UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

A characterization of node lifetime distributions in the PlanetLab test bed

Permalink

https://escholarship.org/uc/item/6fs431h2

Author

Verespej, Hakon Gabriel

Publication Date

2010

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA, SAN DIEGO

A Characterization of Node Lifetime Distributions in the PlanetLab Test Bed

A Thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Hakon Gabriel Verespej

Committee in charge:

Professor Joseph Pasquale, Chair
Professor Keith Marzullo
Professor Amin Vahdat

2010

Copyright

Hakon Gabriel Verespej, 2010

All rights reserved.

The Thesis of Hakon Gabriel Verespej is approved and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____
Chair

University of California, San Diego

2010

DEDICATION

To Joe, for the inspiration, the continual motivation,
and the guidance that lead me to achieve a dream.

And to Mom and Dad, for encouraging
and supporting me every step of the way.

Our similarities bring us to a common ground;

Our differences allow us to be fascinated by each other.

Tom Robbins

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Dr. Joseph Pasquale, who chaired the review committee for this thesis and who worked with me through many, many drafts. I would also like to acknowledge Dr. Keith Marzullo and Dr. Amin Vahdat, who provided their input and guidance as members of the review committee and kept me encouraged throughout the process.

ABSTRACT OF THE THESIS

A Characterization of Node Lifetime Distributions in the PlanetLab Test Bed

by

Hakon Gabriel Verespej

Master of Science in Computer Science

University of California, San Diego, 2010

Professor Joseph Pasquale, Chair

In this thesis, we identify patterns of node-lifetimes among nodes participating in PlanetLab. While it is common to assume that time-to-failure follows an exponential distribution, we show that this assumption does not hold true in trace data collected from the PlanetLab test bed. By applying clustering techniques to over a year's worth of time-to-failure data from PlanetLab, we identify a set of six node-lifetime distributions that are well-suited to characterizing the nodes of this particular dataset. Because of the uniqueness of the distributions, the characterization provides the additional benefit of allowing us to infer anomalies such as common administrative policies, outages, and other such events.

Chapter 1

Introduction

1.1 Describing Node Lifetimes in a Distributed System

In distributed systems research, it is common to assume a node-lifetime model in which lifetimes (a lifetime being the duration of a single participant session) are exponentially distributed [6, 15, 18, 23, 24, 25]. That is, the assumption that if we observe node lifetimes in a system for a very long time, we will find that the distribution of these observed participation lengths approaches an exponential distribution.

This assumption of exponentially distributed node lifetimes has been shown to be an inaccurate model of availability for various systems including disk drive populations, computing clusters, and P2P networks [5, 10, 21, 23, 31, 32]. In [4], the authors take the approach of modeling node lifetimes by mixing three distributions since none of the distributions independently provides an adequate representation of the system's behaviors.

The appropriateness of a node-lifetime model is also subject to the particular system to which it is applied. One can reasonably expect measurements taken from a public P2P system to reveal a different set of behaviors than measurements taken from a small cluster of web servers. This implies that it may be very difficult to find a simple distribution that can be well-fitted to all such systems.

Consider also that while the memoryless property [29] of the commonly used exponential distribution is mathematically convenient, it may not provide an accurate reflection of reality. For example, when a system administrator enforces a daily restart policy, the likelihood that a given node will remain up decreases as the amount of time it has been up increases.

Given these issues, it is desirable to find an alternative way to describe the characteristics of node lifetimes in distributed systems with greater accuracy.

1.2 The Challenge of Describing Node Lifetimes

Designing real systems and applying appropriate policies are difficult tasks that require many factors to be considered. Creating a system that must meet some set of performance requirements mandates that the right design-time assumptions be made. This, in turn, requires a designer or administrator to paint a complete and accurate picture of the infrastructure on which the system will be built without overburdening and distracting the designer with too much detail. The specific area of this picture that we seek to fill in is the understanding of the lifetimes of nodes participating in such a system.

In essence, we wish to increase our knowledge about nodes' lifetimes so that we may increase our knowledge about system behavior in general. We must do this in such a way that we do not get lost in detail, while at the same time avoiding the loss of too much information by over-generalizing. For illustration, consider a system with 50 nodes and a simple monitoring scheme in which a dedicated node tracks the availability of the other 49 nodes by pinging each of them once every fifteen minutes, after which the results are recorded to a persistent store. This means we are storing *49 pings * 96 iterations per day = 4704 data entries per day*. Converting this data to lifetimes will result in a much smaller amount of information if the nodes are highly stable and do not have frequent periods of unresponsiveness. However, when the nodes are prone to being unresponsive as would be the case in a system with poor connectivity, we will continue to have large quantities of data even after reducing the ping results to lifetimes. One could, in this case, approximate the behavior of the nodes by applying a standard distribution, but that risks over-generalization which may fail to provide a useful description of the behaviors.

1.3 An Alternative Approach

In this work, we look at an alternative approach to understanding node lifetimes. We apply clustering techniques to trace data obtained from a real system to create a set of distributions that describe behavioral patterns of the nodes sampled. In Chapter 2, we describe the methods with which we analyzed the trace data. Chapter 3

discusses the results of our analysis and Chapter 4 discusses the application of our findings. We present related work in Chapter 5, and conclude in Chapter 6.

Chapter 2

Methodology

2.1 PlanetLab Trace Data

In this thesis, we study a distributed system called PlanetLab [22]. PlanetLab is a test bed consisting of hundreds of nodes used for various research purposes including deploying and testing distributed applications. The nodes participating in the system are hosted by a large number of institutions around the world, including many universities. PlanetLab allows users to register for a share of the system's collective resources, after which those users can consume their share as needed.

Our objective in studying PlanetLab is to characterize the system by describing how its member nodes act in terms of their responsiveness to ping requests. For a node, we define a continuous period of responsiveness as a "lifetime". In other words, this is a duration over which the node's resources are available for consumption by the system's users. We refer to the generalization of a node's observed lifetimes as

the node's lifetime behavior. By extension, we make generalizations over the collective set of nodes and refer to these as the system's lifetime behaviors.

To be able to characterize PlanetLab as described above, we need liveness data for the nodes in the system. That is, we need historic information that will allow us to approximate the lifetimes of all participating nodes over a continuous period of time, and preferably a long one. Fortunately, there exists a well-known set of liveness data called all-pairs-ping [37]. This dataset contains a multi-year history of the results of ping requests sent between every pair of nodes in PlanetLab. These pings were performed at 15 minute intervals. For this thesis, we use an interpolation of the ping data [8] that has a timeline for each participating node, showing when the node was responsive and when it was not. The interpolation we use considers a node to be responsive when at least half of the pings sent to it during the most recent ping iteration were successful. We discuss the decision to use this interpretation of responsiveness in Chapter 2.2.

The dataset covers the period from January, 2004 to June, 2005 with about 200 to 400 nodes participating in the system at any given moment. Nodes in the dataset are identified by IP address. There are 720 unique IP addresses in the dataset with 669 of them each contributing at least one lifetime. To contribute a lifetime, a node must have been viewed as responsive at least once during the time range covered by the dataset.

The all-pairs-ping dataset was chosen because it is easily attainable, easy to interpret, and covers a substantial time-span. Additionally, it is generally familiar to

the academic and broader research communities. We find evidence of this by its use in published research on the topic of availability [20, 25, 34, 36].

2.2 Trace Data Interpretation

In this study, we consider a node from the PlanetLab dataset to be responsive when at least half of the ping requests sent to it during a given ping iteration are successful. This definition is positioned as a reasonable middle ground between requiring only one successful ping, in which case we may consider the node responsive when it is not very useful to the system, and requiring that all pings be successful, in which case we may consider the node unresponsive when it remains quite useful to the system. While this definition of responsiveness makes sense for our study, given the nature of PlanetLab and the all-pairs-ping dataset, we compare it to another interpretation of responsiveness to illustrate the importance of making a reasoned choice regarding how to interpret liveness data.

Figure 2.1 shows the lifetime distributions of a sample of nodes from the PlanetLab dataset. For this figure, a node is considered responsive if at least one ping sent to it during a ping iteration is successful. Figure 3.1 shows the lifetime distributions for the same set of nodes, but with the requirement that at least half of the pings sent to a node be successful to consider it responsive. These figures show both a number of similarities and a number of differences. These similarities and differences depend on the nature of the underlying causes of ping failures. For example, if a node is taken offline for several hours for hardware replacement, both interpretations of

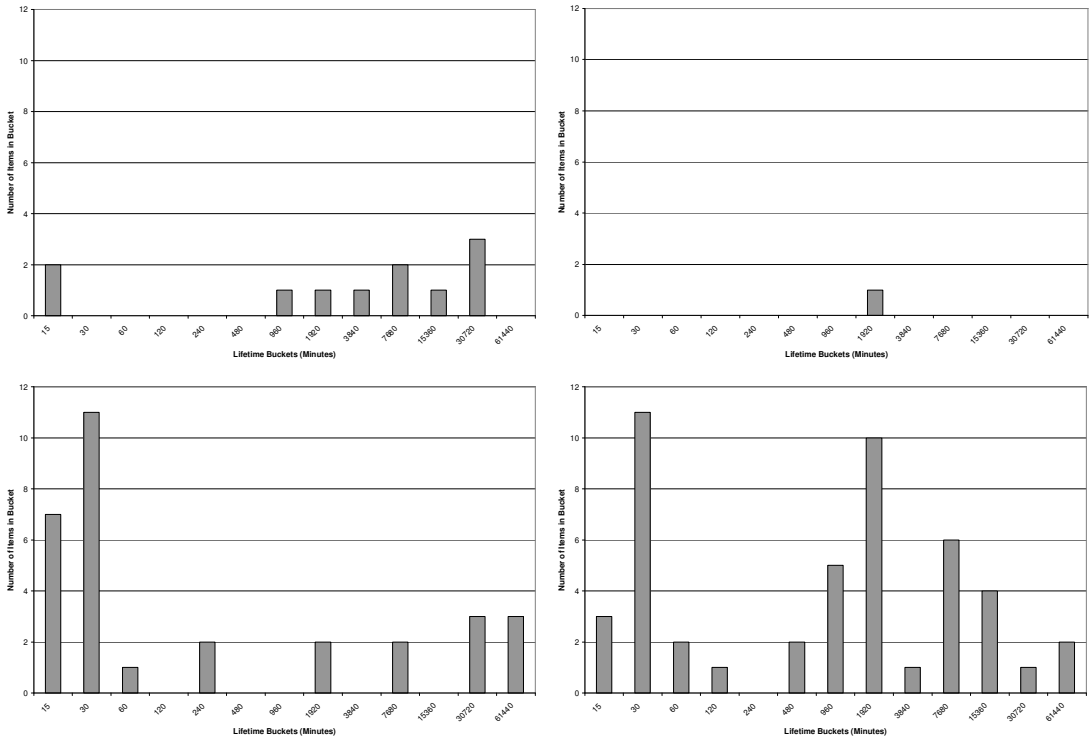

Figure 2.1: A sample of lifetime distributions for PlanetLab nodes, for which a node is considered responsive if it is successfully pinged at least once in a ping iteration. Figure 3.1 shows the same distributions for the same set of nodes, but requiring at least half of the pings in a ping iteration to be successful to consider the node responsive. Similarities and differences between these two sets of distributions depend on the nature of the underlying causes of ping failure.

responsiveness will determine the node to be unresponsive. On the other hand, if a node loses connectivity to a majority of other nodes, but local nodes continue to be able to reach it, it will appear responsive if only one successful ping is required, but will appear unresponsive if half of the pings sent to it are required to be successful. As can be seen by comparing Figures 2.1 and 3.1, this difference can have a substantial impact on the distribution of lifetimes for a node.

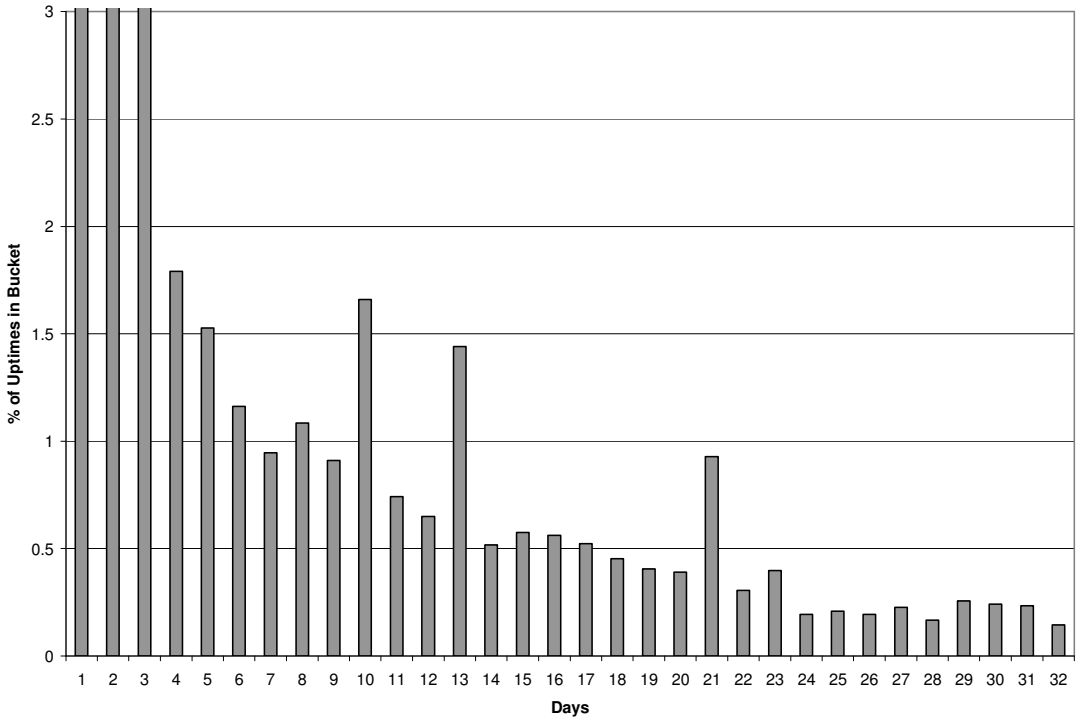

Figure 2.2: A histogram bucketing node lifetimes into 24-hour buckets over a 32-day period in which a node is considered responsive if it is successfully pinged at least once in a ping iteration. Figure 3.2 shows the same histogram, but where at least half of the pings in a ping iteration must have succeeded for the node to be considered responsive. The non-viewable values in the 1-day, 2-day, and 3-day buckets are 73.48, 4.56, and 3.13, respectively.

