# Optimal Adaptive Bandwidth Monitoring for QoS Based Retrieval

Yinzhe Yu, Irene Cheng and Anup Basu (Senior Member)

Department of Computing Science

University of Alberta

Edmonton, AB, T6G 2E8, CANADA

{yinzhe, anup, lin}@cs.ualberta.ca

**ABSTRACT**

**Network aware multimedia delivery applications are a class of applications that provide certain level of QoS guarantees to end users while not assuming underlying network resource reservations. These applications guarantee QoS parameters like media object transmission time limit by actively monitoring the available bandwidth of the network and adapting the object to a target size that can be transmitted within a given time limit. A critical problem is how to obtain an accurate enough estimation of available bandwidth while not wasting too much time in bandwidth testing. In this paper, we present an algorithm to determine optimal amount of bandwidth testing given a probabilistic confidence level for network-aware multimedia object retrieval applications. The model treats the bandwidth testing as sampling from an actual bandwidth population. It uses statistical estimation method to quantify the benefit of each new bandwidth-testing sample, which is used to determine the optimal amount of bandwidth testing by balancing the benefit with the cost of each sample. Our implementation and experiments shows the algorithm determines the optimal amount of bandwidth testing effectively with minimum computation overhead.**

**EDICS: 5 Multimedia Communication and Networking (5-QOSV Quality of Service)**

# I. INTRODUCTION

The Internet was originally designed to offer only one level of service ("best effort") to all its service users. In such an environment, all the data packets put onto the Internet has the same priority; the networks do not provide any guarantee on when the packet will be delivered or even whether it will be delivered at all. However, as new applications continue to emerge, more and more applications now need to provide some kind of quality guarantees to its users. How to provide this consistent service quality is now an active research area termed "Quality of Service" (QoS) [1-5]. One solution to the problem is to allow the application to reserve a certain quality of network services in advance [6,7,8]. Once such a level of service is reserved, the underlying network will commit the corresponding resources needed and guarantee the delivery of such service. Unfortunately, some network architecture and implementations may not support reservation or only support reservation to a limited degree. One alternative approach that has generated a lot of interest in QoS based multimedia delivery application is the so-called network-aware application [9,10,11]. These applications do not rely on the underlying network to provide them a guaranteed quality. Instead, they actively monitor the performance variation of the network, and attempt to adjust its resource demands in response. For example, if an application needs to deliver a certain multimedia object to a client, the application adapts the volume of data (in turn the quality of media) to be transferred to the user given different network bandwidth availability. If the bandwidth is low, the application reduces the size of the object and thus reduces demands on the network; and if the bandwidth is high, the application increases its demand to fully utilize the additional resource. In such a method, the application can guarantee to deliver the object to its client within a pre-defined time limit. Several prototype systems have

demonstrated this idea. In [12], the authors present a Tele-Learning system that deliver multimedia course materials (including JPEG images) to students over the Internet. The system allows the users to specify QoS parameters (time limit, resolution and quality of image) at the client side, and adapt the resolution and quality of the requested image on the server side to provide best quality image to the user while restricting transmission time within the time limit.

Generally speaking, in order to deliver the user-requested media adaptively, the network-aware application needs to go through three phases. First, the application needs to monitor the current network condition and estimate the available bandwidth in a short future period during which requested media will be transmitted. Second, given the specified time limit and estimated bandwidth, the application can calculate the volume of data the network is able to handle, and then adapts the media to that size. For example, if the media is a JPEG image, the application can adapt the image in terms of resolution, JPEG compression factor to produce a tailored JPEG image with a certain target size. Then in the third phase, the tailored file is transmitted to a user. Obviously, the bandwidth monitoring and estimation phase is of critical importance. Only an accurate estimation of available bandwidth allows the user request to be precisely met.

Two approaches exist to address the problem of accurate bandwidth estimation. In the first approach, the media server maintains a connection for each client currently active. The server sends testing packets periodically to the client and calculates the available bandwidth based on the time to transmit the testing packet. This maintains a history of bandwidth samples. When media delivery request is made to the server, the server estimates the available bandwidth of the future period by using a type of average of the history bandwidth samples. This method is used

in [12]. However, there are several problems with it. Since the method uses a relatively long history of bandwidth to predict future bandwidth, the estimation result is not very accurate. And, the approach assumes the availability of history bandwidth information which sometime just not the case. More seriously, periodically sending random bits along the network for bandwidth testing purpose when no real data transmission is actually required can be a significant waste of total bandwidth of networks, since it may interfere with traffic of other users running congestion control protocol.

