Reducing the Transfer Time for Large Files in High Performance Networks

XueJun Mao, Victor Frost, Joseph Evans, Mahesh Akarapu
Dept. of Electrical Engineering and Computer Science
University of Kansas, Lawrence, KS 66045

Abstract

In this project, we propose a system architecture which uses an advice server to improve the performance of very large file transfers in high performance networks; the advice server predicts the behavior of changes in available bandwidth and responds to the data server accordingly. It is assumed that the network will exhibit a periodic fluctuation, a linear relationship will be used to model the change in bandwidth over short time intervals. The advice server will collect network measurement data and record into database for network traffic and congestion state modeling. Here, the general mathematical theory for the algorithm will be introduced first; then the algorithm used by the advice server will be presented.

Introduction

In the Internet, file transfer is one of the major applications; for some research applications, very large files are common, e.g., multiple GB files. During the entire file transfer time, the bandwidth of the link and the congestion states of each hop may change. Beyond the short term response of TCP, the standard FTP program does not provide dynamic adjustments to respond to these changes. To decrease the total file transfer time, such a long term traffic flow needs a different congestion control mechanism as compared to short term traffic flows [1].

Some previous work has been performed to improve the transfer rate while avoiding congestion, for example, controlling the transfer rate as a function of the loss, or the congestion state [2]. Other methods use parallel sockets to transfer different segments of a file simultaneously, or splitting the file into several servers to provide these segments [8]. Other more complicated techniques use the different traffic models. But all these methods require significant network configuration or complex algorithms. We intend to obtain performance improvement by changing the ftp server. For the transfer of large file, the long term traffic behavior is more important. The available bandwidth is composed of two parts: one is stable changing part; another is a random variable that reflects the burst nature of network traffic, here we assume the second one is noise.

It has been previously demonstrated that explicit congestion notification can improve performance for large file transfer [4]. In the system architecture in [4], the state of the network is probed by a separate server and notifies the data server when required. The network probing and advising service is proved valuable to enhance the performance of data server, e.g., one implementation is Network Characterization Service (NCS) [6]

Many tools are available to perform network testing, such as iperf[12], pchar[13], pipechar[11]. While these tools can provide information about current network state, they can not predict traffic trends, which is necessary to adjust parameter during large file transfer. A database is needed to collect all the data from these tests, while file transfer parameters are adjusted according to the network traffic history data with certain algorithm. This database will record all the information collected by the measurement tools. We seek a balance between the congestion control and high performance. Hence the network state determination is an important aspect of this project; from the network state, we can calculate the proper parameters for file transfer.

In this project, we model the changes in available bandwidth and use the result to improve large file transfer performance, the goal is to reduce total transfer time. The congestion state for file transfer will be determined by the response obtained from an advice server. Moreover, one of important methods to improve performance is to use the optimal TCP buffer size. The optimal TCP buffer size can be calculated by following formula [3]:

```
optimal TCP buffer = RTT * (bandwidth of bottleneck link)
```

For relatively short files, bandwidth can be assumed to be constant; but for very large files, a model is needed to build to predict when to adjust the optimal TCP buffer.

Another hypothesis here is that if there is no congestion in the link, we can turn off congestion control and set the optimal TCP buffer size to get better performance. So we also monitor the congestion state of the link from server to client. The architecture of the system considered is shown in Figure 1.

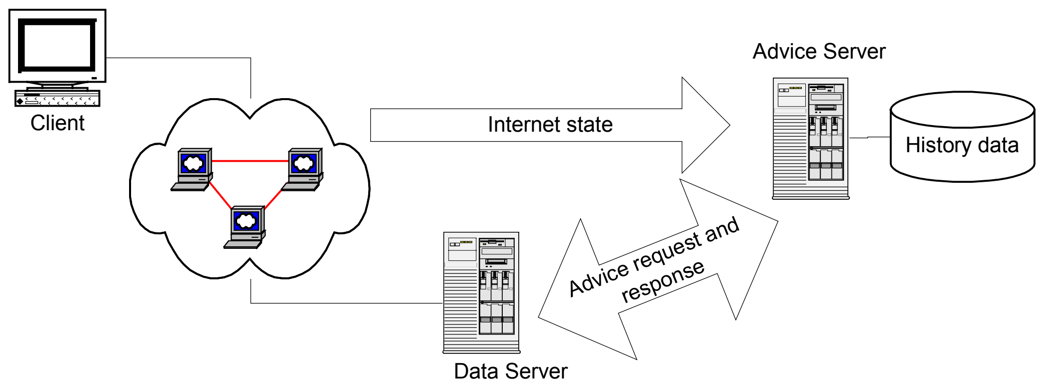

Figure 1 System architecture

The important components are:

- Data server: data server stores the file to be transferred and provides service for customers. The data server is unaware of the network state, it requires the advice server to provide parameters for file transfer and communicates with client to set these parameters.
- Advice server: advice server continuously collects the network state, the results are stored in a database. When the data server sends an advice request, the advice server analyzes the data in database and provides suggested parameters for the transfer.

