Comparing Network Bandwidth Time-Series*

Matthew S. Allen
Department of Computer Science
University of California, Santa Barbara
msa@cs.ucsb.edu

John Brevik
Department of Mathematics & Statistics
California State University, Long Beach
jbrevik@csulb.edu

Rich Wolski
Department of Computer Science
University of California, Santa Barbara
rich@cs.ucsb.edu

ABSTRACT

Today, internet researchers, engineers, and application writers have at their disposal a number of methods for measuring end-to-end internet performance. Additionally, many wide-area applications make heavy use of measurement techniques to optimize their performance. Despite this, there is no widely accepted method for determining if two tools or techniques produce equivalent results, or if feedback from a tool is relevant to the application that employs it. In this paper, we apply current technologies in time series databases and network performance modeling to the problem of comparing network bandwidth time series. Using these techniques, we present a methodology to evaluate the level of similarity between two time series.

1. INTRODUCTION

Developers and engineers on today's internet have a number of techniques and tools at their disposal for evaluating end-to-end internet performance. Tools like the Network Weather Service [17] and Pinger [5], and their predecessors Iperf [13] and NetPerf [10], measure internet properties using direct probes. Other tools like PathLoad [9] and PathChirp [14] infer path properties using a storm of small packets much like Van Jacobson's pathchar utility did in the early days of network measurement. Additionally, a wide array of measurement heuristics see deployment as part of existing wide area applications.

Wide-area distributed applications employ these tools and techniques to aid in decision making. Grid applications make heavy use of monitoring techniques to schedule both distributed jobs and data transfers [16, 7]. Distributed storage systems use network measurements to chose which copy of a duplicated data item to download [2, 3]. Overlay peer-to-peer networks use similar techniques to optimize their routing tables and select high-bandwidth peers [19, 20].

*This work was supported, in part, by NSF grants numbered 0305390 and 0123911.

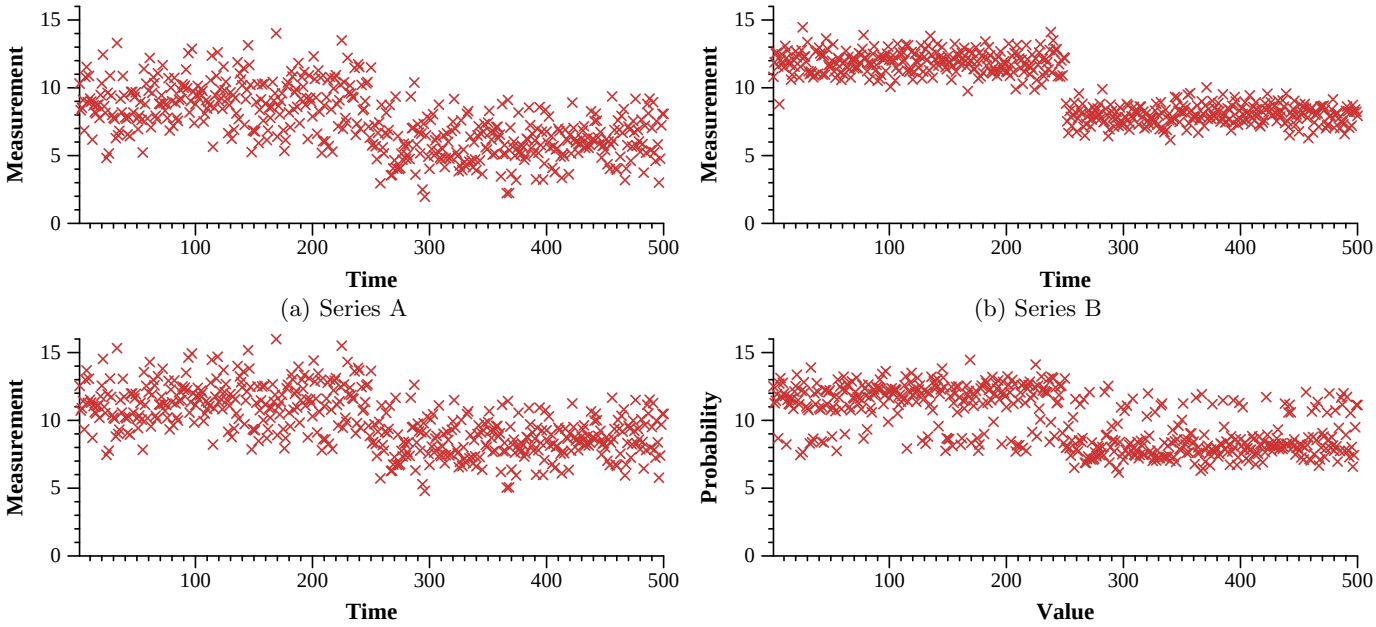
GridNets 2007 October 17-19, 2007, Lyon, France.
Copyright 2007 ICST ISBN 978-963-9799-07-3.
DOI 10.4108/gridnets.2007.2241

Given this, there are compelling reasons to develop techniques to automatically compare network bandwidth series. On one hand, measurement tool developers could automatically determine if an experimental measurement algorithm produces results similar to an established method. Application writers, on the other hand, could test how well a stream of probes match observed transfers. While it is possible to perform such comparisons visually, this is prohibitively difficult when comparing large amounts of data. And with the tremendous diversity of internet links, performing extensive comparisons can be very important. While there have been some efforts in recent years to develop such a methodology [12], there is still much work to be done.

