PROC. IST IEEE WORKSHOP ON NETWORKING AND DATABASES, 2005

On stationarity in Internet measurements through an information-theoretic lens

Balachander Krishnamurthy
AT&T Labs – Research

Harsha V. Madhyastha
University of Washington

Suresh Venkatasubramanian
AT&T Labs – Research

Abstract

Streaming change detection schemes are of great interest in the context of data warehouses, data cleaning systems, network traffic anomaly detection and other measurement scenarios. The goal of change detection systems is to determine whether fundamental characteristics of a data stream have changed, either temporally (with respect to prior data) or spatially (with respect to other streams).

We present an application of an information-theoretic change detection scheme to determining stationarity in Internet measurements. Our experiments indicate that this generic scheme, despite being oblivious to the nature of the data, matches a more domain specific approach for this problem, and requires no domain knowledge to work effectively.

1. Introduction

Change detection schemes, whether they be intrusion detection mechanisms, network traffic alarm systems, or data cleaning schemes, have been studied extensively in specific domains. In recent years, there has been some interest in the databases and data mining community in developing generic change detection schemes that take as input a stream of records of varying types (categorical, numerical, multidimensional), and use *non-parametric* methods to determine whether intrinsic qualities of the data have changed.

The focus on generalness is important so that the methods developed can be applied to a wide range of applications. Generalness is also crucial in situations where the data stream cannot be modeled effectively using standard models (i.e parametrically). A simple example would be the records that arrive at a data warehouse for storage; often, there are human and machine-induced discrepancies in these records, and we would like to determine such discrepancies in the absence of detailed models describing the data.

In recent work [3], an *information-theoretic approach* to change detection has been developed. The authors define a notion of *distance* between data stream windows based on computing the *Kullback-Liebler* distance (KL-distance) between appropriately defined probability distributions, and use the statistical method of *bootstrapping* to provide a formal probabilistic guarantee for change detection. This approach is non-parametric; it requires *no* assumptions on the data streams. It has also been shown to be effective for a variety of multidimensional data sets with different notions of change.

This change detection scheme consists of two black boxes; one which computes a distance between two windows, and a second which for statistical significance. These black boxes are independent of each other. Any distance function where a zero implies identity and larger values imply a greater dissimilarity can be used as the first black box without affecting the structure of the method; this is the main advantage of this approach. In addition, because of the modularity of our scheme, domain-specific information could be used to supplement the KL-distance and create a modified distance function that would improve the method. Thus, this approach provides a generic framework for change detection that can be tailored quite easily for specific domains.

There are many change detection schemes in the literature, and a full discussion is beyond the scope of this paper. Our focus here is not on the problem of change detection *per se*, but on its application to the problem of stationarity.

2. Applying change detection to an Internet measurement application

Internet properties are notoriously hard to characterize given the large number of entities, the inherent variability of the measures, and the periodic events that cause instability. We decided to take an existing data set that had been gathered for the triggered measurement project ATMEN [7] and explore the applicability of our change detection procedure for determining stationarity.

The measurement application we chose is a fairly generic one. PlanetLab [1] is rapidly gaining in popularity as a measurement platform on the Internet. Numerous research projects are carried out using subsets of the roughly 350 nodes of PlanetLab. The studies involve series of measurements on different layers of the protocol stack for various applications. Such studies often require repeated probes

from multiple nodes over a period of time that can range from a few hours to a few weeks. The large number of servers involved in typical applications require that extensive measurements be carried out. The measurements involve numerous protocols (DNS, TCP, BGP, HTTP *etc.*) and multiple entities (routers, hosts, DNS and Web servers *etc.*), often for the same application.