To address the problem mentioned above, we could measure the bandwidth "on the fly." When the client requests a multimedia object and specifies a time limit, the server first spends a fraction of the time on bandwidth testing. The remaining portion of the time limit is then used to adapt and transmit the media. In such a method, the bandwidth information obtained is most up to date; the entire traffic including both bandwidth testing and real transmission will be bounded by the user specified time limit; and there will not be any unnecessary bandwidth testing when transmission is not required. Because of the congestion avoidance mechanism of various TCP protocol implementations, the throughput achieved by a connection tends to fluctuate around a range close to the maximum bandwidth available over the connection. Therefore, by taking multiple bandwidth samples at the starting fraction of a time interval, we can make a best estimation of the bandwidth available in the whole time period. A limitation of this method is that since we use the bandwidth samples at the beginning of a time interval to represent the entire duration of the time interval, the interval cannot be very long; otherwise the overall traffic over the entire path may change significantly and unpredictably influence the available bandwidth. As we shall see later, in our experiments, 10-30 second time-intervals seem to be appropriate for our

experimental environment.

A problem associated with the proposed approach is that the fraction of time limit used for bandwidth testing is a pure overhead, reducing the portion of user specified time that really used for actual data transmission. The challenge is one of balancing the benefits of more accurate estimation of bandwidth (at the cost of more bandwidth testing time) and more transmission time (at the cost of less accurate estimation).

In this research, we propose a mathematical model to quantify the benefit and loss of bandwidth estimation spending and use it to achieve an optimal proportion of bandwidth testing. Bandwidth testing is treated as sampling from a population, and providing estimates based on "probabilistic confidence levels." When each new sample is taken, we calculate how much new knowledge of the population has been gained. We then compare this benefit with the cost of one sample (the time spent). We continue the sampling until the marginal benefit is less than the cost.

We implement the method in a Java program and test it on two typical network environments, campus network and dial-up network. To our knowledge, this is the first research that provides quantitative results as to how effective the bandwidth monitoring part is in a network-aware application.

We present the mathematical model in Section II and describe our implementation and experimentation in Section III. Section IV gives a comparison of our research to related works by other authors. Section V is the conclusion of the paper.

## II. MATHEMATICAL MODEL

We assume a server and a client need to make a one-off transmission of a multimedia object (from server to client). No previous bandwidth information is available for the connection between the server and client. The user on the client side has specified a time limit $T$ for the transmission. Neither exceeding the time limit $T$ nor under-utilizing the time $T$ is desirable. The first fraction $t$ of $T$ will be used for bandwidth testing to obtain the bandwidth estimation $B$ for the future period $T-t$. The server then adapts the multimedia object to be exactly of the size $(T-t)\cdot B$ and then transmits it to the client. An over-estimation of bandwidth will make the transmission exceed the time limit, while under-estimation will under-utilize the time limit, thus actually delivering a media object that could have had better quality (with a larger size). Our problem is how to determine the appropriate $t$, and deliver as much data as possible to the client within the time limit specified. To simplify the model, we ignore the time needed for adaptation of a media object, since objects can be stored with multiple resolution levels or be dynamically adapted in a short time given a fast processor.

To estimate the bandwidth, we use slices of equal length. Suppose each time slice has length $t_s$, we then have $T/t_s$ time slices. Each time slice has a bandwidth value $x_i = \frac{C_i}{t_s}$ (i=1,2…$T/t_s$), where $C_i$ is a count of bytes received during the i$^{th}$ time slice. These bandwidth values can be viewed as random variables. We obtain the bandwidth value of the first $t/t_s$ slices, and then use the average of these samples to estimate the average of the $T/t_s$ population.

We first define the following variables for further discussion.

**DEFINITION 1: Basic Notation**

- $T$ is the time limit for the multimedia object transmission specified by a user.
- $t$ is the time used for bandwidth testing.
- $t_s$ is the length of each time slice used for obtaining bandwidth samples.
- $N = \frac{T}{t_s}$, $N$ is the number of time slices in period T, which generates our bandwidth *population*, $X_1,\ X_2,...,X_N$.
- $n = \frac{t}{t_s}$, $n$ is the number of time slices in period t, which generates our bandwidth *samples,* $x_1,\ x_2,...,\ x_n$.
- $\mu = \frac{\sum_{N} X_i}{N}, \mu$ is the average of N bandwidth population, the actual average bandwidth we are trying to calculate.
- $\sigma^2 = \frac{\sum_{N} (x_i - \mu)^2}{N}, \sigma^2$ is the variance of N bandwidth population.
- $\bar{x} = \frac{\sum_{n} x_i}{n}$, $\bar{x}$ is the average of the n bandwidth samples $x_1,\ x_2,...,\ x_n$.
- $s^2 = \frac{\sum_{n} (x_i - \bar{x})^2}{n-1}, s^2$ is the variance of the n bandwidth samples.