Unfortunately, calculating the similarity between two end-to-end performance time series is a challenging endeavor. There are a number of subtle differences between two series that should be ignored during comparison. Also, the time series themselves are complicated, non-stationary processes with significant changes in mean and variance. The following specific issues must be considered when comparing bandwidth time series:

1. Network paths experience abrupt changes in their performance characteristics. Methods that do not consider these abrupt changes may not draw the correct information from statistics like mean, variance, and the empirical CDF.
2. Series may show similar performance changes, but differ in scale. Comparison methodologies should be resistant to differences that can be removed using linear transformations on the data.
3. Network measurements always exhibit some sampling error, but the distribution of these errors is unknown. Therefore, any method that assumes an underlying error distribution may produce erroneous results.

In this paper, we present a novel method for determining the level of similarity between network bandwidth time series. This method draws on a number of different techniques to produce its comparison metric. We adapt techniques used in the time series database community to understand and remove difference in shift and scale between series. We utilize change-point detection methods employed to automatically locate abrupt changes in performance characteristics. Finally, we make use of empirical properties of time series



(a) Series A (b) Series B

(c) Linear Transform of A (d) ECDF Translation of A

Figure 1: Synthetic series demonstrating potential difficulties in relating network bandwidth time series with abrupt performance changes.

data, as well as non-parametric modeling techniques, to develop a methodology that relies on few assumptions about the underlying data. We evaluate the performance of this methodology for a few case studies, and we discuss some issues that impact its performance.

2. COMPARING SERIES WITH ABRUPT PERFORMANCE CHANGES

In the world of network monitoring, few tools or techniques produce exactly the same results. However, there are frequently only minor differences between time series, and it is preferable to ignore these differences during comparison. In particular, a successful comparison must be insensitive to differences in scale. There are a number of techniques for accomplishing this, but here we will discuss two of the most significant: linear transformations [1, 4, 6] and empirical CDF (ECDF) translations [12, 15]. Both of these techniques, which will be explained below, rely on computing statistics of the time series over a window of history to create a transformation function.

While these algorithms perform well in many situations, they are potentially complicated by one important link property. End-to-end link performance is known to change abruptly, which has been discussed thoroughly in [18], but is also present in the data sets shown in this paper. Unfortunately for the two methods mentioned above, these abrupt changes can invalidate the statistics that are used to remove differences in scale when performing a comparison.

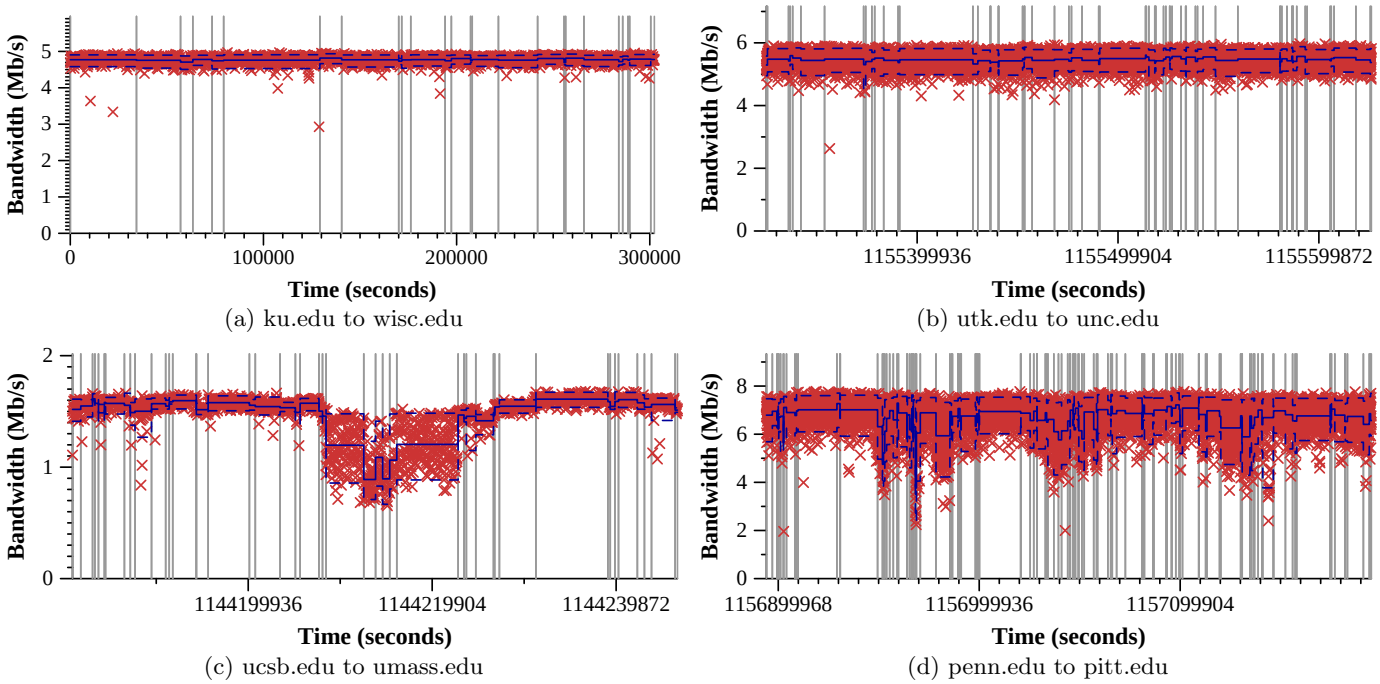
Figures 1(a) and 1(b) are artificially generated time series used to illustrate this problem. These two series represent the types of differences we would hope to ignore when comparing two bandwidth series. The values in series A are generated from a normal distribution with (mean) $\mu = 9$ and (variance) $\sigma^2 = 3$ for the first half of the trace. At the half-way mark, the generating distribution changes to $\mu = 6$ and $\sigma^2 = 2$. Series B, on the other hand, starts with $\mu = 12$ and $\sigma^2 = 0.75$, and transitions to $\mu = 8$ and $\sigma^2 = 0.5$. In both series, the mean and variance both drop by 33% at the half-way mark. Thus, while they contain different values, they are clearly very similar in form.

