# Fast Pattern-Based Throughput Prediction for TCP Bulk Transfers

Tsung-i (Mark) Huang and Jaspal Subhlok
Department of Computer Science
University of Houston
Houston, TX 77204-3010
Email: {tihuang, jaspal}@cs.uh.edu

## Abstract

*The ability to quickly predict the throughput of a TCP transfer between a client and a server, or between peers, has wide application in scientific computing and commercial computing. This paper presents a new approach to fast prediction of overall throughput of a large TCP file transfer. The method constructs the time series of windows of segments arriving at the receiver, and predicts future throughput by exploiting knowledge of how TCP manages transfer window size. When the file transfer time series resembles a known TCP pattern, this information is utilized for prediction, otherwise simple heuristics are used. We have compared TCP pattern based prediction against traditional methods like a simple moving average, exponential weighted moving average, and aggregate measured throughput on a large suite of real life TCP traces. Our results show that TCP pattern based prediction generally performs as well or better than the best of other methods in any given scenario.*

## 1. Introduction

The performance of any network based application depends on the throughput that can be obtained on network paths. Estimation of throughput is an important component of applications and resource management in network computing. As examples, replica selection in a grid environment [19], selection of caches in web content delivery services [14], mirror site selection for downloading, and selection of P2P peers, are all primarily driven by the expected throughput on network paths. In a grid environment, network performance is an important part of grid services [9].

Over half of today's network traffic uses Transmission Control Protocol (TCP) [15]. Bulk transfer utilities like GridFTP [5] and storage access protocols like GASS [4] are based on TCP. The goal of this paper is to predict the throughput of a TCP stream by inspecting the pattern of arrival of segments at the receiver.

Most bandwidth measuring tools focus on available bandwidth or bottleneck bandwidth of a network path or a particular link. However, we focus on the related but different problem of expected bandwidth of one TCP stream. Iperf [2] is designed to measure maximum TCP bandwidth. To get an accurate measurement, sufficient amount of data (set by size or duration) needs to be sent over the measured link which would interfere with existing network traffic. Other prior work by Sang and Li [13] has used ARMA and MMPP statistical models to predict future bandwidth, under the assumption that traffic is stationary and can be modeled statistically.

The most popular tool for estimating available bandwidth for grid computing is Network Weather Services (NWS) [20]. NWS measures available bandwidth by transferring a block of data and measuring the time taken for the transfer. However, the measurement may not be representative of the long term bandwidth. One reason is that a TCP connection may be in slow start for a large part of the sample transfer. Vazhkudai et al. [18] show that NWS predicted bandwidth can be as much as 5 to 34 times lower than measured bandwidth on high speed networks, and propose some improvements. Swany [16] and Vazhkudai [17] use statistical methods to find a correlation between predicted and measured bandwidths. Primet et al. [12] compare the throughput results from NWS and measurements from Iperf, and propose new prediction models by limiting the influence of slow start to improve the accuracy of throughput.

The central contribution of this paper is to use knowledge of TCP patterns to make faster and more accurate bandwidth prediction of a TCP flow. A TCP flow normally starts with slow start, followed by a steady-state governed by TCP congestion control and flow control. In principle, the behavior of a TCP flow is predictable. The throughput of a TCP flow can be formulated in terms of Round-Trip Time (RTT),

Maximum Segment Size (MSS), and loss rate [10]. However, RTT is variable and hard to estimate [7] and a reliable loss rate cannot be obtained until a large number of segments have been collected. Also, a flow often does not reach a steady state when other network traffic is changing dynamically. Further, when a TCP sender is handling multiple streams, such as a Web server, the sender's load and policies are an important component of the TCP data pattern. Because of these reasons and the complexity of analysis, theoretical results that describe TCP behavior are not sufficient for quick throughput prediction in practice.

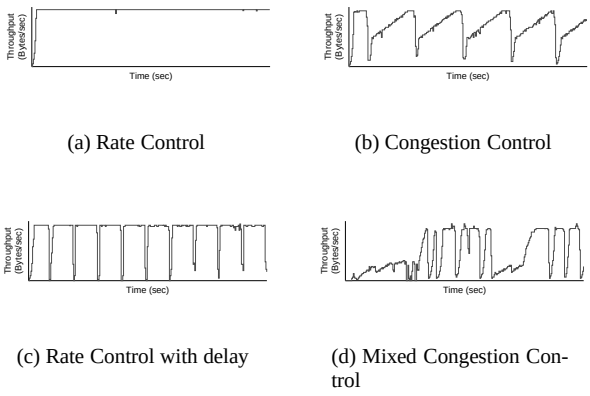
In this work we combine a knowledge of TCP patterns with heuristics to develop the framework for an intelligent real-time throughput predictor. The stream of incoming TCP segments is converted to a time series of throughput values. This time series is analyzed to generate a prediction for future throughput. This throughput prediction scheme has been validated with experiments on a large number of real-life TCP traces. We establish that our TCP pattern-based throughput prediction is superior to basic prediction methods like simple moving average, exponential weighted moving average, and employing aggregate throughput as a predictor for future throughput.