With the notation defined above, our problem can be stated as "given n, N, $\bar{x}$, $s^2$, estimate $\mu$."

In statistical practice, $\bar{x}$ is usually used as an estimate of $\mu$. Actually, given n, N, $\bar{x}$ and $s^2$, the

random variable $\mu$ has a probability distribution centered at $\bar{x}$. If we assume the random variable X (the bandwidth in one slice) is a normal variable (i.e., follows the normal distribution), then according to statistical theory, $d = \frac{\mu - \bar{x}}{\frac{s}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}}}$ (1) is a continuous random variable following the "Student's t-Distribution" with degree of freedom *n* (number of samples) [13]. [1] We use character *d* instead of *t* to specify the t-Distribution random variable to avoid name confliction with the time *t* we have already defined.

We believe that the assumption of normality of variable x is reasonable, since within a short period the available bandwidth tends to approach a constant. Moreover, the t-distribution is said to be quite "robust", which means that the assumption of normality of X can be relaxed without significantly changing the sampling distribution of the t-distribution (Equation (1)).

Thus given n, N, $\bar{x}$, $s^2$, and t-distribution, we now have the probability distribution of $\mu$.

$$\mu = \bar{x} + d \cdot \frac{s}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}} \quad (2)$$

This means that when we take *n* bandwidth samples, we can expect the average bandwidth within period *T* to be a random variable with probability distribution following Equation (2).

With the distribution of $\mu$, we can infer the $100(1-\alpha)\%$ confidence interval for $\mu$ as follows,

[1] Mathematically, the random variable t is defined as a standardized normal variable z divided by the square root of an independently distributed chi-square variable, which has been divided by its degree of freedom; that is, $t = z / \sqrt{\chi^2 / n}$.

$$\bar{x} - d_{(\alpha,n)} \cdot \frac{s}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}} \le \mu \quad (3)$$

(3) tells us that the value of $\mu$ is greater or equal to $\bar{x} - d_{(\alpha,n)} \cdot \frac{s}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}}$ with a probability of $100(1-\alpha)\%$. With this observation we define bandwidth estimation as follows.

**DEFINITION 2: Safe Bandwidth Estimation**

We define $\mu_{est} = \bar{x} - d_{(\alpha,n)} \cdot \frac{s}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}}$ to be a safe estimation of the future bandwidth. According to the theory described above, we have a confidence of $100(1-\alpha)\%$ that the actual bandwidth will not be lower.

We mentioned that $\mu$ is a random variable centered at $\bar{x}$. From Definition 2, we can see that in order to provide a time limit guarantee we actually under-estimate the bandwidth. This under-estimation has its cost; if we use $\mu_{est}$ as future bandwidth estimation to determine the maximum size of the media object, the actual transmission will most likely under-utilize the remaining time T-t.

It can be shown that the actual time under-utilized is on the average $(T-t) - \left(\frac{\mu_{est} \cdot (T-t)}{\mu}\right)$, or $(T-t)\left(1 - \frac{\mu_{est}}{\mu}\right)$. So we define the under-utilized time $\tilde{T}$ as follows:

**DEFINITION 3: Under-utilized Time**

$\widetilde{T}(n) = \left(1 - \frac{\mu_{est}}{\bar{x}}\right)(T - t) = \frac{d_{(\alpha,n)} \cdot \frac{s}{\sqrt{n}} \sqrt{\frac{N-n}{N-1}}}{\bar{x}} \cdot (T - n \cdot t_s)$ . It determines the average under-utilization time with the safe estimation.

Given all the definition and analysis above, we are ready to determine how much bandwidth testing to perform in order to maximize the data delivered to a user while conforming to the time limit requirement set by the user.

