge, a pattern that was seen in several machines in the data set. This suggests that part of the observed surge in the overall population numbers could be attributable to a change in the malware instruction set, that resulted in a short period of increased activity for already infected machines, along with the rise of new infections.

The visualization was customized to reasonable default values and automated in order to create dashboards for all 154 machines in the Waledac data set. Analysis of machines revealed several different kinds of patterns, a selection of which are explored in the graphs below as representatives of some typical behavior.

Figure 4.20 shows an example of very stable baseline rate behavior, with only occasional spikes. Like Machine 34, Machine 23 appears to be a machine which is rarely turned off, does not appear to follow workday usage patterns, and is rarely used in a way that opens new network connections (thus causing a UDP rate spike). The MCMC iterations show no signs of non-convergence, and the posterior distribution of the baseline rate q appears to agree with the prior information from the sandbox testing. The state values are very stable, not due to lack of mixing, but to a good fit to the data. The decay-to-decay transition rate γ_d is estimated near 1.

Machine 24, shown in Figure 4.21, again seems to be a stable baseline rate q as well as well behaved MCMC iterations, but this machine is much more likely to spike in its daily use. The spike rate of 6.5 agrees with the spike rate of approximately 5 to even 10 times the baseline rate that was observed in sandbox testing. Despite the large spikes, Machine 24 consistently reverts back to its baseline, with a steep geometric decay rate multiplier of 0.43. The model captures this periodic spiking behavior relatively well and does not show any obvious signs of mis-fit. A total of 65 out of the 154 machines had baseline rates q between 1 and 6, with spike rates ω less than 10.

Despite being included in the set of possible singleton machines as cross-referenced by the Waledac data, "Machine" 54, shown in Figure 4.22, displays some evidence of multiple infected hosts communicating through it. The estimated baseline rate is 20.7, approximately 4 times higher than was estimated from sandbox tests. Obvious daily patterns follow workday cycles and could be the result of multiple users initiating network connections during business hours. There also appear to be several categorical shifts in

the baseline, possibly a result of one or more of the underlying machines being turned off. Parameters show no signs of non-convergence, except for some fluctuations in the off-state transition probability parameters. This is not surprising, since there are no hours with 0 counts in the machine history. The decay rate for this machine is also relatively slow (0.68) as compared with machine 24, which could be explained by more variability among the times when spikes are initiated among the underlying machines.

Machine 111, shown in Figure 4.23, is an example of a possible model mis-fit. The model suggests that nearly every observed non-zero count is a spike, but that the machine is mostly turned on with a baseline rate too low to admit many observations. This interpretation results in a baseline rate that sinks toward 0 in the model, and a spike rate that raises proportionately such that $q\omega$ is approximately equal to 4, matching the mean of the non-zero counts. The MCMC displays show obvious signs of non-convergence, and the spike rate raises to the ceiling of the limit of 24 that was built into the model.

A machine with a very low baseline rate and a high spike rate is not impossible; the low rate does contradict the prior information found in sandbox tests, but network properties or rate-limiting on the allocation side may cause the transmission rate of UDP packets to drop. However, it is more likely that the current transition model provides a bad fit to these types of machines, or that the higher variability without an observable baseline is the result of a profile of user activity that is not captured well by the spike-decay model.

Machine 50, shown in Figure 4.24, is another example of a machine with a very low baseline rate and high spike rate. This machine is typical of the 12 machines observed with persistent activity, low baseline rates ($q < 0.25$) and high spike rates ($\omega > 23$), that nonetheless showed few convergence issues according to the Geweke test. These machines show sporadic activity following workday patterns. It appears that the activity does not last long enough to "settle down" to a visible baseline, before machines are turned off. Partly, the low baseline rate may be affected by weekday vs. weekend patterns, which were not taken into account in the transition model. But the sinking of the rate q does not appear to be due to confusion in the state space, as with Machine 111. It is difficult to tell from the time series plot in the dashboard whether Machine 50 spikes in its patterns, but the pattern is clearly evident in an hourly quantile plot (Figure 4.25), such as was used in Section 3.4.2 to find the periodic offsets for each network.

It appears as though a stricter lower bound for the baseline rate q–for example a lower bound of $q = 1.0$–

would have little effect on the model's overall fit to these machines besides lowering the spike rate. This bound can aid interpretability of the stable baseline rate that would be observed if machines were left on for longer periods of time. A stricter prior on the decay rate α may also help to disambiguate these situations; the rate α for Machine 50 is estimated as 0.87, which is high relative to the prior $\pi(\alpha)$.

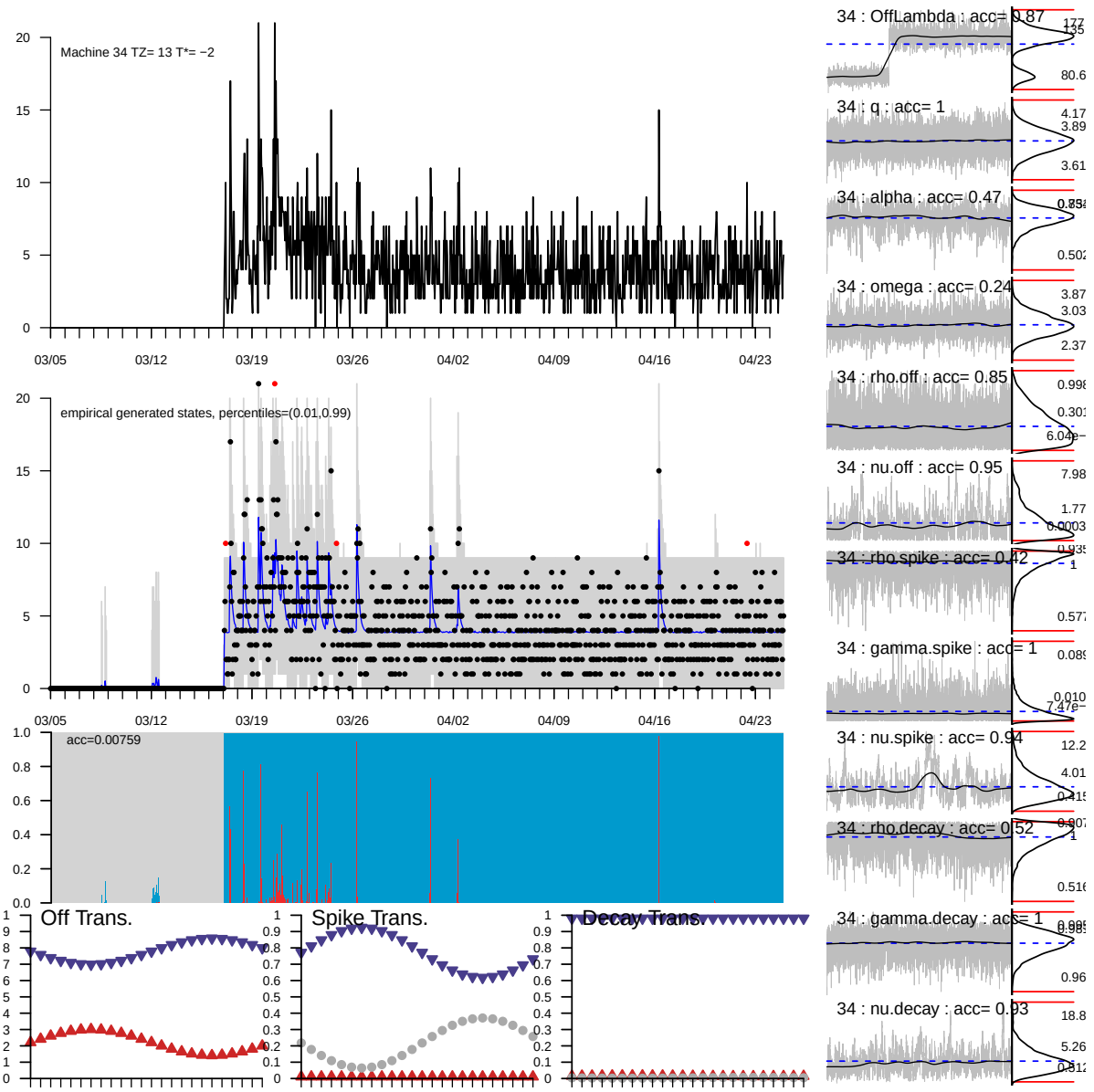

Figure 4.19: A complete dashboard visualization for Machine 34. A bimodal distribution in the mean time spent in off states can be explained by a categorical change in states prior to March 17th. The fact that the model does not iterate through states where the birthdate is March 17th as opposed to March 5th is an indication of either model mis-fit or poor mixing in the chain.

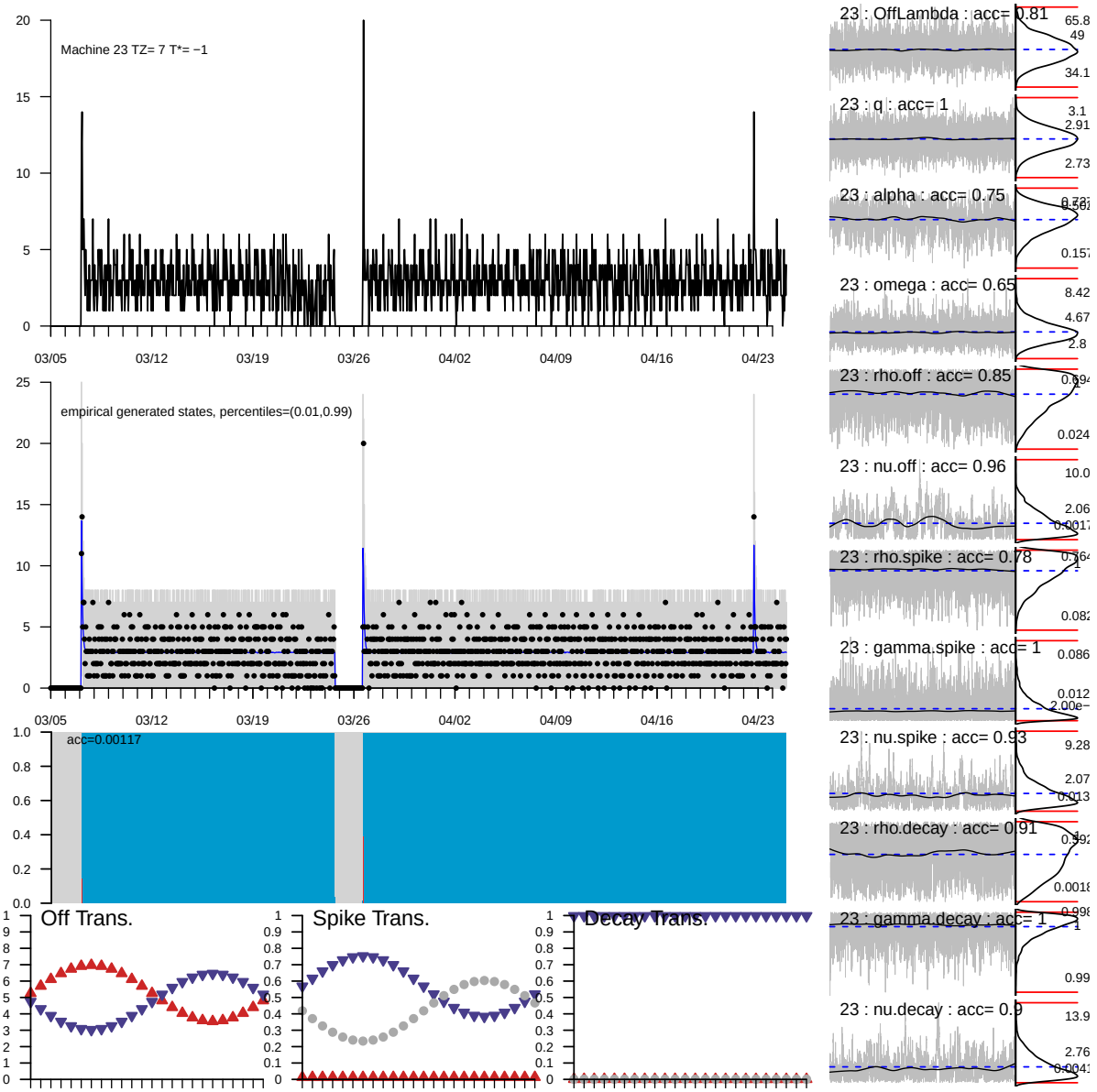

Figure 4.20: A steadily scanning machine with very few spikes. The MCMC iterations show no obvious signs of non-convergence. The decay-to-decay transition rate γ_d is estimated near 1.

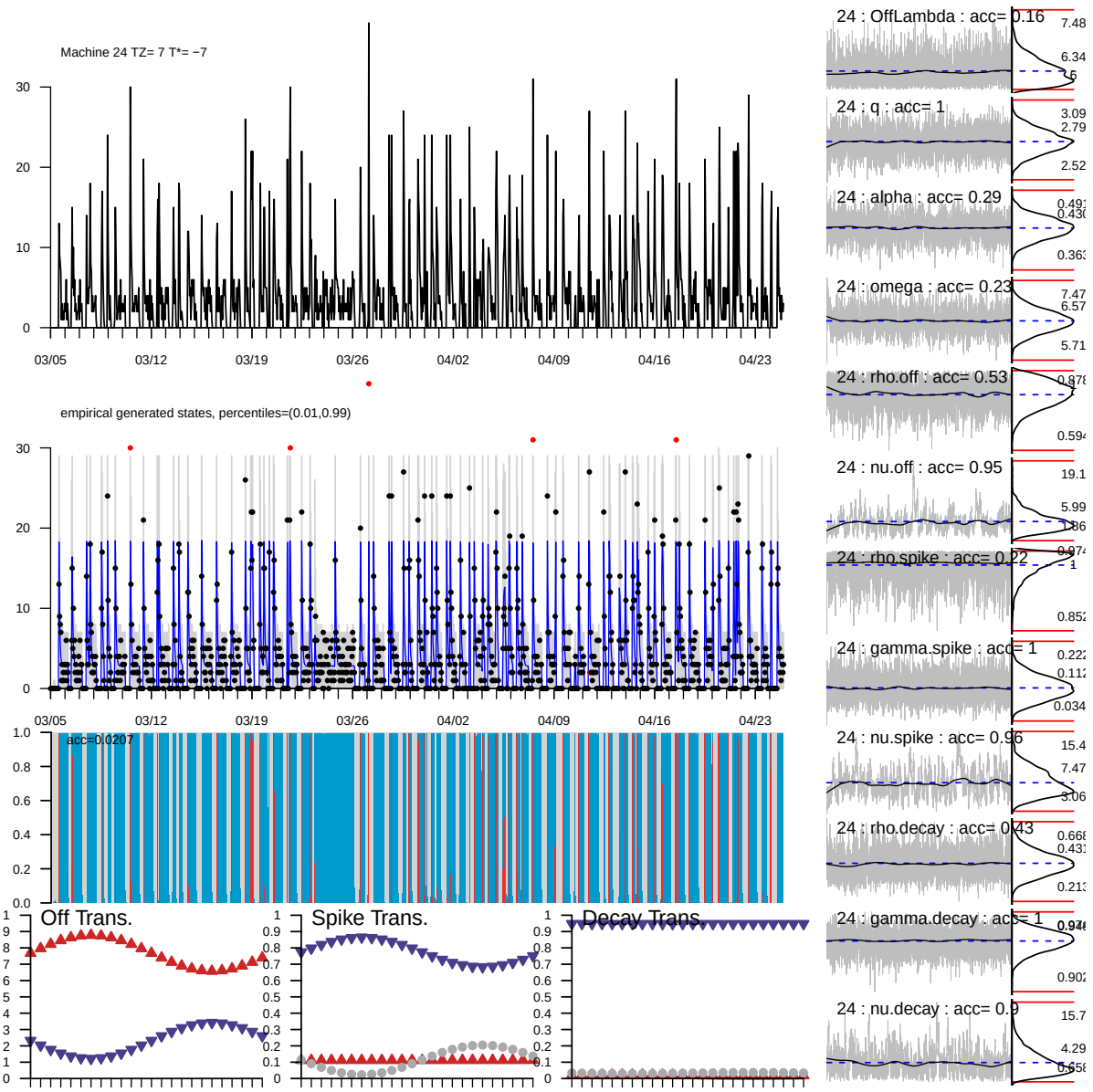

Figure 4.21: A machine that shows obvious periodic spiking patterns. The model appears to capture this pattern well. The MCMC iterations show no obvious signs of non-convergence, and the baseline rate is consistent with sandbox tests (between 3 and 5 per hour), with a spike rate of 6.5. Despite the periodic spikes, the machine very regularly reverts back to the baseline rate.

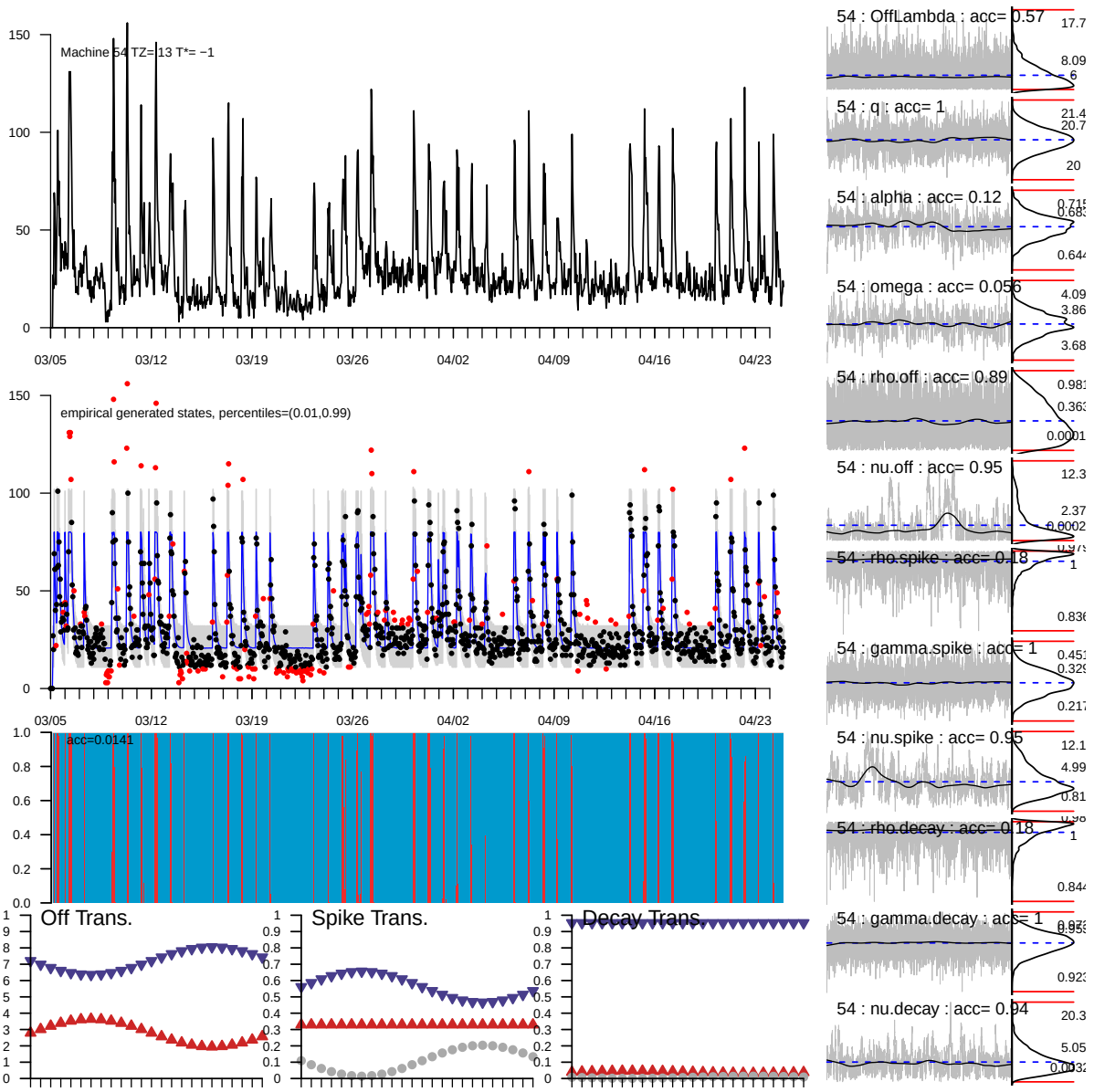

Figure 4.22: A "machine" that shows evidence of multiple active hosts. The estimated baseline rate is 20.7, approximately 4 times higher than was estimated from sandbox tests. Obvious daily patterns follow workday cycles and could be the result of multiple users initiating network connections during business hours. There also appear to be several categorical shifts in the baseline, possibly a result of one or more of the underlying machines being turned off. Parameters show no signs of non-convergence, except for some fluctuations in the off-state transition probability parameters. This is not surprising, since there are no hours with 0 counts in the machine history.

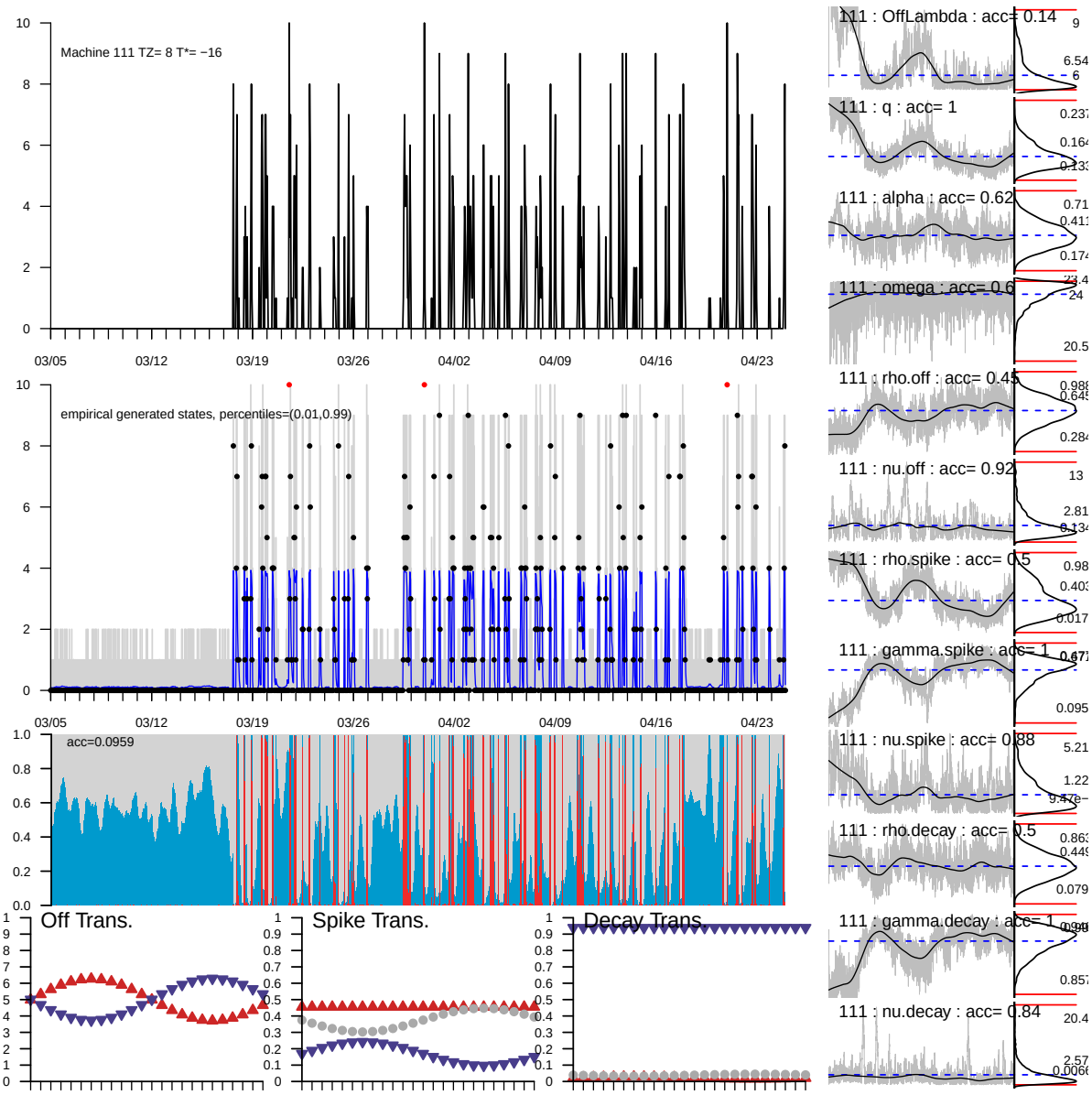

Figure 4.23: Machine summary typical of a machine with q vs. ω "confusion" brought about by mis-fit of the transition model. Though the machine appears to have a steady scan rate, overuse of spike states with nearly every non-zero count leads to a low value of q (0.16) to account for decay states with 0 counts, and a high value of ω (23.4) to compensate. The product $q\omega = 3.93$ which typifies the active scan rate in the population. MCMC iteration chains plotted after the burn-in period (right column) show obvious signs of non-convergence.

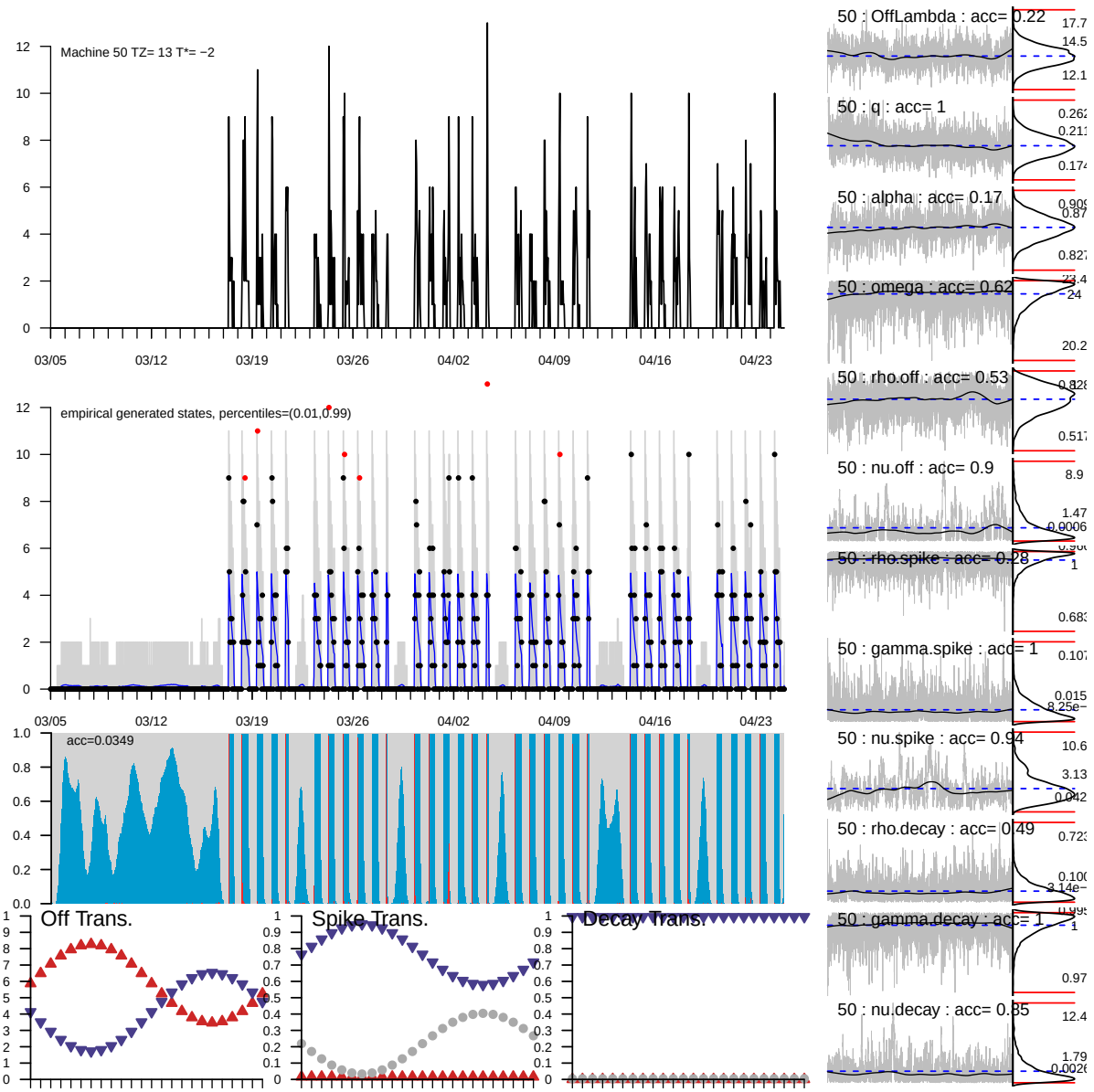

Figure 4.24: Machine summary typical of the 12 machines with few convergence issues and persistent activity, but low baseline rates ($q < 0.25$) and high spike rates ($\omega > 23$). These machines show sporadic activity following workday patterns. The transition model appears to provide a good fit, but it also appears that the activity does not last long enough to "settle down" to a visible baseline, before the machine is turned off.