Figure 2.2 shows the distribution of the aggregation of lifetimes of all nodes under the one-successful-ping requirement. Figure 3.2 shows the same aggregation using the half-of-all-pings requirement. Surprisingly, they are similarly shaped, indicating that despite the differences in the distributions of individual nodes, the general trend has some similarities. This may indicate that the importance of certain events that cause unresponsiveness outweigh others for the all-pairs-ping dataset. One notable difference between the two results that is not observable in the distributions is

node\bucket	b1	b2
n1	0	1
n2	1	1

Figure 2.3: An example of the type of matrix mapping produced by our analysis tool. The mapping shows the number of entries each node has in each bucket.

that the one-response requirement resulted in fewer total lifetimes (26929 vs. 27447), which can be attributed to nodes being considered unresponsive less frequently. The one-response requirement also resulted in a larger weight in the 15-minute bucket (73.48% vs. 63.27%), which is a result of nodes becoming unresponsive less rapidly due to the smaller number of successful pings required. This has the observable effect of a larger tail with the half-of-all-pings requirement. That is, the lower percentage of lifetimes falling into the first bucket resulted in an increase in the percentage of lifetimes in other buckets.

Given the potential effect of using different interpretations of data, we reiterate that one must be careful to make a reasoned choice as to how to interpret liveness data rather than choosing arbitrarily.

2.3 Trace-Analysis Tool

To analyze the liveness data, we built a trace-analysis tool. After initializing the tool by loading a trace file, a user may perform various analytical and transformative operations over the trace data. Among other tasks, we used this tool to

generate node lifetimes for participating members of the system and to sort those lifetimes into buckets of various ranges of lifetime.

Our tool also has the capability to generate a matrix that maps individual participants to lifetime buckets. This essentially treats each bucket as a characteristic of the nodes. The value of the mapping, then, indicates the degree to which a node shows a particular characteristic.

As an example, let us consider two buckets, *b1* and *b2*, where *b1* covers 0 to 24 hours and *b2* covers everything beyond 24 hours. Let us further consider two nodes, *n1* and *n2*. A single lifetime of 1 year was observed for *n1*. Lifetimes of 24 hours and 6 months were each observed for *n2*. The matrix mapping our tool would produce for this set of inputs is shown in Figure 2.3.

2.4 Cluto, a Clustering Tool

To identify common patterns of behavior among nodes, we feed characteristics mappings created by our trace-analysis tool into another tool called Cluto [12]. Cluto is a powerful statistical analysis program developed to handle clustering tasks in the field of biology. As input, it takes several tuning parameters and a matrix mapping like those generated by our trace-analysis tool. It then runs a clustering algorithm over the data in the mapping and outputs a number of element groupings (i.e. clusters). Cluto also provides information on the characteristics that most strongly bind the members in a given group to one another, as well as those that differentiate the group as a whole from other groups.

2.5 Characterizing PlanetLab Nodes

This combination of tools gives us the ability to cluster the nodes captured in trace data by the common characteristics of their lifetime distributions. For the resulting clusters, we are able to determine the qualifiers, or key behavioral characteristics, that define the classifications. In the next chapter, we discuss our application of the classification process to the PlanetLab trace data.

Chapter 3

Results

3.1 Analysis of Individual Nodes

To analyze the PlanetLab trace data, we started by using our analysis tool to form distributions of the lifetimes of each node. Figure 3.1 shows a sample of typical lifetime distributions for individual nodes in the PlanetLab dataset. As with these nodes, it is difficult to identify a characterizing distribution for most of the individual nodes in the dataset. While this says little about whether observing the whole population of nodes will reveal a common distribution or whether observing the individual nodes for a much longer period of time will do so, it does suggest that further investigation is in order.

3.2 Aggregate Analysis

The charts in Figures 3.2, 3.3, and 3.4 show aggregate data that includes lifetimes from the entire set of nodes. As will be described in Chapter 3.3, when we

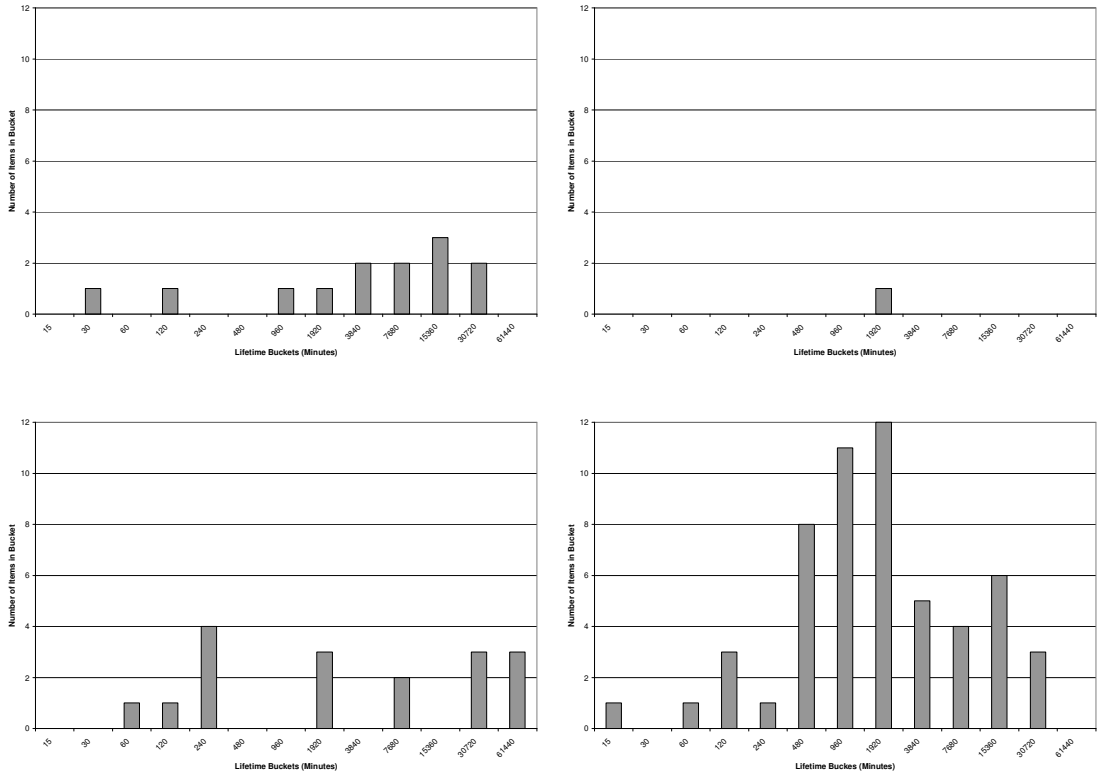

Figure 3.1: A sample of typical lifetime distributions found for individual nodes in the PlanetLab dataset. As exemplified here, it is difficult to identify any particular characterizing distribution through observation of the lifetime data of individual PlanetLab nodes. In these charts, the x-axis is composed of buckets covering ranges of time. The y-axis is the number of lifetimes that fell within a specific bucket. Note that the sizes of the buckets on the x-axis increase exponentially.

filter the data from our dataset for relevance in terms of value towards characterization of nodes, no lifetimes greater than 679 hours (about 28.3 days) show up as important influencers. For this reason, the aggregations only show lifetimes that are up to 32 days in length. This is still 27447 (95.6%) of the 28705 lifetimes in the full dataset. In these charts, the data is displayed at three different levels of granularity: Figure 3.2 uses buckets that are 1 day in size, Figure 3.3 uses buckets that are 1 hour in size, and Figure 3.4 uses buckets that are 15 minutes in size.

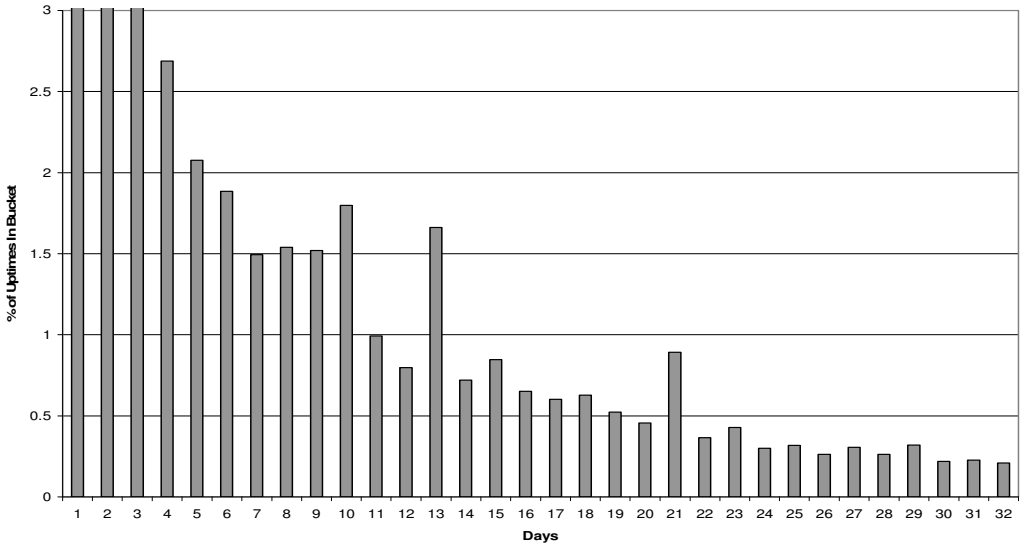

Figure 3.2: A histogram bucketing node lifetimes into 24-hour buckets over a 32-day period. Some buckets are more highly concentrated than their neighbors. The weight of the 13-days bucket, for example, indicates that the data set contained many instances of lifetimes just short of two weeks. The non-viewable values in the 1-day, 2-day, and 3-day buckets are 60.5, 6.05, and 4.09, respectively.

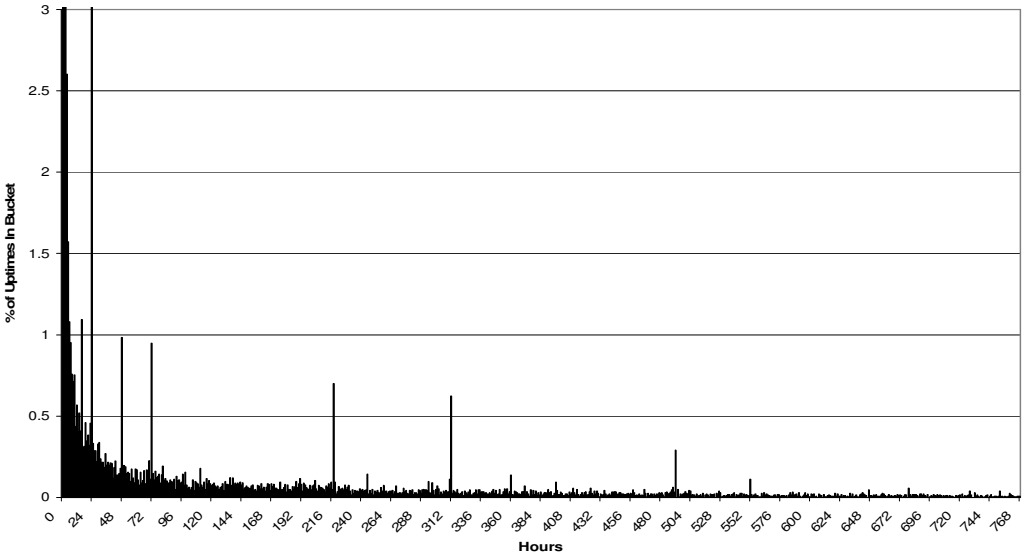

Figure 3.3: A histogram bucketing node lifetimes into 1-hour buckets over a 32-day period. Some buckets are very heavily concentrated in relation to their neighbors. The degree to which they are concentrated hints that they are not a result of chance. The non-viewable values in this image include 23.7 in the 1 hour bucket, 14.7 in the 2 hours bucket, 4.6 in the 3 hours bucket, and 3.02 in the 24 hours bucket.

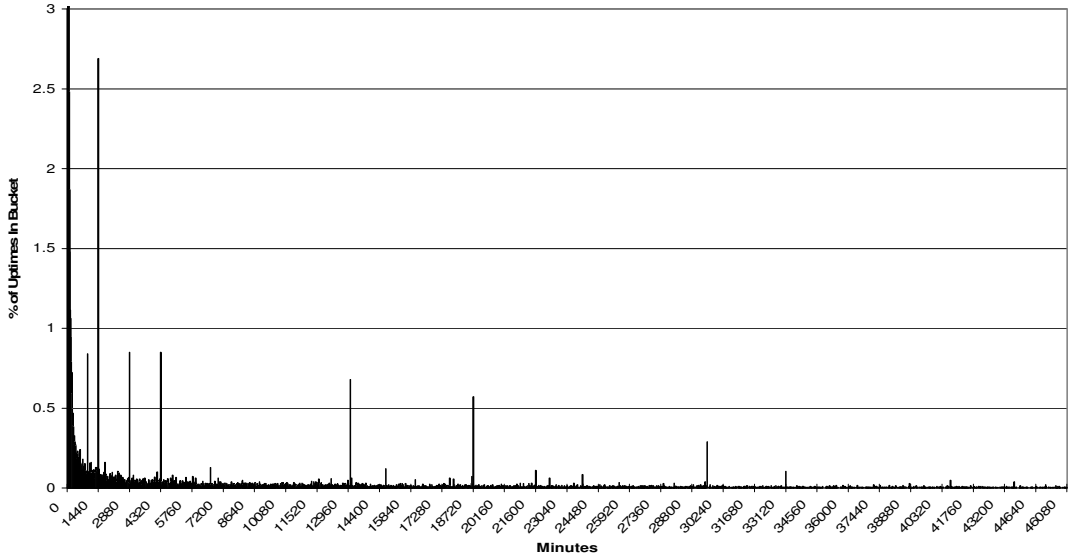

Figure 3.4: A histogram bucketing node lifetimes into 15-minute buckets over a 32-day period. The shape in this figure is similar to that in Figure 3.3. Further, each noticeable spike in Figure 3.3 has a single matching spike in this figure. This tells us that the spikes in Figure 3.3 were a result of many node lifetimes falling within a tight 15-minute period rather than being evenly-distributed over the entire hour. The non-viewable values in this image include 3.64 in the 15-minutes bucket, 6.92 in the 30-minutes bucket, 6.34 in the 45-minutes bucket, 6.84 in the 60-minutes bucket, 6.23 in the 75-minutes bucket, and 4.14 in the 90-minutes bucket.