From Definition 3, we assert $\widetilde{T}$ is decreasing as *n* grows and the rate of decrease diminishes (we provide an informal proof in Theorem 1 below) when *n* is relatively small compared to *N*. This means as we take more bandwidth samples, we can estimate the future bandwidth more accurately with the same confidence $\alpha$, thus reducing the cost of time under-utilization. We can denote the benefit of each new sample as $\Delta\widetilde{T} = \widetilde{T}_n - \widetilde{T}_{n-1}$. As we said, $\left|\Delta\widetilde{T}\right|$ decreases as n grows. On the other hand, the cost of each new sample is a constant, $t_s$, which is a time slice. So we can compare $\left|\Delta\widetilde{T}\right|$ with $t_s$ each time a new sample is obtained, the process of bandwidth testing continues until $\left|\Delta\widetilde{T}\right|$ is less than $t_s$. The pseudocode in ALGORITHM 1 below illustrates the method.

**Theorem 1: $\widetilde{T}$ Decrease as *n* grows, and the rate of decrease diminishes when n is relatively small compared to N.**

**Proof**: Since $\widetilde{T} > 0$, the statement is equivalent to $\widetilde{T}(n)' < 0$ and $\widetilde{T}(n)'' > 0$. We write $\widetilde{T}(n)$ as follows,

$$\widetilde{T}(n) = \frac{d_{(\alpha,n)} \cdot \frac{s}{\sqrt{n}} \cdot \sqrt{\frac{N-n}{N-1}}}{\bar{x}} \cdot (T - n \cdot t_s) = \frac{s}{\bar{x} \cdot \sqrt{N-1}} \cdot d_{(\alpha,n)} \cdot \sqrt{\frac{N}{n} - 1} \cdot (T - n \cdot t_s)$$
$$= c \cdot f(n) \cdot g(n) \cdot h(n)$$

(4)

Since $s$ and $\bar{x}$ are random variables determined by the characteristics of the bandwidth population, we view them as part of the constant *c*. We denote $f(n) = d_{(\alpha,n)}$, $g(n) = \sqrt{\frac{N}{n} - 1}$, $h(n) = (T - n \cdot t_s)$.

Thus, we have $g(n)' = -\frac{N}{n^2 \cdot \sqrt{\frac{N}{n} - 1}}$, and $g(n)'' = \frac{N(3N - 4n)}{4 \cdot n^4 \cdot \left(\frac{N}{n} - 1\right)\sqrt{\frac{N}{n} - 1}}$ (5).

We have $g(n)' < 0$, and $g(n)'' > 0$ for $n < \frac{3}{4}N$.

We also have $h(n)' = -t_s < 0$, and $h(n)'' = 0$ (6).

Thus, if we denote $l(n) = g(n) \cdot h(n)$, then

$$l(n)' = g(n)' \cdot h(n) + g(n) \cdot h(n)',$$

$$l(n)'' = g(n)'' \cdot h(n) + 2 \cdot g(n)' \cdot h(n)' + g(n) \cdot h(n)'' \quad (7)$$

Therefore, with (5)(6)(7), we have $l(n)' < 0$, and $l(n)'' > 0$ for $n < \frac{3}{4}N$ .

Finally, when need to show that for $f(n) = d_{(\alpha,n)}$, we have similar results that $f(n)' < 0$, and $f(n)'' > 0$. As *f(n)* is a very complex function, a strict mathematical proof of the property is beyond the scope of this paper. To prove the property numerically, we calculated the values of $d_{(\alpha,n)}$ for $\alpha \in \{0.75, 0.90, 0.95\}$, and *n = 1…500,* and verified that *f(n ) - f(n-1) < 0* and *f(n)-f(n-1)>f(n-1)-f(n-2).* We calculated the values of $d_{(\alpha,n)}$ using the algorithm described in [15] and compared these values with values provided in the t-distribution table in [16] for verification purpose.

Therefore, $\tilde{T}(n)' = f(n)' \cdot l(n) + f(n) \cdot l(n)' < 0$,

$$\tilde{T}(n)'' = f(n)'' \cdot l(n) + 2 \cdot f(n)' \cdot l(n)' + f(n) \cdot l(n)'' > 0$$

for $n < \frac{3}{4} N$.