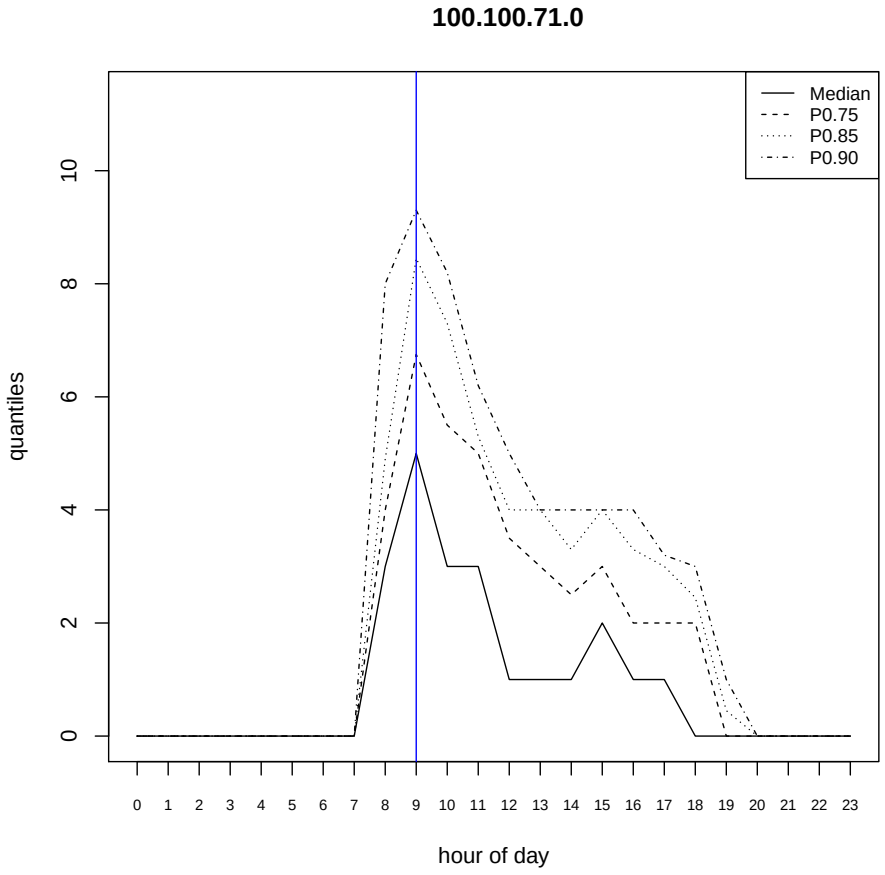

Figure 4.25: Quantile plot of connection attempts by hour of day for Machine 50. Though the estimates of q and ω show signs of non-identifiability, the problem is not due to the same confusion as is evident in Machine 111; Machine 50 does display a clear daily pattern of spike and decay that follows workday hours.

4.6.6 Discussion

The single-host model developed in Section 4.1 was informed more by sandbox tests of Conficker-C than by the observation of machines "in the wild", where behavior can depend more on network properties and on how the malware interacts with day to day user activity. Sandbox tests have historically been geared toward discovering signatures of infection in a machine's operating system, as opposed to studying the variability of behavioral network profiles. Sandbox tests are expensive to carry out, and they are often performed in the context of studying thousands of malicious files, so it is not unusual for a malicious executable to be run through a sandbox only once, with one operating system configuration, for only a few minutes,

before moving to the next malicious file. For this reason, the model of active times and the association of on/off activity and network connections with spikes is preliminary and could benefit from further analysis of infected machines in a more representative network environment than an isolated sandbox–for example a honeynet, or an external network that can be strongly vetted as a one-to-one static allocation of machines to IP addresses. Still, the preliminary model provided a good fit to many of the 154 IP addresses that were later observed as static one-to-one allocations with machines infected by Waledac.

Studying the Waledac IP addresses invites the question: to what extent do we trust that these IP addresses represent individual machines? If the addresses could be truly vetted as individual machines representative of the Conficker-C population as a whole, then the analysis of these IP addresses can be used to inform the priors to account for the interaction between malicious behavior and user activity, as well as informing the distribution of active behavior profiles in the overall population. But while the one-to-one relationship observed in the Waledac data does suggest some stability in these IP addresses, it is by no means a certain indication of a directly observable population.

Is Machine 54 really a small population of infected machines? A UDP scanning process could theoretically initiate a maximum of 72000 scan requests in a single hour, which, on the network telescope used in this study, would result in an approximate hit rate of 108 requests per hour for a spiking machine. This seems consistent with the observed counts on Machine 54, if at the high end of the spectrum of rates. Furthermore, UDP processes can be initiated for multiple network connections on one machine, which could explain large baseline rates with categorical changes over time. Information on the prevalence of such "heavy hitting" UDP scan machines (if they exist at all) is necessary for better clarification of the tails of the population prior distribution for the generation parameters q, ω, and α, which in turn affect population estimates in networks that are not directly observable.

Population estimates for the generation-allocation model are sensitive to the prior information in the single host model; without an informed prior, the likelihood cannot distinguish one large heavy hitter from many smaller machines. But this prior sensitivity does not necessarily relegate population estimates to a purely subjective interpretation. Many of the quantities that account for prior uncertainty in this study can be measured and estimated through a careful survey or census of user profiles and of network administration, hardware, throughput, and behavior. A more thorough study of Conficker-C's behavior in honeynets and

on infected networks can help to provide information to disambiguate these situations. In general, when a broad population study is of interest for a botnet, it may be useful to run the botnet's malicious code through multiple sandbox tests that simulate common virtual environments where infected machines reside, for example:

- A corporate desktop with typical workday activity;
- A university computer lab;
- A personal laptop;
- A family PC;
- An Internet cafe;
- An email server or other kind of network infrastructure machine;
- A small business network contracting space through an ISP.

Although some questions remain about the veracity of the Waledac data set representing individual machines, analysis of the data set did reveal the need for some adaptations to the single host model and to the estimation scheme, that can be incorporated into future versions:

- Modeling hourly survival with a single parameter is not sufficient to capture clean-ups and deaths in the population; the model should be adapted to estimate probability of cleanup on an hour-by-hour basis.
- To help mixing in proposals of new birth and death times for a machine, it may be advantageous to update $(t^{\beta}, t^{\dagger}, \epsilon)$ jointly with the mean time in off states λ_o, to avoid rejecting later birth times or earlier death times due to a sudden removal or addition of a large outlier to the collection of strings of consecutive off states.
- To avoid the difficulty of q sinking toward 0, institute a strict lower bound in the prior, that still admits reasonable baseline rates in accordance with sandbox testing.
- To help mixing for states, consider a joint proposal for runs of consecutive states.

- Incorporate a single-model rate increase for the hours following the March 17th, to account for machines that may have received new instructions as opposed to an increase in the overall population or in active machines.

- The decay rate α should be fairly steep for a single machine; high decay rates (and thus slow decay) could be an indication of churn or tail-off of activity among larger populations of infected machines.

- It may be useful to define a joint prior for the parameters $\theta = (q, \omega, \alpha)$ in order to alleviate some of the identifiability and model mis-fit difficulties in the MCMC estimation.

Chapter 5

Inference For Non-informative Network Allocation

A non-informative network allocation is one for which all possible allocations between generated counts and IP addresses are equally likely, possibly within a set of strict boundary thresholds. This arises most notably in practice with many-to-one assignments. The IP address space for IPv4 consists of 2^{32} possible addresses, although not all of these addresses are routable. The Internet Assigned Numbers Authority (IANA) coordinates the assignment of IP address space to entities such as governments, companies, and universities. IP addresses are also allocated to and ISPs, who portion out their space among many sub-networks associated with organizations and individuals. There are several reasons why institutions may not wish to assign every machine on their network to an externally facing, routable IPv4 address. Firstly, these addresses are scarce and costly resources. Secondly, stricter control of external access points can facilitate network monitoring and security measures. If all communications are required to pass through a single point of contact, it is easier to implement firewalls, access control lists and behavioral monitoring.

NAT is a common protocol used on the Internet IPv4 infrastructure. A network that is configured to use NAT often has only a single routable IP address, that is used as a gateway for any number of internal machines connected to the network. Internally, machines are assigned IP addresses corresponding to private address space according to the Internet Engineering Task Force (IETF) Request For Comments (RFC) doc-

ument RFC1918 [1]. Address blocks $10.0.0.0/8$, $192.168.0.0/16$, and $172.16.0.0/12$ are reserved for private allocation and are considered un-routable or sinkholed when requested over the Internet. Administrators setting up internal networks do not need to coordinate with IANA in order to assign addresses in these blocks to their internal devices. The NAT protocol then acts as a bridge between the local network and the global Internet.

Any internal machines wishing to connect to external resources do so by communicating with the NAT gateway device, which forwards these requests through the external, routable IP address, and keeps track of the communications on a machine-by-machine basis by maintaining tables of open connections and by routing requests to the appropriate private addresses. From the perspective of the outside observer, all communications from any machine in the internal network are seen as originating from the single routable IP address. In practice, the term *NAT* is used interchangeably to describe the protocol or the IP address that is seen as the external gateway. The same kind of non-informative network allocation also arises for behavior observed via the HTTP (web traffic), on networks that use HTTP proxies in order to provide security and content filtering. Again, internal machines wishing to connect with the outside world are all routed through a single access point, which forwards the connections using its own IP address as the local end point.

A collection of IP addresses that are known proxies or NAT devices, each housing a distinct population of machines behind it, is a practical example of a non-informative network structure; within the NAT network, all traffic passes through the same IP address with probability 1, at any time of the day. Prior information can be used to determine the boundaries of IP addresses that can host possible NAT devices with distinct populations of machines behind them. IANA net block assignments, as well as sub-assignments among Regional Internet Registries, are publically available and can provide information on hard IP address boundaries across which overlap or switching of machines is highly unlikely. Furthermore, NAT is a popular configuration for large corporate networks, and the machines residing behind a gateway are often homogenous in their operating systems and policies. Corporate networks also tend to display regular spikes of activity in concordance with typical working hours (8am to 6pm), with nearly all machines residing in the same time zone. This homogeneity helps to distinguish corporate NAT traffic, which may travel between IP addresses due to load-balancing, from aggregations of more diverse activity across net blocks assigned to

[1] `http://tools.ietf.org/html/rfc1918`

other institutions such as ISPs.

Section 5.1 presents the network model for a proxy, NAT device, or gateway, and describes inference for H and the generation process parameters ξ and κ in this case. Section 5.2 describes the complete conditional distribution for counts y_t, as well as the reversible jump steps for merging and splitting machines in a NAT environment for the Conficker-C single host model. Section 5.3 presents an analysis and preliminary results from a subset of $/24$s in the data set, with the goal of adapting the constraints of the single-host model and assessing the estimation methods.

5.1 Inference for a NAT device, gateway or other proxy

In the directly observable population, the number of machines H is known and equal to J, and the counts are visible such that $w_{ijt} = y_{it} = x_{jt}$ for all t. For a NAT device, these equalities no longer hold; H is unobserved, as is w_{ijt} and y_{it}. Both of the latter quantities can be considered as data augmentation variables; they are not particularly of interest in the study of the malware behavioral model or in estimation of the population size, but including them in the joint distribution for the generation and allocation model makes the likelihood easy to write down, and leads to simpler proposal steps in the MCMC.

For a NAT with a single outward-facing IP address, the allocation model in equation 2.5 reduces to $J = 1$ and $w_{i1t} = y_{it}$ with probability 1. The graph $\mathbf{W}$ is thus the collection $y_{11}, \ldots, y_{HT}$ of all counts generated from infected machines throughout the observation window. The observed value x_t at each hour is the sum of the activity generated from all infected machines in that hour. The posterior distribution has the form:

$$\begin{aligned} p(\mathbf{W}, H, \kappa, \xi \mid \mathbf{x}) &\propto \pi(H)\pi(\kappa) \prod_{i=1}^{H} \pi(\xi_i \mid \kappa) \prod_{t=1}^{T} f(y_{it} \mid \xi_i, y_{i1}, \ldots, y_{it-1}) \\ &\times \prod_{t=1}^{T} 1_{\{\sum_i y_{it} = x_t\}}, \end{aligned} \tag{5.1}$$

where $1_{\{C\}}$ is the indicator function that returns 1 when the expression C is true, and 0 otherwise. The changes in this distribution from the directly observable population model are the prior $\pi(H)$ for the un-

observed population size, and the summation constraint on the machine counts. The model is invariant to permutation of labels among machines $1, \ldots, H$; to induce identifiability, the machines can be labeled according to an ordering of the parameters ξ_i.

The conditional distribution $\pi(\kappa, \xi \mid H, \mathbf{W}, \mathbf{x})$ does not depend on the IP address counts $\mathbf{x}$, and is proportional to the posterior distribution of hyperparameters and generation parameters given the machine counts $\mathbf{y}$ in the directly observable population. Thus the steps designed for updating κ and ξ in the directly observable MCMC scheme can be used in the model for the NAT as well. It remains to design the steps for updating the counts $y_{11} \ldots y_{HT}$ as they are assigned among machines, and for updating the population H.

For any single count y_{it} in the graph, the summation constraint $\sum_i y_{it} = x_t$ forces the conditional distribution $\pi(y_{it} \mid \mathbf{W}_{|y_{it}}, H, x_1, \ldots, x_T, \kappa, \xi)$ to put probability 1 on the current value. Conditional independence in the generation model among the observations $y_{i1}, \ldots, y_{iT}$ yield a simplified expression for the joint conditional distribution of counts $y_{1t}, \ldots, y_{Ht}$ for a given time t. Suppose the generation function $f(y_{it} \mid \xi_i, y_{i1}, \ldots, y_{it-1})$ can be written as $f(y_{it} \mid \xi_i)$, a function of y_{it} and the parameter ξ_i. This arises for example in a hierarchical Hidden Markov structure where the time dependence in generation is modeled through the parameters $\xi_{i1}, \ldots, \xi_{it}$. Then the complete conditional distribution $\pi(y_{1t}, \ldots, y_{Ht} \mid \mathbf{W}_{|y_{1t}, \ldots, y_{Ht}}, H, x_t, \kappa, \xi)$ has the form:

$$\pi(y_{1t}, \ldots, y_{Ht} \mid \mathbf{W}_{|y_{1t}, \ldots, y_{Ht}}, H, x_t, \kappa, \xi) \quad \propto \quad 1_{\{\sum_i y_{it} = x_t\}} \prod_{i=1}^{H} f(y_{it} \mid \xi_i). \tag{5.2}$$

The population hyperparameters κ and the counts in times $1, \ldots, t-1, t+1, \ldots, T$ all cancel from the expression, and the conditional distribution can thus be written then as $\pi(y_{1t}, \ldots, y_{Ht} \mid H, x_t, \xi)$.

The space of probable population values H cannot be explored by a traditional Metropolis-Hastings or Gibbs step due to the necessity to change the dimensions of the parameter space. Within an MCMC framework, the posterior distribution of H is explored by adding or deleting machines to the graph, using a merge and split technique. A merge of two machines into one decrements H by 1, while a split of one machine into two increments H by 1. The value H can also be envisioned as an index to a family of models for model selection. As it resides in a discrete space, the marginal posterior distribution $p(H = h \mid \mathbf{x})$ is equal to the prior $\pi(h)$ multiplied by the normalizing constant $m(\mathbf{x})$ of the posterior distribution of all model parameters and hyperparameters given the IP address counts with the population size fixed at h.

5.2 NAT inference for Conficker-C

5.2.1 Updating y_{it}

For the Conficker-C model, the count y_{it} at time t is a Poisson distributed variable conditionally independent of $y_{i1}, \ldots, y_{it-1}$ given the mean parameter $\lambda_{it} = \lambda(\eta_{it}, q_i, \omega_i, \alpha_i)$, as described in Section 4.1. Consider the conditional distribution of counts across infected machines for a fixed hour t; the time index is suppressed in the following expressions for ease of notation. The complete conditional distribution $\pi(y_1, \ldots, y_H \mid H, x, \xi)$ in equation 5.2 has the form:

$$\pi(y_1, \ldots, y_H \mid H, x, \xi) \quad = \quad \frac{\prod_{i=1}^{H} e^{-\lambda_i} \lambda_i^{y_i} (y_i!)^{-1} 1_{\{\sum_i y_i = x\}}}{\sum\limits_{y_1 \ldots y_H : \sum_i y_i = x} \prod_{i=1}^{H} e^{-\lambda_i} \lambda_i^{y_i} (y_i!)^{-1}}$$

This expression is the kernel of a multinomial distribution. The term $e^{-\sum_i \lambda_i}$ does not depend on the observations $y_1 \ldots y_H$ and cancels from the numerator and denominator. Multiply the numerator and denominator by $x!$ to obtain:

$$\pi(y_1, \ldots, y_H \mid H, x, \xi) \quad = \quad \frac{\frac{x!}{\prod_i y_i!} \prod_i \lambda_i^{y_i} 1_{\{\sum_i y_i = x\}}}{\sum\limits_{y_1, \ldots, y_H : \sum_i y_i = x} \frac{x!}{\prod_i y_i!} \prod_i \lambda_i^{y_i}}$$

By the Multinomial Theorem, the denominator reduces to $(\sum_i \lambda_i)^x = \prod_i (\sum_k \lambda_k)^{y_i}$. This normalizes the mean values $\lambda_1, \ldots, \lambda_H$ in the numerator to a set of probabilities that sum to 1, and the complete conditional distribution is:

$$\pi(y_1, \ldots, y_H \mid H, x, \xi) \quad = \quad \left(\begin{array}{c} x \\ y_1, \ldots, y_H \end{array} \right) \prod_{i=1}^{H} \left(\frac{\lambda_i}{\sum_k \lambda_k} \right)^{y_i} 1_{\{\sum_i y_i = x\}},$$

where $\left(\begin{array}{c} n \\ k_1, \ldots, k_H \end{array} \right)$ is the multinomial coefficient for partitioning n into H subsets of size k_1 through k_H.

Thus at each time t the conditional distribution of observed counts from equation 5.2 is multinomial with probability vector equal to the renormalized mean vector, and total equal to the summation

constraint x_t. This multinomial result also holds for the conditional distribution of any sub-collection $y_{st} : s \subset \{1, \ldots, H\}$, with the means again normalized across elements of s, and the total equal to $\sum_{i \in s} y_s$, subtracting the sum of fixed elements $\{s\}^C$ from the summation constraint x_t. Updating (or rearranging) of counts at any hour can thus be achieved by a Gibbs step in the chain. In a random sweep strategy, if the count rearrangement move type is selected, then a network or NAT proxy with more than one associated machine is selected at random from the population, and a series of hours are selected at random from $t \in [0, ..., T-1]$. Then the observed network counts x_t are arranged among each machine i according to its mean λ_{it}.

5.2.2 Merging and splitting machines

Merging and splitting machines is a very high-dimensional move, with many possible points of failure in generating "close" candidates. Let $M = (\boldsymbol{\eta}, t^{\beta}, t^{\dagger}, \epsilon, \psi, \theta, \boldsymbol{y})$ be the full set of machine and graph parameters that need to be generated for a new machine in a merge-split step. There are two proposal distributions to consider: $g_{2\to1}(M_{12} \mid M_1, M_2)$ and $g_{1\to2}(M_1, M_2 \mid M_{12})$. Note that the updating of the number of machines H is implicit: a merge decrements H and a split increments H.

In traditional MCMC methods across high dimensions, blocking and Gibbs or Metropolis-within-Gibbs steps can be used to update a vector of parameters one element (or one small group) at a time. Suppose $\theta = (\theta_1, \ldots, \theta_I)$ is a multidimensional parameter. In a sweep strategy, each component may be updated in sequence, such that the i-th element or block can be updated at step m with a draw from the complete conditional distribution $\pi(\theta_i^* \mid y, \theta_1^m, \ldots \theta_{i-1}^m, \theta_{i+1}^{m-1}, \theta_I^{m-1})$, without violating the detailed balance condition. In essence each component updating step can be seen as a different move type that transitions the state along only the i-th block dimension using a transition kernel g_i. In this case each component can be updated independently of other move types using the acceptance ratio:

$$\alpha_i \quad = \quad \min\left\{1, \frac{\ell(y; \theta : \theta_i = \theta_i^*)\pi(\theta_i^*)g_i(\theta_i^* \to \theta_i^m)}{\ell(y; \theta : \theta_i = \theta_i^m)\pi(\theta_i^m)g_i(\theta_i^m \to \theta_i^*)}\right\}$$

But this acceptance ratio can only be calculated component-wise when the state can be held fixed for all components except θ_i, in which case both likelihoods are well-defined. In a merge-split situation, this is not the case. For any state in the graph, assume that the probability of choosing a merge vs. a split state is

a constant, $p_{2\to1}$, with $p_{1\to2} = 1 - p_{2\to1}$. Let s_* be the probability of selecting the candidate machine or machines from the graph for the move type attempted. The detailed balance condition to uphold is

$$\begin{aligned} & \ell(\boldsymbol{y}; M_{12})\pi(H, M_{12})(p_{1\to2})(s_{1\to2})g_{1\to2}(M_1, M_2 \mid M_{12}) \\ = \quad & \ell(\boldsymbol{y}; M_1, M_2)\pi(H+1, M_1, M_2)(p_{2\to1})(s_{2\to1})g_{2\to1}(M_{12} \mid M_1, M_2) \end{aligned} \tag{5.3}$$

For the general case, suppose M is the state at the current iteration and M^* is the proposal. Although the components of M^* cannot be accepted/rejected individually, they can be designed in blocks. Let $M_i^* \in M^*$ be a sub-block of parameters in the proposal, with corresponding block of parameters $M_i \in M$ interpreted as their paired components for merged vs. split machines (for example, q_{12} and its corresponding rates (q_1, q_2)). Let $\boldsymbol{\delta}_i$ and $\boldsymbol{\delta}_i^{'}$ be the variables required to match dimensions in M_i^* and M_i, with candidate proposals $g_i(\boldsymbol{\delta}_i)$, $g_{i'}(\boldsymbol{\delta}_i^{'})$ and transformation $h_i(M_i, \boldsymbol{\delta}_i) \to (M_i^*, \boldsymbol{\delta}_i^{'})$ independent of any other block $j \neq i$; this transformation has Jacobian $|J_i| = \left|\frac{\partial(M_i^*, \boldsymbol{\delta}_i^{'})}{\partial(M_i, \boldsymbol{\delta}_i)}\right|$ and its reciprocal J_i^{-1} for the reverse move. The full merge-split step of all components can be seen as a composite of the functions h_i with a block diagonal Jacobian matrix, and detailed balance holds across the entire space when the acceptance ratio $\alpha = \min\{1, A\}$ where

$$A = \frac{\ell(\boldsymbol{y}; M^*)\pi(H^*, M^*)}{\ell(\boldsymbol{y}; M)\pi(H, M)} \frac{p \;\; s}{p_* s_*} \prod_i \frac{g_{i'}(\boldsymbol{\delta}_i^{'})}{g_i(\boldsymbol{\delta}_i)} |J_i|$$

A similar result holds if the transformations h_i are designed as a strict sequence of conditional distributions based on variables sampled from blocks $k < i$. In this case the joint distribution of the proposal is the product of the conditional distributions $g_i(\boldsymbol{\delta}_i|\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_{i-1})$, and the Jacobian matrix is lower block triangular, which again has determinant equal to $\prod_i |J_i|$.

The strict sequence of conditional distributions is the approach used for the Conficker-C merge-split step. Green's reversible jump MCMC dimension matching technique (Green, 1995) is used to account for relationships across dimensions in the continuous variables, while categorical variables can be updated using discrete probability tables. The scheme for updating a component block must try to find a good balance between a proposal that has overall high likelihood and interpretability relative to the dimensions already sampled, but that is also "close" to the parameters of the current machine or machines in the graph. Section 5.2.3 outlines the general form of proposal distribution that is used for transformations of continuous

parameters across dimensions in a component. Section 5.2.4 discusses proposal distributions and ordering strategies for updating each component in M. Section 5.2.5 outlines selection probabilities of move types and candidates, suggests a prior for H, and briefly addresses identifiability.

5.2.3 Generating merge-split candidates for continuous parameters

Suppose a parameter $l < \theta < u$ is linked interpretably to its differently-dimensional counterpart $l < \theta' < u$ through a function $g(\theta)$. A simple example is an unbounded average with $l = -\infty$, $u = \infty$, $g(\theta) = \theta$ and $g^{-1}(\theta'_1, \theta'_2) = \frac{1}{2}(\theta'_1 + \theta'_2)$. A 2-dimensional mapping could, for example, generate $\delta \sim N(0, 1)$ and set $g(u, \theta)$ as:

$$\begin{aligned} \theta'_1 &= \theta + \delta \\ \theta'_2 &= \theta - \delta, \end{aligned}$$

with the inverse $g^{-1}(\theta'_1, \theta'_2) = ((\theta'_1 + \theta'_2)/2, (\theta'_1 - \theta'_2)/2)$, and Jacobians equal to 2 and $\frac{1}{2}$ for g and g^{-1}. This produces symmetric candidates in a split step that always exactly satisfy the constraint of g. If θ'_1 and θ'_2 are generated as the i-th component proposal of a larger split step, the acceptance ratio for the step is adjusted by the RJMCMC ratio:

$$\alpha_i = 2\left[\frac{1}{\sqrt{2\pi}\sigma}\exp\{-u^2/2\sigma^2\}\right]^{-1},$$

and its reciprocal for the merge step,

$$\alpha_i^{-1} = \frac{1}{2}\frac{1}{\sqrt{2\pi}\sigma}\exp\{-[(\theta'_1 - \theta'_2)/2]^2/2\sigma^2\}$$

For bounded variables on $[l, u]$, a rescaled asymmetric average can be used, for example by simulating a variate δ on the range $[0, u - \theta]$ and setting

$$\begin{aligned} \theta'_1 &= \theta + \delta \\ \theta'_2 &= \theta - \frac{\delta}{q - u}(q - l). \end{aligned}$$

When $\delta \sim \beta^*(\theta, u-\theta, \alpha, \beta)$ is a scaled Beta value, candidates satisfy the constraint

$$\theta = \frac{u\theta_2' - l\theta_1'}{(u-\theta_1') + (\theta_2' - l)}.$$

This method produces symmetric candidates around θ when $\theta = \frac{u-l}{2}$, and the expected value of the distance $D = \theta_1' - \theta_2'$ is independent of the position of θ across the range. The Jacobian of the transformation $g(\theta, \delta)$ is

$$\left|\frac{\partial(\theta, u)}{\partial(\theta_1', \theta_2')}\right| = \left|\frac{\partial\theta}{\partial\theta_2'}\right| = \frac{u\theta_1' - l\theta_2' + l^2}{\left[(u-\theta_1') + (\theta_2' - l)\right]^2},$$

and the acceptance ratio for a split step is adjusted by the product of the Jacobian and the scaled Beta p.d.f. of the candidate δ.

The result is also asymmetric in that θ_1' is the maximum value and θ_2' is the minimum. It is equivalent either to assign the generated deviates $\theta_{\max}$ and $\theta_{\min}$ among θ_1', θ_2' using a coin flip of probability p, or to add a step to the simulation that with probability p uses the reciprocal interval $[l, \theta]$ as the basis for generating the deviate δ, and sets $\theta_1' = \theta - \delta$ with appropriate rescaling to obtain the maximum value θ_2', and adjusting α_i by p to account for the choice.

Finding a suitable interpretable transformation on a bounded interval can result in changes of scale. For example, suppose that the relationship is a scaled average, $\theta = \frac{\phi}{2}(\theta_1' + \theta_2')$, where $1 < \phi < 2$ is a deterministic calculation of θ_1', θ_2' and external state variables from a prior block proposal. This arises in the component-wise updating if, for example, candidate state values η are proposed before baseline rate parameters q. Suppose that two machines have only baseline level activity across their ranges. When merging machines with overlapping active hours, the merged rate q_{12} should be near the sum of the two individual rates, $q_1 + q_2$. However, if the two individual machines operate mutually exclusively, the resulting merged baseline rate q_{12} should be closer to the average of the two individual rates in order to minimize the deviance of the counts y_t from the mean. The function $\frac{\phi}{2}(q_1 + q_2) = q_{12}$ maps $[l, u] \times [l, u]$ to $[l\phi, u\phi]$. To enforce detailed balance in the case of overlapping regions, it becomes impossible to split a machine into overlapping regions when the merged rate $q_{12} < l\phi$, and impossible to merge two machines when $q_1 + q_2 > u\phi$. These candidates cause an immediate rejection of the merge/split step. When the necessary

constraints hold, an asymmetric average centered at q_{12}/ϕ can be used to generate two candidates, with the Jacobian adjusted accordingly and the appropriate reciprocals and inverses used for the merge step.

Another way to proceed is to add another degree of freedom while preserving the interpretable relationship $g(\theta)$ in expected value. Detailed balance is maintained with a suitable proposal f that generates deviates θ' such that $l < \theta' < u$ and $E(\theta') = g(\theta)$. For the reverse step, a proposal function f' is used to generate deviates θ such that $E(\theta) = g^{-1}(\theta')$. This scheme is more of a traditional MCMC step than Green's example, as it does not rely on deterministic transformations of the variables θ and θ'. Instead of a two-dimensional mapping from $(\theta, \delta) \rightarrow (\theta'_1, \theta'_2)$, the dimension matching for the merge-split RJMCMC step occurs over the three-dimensional mapping of $(\delta_1, \delta_2, \theta) \rightarrow (\theta'_1, \theta'_2, \delta)$. For the simple unbounded average, let $\delta \sim N(\frac{1}{2}(\theta'_1 + \theta'_2), \sigma)$, and let $\delta_1, \delta_2 \sim N(\theta, \sigma)$. Set the identity transformation:

$$\begin{aligned} \delta &= \theta \\ \theta'_1 &= \delta_1 \\ \theta'_2 &= \delta_2 \end{aligned}$$

The Jacobian $\left|\frac{\partial(\theta,\delta_1,\delta_2)}{\partial(\delta,\theta'_1,\theta'_2)}\right| = 1$ in both directions, and the interpretability arises through the mean values. When the proposals $f_1(\delta_1|\theta)$, $f_2(\delta_2|\theta)$ and $f_3(\delta|\theta'_1, \theta'_2)$ are the i-th components of a larger merge-split that preserves order of operations in both directions, the overall acceptance ratio is adjusted by the ratio of proposal densities:

$$\alpha_i = \frac{f_{12}(\delta|\theta'_1, \theta'_2)}{f_1(\delta_1|\theta), f_2(\delta_2|\theta)}$$

for a split step, and $(\alpha_i)^{-1}$ for its paired merge step. This three-dimensional scheme can also be used for independence steps, or for steps in a block i that depend only on proposal values in previously sampled blocks and not on the cross-dimensional parameters.