Perhaps the simplest method for eliminating differences in shift and scale is to perform a linear transformation on the data. This method is commonplace in time series database research because of its low computation overhead. To accomplish this, we construct a new series using the sample mean m and the sample variance v computed over the two series:

$$x_c = \frac{x_a - m_a}{\sqrt{v_a}} * \sqrt{v_b} + m_b$$

Due to the presence of the change-point, however, the sample mean and variance are no longer particularly meaningful as statistics. As a result, the transformed series shown in figure 1(c) diverges significantly from the target series. Here, we have $\mu \approx 11.3$ for the first half and $\mu \approx 8.5$ for the second. As mentioned in [6], this problem can be mitigated by smoothing the series with a moving average prior to scaling. This transformation will obfuscate the sampling error of the measurements in the series, but it will not eliminate this problem. Also, moving averages will respond slowly to the abrupt changes we see here, which is not desirable.

Another more robust method, developed in this research group, uses the ECDFs of the two series to translate from one to the other. In this technique, the ECDF is calculated



(a) ku.edu to wisc.edu

(b) utk.edu to unc.edu

(c) ucsb.edu to umass.edu

(d) penn.edu to pitt.edu

Figure 2: Examples of bandwidth interval models

for each series. Values can then be translated using the following function:

$$x_d = ECDF_b^{-1}(ECDF_a(x_a))$$

Again, the presence of the change-point impacts the appropriateness of the ECDF, since it is actually capturing two separate distributions. Figure 1(d) shows the results of this translation on the example series. In this case, the problem we see with the series is the modal pattern throughout the trace. This is caused because in series A, the two distributions overlap, while in series B they do not. Values that fall in the overlapping range of series B have a chance of being paired with an inappropriate value in series A.

It should be noted that both these methods perform well in a variety of applications. In particular, the ECDF translation method works well in almost any situation except the one described. However, both techniques can be adversely affected by the types of change-points we see in network bandwidth series. However, either method could be improved by proactively detecting change-points and discarding data prior to the the change.

3. COMPARISON METHODOLOGY

In this section, we describe a method for comparing network bandwidth time series. This method has a number of properties that are important when considering these types of series. First, it is fully non-parametric, meaning it makes no assumptions about the distributions of the measurement values. Additionally, it is specifically designed to work with data that shows abrupt changes in location and shape. Finally, it is designed to resist differences in scale between the series, and only detects differences that cannot be eliminated through linear transformations of the data.

There are two phases to this technique. In the first phase, we generate a *change-point model* for each series. These models represent a time series as a collection of regions demonstrating consistent performance characteristics. In the second phase, we apply each model to all of the other series that we wish to compare it with. We perform a transformation on each model to remove difference that could be represented by a linear transformation. This test produces a metric describing the quality of the match between the two series.

3.1 Change Point Modeling

Our modeling methodology makes use of a non-parametric recursive change-point detection method. Our method falls into the same general class as the method presented in [18]. It works as follows:

1. Partition the series into two sections.
2. Perform a hypothesis test and calculate probability that the difference between the two sections could be caused purely by random chance (called the p-value)
3. Repeat 1 and 2 for all possible partitions of the series.
4. Choose the partition that results in the minimum p-value. If this value falls below a significance level chosen as a parameter to the method, mark the partition as a change-point.
5. If a change-point was found, recursively apply steps 1 through 5 on the two resulting sections.
6. When the algorithm completes, test each change-point with respect to the new change-points that may have

been added near it. To do this, consider every pair of adjacent sections. Perform steps 1 through 4 on the two combined sections to see if the change-point that divides them should be moved or removed.