A natural question to ask is "can we determine a suitable number of measurements to be obtained?". The answer depends very much on the particular application, the variability involved in the entities measured, and the *overall issue of stationarity along numerous axes*: temporal, spatial, and application-level. For the purpose of this paper, we use the term "stationarity" to refer to properties of a data stream that are (roughly) invariant over time or space. If the characteristics of a certain measurement are relatively stable over time (for example), we can reduce the number of samples required to characterize this measurement. In general, exploiting various axes of stationarity will allow us to reduce the number of measurements made while still being able to detect most changes that occur. However, this requires the characterization of the stationarity exhibited by various measurement parameters. Here, we hope to exploit the K-L technique to answer the temporal aspect of stationarity of download time and its components. Our primary goal in this paper is to see how easy it is to apply the K-L technique as a generic black box and validate its degree of effectiveness. Overall, our goal is to answer the following question: How large is the *semantic gap* between this generic change detection scheme and more domain specific methods in the context of stationarity?

3. Change detection

We briefly describe the KL-distance change detection procedure. It takes as input a stream of data records, where records may have both categorical and numerical attributes. Two *windows*, each of size W, are maintained over the input stream. Change is then expressed as a distance computed between these two windows. Intuitively, the two windows represent a reference pattern and a test pattern; the distance function determines whether the test pattern differs significantly from the reference pattern.

The power and generality of this approach comes from the choice of distance function. Each window of data is viewed as representing samples from an (unknown) probability distribution, and we then define a distance function in terms of these probability distributions. The function we use is called the *Kullback-Liebler* distance, also known as the *relative entropy* [2].

Formally, if we are given two probability distributions p, q over a set of atoms X (thus, $0 < p(x) \leq 1, 0 < q(x) \leq 1, x \in X$, and $\sum_{x \in X} p(x) = \sum_{x \in X} q(x) = 1$), the Kullback-Liebler (KL) distance from p to q is defined as

$$D(p\|q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

The KL-distance is widely used in information theory. In our context, mathematical statistics and learning theory provides a more relevant interpretation of the KL-distance as a natural distance function between probability functions, closely analogous to the sum-of-squares distance between two vectors in a Euclidean space. Note that this measure is asymmetric; $D(p\|q) \neq D(q\|p)$ in general. This is not a serious problem in practice for this particular application.

Given a window, we construct an empirical probability distribution from it by binning the data items. For categorical attributes, the bins are the set of values the attribute can take; for multidimensional numerical attributes, we build a *quad-tree* in the appropriate dimension and use the leaves of the quad tree as bins. Note that the quad tree is actually built on the union of the two windows so that they share the same set of bins (i.e., atoms $x \in X$).

Once this is done, the number of data items in each bin can be normalized to compute an empirical probability distribution, and then the KL-distance can be computed using the formula above (with a correction for the case when a bin count is zero). Updating the distance calculation as the windows slide across the data stream is easy; quad trees can be updated in time proportional to their depth, which in practice is usually logarithmic in the size of the window (the worst case update time is linear in the size of the window, but is rarely encountered in practice).

3.1. Bootstrapping

Once we compute the KL-distance $\hat{d}$ between two windows, we have to evaluate this value for statistical significance. To do this, we use the framework of *hypothesis testing* and *bootstrap sampling*. The null hypothesis is that the two windows arise from the same underlying distribution and thus no change has occurred. A traditional statistical approach would model the distribution of values of the test statistic (in this case the KL-distance) and determine the probability that $\hat{d}$ could have been measured under the null hypothesis. This is very hard in general, and can only be done for structured test statistics like the mean and standard deviation.

We will use a more general data-driven approach called bootstrapping. The idea of bootstrapping is to construct empirically a distribution of the test statistic that reflects the null hypothesis, and use this to determine the significance

of a measurement[1]. Under the null hypothesis, the two windows arise from the same distribution, and thus their mixture also arises from the same distribution. Thus, our bootstrapping procedure is as follows. We combine the data from the two windows and construct an empirical distribution as described above. We then sample $2W$ times (with replacement) from this distribution, and declare the first W items as the first window and the second W items as the second window. We then compute the KL-distance between these two windows, giving us an estimate d_1. We repeat this procedure B times, obtaining estimates $d_1, d_2, \ldots d_B$.

Fixing a significance threshold $\alpha = 99\%$, we then check if the original estimated distance $\hat{d}$ lies outside the α-percentile of the sorted set of d_is (i.e., it is in the top 1% of samples). If so, we declare a change to have occurred, and reset our test window to start at the data item just after the window where this happened. The choice of α represents a tradeoff; the smaller the value of α, the fewer the number of false negatives, but the larger the number of false positives. Experiments on different kinds of data sources indicate that this choice of value represents a reasonable tradeoff between the two kinds of error; true changes are not missed, while keeping the false positive rate as low as possible.