5.2.4 Component merge-split steps and ordering

Merging and splitting machines requires generating new sets of parameters M, consisting of the high-dimensional vector $(\boldsymbol{\eta}, t^{\beta}, t^{\dagger}, \epsilon, \psi, \theta, \boldsymbol{y})$, for any new machine that is added into the graph. The goal is to

block the variables suitably into subcomponents M_i and to find an order of updating $M_1 \rightarrow ... \rightarrow M_k$, and a resulting set of candidate proposal distributions, that will simplify the detailed balance condition and lead to good mixing in the chain. It is useful to examine the format of the machine-specific components of the acceptance ratio as a product of candidate proposal distributions and path likelihood ratios, with the path likelihoods expressed in terms of priors for the continuous variables, an expression for birth/death/immunity and off states, and a product of state transition probabilities and Poisson count ratios over time.

Conditional on both $\boldsymbol{\eta}$ and θ, the connection rate at any time t is given by equation 4.4 in Chapter 3 as:

$$\lambda_t(\eta_1, \ldots, \eta_{t-1}, q, \omega, \alpha) \quad = \quad q(1 + \alpha^k(\omega - 1)), \tag{5.4}$$

where the lag k depends on the last encountered spike or off state. The dependence of λ_t on $\boldsymbol{\eta}$ and θ is suppressed for ease of notation, and the connection rate at time t is written as λ_t. Let

$$O_{\boldsymbol{\eta}} \quad = \quad \sum_{t \in \mathcal{T}_{o|o}} c_t$$

be the number of off-to-off state transitions in the path characterized by $\boldsymbol{\eta}$, and let $\left[O_{\boldsymbol{\eta}}\right]_u^v$ be the number restricted to the range $u \leq t \leq v$. Let

$$R_{\boldsymbol{\eta}} = |\mathcal{T}_{o|o}|$$

be the number of runs of off states observed in the path characterized by $\boldsymbol{\eta}$, and let $\left[R_{\boldsymbol{\eta}}\right]_u^v = |u \leq t \leq v : t \in \mathcal{T}_{o|o}|$ be the number of runs of off states restricted to the interval $[u, v]$. Given $\boldsymbol{\phi}, \zeta$ and λ_o, the complete conditional distribution of off states, birth, death and immunity can be written as

$$\pi(t^\beta, t^\dagger, \epsilon, \mathcal{O} \mid \boldsymbol{\phi}, \zeta, \lambda_o) \quad \propto \quad \zeta^\epsilon \left[(1-\zeta)(1-\phi_{t^\dagger}) \prod_{t \in \mathcal{S} \cup \mathcal{D}} \phi_t\right]^{1-\epsilon} \tag{5.5}$$

$$\times \quad \frac{\exp\left\{\lambda_o \left[R_{\boldsymbol{\eta}}\right]_{t^\beta}^{t^\dagger}\right\} \lambda_o^{\left[O_{\boldsymbol{\eta}}\right]_{t^\beta}^{t^\dagger}}}{\prod_{t \in \mathcal{T}_{o|o}} c_t!} \prod_{t \in \mathcal{O}} 1_{y_t = 0}. \tag{5.6}$$

This follows the form of Equation 4.13 from Section 4.5.3, but includes the likelihood of all runs of off-

state transitions as opposed to only the ones that may be affected by a change in the birth or death dates. The dependence on $\boldsymbol{\phi}, \zeta$ and λ_o is suppressed for ease of notation, and the likelihood of off states, birth, death, and immunity is written as $\pi(t^\beta, t^\dagger, \epsilon, \mathcal{O})$. Finally, for $t < T-1$, denote the active state transition probabilities as

$$\pi(\eta_{t+1} \mid \eta_t, \psi) \quad = \quad \left[\pi_{\eta_t|\eta_{t+1}}(t, \psi)\right]^{1_{\{\eta_t \neq o, \eta_{t+1} \neq o\}}},$$

where $\pi_{\cdot|\cdot}(t, \psi)$ is the transition probability defined in Section 4.1.2 as a function of the parameters ψ, and $\pi(\eta_T \mid \eta_{T-1}, \psi) = 1$ to indicate no further transitions on the last hour of the observation period.

To construct the acceptance ratio, firstly suppose that each component of the parameter vector M is proposed independently (such that order of the proposals does not matter). For a split step that takes M_{12} to individuals M_1 and M_2, let $\alpha_{1\to2}$ be the acceptance ratio excluding the prior on H and adjustments for move type and machine selection:

$$\alpha_{1\to2} \quad = \quad \frac{\ell(M_1, M_2; \boldsymbol{x})\pi(M_1, M_2)g_{2\to1}(M_{12} \mid M_1, M_2)}{\ell(M_{12}; \boldsymbol{x})\pi(M_{12})g_{1\to2}(M_1, M_2 \mid M_{12})}.$$

The format of $\alpha_{1\to2}$ is as follows:

$$\begin{aligned}
\alpha_{1\to2} &= (\alpha_{1\to2}|_\theta)(\alpha_{1\to2}|_\psi)(\alpha_{1\to2}|_\eta)(\alpha_{1\to2}|_{\beta\delta\epsilon})(\alpha_{1\to2}|_y) \\
&= \prod_{\theta_i \in \theta} \frac{\pi(\theta_{i,1})\pi(\theta_{i,2})}{\pi(\theta_{i,12})} \times \frac{g_i(\theta_{i,12} \mid \theta_{i,1}, \theta_{i,2})}{g_i(\theta_{i,1}, \theta_{i,2} \mid \theta_{i,12})}|J_i| && (5.7) \\
&\times \prod_{\psi_k \in \psi} \frac{\pi(\psi_{k,1})\pi(\psi_{k,2})}{\pi(\psi_{k,12})} \times \frac{g_k(\psi_{k,12} \mid \psi_{k,1}, \psi_{k,2})}{g_k(\psi_{k,1}, \psi_{k,2} \mid \psi_{k,12})}|J_k| && (5.8) \\
&\times \prod_{t=0}^{T-1} \frac{\pi(\eta_{t+1,1} \mid \eta_{t,1}, \psi_1)\pi(\eta_{t+1,2} \mid \eta_{t,2}, \psi_2)}{\pi(\eta_{t+1,12} \mid \eta_{t,12}, \psi_{12})} \times \frac{g_\eta(\eta_{t,12} \mid \eta_{t,1}, \eta_{t,2})}{g_\eta(\eta_{t,1}, \eta_{t,2} \mid \eta_{t,12})} && (5.9) \\
&\times \frac{\pi(t^\beta, t^\dagger, \epsilon, \mathcal{O})_1\pi(t^\beta, t^\dagger, \epsilon, \mathcal{O})_2}{\pi(t^\beta, t^\dagger, \epsilon, \mathcal{O})_{12}} \times \frac{g_{\beta\delta\epsilon}((t^\beta, t^\dagger, \epsilon)_{12} \mid (t^\beta, t^\dagger, \epsilon)_1, (t^\beta, t^\dagger, \epsilon)_2)}{g_{\beta\delta\epsilon}((t^\beta, t^\dagger, \epsilon)_1, (t^\beta, t^\dagger, \epsilon)_2 \mid (t^\beta, t^\dagger, \epsilon)_{12})} && (5.10) \\
&\times \prod_{t=0}^{T-1} \frac{\exp\{-(\lambda_{t,1} + \lambda_{t,2})\}\, \lambda_{t,1}^{y_{t,1}} \lambda_{t,2}^{y_{t,2}}}{\exp\{-\lambda_{t,12}\}\, \lambda_{t,12}^{y_{t,1}+y_{t,2}}} \frac{(y_{t,1} + y_{t,2})!}{(y_{t,1})!(y_{t,2})!} \times \frac{g_y(y_{t,12} \mid y_{t,1}, y_{t,2})}{g_y(y_{t,1}, y_{t,2} \mid y_{t,12})} && (5.11)
\end{aligned}$$

This format highlights several relationships useful for developing order among candidate proposals.

Observed counts y_t : The splitting of $y_{t,12}$ into $y_{t,1}$ and $y_{t,2}$ should take place after both states $\boldsymbol{\eta}$ and Poisson parameters θ are generated. At any hour t, the counts $y_{t,1}$ and $y_{t,2}$ of the component split machines must sum to the value $y_{t,12}$ of the merged machine, in order to preserve the observed sum x_t of the network touchpoint when counts for all other machines are held fixed. This requirement leads to a fully deterministic step for the component merge step $g_y(y_{t,12} \mid y_{t,1}, y_{t,2})$, and allows for y_{12} to be treated as a fixed constant in any proposal distribution g across any order of updating. It is also clear from the discussion in Section 5.2.1 that at all t, the conditional distribution of y_1, y_2 given sampled candidates λ_1, λ_2 and total y_{12} is Binomial with total count equal to y_{12} and probability of assignment of a single hit to machine 1 equal to the ratio $\frac{\lambda_1}{\lambda_1+\lambda_2}$. If this Binomial distribution is used as a proposal distribution in the last stage of the block updating, then the component log-ratio in Equation 5.11 has the form:

$$\log(\alpha_{1\to 2}|_y) \quad = \quad \sum_{t=0}^{T} \{\lambda_{t,12} - \lambda_{t,1} - \lambda_{t,2} + (y_{t,12})[\log(\lambda_{t,1} + \lambda_{t,2}) - \log(\lambda_{t,12})]\} \,. \tag{5.12}$$

The value is equal to 0 at hours t when $\lambda_{t,12} = \lambda_{t,1} + \lambda_{t,2}$, so any updating strategy for the components will be balanced for both merge and splits when the proposals $\boldsymbol{\eta}$ and θ yield rates comparable in magnitude across all hours. Inequalities between rates are magnified for large observed values $y_{t,12}$. Equality of rates across all t cannot generally be achieved when $T > dim(\theta, \boldsymbol{\eta})$ since λ_t is a deterministic function of the states and Poisson rate parameters. However, a general updating strategy should aim to balance positive vs. negative magnitudes in the difference between λ_{12} and $\lambda_1 + \lambda_2$ across all t.

Transition probabilities ψ : Transition probabilities ρ, ν and γ enter into the acceptance ratio through priors and proposals in Equation 5.8 and in the path likelihoods in Equation 5.9. The off-state sequence average λ_o enters through Equation 5.8 and through the likelihood of observed off-to-off transitions and runs in Equation 5.10. In the single host model, both λ_o and γ can be sampled conditional only on path parameters $\boldsymbol{\eta}$ using a Gibbs step. Detailed balance for an independence step for a parameter $\phi \in \psi$ can be maintained across the three-dimensional space $h(\phi_{12}, u_1, u_2) \to h^{-1}(u, \phi_1, \phi_2)$, with the map h equal to the identity function. This results in a determinant equal to 1 and a jumping ratio equal to the appropriate ratio of component proposal distributions.

With candidates $\boldsymbol{\eta}$, t^β and $t^\dagger$ sampled prior to ψ for the merge or split, the single-model Gibbs step can

be used again in both directions for λ_o and γ, resulting in a cancellation of posterior and proposal terms and a net contribution of 1 to the product of component acceptance ratios. For ρ and ν, the single-host model requires a Metropolis-Hastings step, which relies on moving only a short distance away from the current value. This tactic cannot be generalized as easily to apply to the merge-split step, as a current value for a sampled candidate does not exist, only for its corresponding merged or split parameters. Existing state-to-state transitions from the candidate path $\boldsymbol{\eta}$ can be used to obtain an empirical periodic transition probability profile curve similar to Figure 4.17 in Section 4.6.5. These can be used to obtain simultaneous maximum likelihood estimates for ρ and ν.

Alternatively, a scaled Beta proposal distribution can be used to generate ν_1, ν_2 from ν_{12} and vice versa, under the assumption that high vs. low periodicity tends to be preserved between merged and split machines. With ν fixed, simple weighted least squares methods can be used to obtain a point estimate for ρ; alternatively, the expected value of the empirical probability at the peak value (corresponding to $t^* = 6$) is equal to ρ. A candidate proposal can use a Beta distribution with mean equal to a weighted average of the empirical least squares or peak estimator, and the prior mean for ρ. When no transitions are available to yield information from $\boldsymbol{\eta}$, the candidate reduces to a sample from the prior. If amplitudes can be considered as linked across the merged vs. split machines, the weighted average could use a function of the amplitude or amplitudes at the current state of the machine(s) sampled from the population as merge or split candidates, as opposed to the prior mean.

The true similarity between periodicity in M_{12} and its sub-machines M_1 and M_2 hinges both on the periodicity in the total counts y_{12} and on the mechanism for merging and splitting states in $\boldsymbol{\eta}$. For designing the order of the candidate machine parameters, it seems that the more intuitive method is to use the existing states from the selected machine or machines, along with the observed counts to inform the candidate state vector(s), and then to use each candidate state vector to inform birth, death, immunity and transition probabilities for the new machine(s).

Birth, death and immunity $(t^\beta, t^\dagger, \epsilon)$: Birth, death and immunity enter into the acceptance ratio through the proposals and likelihood ratios in Equation 5.10. If two immune machines always merge into an immune machine, the component acceptance ratio is equal to ζ. It is convenient to rely on the proposal $\boldsymbol{\eta}$ in order to generate new birth, death, and immunity states, as the conditional distribution of $(t^\beta, t^\dagger, \epsilon)$ has at most

six possible configurations when $\boldsymbol{\eta}$ is set. Detailed balance can thus be maintained using an appropriate probability table conditional on the states $\boldsymbol{\eta}$ for both the current machine(s) and candidate machine(s) that preserves immunity for the merge of two immune machines.

Sampling λ_o using a Gibbs step for any merge or split attempt accounts for the contribution to the acceptance ratio of the extra set of off states accrued by a change in birth or death dates (Equation 5.6). With $\boldsymbol{\eta}$ set, the remaining effect of immunity vs. susceptibility in the acceptance ratio is due to the relative likelihood of immunity in the population vs. the product of survival probabilities at all active states if the machine were not immune; this is the tradeoff between the terms indexed by ϵ and $(1-\epsilon)$ in the conditional distribution in Equation 5.5. The immune rate and the survival probabilities in Equation 5.5 are global parameters and not machine-specific, thus the conditional distribution of a machine's immunity vs. susceptibility can be calculated for both candidate machines and current machines. Candidate immunity status and death dates can be sampled in proportion to this likelihood ratio, independent of the immunity status of the current machine or machines; alternatively, a mapping between the candidate(s) and the current machine(s) can be implemented using a probability table suitably renormalized for merges vs. splits.

States and rate parameters : Order of operations for updating $\boldsymbol{\eta}$ vs. θ is not as evident from the format of $\alpha_{1\to 2}$ as for the rest of the parameter vector. Once a candidate path $\boldsymbol{\eta}$ is sampled, the proposal distributions for transition parameters and birth/death/immunity status can be designed to minimize impact on the acceptance ratio. But equation 5.12 demonstrates that the acceptance ratio is highly sensitive to the overall hit rate λ_t at each t, especially when $y_{t,12}$ is large. Mixing of the merge-split steps in the chain is dependent mainly on finding a good method for proposing states, and on controlling the deviation of the merged vs. split hit rates λ_t from the total counts. A preliminary implementation explores splitting of the state at time t conditional on the observed total count $y_{t,12}$ and on the lag admitted at time t by the split steps performed at times $0,\ldots,t-1$ for each candidate machine, with updating of θ following conditional on the sampled states.

States $\boldsymbol{\eta}$: It is reasonable to implement a proposal distribution for states that preserves off to (off,off) state transitions and vice versa. A state $\eta_{t,12}=o$ cannot be split into anything except $\eta_{t,1}=o, \eta_{t,2}=o$. Two states $\eta_{t,1}=o, \eta_{t,2}=o$ cannot be merged into anything except $\eta_{t,12}=o$. Conversely, a state $\eta\neq o$

cannot be split into two off states $\eta_{t,1} = o, \eta_{t,2} = o$, even if $y_{t,12} = 0$, and for a merge, if either $\eta_{t,1} \neq o$ or $\eta_{t,2} \neq o$, then $\eta_{t,12} \neq o$ even if $y_{t,12} = 0$. This requirement preserves shared runs of off states, and ensures that every nonzero count $y_t > 0$ is associated with at least one underlying state that admits an overall scan rate $\lambda_t > 0$. It also ensures that runs of activity in split candidate machines will be at most equal to the runs of activity in the merged machine.

It is also reasonable to preserve dual spike states $\eta_{t,1} = s, \eta_{t,2} = s$ mapping to a spike state $\eta_{t,12} = s$ for a merge step, although the split step need not map a spike $\eta_{t,12} = s$ to two spike states in the candidate machines; a split should have the flexibility to propose machines with non-overlapping activity profiles.

A simple merge-split step for states, shown in Table 5.1 deterministically maps $\eta_{t,1}$ and $\eta_{t,2}$ to the maximum state (ordered by $o < d < s$) for a merge of two states and is uniform among possibilities for a split of one state into the possible components that could have generated it. This simple mapping ignores information from observed counts and from previously updated hours; in practice, this leads to arbitrary assignment of on/off states between split step candidates and to large state-dependent discrepancies between the hit rates $\lambda_{t,1} + \lambda_{t,2}$ and $\lambda_{t,12}$ in spite of attempts to keep θ_{12} close to θ_1 and θ_2.

$\eta_{t,1}$	$\eta_{t,2}$	$\eta_{t,12}$
o	o	o
o	d	d
o	s	s
d	o	d
d	d	d
d	s	s
s	o	s
s	d	s
s	s	s

Table 5.1: A simple merge-split step for states, depending only on the value of η_t for the current machines. Uniform selection among split steps gives one choice for an off state, three choices for a decay state, and five choices for a spike state. However, a uniform selection among split states for this deterministic merge generally leads to low acceptance rates. A more careful probabilistic selection of split states, for example based on $y_{t,12}$ and the lags accrued from earlier hours, can be used to propose more interpretable candidates.

Non-uniform split probabilities: Keeping the merge step deterministic, the following adaptation can be used to generate candidates for the split states that are more interpretable with respect to the sum of observed counts $y_{t,12}$ and to the transition model. For updating hour t, define $k_{t,1}$ and $k_{t,2}$ as the current lag value

calculated for state $t-1$ if η_{t-1} is on, and -1 otherwise. If $k = \infty$ (indicating a pure baseline state), or if $t = 0$, then set k equal to T for ease of computation. For off states, define $o_{t,1}$ and $o_{t,2}$ as the length of the current consecutive run of off states leading up to $t-1$ for machines 1 and 2 respectively. Finally, let $p_t(y_{12})$ be the empirical CDF value of counts $y_{t,12}$ associated with active states $\eta_{t,12}$ in the current machine (possibly renormalized to bound $p_t(\max[y_{t,12}])$ away from 1.0 and $p_t(\min[y_{t,12}])$ away from 0.0).

The CDF probability $p_t(y_{12})$ is used to choose the (high, high) state, while the lag probabilities are used to determine which machine is switched off (or set to the lower lag value for a selection of (s, d) states from a spike state). Under this strategy, states corresponding to larger observed values $y_{t,12}$ have a higher probability of being assigned to higher state configurations.

For splitting $\eta_{t,12} = d$, choose $\eta_{t,1} = \eta_{t,2} = d$ with probability equal to $p_t(y_{12})$, and with probability $1 - p_t(y_{t,12})$, set one machine to an off state. For splitting $\eta_{t,12} = s$, choose $\eta_{t,1} = \eta_{t,2} = s$ again with probability equal to $p_t(y_{12})$. If this state is rejected, perform another step with probability $p_t(y_{12})$ to choose an ordering of (s, d), and with probability equal to $(1 - p_t(y_{12}))^2$, choose an ordering of (o, s).

If both machines are currently on at time $t-1$, then machine 1 is set to the lower state with probability equal to $\left(\frac{1}{2}\right)^{\frac{k_{t,1}+1}{k_{t,2}+1}}$. This strategy favors turning off or decaying machines that have been active longer and are closer to their respective baselines. If machine M_1 is currently turned off at time $t-1$, and machine M_2 is on, then M_1 is chosen to be turned off at time t with probability equal to $1 - \left(\frac{1}{2}\right)^{o_{t,1}+1}$, and similar for when machine M_2 is off and M_1 is on. This strategy favors turning one machine off in longer progressions and instituting non-overlapping machine activity for middle-to-low valued counts, as opposed to switching between random assignment of off and decay states among machines. If both machines are off at time $t-1$, then machine M_1 is chosen to be switched on with probability equal to $\left(\frac{1}{2}\right)^{\frac{o_{t,1}+1}{o_{t,2}+1}}$. Both machines M_1 and M_2 are set to off states at time t if and only if $\eta_{t,12} = o$.

Rate parameters θ : For a very simple merge-split step, each of q, ω and α can be updated independently, using information from their corresponding value(s) in the current machine(s) and minimal information from states $\boldsymbol{\eta}$. A 3-dimensional dimension mapping step, as outlined in Section 5.2.3, that preserves expected values up to boundary constraints, is implemented for the proposal distributions $g_{1\to2}(q_{12} \mid q_1, q_2)$ and $g_{2\to1}(q_1, q_2 \mid q_{12})$. Let A_1 be the number of states for which machine M_1 is active, A_2 be the number of states for which machine M_2 is active, and $A_{1\cup2}$ be the number of states for which both machines are active.

The proposal generates candidates such that:

$$E\left(\frac{A_1}{A_{1\cup 2}}q_1 + \frac{A_2}{A_{1\cup 2}}q_2\right) = q_{12} \tag{5.13}$$

and vice versa. If the machines completely overlap, this constraint sets $q_1 + q_2 = q_{12}$ in expectation, whereas if they do not overlap, it sets q_{12} as the weighted average of the two split rates in expectation, allowing for some flexibility due to upper and lower bounds. The spike rates ω and decay rates α are generated using scaled Beta values preserving the merged rate as the average of the two split rates. In practice however, this simple method is not robust enough to admit reasonable acceptance ratios in light of Equation 5.12.

Enforcing stricter constraints in θ : Both the candidate states and the aggregated counts $y_{t,12}$ for non-overlapping values can be used to help inform the values of q, ω and α. Infinite or very large lag values yield a sample of counts near to the baseline rate q, while spikes (at lag 0) provide direct information on the spike rate ω. Because large values of y_{12} can inflate small differences in $\lambda_1 + \lambda_2$ vs. λ_{12}, a first strategy for choosing candidates is to enforce the relationship

$$q_{12}\omega_{12} = q_1\omega_1 + q_2\omega_2 \tag{5.14}$$

in the merge-split step, while simultaneously preserving all upper and lower bounds on parameters. This format sets the overall hit rates equal in Equation 5.12 when $\eta_{t,12} = \eta_{t,1} = \eta_{t,2} = s$. Once the baselines q and spike rates ω have been set, candidates for the decay rates α can be proposed with expected value that minimizes Equation 5.12.

A strategy for dimension matching between parameters $(q_{12}, \delta_{q_1}, \delta_{q_2}, \omega_{12}, \delta_\omega)$ and $(\delta_{q_{12}}, q_1, q_2, \omega_1, \omega_2)$ is to generate first the baseline rates conditional on current baseline values and state variables, and then to generate the spike rates conditional on the proposed baseline rates in order to enforce the constraint in Equation 5.14.

The difficulties in interpreting the hourly rate λ_t across the sampling period arise mainly due to the fact that decay states admit many different possible values of λ_t. This is due to the dependence in Equation 5.4 on the lag k and the decay rate α. As a preliminary measure for generating viable merge and split candidates, a

reduced and simplified single host model can be used for suspected NATs, that fixes $\alpha = 0$. As a function of $\boldsymbol{\eta}$ and θ, this reduced model admits only three possible connection rates for a machine: turned off ($\lambda_t = 0$), spiking ($\lambda_t = q\omega$), or baseline ($\lambda_t = q$). This reduced model sacrifices the ability to estimate the correlations in decaying rates after a spike that are observed in sandbox tests and in single machines. But the prior mean values for θ (Table 4.1) suggest that on average, individual machines decay back to very near baseline rates by 4 or 5 hours after spiking. The quick return to baseline is also reflected in the study of the Waledac machines in Chapter 4. For non-informative estimation, using the reduced model has the possibility of causing overestimation in H, as a population with more machines yields more degrees of freedom for fitting the patterns of decay seen in NATs and DHCP regions due to machines spiking patterns after turning on.

5.2.5 Population priors and sampling adjustments

Accounting for $\pi(H)$: At iteration m of the Markov chain, the population size H_m is incremented with an accepted split step and decremented with an accepted merge step. Adjusting the likelihood ratio for the graph requires a term of $\frac{\pi(H_m+1)}{\pi(H_m)}$ for a split, and $\frac{\pi(H_m-1)}{\pi(H_m)}$ for a merge. A negative binomial distribution is used for the prior on the number H of infected machines behind a typical NAT device, as the negative binomial can be interpreted as an over-dispersed Poisson distribution with more flexibility in the relationship between its mean and variance than a standard Poisson distribution. The negative binomial distribution with parameters n and p represents the number of failures which occur in a sequence of Bernoulli trials with success probability p until n successes are reached. But it also arises as a mixture of overdispersed Poisson distributions, each with scale parameter equal to $\frac{1-p}{p}$ and shape parameter equal to n. The mean is equal to $\frac{n(1-p)}{p}$ and the variance is $\frac{n(1-p)}{p^2}$. The ratio of the relevant p.m.f values for a negative binomial variable with parameters n and p are:

$$\begin{aligned} \frac{\pi(H_m+1)}{\pi(H_m)} &= \frac{(H_m+n+1)(1-p)}{H_m+1} \\ \frac{\pi(H_m-1)}{\pi(H_m)} &= \frac{H_m}{(H_m+n-1)(1-p)} \end{aligned}$$

For situations where allocations are non-informative but general network properties may hold information about H (for example, the average number of active IP addresses per hour in a DHCP pool), the prior

mean and variance can be written as functions of the network properties; the information is only used to re-weight the likelihoods for merges and splits, and does not affect the inference for generation and allocation among proposed machines.

Accounting for unequal sampling probabilities: The example of a set of mutually exclusive NAT devices can be modeled as a collection of non-informative network spaces, that possibly may share global parameters such as hourly survival rates and rates of immunity among the population. In such a situation, not all pairs of machines are valid to merge together. A split step requires that only one machine be sampled, but a merge step requires that two machines within the same NAT boundary be sampled. This asymmetrical sampling enters into the detailed balance and acceptance ratio through the probabilities $s_{2\to1}$ and $s_{1\to2}$ in Equation 5.4. Let H_m be the total number of machines in the graph at iteration m, and for any bounded sub-network j, let H_{jm} be the number of machines in that sub-network at iteration m.

Merge step: A candidate M_1 is chosen with probability equal to $\frac{1}{H_m}$. Within the sub-network j to which M_1 belongs, a candidate M_2 is chosen amongst its peers with probability equal to $\frac{1}{H_{jm}-1}$ (excluding M_1 itself). The order of selection and the assignment of identities 1 and 2 to the candidate individuals does not matter, and thus there are two ways to select the pair, each with this probability. For the reverse step, the merged candidate M_{12} is chosen from the updated graph with probability equal to $\frac{1}{H_m-1}$, accounting for the new size of the network due to the merge. Thus the proposal ratio for a merge step must be adjusted by the ratio:

$$\frac{s_{1\to2}}{s_{2\to1}} = \frac{H_m(H_{jm}-1)}{2(H_m-1)}$$

Split step: A candidate M_{12} is chosen with probability equal to $\frac{1}{H_m}$. For the reverse step, the candidate M_1 is chosen from the updated graph with probability equal to $\frac{1}{H_m+1}$. Within the sub-network j to which M_1 belongs, the candidate M_2 is chosen amongst its peers with probability equal to $\frac{1}{H_{jm}+1-1}$ (accounting for the new size of the sub-network due to the merge, and discounting the machine already selected). Again, the order of selection and assignment of identities 1 and 2 is arbitrary, thus there are two ways to select the pair. Thus the proposal ratio for a split step must be adjusted by the

ratio:

$$\frac{s_{2\to 1}}{s_{1\to 2}} = \frac{2H_m}{(H_m+1)H_{jm}}$$

Identifiability of labels in the likelihood: At any iteration m, the graph is equivalent up to permutation of the labels assigned to machines. For a graph with H machines, this yields $H!$ unique permutations of labels $1,\ldots,H$ that represent the same state. However, when merging or splitting machines, the labels of all non-selected machines are held fixed, and the acceptance ratio need only take into account the arbitrary labeling of new machines. In a merge step, there is only one choice of labeling for the new machine, and no multipliers are necessary. In a split step, the likelihood must be adjusted for the fact that the assignment of labels M_1 and M_2 from M_{12} is arbitrary. Barring the spawning of two identical machines (which due to continuous-valued parameters has probability 0 of occurring), the new state of the graph can be reached by splitting M_{12} into candidates arbitrarily labeled M_1 and M_2, or into candidates arbitrarily labeled M_2 and M_1. Thus the proposal distribution $g_{1\to 2}(M_1, M_2 \mid M_{12})$ for the two-machine state for a split is multiplied by 2 for both forward and reverse moves. This multiplicity cancels with the multiplicity of the selection probabilities, resulting in the fact that the basic merge-split step is invariant to the effect of permutations.