To check whether the data is exponentially distributed, we performed a Chi-Square goodness-of-fit test [28] on the day buckets shown in Figure 3.2. The Chi-Square Goodness of Fit Test judges whether an observed distribution differs from a test distribution, which in this case is the exponential distribution. In this test, a Chi-Square statistic is calculated based on the difference between the observed data and what the (exponential) distribution would have yielded. This statistic is then tested against the value from the Chi-Square distribution that corresponds to a desired level of significance.

Testing the buckets from Figure 3.2 resulted in a Chi-Square statistic value of 43163. For a significance level of 5%, the statistic value must be less than 42.6 for us to accept the exponential distribution as a good fit. The result of our test clearly does not satisfy this. A large portion of the difference between the observed data and the output from the exponential distribution is attributable to the bucket covering the first day, which holds slightly more than 60% of the entries (which is not clearly visible in Figure 3.2). To be sure that this was not just anomalous behavior causing an incorrect failure of the goodness-of-fit test, we recalculated the Chi-Square statistic. For the recalculation, we excluded the first bucket and all of its entries and adjusted the exponential distribution accordingly. Doing so still resulted in a Chi-Square statistic value of 873. With one less bucket than before, for a significance level of 5%, the Chi-square statistic value must be less than 41.3 for us to accept the exponential distribution as a good fit. Again, the check failed by a large margin.

Figures 3.2, 3.3, and 3.4 make apparent several buckets of heavy concentration. These manifest themselves as spikes in the data, where the number of items in a particular bucket is substantially greater than the number of items in the buckets surrounding it. For example, the 48-hour bucket in Figure 3.3 contains about 0.98% (282) of all observed lifetimes. The bucket immediately preceding this (the 47-hour bucket) contains about 0.18% (51) and the bucket immediately following (the 49-hour bucket) contains about 0.12% (33). Thus, the 48-hour bucket is well over 500% more concentrated than its adjacent neighbors, causing a substantial and visible spike in the data.

For any bucket showing a spike in the 1-hour granularity, there is a single corresponding spike within the four buckets of 15-minute granularity that fall within that hour. That is, the appearance of a spike at the coarser 1-hour granularity is not due to heightened activity over the whole range of time covered by the bucket, but is instead attributable almost entirely to a concentrated period of no more than 15 minutes. This phenomenon causes the same behavioral pattern to emerge at different levels of granularity. To observe this, we again refer to the heavily concentrated 48-hour bucket from Figure 3.3. In Figure 3.4, which has a 15-minute bucket size, we find the four buckets in the 47 to 48 hours range to have concentrations of 0.066% (19), 0.035% (10), 0.031% (9), and 0.85% (244), respectively. The last of these buckets is over 1200% more concentrated than any of the other three. Thus, we see that the spike in the hour bucket is attributable to the activity in the tighter 15-minute time-range from 47.75 to 48 hours. Because of the fact that this occurs in so many places over the distribution and that there is significant space between any two points of concentration, it seems unlikely that the concentrations are a result of coincidence. Instead, it is much more likely that something caused nodes to behave similarly.

In the remainder of our analysis, we focus primarily on the 1-hour granularity because nearly the same behavioral pattern is observable with 1-hour buckets as is observable with 15-minute buckets, but allows us to work with a manageable number of buckets. We prefer this to the 1-day granularity since 1-day granularity is too broad and loses interesting patterns such as spikes occurring at the end of a fixed number of days.

3.3 Forming Clusters of Nodes

To isolate the most important characteristics for clustering the behaviors of nodes in PlanetLab, we started by running Cluto over the entire set of data and observing which buckets it identified as the most relevant for forming and distinguishing node groups. We augmented this initial set of buckets by taking its union with the set of all buckets from the original data that were several times more concentrated than either the preceding or proceeding bucket. Thus, the resulting set of buckets consisted of those chosen by Cluto for relevance and those chosen by hand for statistical interest. We ran Cluto once more on this set of buckets to identify the 15 most relevant, which is a number that allows us to include the most powerful distinguishing characteristics and a fair portion of less powerful, but still relevant, characteristics.[1] In a few cases, multiple buckets were merged into a single bucket since Cluto used them similarly in forming and distinguishing groups. While these individual buckets were useful in creating the clustering, there was no need to keep them separate, since Cluto gave the same results whether they were merged or not. An example of this is the merging of the 2 to 3 hour bucket with the 3 to 4 hour bucket to form the 2 to 4 hour bucket. The resulting set of 15 buckets includes 0 to 1 hour, 1 to 2 hours, 2 to 4 hours, 6 to 9 hours, 15 to 16 hours, 21 to 23 hours, 23 to 24 hours, 47 to

[1] The number of characteristics that must be included to achieve the same result as using the full set of characteristics depends on how many clusters are being formed and the relative strength of the characteristics chosen. In the PlanetLab dataset used for this thesis, the dominant characteristics are influential enough that it is possible to form many clusters with little more than 10 characteristics and get the same results as if the entire set had been used.

48 hours, 71 to 72 hours, 116 to 117 hours, 190 to 191 hours, 217 to 218 hours, 311 to 312 hours, 491 to 492 hours, and 678 to 679 hours.

With the small set of important buckets, we performed a number of clustering runs, for which we configured Cluto to form an incrementally increasing number of clusters with each proceeding run. Figures 3.5 through 3.13 chart the results, with each progressive chart showing the outcome of a run in which the number of clusters formed was one more than in the previous. The x-axis in the charts is comprised of the buckets used for clustering and the y-axis is the average number of entries in a bucket for all nodes in a given cluster. The clustering parameters used to generate these results included direct clustering for the clustering algorithm, correlation coefficient for the similarity measurement, and a clustering goodness criterion that tries to balance intra-cluster similarity with inter-cluster differentiation (denoted as "H2" in the Cluto documentation [12]). The direct clustering algorithm works by starting with a set of randomly formed clusters and iteratively tweaking the node assignment for each cluster to improve the quality of the clustering as measured by the clustering goodness criterion. This is as opposed to repeated bisections, which starts with a single cluster of all nodes and makes bisections until the desired number of clusters is reached. The direct clustering algorithm is considered to be a good choice when the goal is to form a relatively small number of clusters [12]. Intuitively, this algorithm benefits from being able to run many iterations of refinement, while algorithms like repeated bisections make choices early on that may lead to a suboptimal result at termination.

With the results of our clustering runs, we must choose which best represents the behavioral patterns we can expect from the nodes in the system. There is no single correct answer to this problem, so choosing is a reasoning task. Clearly, we want the nodes in any given cluster to adhere well to the particular set of characteristics which define that cluster. This is easily achieved by using a large number of clusters. However, since we are looking for general behaviors, we also want to minimize the number of clusters used to represent node behaviors. Thus, we must reason about which clustering best satisfies both of these goals.

3.4 Selecting a Representative Cluster

To help select a clustering, we compare the behaviors exhibited by the clusters in Figures 3.5 through 3.13, which show the average number of entries in each bucket for each cluster. We extend this with an examination of Figures 3.14 through 3.22, each of which shows the standard deviation of the number of entries in each bucket for each cluster. These standard deviation values act as an indication of behavioral cohesiveness among the nodes of a cluster. The sizes of the various clusters from each run are reported in Table 3.1. We use this table to help justify whether the clusters formed during each progressive run show behaviors that are worth distinguishing from the clusters of the previous run. The same clustering data was used to generate the averages in Figures 3.5 through 3.13, the standard deviations in Figures 3.14 through 3.22, and the cluster sizes in Table 3.1.

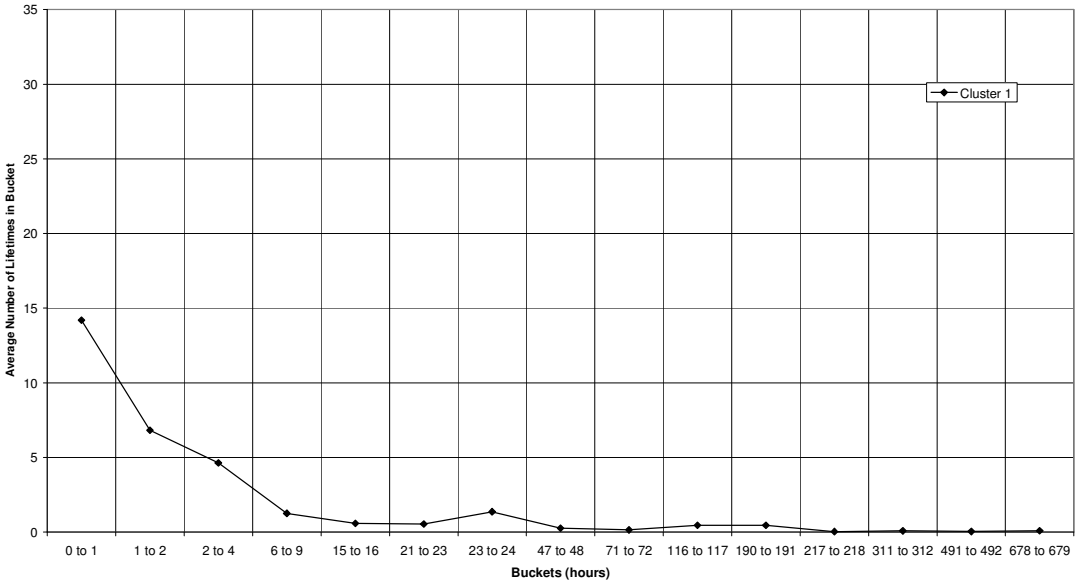

Figure 3.5: The distribution produced by a clustering run parameterized to group nodes into a single cluster. This run makes no distinction between behavioral patterns and shows aggregate behavior for the entire set of nodes.

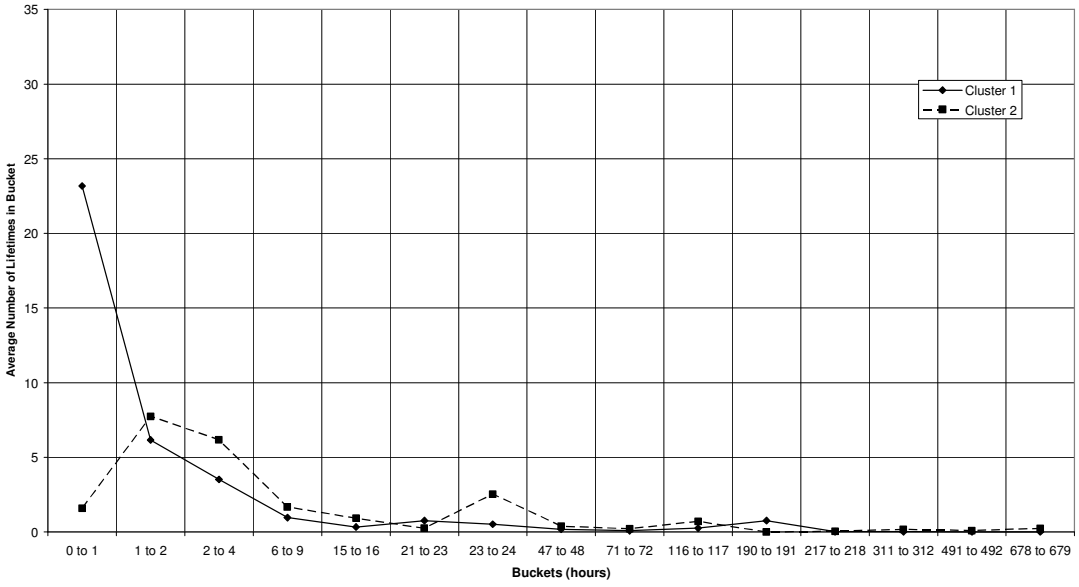

Figure 3.6: The distribution produced by a clustering run parameterized to group nodes into two clusters. While there are some similarities between the behavioral patterns of the two groupings, they bear traits that solidly distinguish them from one another.

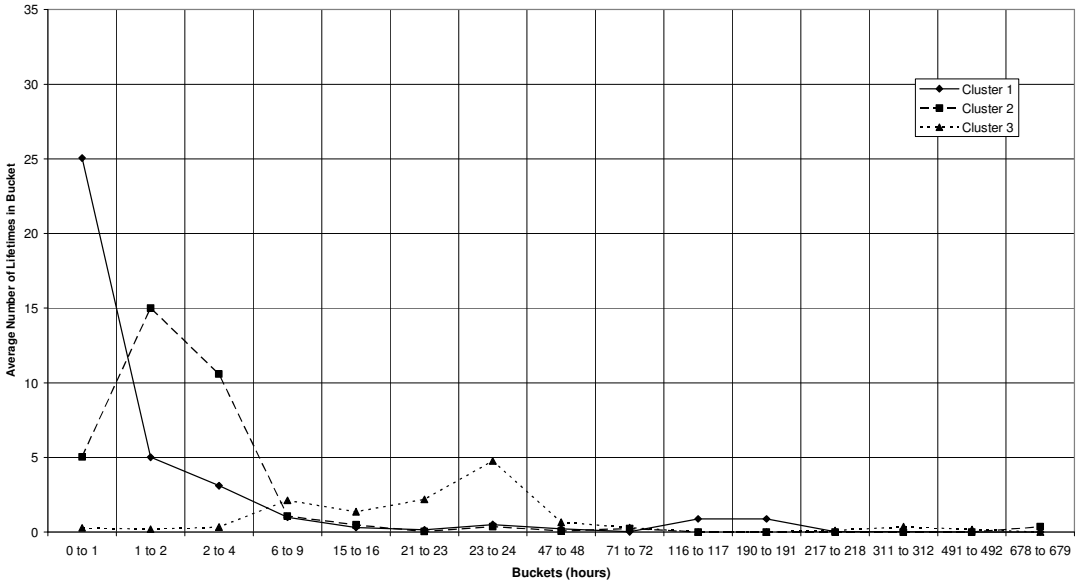

Figure 3.7: The distribution produced by a clustering run parameterized to group nodes into three clusters. Here we see three very unique behavioral patterns. At this point, we can see that nodes can be grouped according to patterns that are a much closer match to their individual behaviors.