We start the process with the first window at the beginning of the stream and the second window immediately following it. The above procedure describes what happens if a change is detected. If a change is *not* detected, we slide the second window one time step forward, keeping the first one fixed. The rationale for doing this, first suggested in [6], is to prevent small changes from accumulating to cause a big change without our being able to detect it.

4. A Specific Application of Measurement Reuse

Internet events such as worms, virus, and flash crowds mandate constant measurement by network operators. A *distributed set of intercommunicating measurement entities* reacting quickly to events and aiding correlation of application-specific measurements would be useful. The triggered network measurement infrastructure ATMEN allows for measurements to be turned on and off for specific durations of time on a subset of co-operating set of measurement sites based on the occurrence of one or more events. It was learned in constructing ATMEN that the cost of carrying out measurements repeatedly from many probing sites could be reduced by reusing measurements prudently. Most measured parameters exhibit varying degrees of stationarity across time and space. The value of a measurement often remains stable over short intervals of time. Commonalities across clusters of measurement sites may allow the measurements to be reused across space. Applications that share common primitive measurement components can allow other applications to reuse them. For example, many applications on the Internet interested in performance may measure a DNS component which may be reusable across applications if there is no significant variation in the timescale of interest.

We studied reuse potential of measurements in a real-world application – performance-based ranking – by ranking a set of Web sites according to their download time. Download time can be decomposed into individual components of DNS lookup, TCP connection setup, and HTTP transfer time. Such a component-based measurement is widely applicable for a variety of Internet measurements (such as P2P applications). We measure these parameters in this application. Irrespective of the methodology used to analyze temporal and spatial stationarity of download time and its components, making inferences that hold for downloads from any arbitrary Web site on any arbitrary client is hard. We would need data for measurements made from every client in the Internet to all Web sites at every instant in time over an extended period of time. As this is impractical, we gathered an approximation of such an ideal dataset.

We designed the methodology to gather measurements keeping in mind that the data we collect should be representative of the ideal dataset. Any analysis done over a non-representative dataset might yield results that were applicable only for the specific set of client sites and Web sites we chose and the inferences made would not extend to measurements of download time and its components made to other Web sites from an arbitrary client site. Thus, we chose a set of Web sites representative of most Web traffic and performed downloads from a geographically distributed set of client sites.

We chose the 88 most popular Web sites and performed download time measurements by obtaining *index.html* from them. These 88 Web sites were selected by merging several popular Web site rankings – Netcraft [10], Mediametrix [8], and Fortune 500 [5]. To obtain good geographic coverage of client sites, we issued download requests from a set of 66 PlanetLab nodes [1] distributed across North America, Europe and Asia spanning academic institutions, research laboratories and commercial sites.

There are no dialup nodes in PlanetLab and among the two DSL nodes that exist, one of them was nonfunctional during the duration of our experiment. We did make measurements from the other DSL node. However, the data collected from this node showed different trends as compared to the 66 mentioned above and hence was not used as part of this study. We reiterate that no particular choice of client and server sites can be considered to be completely repre-

1 A detailed description of bootstrapping is beyond the scope of this paper; we refer the reader to [4] for more details.

sentative. Our choice of using 66 geographically distributed PlanetLab nodes and the 88 most popular Web sites is a plausible choice given the resources at hand.

Each client site performed a download once every 5 seconds, implying that *index.html* from each of the 88 Websites was downloaded approximately once every 7.5 minutes on each site. To gather measurements over a sufficiently long period of time, we performed this periodic download on each of the 66 client sites for 17 days in July 2004. We used a slightly modified version of *httperf* [9] for the downloads, logging the DNS lookup, TCP connection setup, and HTTP transfer times.