5.3 Preliminary analysis

A C++ library was built for performing random sweeps of MCMC steps from Chapter 4, with a few adaptations to the single-machine model (discussed in Section 5.3.6), augmented to update y using the Multinomial model from Section 5.2.1 in order to apply across a fixed population of H machines observable only up to its aggregated counts. The implementation also included methods for updating H by merging and splitting machines using the component progression of $\boldsymbol{\eta} \to (t^{\beta}, t^{\dagger}, \epsilon) \to q \to \omega \to \alpha \to \psi \to \boldsymbol{y}$ suggested by the analysis in Section 5.2.2. Four networks were chosen as case studies. Due to low acceptance rates of merge-split steps, a "within-model" strategy was used for preliminary analysis, with four different underlying populations proposed for each network. MCMC output was analyzed in order to assess the simultaneous effects of the modeling decisions and parameter updating steps implemented for the three main components of population estimation under the non-informative generation and allocation framework:

1. the single-machine malware behavioral model;

2. the user activity transition model;

3. the birth/death/immunity model.

Although designed in separate stages, these three components are not estimated in isolation; the decisions made for one setting can change the posterior interpretability of the others. Understanding and adapting these components in the non-informative setting is also a crucial step to designing informative network models as well.

5.3.1 Data sets

Non-informative MCMC chains were studied for four different aggregated network counts or suspected NAT devices in the collected data. Machine 54 from the Waledac informed data set (Figure 4.22) was selected as an example of a small-scale suspected NAT device likely housing fewer than 10 individual machines with what appears to be relatively stable birth and death rates before and after the March 16th surge. From the sample of $1,091,013$ network blocks observed across the study, the 15 blocks with fewer than 3 active IP addresses and the highest average connection rates per IP per hour were examined. Two net blocks were selected as interesting case studies. To compare the effects of corporate NAT activity to a suspected DHCP pool, one collection of net blocks with suspected re-allocations of machines among touchpoints was also selected for further study, with all IP address information ignored and treated as non-informative. For each of these four networks, the aggregated counts were first treated as a single machine in a directly observable population in order to provide a preliminary network profile. Markov chains were run using the single-host model and estimation details from Section 4.6, assuming a single immune machine was responsible for all observed activity in the network. Chains were run for $100,000$ iterations and thinned by 50.

Network 109143, shown in Figure 5.1, corresponds to a United States IP address in Virginia. This network had similar patterns and structure as Machine 54 but on a much larger scale, with a baseline estimate of approximately 247 connections per hour, which spiked regularly to between 3 and 4 times that number as computers were turned on during the work week (barring Friday April 10th, the Good Friday holiday before Easter). A shift in baseline rates after April 5 may indicate the start of cleanup procedures, but as

with Machine 54, the activity appears relatively stable across the observation window.

Network 58644, shown in Figure 5.2, corresponds to an Indian IP address. This network displayed weekday vs. weekend patterns but had less regularity than the US network. It also had an obvious cleanup pattern that started on approximately the 30-th of March, where the baseline rate appeared to drop from near 300 connections per hour, to fewer than 100. The single-machine model cannot account for this drop as a changing baseline rate, and as a result the MCMC estimates a global baseline of 85 reached only toward the end of the collection window. Despite the obvious daily pattern that persists, the model infers that all spike activity occurs in the early days of the data collection, and that the "machine" has a very slow decay rate. Allowing for non-immunity in this case would not help the single-model estimates, as all activity is posited to come from a single machine that has observable infection activity (albeit decreasing) across the entire observation period.

The final network studied is an aggregate of 128 $/24$ networks belonging to a Ukrainian Telecom company, described in Figure 3.2 in Section 3.2. The network displayed three different and distinct IP address patterns during the observation period, which suggests that IP addresses were not reliably linked one-to-one with machines. A non-informative network model ignores all information from IP addresses and uses only the connection attempts per hour as the indication of machine activity. In contrast to the suspected NAT addresses in Figures 5.1 and 5.2, the aggregated activity across the Ukrainian $/17$, shown in Figure 5.3, appeared to have more sustained "spikes" during daylight hours that dropped off overnight, irrespective of workdays vs. weekends. The single-host model estimates sustained spike activity as more machines come online, with a baseline rate corresponding to the nightly lull and a very sharp decay rate (0.47, as opposed to 0.89 and 0.86 for the suspected NAT devices).

Despite the irregularities that are apparent in these aggregated networks, the MCMC chains for continuous parameters generally appear stable. However, large systematic deviations in the empirical predictive plots indicate that while the estimates may converge, the single-host model fits poorly. The models also stray wildly from the informative priors for Poisson rate parameters based on sandbox testing. For example, using the Gamma$(1.5, 3)$ prior from Section 4.5.2, the prior density of the baseline rate $q = 247$ seen for Network 109143 is equal to approximately 6×10^{-36}.

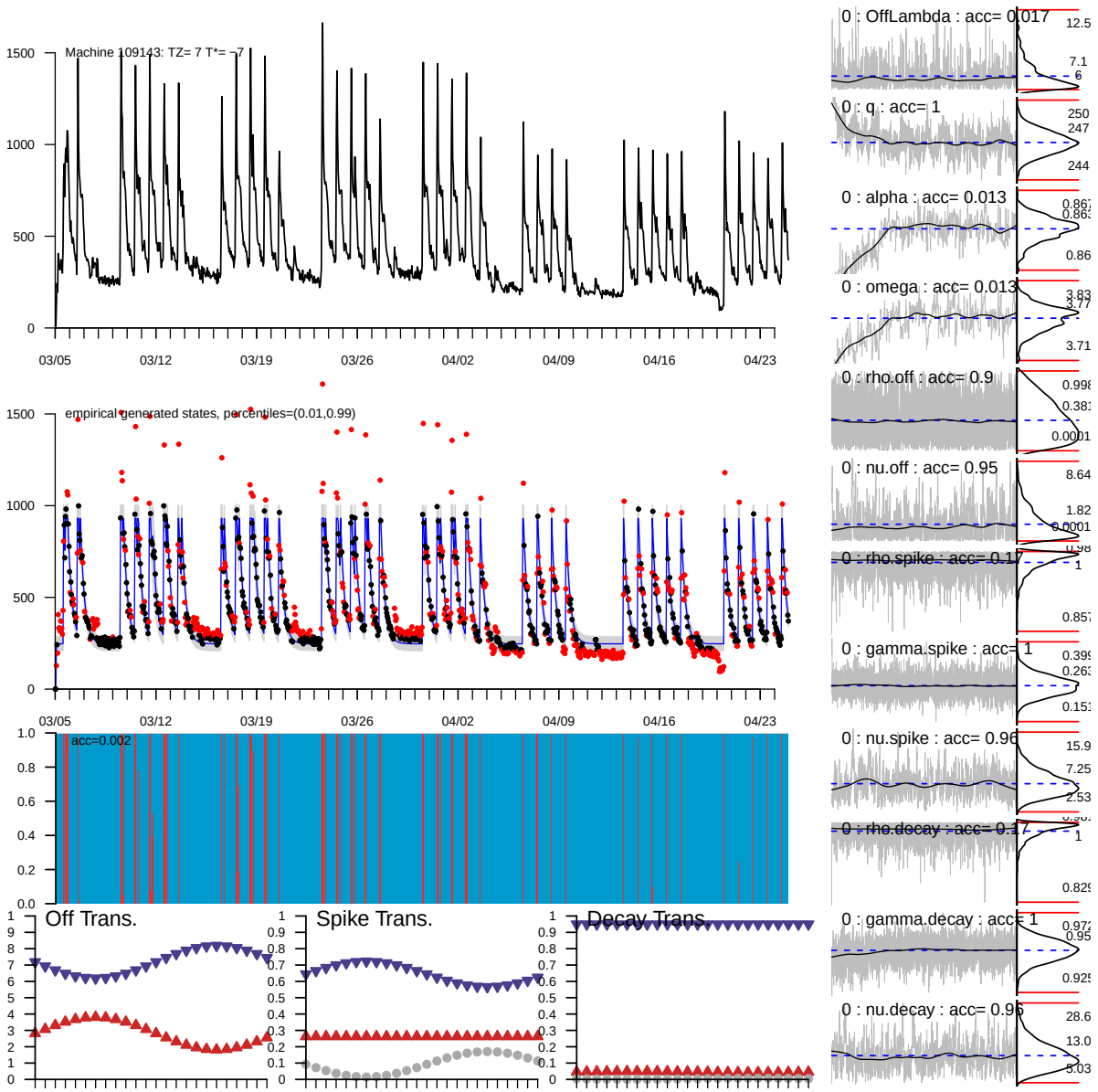

Figure 5.1: A complete dashboard visualization for Network 109143 (one IP address in the United States). This network follows a very strong workday vs. weekend pattern that suggests it belongs to a single corporate entity.

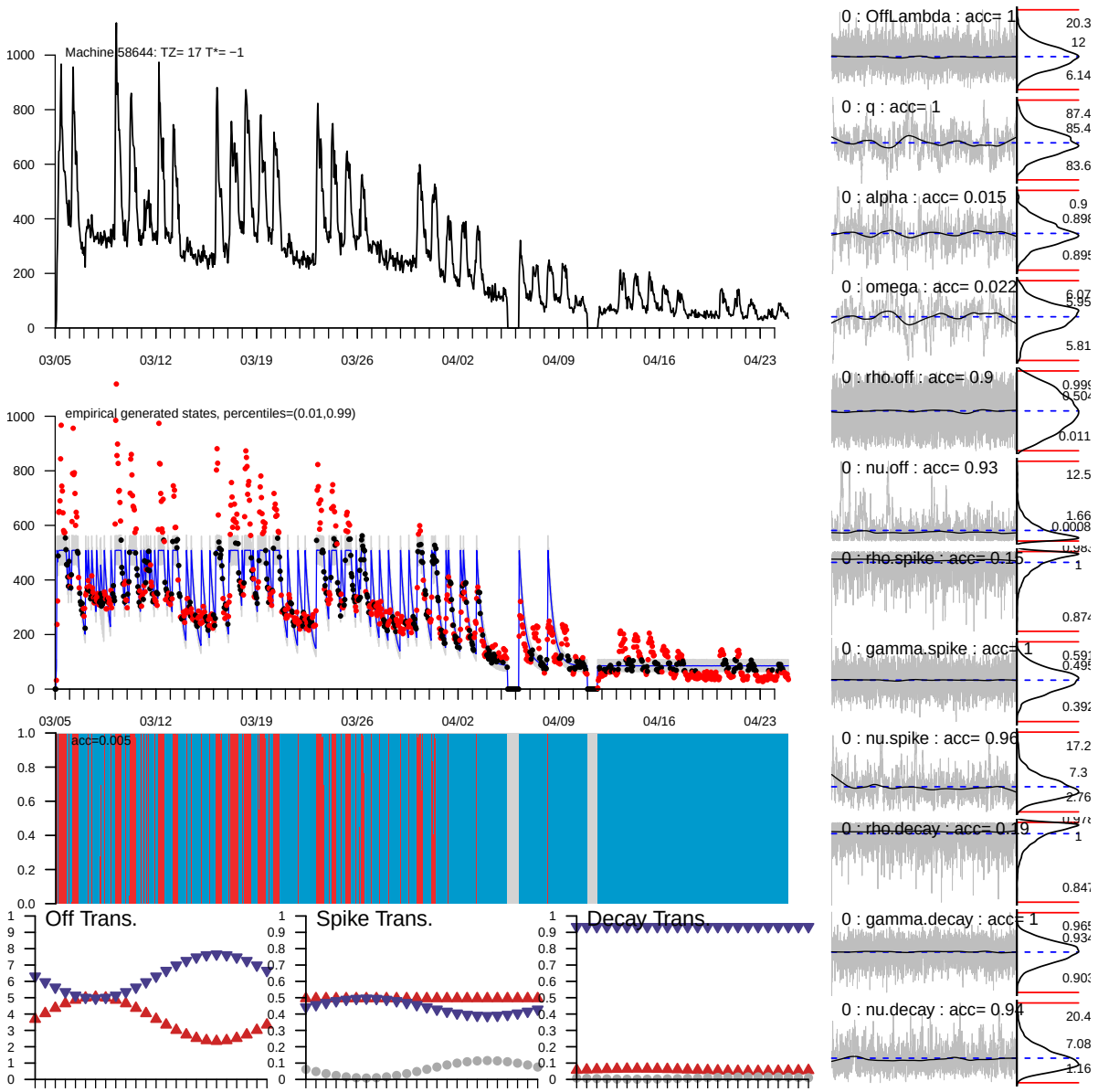

Figure 5.2: A complete dashboard visualization for Network 58644 (one IP address in India). This network follows some daily patterns but also appears to be subject to significant cleanup during the month of April.

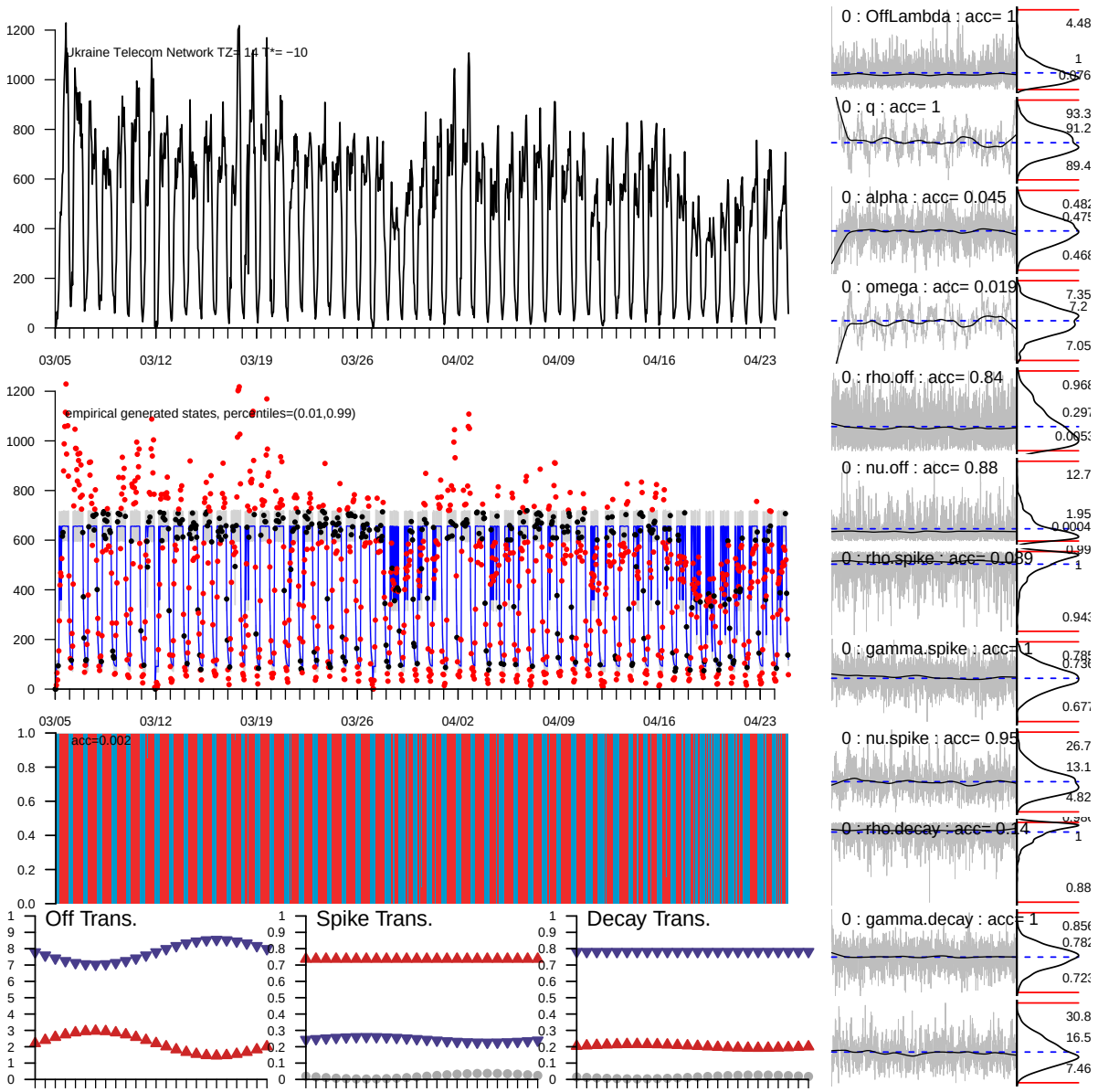

Figure 5.3: A complete dashboard visualization for aggregated counts (ignoring all IP address information) for the Ukrainian /17 network described in Figure 3.2 in Section 3.2. As opposed to the single-IP suspected NAT networks, the aggregated suspected DHCP region shows more sustained "spikes" corresponding to widespread activity across both weekdays and weekends, with drops due to nightly lulls irrespective of weekends vs. weekdays.

5.3.2 Non-informative MCMC estimation details

Implementation: An object-oriented implementation of the non-informative generation and allocation model, including individual machine updating, global parameter updating, count rearranging steps, and non-informative merge-split updating, was built using C++ and the GNU scientific library. The model was designed using several different kinds of C++ classes, including objects representing Machines and Networks as well as their associated parameters, counts, and allocations. A virtual Parameter Updater class was implemented, which acted on Machines and contained derived classes with the relevant prior and likelihood information for updating each of the parameters in $(\psi, \theta, \boldsymbol{\eta}, t^{\beta}, t^{\dagger}, \epsilon)$. Global Parameter objects were included that maintained the proper sufficient statistics necessary for efficiently updating survival and immunity rates as the Machine states changed within the graph. Network parameters included a boundary ID which allowed for modeling of a population of mutually exclusive NATs or network spaces that nonetheless may share global immunity and survival rates across boundaries. The top-down approach facilitates treating the mathematical form of the generation model $f(y)$ from Equation 5.2 as a module that can be adapted for different models of Conficker-C behavior, new user activity models, or other behavioral count models, without redesigning the mechanisms for manipulating the structure of the graph due to MCMC updating steps. At the top layer, the graph mechanisms rely only on Machines and Networks. All virtual Parameter Updater classes rely on the parameter and data object classes contained in Machines and Networks, without relying directly on the structure of their content; only derived classes describe the specific conditional distribution functions that access parameters and counts.

The implementation also included a Count Rearranger object, which acted on a Network object and its linked Machine objects in order to perform both the Multinomial Gibbs step updating y_{it} for fixed H, and the assignment of counts among M_1 and M_2 in the Split step of the merge-split updating. Implementation included efficient copy constructors for both Machines and Networks, as well as insertion and deletion functions, and efficient methods for randomly sampling objects from the graph. Merge-Split Updaters were also implemented as a virtual class acting on a set of connected components (both Machines and their associated Network touchpoints for the eventual incorporation of informative allocation) with derived subclasses for the progression of component steps outlined in Section 5.2.4, although the initial component steps that were implemented were too simplistic to admit reasonable acceptance ratios (see Section 5.3.5).

Fixed population study: For a preliminary analysis of the effect of multiple machines on active size estimates and machine-level parameter estimates behind a NAT, several different fixed population sizes were run for each of the four networks chosen for study. Fixed sample sizes were set at the ratio of the posterior mean $\tilde{q}_y$ estimated under the single machine model, to the prior mean $\tilde{q}$ from the restricted Gamma distribution, as well as the ratio of the posterior spike state rate $\tilde{q}_y\tilde{\omega}_y$ to the prior mean $\tilde{q}$, and two other proposed sample sizes along the range between these extremes.

The ratio of the baseline posterior mean to the baseline prior mean can be interpreted as a ballpark estimate of the population, assuming that the baseline represents saturated activity from all infected hosts, and that the spike patterns observed in the data are due solely to these machines' spike and decay patterns. The ratio of the spike state posterior mean to the baseline prior mean can be interpreted as a ballpark estimate of the population, assuming that the observed spike patterns arise due to an influx of machines turning on at these times and beaconing at a common baseline rate. The truth of the underlying machine activity for a NAT or DHCP region is likely some combination of these extremes. The lower bound of the footprint is informed by the distribution of simultaneous activity at low-level hours such as weekends and holidays. The upper bound for the footprint depends not only on simultaneous activity at high-level hours, but also on the user activity model that informs the extent of "churn" and overlap in activity among the underlying users.

To initialize the graph for the one-to-many allocations of fixed size H described by Table 5.2, a single Network object was created containing the aggregated observed counts x_t for the NAT or DHCP region of study, as well as the estimated offset parameter t^* for the transition model. A total of H Machine objects were created, each linked to the single Network object, with offset value t^* fixed and shared among all constituents. For each hour t, counts were assigned to each Machine object using a Multinomial$(N = x_t, \pi_1 = \cdots = \pi_H = \frac{1}{H})$ distribution. Afterward, initial parameters for each Machine were set using the method S_3 outlined in Section 4.6.2 and Table 4.6.2.

Move selection and output: For updating the graph, a random sweep strategy was used that chose one of three steps with equal probability at each iteration:

1. Select a machine at random; perform one update step for (ψ, θ), one update step for each of 15 states at randomly chosen times between $(t^\beta, t^\dagger)$, and 10 update steps for parameters $(t^\beta, t^\dagger, \epsilon)$;

Network	Sample	H	Iterations	Burn-in	M	M_h	M_y	$\bar{M}_t$
Machine 54	$\tilde{q}_y/\tilde{q}$	5	374968	299974	767	5000	306	2.0
	Mid. 1	8	599922	479937	1223	5000	490	3.4
	Mid. 2	12	899927	719941	1786	5000	735	5.0
	$\tilde{q}_y\tilde{\omega}_y/\tilde{q}$	17	1274989	1019991	2608	5000	1042	7.3
Network 109143	$\tilde{q}_y\tilde{q}$	55	1462458	1169966	2875	1806	1195	8.1
	Mid. 1	100	931977	745881	1880	628	761	4.9
	Mid. 2	150	658367	526693	1357	295	538	3.6
	$\tilde{q}_y\tilde{\omega}_y\tilde{q}$	207	495513	396410	996	160	404	2.8
Network 58644	$\tilde{q}_y/\tilde{q}$	19	1999996	1599996	3897	7407	1634	10.9
	Mid. 1	50	1628504	1302803	3180	2171	1330	8.8
	Mid. 2	83	1155761	924608	2269	928	944	6.2
	$\tilde{q}_y\tilde{\omega}_y/\tilde{q}$	113	896979	717583	1832	529	733	5.3
Ukrainian Telecom Network	$\tilde{q}_y/\tilde{q}$	20	999814	799851	1973	3333	817	5.9
	Mid. 1	65	1393067	1114453	2731	1429	1138	7.5
	Mid. 2	110	905904	724723	1840	549	740	4.9
	$\tilde{q}_y\tilde{\omega}_y/\tilde{q}$	146	706250	565000	1498	322	577	3.8

Table 5.2: Fixed population sizes, MCMC iterations and burn-in used for four different MCMC runs for each of the four aggregated networks in the preliminary non-informative network study. Each iteration was one realization of a random sweep between three different move types. All output was thinned by approximately 1 in 100 iterations, leading to the total number of iterations M sampled for estimating simultaneous active size counts and machine parameters in each model. The value M_h is the approximate number of updates performed on each machine h in the chain after burn-in, and M_y is the approximate number of count rearrangement steps performed in the chain per hour after burn-in. The value $\bar{M}_t$ is the average number of counts sampled for each hour in the chain, used for summarizing individual hourly machine traces.

2. Perform one updating step for each global survival and immunity rate;

3. Select 150 hours t at random and rearrange counts y_{it} among all machines at each of the selected hours.

Each Network was treated as a separate graph, with no sharing of information among survival and immunity parameters. Markov Chains were run concurrently for 12 hours on a MacBook Pro personal laptop, resulting in between $300,000$ and 2 million iterations per graph, with fewer iterations run for larger graphs due to the increase in computational complexity for searching the machine space, and the increased memory and Input/Output details required for those graphs. Chains were directed to produce output at each selected iteration with probability 0.01. As output, the MCMC chain logged the current iteration number, the current state of (ψ, θ) for each machine, as well as aggregated counts of the number of active (non-

off) machines per hour and the number of infections at each hour. Rearranged counts were also logged at a thinned rate of approximately 1 per 100 instances of the count rearranging step. As output, the count rearranger step logged the current iteration number, the Network and boundary ID selected (always the sole aggregated Network object from the graph in this case), the hour t chosen, the aggregated count x_t, and the machine ID number, mean value and count total sampled using the multinomial distribution for each machine at that step. From that output, posterior estimation of parametric hit rates λ_{it} per hour and counts y_{it} per hour could be calculated for each machine i.

Posterior means calculated from single-model output were used to summarize (ψ, θ) for each machine in the population. Output from the count rearranging steps was used to summarize λ_t and y_t by hour to display a profile of activity for each machine. Posterior means of the number of active machines per hour were used to generate active size estimates under the assumption of a fixed population size of H machines.

Machine summaries were examined in two different ways: by fixed machine ID, and by baseline order. All posited machines behind the suspected NAT or DHCP region are interchangeable up to ID, which results in the *label-switching* problem as described by Stephens (2000). Because labels are arbitrary among machines, a perfectly mixing chain will eventually lead to multi-modal posterior distributions of parameters for unique IDs in a heterogeneous underlying machine population. These posterior distributions can be interpreted jointly among continuous parameters and states, but the posterior mean becomes less interpretable as a summary for an individual parameter. To combat evidence that this problem was arising at least for the rate parameters θ, machines $1, \cdots, H$ were re-labeled at each iteration according to the order statistics $q_{(1)}, \cdots, q_{(H)}$ for their baseline rates. The posterior means calculated from MCMC output then summarize the parameters and activity profile of the machine with the smallest baseline at each iteration, through to the machine with the largest baseline at each iteration. This does not protect against multi-modal distributions in other parameters, but it can at least help to provide some interpretable identification for heterogeneous machine populations. All estimates discarded approximately the first 80% of iterations of the chain as burn-in. The reason for using such a long burn-in rate is twofold: firstly, each iteration updated only small subsets of model parameters in very large spaces, thus likely increasing the amount of iterations needed for the chain to converge. Secondly, using a smaller percentage of iterations can be interpreted as a snapshot of a chain that may still be subject to categorical shifts in estimates due label-switching or multiple modes. Shorter chains

can be more easily interpreted as representing one of several local and interpretable modes, with posterior means summarizing these local modes for the preliminary analysis.

The non-informative model used several adaptations to the to the model from Section 4.6, as suggested in the Chapter 4 discussion. These changes included instituting a lower bound for the baseline rate q, updating the survival and immunity models to be time dependent, and jointly proposing a new off state mean λ_o for state updating steps that can split or merge runs of off states. Section 5.3.6 describes the details.

5.3.3 Fixed population parameters

Preliminary analysis of the MCMC output for the non-informative model indicated several broad trends across all networks in the case study:

1. As the population size increased, machine-specific parameter estimates tended to favor a population with a few outliers with larger baseline rates, and most other machines with small baseline rates and comparatively large spike rates;

2. As the population size increased, more machines were shown to be turned off during lulls in activity on the aggregated network; however, in most cases the majority of machines were modeled as active at any given hour.

3. Model traces showed little evidence of long periods of inactivity, and no evidence of clean-up or non-immunity among postulated machines.

4. Flexibility in the decay rate parameter α enabled the transfer of large systematic problems for the single-machine model on the aggregate network to the postulated machines with the largest baseline rates.

Tables 5.3, 5.4, 5.5, and 5.6 summarize the population posterior means for the estimated populations for each of the four networks studied. Baseline-ordered posterior means are summarized in each table; for each network and sample size, both ordered and unordered posterior means for q and ω were similar. Baseline rates and spike rates were generally negatively correlated, leading to similar overall hit rates at spike states among postulated machines. MCMC traces for rate parameters still showed some evidence of multi-modality, attributed to small sample sizes and poorer mixing. Figures 5.4 and 5.5 show examples for

the baseline, spike rate, and overall hit rate at spike state for the machines with the largest baseline rates q in the populations for the large NAT behind Network 109143 and the DHCP region for the Ukrainian Telecom company.