Background and Principle

The data collected of available bandwidth from internet shows that the network traffic exhibits a periodic behavior, although the change in available bandwidth is not smooth, as shown in Figure 2. Typically, the available bandwidth will increase at night and decrease in the day.

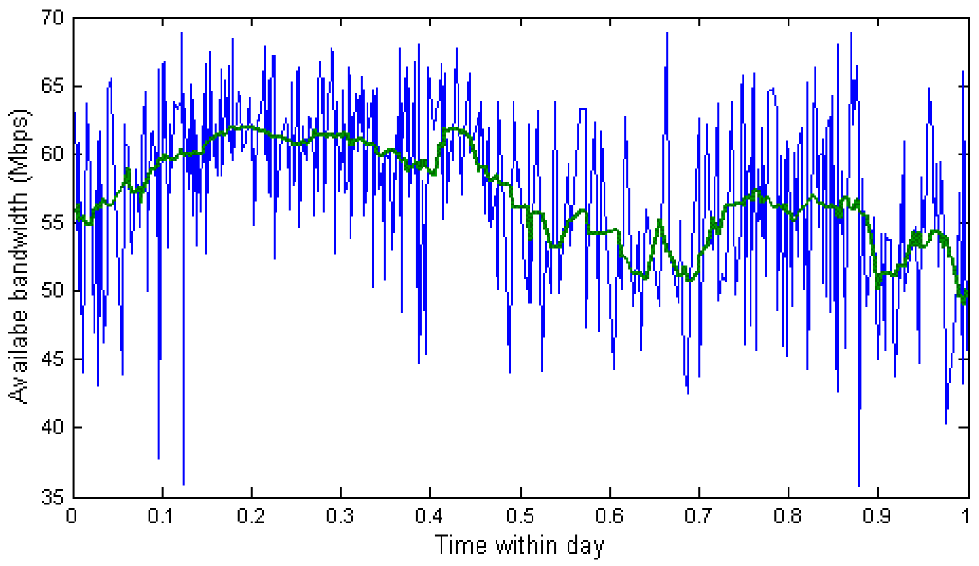

Figure 2 Observed available bandwidth

Figure 2 is a typical change in available bandwidth for a router output port for one day. The thick curve is two hour moving average.

In our model, we regard the available bandwidth as being composed of two parts: one is deterministic; another is random to reflect the burstyness of network traffic, it can be taken as noise. For the first part, a linear regression method is used to model the traffic for short time intervals; that is, at any time we assume a linear relationship between time and available bandwidth over a short period of time. For example, when the advice server receives request at 2:00 am, it calculates the mean available bandwidth and bandwidth rate of change at 2:00 am according to historical data.

Assume the relationship between available bandwidth and time is a linear,

$$b = \beta_0 + \beta_1 \tau$$

b represents available bandwidth

τ represents time relative to the request time

β_0 will be the available bandwidth at the time of the request, β_1 will be rate of change of the available bandwidth, it may be negative.