The rest of this paper is structured as follows. Section 2 describes the methodology of TCP pattern-based throughput prediction. Section 3 outlines the experiment setup and discusses the results. We present some conclusions in Section 4.

## 2. Methodology

### 2.1. TCP Flow Patterns

Data transfer patterns generated by classical TCP flow control and TCP congestion control in steady state are illustrated by example traces in Figure 1(a) and 1(b), respectively. Clearly if those patterns could be identified and the TCP connection followed that pattern for the duration of the transfer, future bandwidth could be predicted accurately. One thing to note is that the pattern in 1(a) may be due to TCP flow control, but it could also be a result of traffic shaping at the server. Hence we have used the more general term "rate control" to describe the pattern. Often, a uniform traffic rate is observed in TCP flows, but with frequent delays, as shown in Figure 1(c). Finally, many flows do not show a clear pattern, such as the example flow in Figure 1(d). Our goal in this research is to identify the flow pattern and use that knowledge to make a future bandwidth prediction. However, if no clear predefined pattern is established, we combine the knowledge with heuristics to make a throughput prediction.




(a) Rate Control (b) Congestion Control

(c) Rate Control with delay (d) Mixed Congestion Control

**Figure 1. TCP data transfer patterns**

### 2.2. TCP Segment Partitioning

The throughput prediction algorithms developed in this paper are based on an analysis of a time series of throughput values. In a typical TCP transfer, the sender sends a window full of data segments and then waits for an acknowledgment. Thus, TCP based patterns exist between number of segments sent per RTT, and the corresponding throughput, which is the time series to work with for throughput prediction. If the throughput is measured as the amount of data received in a fixed time period (we refer to this as *Fixed Interval Throughput*), or the instantaneous throughput based on the amount of data in a segment and the time since the last segment was received (we refer to this as *Instant Throughput*), then there will be large meaningless fluctuations in the throughput time series. This is illustrated in Figure 2. Instant throughput shows a huge variation (from 220 Bytes/sec to 1 GBytes/sec) and fixed interval throughput over 100 millisecond shows a large variation (from 40 KBytes/sec to 121 KBytes/sec) when the data transfered per RTT, and the corresponding per RTT throughput, are steady around 70 KBytes per second.

Clearly, we need to measure the data transferred per RTT to develop a meaningful time series for throughput prediction. For this we need to determine the RTT, which itself could be changing during the course of a TCP transmission. Many RTT estimation techniques have been proposed in literature [7,8]. We have chosen to employ the method of Jiang and Dovrolis [7] based on SYN-ACK (SA) RTT estimation. Ideally RTT should be updated continuously but we are currently using a fixed RTT. Our empirical observation is that a fixed SA RTT partitioning is a simple and effective method for our goals.

We partition the sequence of segments in chunks that are received per RTT, and generate a new per-RTT throughput series as follows:

The collected info is a time series, $TS$, of TCP segment arrival times and payload sizes. $TS_i = (T_i, B_i)$, where $T_i$