5. Our experimental setup

We next present the results of applying our change detection procedure on this data to analyze the temporal stationarity of download time and its components. As IP addresses of PlanetLab nodes and Web site names hold no significance for the change detection algorithm, we enumerated the 66 PlanetLab nodes and 88 Web sites in a random order and mapped them to the sets $[0..65]$ and $[0..87]$, respectively. For each pair of (PlanetLab node,web site), the set of measurement values were normalized by dividing by the maximum value of the particular type of measurement. As outlined in the previous section, our change detection procedure operates on a stream of records. Hence, the input to the change detection algorithm was a sequence of normalized values for each (i, j) tuple, where $i \in [0, 65]$ and $j \in [0, 87]$, with one sequence each for DNS lookup time, TCP connection setup time, HTTP transfer time and total download time.

For each tuple (i, j), we recorded the number of changes detected by the change detection scheme. We then computed the standard deviation of these changes for a fixed Planetlab node i, over alll websites j. Each of these computations was performed for all the four measurement types, over varying window sizes.

5.1. Domain-specific validation

In order to validate our change detection scheme, we used a common methodology for determining variability based on ratios of measurements recorded at different time intervals. For each (i, j) tuple, $i \in [0, 65], j \in [0, 87]$, we computed the ratio of successive measurements in the normalized sequence. We then retained the ratio at the X^{th} percentile of these measurements, calling this the *intrinsic variability threshold* for a given X and tuple (i, j). Next, for each (i, j) pair, and for differing values of W, we collected the ratios between elements W measurements apart in the normalized sequence of measurements. Once again, we computed the X^{th} percentile value, calling this the *variability threshold* for a fixed W and X.

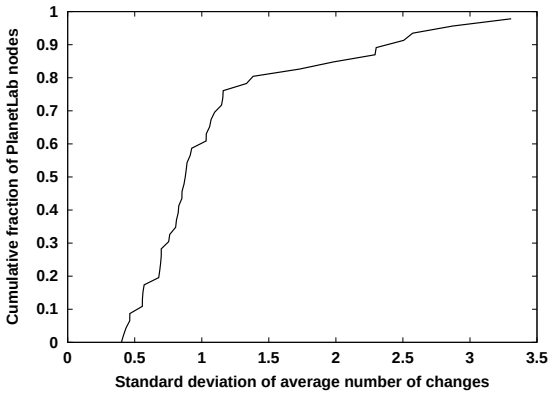


Figure 1. CDF of standard deviation across window sizes of average number of changes detected in TCP connection setup time for each PlanetLab node

A pair (i, j) was deemed to have *low-variability* for a given W if the variability threshold was within 5% of the intrinsic variability threshold for this pair. Thus, for each PlanetLab node i, we obtained a count of the number of low-variability pairs (i, j) for a fixed W and X. Allowing the window size W to vary from 15 minutes (equivalent to 2 units) to 4 days (equivalent to 768 units), and threshold X to vary between 70% and 95%, we obtain a variability score for each PlanetLab node i, computed as the standard deviation of the counts over all window sizes and thresholds.

The intuition behind this calculation is that for a relatively stable node, the ratios recorded over different time windows and different percentiles do not vary significantly, whereas if a node has erratic behavior, the number of low-variability pairs will fluctuate over window sizes and thresholds, yielding a high standard deviation.

6. Analysis and Results

We start with an overview of our results. Figure 1 charts the cumulative distribution of the number of PlanetLab nodes vs. the standard deviation of the number of changes, measured across all window sizes with respect to TCP connection setup time. The sharp "knee" in the plot indicates that a large majority of the nodes have a small standard deviation (below 1.0); only a few show large variability in their behavior.

A similar plot results if we plot the variation in the number of low-variability nodes (Figure 2). Once again, there is a sharp "knee" with only a few nodes displaying high variability.