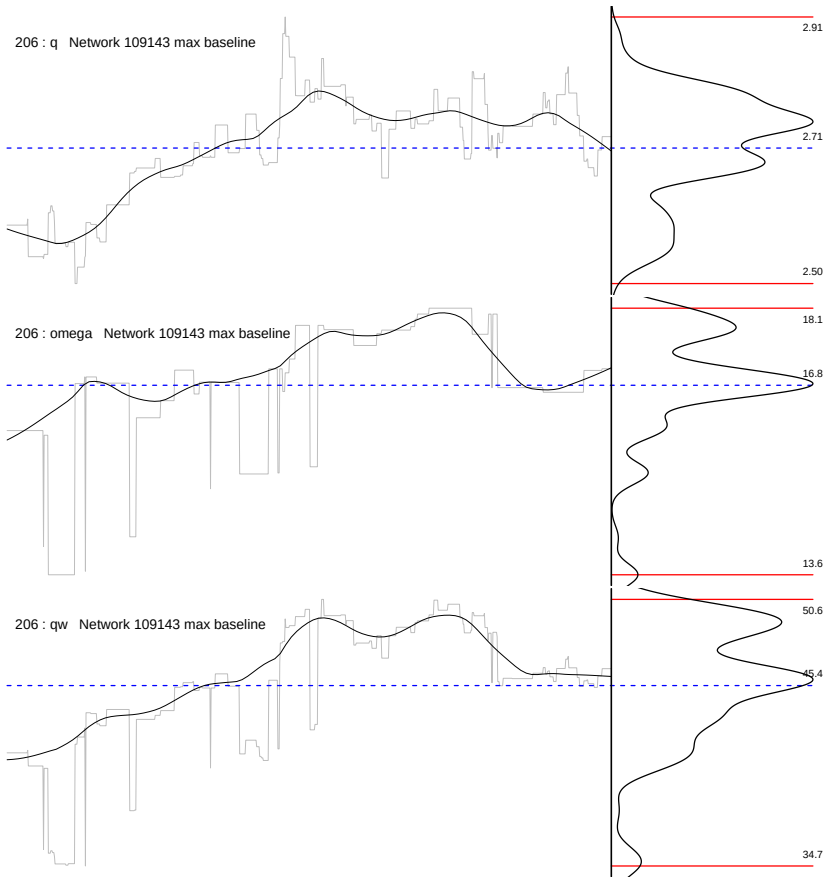

Figure 5.4: MCMC sparkline plots for the baseline rate q, spike rate ω and their product, for the machine with the largest ordered baseline in the model for Network 109143 with 207 machines.

In general, the populations were modeled as collections of relatively homogenous machines, although ordering among the baseline rates q did impose some structure. The models favored populations of persistently active machines with very low baseline rates and comparatively high spike rates for the NAT networks, with only a few machines with higher baseline rates. The apparent sinking of q toward the lower bound for a majority of machines in the population does not agree with the informed priors from sandbox testing; however, the directly observable Waledac data set also showed similar tendencies, arising at least partially from the complexities in the relationship between rate parameters θ and states $\boldsymbol{\eta}$. For the non-informative populations, the *a posteriori* inference for large fixed populations is that activity at lulls in the network is

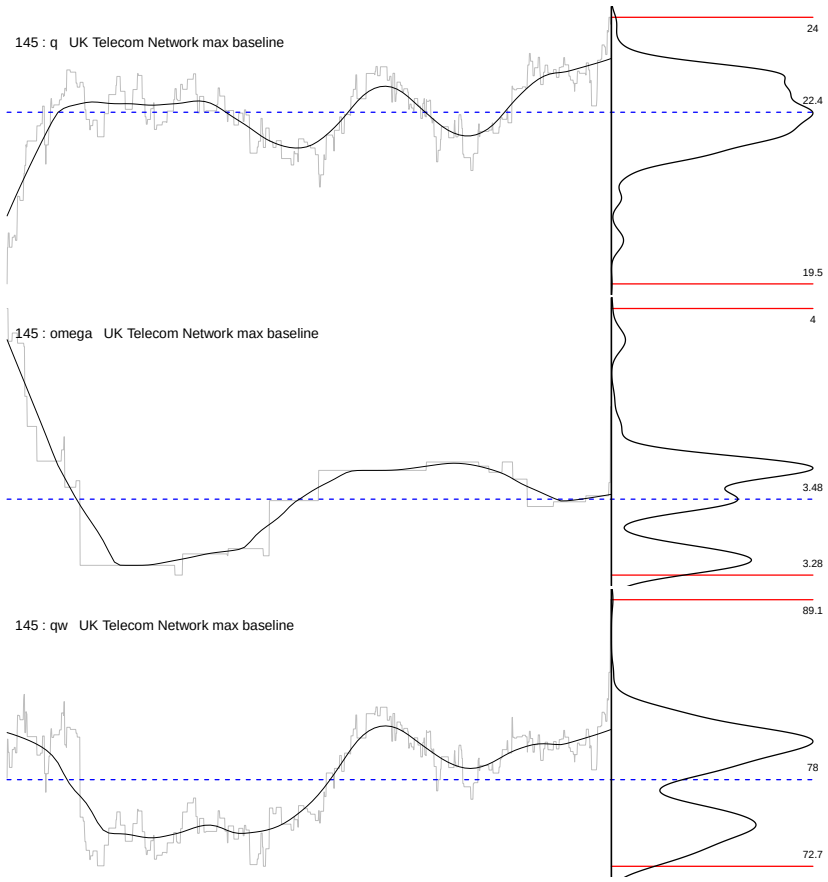

Figure 5.5: MCMC sparkline plots for the baseline rate q, spike rate ω and their product, for the machine with the largest ordered baseline in the model for the Ukrainian Telecom DHCP region with 146 machines.

comprised mainly of a few "heavy hitters" with large baselines, and many more ephemeral machines with very low rates at these times; that nonetheless spike dramatically during peak activity. A more robust model for active vs. inactive states may help the interpretation of this model. The baseline and spike rates for the large population of ephemeral machines could settle closer to the prior modes if these machines could be modeled as turned off during lulls in activity, as opposed to spiking dramatically from a low-baseline active state as the workday begins.

The average time spent consecutively turned off increased as population size H increased for all networks, but even for the largest populations modeled, no machines appeared to be modeled as persistently turned off, and most populations were modeled as having a majority of machines turned on in any given hour. This result is likely partly due to the focus on a 24-hour periodic transition model with a generally low prior for the parameter λ_o, but it is also likely that the updating steps for η_t may not mix well enough

Waledac Machine 54									Correlation			
H	Par.	Min.	Q1	Med.	Mean	Q3	Max.	Avg. σ	q	ω	α	λ_o
5	q	1.80	2.26	3.62	4.05	5.99	6.60	0.300	1.00	-0.87	0.10	-0.97
	ω	3.87	4.07	8.48	8.71	9.19	17.95	1.900	-0.87	1.00	-0.13	0.77
	α	0.38	0.63	0.67	0.63	0.70	0.76	0.057	0.10	-0.13	1.00	0.08
	λ_o	1.77	1.87	3.76	3.76	5.36	6.02	0.650	-0.97	0.77	0.08	1.00
8	q	1.05	1.17	1.36	2.00	2.34	4.33	0.180	1.00	-0.90	-0.72	-0.81
	ω	5.83	8.37	12.70	12.07	15.74	17.56	2.800	-0.90	1.00	0.38	0.56
	α	0.58	0.64	0.67	0.66	0.68	0.75	0.110	-0.72	0.38	1.00	0.79
	λ_o	3.67	4.08	4.70	4.46	4.81	4.92	3.500	-0.81	0.56	0.79	1.00
12	q	1.02	1.08	1.20	1.72	1.53	5.01	0.130	1.00	-0.91	-0.27	-0.28
	ω	3.69	12.05	15.35	13.21	16.29	16.63	2.700	-0.91	1.00	0.51	-0.04
	α	0.49	0.58	0.58	0.58	0.59	0.60	0.094	-0.27	0.51	1.00	-0.02
	λ_o	3.83	4.56	5.18	5.24	5.79	6.70	2.500	-0.28	-0.04	-0.02	1.00
17	q	1.01	1.04	1.09	1.21	1.24	2.00	0.057	1.00	-1.00	0.72	0.93
	ω	9.66	14.55	15.64	14.90	16.18	16.37	3.000	-1.00	1.00	-0.71	-0.94
	α	0.54	0.54	0.55	0.55	0.55	0.58	0.150	0.72	-0.71	1.00	0.62
	λ_o	5.38	6.16	6.92	7.90	8.84	13.00	4.700	0.93	-0.94	0.62	1.00

Table 5.3: Summaries of population means for Off State transition and Poisson rate parameters for four populations posited for Machine 54 from the Waledac informed singleton set. Machines were identified in the MCMC chain by ordering of the baseline rate q.

to explore local modes that include long runs of off-to-off state transitions. The method for setting initial states and counts y_t among the proposed machines favored initializations that did not generally include long strings of inactive states. Randomly updating η_t one state at a time, even with the proposal of a new mean λ_o simultaneously, did not appear to mix well enough to encourage configurations with large runs of off-to-off transitions.

Locality in the off-to-off state transitions also affected the model for clean-up and immunity. In the model, a machine is more likely to be considered a candidate for clean-up if its current configuration includes a long string of inactive states at the end of the observation window. Despite expanding the survival model to include hour-by-hour survival rates, the survival trend (see section 5.3.6) was informed by the trends across all observed networks, and was not yet flexible enough to adapt to the pattern of decay observed in Network 58644. In the current configurations of the single-machine model, the user activity trends, and the survival model, the *a posteriori* conclusions from modeling indicate that populations in the range between the active size at lulls in the network, and the active size at peaks, are stable and active throughout the period

Network 109143									Correlation			
H	Par.	Min.	Q1	Med.	Mean	Q3	Max.	Avg. σ	q	ω	α	λ_o
55	q	1.02	1.38	1.91	3.61	3.81	21.40	0.220	1.00	-0.86	0.90	0.49
	ω	3.21	12.74	17.12	15.19	18.60	19.92	3.600	-0.86	1.00	-0.76	-0.44
	α	0.39	0.46	0.47	0.51	0.51	0.94	0.180	0.90	-0.76	1.00	0.54
	λ_o	0.16	1.04	1.28	1.35	1.56	2.85	1.800	0.49	-0.44	0.54	1.00
100	q	1.00	1.18	1.46	2.00	2.00	11.62	0.081	1.00	-0.86	0.88	0.71
	ω	8.70	16.30	16.60	16.00	17.10	17.90	4.800	-0.86	1.00	-0.91	-0.55
	α	0.34	0.40	0.41	0.46	0.44	0.94	0.160	0.88	-0.91	1.00	0.71
	λ_o	0.97	1.55	1.68	1.79	1.85	4.69	2.400	0.71	-0.55	0.71	1.00
150	q	1.00	1.10	1.25	1.47	1.57	6.22	0.031	1.00	-0.59	0.92	0.26
	ω	6.94	16.03	16.44	16.23	16.72	17.81	5.500	-0.59	1.00	-0.62	-0.30
	α	0.40	0.43	0.45	0.47	0.47	0.96	0.190	0.92	-0.62	1.00	0.41
	λ_o	0.42	1.65	1.86	1.88	1.98	4.72	2.200	0.26	-0.30	0.41	1.00
207	q	1.00	1.03	1.08	1.15	1.17	2.71	0.011	1.00	0.58	0.83	-0.39
	ω	14.60	15.50	15.90	16.30	16.70	22.50	5.900	0.58	1.00	0.40	-0.47
	α	0.33	0.42	0.42	0.44	0.43	0.91	0.180	0.83	0.40	1.00	-0.27
	λ_o	0.63	1.32	1.69	1.94	2.35	4.55	9.300	-0.39	-0.47	-0.27	1.00

Table 5.4: Summaries of population means for Off State transition and Poisson rate parameters for four populations posited for the US network (109143) located at $63.96.204.0/24$. Machines were identified in the MCMC chain by ordering of the baseline rate q.

of observation.

5.3.4 Activity profiles for fixed populations

For further diagnostics, population profiles including the average number of active machines per hour, and traces of the mean counts y_{it} assigned to the machines with the top and bottom ordered baseline rates, were also produced for each population. Figures 5.6 and 5.7 show examples for Network 58644. Posterior mean activity counts (all machines not turned off) per hour, averaged across all iterations post burn-in, are plotted in the top left corner, with a lowess curve added to summarize large-scale trends. A scatter-plot of the ordered posterior baseline and spike rates (q and ω) is shown at the top right. Traces of the average count y_t for the top 3 ordered baselines as well as the bottom 2 ordered baselines are plotted to indicate the range of machine activity in the population. Each trace is accompanied by a parametric profile of the spike-to-decay pattern for the machine, using the posterior means for $q_{(i)}$, $\omega_{(i)}$, and $\alpha_{(i)}$ for the i-ordered machine.

Because output from the count rearranger did not include ranked baselines (only the means λ_{it}), the

Network 58644									Correlation			
H	Par.	Min.	Q1	Med.	Mean	Q3	Max.	Avg. σ	q	ω	α	λ_o
19	q	1.28	1.69	2.24	3.02	3.06	10.86	0.220	1.00	-0.93	0.76	-0.87
	ω	7.48	20.34	20.72	19.79	21.83	22.49	2.000	-0.93	1.00	-0.64	0.68
	α	0.37	0.42	0.48	0.54	0.56	0.95	0.150	0.76	-0.64	1.00	-0.74
	λ_o	12.50	14.90	15.50	15.20	15.80	16.50	2.600	-0.87	0.68	-0.74	1.00
50	q	1.00	1.04	1.11	1.28	1.30	2.76	0.031	1.00	0.89	0.79	-0.74
	ω	20.50	20.60	20.70	21.10	21.20	23.40	2.400	0.89	1.00	0.87	-0.64
	α	0.55	0.56	0.59	0.61	0.61	0.82	0.170	0.79	0.87	1.00	-0.41
	λ_o	12.60	16.20	16.30	16.10	16.40	17.40	3.100	-0.74	-0.64	-0.41	1.00
83	q	1.00	1.02	1.04	1.08	1.09	1.71	0.012	1.00	0.96	0.98	0.72
	ω	17.50	17.90	18.20	18.60	18.80	23.30	5.500	0.96	1.00	0.95	0.75
	α	0.54	0.55	0.56	0.59	0.59	0.95	0.170	0.98	0.95	1.00	0.61
	λ_o	12.10	12.40	12.60	12.80	13.10	14.60	3.600	0.72	0.75	0.61	1.00
113	q	1.00	1.01	1.02	1.04	1.04	1.92	0.006	1.00	0.81	0.95	0.79
	ω	14.80	15.30	15.80	16.10	16.40	23.60	7.800	0.81	1.00	0.92	0.93
	α	0.49	0.51	0.53	0.54	0.54	0.98	0.180	0.95	0.92	1.00	0.90
	λ_o	9.44	9.88	10.09	10.20	10.34	13.02	3.300	0.79	0.93	0.90	1.00

Table 5.5: Summaries of population means for Off State transition and Poisson rate parameters for four populations posited for the Indian network (58644) located at $59.145.136.0/24$. Machines were identified in the MCMC chain by ordering of the baseline rate q.

count traces are calculated using machine ID order, with IDs sorted by the highest to lowest posterior mean. This makes it more difficult to discern persistent periods of inactivity for the ordered machines in the trace plots. However, despite this change in identifiability, traces for the extrema of ID-ordered mean counts appear relatively similar to the profiles described by their respective ordered posterior means.

Figure 5.6 shows the activity profile for a population of 50 machines postulated for Network 58644. The proposed population size is too small to admit significant downtime for any of the 50 proposed machines, and the declines are explained by a decrease in spike activity for the population. Despite the inclusion of survival and clean-up components in the model, posterior inference attributes the decline in observed counts to a small set of machines with high decay rates that dramatically decline at the onset of decline in the observed network counts. Figure 5.7 shows the activity profile for the same network with a population of 113 machines. Activity is still modeled as saturated among all machines leading up to April 5th, but activity is shown to drop to levels similar to the levels seen for the 50-machine population during the course of the declining trend in the observed aggregated counts for the network. Again, the decrease in activity among

Ukrainian Telecom Network									Correlation			
H	Par.	Min.	Q1	Med.	Mean	Q3	Max.	Avg. σ	q	ω	α	λ_o
20	q	1.83	2.85	6.79	6.81	10.09	13.85	0.360	1.00	-0.93	-0.31	-0.60
	ω	5.01	5.34	8.01	11.80	19.61	23.26	1.100	-0.93	1.00	0.10	0.62
	α	0.13	0.16	0.17	0.18	0.21	0.26	0.053	-0.31	0.10	1.00	0.21
	λ_o	1.07	1.30	1.99	2.06	2.34	3.59	0.830	-0.60	0.62	0.21	1.00
65	q	1.00	1.02	1.05	2.62	2.52	13.42	0.110	1.00	-0.96	0.42	0.45
	ω	5.94	20.67	22.48	19.72	22.63	23.37	1.400	-0.96	1.00	-0.24	-0.48
	α	0.16	0.23	0.24	0.24	0.24	0.35	0.064	0.42	-0.24	1.00	0.11
	λ_o	0.87	0.98	1.11	1.65	2.28	3.86	1.000	0.45	-0.48	0.11	1.00
110	q	1.00	1.01	1.03	1.74	1.12	20.15	0.059	1.00	-0.95	0.48	0.02
	ω	4.37	22.52	22.63	22.15	22.86	23.79	1.400	-0.95	1.00	-0.37	0.23
	α	0.24	0.26	0.27	0.29	0.27	0.47	0.066	0.48	-0.37	1.00	0.51
	λ_o	1.58	2.17	2.33	2.57	2.68	4.36	1.500	0.02	0.23	0.51	1.00
146	q	1.00	1.01	1.04	1.67	1.36	22.41	0.041	1.00	-0.93	0.49	-0.41
	ω	3.48	22.79	22.93	22.58	23.19	23.58	1.000	-0.93	1.00	-0.29	0.61
	α	0.27	0.30	0.31	0.32	0.32	0.46	0.076	0.49	-0.29	1.00	-0.01
	λ_o	1.51	3.39	3.71	3.69	4.10	4.64	1.300	-0.41	0.61	-0.01	1.00

Table 5.6: Summaries of population means for Off State transition and Poisson rate parameters for four populations posited for the Ukrainian Telecom Network located at $94.179.128.0/17$. Machines were identified in the MCMC chain by ordering of the baseline rate q.

low-baseline machines is attributed partly to a decrease in spiking activity. But the off-to-off state transition means λ_o estimated for this network were the highest of any of the four networks studied, with a mean of 10.2 hours for the largest population size. The value of λ_o estimated for each machine still drops as a function of population size, indicating that the decline in activity is modeled as mixed homogenously among available machines, rather than attributed to a few machines being cleaned up and permanently switched off.

The flexibility of the decay rate α to inflate in order to model large trends is shown in the profile for the machine with the maximum baseline. The decay rate prior is sharply peaked around 0.4; in this example the maximum posterior mean for α is 0.98, leading to a poor interpretation of the most active machine in the underlying population. Similar decay rate inflation is seen for the NAT network 109143 (Figure 5.8). It is interesting to note the difference between the trends for baselines and spike rates in this largest NAT, vs. the same trends in the DHCP region for the Ukrainian Telecom company. The sharp daily drops in observed counts are sufficient to keep posterior decay rates small in the population. However, as opposed to the NAT network that universally estimates low baseline rates among machines as the population grows, the baseline

rates for the DHCP region settle into several large outliers, still with similar overall hit rates at spike states among machines.

MCMC chains were also run for 2 million iterations for the largest population size for each network, with the decay rate α fixed at the prior mean of 0.4. Figures 5.10 and 5.11 shows the results for the large NAT networks 58644 and 109143. Fixing $\alpha = 0.4$ resulted in a slight drop in average active machines per hour as the baseline rates increased toward the prior mode, and in more interpretable patterns of activity among all of the resulting machines in the population.

According to posterior inference for the non-informative model, no machines in the study had long periods of inactivity, and all machines were immune to clean-ups. Though this result could partially be due to population sizes that are still too low to admit any churn or deaths among active machines, it is also likely due to an inadequate survival model and to the difficulties in updating individual machines to include long periods of inactivity. As an exploratory measure, a population of 100 machines was postulated for Waledac Machine 54 and a Markov chain run for 3 million iterations. Assuming all machines are active, this population size is unreasonable for the given network; it is well above both the stable baseline of 20.7 connections per hour for the single-machine model and the spike state rate of 80 connections per hour.

Figure 5.12 shows the resulting active state counts as well as ordered baseline rate and spike rate posterior means for the population. The activity profile follows very closely with the patterns of observed counts for the machine, and for the most part, baseline rates q for the population are close to 1, and spike rates ω range between 1 and 2. The largest baseline machine ($q_{(100)} = 1.62$) is the only machine with a spike rate above 2; the machine has a spike rate of $\omega_{(100)} = 22.04$ and an off-to-off state transition mean $\lambda_o = 25.3$ hours. Transition probabilities for this machine suggest that it tends to turn off regularly and then to spike at regular intervals. The average off-to-off state transition mean for baselines $q_{(1)} \cdots q_{(99)}$ is 6.8 hours. It appears that the model uses the machine associated with $q_{(100)}$ to account for the very large spikes in the observed counts for Waledac Machine 54, with associated churn among smaller machines, but again there is less evidence for long periods of inactivity among the churning machines. This could also be an effect of the lack of spike activity on the weekends; the large value of λ_o suggests a day-to-day pattern of inactivity for this machine.

This population is the first that starts to show minor evidence of non-immunity among machines, with

some posterior probabilities of immunity among machines sinking very slightly below 1.0. However the death dates for non-immune machines (the last non-off state) are on average the last hour of the data collection window. This model may benefit from a reversible jump Add/Delete step, that at a Delete step proposes to remove the machine with the lowest baseline rate from the graph, while rearranging counts among all remaining machines, and at an Add step, proposes to add a machine with a low baseline and activity profile to the graph.

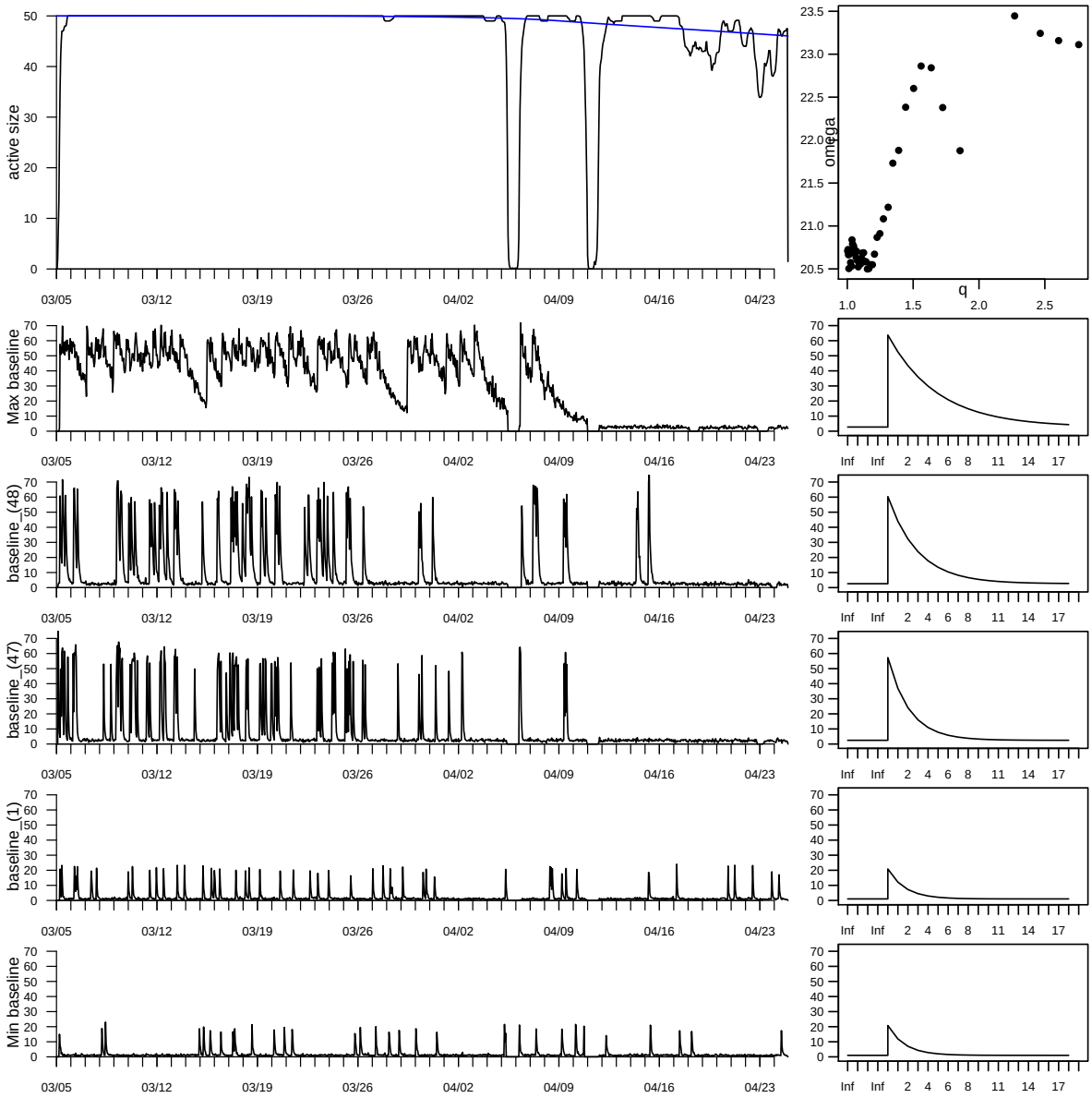

Figure 5.6: Activity profile for a population of 50 machines postulated for Network 58644. The proposed population size is too small to admit significant downtime for any of the 50 proposed machines, and the declines are explained by a decrease in spike activity for the population. Despite the inclusion of survival and clean-up components in the model, posterior inference attributes the decline in observed counts to a small set of machines with high decay rates that dramatically decline at the onset of decline in the observed network counts.

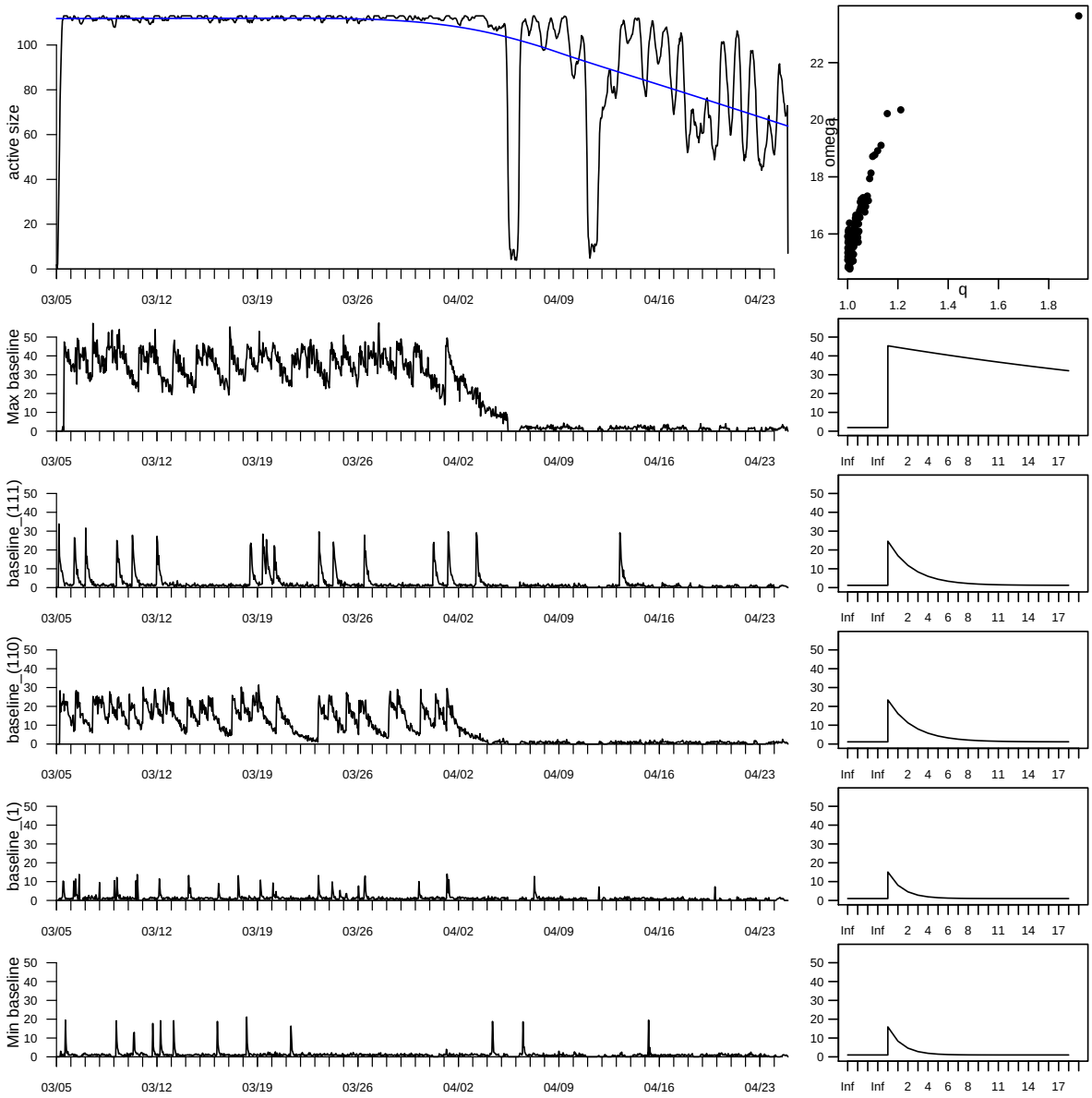

Figure 5.7: Activity profile for a population of 113 machines postulated for Network 58644. Activity is still modeled as saturated among all machines for the pre-cleanup period leading up to April 5th, but activity is shown to drop to levels similar to the levels seen for the 50-machine population during the course of the declining trend in the observed aggregated counts for the network. Again, the decrease in activity among low-baseline machines is attributed to a decrease in spiking activity. The flexibility of the decay rate α to inflate in order to model large trends is shown in the profile for the machine with the maximum baseline. The decay rate prior is sharply peaked around 0.4; in this example the maximum posterior mean for α is 0.98, leading to a poor interpretation of the most active machine in the underlying population.