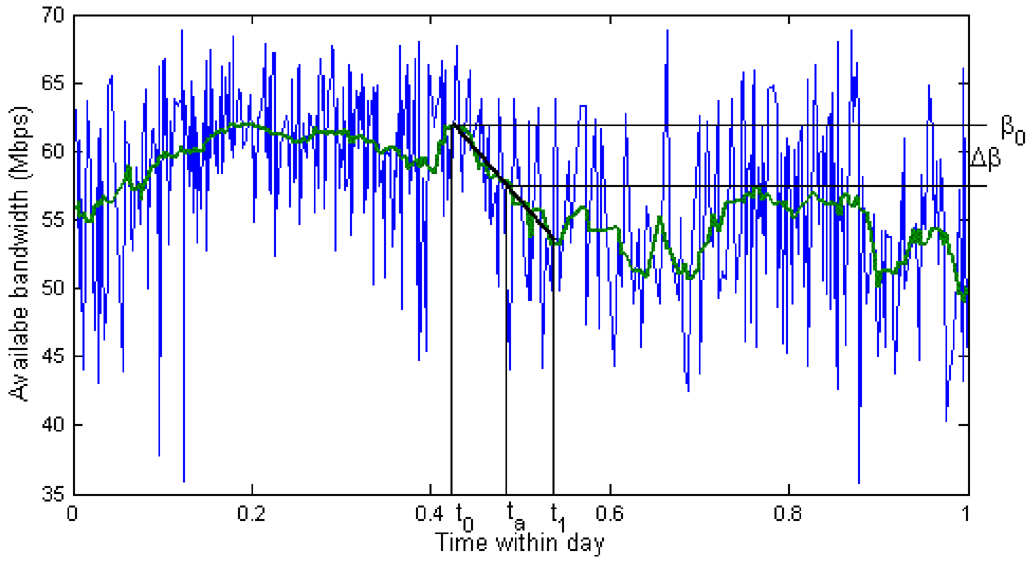

Figure 3 Illustration of linear model

In Figure 3, the advice request arrives at time t_0. The linear regression is done for data between t_0 and t_1 ($\Delta t_1 = t_1 - t_0$ is a configured constant). β_0 will be the available bandwidth at t_0, β_1 is the slope, representing the rate of change in available bandwidth. If we require that the change in available bandwidth will not exceed $\Delta\beta$, then the next advice time will be t_a.

The database in the advice server will contain a series of observed pairs $(b_i, \tau_i), i = 1, \ldots n$, from which the estimators of the coefficients β_0, β_1 can be calculated, that is,

$$\hat{\beta}_1 = \frac{n\sum b_i \tau_i - (\sum b_i)(\sum \tau_i)}{n\sum \tau_i^2 - (\sum \tau_i)^2}$$

$$\hat{\beta}_0 = \frac{\sum b_i - \hat{\beta}_1 \sum \tau_i}{n}$$

The measurement contains noise so that the model becomes

$$b_i = \beta_0 + \beta_1 \tau_i + \varepsilon_i$$

We assume that:

- ε_i is a Gaussian random variable
- Mean value of ε_i is 0, i.e., $E(\varepsilon_i) = 0$
- Variance of ε_i is constant. (Therefore, $\varepsilon_i \sim N(0, \sigma^2)$)
- Correlation of ε_i and $\varepsilon_j (i \neq j)$ is 0
- Correlation of ε_i and b_i is 0

Under these assumptions:

- $\hat{\beta}_1$ is an unbiased estimator of β_1
- $\hat{\beta}_0$ is an unbiased estimator of β_0
- The goodness of fitness is measured by
$$R^2 = 1 - \frac{\Sigma (b_i - (\hat{\beta}_0 + \hat{\beta}_1 \tau_i))^2}{\Sigma b_i^2 - (\Sigma b_i)^2 / n}$$

Implementation of Advice Server

Currently we use pipechar[11] to collect available bandwidth data. Each time pipechar is executed, it reports the available bandwidth and congestion state for each hop from the date server to the potential client. The advice server maintains a database, which records the following observations

$$(ip, \tau, b, c)$$

ip is the ip address of each hop, τ is test time, *b* is measured available bandwidth at this hop and time τ , *c* is a Boolean that indicates whether congestion is occurring (TRUE) or not (FALSE) at this hop and time τ .

When the data server receives a file transfer request from client, it sends a request to the advice server. The request include the following parameters:

- ip address of client
- file length or remaining file length to be transferred

When the advice server receives the request, it determines the following parameters to send back to the data server:

- Flag to indicate whether to use congestion control or not in TCP
- RTT from server to client.
- Current available bandwidth
- Next advice time

RTT can be tested when the advice request arrives at the advice server, either ICMP programming or ping can be used to obtain the result. To get the congestion control flag, the advice server searches for the most recent observation of this destination from database. The result is a series of stored information in the form of

$$(ip_1, \tau, b_1, c_1)$$
$$\ldots\ldots$$
$$(ip_n, \tau, b_n, c_n)$$

Here, ip_is are ip addresses of hops from the server to the destination, b_is are available bandwidth of corresponding hops, c_is are flags of congestion states. If the most recent test time t is not close enough to current time, the congestion control flag is set to inform the data server to use congestion control during transmission. If the nearest test time t is close enough to current time, the congestion control flag is determined by congestion flags. If anyone of them is TRUE, which means congestion exists at this hop, the congestion control flag will be turned on; if none of them is true, the congestion control flag will be turned off.