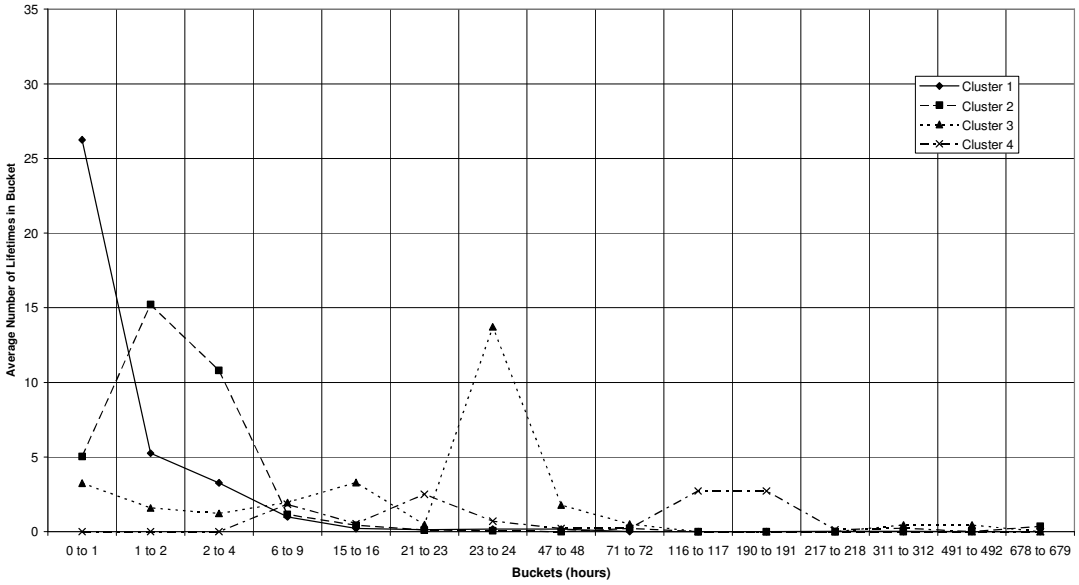

Figure 3.8: The distribution produced by a clustering run parameterized to group nodes into four clusters. We continue to see the emergence of distinct clusters.

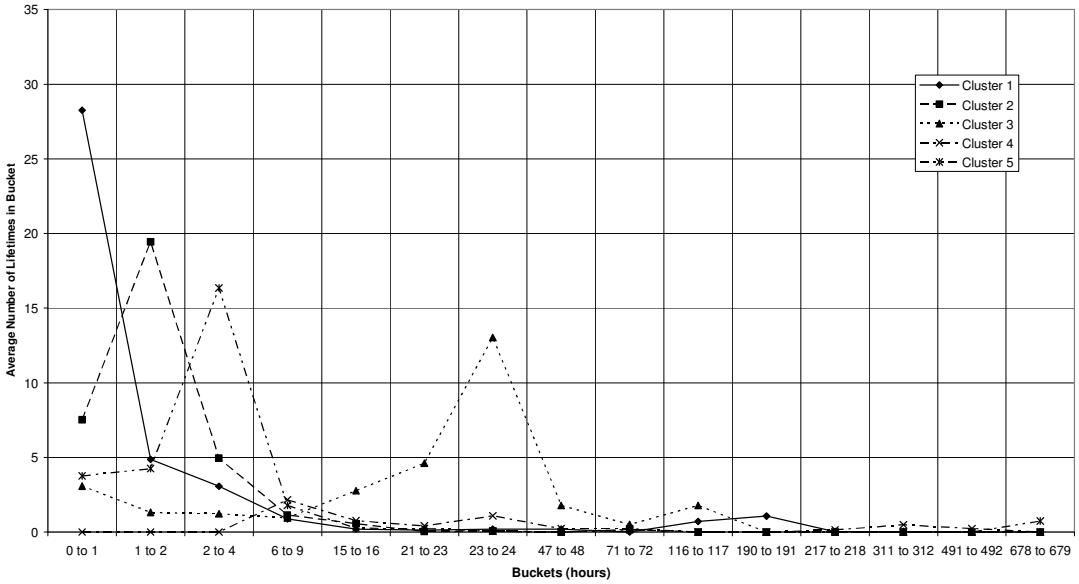

Figure 3.9: The distribution produced by a clustering run parameterized to group nodes into five clusters. Increasing the number of clusters continues to have significant impact, as observed by comparing to the four-way clustering in Figure 3.8.

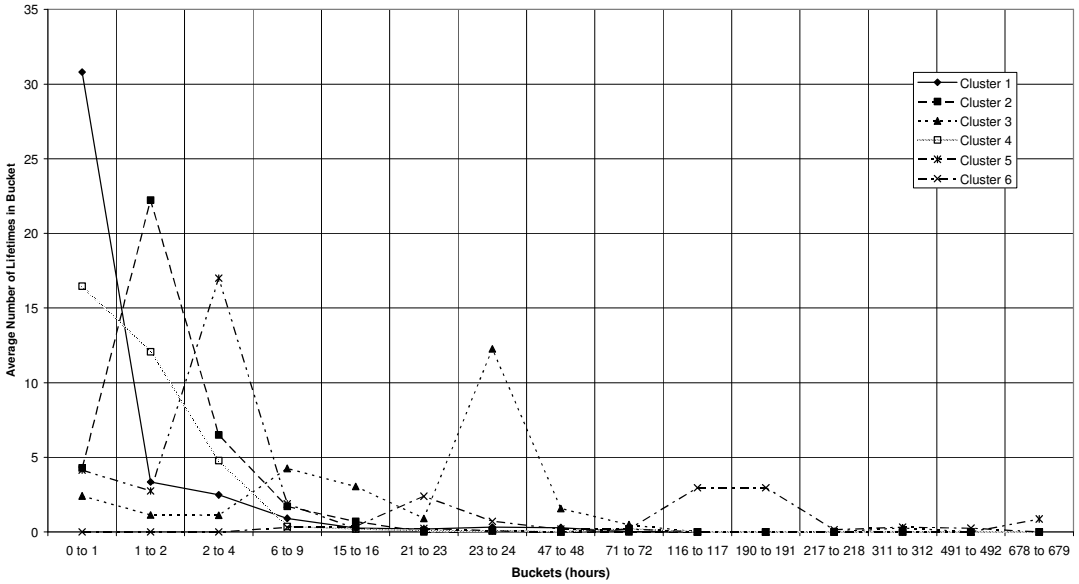

Figure 3.10: The distribution produced by a clustering run parameterized to group nodes into six clusters. At this point, the existing clusters seem to be a little more stable. Visual comparison to the five-way clustering in Figure 3.9 shows less evidence of change in existing clusters, though the emerging cluster in this figure does show distinct behavior.

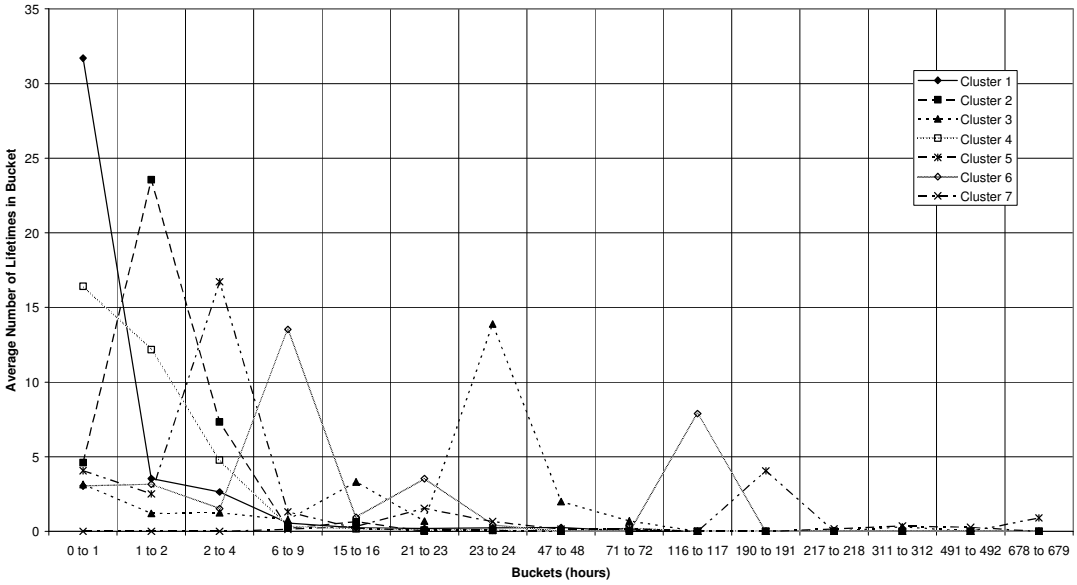

Figure 3.11: The distribution produced by a clustering run parameterized to group nodes into seven clusters. We see some change with the additional cluster, though many of the behavioral groupings from the runs with fewer clusters remain much the same.

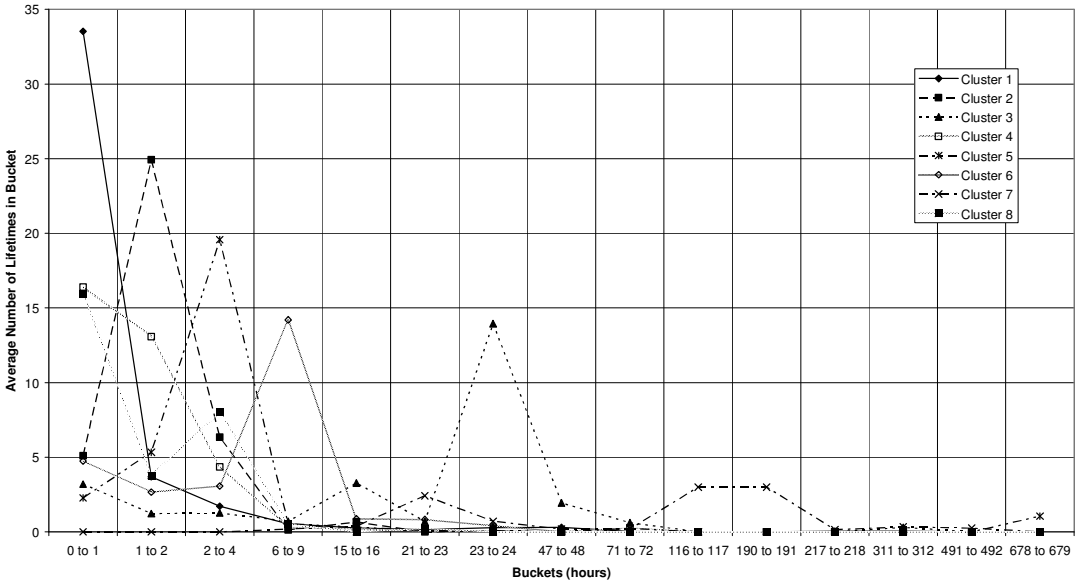

Figure 3.12: The distribution produced by a clustering run parameterized to group nodes into eight clusters. Many of the clusters in this clustering are very similar to those in previous clusterings, an indication that the added cluster isolates overly refined behaviors.

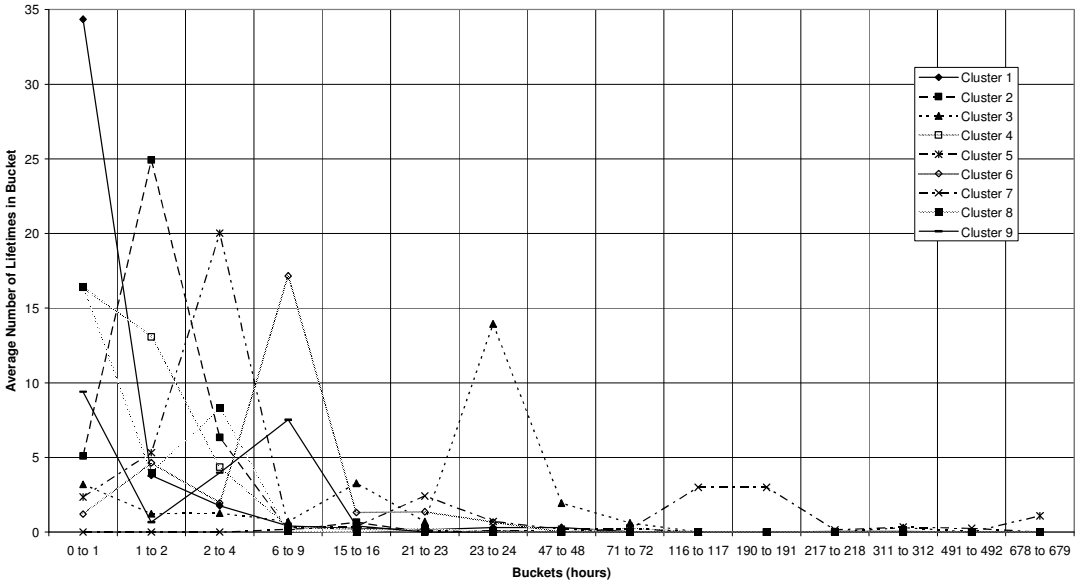

Figure 3.13: The distribution produced by a clustering run parameterized to group nodes into nine clusters. As in Figure 3.12, we again see little change to the clusters seen in previous runs.

Number of Clusters	Sizes of Clusters	Mean Percent-of-Nodes Cluster Size	Standard Dev. of Percent-of-Nodes Cluster Size
1	669 (100%)	100	-
2	390 (58%), 279 (42%)	50	11.31
3	339 (51%), 189 (28%), 141 (21%)	33.3	15.70
4	319 (48%), 56 (8%), 184 (28%), 110 (16%)	25	17.40
5	280 (42%), 90 (14%), 56 (8%), 141 (21%), 102 (15%)	20	13.13
6	218 (33%), 98 (15%), 77 (12%), 61 (9%), 114 (17%), 101 (15%)	16.7	8.40
7	210 (31%), 89 (13%), 55 (8%), 74 (11%), 111 (17%), 38 (6%), 92 (14%)	14.3	8.24
8	77 (12%), 184 (28%), 63 (9%), 54 (8%), 41 (6%), 57 (9%), 93 (14%), 100 (15%)	12.5	6.93
9	77 (12%), 178 (27%), 61 (9%), 25 (4%), 54 (8%), 29 (4%), 52 (8%), 93 (14%), 100 (15%)	11.1	7.09

Table 3.1: Characteristics of the clusters generated during our study of PlanetLab nodes. Notice that the relative sizes of the clusters in the six-way clustering are significantly more balanced than the runs with fewer clusters. However, beyond six clusters, the balance does not substantially improve and we find the introduction of much smaller clusters, which means that the additional clusters may be less helpful in identifying shared behaviors.