In [18], the authors apply the Fligner-Policello Rank-Sum Test in the method just described. This non-parametric hypothesis test is designed to detect differences in the median of two samples without being affected if the variance of the samples differ. However, in our application we are interested in changes in the distribution, so this test is not appropriate. Instead, we utilize the Kolmogorov-Smirnov Test, another non-parametric test that detects any difference in the distribution of two samples. This includes, but is not limited to, differences in central tendency and variance. The Kolmogorov-Smirnov Test has been applied in this type of change-point detector in a number of different disciplines [8, 11].

We will note that this method is not a statistically rigorous method–it is a heuristic. It is a misuse of hypothesis testing to claim that there **is** a difference between two samples. It can only be used to to show that the difference that exists is unlikely to be due to random chance. Nevertheless, this method performs reasonably well for our purposes.

The final component of our modeling process is to determine the range of values we expect to see in each partition. First, we calculate the *expected bandwidth* to be the mean of all the values in a given partition. Next, we define the *bandwidth range* to be the interval that captures 95% of the values for that partition, which we determine empirically using the 0.025 and 0.975 quantiles of the partition. Once this is calculated for each partition, we have a description of the bandwidth measurement values we would expect to see at any given time in the trace. Figure 2 is an example of our methodology applied to 4 different series. In these graphs, the change-points are shown as vertical grey lines in the background, and the bandwidth interval is superimposed on top of the series.

3.2 Model Scaling

Our goal with this methodology is to compare series in a manner that is resistant to differences in shift and scale. To accomplish this, it is necessary that we be able to transform the model for one series such that it is on the same scale as another. In this section, we describe how we accomplish this.

Obviously, the model we described previously can be represented by three piecewise linear functions describing the expected bandwidth, upper, and lower quantiles at time t. We call these *model functions* $m(t)$, $u(t)$, and $l(t)$, respectively. We define $p_{i,start}$ and $p_{i,end}$ to be the start and end time of the i^{th} partition of the model. We also define s_i to be the appropriate statistic (mean, 97.5% quantile, or 2.5% quantile) of that partition, and n to be the total number of partitions in the model. Then, each of the model functions m, u, and l can be expressed with a function of the form:

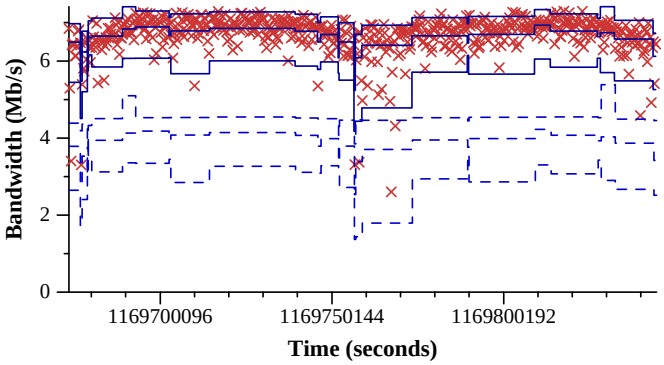


Figure 3: Example of a transformed model

$$f(t) = \begin{cases} s_1 & \text{for } t \geq p_{1,start} \text{ and } t < p_{1,end} \\ s_2 & \text{for } t \geq p_{2,start} \text{ and } t < p_{2,end} \\ \dots & \dots \\ s_n & \text{for } t \geq p_{n,start} \text{ and } t \leq p_{n,end} \end{cases}$$

Our transformation method relies on a simple technique to calculate a linear transformation that minimizes the distance between two such model functions. To place a model function for A on the same scale as a model function for B, we construct a *distance function* that expresses the distance between these two model functions. The distance function includes two scaling coefficients, r_{AB} and s_{AB}, which describe a linear transformation of the model function. This distance function is:

$$D = \int_{start}^{end} (f_B(t) - (r_{AB} f_A(t) + s_{AB}))^2 \, dt$$

By solving for values of s_{AB} and r_{AB} that minimize D, we can perform a transformation of f_A that places it on the same scale as f_B. This formula provides us the fundamental tool to scale the entire model for series A to fit series B. Recall that we have three functions: m, u, and l. Before we perform the scaling operation, we represent the upper and lower bounds in terms of their relationship to the mean. We define upper and lower bound offset functions, which are:

$$u'(t) = u(t) - m(t)$$
$$l'(t) = m(t) - l(t)$$

We apply our transformation formula to m_A, u'_A, and l'_A, computing transformation coefficients r_{AB} and s_{AB} for each separately. This results in a collection of functions of the form:

$$\hat{f}_A(t) = r_{AB,f} f_A(t) + s_{AB,f}$$

Finally, we recombine $\hat{m}_A$ with $\hat{u}'_A$ and $\hat{l}'_A$ to get $\hat{u}_A$ and $\hat{l}_A$. These final three functions express the model for A transformed so that it is on the same scale as B. Figure 3 shows