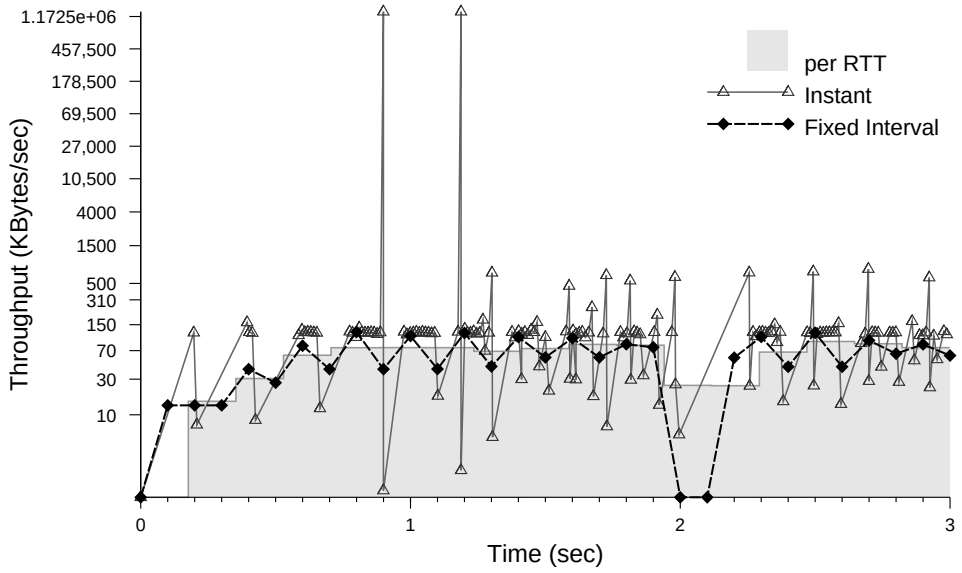



**Figure 2. Different throughput series for a TCP flow. per RTT throughput is based on data received over a measured fixed RTT of 176 msecs. Instant throughput is based on the size of a data block and the time elapsed since the previous block was received. Fixed Interval throughput is based on the data received in a trailing window of 100 msecs. Note that the Y-axis is shown in log scale.**

is the $i$-$th$ segment's arrival time and $B_i$ is its payload size. Then we apply fixed SA RTT partitioning to $TS_i$ and a new time series $RTh$ is generated, where $RTh_j$ is the throughput over the $j$-$th$ RTT.

The new time series is based on the amount of data that arrived in each RTT. We can think of it as a throughput time series, where each data point is the average throughput over one RTT.

## 2.3. Throughput Prediction using TCP Patterns

After partitioning, we have a time series of past throughput values. The goal is to predict average future throughput for the TCP stream. Our first objective is to identify TCP patterns in the throughput time series illustrated in Figure 1. We restate the main patterns.

- **Rate Control limited (RC)**: the receiving rate of TCP segments is limited by receiver's buffer size (flow control) and/or sender's management of outgoing TCP streams, or is application limited [21].
- **Congestion Control limited (CC)**: the receiving rate of TCP segments is limited by network's condition and driven by the repeating sequence of packet loss followed by slow start and congestion avoidance phases.

Figures 1(a) and 1(c) show typical Rate Control limited patterns. Figure 1(b) shows a Congestion Control limited pattern. Figure 1(d) shows varying behavior, possibly rate control limited with occasional packet loss.

The throughput prediction algorithm attempts to find the main characteristics of the throughput series as follows:

- *Flat regions:* where throughput practically remains constant.
- *Linear Climb regions:* where throughput climbs steadily as in congestion avoidance.
- *Exponential Climb regions:* where throughput climbs exponentially as in slow start.
- *Drop points:* where throughput drops to half or more of previous values.

The identification of these regions is done in a forgiving manner since the data transfer patterns rarely follow strict TCP patterns in practice. The simple moving average of a small window of time series values is used for identification instead of individual values. This ensures that a small network delay or variation in RTT will generally not prevent a pattern from being identified.

Based on the above factors, the point where the slow start ends, regions of fixed rate flow, and congestion control cycles, are identified. We will refer to the point where the slow start phase of a flow ends as $T_{PS}$, representing the peak of slow start.

Given a window size $w$ RTTs worth of TCP segments, which means $w$ throughput values in the time series, the general rules for throughput prediction are the following:

1. **RC-based prediction**: If a TCP flow is flat from $T_{PS}$ through the remaining window to series value $w$, the

predicted throughput is the average throughput from $T_{PS}$ to $w$.

2. **CC-based prediction**: If the rule for RC-based prediction does not apply and a TCP flow has more than three complete congestion control cycles after $T_{PS}$, the predicted throughput is the average throughput over the most recent three complete congestion control cycles.

3. **Window-based prediction**: If the rules for RC and CC based prediction do not apply, the predicted throughput is the average throughput from $T_{PS}$ to $w$.

This description obviously does not cover all possible cases (e.g., special cases when no slow start is detected or slow start consumes the entire window) or full details, but it captures the core of the throughput prediction algorithm. Basically, the initial slow start phase is not used for prediction since it is not representative of normal behavior. Subsequently, if a congestion control pattern is detected, that information is used for prediction, else averaging over the relevant region is used.