While we will discuss later whether the behaviors isolated by each progressive run are worth isolating, it is interesting to observe in Figures 3.5 through 3.13 that each increment in the number of clusters introduces a "new" cluster with distinct behavior. To be more specific, when we increase the number of clusters in our clustering run from n to $n+1$, the additional cluster in the run with $n+1$ clusters is characterized by different behavior than the clusters seen in the run with n clusters. Further, the remaining clusters in the run with $n+1$ clusters are quite similar to those in the run with n clusters. As an example of this, observe that Cluster 1 in Figure 3.5 appears to exhibit very similar characteristics to Cluster 1 in Figure 3.6. Thus, in the latter, we have the appearance of a "new" cluster (Cluster 2) with unique behavior, while the original cluster (Cluster 1) becomes more cohesive.

Although Figures 3.5 through 3.13 are quite useful in there own right, they only show the average number of entries the nodes in each cluster have in each bucket. This does little to provide any indication about whether the nodes in a cluster adhere well to that cluster's defining traits. Thus, we instead refer to the standard deviation charts in Figures 3.14 through 3.22 for guidance in identifying clusters that might have subsets of nodes that do not adhere well to the general behavior of their respective clusters. We illustrate this with the observation that in the 1-way clustering from Figure 3.5, the average number of entries in the 23-to-24-hours bucket is not particularly high (1.4 entries). However, the complementary standard deviation chart in Figure 3.14 shows a noticeable level of variation in this bucket (5.5 entries), which tells us that there may be some subset of nodes within the cluster with a high degree of

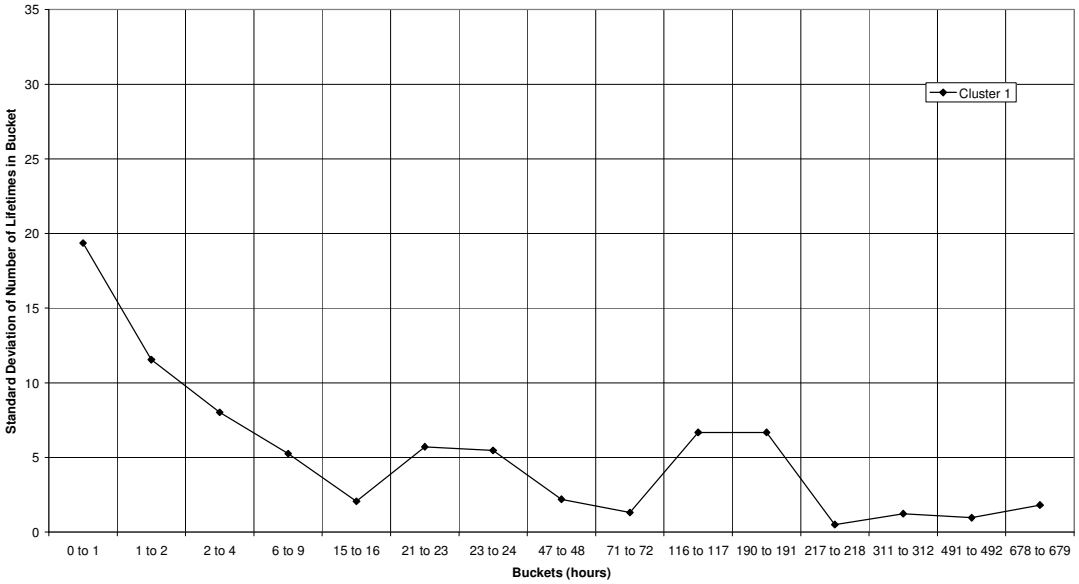

Figure 3.14: The standard deviation within each bucket for the 1-way clustering of Figure 3.5.

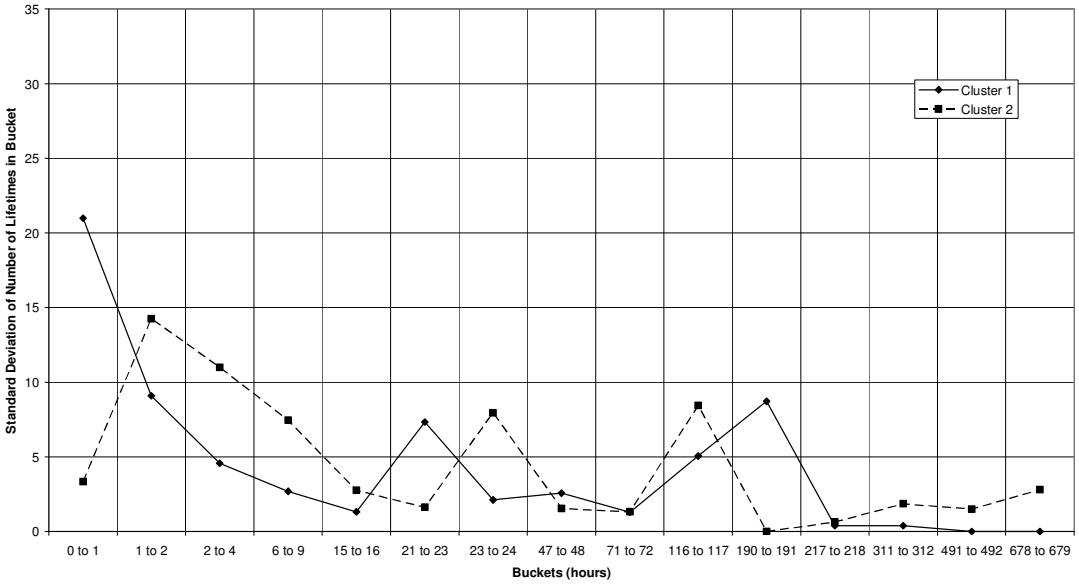

Figure 3.15: The standard deviation within each bucket for the 2-way clustering of Figure 3.6.

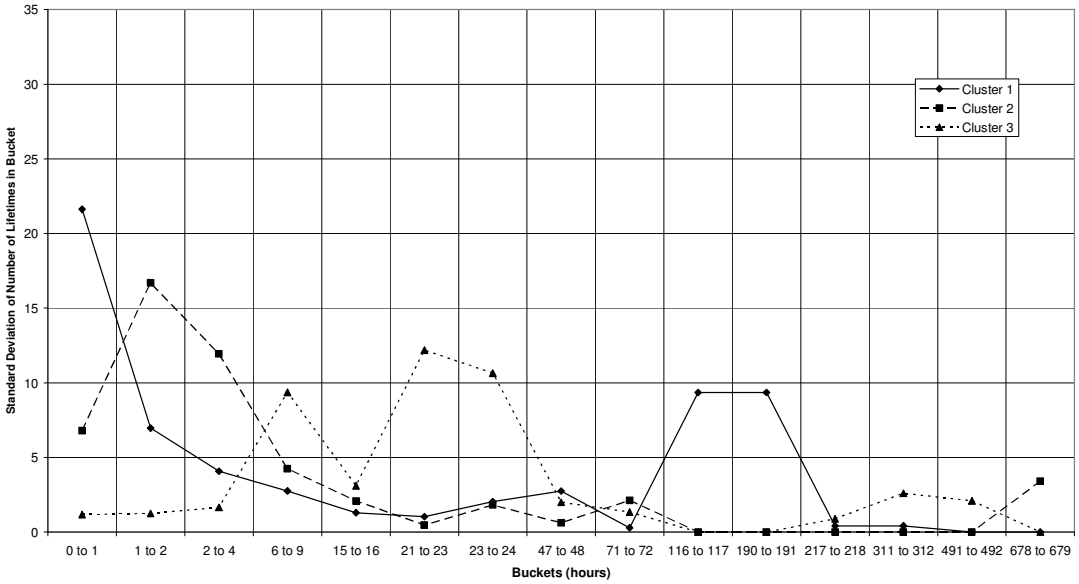

Figure 3.16: The standard deviation within each bucket for the 3-way clustering of Figure 3.7.

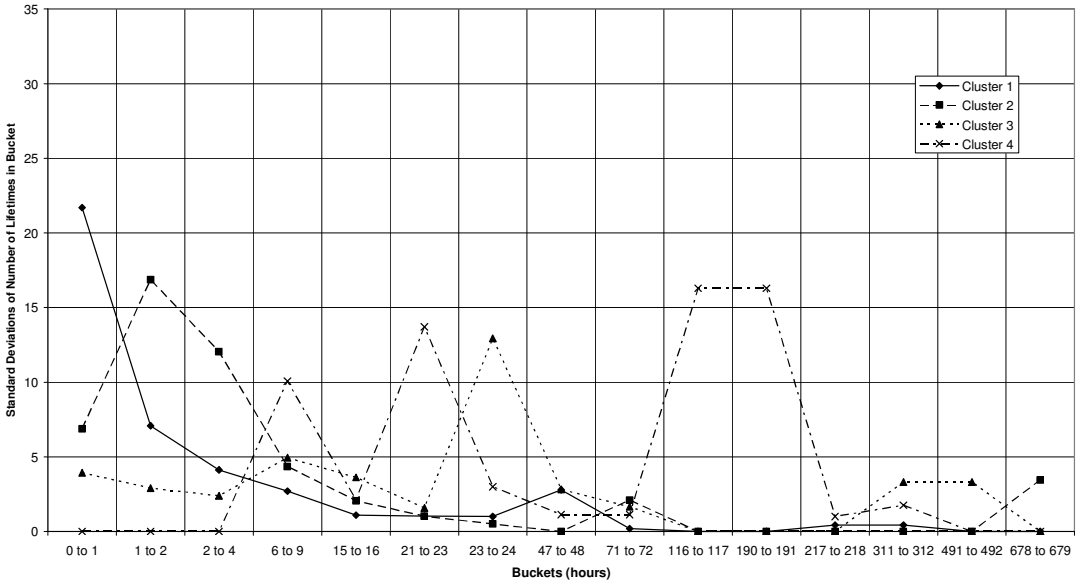

Figure 3.17: The standard deviation within each bucket for the 4-way clustering of Figure 3.8.

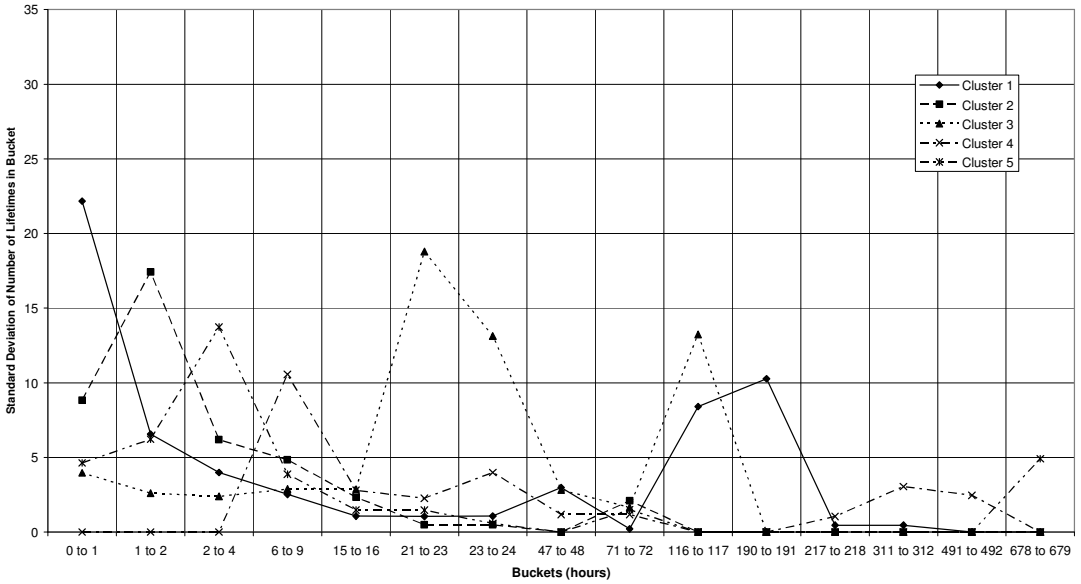

Figure 3.18: The standard deviation within each bucket for the 5-way clustering of Figure 3.9.

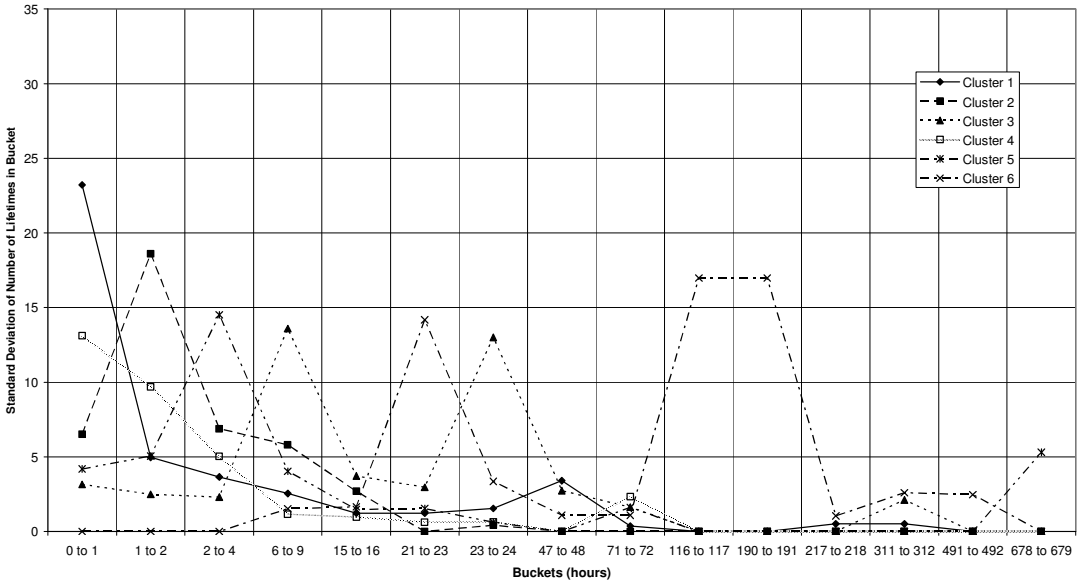

Figure 3.19: The standard deviation within each bucket for the 6-way clustering of Figure 3.10.