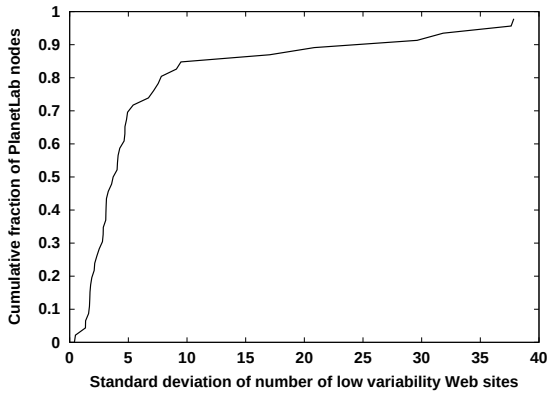


Figure 2. CDF of standard deviation across timescales and variability thresholds of number of Web sites which showed low variability in TCP connection setup time for each PlanetLab node

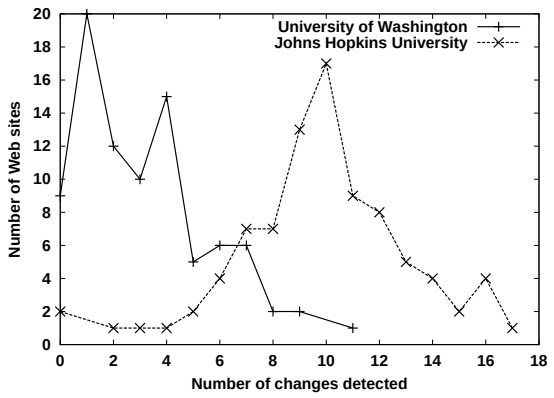


Figure 3. Histogram of number of changes detected for two PlanetLab nodes. The x axis measures the number of changes, and the Y axis records the number of web site streams to which these many changes were detected.

What these graphs indicate is that overall, a small fraction of the PlanetLab nodes exhibit high variability in their measurements, and the rest exhibit fairly stable behavior. However, does our (generic) change detection scheme detect the same nodes as being highly variable as the more domain specific ratio method – one that is fairly common in network measurements ? If we examine the set of high variability nodes in each case, (i.e the nodes "above the knee" in each graph) we find a high level of similarity. Specifically, we ranked the nodes in decreasing order of standard deviation for both the ratio method and the change detection method, for the different measurement units, and retained the nodes "above the knee" in each case. Comparing the rankings generated by the two approaches, we found that there was a high correlation between the two rankings; when we considered the top 22 ranked nodes for both TCP connection setup time and HTTP transfer time, there were 16-17 overlaps.

To illustrate the difference in the change histograms, we plot in Figure 3 the distribution of websites across changes for two nodes; a low-variability node (the University of Washington) and a high-variability node (Johns Hopkins University). These are plotted for a fixed window size of 25, and illustrate the distinct difference in the histograms of the two cases.

Although the rankings match quite well, there are still some discrepancies. In Figure 4 we illustrate the discrepancies with two examples; the first being a node that the change detection scheme deems highly variable, and the second being a node that the ratio method deems highly variable. In each figure, we used a representative (node, web site) pair.

In general, discrepancies occur because the ratio method assigns a single bit to a stream, declaring it highly variable or not. The change detection method, on the other hand assigns a numerical strength of variability to a stream. Thus, when aggregating, the schemes may differ slightly when a few streams skew the outcome. This phenomenon is show in Figure 4(b); only the displayed stream has high variability, increasing the change detection. Overall though, the number of highly varying streams remains small, and thus so does the estimate provided by the ratio method. In Figure 4(a), the change detection procedure detects a single change; the ratio method views this stream as highly variable because for different window sizes and thresholds, many high ratios will be recorded.

Finally, we discuss parameter choices. The window size appears to be the most crucial parameter, as it controls to an extent the resolution of changes that we discover. A window size too large, and small changes will be drowned out; too small, and spurious changes will be detected, or some changes might even be missed. The window size also controls the measurement rate. A window size that captures all changes also indicates the number of measurements needed; the measurements should be sufficient to cover all windows (once again, a larger "true" window size would imply that fewer measurements are needed to estimate the variability of a node). Our change detection scheme has an inherent false positive rate that is independent of the data, and thus generates fewer errors with increasing window size.