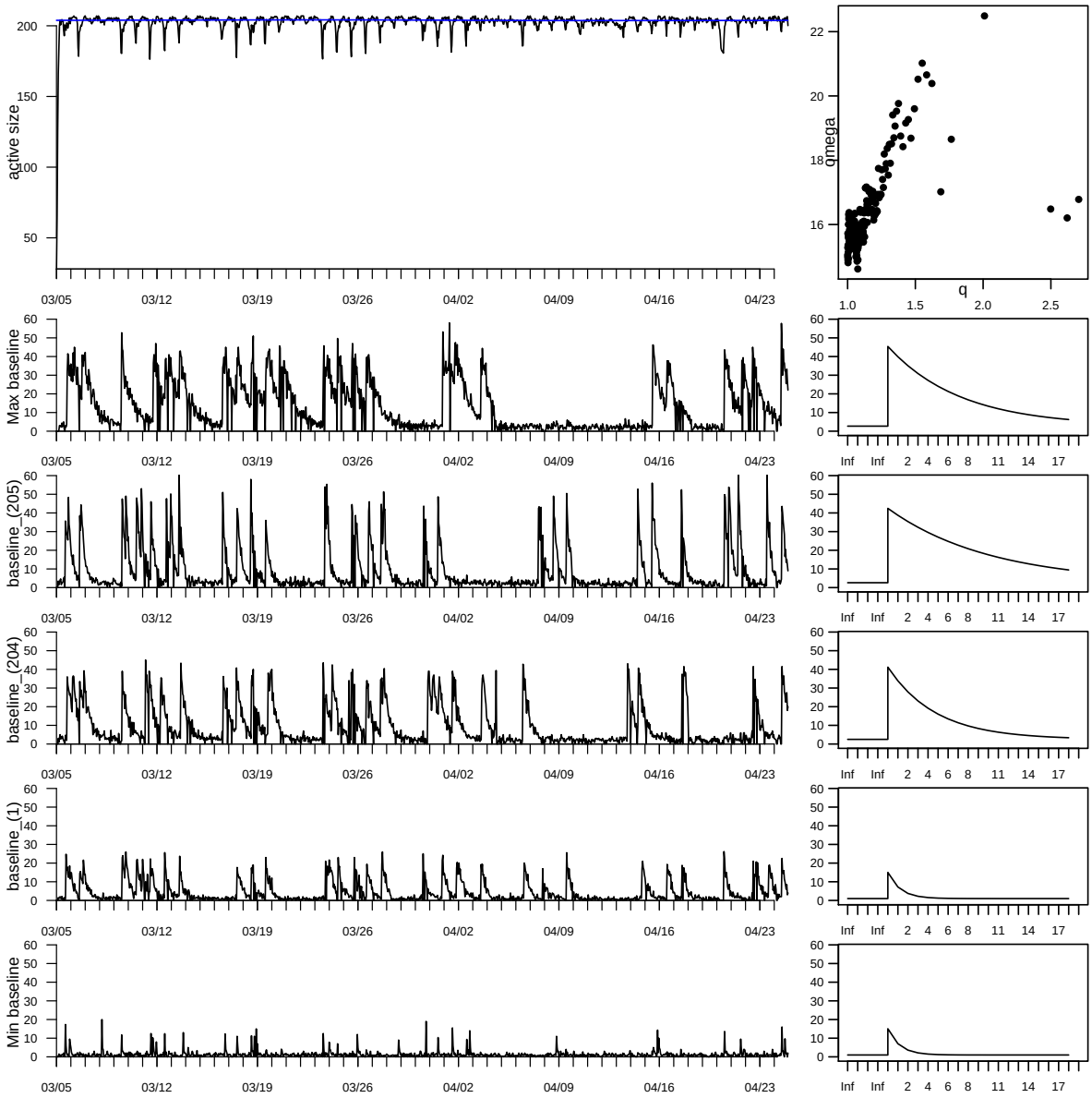

Figure 5.8: Activity profile for a population of 207 machines postulated for Network 109143. Machines with the largest baseline rates q also admit large decay rates α that do not coincide with the priors for single machines informed by sandbox testing. The transition model for user activity also does not appear to adequately explain the spikes as lulls due to machines being turned on which were turned off overnight. Activity patterns instead indicate that machines are almost always turned on, but a few in the population are turned off for only one or two hours prior to being turned on again and subsequently spiking. The average off-to-off state transition mean λ_o for this population was only 1.94 hours.

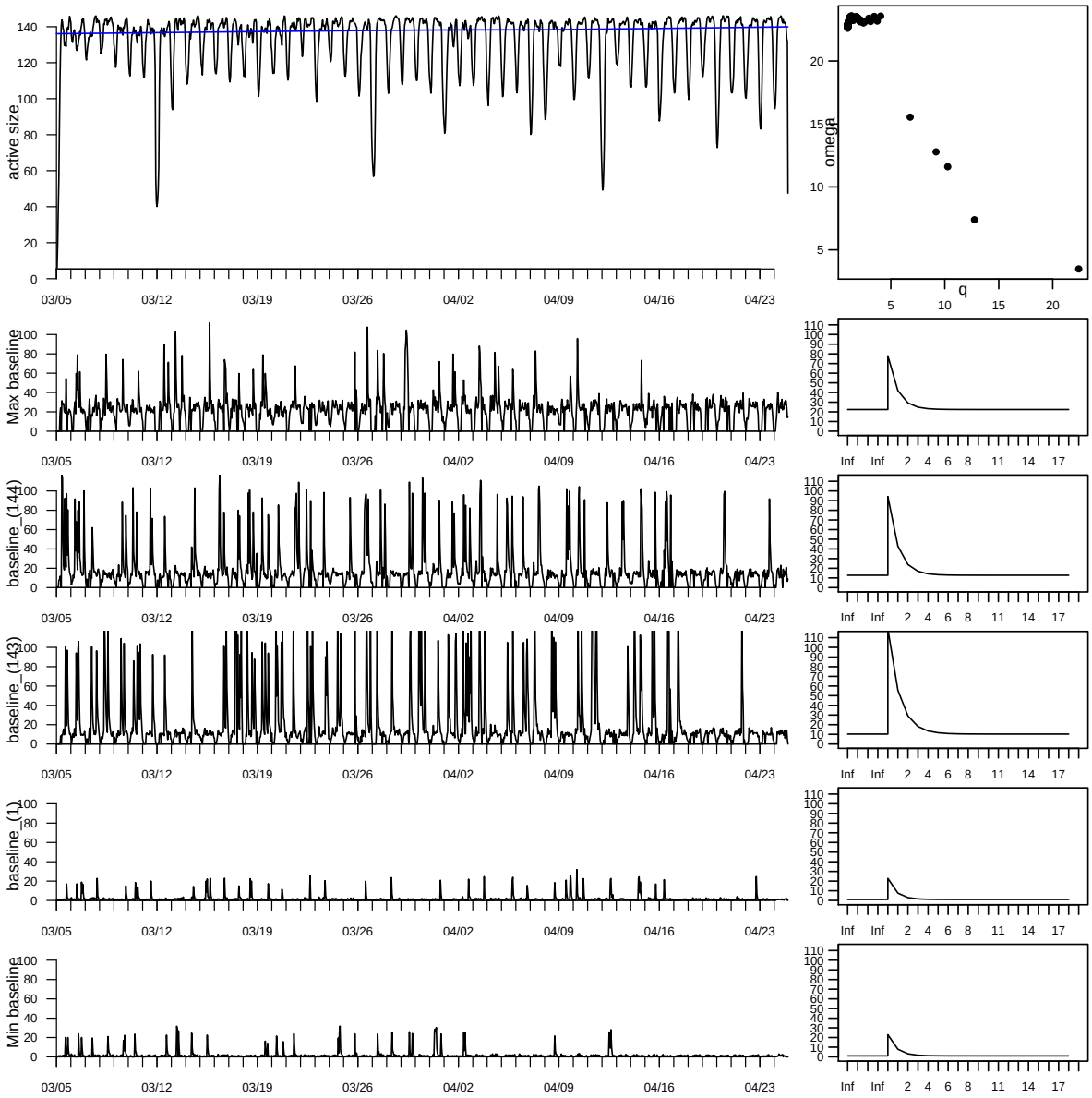

Figure 5.9: Activity profile for a population of 146 machines postulated for the Ukrainian Telecom network DHCP region. Machines appear much more homogeneous in daily activity patterns and overall spike rates than for the NAT networks in the study. Daily drops in activity in the network are transferred to daily turn-offs among machines. Decay rates α are low due to the very steep drops in daily vs. nightly counts.

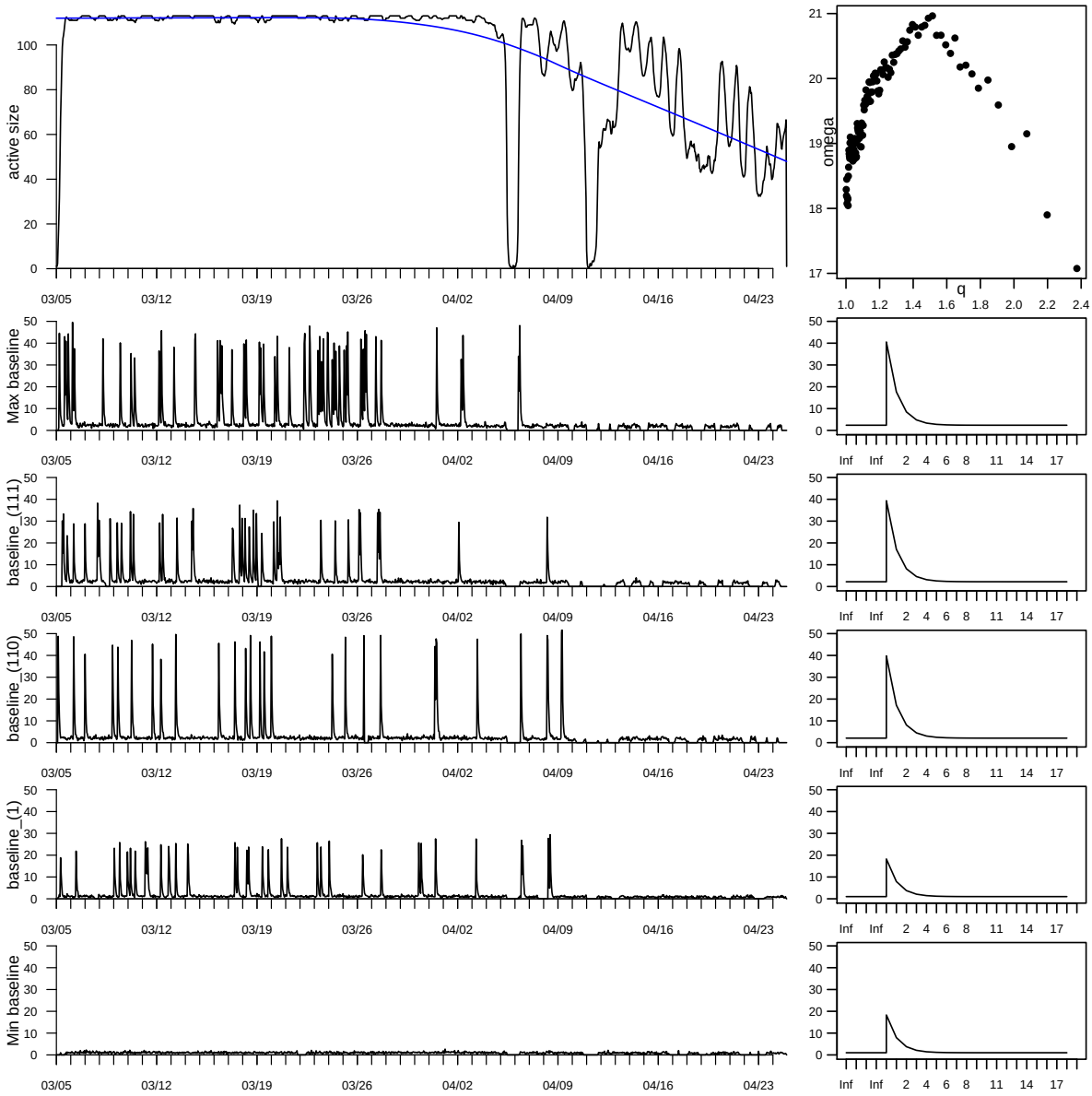

Figure 5.10: Activity profile for 113 machines associated with Network 58644, treating the decay rate α as fixed at the prior mean 0.4. Hourly activity profiles decrease slightly while baseline rates q increase in the population toward the prior mode. Although the decay patterns in the overall network are still attributed partially to fewer spikes among machines, the patterns of counts y_t for the machines with large baseline rates appear more interpretable as single-machine activity.

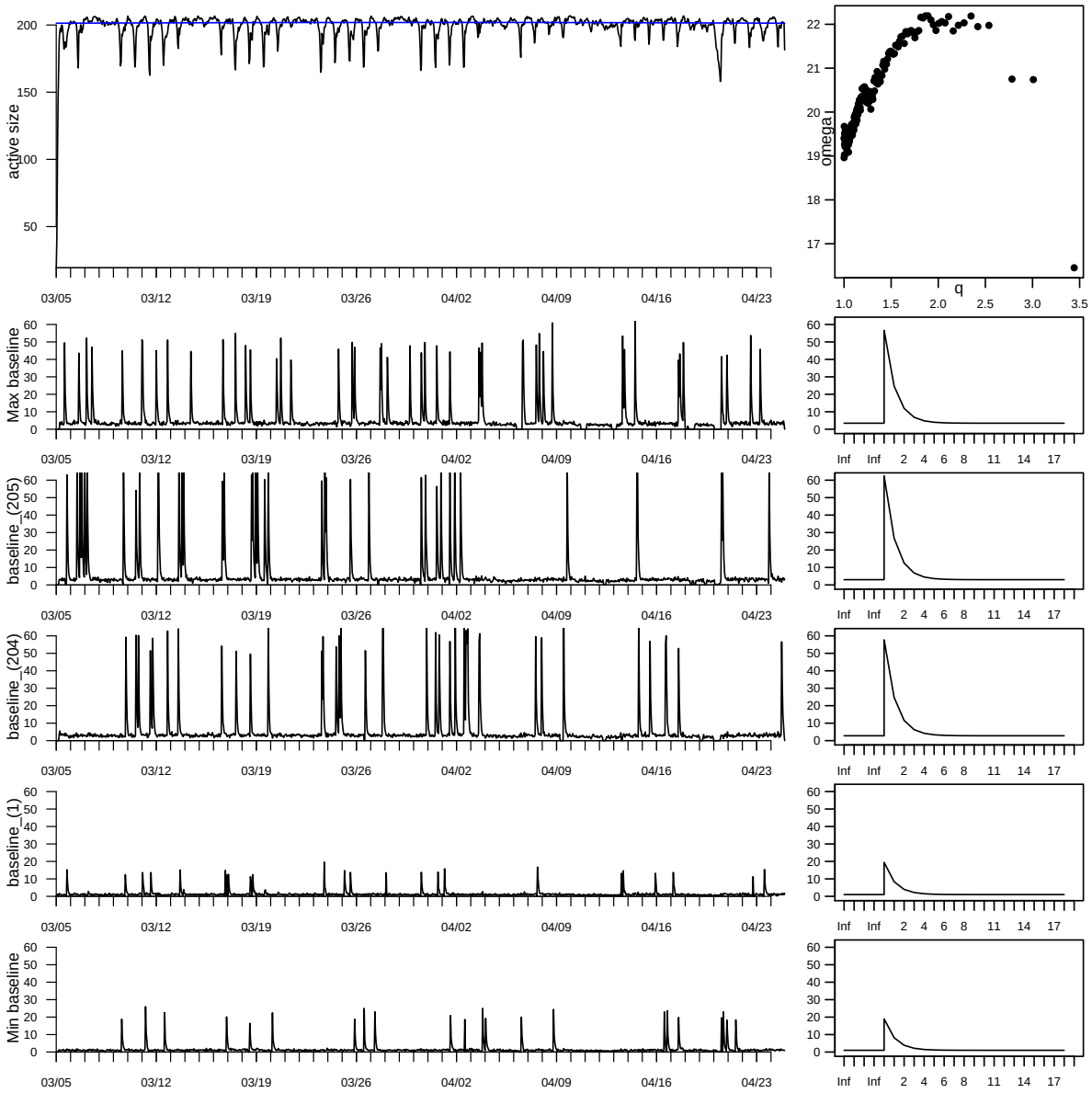

Figure 5.11: Activity profile for 207 machines associated with Network 109143, treating the decay rate α as fixed at the prior mean 0.4. Again, hourly activity profiles decrease as the baseline rates increase among the population, with more interpretable patterns of activity attributed to the constituent machines.

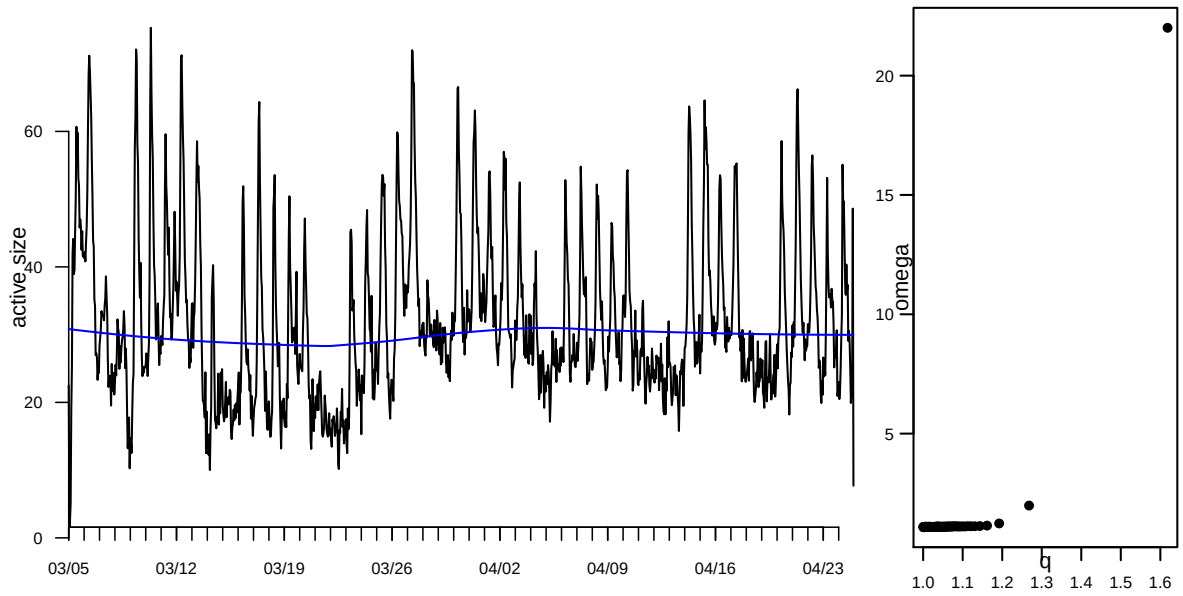

Figure 5.12: Activity count profile for 100 machines associated with Machine 54, treating the decay rate α as fixed at the prior mean 0.4. The largest baseline rate is also associated with the longest posterior off-to-off state transition mean λ_o. The active number of machines follows closely to the patterns in the observed counts for Machine 54, with one machine attributed to a daily pattern of ephemeral large spikes ($\omega = 22.04$, the only spike rate higher than 2.0 in the population).

5.3.5 Output analysis for Merge-Split steps

A preliminary implementation of Merge-Split updating was also included in the C++ graph structure. The order of parameters followed the progression $\boldsymbol{\eta} \to (t^\beta, t^\dagger, \epsilon) \to q \to \omega \to \alpha \to \psi \to \boldsymbol{y}$. States were merged deterministically according to Table 5.1, and were split with non-uniform probabilities outlined in Section 5.2.4 in order to help interpretability. Conditional on the state, the birth, death and immunity updates were implemented using the merge probabilities shown in Table 5.7. Definitions follow Table 4.5.3 in Section 4.5.3: $t_{\cdot\ell}$ is the latest hour for which $\eta_t \neq o$, and $\epsilon = 0$ implies susceptibility to cleanup while $\epsilon = 1$ implies immunity. The table was suitably renormalized for split steps. In split situations where the proposed states $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$ prohibited birth before time 284 for both proposed machines, the corresponding columns are replaced by zeroes in the probability table.

Independent Gibbs steps based solely on $\boldsymbol{\eta}$ were implemented for the parameters λ_o, γ_s and γ_d. The periodic baseline ν_a was merged and split using a 3-dimensional step that mapped $\nu_{12,a}$ to $\frac{1}{2}(\nu_{1,a} + \nu_{2,a})$ in expected value and vice versa. The value of ρ_a was sampled based on $\boldsymbol{\eta}$ alone, using a Beta distribution

centered at the value obtained by measuring the proportion of the relevant state transitions in $\boldsymbol{\eta}$ observed at the apex time $t^* = 6$, weighted by one additional observation centered at the prior mean of 0.5 for ρ. Counts were merged deterministically and split according to the Binomial distribution with $N = y_{t,12}$ and probability equal to the renormalized means $\lambda_{t,1}$ and $\lambda_{t,2}$.

Several different approaches were tried for sampling rate parameters, with little success. The merge-split step for the baseline rate q was implemented using a 3-dimensional step preserving interpretability due to Equation 5.13, adjusting the modes of the scaled Beta proposal distribution when this value was out of range for the value of the parameter(s) in the current machine(s). The spike rate ω and decay rate α were updated using the same simple 3-dimensional step as the periodic baseline ν_a. Alternatively, the decay rate α was set to 0, and baseline rates and spike rates were proposed using simple method of moments estimators applied to Equation 5.12. Overall, the goal was to sample rate parameters close to the mode of the complete conditional distribution given $\boldsymbol{\eta}$, but this is still a heuristic design in the trial-and-error phase.

Tight sample sizes ($\sigma = 600$) and looser sample sizes ($\sigma = 100$) were attempted for the scaled Beta proposal distributions for all three-dimensional merge-split steps. These peaked distributions were chosen as an attempt to implement near equivalence as in Green's two-dimensional case, without having to account for one-to-one mappings across the changes of scale that arose in the interpretable transformations. Furthermore, tight sampling bounds cause difficulty with the reverse move as the chain settles in to its steady state; if the parameter values at the current iteration are far from the conditional mode, it will reduce the likelihood of acceptance due to inclusion of the reverse step in the numerator of the acceptance ratio.

To aid analysis and design for merging and splitting in the future, the C++ implementation can be set to print the full trace of all component steps for merge-split updaters. A typical example for Machine 54 is shown in Figure 5.13. From an initial population of 8 machines proposed to be responsible for the observed activity, an attempt is made to merge two machines together into one. This is an example of a reduced model in which decay rates α have been set universally to 0. The log-candidate values shown at the end of each proposal line are the ratios $\frac{g_{1\to 2}}{g_{2\to 1}}$ for each component step, multiplied by the prior ratios for ψ and θ, and also by the full component log-likelihood for the conditional Gibbs steps (resulting in a value of 0). The partial sum shown for the counts at each of the 1224 hours is the accumulation of both likelihood and jumping ratio due to Equation 5.12. The likelihood ratios for the remaining transition probabilities

and birth/death/immunity status are added at the end of the step. The likelihood ratios appear healthy for the birth/death/immunity and transition probability components. However this step has a large negative contribution from the proposals for $\boldsymbol{\eta}$ (-1503), ω (-876) and the accumulation of the log-acceptance ratios associated with y_t (-224). The proposal for ω seems to be an effect of the peaked Scaled Beta proposal distribution, and from the trace output it seems like the spike rate should be lower than what is proposed. More troublesome are the contributions from state proposals and the slow accumulation of negative values for the hourly counts. Chains including merge-split updating were run multiple times through $10,000+$ iterations for the four networks in the case study under various starting population sizes and using several different modeling and updating schemes, but produced no acceptances.

		M_{12}					
M_1	M_2	$(0, t_{12,\ell}, 0)$	$(0, T, 0)$	$(0, T, 1)$	$(284, t_{12,\ell}, 0)$	$(284, T, 0)$	$(284, T, 1)$
$(0, t_{1\ell}, 0)$	$(0, t_{2\ell}, 0)$	0.50	0.25	0.25	0	0	0
$(0, t_{1\ell}, 0)$	$(0, T, 0)$	0.33	0.33	0.33	0	0	0
$(0, t_{1\ell}, 0)$	$(0, T, 1)$	0.33	0.33	0.33	0	0	0
$(0, t_{1\ell}, 0)$	$(284, t_{2\ell}, 0)$	0.33	0.33	0.33	0	0	0
$(0, t_{1\ell}, 0)$	$(284, T, 0)$	0.33	0.33	0.33	0	0	0
$(0, t_{1\ell}, 0)$	$(284, T, 1)$	0.33	0.33	0.33	0	0	0
$(0, T, 0)$	$(0, t_{2\ell}, 0)$	0.33	0.33	0.33	0	0	0
$(0, T, 0)$	$(0, T, 0)$	0.25	0.50	0.25	0	0	0
$(0, T, 0)$	$(0, T, 1)$	0.33	0.33	0.33	0	0	0
$(0, T, 0)$	$(284, t_{2\ell}, 0)$	0.33	0.33	0.33	0	0	0
$(0, T, 0)$	$(284, T, 0)$	0.33	0.33	0.33	0	0	0
$(0, T, 0)$	$(284, T, 1)$	0.33	0.33	0.33	0	0	0
$(0, T, 1)$	$(0, t_{2\ell}, 0)$	0.33	0.33	0.33	0	0	0
$(0, T, 1)$	$(0, T, 0)$	0.33	0.33	0.33	0	0	0
$(0, T, 1)$	$(0, T, 1)$	0	0	1.00	0	0	0
$(0, T, 1)$	$(284, t_{2\ell}, 0)$	0.33	0.33	0.33	0	0	0
$(0, T, 1)$	$(284, T, 0)$	0.33	0.33	0.33	0	0	0
$(0, T, 1)$	$(284, T, 1)$	0	0	1.00	0	0	0
$(284, t_{1\ell}, 0)$	$(0, t_{2\ell}, 0)$	0.33	0.33	0.33	0	0	0
$(284, t_{1\ell}, 0)$	$(0, T, 0)$	0.33	0.33	0.33	0	0	0
$(284, t_{1\ell}, 0)$	$(0, T, 1)$	0.33	0.33	0.33	0	0	0
$(284, t_{1\ell}, 0)$	$(284, t_{2\ell}, 0)$	0	0	0	0.50	0.25	0.25
$(284, t_{1\ell}, 0)$	$(284, T, 0)$	0	0	0	0.33	0.33	0.33
$(284, t_{1\ell}, 0)$	$(284, T, 1)$	0	0	0	0.33	0.33	0.33
$(284, T, 0)$	$(0, t_{2\ell}, 0)$	0.33	0.33	0.33	0	0	0
$(284, T, 0)$	$(0, T, 0)$	0.33	0.33	0.33	0	0	0
$(284, T, 0)$	$(0, T, 1)$	0.33	0.33	0.33	0	0	0
$(284, T, 0)$	$(284, t_{2\ell}, 0)$	0	0	0	0.33	0.33	0.33
$(284, T, 0)$	$(284, T, 0)$	0	0	0	0.25	0.50	0.25
$(284, T, 0)$	$(284, T, 1)$	0	0	0	0.33	0.33	0.33
$(284, T, 1)$	$(0, t_{2\ell}, 0)$	0.33	0.33	0.33	0	0	0
$(284, T, 1)$	$(0, T, 0)$	0.33	0.33	0.33	0	0	0
$(284, T, 1)$	$(0, T, 1)$	0	0	1.00	0	0	0
$(284, T, 1)$	$(284, t_{2\ell}, 0)$	0	0	0	0.33	0.33	0.33
$(284, T, 1)$	$(284, T, 0)$	0	0	0	0.33	0.33	0.33
$(284, T, 1)$	$(284, T, 1)$	0	0	0	0	0	1.00

Table 5.7: Merge-split probability table for $(t^{\beta}, t^{\dagger}, \epsilon)$ conditional on $\boldsymbol{\eta}$ and $\boldsymbol{\eta}^*$. Definitions follow Table 4.5.3 in Section 4.5.3: $t.\ell$ is the latest hour for which $\eta_t \neq o$, and $\epsilon = 0$ implies susceptibility to cleanup while $\epsilon = 1$ implies immunity. In split situations where the proposed states $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$ prohibit birth before time 284 for both proposed machines, the corresponding columns are replaced by zeroes in the probability table.

```
::Merge/Split, Repeat 1 times::
478 MergeSplit Prior: Merge Step: current H = 8 Pair boundary = 8 prior and selection log-likelihood = 1.25435

478 MergeSplit_state: Merge Step| current=    7: (4 off,  93 spike, 1127 decay)
                                |            0: (5 off,  81 spike, 1138 decay)
                                | proposed= 168: (4 off, 125 spike, 1095 decay)
                                | log-candidate=-1503.65

478 MergeSplit_Birth/Death/Immune: Merge Step| current= 7: (0 1224 1) 0: (0 1224 1) | proposed= 168: (0 1224 1) | log-candidate=-1.25276
478 MergeSplit_q: Merge Step                 | current= 7: 3.05175    0: 2.96358    | proposed= 168: 5.2103     | log-candidate=16.5496
478 MergeSplit_omega: Merge Step             | current= 7: 6.03021    0: 4.66525    | proposed= 168: 6.39826    | log-candidate=-876.699
478 MergeSplit_OffLambda: Merge Step         | current= 7: 4.05209    0: 0.858733   | proposed= 168: 5.0808     | log-candidate=6.33713
478 MergeSplit_nu.off: Merge Step            | current= 7: 4.47661    0: 5.74651    | proposed= 168: 4.54817    | log-candidate=2.2075
478 MergeSplit_rho.off: Merge Step           | current= 7: 0.29987    0: 0.14101    | proposed= 168: 0.0379439  | log-candidate=0
478 MergeSplit_gamma.spike: Merge Step       | current= 7: 0.147287   0: 0.199031   | proposed= 168: 0.327309   | log-candidate=-5.48677
478 MergeSplit_nu.spike: Merge Step          | current= 7: 4.07468    0: 7.35146    | proposed= 168: 5.77481    | log-candidate=-0.4551
478 MergeSplit_rho.spike: Merge Step         | current= 7: 0.961942   0: 0.999976   | proposed= 168: 0.445725   | log-candidate=0
478 MergeSplit_gamma.decay: Merge Step       | current= 7: 0.949726   0: 0.959397   | proposed= 168: 0.949285   | log-candidate=3.53473
478 MergeSplit_nu.decay: Merge Step          | current= 7: 5.41721    0: 4.3468     | proposed= 168: 5.55861    | log-candidate=1.16802
478 MergeSplit_rho.decay: Merge Step         | current= 7: 0.985267   0: 0.991774   | proposed= 168: 0.85381    | log-candidate=1.21901

   4: Counts 9  <- 6 3  : States 2 <- 2 2 : Rates 5.2103  <- 3.05175 2.96358 : log-likelihood = -0.488037  partial sum  = -0.488037
   5: Counts 7  <- 5 2  : States 2 <- 2 2 : Rates 5.2103  <- 3.05175 2.96358 : log-likelihood = -0.200688  partial sum  = -0.688724
   6: Counts 4  <- 2 2  : States 2 <- 2 2 : Rates 5.2103  <- 3.05175 2.96358 : log-likelihood = 0.230336   partial sum  = -0.458389
   7: Counts 19 <- 4 15 : States 1 <- 2 1 : Rates 33.3368 <- 3.05175 13.8258 : log-likelihood = -3.52639   partial sum  = -3.98477
   8: Counts 14 <- 8 6  : States 2 <- 2 2 : Rates 5.2103  <- 3.05175 2.96358 : log-likelihood = -1.20641   partial sum  = -5.19118
   9: Counts 8  <- 4 4  : States 2 <- 2 2 : Rates 5.2103  <- 3.05175 2.96358 : log-likelihood = -0.344362  partial sum  = -5.53555
[. . . ]
1220: Counts 0  <- 0 0  : States 2 <- 2 2 : Rates 5.2103  <- 3.05175 2.96358 : log-likelihood = 0.805033   partial sum  = -225.203
1221: Counts 6  <- 2 4  : States 2 <- 2 2 : Rates 5.2103  <- 3.05175 2.96358 : log-likelihood = -0.0570133 partial sum  = -225.26
1222: Counts 6  <- 3 3  : States 2 <- 2 2 : Rates 5.2103  <- 3.05175 2.96358 : log-likelihood = -0.0570133 partial sum  = -225.317
1223: Counts 2  <- 1 1  : States 2 <- 2 2 : Rates 5.2103  <- 3.05175 2.96358 : log-likelihood = 0.517684

MergeSplit_Counts: Merge Step| current= 7:(1.10107 Chi-sq, 1220 df) 0:(1.09463 Chi-sq, 1219 df)  | proposed= 168:(0.0339689 Chi-sq 1220 df)
                             | log-likelihood=-224.799

478 Total log-jump ratio = -2581.33: log-jump + prior = -2580.07
478 Off state and immunity log-ratio = -4.21882
478 Transition probability log-ratio = 53.955
478 Full Log-likeihood ratio for proposal vs. current = 49.7361
478 Total log-likelihood ratio for the proposal = -2530.34 : log(alpha) = -1.75177 : rejected
478 Deleted proposed machine 168
```

Figure 5.13: MCMC output trace for a merge step applied to Machine 54. Large negative accumulations in component log-candidate ratios for the state proposals as well as sinking partial log-likelihood values per count assignment per hour show the importance of generating interpretable split states, as well as the sensitivity of the merge-split step to small deviations in the proposed parameters. An apparent mis-step in proposing a spike rate ω results in a low log-candidate value and an overly high merged rate $q_{12}\omega_{12}$ that propagates through the entire run of counts.