**ALGORITHM 1: Optimal Bandwidth Testing**

Obtain samples $x_1, x_2, x_3$;

Calculate $\tilde{T}_2$, $\tilde{T}_3$, and $\Delta\tilde{T}_3 = \tilde{T}_3 - \tilde{T}_2$;

n = 3;

while ($|\Delta\tilde{T}_n| > t_s$) {

n = n + 1;

Obtain sample $x_n$;

Calculate $\tilde{T}_n$ and $\Delta\tilde{T}_n = \tilde{T}_n - \tilde{T}_{n-1}$;

}

return $\mu_{est}$;

## III. IMPLEMENTATION AND EXPERIMENTS

We answer the following questions through experiments: (a) How well does the model work in achieving our goal of performing optimal amount of bandwidth testing in bandwidth estimation? (b) How well does the model deliver the final QoS result to end-users? For example, if a user specifies its target transmission time limit associated with a confidence goal (say "meet the time limit by 95% time"), can the model really achieve this level of guarantee? Finally, we are interested in the complexity of the model. Can we implement the model efficiently so that we can make decisions based on them for retrieval applications while estimating bandwidth? In this section, we describe our implementation of the model in Java and then present the experimentation results that answer these questions.

Our implementation consists of two programs, a server and a client. The server is a background process on the server host. It listens to a known port on the server machine and waits for a connection request from a client. The client is a Java Applet embedded in an HTML page. It includes a GUI that allows a user to specify parameters for an object transmission. There are three parameters; "Time Limit" gives the time a user wants to use for object transmission, "Confidence Goal" is the system's guarantee that with this probability the transmission will be finished within the time limit, "Time Slice" is the time used for each bandwidth sample. Figure 1

below is a screen capture of the Client program showing the GUI for user parameter setting.

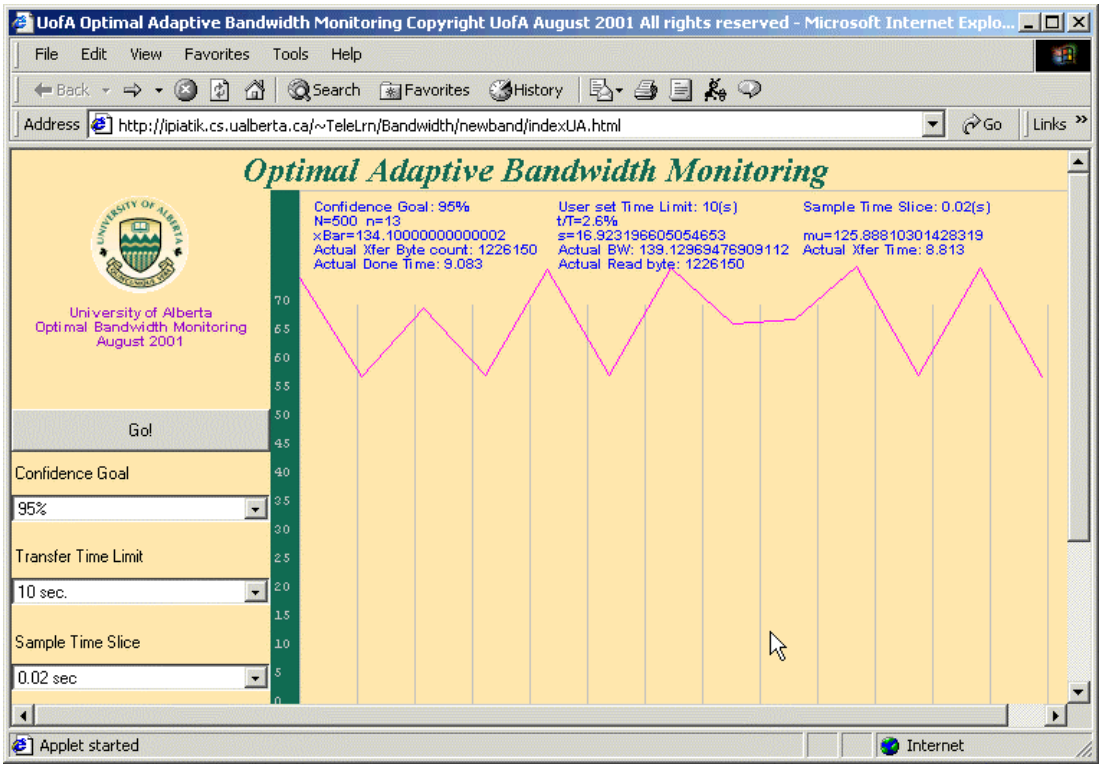


***Figure 1: Screen capture of the Client program***

Once the user sets the parameters and presses a "Go!" button, the following steps are executed:

1. The Client first creates a socket to the server program by connecting to the known port on the server host. It then sends a TEST command to the server. The server responds to the TEST command by sending a stream of random bytes to the client via the connection.