### 2.4. Illustration of Throughput Prediction

Figure 3 and Figure 4 illustrate throughput prediction for two time series obtained from actual TCP traces. The durations of the traces in Figure 3 and Figure 4 are 475 RTTs and 1479 RTTs, respectively. The figures show predicted throughput and the error between predicted and measured throughput for varying window sizes used for prediction. The window always starts from the 1st RTT and goes up to 100 RTTs. Predictions are always made for the next 100 RTTs. Predictions are made by our method labeled "TCP Pattern" as well as with 3 other methods for which the predicted throughput is: 1) Average (or Aggregate) throughput in the full window, 2) Simple moving average of the last 10 throughput values, and 3) Exponential weighted moving average (EWMA) of the last 10 throughput values.

In Figure 3, the rule for RC-based prediction is used from 15th to 25th RTTs. After a significant drop of throughput at the 27th RTT (due to a lost segment), the Rate Control pattern does not hold, but no other pattern is identified. Hence, the default rule for Window-based prediction is used from the 30th RTT onwards. For this example, the pattern based prediction has lower errors than others as it excludes the slow start phase for prediction, and averages over a longer period than moving average approaches.

In Figure 4, Congestion control based rule is used for prediction from the 65th RTT onwards as 3 complete CC cycles (14-29, 29-43, and 43-63) are observed. Note that the throughput pattern is far from textbook congestion control pattern, but the algorithm makes the judgment that the pattern is best identified as congestion control.

**Table 1. Characteristics of collected traces**

| Terms | Values | Comments |
|---|---|---|
| Number of traces | 461 | |
| Downloaded file size | 26–34 MB | Average: 30 MB |
| Unique web sites | 290 | Debian/Gentoo mirrors |
| Average number of segments per trace | 24,062 | (min/max/median) = (17,025/69,866/24,412) |
| Retransmitted segments | 0.09% | 97 out 461 traces |
| Average number of retransmitted segments per trace | 103.6 | 97 traces (min/max/median) = (0/2,672/4) |
| Average SA RTT | 0.1696 sec | (min/max/median) = (0.02/2.91/0.155) |
| Average number of RTTs per trace | 2,589 | (min/max/median) = (143/110,673/662) |

## 3. Experiments and Results

The goal of our experiments is to validate TCP pattern-based throughput prediction with TCP traces obtained by downloading large files from web servers. We compare throughput predictions made by the TCP pattern based procedure against simple prediction methods like simple Moving Average, Exponential Weighted Moving Average (EWMA), and Aggregate throughput prediction, which is the average throughput over the entire time series that is available for prediction. A small part of the trace is used to make a prediction about future throughput, and measured throughput is compared against predicted throughput. The average error in prediction is used to compare the accuracy of prediction methods. The experiments consisted of three stages: trace collection, throughput prediction, and result analysis.

### 3.1. Experimental Setup

A total of 461 TCP transfer traces were collected from two machines running RedHat Linux located at the University of Houston. These two machines used default TCP window size, since the environment is set for a typical Internet user without any special tuning. Files with an average size of 30 MBytes were downloaded using wget [11] from mirror sites of Debian Linux [1] and Gentoo Linux [3]. TCP headers of those traces were collected using tcpdump [6]. University of Houston has two network uplinks to the Internet, one with a capacity of 30 Mbps via Verio and the other with a capacity of 622 Mbps via Texas GigaPOP (Internet2). The characteristics of these traces are summarized in Table 1.

Traces are then processed with a perl script to compute SA RTTs and a time series consisting of segment arrival times and the number of bytes in a segment. The computed value of RTTs for the traces varies from 0.02 to 2.91 seconds. Retransmitted segments are included. The number of retransmitted segments is only 0.09% of the total number segments, so they have little impact on throughput predic-