5.3.6 Adaptations to the single host model

The non-informative network models used several adaptations to the single host model outlined in chapter 4:

- Instituting a lower bound of 1.0 and an upper bound of 2000 on the baseline rate parameter q.
- Updating the survival probabilities to be time-dependent, with a piecewise linear prior on the hyper-parameters for each hour;
- Changing the State proposal and Birth/Death/Immune proposal steps to include proposing a new off state mean λ_o in conjunction with changes to states with $y_t = 0$ and for the long runs of states added or deleted for the births and deaths.
- Instituting an unrestricted Gamma prior for the off state mean λ_o, to avoid unstable calculations in the likelihood when the MLE of λ_o given the states $\boldsymbol{\eta}$ is sharply peaked below the lower bound in the scaled Beta prior.

Survival rates A piecewise linear prior is used to relate the survival likelihoods for each hour. The prior is constructed to reflect greater susceptibility to detection with the onset of the March 16th surge, and a gradual increase in survival probability as the surge dies down. Figure 5.14 shows the prior with set points at the start and end of the data collection, and at March 16th, 20:00 hours UTC, March 19th, 20:00 hours UTC, and March 30, 04:00 hours UTC. The line at hour t describes the prior mean μ of the survival probability for that hour with a sample size $\nu = 10$ across all hours, using a the parameterization of the Beta distribution characterized by $\alpha = \mu\nu$ and $\beta = (1 - \mu)\nu$. Survival rates are universally high because the penalty accumulates geometrically on an hourly basis. Prior means are set to 0.98 hourly for all hours between the start of the data collection and the March 16th surge. At the start of the surge, the prior sinks linearly to a low of 0.9 occurring three days into the surge. The prior then grows linearly to 0.95 for two weeks following the low point. For the remainder of the data collection window, the prior increases linearly from 0.95 back up to 0.98.

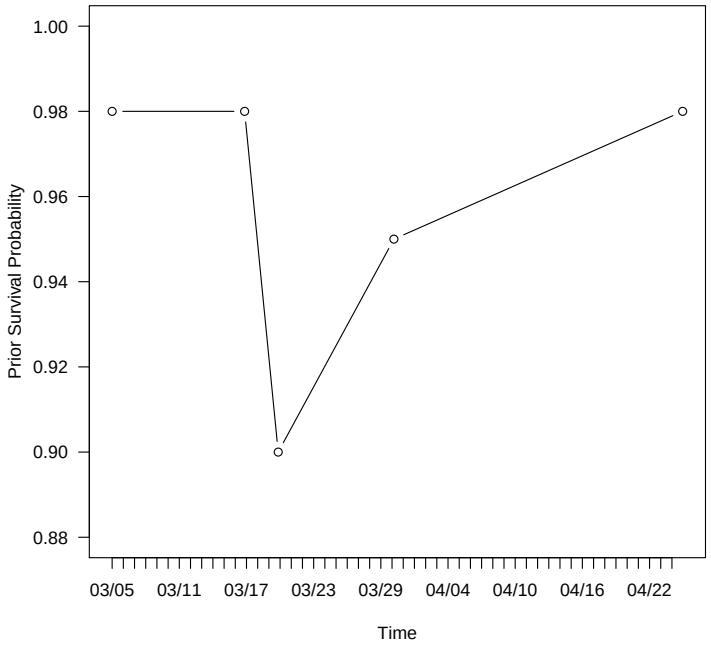

Figure 5.14: A piecewise linear prior for hourly active survival rates.

Joint updating in state proposals Both Birth/Death/Immunity steps and state-changing steps that propose an on-to-off count have the possibility to drastically change the distribution of runs of consecutive off-to-off transitions. In order to aid mixing for updating these discrete parameters, at MCMC step m, they are augmented to include the proposal of a new value for λ_o. The theory for states η_t is described below; the step for the discrete distribution of $(t^\beta, t^\dagger, \epsilon)$ is similar.

A joint proposal distribution $g((\eta_t, \lambda_o) \mid \psi^m, \theta^m, \boldsymbol{\eta}^m, \kappa, \boldsymbol{y})$ is used to generate both a new discrete state and a new value of λ_o, when the state value η_t has the possibility to change the distribution of off states in the data (that is, $y_t = 0$). The proposal distribution is built in sequence:

$$g((\eta^*, \lambda_o^*) \mid \psi^m, \theta^m, \boldsymbol{\eta}^m, \kappa, \boldsymbol{y}) \quad = \quad g_1(\eta^* \mid \psi^m, \theta^m, \boldsymbol{\eta}^m, \kappa, \boldsymbol{y}) g_2(\lambda_o^* \mid \psi^m, \theta^m, \boldsymbol{\eta}^*, \kappa, \boldsymbol{y})$$

Suppose the current state $\eta_t^m = a_0$. Since there are only two possible states for the proposed $\eta_t^* = a_1$, the first proposal can be a simple discrete probability. The generation of a value λ_o^* can be done based on the selected new path variable using the proposal step outlined in Section 4.5.3. When $\pi(\lambda_o) = U(l_{\lambda_o}, u_{\lambda_o})$,

the complete conditional distribution for λ_o given the selected state η^* is a Gamma distribution restricted to the interval $[l_{\lambda_o}, u_{\lambda_o}]$.

Let $[p(t, a; \boldsymbol{\eta}, \theta, \psi, \boldsymbol{\phi}, \epsilon, \boldsymbol{y})]_u^v$ be the conditional path likelihood from Equation 4.11, restricted to the relevant run of states affected by a change at time t. Let $\boldsymbol{\Gamma}(x, \alpha, \beta)$ be the CDF of the Gamma distribution with parameters α and β and mean equal to $\alpha\beta$. Let $O_{\boldsymbol{\eta}}$ be the number of off-to-off state transitions in the path characterized by $\boldsymbol{\eta}$, and let $\left[O_{\boldsymbol{\eta}}\right]_u^v$ be the number restricted to the range $u \leq t \leq v$, as defined in Section 5.2.4. Also as defined in Section 5.2.4, let $R_{\boldsymbol{\eta}}$ be the number of runs of off states observed in the path characterized by $\boldsymbol{\eta}$, and let $\left[R_{\boldsymbol{\eta}}\right]_u^v = |u \leq t \leq v : t \in \mathcal{T}_{o|o}|$ be the number of runs of off states restricted to the interval $[u, v]$. Using this notation, the acceptance ratio for the proposed value (η^*, λ_o^*) becomes

$$
\begin{aligned}
r \;=\; & 1_{\{l_{\lambda_o} < \lambda_o^* < u_{\lambda_o}\}} \frac{g_1(a_0 \mid \psi^m, \theta^m, \boldsymbol{\eta}^*, \kappa, \boldsymbol{y})}{g_1(a_1 \mid \psi^m, \theta^m, \boldsymbol{\eta}^m, \kappa, \boldsymbol{y})} \frac{[p(t, a_1; \boldsymbol{\eta}^*, \theta^m, \psi^*, \boldsymbol{\phi}^m, \epsilon^m, \boldsymbol{y})]_u^v}{[p(t, a_0; \boldsymbol{\eta}^m, \theta^m, \psi^m, \boldsymbol{\phi}^m, \epsilon^m, \boldsymbol{y})]_u^v} \\
& \times \frac{\exp(\lambda_o^* \left[R_{a_1}\right]_u^v)(\lambda_o^*)^{-\left[O_{a_1}\right]_u^v}}{\exp(\lambda_o^m \left[R_{a_0}\right]_u^v)(\lambda_o^m)^{-\left[O_{a_0}\right]_u^v}} \frac{R_{a_0}^{O_{a_0}+1}(O_{a_1})![\boldsymbol{\Gamma}(u_{\lambda_o}, O_{a_1}+1, 1/R_{a_1}) - \boldsymbol{\Gamma}(l_{\lambda_o}, O_{a_1}+1, 1/R_{a_1})]}{R_{a_1}^{O_{a_1}+1}(O_{a_0})![\boldsymbol{\Gamma}(u_{\lambda_o}, O_{a_0}+1, 1/R_{a_0}) - \boldsymbol{\Gamma}(l_{\lambda_o}, O_{a_0}+1, 1/R_{a_0})]}.
\end{aligned}
\tag{5.15}
$$

In practice, although the rejection method can be used to sample from a Gamma distribution restricted to a range $[l_{\lambda_o}, u_{\lambda_o}]$, it becomes inefficient when the likelihood is sharply peaked outside the allowed range; generally this occurs when the maximum likelihood estimate of λ_o given the states $\boldsymbol{\eta}$ lies below the lower bound l_{λ_o}. Finding an efficient covering distribution for the tail of the Gamma is a non-trivial task. When the MLE lies below the lower bound, both CDF terms in the acceptance ratio are close to 0. This leads to computational errors in the MCMC code. If $\pi(\lambda_o) = \Gamma(\alpha_{\lambda_o}, \beta_{\lambda_o})$, unrestricted in its range, the complete conditional distribution for λ_o is a conjugate Gamma distribution, and the CDF terms in the likelihood ratio from equation 5.16 are not needed. To retain the interpretation that off states tend to occur in runs of generally 6 to 8 hours (following traditional weekday traffic patterns), a prior is chosen with shape parameter $\alpha_{\lambda_o} > 1$, which puts comparatively low density on very low mean values. On the other hand, the scale parameter should be large to accommodate days or possibly weeks of inactivity for more ephemeral machines. Figure 5.15 shows the $\Gamma(2, 50)$ distribution (mean = 100 hours) that was used as the prior on λ_o for non-informative network modeling.

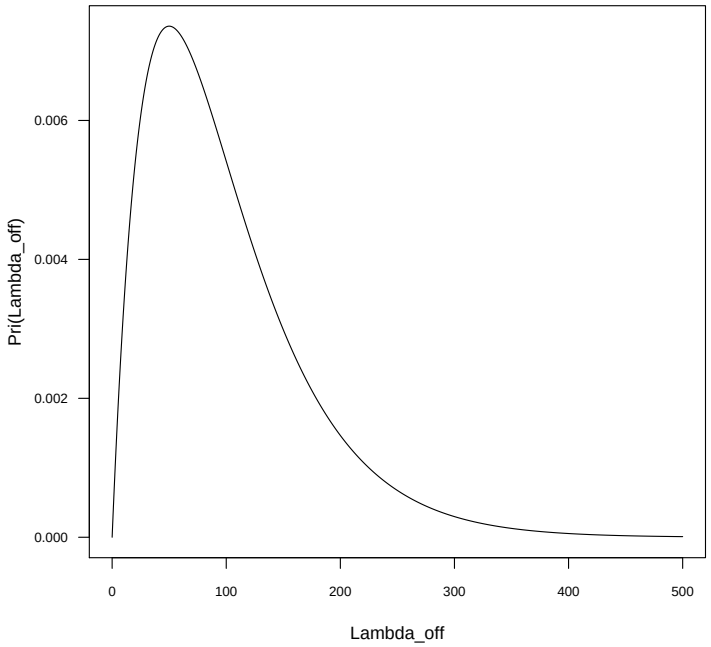

Figure 5.15: Unrestricted Gamma distribution used as the prior $\pi(\lambda_o)$ for non-informative network modeling.

5.3.7 Discussion

This preliminary study of a few interesting non-informative networks is useful for assessing modeling decisions and estimation techniques for future analysis of both Conficker-C specifically and the general generation-allocation framework.

Modeling: The reduced non-informative network model from Equation 5.2 arises from a hidden Markov structure in which the count y_t at time t depends only on the values ξ. For the Conficker-C model, this means that the dependence structure of counts from hour to hour relies only on the parameters ψ and the hit rates λ_t which are functions of the states $\boldsymbol{\eta}$ and Poisson rate parameters θ. Using this structure, the non-informative network model is essentially a collection of convolutions of counts that are conditionally independent given the underlying structure, but not identically distributed.

Prior information available for the likelihood of high and low counts for a single active machine can help to inform the breakdown of counts among active machines at any given time, which can help to place a lower

bound on the footprint population size that underlies a set of observed aggregates. But the footprint estimate itself is sensitive to the transition model that describes the likelihood of long periods of activity or inactivity for single machines. In cases where the populations are open to births and deaths, the footprint estimate is also sensitive to the survival model that generates new infections and clean-ups. If little information is available for these components, or if the functional forms are not chosen well, the posterior estimates of activity for even an informed sandbox generation model can be difficult to interpret.

For large NATs, it may be useful to derive a transition model that accounts for workday vs. weekend activity among users, and to posit a population consisting of several different types of machines. The transition model that was used for Conficker-C accounted only for a 24-hour periodicity, which was not robust enough to capture the fact that many machines did not turn on during the weekends. The transition model can be adapted to incorporate a mixture of persistently active machines (from users who rarely turn their computers off) and ephemeral machines that turn on only for the course of the workday. A hierarchical model that gives some flexibility in estimating the prior proportions of these two machine types may help to make population estimates for completely non-informative corporate NAT environments more interpretable, and keep posterior estimates of single-machine activity in confirmation with information from sandbox tests and directly observable populations.

Large DHCP regions such as the Ukrainian Telecom Network are not strictly non-informative, but it is interesting to study the differences between aggregated activity profiles for such networks versus corporate NATs. The main difference seems to be that the effect of machines turning off and on is spread across wider ranges of time. For this preliminary study, the periodic peak hour t^* was fixed at the weighted average of the values estimated for all $/24$ net blocks within the Telecom network. Relaxing this value and letting it be estimated separately for different machines may help the model account for the smoother shape of spike and decay in less homogenous DHCP regions.

The decay patterns seen in the Indian Network 58644 seemed to be evidence of clean-up behind the NAT network. Although the survival model was informed by broad patterns across all 33 million Conficker-C-infected IP addresses seen during the two-month observation period, posterior estimation for Network 58644 did not attribute the decline to deaths. Including hyperparameters for the piecewise linear survival model can help to make it more interpretable across the different patterns of clean-up that may arise within

different networks. Network administration policies and self-policing can certainly increase or decrease the chances of an infection being noticed and cleaned up versus general immunity of machines. Susceptibility to clean-up may also be more "bursty" for non-immune machines than can be modeled by a geometric survival rate that prohibits any hourly survival rate equal to 1. Thus it may be helpful to model survival probabilities at a higher granularity than hourly machine activity. For example, susceptibility to clean-up can be modeled as a daily risk rather than an hourly risk. The effect would be like a smoother or histogram bin size applied to clean-ups, and may help alleviate some of the difficulties arising from multiplying many hundreds of Beta-distributed geometric decay rates together to obtain survival probabilities.

Finally, though the peaked Beta prior on the decay parameter α was sufficiently constrained to produce interpretable results for the directly observable population from Chapter 4, it too easily increased to absorb large-scale shifts in baselines for aggregate counts in the non-informative setting. This slow decay was propagated across all observed hours for the machines with large baseline rates in the proposed populations, and led to high mean values and subsequent assignment of large counts y_t to these machines. The low prior weight of the large decay rate was not enough to pull the non-informative MCMC estimation scheme away from these configurations. Sandbox testing showed clearly that machines settled into baseline within only a few hours of spiking, and the posterior distribution of α among the directly observable population confirmed this analysis. It is enlightening to allow α some flexibility for treating aggregations as single machines, as it was used for fitting single-machine models to the four suspected NAT and DHCP networks in the non-informative analysis. But in order to maintain interpretability of individual machines as H is changed, the decay rate should either be fixed to an interpretable value, or implemented with a strict upper bound $u_\alpha < 1$.

Estimation: The biggest question that arises from this preliminary study is whether or not a practical method for exploring the posterior distribution of the population size H can be achieved for the generation-allocation model. This can be done using either a between-model approach, that updates H in concurrence with other machines and parameters in a single MCMC, or a within-model approach that treats H as an index to a model family. Section 6.1.2 addresses the latter approach as future work, and this discussion focuses on improving the methods for adding or deleting machines into the population during a single MCMC run.

The evidence for the practicality of a within-model approach to H is still entwined with the question of properly informing and identifying the components of the single-machine model as well as efficiently

exploring the posterior parameter space when H is fixed. Interpretability is gained by fixing the range of possibilities for the decay rate α and adapting the survival and transition models to better reflect the populations studied. After implementing these adaptations, three steps are proposed for aiding estimation:

- Proposing runs of off states associated with 0 counts in updating individual machines
- Including a Metropolis-Hastings version of the count rearranging step that proposes changes of state η_{it} in addition to new arrangements of counts y_{it}.
- Redesign of the Merge-Split step and exploration of some other possibilities for adding or deleting machines.

From the preliminary study, it is apparent that efficient exploration of the posterior space of H machines behind aggregate network touchpoints requires some larger-dimensional updating steps for single machines, and perhaps also for arrangement of counts among machines. The model adaptations for Chapter 5 were also used in further analysis of the Waledac informed singleton set from Chapter 4, with preliminary indications that the new survival model and off-state updating step were admitting non-immune machines and death dates $t^\dagger < T$ among the population. The analysis of the over-large population for Machine 54 suggests that the mechanics of the birth, death, and immunity model were working in the non-informative setting, but that few machines were estimated with long strings of inactive states. To address this, the single-machine model can be adapted to include joint proposals of consecutive strings of states with $y_t = 0$ to runs of decay or off states. Similar to the adaptation proposing a new value of λ_o for a change in a single state η_t associated with a 0 count, for long strings of counts, new values of both λ_o and q can be proposed simultaneously.

Despite using information from all machines at each hour, the Gibbs step for rearranging counts can be considered as a low-dimensional or "short" step among machines. This is especially true for machines at the extreme values for baseline and spike rates; a few machines with large overall rates λ_{it} will tend to dominate the assignment of aggregate counts, which in turn reinforces the large estimates for rate parameters θ. One way to implement a "wider" step at the count rearranging step is to propose not only a rearrangement of counts among machines at time t, but also an update of the underlying state, for example proposing a change of some decay states associated with 0 counts to off states, thus increasing the weight of remaining decay states among active machines to counteract large contributions from the extreme rates at the hour. The

step would be adapted from a Gibbs step to a Metropolis-Hastings step that included terms similar to those developed in Chapter 4 for updating a single state in the chain.

The merge-split strategy was chosen as an initial methodology for updating H based on the similarities of the non-informative network model to finite mixture models akin to those studied in Richardson and Green (1997). But for the generation-allocation model, the updating step is highly multidimensional and difficult to control. Merge-split steps have been noted as poorly-mixing for other less highly-dimensional contexts (Al-Awadhi et al., 2004), and even with relatively good mixing, MCMC chains would need to be run for many millions of iterations in order to explore fully the posterior space. Simplifying the generation model to admit fewer overall hit rates λ can help in designing steps that lead to reasonable acceptance ratios. When starting out from a single large machine, a split step can be designed that uses information about categorical shifts in the deviations of y_t at high-decay and baseline states from the baseline rate q. Reversing the order of updating for $\boldsymbol{\eta}$ vs. θ may also make it easier to design interpretable 2-dimensional steps that map (θ_1, θ_2) to (θ_{12}, δ) and vice versa. It may also be useful to re-imagine the merge-split step as time-dependent; though the Conficker-C study was a retrospective across all hours, ongoing study of botnets involves updating counts on a time-by-time basis.

Another way forward is to re-imagine the updating of H with steps other than merge-split steps. For example, the study of the large population for Machine 54 suggests that a useful step may be adding or deleting a machine with generally low-valued rates across the observation period. When the current population size is equal to 1 this step is similar to a merge-split step, but for larger population sizes it would be able to spread allocation of counts to and from small-valued machines across all other machines in the population, possibly reducing the effect of a change in counts on the associated Poisson log-likelihoods. An Add/Delete step that adds a machine with very low activity can also be seen as a kind of weak non-identifiability centering method in an RJMCMC context, as suggested by Brooks et al. (2003). This step would add very low-valued machines, but coupled with a wider count-rearranging step that updated larger components of the parameter vector M, may be able to quickly build up a non-trivial activity profile for the proposed machine.

Chapter 6

Recommendations and Future Work

This research has outlined a general framework for modeling botnet behavior through the lens of IP space, by modeling the measurable, observable activity–in the form of a count representing a log-in, communication attempt, or volume of information sent across a network–as the result of a machine-level generation model informed by sandbox tests and user activity profiles, and a network-level allocation model informed by network protocols and IP address assignment. Preliminary analysis focused on showing how to develop the functional form of the behavioral model, and on how to use it in estimation for a directly observable population and for a non-informative population such as a collection of machines behind a NAT device. A proof of concept was explored using the Conficker-C botnet as a case study. Section 6.1 outlines some future directions and recommendations for expanding analysis beyond non-informative networks and Conficker-C. Section 6.2 provides some concluding remarks.

6.1 Future work

6.1.1 Single machine modeling

Suggestions for future modeling efforts for Conficker-C specifically are addressed in Sections 4.6.6 and 5.3.7. The case study with Conficker-C also highlights several broader goals for developing behavioral models that can help to inform machine- or device-based population estimates when activity is observed via IP address:

- **Simplicity**: A flexible model for counts can be useful for directly observable populations, but even informed priors may be overwhelmed by low-likelihood configurations of counts when T is large. Flexible priors that seem reasonable for the directly observable case can nonetheless cause strange configurations when the individual counts y_{it} are visible only up to the aggregated counts x_t. If possible, a simple model for counts would relegate all dependence between time periods to the transition model for user activity. Of course, some kinds of activity–such as user-initiated functionality in the malware–may be correlated directly with user activity patterns as well as previously generated volumes. Modeling the dependencies in rates as short-lived (for example allowing spiking in activity but forcing quick returns to baseline), can help to translate the interpretability of the single-machine model to the larger population.

- **Functional accuracy**: Even for behaviors that do not change rate with increased user activity, a behavioral model will need to account for active vs. inactive machines. The model for user activity should be flexible enough to include weekday versus weekend patterns, and to allow for both persistent and ephemeral users. Though the parametric form for transitions in the Conficker-C study was efficient in the number of parameters estimated, analysis of both the directly observable and non-informative populations indicated that it fit the data poorly. For future studies, it may be a better choice to estimate independent 24-hour periodic probabilities of activity, with flexibility to account for weekends and holidays. For large NAT networks, a mixture of multiple models of activity may be required to account for machines that have persistent low levels of activity versus machines whose activity follows much more closely with the Monday through Friday work schedule.

- **Granularity**: Generation and allocation is a continuous process that is aggregated across discrete time periods t for a first attempt at modeling. Depending on the type of behavior that is modeled, the time period can be as short as a few seconds, or as long as a day. The granularity for a model should be selected to minimize the conflation of variability that arises due to changes in the generation rate with variability that arises due to changes in user activity. This active versus inactive variation is not modeled by sandbox tests. For example, if the malicious software is programmed to check in every 24 hours, then a time period aggregating counts by $t = 1, 2, 3 \ldots$ days would still sum homogenous activity volumes from all underlying machines, conflated only with active vs. inactive patterns. But

for counts such as spam volumes, it is more useful to pick a level of granularity that does not aggregate more continuous measures of volume across potentially different profiles of active vs. inactive hours within the day. Conficker-C was modeled by hour because machines tend to stay either active or shut down for periods longer than 1 hour.

6.1.2 Marginal likelihood methods

The value H can be seen as an index to a model family, with the reversible jump MCMC merge/split or add/delete step seen as a "between-model" method for traversing the model indexing space (Friel and Pettitt, 2008). Alternatives to reversible jump MCMC methods are "within-model" methods for calculating the marginal likelihood. These methods are employed in situations where, for example, the number of components in a mixture model is of more interest than the model parameter estimates. The normalizing constant $m(y)$ is the marginal probability of the data y under the model with parameter θ, which can be calculated using MCMC output.

There are several options for applying marginal likelihood to the generation and allocation model. In cases such as mid-sized NATs, the marginal likelihood $\pi(m(\mathbf{x}) \mid h)$ may tail off quickly toward 0 for relatively small values of h. A good approximation of the marginal posterior of H can be obtained by calculating $\pi(h)\pi(m(\mathbf{x}) \mid h)$ for a finite set of values $h = 1, \ldots H_0$, using MCMC output to estimate the marginal likelihood for each value, for example using the method of power posteriors (Friel and Pettitt, 2008). When the priors $\pi(H)$ and $\pi(\kappa)$ are constrained to the effect that the probability of large accumulations of counts y_{it} for a single machine approaches 0 fast enough, then a Central Limit theorem may hold for H. In this case, for larger NATs, a Normal approximation to the marginal posterior distribution can be efficiently employed, again using MCMC iterations, if the mode $\tilde{H}$ can be found quickly. Then $p(H \mid \mathbf{x})$ can be approximated as Normal with mean equal to $\tilde{H}$ and standard deviation derived from the ratio of Bayes factors $[\pi(\tilde{H})\pi(m(\mathbf{x}) \mid \tilde{H})]/[\pi(\tilde{H}+1)\pi(m(\mathbf{x}) \mid \tilde{H}+1)]$ (Kass and Raftery, 1995).