2. The Client receives the testing stream on the other side of the connection and at the same time counts the bytes received. For every time slice, the client divides the number of bytes received by the time slice to determine the bandwidth of this slice. This is a new bandwidth sample $x_n$.

3. The Client calculates $\bar{x}$, $s^2$, $\mu_{est}$, and $\Delta\tilde{T}$ as defined in Section II and compares $\left|\Delta\tilde{T}\right|$ to $t_s$.

If $\left|\Delta\widetilde{T}\right| > t_s$, it repeats Step 2; otherwise it proceeds to Step 4.

4. The Client stops the testing connection, makes a new connection to server and sends a TRANSMIT command to server, along with a target size of the object which equals to $(T-t)\cdot\mu_{est}$.

5. The Server sends a byte stream of length $(T-t)\cdot\mu_{est}$ to the Client.

6. The Client receives the byte stream representing the object, records the finish time, calculates the statistics about this transmission and updates them on the right side region of the applet, as shown in Figure 1.

To minimize the computation of $\widetilde{T}$ when each new sample is obtained, parts of it can be pre-computed and stored in a static array of constants. We organize the component $\frac{d_{(\alpha,n)}}{\sqrt{n}}\cdot\sqrt{\frac{N-n}{N-1}}$ into a two-dimensional array of constants, with $(\alpha,n)$ representing the two dimensions, so that the computation of $\widetilde{T}$ takes only two floating point multiplication and one division given $\bar{x}$ and $s$. Moreover, the computation $s^2$ can be performed using the fast computation form $s^2 = \frac{1}{n-1}\left(\sum_n x_i^2 - n\bar{x}^2\right)$. So, if we maintain the intermediate values $\sum_n x_i$ and $\sum_n x_i^{\,2}$, the computation costs of $\bar{x}$ and $s$ are minimized.

We performed experiments in two typical network environments. The first one is the University of Alberta campus LAN. Both the server and the client runs Red Hat Linux 6 and the client uses Netscape Communicator 4.0 as browser. We present the statistics of our experiments in Table 1 below.

| ***Confidence Goal*** | 95% | 75% |
|---|---|---|
| ***Average No. Of Sample*** | 14.6 | 8.7 |
| ***Sample Average (kbps)*** | 141.1 | 140.5 |
| ***Sample Variance (kbps)*** | 8.07 | 8.65 |
| ***Estimation (kbps)*** | 137.5 | 138.6 |
| ***Actual Average (kbps)*** | 140.9 | 140.7 |
| ***Actual Transfer Time (s)*** | 9.48 | 9.66 |
| ***Actual Total Time (s)*** | 9.79 | 9.85 |
| ***Total No. Of Exception in 100 runs*** | 4 | 20 |

***Table 1: Experiment results on campus network, with Time Limit of 10s, Time Slice of 0.02s. Results are for two different confidence goals.***

The results in Table 1 are based on experiments with parameters "Time Limit" = 10 seconds, "Time Slice" = 0.02 seconds. We give the results for "Confidence Goal" = 95% and 75% ($\alpha = 5$ and $\alpha = 25$). For each parameter set, we perform the transmission 100 times and report the averages.

From the results in Table 1, we can see that when $\alpha = 5$, it takes only 14.6 samples out of 500 (N=10/0.02=500) to achieve the optimal amount of bandwidth testing. Only 4% of the transmissions exceed the time limit and we still utilize 94.8% of the time limit for actual application data transfer. The results for $\alpha = 25$ are similar, except that it takes fewer testing samples and thus leads to a less accurate estimate of bandwidth. This in turn produces more

transmission exceeding the time limit (20%) but utilizes a larger portion of time limit for actual application data transmission (96.6%).