The available bandwidth and next advice time will be calculated according to historical data of the minimum bandwidth hop. When an advice request arrives, the first step is to calculate the available bandwidth and rate of change of available bandwidth for each previous day with test data.

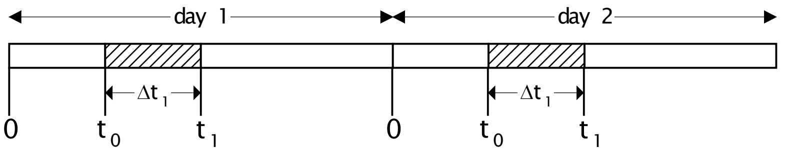

Figure 4 Illustration of calculation

The advice request comes at time t_0 in current day. For each previous day illustrated in Figure 4, we do linear regression for history data from t_0 to $t_1 = t_0 + \Delta t_1$; then we get available bandwidth $\beta_0(i)$ and rate of change in available bandwidth $\beta_1(i)$, i is the index of the previous days.

Current available bandwidth will be estimated as the mean value of series $\beta_0(i)$

$$\beta_0 = \frac{1}{N}\sum_{i=1}^{N}\beta_0(i), \qquad N \text{ is total number of days.}$$

This value will be sent back to the data server.

For the rate of change in available bandwidth $\beta_1(i)$, we calculate the mean value β_1 and standard deviation σ_{β_1}, then the rate of change in available bandwidth will be between $\beta_1 - t_z\sigma_{\beta_1}$ and $\beta_1 + t_z\sigma_{\beta_1}$ (t_z is determined by confident level requirement of normal distribution. According to theory, $\beta_1(i)$ has student-t probability distribution function, hence $\beta_1(i)$ can be approximated by normal distribution. For example, $t_z = 2$ is for 95% confidence).

Denote $S = \max(|\beta_1 - t_z\sigma_{\beta_1}|, |\beta_1 + t_z\sigma_{\beta_1}|)$, S will be the maximum rate of change in available bandwidth. That is, from t_0, after a certain time interval τ, the maximum change in available bandwidth will be $S \times \tau$.

The upper bound of tolerated change in available bandwidth can be determined either by absolute value or percent of estimated bandwidth β_0, denote it as $\Delta\beta$. Then

$$S \times \tau < \Delta\beta \Rightarrow \tau < \Delta\beta / S$$

which gives the next advice time $t_a = t_0 + \tau$.

Data Collection

Currently the traffic data from interface 164.113.234.206 is being monitored. The typical data for one week is shown in Figure 5.

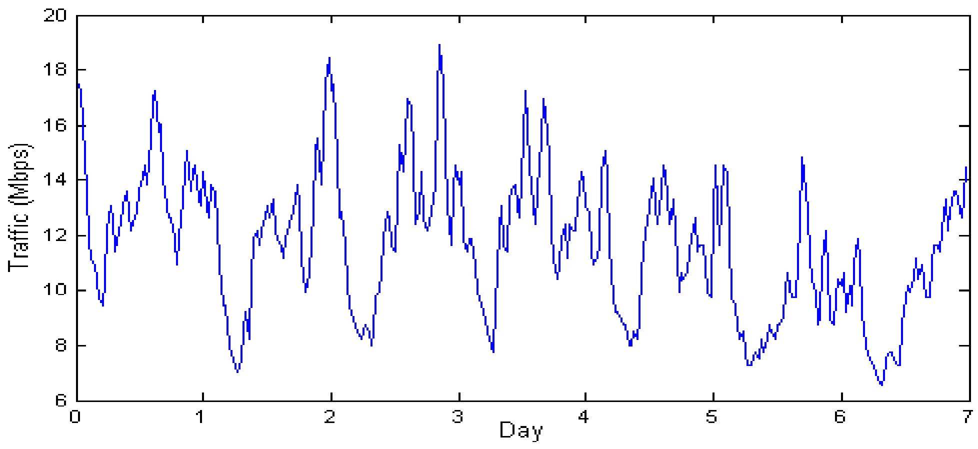

Figure 5

Within one day, the typical data is shown in Figure 6.