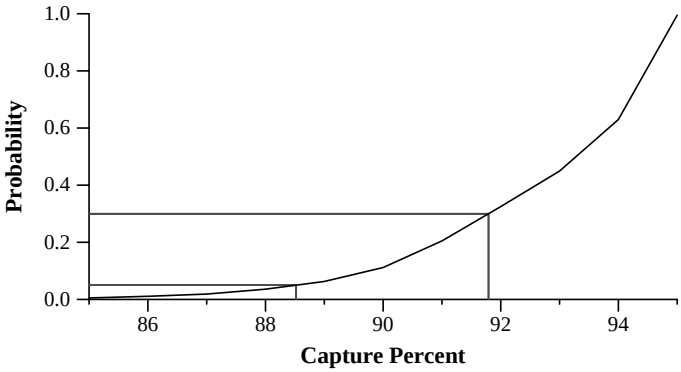


Figure 4: CDF of capture percentages from IID random samples of length 100

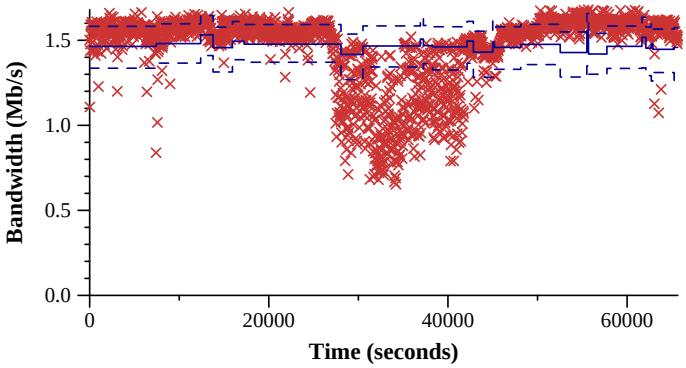


Figure 5: Model for ku-wisc fitting the ucsb-umass series, an example of a poor model fit

	ku-wisc	utk-unc	ucsb-umass	penn-pitt
ku-wisc	94.9%	94.2%	55.4%	88.6%
utk-unc	93.6%	94.9%	51.4%	88.9%
ucsb-umass	93.7%	94.3%	94.6%	88.7%
penn-pitt	93.9%	94.3%	51.1%	94.8%

Table 1: Cross-capture table for example series in Figure 2

as example of this method in action, where the dashed lines show the model before it was scaled, and the solid lines show the same model after scaling.

3.3 Model Cross-Capture

Our comparison technique makes use of a metric we call *capture percentage*, which is the percentage of points in a series that fall within the bandwidth range of a model. The method presented earlier in this section produces a model that will always capture very close to 95% of the values in the series it was constructed from. So, if we are comparing two network bandwidth time series A and B, we can say they are equivalent if the model for A captures B roughly 95% of the time and the model for B captures A roughly 95% of the time.

This method is a heuristic, and there are no statistically rigorous bounds on how close capture percentages should be to 95%. Thus, we evaluate the range of reasonable values empirically. Figure 4 shows an ECDF of capture percentages produced when applying our method to two series of 100 values drawn from a normal distribution with no change-points. This ECDF was computed by generating 10,000 such series pairs and computing the capture percentages between them. We use this graph as a benchmark for determining goodness of fit. Since these values show capture percentage between series drawn from the same distribution, we expect matching series to fall into the range of values shown here. Here, 70% of the values fall above 91.8%, which we classify as a "good match". Capture percentages greater than 88.5%, which account for all but 5% of the data, are also likely to indicate matching series. Thus, we classify these as a "likely match". Finally, we classify values less than 88.5%, which should rarely occur in matching series, as an "unlikely match".

Our primary method for comparing collections of time series is the cross-capture table. Table 1 is an example using the four series shown in figure 2. The rows represent the model for a series, and the columns represent the series being captured. So, for example, the value in the bottom-left corner shows that the penn-pitt model captured 93.9% of the values of the ku-wisc series. To determine the equivalence of two series, locate the capture percentage in both directions, and use the minimum value. In this case, we see that in the other direction the capture percentage is 88.6%, or a moderate match.

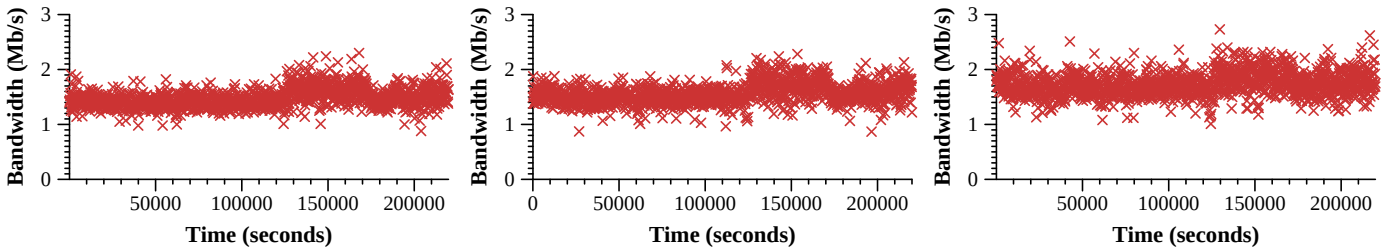
It is important, when doing comparisons, that we base our evaluation on how well each model fits the other series. Computing this capture percentage from only one model to another series is not sufficient. As an example, consider the two series *ucsb-umass* in figure 2(c) and *ku-wisc* in figure 2(a). These series are very different, with the former containing a number of distinctly different regions, and the later being nearly flat. However, we see from the table that the ucsb-umass model captures the ku-wisc series 93.7% of the time–a "good match". This is a result of the scaling operation, which can easily turn any model into a flat line by setting $r = 0$. The true indication of fit comes from the capture percentage in the other direction, shown in Figure 5. Here, the capture percentage of 55.3% reveals the lack of a match.