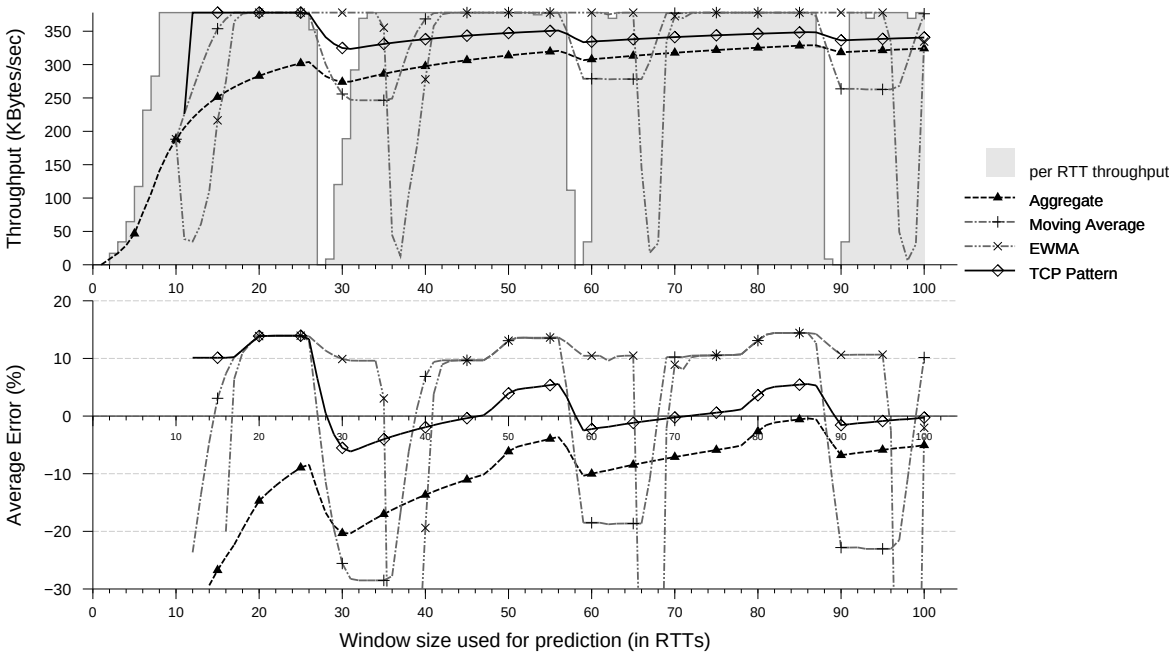



**Figure 3. Illustration of throughput prediction and prediction error with TCP pattern and other methods for a Rate Controlled flow. All predictions are made for the next 100 RTTs.**

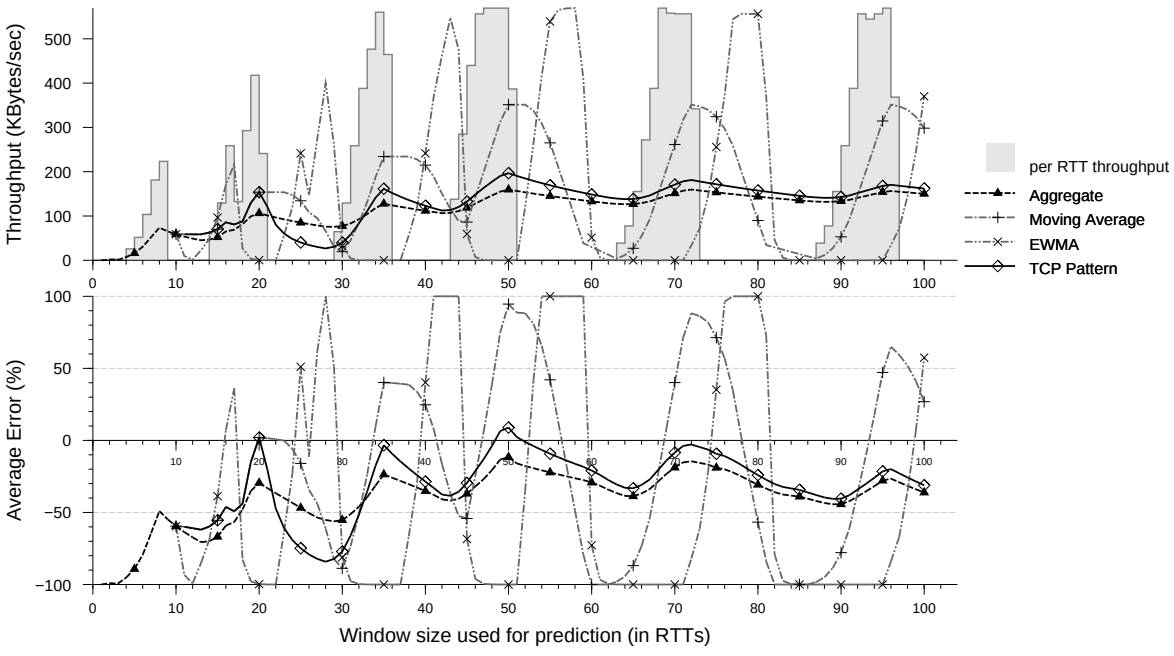



**Figure 4. Illustration of throughput prediction and prediction error with TCP pattern and other methods for a Congestion Controlled flow. All predictions are made for the next 100 RTTs.**

**Table 2. Throughput prediction methods**

| Name | Comments |
|---|---|
| Moving Average | previous 10 RTTs |
| Exponential Weighted Moving Average (EWMA) | previous 10 RTTs weight = 0.125 |
| Aggregate throughput | average over entire prediction window |
| TCP Pattern prediction | |

tion. This information is converted to a time series consisting of per-RTT throughput values.