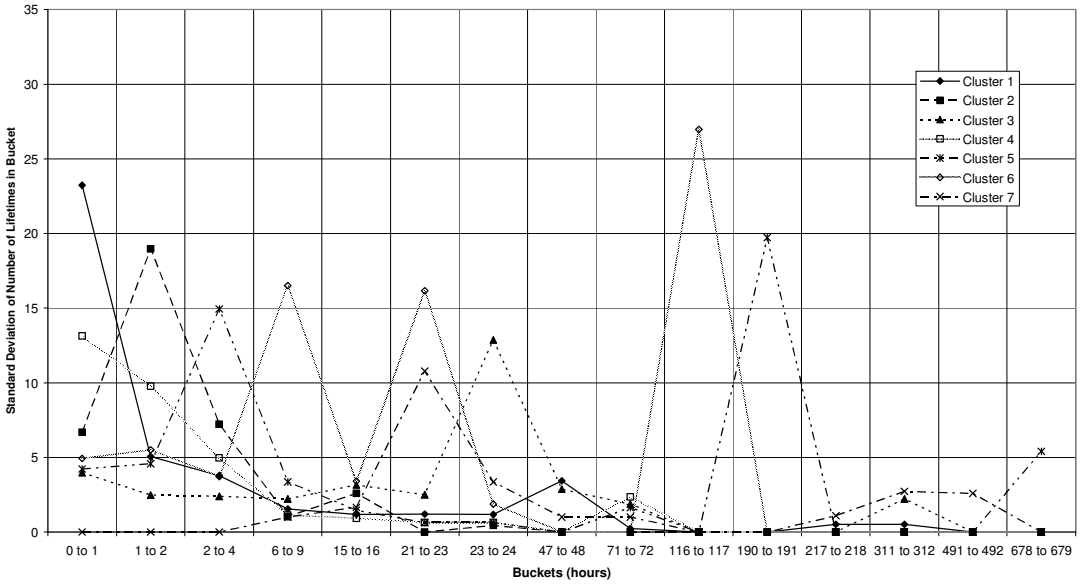

Figure 3.20: The standard deviation within each bucket for the 7-way clustering of Figure 3.11.

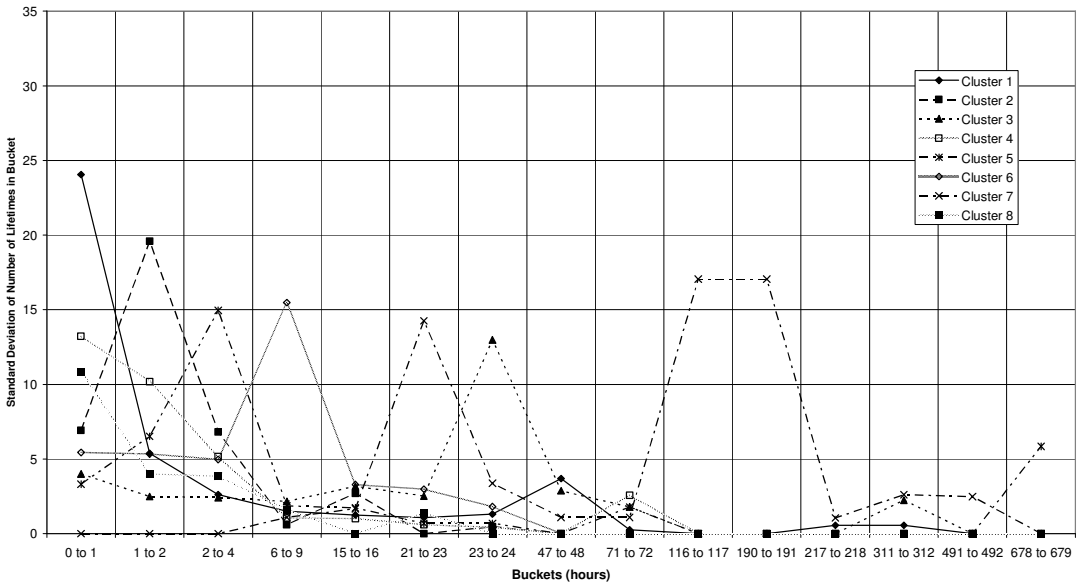

Figure 3.21: The standard deviation within each bucket for the 8-way clustering of Figure 3.12.

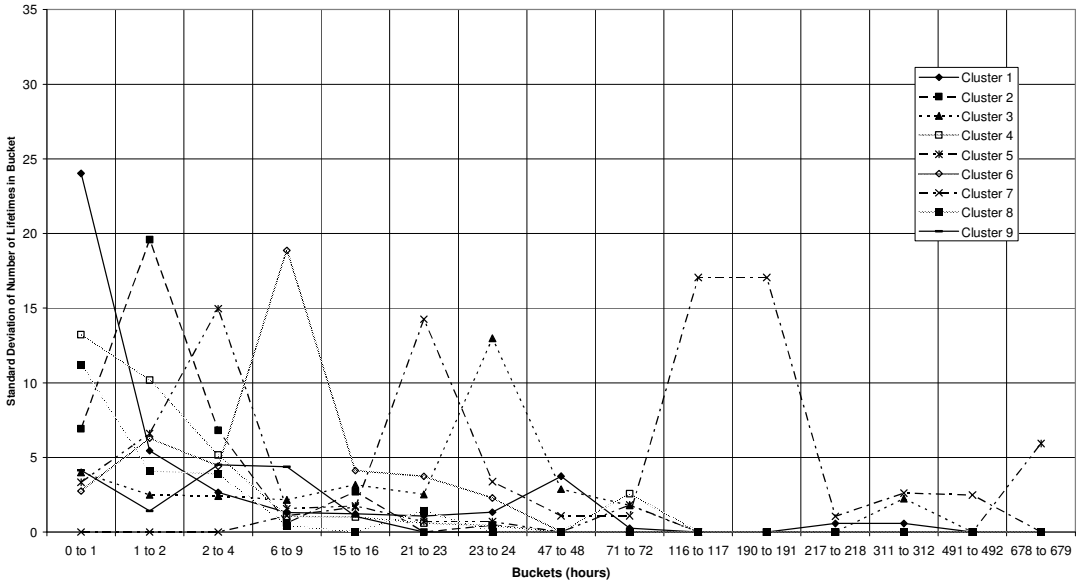

Figure 3.22: The standard deviation within each bucket for the 9-way clustering of Figure 3.13.

concentration in the bucket. Essentially, we are observing that the standard deviations chart has a different shape than the averages chart.

Moving on to Figure 3.6, we see that the set of nodes in the newly formed cluster, Cluster 2, has a higher average number of entries in the 23-to-24-hours bucket (2.5 entries) than the single cluster in Figure 3.5. At the same time, the removal of these nodes from the original cluster, Cluster 1, causes the cluster's average number of entries in that bucket to drop (to 0.5 entries). The simultaneous occurrence of these changes indicates that the 23-to-24-hours bucket is a distinguishing characteristic between the two clusters. We interpret from this that both of these clusters are meaningful and that dividing the original cluster was progressive rather than arbitrary.

To add support to the reasoning about whether or not going from *n* to *n+1* clusters supports our goals, we refer to Table 3.1, where we find both absolute and relative cluster sizes. Continuing with our example of the transition from 1 cluster to 2 clusters, we find that both clusters in the two-way clustering are fairly sizeable. Each holds over 40% of the available nodes. If providing for an additional cluster only allows us to isolate a very small percentage of nodes, the move may be distinguishing behaviors that are of little interest since our aim is to distinguish broad, group-wise characteristics. On the other hand, we must be careful to observe whether isolating the small set of nodes has a significant impact on the remaining clusters. If that is the case, then allowing the nodes to separate into their own cluster is clearly beneficial.

To further demonstrate our reasoning process, we observe the effect of increasing the cluster count from 2 to 3. By comparing the 2 clusters of Figure 3.6 to the 3 clusters of Figure 3.7, we find that the more prominent characteristics of Cluster 2 in Figure 3.6 have been divided between Cluster 2 and Cluster 3 in Figure 3.7. The significance of this change is evidenced in Table 3.1, where we find that no cluster in the 3-way clustering has less than 20% of all nodes.

Using the logic and process described above, we identified the 6-way clustering shown in Figure 3.10 as being the most suitable description of general node behavior for the PlanetLab dataset. Our reasoning is as follows. Comparing the 6-way clustering in Figure 3.10 with its respective standard deviation chart in Figure 3.19, we find that the shape-wise patterns are fairly similar. This supports the idea that there is little cause to further increase the number of clusters since the general behavioral

characteristics of the nodes are similar. Visual inspection of Figures 3.5 through 3.13 also indicates that the dominant sets of clusters have already been mostly established and tend to be persistent beyond 6 clusters.

Table 3.1 shows that beyond six clusters, adding additional ones tends to result in several clusters that are of much smaller size relative to the other clusters and to the mean cluster size. This is an indication that the behaviors in the additional clusters are quite specific and applicable to only a small set of nodes. As we have stated, we want to capture general behavioral patterns with as few clusters as possible. Observe in Table 3.1 that the smallest cluster in the 7-way clustering (38 nodes) is 33% smaller than the smallest cluster in the 6-way clustering (61 nodes). The 7-way clustering's smallest cluster is also around 42% of the mean cluster size for that run, whereas the 6-way clustering's smallest cluster is around 54% of the respective mean.

We also find that that the 6-way clustering is a better choice than the 5-way clustering. There is a significant difference between the 5-way clustering's behavioral chart in Figure 3.9 and the 6-way clustering's behavioral chart in Figure 3.10. In the 6-way clustering, a new behavioral pattern that is substantially different in nature from those of the 5-way clustering is introduced as Cluster 4. This cluster has its main area of concentration spread over the first three buckets. We also find from Table 3.1 that the largest cluster in the 5-way clustering (280 nodes) is 28% larger than that in the 6-way clustering (218 nodes). To get a sense for what this means, one can simplify the effect of the change as the added cluster allowing a significant portion of the large

cluster's nodes to be moved to clusters with more suitable behavioral patterns.[2] The smallest cluster in the 5-way clustering (56 nodes) is actually smaller than that in the 6-way clustering (61 nodes). The large reduction in the standard deviation of the cluster size is evidence of the how incrementing the number of clusters affected the balance of nodes among those clusters. The standard deviation of the mean cluster size for the 6-way clustering is 36% smaller than the standard deviation for the 5-way clustering, while the mean cluster size is only 17% smaller.

If the decrease from 6 clusters to 5 clusters overly-generalizes behavioral characteristics, then the decrease from 6 clusters to 4 will do so even more. This is because at the extreme, the 5-way clustering will differ from the 4-way cluster by introducing a cluster with a single node, while the remaining clusters go basically unchanged. Given that we are looking at 669 nodes in total, isolating a single node into its own cluster when there are so few clusters indicates that the node is very exceptional and represents a special case. Therefore, this is a case in which the 4-way clustering and 5-way clustering are very nearly equivalent, meaning that finding the 6-way clustering to be more appropriate than the 5-way clustering implies that the 6-way clustering is more appropriate than the 4-way clustering. Likewise, if the increase from 6 to 7 clusters causes over-refinement of the behaviors, the best case is that the level of refinement is very nearly equivalent as the number of clusters increases.

We note that if only a few clusters have been formed from a large set of nodes and a single node has been isolated into its own cluster, one should consider whether

[2] This is a simplification because the large clusters are similar, but not equal. Since clusters are defined by the set of nodes they contain, they are only equal when they contain the exact same set of nodes.

the exceptional node should be excluded from the data set based on the goals of the study and the effect of the exclusion. In this study we did not face any such situation.

3.5 Understanding Observed Behavioral Characteristics of Clusters

Given the results of our behavioral clustering of PlanetLab nodes, we are interested in understanding why certain characteristics were common among nodes. Since there are a fair number of characteristics to look at, we begin by selecting several that we might have anticipated as having been key distinguishers even without prior knowledge of the system. The initial buckets we consider are those covering 23 to 24 hours, 47 to 48 hours, and 71 to 72 hours.

Figure 3.23 provides an enhanced view of the distribution of node lifetimes within the range of 20 to 80 hours. We find that each of the three target periods have a single, heavily-concentrated bucket at the 15-minute granularity, providing some validation for our discussion of tight points of concentration in Chapter 3.2. A single heavily-concentrated bucket at 15-minute granularity means that an exceptionally large number of nodes were characterized by lifetimes falling into that bucket and that the lengths of these lifetimes were all within no more than 15 minutes of one another. By looking at these buckets' contributors, we found that the heavy concentration was a result of many nodes having similar lifetimes rather than a small number of nodes contributing many entries. The 23 to 24 hours bucket has an average of about 2.4 entries from each of 357 unique nodes, the 47 to 48 hours bucket has an average of

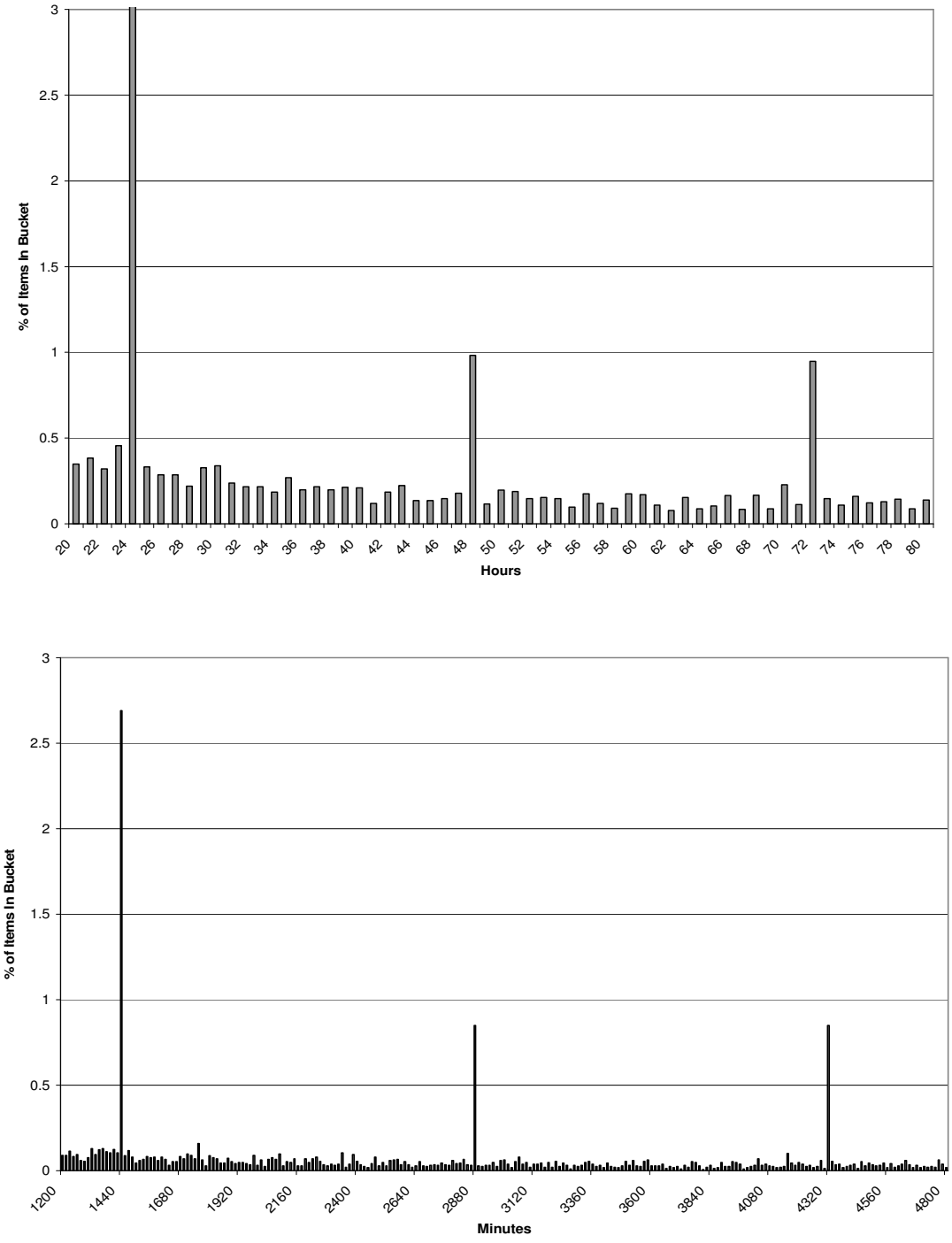

Figure 3.23: Close-ups of Figure 3.3 and Figure 3.4, respectively. These close-ups are focused on the buckets between 20 and 80 hours and include three prominent spikes. These spikes are in the 24-hours bucket, the 28-hours bucket, and the 72-hours bucket. Observe that each of the spikes in the 1-hour buckets corresponds to a single spike in the 15-minute buckets rather than to several over-weighted entries.

about 1.1 entries from each of 259 unique nodes, and the 71 to 72 hours bucket has an average of about 1.1 entries from each of 258 unique nodes.