4. METHODOLOGY PERFORMANCE

In this section, we will discuss the performance of our test under a couple of applications where we expect similar performance. In the methodology section we showed how the test performs for a handful of series that were clearly different. In this section, we present studies of how this test performs with series that are more similar. First, we will examine how our methodology performs with direct measurement probes generated by different tools or different probe sizes. We will then investigate how well our test performs with different sampling frequencies.

4.1 Comparing Direct Measurement Tools

This methodology was originally conceived to compare the results of different bandwidth measurement tools. There exist a large number of tools that measure network bandwidth by probing the network. Their implementations are extremely similar. Each tool runs two applications–a source



(a) NetPerf (b) NWS (c) IPerf

Figure 6: Measurements recorded from ucsb.edu to wisc.edu by three different tools

	NetPerf	NWS	IPerf
NetPerf	95.2%	93.3%	93.8%
NWS	94.3%	94.8%	94.1%
IPerf	92.4%	91.8%	95.1%

Table 2: Cross-capture table for NetPerf, NWS, and Iperf tools shown in Figure 6

	64 KB	512 KB	8196 KB
64 KB	95.1%	82.5%	88.1%
512 KB	89.8%	94.2%	93.3%
8196 KB	93.9%	88.6%	95.1%

Table 3: Cross-capture table for NWS with 64 KB, 512 KB, and 8196 KB probe sizes shown in Figure 8

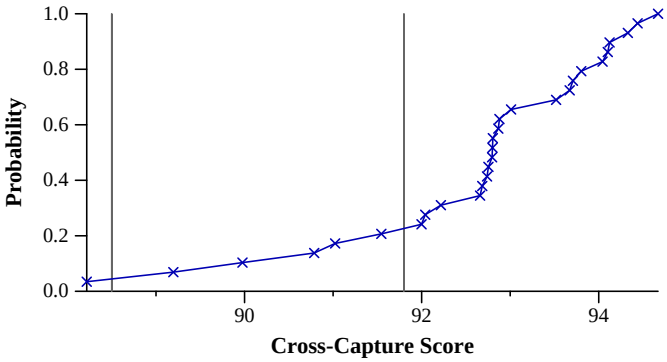


Figure 7: ECDF of model fit metrics for experiments comparing Iperf, Netperf, and NWS

on one machine, and a destination on the other. To perform a measurement, the source opens a TCP connection to the destination, starts a timer, and initiates a transfer. The destination responds when the transfer is complete, and the source calculates the observed bandwidth by dividing the size of the transfer by the transfer time. Tools may differ with respect to the socket conditioning they perform or the loop structure of their test, but share this same basic implementations.

Intuitively, we would expect that there is little to no difference between these two tools. In practice, however, measurement series do exhibit clear differences. For instance, figure 6 shows the three different measurement streams from ucsb.edu to wisc.edu. Here, we see that although these results look very similar to the eye, there are also clear difference between them. For instance, the IPerf reports a consistently higher bandwidth, and also clearly has a higher sampling error. Netperf, on the other hand, exhibits slightly lower measurements and less sampling error.

Despite these differences, the measurement series visually appear similar, and we would expect our comparison method to ignore these differences. Table 2 demonstrates that this does, in fact happen. Each value in the cross capture table falls within the "good match" range. Thus, our method detects that the differences between these tools can be accounted for using change-point aware linear transformations.

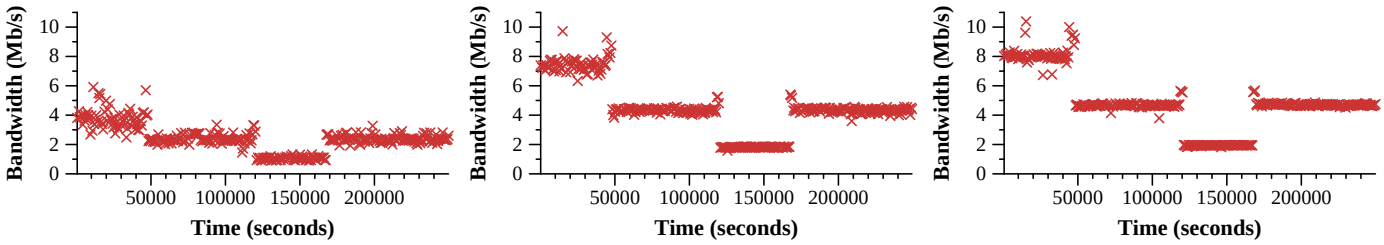
We repeated this experiment across 10 different data sets similar to the ucsb.edu to wisc.edu experiments above. The results of these comparisons are shown in figure 7. This graph shows us that over 80% of the series compared in this test showed a cross capture score over 91.8%–a "good match". Only one of these comparisons resulted in an "unlikely match". This corroborates our intuition that tools like NWS, Iperf, and NetPerf produce results with no meaningful difference for evaluating link performance. Over the links we tested, they could be used interchangeably with no adverse effects.