In Figure 5 we plot the average number of changes for a given node over different window sizes. We picked three nodes at random from the high variability set and three nodes at random from the low variability set. Two obser-

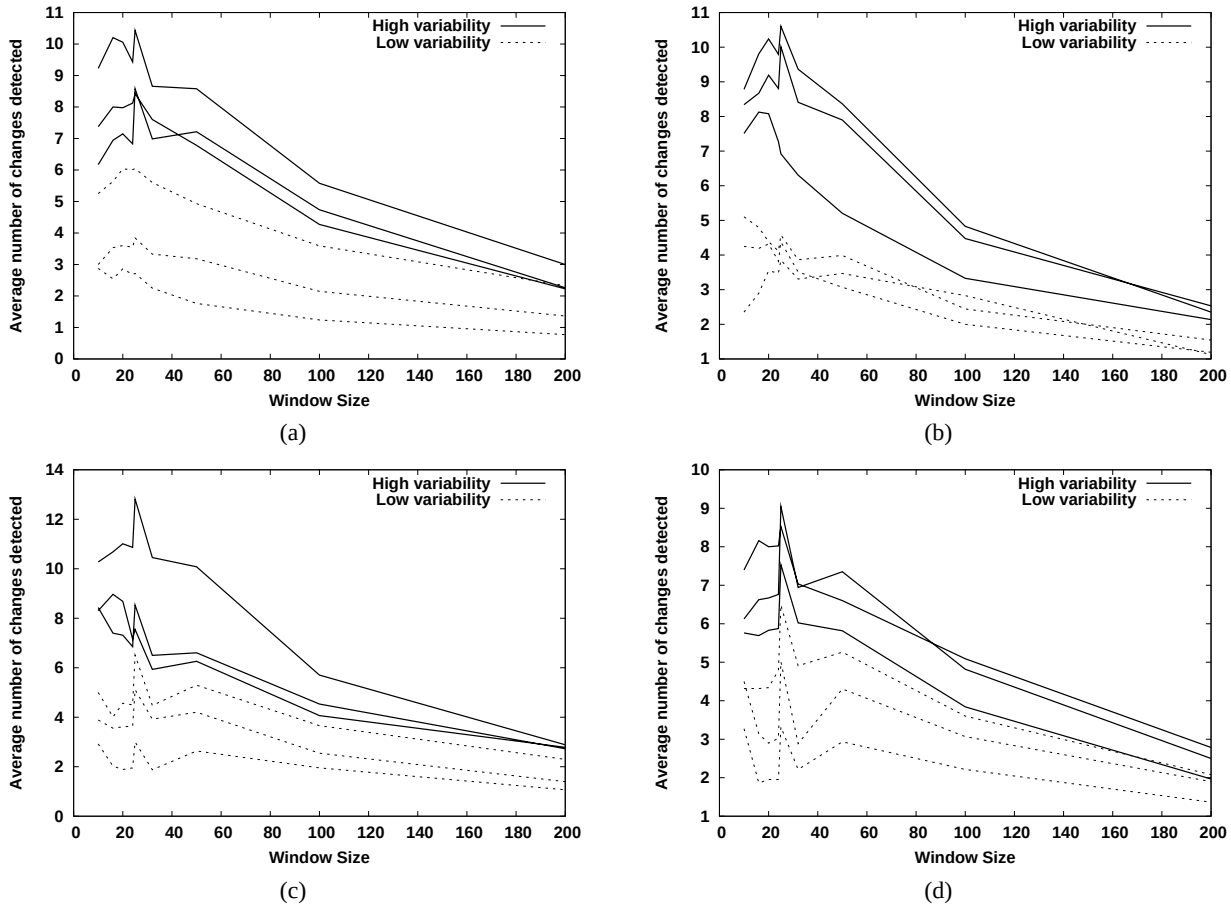


Figure 5. Distribution across window sizes of average number of changes detected on each PlanetLab node for (a) DNS lookup time (b) TCP connection setup time (c) HTTP transfer time and (d) total download time