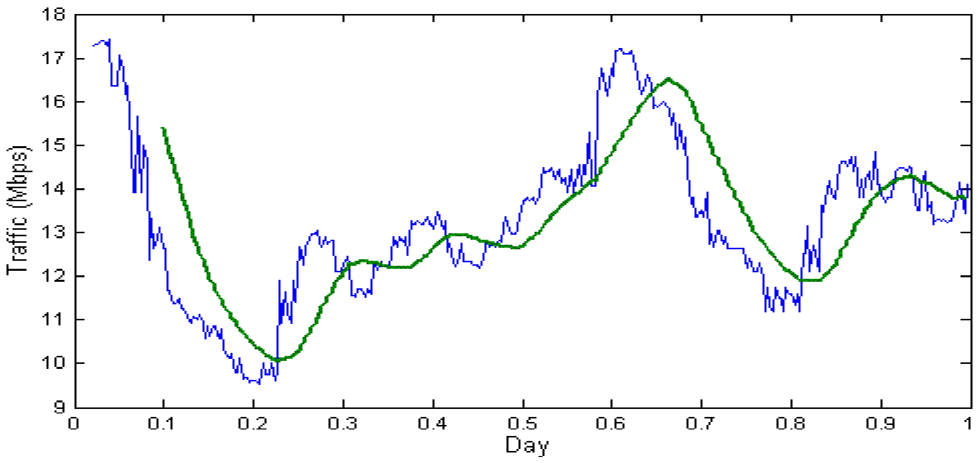

Figure 6

The sampling frequencies is 5 min, all of these data will be collected and encapsulated into database used by advice server. In Figure 6, the dark curve is 2 hours moving average. Comparing with the Figure 2, they match well.

Conclusion

Above discussion focuses on a model of the change in available bandwidth as a function of time of day, which can be used by data server to improve throughput for large file transfers. A great deal of work still needs to be done, especially more detailed analysis of the bandwidth measurements and configure optimal parameters. Further work may be on other traffic models and the effective strategy to test network, organize and update the database. We believe the methodology presented here will reduce the transfer time for large files.

Reference

[1] Anees Shaikh, Jeniffer Rexford, Kang G. Shin, Load-Sensitive Routing of Long-Lived IP Flows, Proc. ACM SIGCOMM, September 1999, pp. 215-226
[2] Carey L. Williamson, Optimizing File Transfer Response Time Using the Loss-Load Curve Congestion Control Mechanism, SIGCOMM 1993, 117-126
[3] Brian L. Tierney, Dan Gunter, Jason Lee, Martin Stoufer, Joseph Evans, Enabling Network-Aware Applications, Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing (HPDC-10), August 2001, LBNL-47611
[4] Kostas Pentikousis, Hussein Badr, and Bilal Kharmah, TCP with ECN: Performance Gains for Large Transfers, SBCS-TR-2001/01, Department of Computer Science, SUNY Stony Brook, March 2001
[5] Shrikrishna Karandikar, Shivkumar Kalyanaraman, Prasad Bagal, TCP Rate Control, Computer Communication Review, V.30, N.1, January 2000
[6] Jin Guojun, George Yang, Brian Crowley, Deb Agarwal, Network Characterization Service(NCS), Lawrence Berkeley National Lab report #47892. 2001
[7] Thomas Gross, Peter Steenkiste, A Perspective on Application/Network Coupling, NOSSDAV 98
[8] Babak S. Noghani, Steve Kretchmen, and Robert D. McLeod, A Novel Approach to Reduce Latency on the Internet: "Component-Based Download"
[9] Van Jacobson, Congestion Avoidance and Control, 1988 ACM O-8979 I-279-9/88/008/03 14
[10] Floyd,S., "TCP and Explicit Congestion Notification", ACM Computer Communication Review, V.24 No.5, p10-23, Oct 1994
[11] pipechar, www-didc.lbl.gov/pipechar
[12] iperf, dast.nlanr.net/Projects/Iperf/
[13] pchar, www.employees.org/~bmah/Software/pchar

[14] Dan Rubenstein, Jim Kurose, Don Towsley, Detecting Shared Congestion of Flows Via End-to-end Measurement, Proceedings of ACM SIGMETRICS'00 , Santa Clara, CA, June 2000

[15] Lixia Zhang, Scott Shenker, David D. Clark, Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-Way Traffic, Proceedings of ACM SIGCOMM '91, September 1991, pp.133-148

[16] Carey L. Williamson, Optimizing File Transfer Response Time Using the Loss-Load Curve Congestion Control Mechanism,

[17] K.Djemame, M.Kara, Agent-Based Rate Coordination Between TCP and ABR Congestion Control Algorithm