4.2 Comparing Different Probe Sizes

Another interesting application for our measurement technique is to compare the outputs of measurement probes of different sizes. Direct measurement tools must always strike a balance between accuracy and intrusiveness. Large transfers generate more accurate and reliable results, but small transfers consume less network resources. Our technique allows us to see if there is a series of linear transformations that matches a small, non-intrusive measurement series with an accurate but intrusive one.

Figure 8 shows three different series generated by the NWS using three different probe sizes. Here, differences are far more pronounced than in the previous case study. In particular, small probe sizes report much lower bandwidth than large probes. Also, the variance in each partition is larger in the small probes. Nevertheless, it is obvious to the eye that these tools produce very similar results, and track the same changes in network conditions. However, the extent of their similarity is not necessarily clear cut.

The cross-capture table shown in table 3 describes the level



(a) 64 KB (b) 512 KB (c) 8196 KB

Figure 8: Measurements recorded from umich.edu to wisc.edu using three different probe sizes

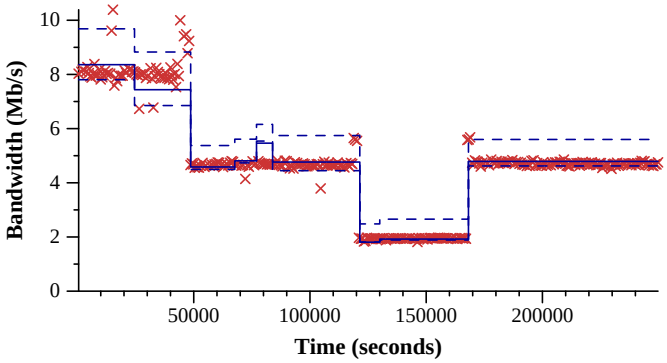


Figure 9: ECDF of model fit metrics for experiments comparing Iperf, Netperf, and NWS

of similarity between these two series. Here, we see that only the 512 KB and 8196 KB measurement series match to a level that is acceptable to our method. The 64 KB probe fails to adequately match either of the larger measurements. Figure 9 elucidates the cause of this result. Here, we see that the 64 KB probe shows a pronounced shift near the 20,000 second mark, as well as a noticeable jump near the 60,000 second mark. These discrepancies indicate that the 64 KB probes respond to network conditions that may not impact larger transfers. By no means do we claim this is a general result. We only show that the application of our tool can reveal differences between series that may not be obvious.

5. SAMPLING FREQUENCIES

In this final section, we analyze how the performance of our method changes with different sampling frequencies. Tests of this nature are typically affected by the frequency with which measurements are made. More data will produce more accurate and consistent results, while less data will cause more variance among the results. This test is no exception.

To understand our method's performance under different sampling frequencies, we modified the experiments reported in section 4.1. These experiments consisted of 10,000 measurements taken over an 80 hour period. The measurement frequency was one measurement every 30 seconds for each of the three tools–Iperf, NetPerf, and the NWS. We decimated these data sets in three different ways to better understand how our method performs.

- **Cropped**–Remove points from the end of both series. This reduces the length of the series, but the periodicity remains unaffected.
- **Evenly Decimated**–Remove points evenly distributed throughout both series. This modification increases the periodicity of the measurements.
- **Unevenly Decimated**–Decimate one series as above, but leave the other series the same. Both series cover the same time period, but one has a longer measurement periodicity and less points.

Figure 10 reports the impact of the three modifications we describe applied with increasing levels of severity. The bars and error bars show the median, 5% quantile, and 95% quantile of the output of our method. These values are derived from ECDFs like the one shown in figure 7. The two demarcation lines in the background mark the "good match" and "likely match" capture thresholds. Here, we see that over half of the series classify as a "good match" up until the level of decimation reaches 95%. Even in series that are reduced by 99%, which equates to either a one hour trace or a measurement periodicity of one hour, we still see that the majority of the points classify as a match. Since it is more difficult produce a meaningful comparison with such short or infrequent measurement series, we consider this an acceptable level of degradation.

It is, however, worth noting that series with less values are more likely to produce an "unlikely match" result, even when there is little impact on the median result. This is evident from the 5% quantile, which is the statistic that is most heavily influenced by decimation. Also, this effect is most pronounced when the series are decimated unevenly. In this graph, the 5% quantile after a 99% decimation was 49.7%. The reason for this performance hit is similar to the issues that arise when performing a linear transformation on time series data as discussed in section 2. In a more densely populated series, more change-points will be detected. This results in more variance among the statistics of each partition. When it is scaled to match a more stable model with less change-points, the effect will be similar to performing a linear transformation on a series with a high variance, and the higher variance series will be flattened.