We then make throughput prediction at every RTT using TCP pattern-based prediction, Moving Average, Exponential Weighted Moving Average (EWMA), and Aggregate Throughput. The parameters for these methods are described in Table 2.

The average error for evaluating throughput prediction is calculated as:

$$\frac{predicted\ throughput - measured\ throughput}{measured\ throughput} \times 100\%$$

For illustration, in Figure 4, at the 30th RTT, the predicted throughput is made based on packets collected up to the 30th RTTs, and the measured throughput is the average throughput between 31th and 130th RTTs. If the measured throughput is close to zero, the average error could rise to infinity. As this is an artifact of the computation methodology, all errors larger than 100% are set to 100%. For example, in Figure 4, at the 55th RTT, the average error for EWMA is over 100% and is set to 100%.

### 3.2. Results and Discussion

We analyzed 461 traces and the average prediction errors over these traces are illustrated in Figures 5 and 6. Since we make throughput prediction at every RTT, it is more convenient to use RTT as the base unit to compare average errors across traces than a fixed amount of time. In Figure 5, the window of data available for prediction is varied from 1 to 100 RTTs and a prediction is made for the duration of 200 RTTs after the point at which the prediction is made.

Figure 5 shows that the average prediction error for TCP Pattern based prediction is the lowest or close to the lowest for all prediction window sizes. The average error for the better prediction algorithms varies from around 12% to 20% depending on the prediction window size. As expected, the error with the best methods is lower for larger prediction window sizes since more data is available for prediction. However, the prediction by Moving Average and EWMA methods is not affected much since they only look at a fixed window in the past.

For small window sizes, the average error for Moving Average is about the same as TCP Pattern prediction. For large window sizes Aggregate prediction is very close to TCP Pattern prediction. EWMA predictions were less accurate than the other methods on average. The results of prediction for the next 100 and 400 RTTs were similar to those reported for 200 RTTs in Figure 5.

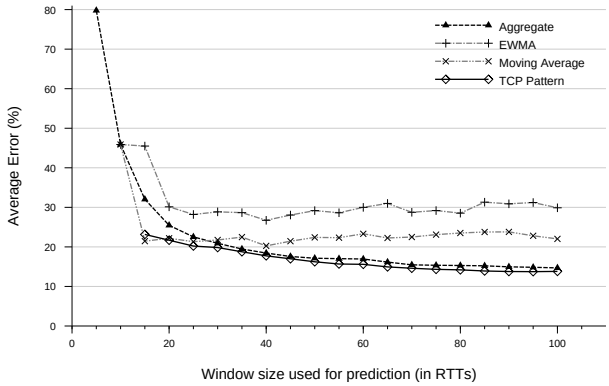



**Figure 5. Performance of different throughput prediction methods. All predictions are made for the next 200 RTTs.**

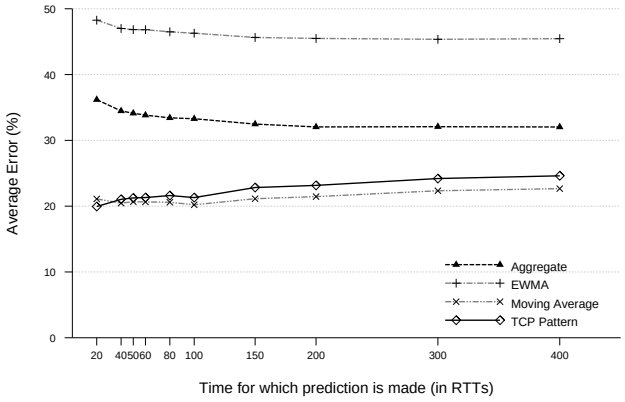
To examine the relationship between prediction accuracy and different prediction methods, we look at Figure 6. For each graph in the figure, the window of RTTs available to make predictions is fixed, while the length of the future for which predictions are made is varied from 20 to 400 RTTs.

In Figure 6(a), the prediction window is 15 RTTs. We observe that Moving Average provides the best prediction in this scenario, slightly better than TCP Pattern prediction. A window size of 15 RTTs is frequently too short for pattern based algorithms. The Aggregate method is not a particularly good predictor because the average typically includes slow start which leads to an underestimate of future throughput.