In the second experiment environment setting, we still use the server in University of Alberta, but use a host connected to a local broadband ISP with cable modem as client. A traceroute result shows there are 11 hops between the client and the host. The client runs Microsoft Windows 2000 Professional and Internet Explorer version 5.0. We present the same set of results in Table 2 below.

| ***Confidence Goal*** | 95% | 75% |
|---|---|---|
| ***Average No. Of Sample*** | 21.4 | 15.9 |
| ***Sample Average (kbps)*** | 37.0 | 35.4 |
| ***Sample Variance (kbps)*** | 6.60 | 7.62 |
| ***Estimation (kbps)*** | 34.3 | 34.0 |
| ***Actual Average (kbps)*** | 37.1 | 35.7 |
| ***Actual Transfer Time (s)*** | 8.69 | 9.03 |
| ***Actual Total Time (s)*** | 9.52 | 9.73 |
| ***Total No. Of Exception in 100 runs*** | 7 | 22 |

***Table 2: Experiment results on home network, with time Limit of 10s, Time Slice of 0.02. Results are for two different confidence goals.***

Comparing Table 2 with Table 1, we can see that the variance of the bandwidth is significantly larger in the home network environment. Therefore, the bandwidth testing part of the program is

forced to take more samples and make more conservative estimation on future bandwidth, which in turn reduces the actual data transfer time. Nevertheless, even in this network environment, 87% of time limit is used in application data transmission when $\alpha = 5$.

We now show the effectiveness of the method by comparing the results of our method with a non-adaptive bandwidth testing method, which means the client always takes the same number of samples and use the average of these as the estimate of future bandwidth. Figure 2 shows the results for the campus network environment described above.

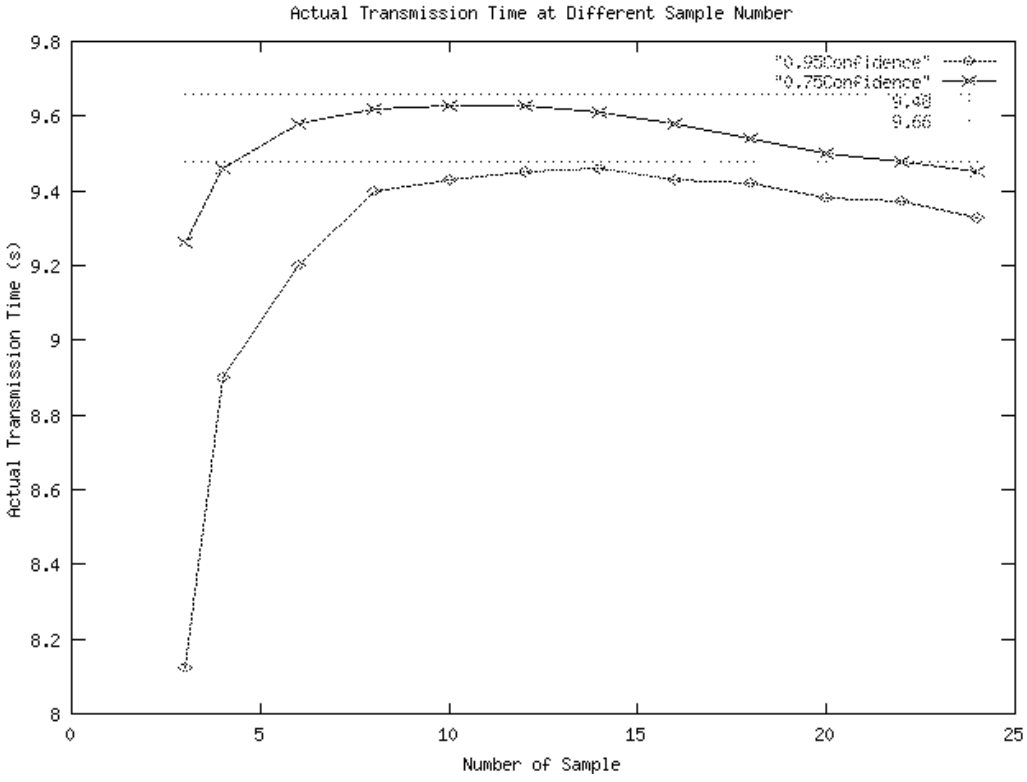


***Figure 2: Actual transmission time with different sample number using non-adaptive bandwidth testing.***

In experiments associated with Figure 2, we use the same parameters as those in Table 1. "Time Limit" is set to 10 seconds; "Time Slice" is 0.02 second; "Confidence Goals" are 95% and 75%. The difference is that this time we do not use the algorithm described in Section II to decide how many testing sample to take, but instead use a constant sample number. We run the transmission for different pre-set sample numbers and plot the actual transmission time (out of time limit 10 seconds) in Figure 2. For easy comparison, we also plot the line y=9.48 and y=9.66 which is the actual transmission time using our algorithm with "Confidence Goal" of 95% and 75% respectively.