6. CONCLUSION AND FUTURE WORK

In this paper we present a novel technique for comparing network bandwidth time series. We feel this technique is robust, flexible, and applicable to a number of problems in

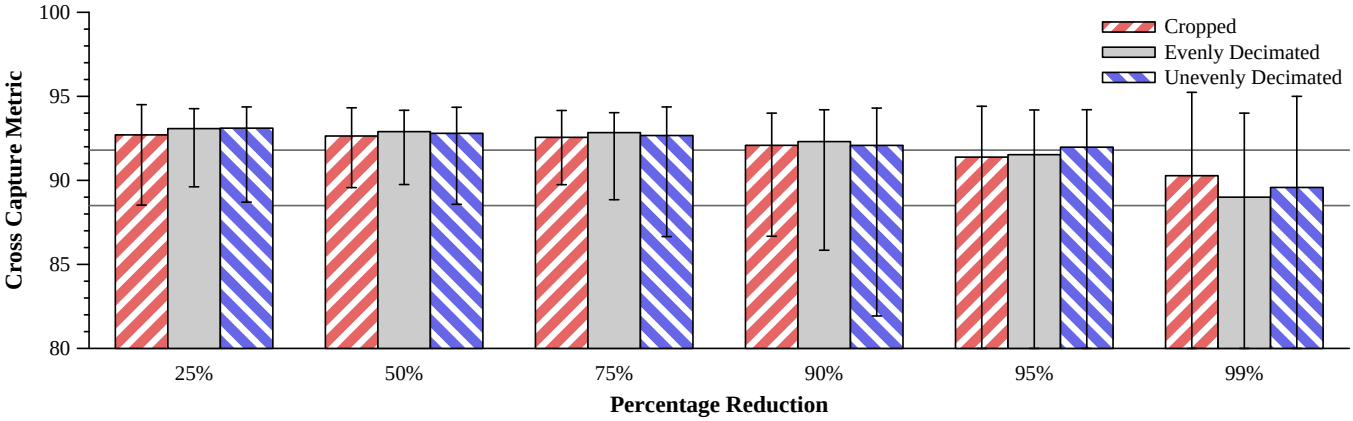


Figure 10: Capture values produced after decimation or modification of measurement series.

the network measurement community. However, while we feel that this document adequately explains this method, and explores the parameters of its operation, this tool is still in its infancy. Only through applying this technique to a number of time series comparison problems will we be able to fully understand its capabilities and address its weaknesses.

There are also a number of areas where this tool could be further modified or enhanced. For example, there are a number of sophisticated hypothesis test and change-point detectors that might improve the modeling component of the method. Also, our use of empirical quantiles to build our model has a low breakdown point, and is sensitive to outliers. Methods for estimating the population quantile or building confidence intervals may enhance the resilience of this tool. Finally, applying on-line change-point detection techniques to bandwidth prediction may prove to be a rewarding research direction. And this technique may help provide groundwork for understanding how to design such a prediction algorithm.

7. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *Foundations of Data Organization and Algorithms*, pages 69–84, 1993.

[2] M. Allen and R. Wolski. The Livny and Plank-Beck problems: Studies in data movement on the computational grid. In *Supercomputing 2003*, November 2003.

[3] R. L. Cards and M. E. Crovella. On the network impact of dynamic server selection. *Computer Networks*, 31(23-24):2529–2558, 1999.

[4] K. K. W. Chu and M. H. Wong. Fast time-series searching with scaling and shifting. In *ACM Symposium on Priciples of Database Systems*, pages 237–248, 1999.

[5] L. Cottrell, W. Matthews, and C. Logg. Tutorial on internet monitoring & pinger at slac.

[6] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *Principles of Data Mining and Knowledge Discovery*, pages 88–100, 1997.

[7] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1998.

[8] A. Inoue. Testing for distributional change in time series. *Econometric Theory*, 17:156–187, November 2001.

[9] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. In *IEEE/INFOCOMM*, August 2002.

[10] R. Jones. The netperf homepage.

[11] S. D. Mohanty and A. JimÃl'nez. Progress in non-parametric change point detection of gravitational wave bursts: the multikscd algorithm. *Classical and Quantum Gravity*, 22(18):1233–1241, September 2005.

[12] M. Murray, S. Smallen, O. Khalili, and M. Swany. Comparison of end-to-end bandwidth measurement tools on the 10gige teragrid backbone. Technical Report TR-05-02, University of Texas at Austin, 2005.

[13] NLANR/DAST. The iperf homepage.

[14] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In *Passive and Active Measurement Workshop*, 2003.

[15] M. Swany and R. Wolski. Multivariate resource performance forecasting in the network weather service. In *IEEE/ACM Conference on High-Performance Computing and Networking (SC2002)*, November 2002.

[16] The Globus Alliance. GridFTP.

[17] R. Wolski. Dynamically forecasting network performance using the network weather service. *Cluster Computing*, 1:119–132, 1998.

[18] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of internet path properties. In *ACM SIGCOMM Internet Measurement Workshop*, 2001.

[19] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, January 2004.

[20] J. Zurawski, M. Swany, M. Beck, and Y. Ding. Logistical multicast for data distribution. In *Workshop on Grids and Advanced Networks*, 2005.