One caveat on the use of marginal likelihood as opposed to a within-model setting, is that information is lost when models have shared parameters. For generation-allocation models, shared information can refer to global survival and immunity rates across populations, or for any hyperparameters κ associated with the prior distributions of generation parameters ξ, for example in the Conficker-C model, including prior

distributions on the hyperparameters a_α and b_α for the decay rate α. Marginal methods such as those used by Basu (1998) and Fienberg et al. (1999) avoid the need for RJMCMC in this situation by decomposing the parameter vector θ into $\theta = (\kappa, \{\xi\}_1^H)$, and sampling a value H from the conditional distribution $\pi(H \mid \kappa)$, marginalized across all parameters whose dimension depends on H, and then sampling $\{\xi\}_1^H$ from the distribution $\pi(\{\xi\}_1^H \mid H, \kappa)$. This method avoids the explicit use of RJMCMC but requires that the hyperparameters κ provide enough transfer of information to make generating the parameters $\{\xi\}_1^H$ efficient. When hyperparameters are fixed in the non-informative model, Basu's method reduces to the calculation of the full marginal likelihood in for finding the conditional distribution of H, as the aggregated counts x_t are the only observed values in the study, and all machine-specific parameters M depend on H. Estimation for informative networks may be able to use the method to transfer information from a set of fixed or high-confidence one-to-one allocations among smaller blocks of obscured activity, similar to the Add/Delete steps suggested in the discussion section for Chapter 5.

Marginal likelihood is also more difficult to calculate using MCMC output when there are large discrepancies among the likelihood values at different iterations. One way to address this is to use the ratio of likelihoods for machines to a mean or average machine at each step, to try and center the likelihood values. Even if the model for activity rates is very simple, exploring the space of $H + 1$ machines as opposed to H machines introduces many new parameters due to the different ways to arrange counts y_t among machines for each time period t. One way to account for this may be to use approximations based on posterior population distributions and posterior means or modes (akin to the information in the summary tables from Section 5.3.3) for the integrals of continuous parameters in the single-machine model, with appropriate multiplicity to account for the rearrangement of labels and the wider spread due to many more possibilities for arrangements of counts.

Overall, adding or deleting machines involves a tradeoff between increasing the population size versus adapting the parameters of the current configuration of machines to best account for the aggregated counts x_t. When the history of counts $1 \ldots T$ gets large, the effect from non-zero prior probabilities can be outweighed by the sheer number of terms in the joint likelihood, even when those prior probabilities are very small. But many of the terms arise from what are essentially nuisance parameters in the likelihood; the interpretability of the underlying components as machines comes from the generation parameters ξ that are

well informed, and from interpretable patterns of activity in the population. The measure of "fit" of a model depends on how well the flexibility of added machines can fit the aggregated counts, and in a confirmatory sense, how well the posterior distributions of single-host population parameters conform to the informed priors. A model selection method such as DIC (Spiegelhalter et al., 2002) may be useful for a "between-model" approach to inference for H, but it is necessary to determine an interpretable but tractable focus for the hierarchical model, and to account for label-switching among the constituent machines.

6.1.3 Informative networks

Informative network allocation uses the full generation-allocation structure from Equation 2.5 for modeling and estimation:

$$\begin{aligned} p(\mathbf{W}, \theta \mid \mathbf{x}) &\propto \pi(H)\pi(\kappa)\pi(\phi \mid \kappa) \\ &\times \prod_{i=1}^{H} \pi(\xi_i \mid \kappa) \prod_{t=1}^{T} f(y_{it}|\xi_i, y_{i1}, \ldots, y_{it-1}) \frac{y_{it}!}{\prod_j w_{ijt}!} \prod_j [g_{ijt}(\mathbf{W}_{1,\ldots,t-1}, \phi_j)]^{w_{ijt}} \end{aligned}$$

The observed counts x_{jt} are still aggregations of the machine-generated activity, but an added hidden layer of allocations w_{ijt} are also present and must be proposed and updated within the MCMC estimation structure. The generation and allocation models are again linked through the machine-level counts y_{it}, and the information about allocation comes from the allocation parameters ϕ. The updating of machine-specific parameters, conditional on observed counts, uses the same conditional distributions as both the directly observable population and the non-informative network setting. But the method for both rearranging counts as well as updating H by merge/split or add/delete methods must now also incorporate new proposals of allocations, or changes in the likelihood of allocations given the newly proposed machines. The complete conditional distribution of allocations depends on graph-connected components of machines and their associated networks. Rearranging of counts, now among the sub-allocations $w_{1jt} \ldots w_{Hjt}$ can be done with a Gibbs step similar to the non-informative Gibbs step, but that adjusts the generation rates by the respective allocation probabilities to account for the multinomial component of the generation-allocation model.

A simple application of informative networks is to include information about the *a priori* probability

of assigning multiple machines to a single IP address within the time period t over which counts y_{it} are aggregated. While NAT devices allocate all activity to the same network touchpoint, large-scale DHCP regions tend to assign devices one-to-one with IP addresses, up to the extent of time a leased IP address is allocated. This means that while DHCP regions may churn through many IP addresses over long periods of observations, the active size counts at any particular time period (when aggregation periods are sufficiently shorter than a typical lease window), are close to one-to-one allocations. Malicious networks engaged in subterfuge to avoid blacklisting may employ very short leases in an effort to spread activity across large portions of IP address space; depending on the granularity of collection, these can show up as either one-to-one allocations over short intervals or even as unusually low activity rates across many machines. As an example, the Ukrainian Telecom network summarized in Figure 3.2 in Chapter 3 and analyzed in Chapter 5 churned through over 26000 IP addresses across the 51-day window of observation for Conficker-C. Intuition suggests that the first few weeks of activity shown in that network correspond roughly to one-to-one allocation per hour, but this changes during the week of April 1st, when the IP address counts rise but the connection rates per IP address sink below the sandbox-informed prior mode for baseline rates q. A first attempt at modeling an informative network would propose a typical DHCP lease rate for large networks such as this one, and then use that model in conjunction with user activity models and hourly IP address counts to inform H within the MCMC architecture.

6.1.4 Statistical theory

There are several implications of the theoretical framework for the generation-allocation model that have been posited but not fully explored or proven.

The general non-informative network: A NAT is one practical example of a non-informative network, with only one IP address to which all activity is assigned. But the generic example is a network with J addresses, among which all allocations at any time period have probability equal to $\frac{1}{J}$. One can consider, for example, ignoring information from behind an unreliable or malicious DHCP network similar to the Ukrainian Telecom Network in the non-informative study. It remains to be proven that in this case, the sub-counts $x_{t1}, \ldots, x_{tJ}$ among all IP addresses at each hour are nuisance parameters for the inference on generation parameters ξ and on the population H. Intuitively, the likelihoods associated with each sub-count

among networks are equivalent up to the constant $\frac{1}{J}$ to the model that treats the non-informative network as a simple NAT with observed counts $x_t = \sum_j x_{tj}$. The goal is to show that in a truly non-informative network, the aggregate counts x_t are sufficient statistics for inference regarding the generation process and the population size. A first approach is to start with the simple case where $J = 1$, that is, the corporate NAT network, and to complete the proof by induction on J.

Links to continuous time models: The discrete time generation-allocation model outlined in Equation 2.5 is a useful empirical summary of counts as they are aggregated among time periods, allocated among IP addresses and reported as summaries. This interpretation is similar to an empirical model as suggested by Cox (1990). But in reality, a machine is assigned to an IP address for lengths of time, and activity is generated in continuous time using for example a Poisson process. Thus if allocation could be measured instantaneously, it would be similar to a continuous time Markov state machine that assigned all communications through an IP address with probability 1 over a specific amount of time. This corresponds to a kind of degenerate multinomial model across networks associated with the machine. At any instantaneous point in time, one of the associated IP addresses has probability 1, and all others have probability 0. But data collection, especially of scan attempts, is expensive to do instantaneously. It requires every communication point to have a corresponding timestamp saved. Instead, what is generally recorded is a time interval and an aggregation of counts. The question to address is, does integration and aggregation of counts over time result in the discrete time generation-allocation model? If so, the model can be considered more as a generative or substantive model, as opposed to just a useful summary.

Preliminary analysis suggests that there is at the least a very simple link between the continuous time Poisson process W_t and Markov models, and the discrete time generation-allocation model. Suppose that a generation Poisson process $f(t)$ produces activity with rate λ_t, and furthermore that the possibly non-homogenous rate λ_t can be broken up into sections $t_0, t_1, \ldots, t_K$ such that $\lambda_t = \lambda_k$, a constant, for $t_{k-1} \leq t \leq t_k$. Suppose that the finite-state, continuous-time Markov process Y_t with states $1, \ldots J$ representing allocation among J IP addresses, is observed over the interval $[t_{k-1}, t_k]$. With the allocations observed, suppose that activity is aggregated so that only a value $X_k = W_k - W_{k-1}$ is observed. The distribution of X_k is Poisson with rate equal to $\lambda_k(t_k - t_{k-1})$. Conditional on the sum X_k across the interval $[t_{k-1}, t_k]$, the constant rate across the interval implies that the interarrival times of the activity points are uniformly

distributed across the interval. This results in a Multinomial distribution for assigning sub-counts to the IP addresses assigned in the interval, with probabilities equivalent to the ratios of time that the Markov process Y_t spent in each address within the interval, to the total time interval.

This argument provides some support for a substantive view of the generation-allocation model, as well as some support for modeling activity counts at a granularity that can be considered piecewise constant. But the theory must be expanded in order to account for aggregation not only of a discrete count across the interval, but of a joint set of tuples that represent a timestamp at which activity was seen, and an associated IP address through which the activity was directed.

6.2 Concluding remarks

As the Internet moves more toward ephemeral addressing systems and away from a one-to-one association of devices with IP addresses, the question of "how big is it?" will continue to evolve beyond simple observable metrics. Malicious actors already use multiple techniques to obscure the sizes of their networks and to try to spread activity among many different IP addresses in order to avoid blacklisting. Even benign network assignment policies can make population estimation based on IP address alone difficult. But a botnet cannot obscure its behavior, whether that behavior arises from spam emails, DDoS volume attacks, or maintaining its own internal communications. These measurements, informed by malware analysis, provide a clear indication of "size" at a device level, that can be used as the basis for measuring the true number of devices that lay behind the network of observable communication points. The Conficker-C case study is a concrete example of behavior-based model development, with a few suggestions for posterior inference for both population size and for studying individual differences in activity pertaining to the malware's behavior.

A behavioral model for a population "in the wild" needs to account for more than just a static study of malicious software, and the generation-allocation model uses several components which need to be modeled well in order to provide useful information:

1. the single-machine malware activity model;
2. the user activity transition model;
3. the birth/death/immunity model;

4. the network allocation model.

The data required to inform these components is a synthesis of avenues of research that already exist in cyber-security. Sandbox testing is a well-established practice in the operational security and malware research communities, but rarely has such information been collected with behavioral network modeling in mind, as opposed to signature development. Statistical formalism is broadly applicable in this situation, and models can improve if data collection focuses on accounting for different possible sources of variability common to infected networks. Baselining activity by device is a staple of network behavioral anomaly detection; the approach is generally taken from an internal as opposed to external perspective, but these results as well as broad-scale observations can all be used to inform user activity models and network allocation models. Botnet propagation is another popular avenue of research, and traditional survival analysis and multiple recapture models for open populations can provide a starting point for modeling births and deaths in a virtual environment.

The Bayesian generation-allocation model and graph structure developed in this report can be used as a general framework for modeling Internet populations based on machine-level behavior. Though estimation methods are preliminary, they provide a formal method for incorporating both the knowledge and the uncertainty from these disparate data sources into the overall study of a botnet's size, power and threat.

Appendix A

Code

C++ code was created using the GNU Scientific Library [1] for estimation of Conficker-C behavior in the graph structure outlined in Chapter 2. Specification files are included for:

- Classes for data and parameters associated with the Conficker-C single-host model for Chapter 4;
- A Global parameter class;
- Machine and Network classes;
- A Distribution class that encapsulates the set-up and tear-down of the GSL random number distributions for log-density evaluation and simulation;
- A Parameter updater class that implements generic MCMC steps for parameters and data for a single machine;
- Parameter sub-classes implementing the MCMC estimation outlined in Table 4.1 from Chapter 4;
- A Count Rearranger class that implements the Multinomial Gibbs step for a conditional Poisson model as outlined in Chapter 5;
- A Merge-Split updater class that implement the merge-split progressions described in Chapter 5 for sets of machines;

[1] `http://www.gnu.org/software/gsl/`

- A Network Graph object that consists of fixed sets of Network touchpoint objects that can spawn associated Machine objects, and that implements Random Sweep MCMC at the level of Machines and Networks;
- Various configuration files for initializing Network and Machine data from collections of counts.

Output analysis and dashboard graphics were produced using the R environment. For more information about the code, please contact the author at `rweaver@cert.org`

Bibliography

Al-Awadhi, F., Hurn, M., and Jennison, C. (2004). Improving the acceptance rate of reversible jump MCMC proposals. *Statistics and Probability Letters*, 69(2):189–198.

Amit, Y. and Grenander, U. (1991). Comparing sweep strategies for stochastic relaxation. *Journal of Multivariate Analysis*, 37:197–222.

Armitage, D. and Ober, H. (2010). A comparison of supervised learning techniques in the classification of bat echolocation calls. *Ecological Informatics*, 5(6):465 – 473.

Basu, S. (1998). *Bayesian estimation of the number of undetected errors when both reviewers and errors are heterogenous*, pages 19–36. Singapore: World Scientific.

Bradlow, E. and Schmittlein, D. (2000). The little engines that could: modeling the performance of world wide web search engines. *Marketing Science*, 19(1):43–62.

Briand, L., Emam, K., Freimut, B., and Laitenberger, O. (2000). A comprehensive evaluation of capture-recatpure models for estimating software defect content. *IEEE Transcripts of Software Engineering*, 26:518–540.

Brooks, S., Giudici, P., and Roberts, G. (2003). Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions. *Journal of the Royal Statistical Society, Series B*, 65(1):3–55.

Burnham, K. and Overton, W. (1978). Estimation of the size of a closed population when capture probabilities vary among animals. *Biometriks*, 65:25–633.

Chan, M. and Hamdi, M. (2003). An active queue management scheme based on a capture-recapture model. *IEEE Journal on Selected Areas in Communications*, 21(4):572–583.

Cormack, R. (1989). Log-linear models for capture-recapture. *Biometrics*, 45:395–413.

Cox, D. (1990). Role of models in statistical analysis. *Statistical Science*, 4:169–174.

Daniels, H. (1954). Saddlepoint approximations in statistics. *Annals of Mathematical Statistics*, 25:631–650.

Darroch, J., Fienberg, S., Glonek, G., and Junker, B. (1993). A three-sample multiple-recapture approach to census population e stimation with heterogenous catchability. *Journal of the American Statistical Association*, 88:1137–1148.

Demptser, A. and Rubin, N. L. D. (1977). Maximum likelihood from incompelte data via the EM algorithm. *Journal of the Royal Ssatistical Society, Series B*, 39:1–38.

Dupuis, J. and Schwarz, C. (2007). A Bayesian approach to the multistate Jolly-Seber capture-recapture model. *Biometrics*, 63:1015–1022.

Faber, S. (2009). Silk Conficker.C Plug-in. `http://tools.netsa.cert.org/wiki/display/tt/SiLK+Conficker.C+Plugin`. CERT Code release.

Fienberg, S. (1972). Multiple-recapture census for closed populations and incomplete contingency tables. *Biometrika*, 59:591–603.

Fienberg, S. (1980). *The Analysis of Cross-Classified Categorical Data*. MIT Press.

Fienberg, S., Johnson, M., and Junker, B. (1999). Classical multilevel and bayesian approaches to population size estimation using multiple lists. *Journal of the Royal Statistical Society: Series A*, 162(3):383–405.

Friel, N. and Pettitt, A. (2008). Marginal likeihood estimation via power posteriors. *Journal of the Royal Statistical Society, Series B*, 70(3):589–607.

Gilks, W., Richardson, S., and Spiegelhalter, D. (1996). *Markov Chain Monte Carlo in Practice*. Chapman Hall.

Gopalaswamy, A., Royle, J., Delampady, M., Nichols, J., Karanth, K., and Macdonald, D. (2012). Density estimation in tiger populations; combininb information for strong inference. http://www.esajournals.org/doi/pdf/10.1890/11-2110.1. Pre-publication draft for the Ecological Society of America.

Green, P. (1995). Reversible jump Markov chain Monte Carlo computation oand Bayesian model determination. *Biometrika*, 84(4):711–732.

Heinemeyer, K., Ulizio, T., and Harrison, R. (2008). *Natural Sign: Tracks and Scats*. Island Press.

Holz, T., Steiner, M., Dahl, F., Biersack, E., and Freiling, F. (2008). 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*, pages 9:1–9:9, Berkeley, CA, USA. USENIX Association.

Kadane, J., Meyer, M., and Tukey, J. (1999). Yule's association paradox and ignored stratum heterogeneity in capture-recapture studies. *Journal of the American Statistical Association*, 94(447):855–859.

Kang, B., Chan-Tin, E., Lee, C., Tyra, J., Kang, H. J., Nunnery, C., Wadler, Z., Sinclair, G., Hopper, N., Dagon, D., and Kim, Y. (2009). Towards Complete Node Enumeration in a Peer-to-Peer Botnet. In *ACM Symposium on Information, Computer and Communication Security (ASIACCS 2009*.

Karanth, K. and Nichols, J. (1998). Estimation of tiger densities in India using photographic captures and recaptures. *Ecology*, 79(8):2852–2862.

Kass, R. and Raftery, A. (1995). Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795.

Levine, R. (2005). A note on Markov chain Monte Carlo sweep strategies. *Journal of Statistical Computation and Simulation*, 75(4):253–262.

Leydon, J. (2010). Google: botnet takedowns fail to stem spam tide. http://www.theregister.co.uk/2010/04/18/google_botnet_takedowns/. Online article for The Register (UK).

Mane, S., Mopuru, S., Mehra, K., and Srivastava, J. (2005). Network size estimation in a peer-to-peer

network. Technical Report TR 05-030, University of Minnesota Department of Computer Science and Engineering.

Mariano, L. and Kadane, J. (2001). The effect of intensity of effort to reach survey respondents: a Toronto smoking study. *Survey Methodology*, 27:131–142.

MaxMind (2010). GeoLite City. `http://www.maxmind.com/app/geolitecity`.

McCullagh, P. and Nelder, J. (1989). *Generalized Linear Models*. Chapman and Hall/CRC.

McLachlan, G. and Peel, D. (2000). *Finite Mixture Models*. John Wiley & Sons, Inc.

McMillan, R. (2010). Spanish Police Take Down Massive Mariposa Botnet. `http://www.pcworld.com/businesscenter/article/190634/spanish_police_take_down_massive_mariposa_botnet.html`. Online article in PCWorld.

Nunnery, C., Sinclair, G., and Kang, B. (2010). Tumbling Down the Rabbit Hole: Exploring the Idiosyncrasies of Botmaster Systems in a Multi-Tier Botnet Infrastructure. In *The 4th USENIX Conference on Large Scale Exploits and Emergent Threats (LEET 2010)*.

Park, T. (2011). Bayesian analysis of individual choice behavior with aggregate data. *Journal of Computational and Graphical Statistics*, 20(1):158–173.

Paxson, V. and Floyd, S. (1995). Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244.

Petersen, C. (1896). The yearly immigration of young plaice into the limfjord from the german sea. *Report of the Danish Biological Station*, 6:5–84.

Porras, P., Saidi, H., and Yegneswaran, V. (2009a). Conficker C Actived P2P scanner. `http://www.mtc.sri.com/Conficker/contrib/scanner.html`. SRI international Code release/document.

Porras, P., Saidi, H., and Yegneswaran, V. (2009b). Conficker C analysis. Technical report, SRI International.

Porras, P., Saidi, H., and Yegneswaran, V. (2009c). Conficker C P2P protocol and implementation. Technical report, SRI International.

Rajab, M., Zarfoss, J., Monrose, F., and Terzis, A. (2007). My botnet is bigger than yours (maybe, better than yours): Why size estimates remain challenging. In *Proceedings of the First Annual Workshop on Hot Topics in Botnets*.

Rasch, G. (1960 (expanded 1980)). *Probabilistic models for some intelligence and attainment tests*. The University of Chicago Press.

Rendon (2011). Conficker Working Group: lessons learned. `http://www.confickerworkinggroup.org/wiki/uploads/Conficker_Working_Group_Lessons_Learned_17_June_2010_final.pdf`. Technical Report produced for the Department of Homeland Security, Air Force Research Contract FA8750-08-2-0141.

Richardson, S. and Green, P. (1997). On Bayesian analsis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society: Series B*, 59(4):731–792.

Roberts, G. (1996). *Markov chain concepts related to sampling algorithms*, chapter 3. Chapman and Hall/CRC.

Royle, J. and Gardner, B. (2011). *Hierarchical spatial capture-recapture models for estimating density from trapping arrays*, pages 163–190. Springer.

Schnabel, Z. (1938). The estimation of the total fish population of a lake. *American Mathematical Monthly*, 45:348–352.

Schwarz, C. and Arnason, A. (1996). A general methodology for the analysis of capture-recapture experiments in open populations. *Biometrics*, 52(3):860–873.

Seber, G. (1982). *The estimation of animal abundance and related parameters*. London: Charles Griffen.

Sekar, C. and Deming, E. (1949). On a method of estimating birth and death rates and the extent of registration. *Journal of the American Statistical Association*, 44(245):101–115.

Shapiro, S. (1949). Estimating birth registration completeness. *Journal of the American Statistical Association*, 45:261–264.

Shuster, S. (2011). Why have hackers hit Russia's most popular blogging service? *Time Magazine*. http://www.time.com/time/world/article/0,8599,2063952,00.html.

Spiegelhalter, D., Best, N., Carlin, B., and van der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society, Series B*, 64(4):583–639.

Stephens, M. (2000). Dealing with label switching in mixture models. *Journal of the Royal Statistical Society, Series B*, 62(4):795–809.

Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmer, R., Kruegel, C., and Vigna, G. (2009). Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *The 16th ACM Conference on Computer and Communications Security (CCS 2009*.

Stover, S., Dittrich, D., Hernandez, J., and Deitrich, S. (2007). Analysis of the Storm and Nugache Trojans - P2P is Here. *Login*.

Tanner, M. and Wong, W. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398):528–540.

the Conficker Working Group (2009). www.confickerworkinggroup.org.

the Mariposa Working Group (2009). http://www.defintel.com/mariposa.shtml.

the SANS Internet Storm Center (2001). http://isc.sans.edu/.

the Shadow Server Foundation (2004). http://www.shadowserver.org.

Tierney, L. and Kadane, J. (1986). Approximations of posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86.

Vaugh-Nichols, S. (2010). Big Botnets and How to Stop Them. http://www.computerworld.com/s/article/9177574/Big_botnets_and_how_to_stop_them. online entry for ITWorld.

Weaver, R. (2010a). Beyond the Top Talkers: Empirical Correlation of Conficker-C Infected IP Space. In *FloCon 2010*. http://www.cert.org/flocon/2010/presentations/Weaver_BeyondTheTopTalkers.pdf.

Weaver, R. (2010b). A probabilistic population study of the Conficker-C botnet. In *PAM 2010, LNCS 6032*, pages 181–190. Springer-Verlag Berlin Heidelberg.

Worth, D. (2010). Businesses fear rise in DDoS attacks in 2010. `http://www.v3.co.uk/v3/news/2256353/businesses-fear-rise-cloud`. article for the V3.co.uk technology blog.

Yule, G. (1903). Notes on the theory of association of attributes in statistics. *Biometrika*, 2:121–134.

Zuckerman, E., Roberts, H., McGrady, R., York, J., and Palfrey, J. (2010). Distributed denial of service attacks against independent media and human right sites. `http://cyber.law.harvard.edu/sites/cyber.law.harvard.edu/files/2010_DDoS_Attacks_Human_Rights_and_Media.pdf`. report by the Berkman Center for Internet and Society at Harvard University.

Glossary

active size A measure of a botnet's size based on the profile of active assets within a short time period.

blacklist A collection of IP addresses that have been marked as suspicious, malicious, or otherwise untrustworthy by network administrators or security analysts. Communications between internal machines on a network and blacklisted external IP addresses are often prohibited. Blacklist is also used as a verb to describe the act of putting an IP address onto a blacklist.

botnet A collection of infected computers that maintain communications to a single human administrator or small organized group of administrators, for the purpose of co-ordinated large-scale attacks.

Command and Control (C&C) channel A dedicated communication line between botnet administrators and their armies of infected machines. Infected machines often check in to Command and Control channels at regular intervals in order to receive instructions.

Distributed Denial of Service (DDoS) Attack A type of cyber attack that aims to shut down servers or resources by bombarding them with connection requests. A distributed denial of service attack is a denial of service attack that originates from many different machines, making it difficult to block or blacklist all incoming requests.

Domain Generation Algorithm (DGA) Any algorithm used by malicious software to generate many random domain names in order to spread communications channels across the space of DNS as opposed to IP addresses.

Domain Name Service (DNS) The architecture and protocol used for translating IP addresses to domain names (eg, `www.google.com`) across the Internet.

Dynamic Host Configuration Protocol (DHCP) A protocol for automatically assigning an IP address from a defined range to a machine or device. Assignments are generally temporary with the assignments expiring after a determined lease period.

Expectation and Maximization (EM) An iterative method for finding the posterior mode in Bayesian inference.

flow data A high level summary of communications over networks, including IP addresses, ports, protocols and volume of information sent.

footprint size A measure of a botnet's size based on activity measured across its entire history.

gateway A single device or IP address that communicates to the Internet on behalf of multiple internal machines.

Hypertext Transfer Protocol (HTTP) The set of rules for transferring files and data over the World Wide Web .

Internet Assigned Numbers Authority (IANA) Authority responsible for the global co-ordination of the top level hierarchy of the Domain Name Service as well as allocating IP addresses to requesting entities such as companies, universities, service providers or individuals.

Internet Engineering Task Force (IETF) A co-operative group that develops and promotes Internet standards.

Internet Protocol (IP) The principal communications protocol suite for relaying packets of information across a network.

Internet Relay Chat (IRC) A chat environment commonly used as a Command and Control channel for botnets.

Internet Service Provider (ISP) An organization that provides access to the Internet and other related services such as Web access and security to its clients.

load-balancing An administrative practice that re-routes traffic during times of heavy volume among several different access points in order to reduce bandwidth.

Markov Chain Monte Carlo (MCMC) An iterative method for exploring the posterior distribution of a parameter θ using specially designed Markov Chains.

Metropolis-Hastings (M-H) A type of Markov Chain Monte Carlo estimation that does not require symmetric candidate proposal distributions.

Microsoft Remote Procedure Call (RPC) A Windows technology that allows client machines to run procedures located on remote servers.

Network Address Translation (NAT) A protocol for translating private IP addresses from an internal network through an external, routable IP address for communication over the Internet. Refers to either the protocol or the device or IP address that is visible as the external gateway.

Network Behavioral Anomaly Detection (NBAD) Methods employed in security software for baselineing typical behavior of assets on an internal network according to a layout of devices and respective protocols, and producing security alerts when behavior deviates from the baseline.

network touchpoint An IP address through which a device communicates over the Internet.

Peer to Peer (P2P) communication An alternative to the traditional client-server architecture that decentralizes the communication structure among related entities in a network by allowing each entity to directly contact another .

protocol A set of well-known instructions and rules that defines how software applications communicate with each other.

proxy A device that communicates over the Internet on behalf of another device.

Request For Comments (RFC) A memorandum in computer network engineering published by the Internet Engineering Task Force, that describes best practices, standards, research or innovations that can be applied to the Internet and related systems.

Reversible Jump Markov Chain Monte Carlo (RJMCMC) A method of Markov Chain Monte Carlo estimation that can be used to explore a parameter space where the dimension of the unobserved variables is also unobserved.

sinkhole To direct or redirect all traffic from a suspicious or malicious IP address to an unused or nonexistent IP address. Can also describe the unused or nonexistent IP address that receives the traffic.

System for Internet-Level Knowledge (SiLK) An open source, command-line tool suite used for collecting and summarizing flow data, high level summaries of network communications.

Transmission Control Protocol The principal protocol for synchronous communication between two machines across a network. TCP is an extension of the IP protocol that provides methods for initiation and shutdown of connections to maintain reliability of data transmission. TCP is comparable to a phone call in telecommunications, with an established protocol for setup, in-line data transmission, acknowledgement of received packets, and teardown of a connection.

User Datagram Protocol The principal protocol for asynchronous communication between two machines across a network. UDP is an extension of the IP protocol that provides methods for fast transmission of datagram packets between two machines, without the need for establishing a synchronous connection, and without checking the integrity, ordering, or completeness of messages sent. UDP is comparable to text messaging in telecommunications; the sender does not require an acknowledgement of receipt, nor a synchronized conversation.

Acronyms

C&C Command and Control.

DDoS Distributed Denial of Service.

DGA Domain Generation Algorithm.

DHCP Dynamic Host Configuration Protocol.

DNS Domain Name Service.

EM Expectation and Maximization.

HTTP Hypertext Transfer Protocol.

IANA Internet Assigned Numbers Authority.

IETF Internet Engineering Task Force.

IP Internet Protocol.

IPv4 Internet Protocol version 4.

IPv6 Internet Protocol version 6.

IRC Internet Relay Chat.

ISP Internet Service Provider.

MCMC Markov Chain Monte Carlo.

M-H Metropolis-Hastings.

NAT Network Address Translation.

NBAD Network Behavioral Anomaly Detection.

P2P Peer to Peer.

RFC Request For Comments.

RJMCMC Reversible Jump Markov Chain Monte Carlo.

RPC Remote Procedure Call.

SiLK System for Internet-Level Knowledge.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.