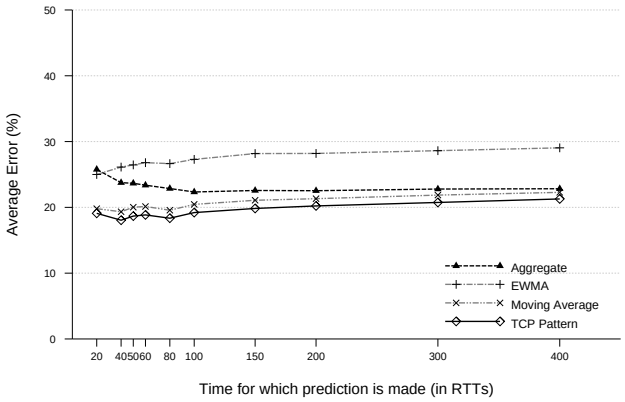
The prediction window is increased to 25 RTTs in Figure 6(b). In this case the TCP Pattern based algorithm is the best predictor, slightly better than Moving Average prediction. The Aggregate method is a better predictor for prediction window size of 25 RTTs than the case of prediction window size of 15 RTTs, since the impact of slow start is not as pronounced on average.

We finally look at Figure 6(c) with a prediction window size of 50 RTTs. In this case again, the TCP Pattern based approach provides the best prediction. The Aggregate method also provides predictions close to the best. The reason is that when a longer window of past historical data is available, historical average tends to become a good predictor. Moving Average method performs worse for a smaller prediction window since it only looks at a fixed window and cannot benefit from the additional information that is available.

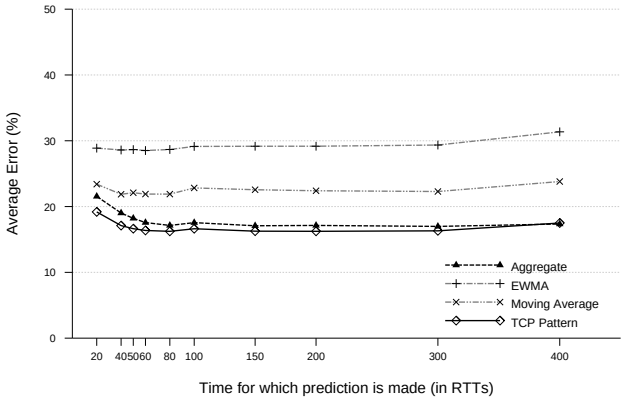
We summarize our observations.




(a) Prediction is made at 15th RTT




(b) Prediction is made at 25th RTT




(c) Prediction is made at 50th RTT

**Figure 6. Performance of different throughput prediction methods. Figures represent predictions made after observing 15, 25, and 50 RTTs of download data.**

- The TCP-pattern based throughput prediction method developed in this paper performs as well or better than other methods in all scenarios.
- Throughput prediction using average throughput over a window of time in the past works well if the window size is large enough. This observation implies that bandwidth measurement tools like Iperf and NWS need to run over a relatively long period of time in order to get accurate measurements.
- Good predictions can be made with window sizes in the range of 25 RTTs. This implies that good predictions are possible within 5 seconds or less for most TCP transfers. However, prediction accuracy does improve modestly as prediction window size is increased.
- Finally, the best predictions from data at the beginning of a TCP transfer still have an average error in the range of 15-20%. The implication is that, although good predictions are possible, there are inherent limits on throughput predictability on the Internet.

## 4. Conclusion

We have presented a simple approach for fast TCP throughput prediction that samples arrival of TCP segments and exploits knowledge of TCP patterns to get good predictions. A fast and accurate predictor of TCP throughput has a role in grid resource management and many network applications. Bandwidth measurement by active probing, like the one used in NWS, can benefit from this research. By analyzing the data pattern generated by the probe, rather than simple averaging, a more accurate throughput prediction can be made.

This is an early report on work in progress. There are several ways in which our predictions can be potentially improved. The process primarily consists of identifying more specific patterns and studying the implications of recognizing the patterns for throughput prediction.

Even in situations where another method is likely to provide as good average results, our method offers some advantages. In general, our approach provides a more accurate prediction when a pattern is identified, and a less accurate one when a heuristic is used for prediction. Hence, we can attach a degree of confidence to our predictors to improve their usage. Also, the method is capable of judging if enough data is available to make a reliable prediction, or more of the transfer should be analyzed. These aspects of research are ongoing.