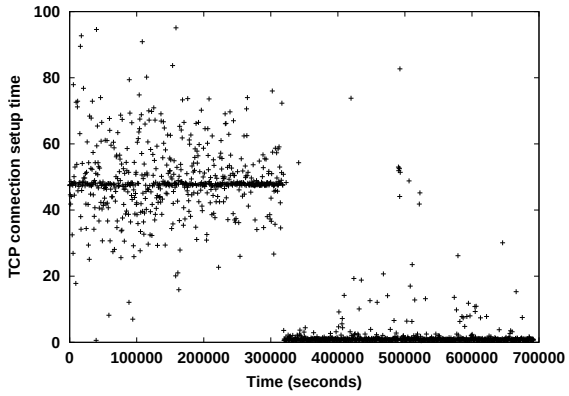
vations emerge from these charts: firstly, that the distinction between high and low variability is quite stable over window size and measurement. Secondly, using the low variability nodes to give a baseline measurement of the false positive rate, a plausible explanation for the discrepancy between the two groups is the increased number of true changes detected for the high variability nodes as the window size decreases.

7. Discussion

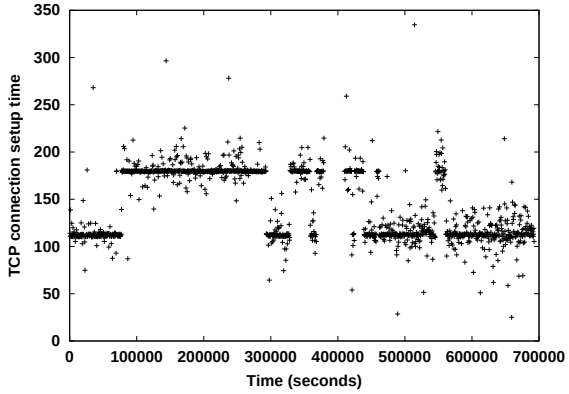
Change detection schemes are legion, and are carefully tailored to specific contexts and measurements. Our purpose in this work was to demonstrate how a general approach to change detection that is *oblivious to the nature of the data being evaluated* can be used effectively in a specific measurement setting, namely that of determining stationarity of nodes for the purpose of optimizing measurements.

How well do we do? In comparing our approach to a domain-sensitive method (the ratio method), we wished to find the "semantic gap"; the penalty we paid for being generic rather than specific. By that measure, our scheme performs very well; we are able to correctly separate high variability and low variability nodes, and even maintain a relative ordering comparable to that produced by a more specific approach. This behavior is consistent for different kinds of measurements; TCP setup times, DNS times and HTTP transfer times being the ones we examine here.

What this suggests is that our approach can be employed fruitfully in a variety of measurement settings, whether the underlying networks be sensor networks, peer-to-peer networks, or IP networks. This approach is not a panacea; domain specificity will in general provide more fine tuned analysis than such a general scheme can provide. However, the closeness with which our results match a more specific approach suggest that a hybrid approach that uses our general method as a "first-cut" to identify anomalies might be



(a)



(b)

Figure 4. Examples of TCP connection setup time distributions which lead to differences between the two ranking procedures. (a) Considered to be highly variable by the ratio method but the change detection procedure detects just one change (b) The ratio method ranks the PlanetLab node to have low variability due to the presence of only one such highly variable Web site, but a large number of changes even for just one Web site drives up the average number of changes, causing it to be labeled as being highly variable.

effective in evaluating stationarity in measurements across different domains.

It is worth pointing out that we applied our change detection mechanism on existing data collected for a different measurement project. The experimental methodology used for data collection was thus independent of our change detection approach, further eliminating biases in our results.

References

[1] Planetlab. `http://planet-lab.org`.

[2] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[3] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. Submitted, November 2004.

[4] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[5] 1999 Fortune 500 companies, Fortune volume 139 number 8, April 26 1999.

[6] D. Kifer, S. Ben-David, and J. Gehrke. Detecting changes in data streams. In *Proc. VLDB*, pages 180–191, 2004.

[7] B. Krishnamurthy, H. V. Madhyastha, and O. Spatscheck. Atmen: A triggered network measurement infrastructure. In *Proc. 14th International World Wide Web Conference*, 2005.

[8] Media Metrix. `http://mediametrix.com`.

[9] D. Mosberger and T. Jain. httperf: A tool for measuring web server performance. *ACM SIGMETRICS Performance Evaluation Review*, 26(3):31–37, Dec. 1998.

[10] The Netcraft Web Server Survey. `http://netcraft.co.uk/survey`.