From these observations, we conclude that these points of concentration are not a product of chance. It is much more likely that they are the result of an influence such as similar administrative policies. Some spikes may also be plausibly attributed to anomalies such as consecutive resource outages, where the first outage causes all affected nodes to synch and the second outage results in lifetime measurements of similar length. One might observe this with temporary network partitions or platform upgrades.

Referring back to Figure 3.3, spikes are also visible for the characteristic buckets at 15 to 16 hours, 217 to 218 hours, 311 to 312 hours, 491 to 492 hours, and 678 to 679 hours.

We can account for the spike at 15 to 16 hours by observing that it resembles expected hours of daytime access, during which a system should be consistently available. This might be, for example, 6am to 10pm. Curiously, the 15-minute bucket that causes the spike in the 1-hour bucket is actually from 930 to 945 minutes (15.5 hours 15.75 hours) instead of being on-the-hour. Although this is somewhat surprising, confidence in the uniqueness of this bucket is reinforced by the fact that the bucket is roughly 10 times more concentrated than nearby buckets. To elaborate, this bucket has 241 entries, while the preceding bucket holds only 9.96% of this value (24 entries) and the proceeding bucket holds only 12.5% of this value (30 entries).

For the various other spikes, explanations can be devised with some thought. We postulate that the overweighting of the 217 to 218 hours (9 days and 2 hours) bucket was caused by multiple PlanetLab outages. A first outage would cause the nodes to synchronize on start time and a second outage would cause the nodes to log the common lifetime duration. This is supported by the fact that 198 unique nodes contributed, but only three contributed more than a single entry. Using Anti-Online's IP Locator [11], we also found that the nodes are in locations as diverse as Berkeley, Boston, Amsterdam, Greece, Taiwan, Australia, and more. For the 311 to 312 hours (13 days) bucket, about 61% of the nodes with entries also had entries in the 217 to 218 hours bucket. This may indicate that some of the nodes from the 217 to 218 hours bucket experienced another outage after the second outage. However, the bucket is right at the end of the 13th day, just shy of 2 weeks. Thus, an alternative explanation is that the additional entries in the bucket may have been the result of maintenance policies in which the last day of a 2-week time-span is spent performing maintenance, requiring nodes to be restarted. The 491 to 492 hours (20 days and 12 hours) bucket is similar in that 70% of the nodes also have entries in the 311 to 312 hours bucket. Interestingly, we again find ourselves just shy of a point with intuitive meaning. A reset at 20 days and 12 hours may indicate restarting machines for maintenance after nearly 3 weeks of lifetime.

In Figure 3.3, we note that a spike exists at 551 to 552 hours (23 days) that was not selected for relevance by the clustering algorithm. Although several nodes share this characteristic, not being selected means that it does not strongly influence the

behavior of nodes. From investigation, we found this bucket has 32 entries coming from a set of nodes in which 82% share one of two three-octet prefixes. This is evidence that the spike was a localized case. We further found that the 15-minute bucket from 22 days and 23.45 hours to 23 days had 30 entries, which hints of a planned reset, perhaps for an upgrade, rather than an unexpected outage. The combination of localization and one-time planned downtime means that the shared characteristic is not a broad behavioral trait, which explains why the clustering algorithm would not have identified it as a relevant characteristic.

The remaining three key characteristics are more interesting. These are 21 to 23 hours, 116 to 117 hours, and 190 to 191 hours. In Figure 3.24, observe that there is no single spike in the range from 21 to 23 hours. Further, the average number of entries in the 15-minute buckets within the range is about 27.9, while the average for the 20 to 21 hours range is 27.5, meaning that the 21 to 23 hours bucket is only slightly over-weighted. Even if we focus on the more weighted 22 to 23 hours range, the average is only 32.75 entries, less than a 20% increase over the average for 20 to 21 hours. This is quite unlike the other cases we have seen, in which a single bucket has many times the number of entries that other nearby buckets do. Despite this, the clustering algorithm chose 21 to 22 hours and 22 to 23 hours as important buckets and we found that combining these two buckets did not affect the results of the clustering. This behavior is likely attributable to the lead-up to 24 hours and loose daily-reset policies, allowing some flexibility on the exact time of the reset. The key observation here is not specifically about the bucket itself, but about the fact that it would have

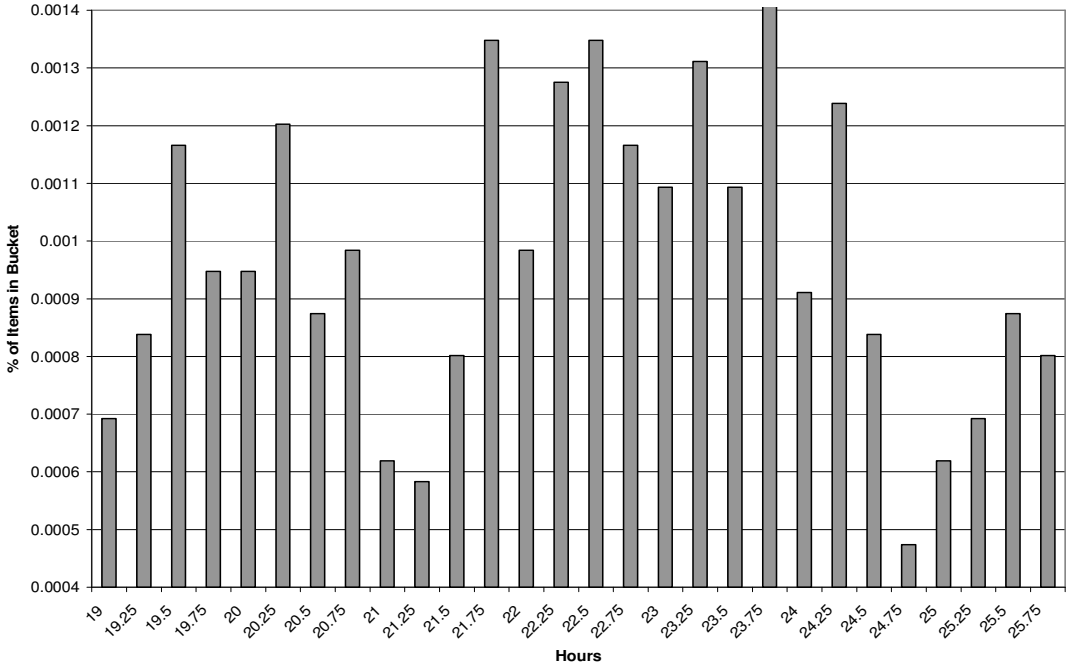

Figure 3.24: A close-up of Figure 3.4, in which we are primarily interested in the range from 21 to 23 hours. Although 21 to 23 hours was selected as a key characteristic by the clustering algorithm, we see that unlike other key characteristics, there was no single distinct spike in this range.

been extremely difficult to identify its importance had we not used clustering.

The distributions of lifetimes near 116 to 117 hours bucket and 190 to 191 hours are shown in Figure 3.25. Neither of these seems particularly noteworthy from looking at the distributions. In the data, we find that only a single node has entries in both the 116 to 117 hours bucket and the 190 to 191 hours bucket. However, many of the nodes with entries in either of these buckets have entries in the 21 to 23 hours bucket. Observing the 6-way clustering in Figure 3.10, we see that all three of these buckets are key characteristics in Cluster 6. It seems that the clustering algorithm identified that it is common for nodes characterized by lifetimes 21 to 23 hours in

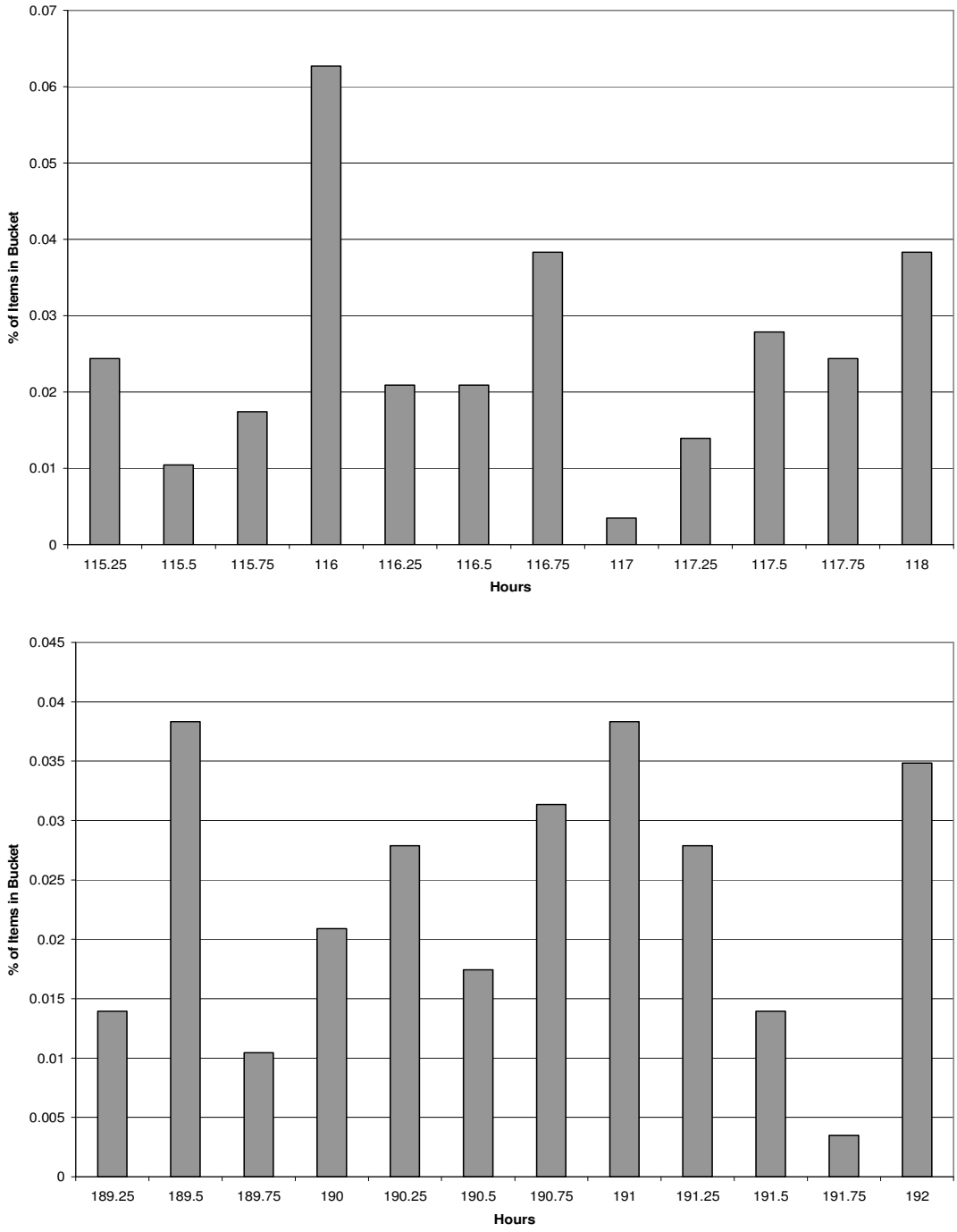

Figure 3.25: Close-ups of Figure 3.4, focused on 115 to 118 hours and 189 to 192 hours, respectively. Neither of these seems noteworthy in isolation, but the clustering algorithm was able to identify a connection between 116 to 117 hours, 190 to 191 hours, and 21 to 23 hours.

length to also be characterized by lifetimes either 116 to 117 hours in length or 190 to 191 hours in length.

At the 1-day granularity, we make some additional observations. There are several buckets of greater weight, which are not explained by spikes in the 15-minute and 1-hour buckets. This indicates that the nodes are behaving similarly, but over a broader period than 1 hour. We see this at 8 days (11520 minutes), which may explain why we do not see a spike around 7 days in the 15-minute and 1-hour buckets. Rather than having lifetimes concentrated into a tight period for the week-long lifetime, they are spread out in the time period between 7 and 8 days.

3.6 Summary of Observations

We summarize here the findings of this chapter. We started by observing that there is no common distribution apparent in individual nodes. The Chi-square goodness of fit test further showed that the exponential distribution is a poor fit for the aggregate distribution of the data.

Rather than trying to find some known distribution that would fit the data well, we formed clusters around the characteristics exhibited by the nodes in the dataset. While increasing the count of clusters that were formed tended to increase the number of unique behavioral patterns identified, the groupings quickly became overly-specific. This resulted in clusters that contained only a very small portion of the nodes. In consideration of both of these concerns, we found 6 to be a well-balanced number of clusters.