The throughput prediction described in this paper is applied to a single TCP stream. Some transfers in grid environment utilize multiple TCP streams and a tuned TCP window size on the host machines. Both these aspects are being addressed in ongoing experiments.

The final goal is an online tool to provide real-time throughput predictions for TCP transfers within a few RTTs. This paper has developed a scientific basis for such a tool.

## Acknowledgment

This research was supported, in part, by the National Science Foundation under Grant No. ACI-0234328 and Grant No. CNS-0410797. Support was also provided by the Department of Energy through Los Alamos National Laboratory (LANL) contract number 03891-99-23 , and by University of Houston's Texas Learning and Computation Center. We wish to thank numerous current and former members of our research group for their contribution to this work especially Yu-Ren Chung who contributed to the early stages of this project.

## References

[1] Debian. Debian linux mirrors. `http://www.debian.org/mirror/list`, November 2004.

[2] Mark Gates, Alex Warshavsky, and Justin Pietsch. Iperf version 1.7.0. `http://dast.nlanr.net/Projects/Iperf/`, 2003.

[3] Gentoo. Gentoo linux mirrors. `http://www.gentoo.org/main/en/mirrors.xml`, November 2004.

[4] Globus. Global access to secondary storage (GASS). `http://www-fp.globus.org/gass/`, 2004.

[5] Globus. GridFTP. `http://www.globus.org/datagrid/gridftp.html`, 2004.

[6] Van Jacobson. TCPDUMP. *Unix Manual Page*, 1990.

[7] Hao Jiang and Constantinos Dovrolis. Passive estimation of tcp round-trip times. In *Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM-02)*, volume 32, 3 of *Computer Communication Review*, pages 75–88, New York, July 2002. ACM Press.

[8] Phil Karn and Craig Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems*, 9(4):364–373, November 1991.

[9] Bruce Lowekamp, Brian Tierney, Les Cottrell, Richard Hughes-Jones, Thilo Kielmann, and Martin Swany. A hierarchy of network performance characteristics for grid applications and services. *Global Grid Forum (GGF), GFD.23*, May 2004.

[10] Matt Mathis, Jeffrey Semke, and Jamshid Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), July 1997.

[11] Hrvoje Niksic. Wget version 1.8.1. `http://www.gnu.org/software/wget/wget.html`, 2003.

[12] Pascale Primet, Robert Harakaly, and Franck Bonnassieux. Experiments of network throughput measurement and forecasting using the network weather service. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID2002)*, pages 383–388, Berlin, Gemany, May 21–24 2002. IEEE.

[13] Aimin Sang and San-qi Li. A predictability analysis of network traffic. In *Proceedings of the 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM-00)*, pages 342–351, 2000.

[14] Mehmet Sayal, Yuri Breitbart, Peter Scheuermann, and Radek Vingralek. Selection algorithms for replicated web servers. *ACM Performance Evaluation Review*, 26(3):44–50, December 1998.

[15] Sprint. IP monitoring project (IPMON), `http://ipmon.sprint.com/`, 2003.

[16] Martin Swany and Rich Wolski. Multivariate resource performance forecasting in the network weather service. In *SC'2002 Conference CD*, Baltimore, MD, November 2002. IEEE/ACM SIGARCH.

[17] Sudharshan Vazhkudai and Jennifer M. Schopf. Using regression techniques to predict large data transfers. *Distributed, Parallel, and Cluster Computing*, April 23 2003.

[18] Sudharshan Vazhkudai, Jennifer M. Schopf, and Ian Foster. Predicting the performance of wide area data transfers. In *16th International Parallel and Distributed Processing Symposium (IPDPS '02 (IPPS & SPDP))*, page 34, Washington - Brussels - Tokyo, April 2002. IEEE.

[19] Sudharshan Vazhkudai, Steven Tuecke, and Ian Foster. Replica selection in the globus data grid. In *Proceedings of the 1st International Symposium on Cluster Computing and the Grid (CCGrid 2001)*, pages 106–113, Brisbane, Australia, May 15–18 2001. IEEE Computer Society Press.

[20] Rich Wolski, Neil Spring, and Chris Peterson. Implementing a performance forecasting system for metacomputing: The network weather service. In *Proceedings of Supercomputing'97 (CD-ROM)*, San Jose, CA, November 1997. ACM SIGARCH and IEEE.

[21] Yin Zhang, Lee Breslau, Vern Paxson, and Scott Shenker. On the characteristics and origins of internet flow rates. In John Wroclawski, editor, *Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM-02)*, volume 32, 4 of *Computer Communication Review*, pages 309–322, New York, August 19–23 2002. ACM Press.