From the plot we see that our algorithm accurately arrives at the optimal number of testing samples to maximize the actual transmission time (in turn transmitting the largest possible size of an object) while guaranteeing the time limit requirement.

## IV. RELATED WORKS

[9] presents a framework for construction of network-aware application and gives a real example of such application, the "Chariot" image search engine. The authors concur that good bandwidth estimation is hard to achieve and need further study. [10] presents some bandwidth estimation techniques using transport and network layer metrics. Our research differs from these papers in several aspects. First, the method described in this paper uses only information obtained at the application layer to provide bandwidth estimation, so it does not assume special support from underlying layers in the network stack and is applicable to a wider group of programmers

considering providing adaptive media delivery features. Second, we present the quantitative results showing how effective the bandwidth monitoring work. The previous works let the user select a time limit for object transmission, and makes the "best effort" to delivery such time limit guarantee. Our work introduces the additional parameter "Confidence goal." We let the user select this confidence goal and guarantee the time limit requirements are met by a probability set by the "Confidence goal."

## V. CONCLUSION

In this paper, we proposed a model to determine the optimal amount of bandwidth testing for network-aware multimedia object retrieval applications. The model treats the bandwidth testing as sampling from an actual bandwidth population. It uses a statistical estimation method to determine the estimation confidence interval when each new bandwidth sample is obtained. This confidence interval is then used to determine whether bandwidth testing should continue. We implement the model in a pair of Java Client/Server programs and demonstrate the effectiveness and efficiency of the model. The strategy effectively determines the optimal amount of bandwidth testing to be performed in different network conditions, introduces negligible computation overhead and meets the user specified QoS requirement (time limit) with meaningful guarantee level (the confidence goal). We focus on communication between a pair of client and server in this research; in future work, we plan to extend this model to the application of monitoring multiple servers on one client.

## REFERENCES

[1] A. Vogel, B. Kerherv'e, G. von Bochmann, J. Gecsei, Distributed Multimedia and QoS: a Survey, IEEE Multimedia, Vol.2 No.2, Summer 1995.

[2] P. Ferguson, G. Huston Quality of Service: Delivering QoS on the Internet and in Corporate Networks, Wiley Computer Publishing, 1998.

[3] I. Miloucheva, Quality of Service Research for Distributed Multimedia Applications. 1995 ACM Pacific Workshop on Distributed Multimedia Systems, Hawaii, Honululu, 1995

[4] G. Pacifici, R. Stadler, An Architecture for Performance Management of Multimedia Networks. In Proceeding of IFIP/IEEE International Symposium on Integrated Network Management, Santa Barbara, June 1995

[5] G. Gopalakrishna, G.Parulkar, Efficient Quality of Serverice in Multimedia Computer Operating Systems, Department of Computer Science, Washington University, Technical report, August 1994.

[6] L. Zhang et al. RSVP: A new resource reservation protocol. IEEE Network, 7(5):8-18, September 1993.

[7] R. Braden, L. Zhang et al. Resource ReSerVation Protocol - Version 1 Functional Specification (RFC 2205), September 1997.

[8] D. Ferrari, A. Gupta, and G. Ventr, Distributed advance reservation of real-time connections. ACM/Springer Verlag Journal on Multimedia Systems, 5(3), 1997.

[9] J. Bolliger and T. Gross. A Framework-based Approach to the Development of Network-aware Applications. *IEEE Transactions on Software Engineering,* 25(5):376-390, May 1998.

[10] J. Bolliger, T. Gross et al. Bandwidth modeling for network-aware applications. In Proceedings of Infocomm'99, March 1999.

[11] X. Wang and H. Schulzrinne, Comparison of Adaptive Internet multimedia applications, IEICE Trans. on Communications, vol. E82-B, no. 6, pp. 806-818, June 1999.

[12] I. Cheng, A. Basu, Y. Zhang and S. Tripathi, QoS Specification and Adaptive Bandwidth Monitoring for Multimedia Delivery. In Proc. Of EUROCON 2001, page 485-488, Bratislava, Slovak Republic, July 2001.

[13] D. Harnett. *Statistical Methods, Third Edition.* Addison-Wesley Publishing Company, Inc. 1982

[14] R. Jain, The Air of Computer Systems Performance Analysis. John Wiley & Sons, Inc., 1991.

[15] William H. Press et al, Numerical Recipes in C, second Edition. Cambridge University Press, January 1993.

[16] Daniel Zwillinger, CRC Standard Mathematical Tables, 30th Edition. CRC Press, January 1996.