In addition to creating clusters, we made an effort to explain the characteristics that served as the basis for forming them. We found evidence of administrative policies, accessibility during work hours, consecutive outages, and localized administration. Clustering gave us the benefit of identifying links between nodes that would otherwise have been difficult to discover. These included patterns that are visually indistinguishable in graphs such as when the behavior is expressed over multiple traits.

Some characteristics we had expected did not end up showing up with 1-hour bucket granularity. However, there did appear to be evidence of these behaviors at the courser 1-day granularity.

Chapter 4

Application

4.1 Behavioral Insight

It is our goal to develop a way in which we can more accurately describe the lifetime behavior of nodes in a distributed system. By clustering nodes, we gain much insight into the nature of their behavioral patterns. We are able to both isolate important commonalities and at the same time filter out anomalous behaviors. To exemplify the importance of filtering out some behaviors, we recall the case of system outages as hypothesized as the cause of the spike in the 217 to 218 hours bucket in Chapter 3.5. A miscellaneous outage will cause the affected nodes to have their lifetime cut short. On the other hand, if power outages are a regular event, this would emerge in the clustering results and may be identifiable to a knowledgeable person. Once the behavioral pattern is identified, the affected nodes are easily identified since they are grouped together.

Having the additional knowledge about the nodes participating in a system is valuable in the respect that it enables more precise design. We consider peer-to-peer systems here.

4.2 "Peer-Based Availability"

In [7], the authors model service availability as peer-based availability. Peer-based availability is availability as measured by a participant in the system, which may appear quite different from measurements made by an outside observer. As a simple example, consider a two-node system in which each node has a measured availability of 50%. If it happens that one node is only available when the other is unavailable, the third-party observer will see 100% availability since it is always the case that at least one node is accessible. On the other hand, if the measurement is taken from one of the two nodes participating in the system, it will instead appear that the system's availability is zero.

Since the process we demonstrated takes behavioral measurements and clusters behavioral patterns, we can apply it to develop a fuller understanding of the peer point-of-view. The analysis presented in this thesis uses an interpretation of the all-pairs-ping dataset from PlanetLab that provides timelines of nodes' lifetimes and downtimes. From this dataset, the lifetime distribution was found for each node. For one interpretation from the peer point-of-view, we would make the slight modification that if a node successfully pinged any other node during a time period, then the pinging node is considered up for that time period. This shifts the view of the system

from the outside observer to the perspective of nodes participating in the system. Clustering behavioral patterns from this data would then reveal which nodes had similar views of the system. Applying this to the two-node example described previously, we would end up with a single cluster of two nodes, each viewing the system as never being available.

4.3 Higher Accuracy Modeling of Node Behavior

As described in Chapter 4.1, the method of analysis described in this thesis is applicable to the numerous cases in systems research and live deployment in which behavioral classification is of value. One of our goals is to better model node behavior. This is achieved by generating behavioral clusters that represent the system of interest. Instead of having a single exponential distribution, we now have several realistic distributions that work together to provide a fuller and more realistic conceptualization. This helps us to attain more accurate results when running models of other system properties, such as resource consumption, that are dependent on this behavior.

4.4 Policy Decisions

Often, systems are designed to be adaptive since it is extremely difficult to handle the effects of dynamic membership [26]. An example of such an adaptive system is the Total Recall storage system [3]. Total Recall allows an administrator to plug in a policy component that tunes the system's behavior based on live measurements so that it can adapt to changes in the system's state.

The authors of [7] found that tuning based on peer-based availability allowed the Total Recall system to achieve a desired service-level agreement with a large reduction in resource consumption. In five test cases, none was more than 0.04% below its respective target. The payoff for this tuning was a reduction in the number of file transfers required to maintain data replication levels by an average of 71.6%. Since system measurements are provided to the component, it would be possible for us to use the method we have described here to make a component that tunes policy based on behavioral classifications.

The ability to apply node classification to policy decisions extends well beyond this tuning application. For example, if we change the clustering algorithm input to be a map linking nodes to the percentage of time they are up during a specific period of the day, we can plan for power consumption. What we are essentially addressing is identifying the level of resource availability needed to meet consumption needs.

Chapter 5

Related Work

Previous studies have shown that for different systems, different models are suitable for representing component lifetimes. Examples of the variety of systems that have been studied include a P2P file-sharing system [5], a student lab at a university [21], worker nodes in a small resource pool [21], Internet hosts [21, 23], and high-performance computing clusters [31, 32]. Although such studies tend to agree that the exponential distribution is typically not a good model for lifetime distribution, the most suitable model depends on the particular case.

Several works have steered away from using well-known distributions like the exponential, instead providing alternative models to represent node lifetimes. In one such study, the authors propose a method to model system-wide churn behavior that arises from individual node behavior, but does not require modeling the behavior of individual nodes [14]. In another, the authors find three noticeable, distinct behaviors in a dataset containing node downtimes measured on a large corporate network [4]. Each of these behaviors can be represented by a well-known distribution. Using this

insight, the authors formed a unique downtime model that was composed of a combination of the three distributions. We point out that identifying such unique behavioral groups is a task well-suited to the clustering techniques described in this thesis.

A number of systems use knowledge of expected lifetimes to make functional decisions. One such system is the Total Recall storage system [3]. Total Recall uses the availability of participating nodes to determine the amount of data redundancy needed to maintain a specific level of availability for the data. To make replication decisions, live measurements of short-term node availability are feed into the redundancy policy, which estimates the likelihood of a set of hosts being available at a given time. In [7], the authors suggest the alternative of quantifying short-term node availability from the viewpoint of the system's participants rather than from the viewpoint of an outsider. The authors show that the number of file transfers required under this alternative scheme is as little as 25% of what was originally thought to be needed to meet the data-availability goal. This seems to be strong evidence of how having a different, perhaps more accurate, model of availability can provide significant benefit.

Instead of modeling availability, it is also possible to simulate it. In [9], the authors present DieCast, which describes a way in which to simulate huge systems in their entirety with only a fraction of the resources needed for actual deployment. The trade-off, however, is that a large amount of time is spent in simulation since the scale-down in resource consumption is achieved through time-dilation.

The authors of [20] explore the state-machine approach to characterizing node lifetimes. Instead of using probabilistic models, they attempt to predict uptimes (i.e. lifetimes) and downtimes using state machines based on branch predication techniques.

With respect to the clustering aspect of our work, an algorithm used to build routing clusters in wireless, ad hoc networks is discussed in [19]. In this algorithm, nodes base cluster membership on the likelihood of routing feasibility. While clusters are formed through selective criteria, the powerful clustering techniques described in this thesis are not used.

Chapter 6

Conclusion

In visually observing the time-to-failure data from the PlanetLab all-pairs-ping dataset, it appeared highly unlikely that the distributions of individual nodes were well-fitted to any particular distribution (Chapter 3.1). Applying a Chi-Square goodness of fit test to an aggregation of the data from the dataset, we found the distribution of this aggregation to be poorly fitted to the exponential distribution, which is sometimes assumed to be an adequate model of the distribution of node lifetimes (Chapter 3.2). In the aggregation, we also observed points of heavy concentration within tight time ranges, indicative of shared behavioral characteristics.

By applying clustering techniques to the dataset, we formed a set of six clusters, each characterized by a distinct behavioral pattern (Chapter 3.4). The standard deviation of cluster size as a percentage of all nodes for this set of clusters was a reasonably small 8.4%, indicating that the clusters were neither overly-specific nor overly-generic. We found that forming more than six clusters results in overly-specific clusters, while forming less than six clusters fails to isolate some important

behavioral patterns that are evident with six clusters.

The behavioral identites of clusters formed in this study hinge on the existence of highly concentrated ranges in the aggregate data. The concentration of lifetimes in some buckets is so high relative to neighboring buckets that it is highly unlikely that the concentrations are a product of chance. Among the concentrations, we were able to reason that various points had arisen from occurrences such as similar administrative policies, consecutive service outages, and localized reset of large numbers of nodes (Chapter 3.5).

While some distinguishing characteristics of the behavioral clusters, such as favoritism toward a single trait, might have been identifiable without the use of clustering, the clustering process was able to work out more complex patterns across multiple traits such as slightly above-average weighting in four or more buckets. In general, we were able to use the clustering algorithm to identify subtle patterns that might otherwise have been missed since they are not obvious in distributions of node lifetimes.

Bibliography

1. R. Bhagwan, D. Moore, S. Savage, and G. M. Voelker. Replication Strategies for Highly Available Peer-to-Peer Storage. In *Proceedings of FuDiCo: Future directions in Distributed Computing*, June, 2002.

2. R. Bhagwan, S. Savage, and G. M. Voelker. Understanding Availability. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.

3. R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. M. Voelker. Total Recall: System Support for Automated Availability Management. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, 2004.

4. W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs. In *ACM SIGMETRICS Performance Evaluation Review, Volume 28, Issue 1*, 2000

5. F. E. Bustamante and Y. Qiao. Friendships that Last: Peer lifespan and its role in P2P Protocols. In *Web Content Caching and Distribution: Proceedings of the 8th International Workshop*, 2004.

6. A. Datta, K. Aberer. Internet-scale storage systems under churn - A study of the steady-state using Markov models. In *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, 2006.

7. R. J. Dunn, J. Zahorjan, S. D. Gribble, and H. M. Levy. Presence-Based Availability and P2P Systems. In *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*, 2005.

8. B. Godfrey. Repository of Availability Traces. Available online at *http://www.cs.berkeley.edu/~pbg/availability/*.

9. D. Gupta, K. Vishwanath, and A. Vahdat. DieCast: Testing Distributed Systems with an Accurate Scale Model. In *Proceedings of the 5th ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

10. T. Heath, R. P. Martin, and T. D. Nguyen. The Shape of Failure. In *Proceedings of the First Workshop on Evaluating and Architecting System dependabilitY (EASY)*, 2001.

11. IP Locator. http://www.antionline.com/tools-and-toys/ip-locate/.

12. G. Karypis. CLUTO – A Clustering Toolkit. *Technical Report #02-017, Department of Computer Science, University of Minnesota*, 2003.

13. L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc., 1990.

14. S. Y. Ko, I. Hoque, and I. Gupta. Using Tractable and Realistic Churn Models to Analyze Quiescence Behavior of Distributed Protocols. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2008.

15. J. Y. B. Lee and R. W. T. Leung. Design and analysis of a Fault-Tolerant Mechanism for a Server-less Video-on-Demand System. In *Proceedings of the 9th International Conference on Parallel and Distributed Systems (ICPADS)*, 2002.

16. D. Leonard, V. Rai, and D. Loguinov. On Lifetime-Based Node Failure and Stochastic Resilience of Decentralized Peer-to-Peer Networks. In *IEEE/ACM Transactions on Networking (TON), Volume 15, Issue 3*, 2007.

17. W. K. Lin, D. M. Chiu, and Y. B. Lee. Erasure Code Replication Revisited. In *Proceedings of the Fourth International Conference on Peer-to-Peer Computing*, 2004.

18. D. D. E. Long and Jehan-François Pâris. On Improving the Availability of Replicated Files. In *Proceeding of the Sixth Symposium on Reliability in Distributed Software and Database Systems*, 1987.

19. A. B. McDonald and T. F. Znati. Design and Simulation of a Distributed Dynamic Clustering Algorithm for Multimode Routing in Wireless Ad Hoc Networks. *Simulation: Transactions of the Society for Computer Simulation and Modeling International, Special Issue on Simulation and Modeling of Computer Systems and Networks*, Vol. 78, No. 70, July 2002, pp. 408-422.

20. J. W. Mickens and B. D. Noble. Exploiting Availability Prediction in Distributed Systems. In *Proceedings of the 3rd Symposium on Networked Systems Design & Implementation (NSDI) - Volume 3*, 2006.

21. D. Nurmi, J. Brevik, and R. Wolski. Modeling Machine Availability in Enterprise and Wide-area Distributed Computing Environments. In *Proceedings of European Conference on Parallel Computing (EUROPAR) 2005*, 2005.

22. PlanetLab – An open platform for developing, deploying, and accessing planetary-scale services. Available online at *http://planet-lab.org/*.

23. J. S. Plank and W. R. Elwasif. Experimental Assessment of Workstation Failures and Their Impact on Checkpointing Systems. In *Proceedings of the Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing*, 1998.

24. D. Powell. Failure Mode Assumptions and Assumption Coverage. In *Proceedings of the 22nd Annual Symposium on Fault-Tolerant Computing (FTCS)*, 1992.

25. S. Ramabhadran and J. Pasquale. Analysis of Long-Running Replicated Systems. In *Proc. of the 25th IEEE Annual Conference on Computer Communications (INFOCOM)*, 2006.

26. S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling Churn in a DHT. In *Proceedings of the 2004 USENIX Annual Technical Conference*, 2004.

27. S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and John Kubiatowicz. Maintenance-Free Global Data Storage. In *IEEE Internet Computing, Volume 5, Issue 5*, 2001.

28. J. L. Romeu. The Chi-Square: a Large-Sample Goodness of Fit Test. *RAC START, Volume 10, Number 4*, 2004.

29. S. Ross. *A First Course in Probability, Sixth Edition*. Pearson Education, Inc., Dehli, India, 2002.

30. S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of Multimedia Computing and Networking (MMCN) 2002*, 2002.

31. B. Schroeder and G. A. Gibson. A large-scale study of failures in high-performance computing systems. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, 2006.

32. B. Schroeder and G. A. Gibson. Disk failures in the real world. In *Proceedings of the 5th USENIX conference on File and Storage Technologies*, 2007.

33. J. Tian and Y. Dai. Understanding the Dynamic of Peer to Peer Systems. In *Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS '07)*, 2007;

34. H. Weatherspoon, B. Chun, C. W. So and J. Kubiatowicz. Long-Term Data Maintenance in Wide-Area Storage Systems: A Quantitative Approach. *Technical Report UCB/CSD-05-1404, EECS Department, University of California, Berkeley*, 2005

35. H. Weatherspoon and J. D. Kubiatowicz. Erasure Coding vs. Replication. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, 2002.

36. P. Yalagandula, S. Nath, H. Yu, P. B. Gibbons, and S. Seshan. Beyond Availability: Towards a Deeper Understanding of Machine Failure Characteristics in Large Distributed Systems. In *Proceedings of the First Workshop on Real, Large Distributed Systems (WORLDS)*, 2004.

37. C. Yoshikawa. All-Sites-Pings for PlanetLab. Available online at *http://ping.ececs.uc.edu/ping/*.