Performance Evaluation of a Distance Learning Program

D.J.Dailey*, K.R. Eno** and J.F. Brinkley**
Depts. Electrical Engineering* and Biological Structure**
University of Washington, Seattle, WA 98195

ABSTRACT

This paper presents a performance metric which uses a single number to characterize the response time for a non-deterministic client-server application operating over the Internet. When applied to a Macintosh-based distance learning application called the Digital Anatomist Browser, the metric allowed us to observe that "A typical student doing a typical mix of Browser commands on a typical data set will experience the same delay if they use a slow Macintosh on a local network or a fast Macintosh on the other side of the country accessing the data over the Internet." The methodology presented is applicable to other client-server applications that are rapidly appearing on the Internet.

INTRODUCTION

Interest in applications that access information over the Internet is growing, as demonstrated by the many browser-like implementations such as Wais, Internet Relay Chat, and Xmosaic, as well as medical applications in the areas of telemedicine, teleradiology, online access to databases, and distance learning [1]. One of the factors in the usefulness of these network based programs is the response time due to the network distance between the client and the server. This factor becomes especially important when images are transferred, because of the large file sizes.

In this paper we describe a methodology for quantifying network performance delay, and apply the methodology to a network-based application, called the Digital Anatomist Browser, that we have been developing for distance learning in anatomy [2].

During a one year experiment with the National Library of Medicine (NLM), a Browser client was placed at NLM in Bethesda, Maryland, and used to access images and other structural information stored on a server in Seattle. Measurements of the mean response time for various student actions were made and combined with the probability of the actions to produce a metric. This metric expresses the overall performance, at various network distances, as a single number. The performance evaluation methodology, developed for this experiment and presented here, is applicable not only to our own Digital Anatomist Browser, but to other wide-area applications whose operations can only be analyzed nondeterministically.

THE DIGITAL ANATOMIST BROWSER

The Digital Anatomist Browser is one module of our overall Digital Anatomist Framework, which is a client-server architecture, in which various problem-solving programs access an evolving knowledge and database of structural information.[2]

The Browser is currently used as an image-based reference atlas for neuroanatomy. Students pick from a list of subjects, each of which consists of a series of image frames, usually depicting a set of serial sections through an anatomical region of the brain. Associated with each image is a set of contours depicting active areas on the image. When the user clicks on an active area the computer displays the name of the object, as well as any textual descriptions about the object. The Browser can also quiz the student, asking him or her to point on the screen to named objects. All information utilized by the Browser is stored on the server, and is sent over the network to the client, which for this study was a Macintosh program written in Supercard.

The Browser has been used to teach neuroanatomy for several years at the University of Washington. In addition several demonstration versions have been installed at remote sites around the country. The performance results in this paper were obtained using local clients at the University of Washington, and a remote client at the National Library of Medicine in Bethesda, Maryland.

QUANTIFICATION METHODOLOGY

The goal of the performance evaluation methodology is to develop a single number (or metric) that takes into account not only the mean delay time for various possible user actions, but also the probability that a user will choose each action. Since the particular sequence of actions taken by the user cannot be predicted in advance, and since network load will cause delay times that vary from one session to the next, the metric is presented as a probabilistic measurement over multiple sessions.

Our general approach is to estimate mean values for each of the time delay components (T_j) associated with each possible user action and from these estimate a mean total delay (T) which is our performance metric. The mean value is estimated using,

$$T = \sum_{j=1}^{M} p(a_j) T_j \tag{1}$$

where $p(a_j)$ is the probability of action j (of M possible actions), and (T_j) is the mean value for the jth delay.

The probabilities are used as weights to reflect the assumption that the overall delay should not be unduly influenced by actions that are performed only rarely, even if those actions have a long delay. The next sections describe the methodology for estimating the components of this metric.

Probabilities of Actions

The usage pattern of the Browser, in terms of the sequence and frequency of actions taken, will vary by student. Our quantification of the Browser performance is designed to represent some "typical usage." This typical usage is a probabilistic representation of the usage patterns and would not necessarily duplicate any one student's particular experience. Therefore observation of a group of students using the Browser over an extended period is an appropriate way to gather data on the usage patterns. The data set generated by such observations provides the information necessary to construct a statistical description of the actions taken by the students when using the Browser. The particular statistics needed for our performance metric are: (1) the probabilities for the actions taken by the student when using the Browser and (2) the probabilities associated with the likelihood of selecting particular images and anatomical information.

The possible actions are shown in Table 1. The probability of choosing each of these actions is based on the frequency of occurrence of each action. The frequency interpretation of the probability for the occurrence of action a_i of the N available actions each having been observed m_i times is,

$$p(a_i) = m_i / \sum_{i=1}^{N} m_i \tag{2}$$

In particular the mix of the three actions enumerated in Table 1 have probabilities for each action: (1) Choose subject $p(a_1)$, (2) Choose frame $p(a_2)$, and (3) Choose structure $p(a_3)$. These quantify the probability of students taking each of the three actions identified in Table 1 during the course of a typical session.

We approximate the probability of accessing the ith file (having size f_i) of the N files that make up the available Browser images using

$$p(f_i) \approx n_i / \sum_{i=1}^{N} n_i \tag{3}$$

where n_i is the number of times file i has been accessed in our test data set. The same approximation is used for $p(c_i)$, the probability of accessing c_i contours associated with the ith image file.

Mean Delays

Our overall performance metric combines the probabilities developed in the last section with measured quantities. To quantify the performance we need to define several observed rates and delays.

In the case of file transfer across the network, the observables are the size of the file (f_i) and the rate at which data is transferred across the network (r^f) in bytes per second. The mean file size is calculated,

$$\bar{f} = \sum_{i=1}^{N} p(f_i) f_i \tag{4}$$

and the mean rate is established by observing a number of data transfers and calculating the sample mean,

$$\bar{r}_f = \frac{1}{N} \sum_{i=1}^{N} r_i^f . \tag{5}$$

Table 1: Actions and Associated Delays

Action	Description	Associated Variables	Delay Type
a_1	**Choose subject** - This is the selection of the subject area to be considered. This is principally a dynamic type of action that has an observable network delay.	S_c	Dynamic
a_2	**Choose frame** - This is the selection of the specific image to be viewed. This action contains both static and dynamic delays delays associated with:		
	Retrieval of image information.		
	Retrieve image size and filename.	I_s	Dynamic
	Retrieve structure names associated with the contours.	t_n	Dynamic
	Retrieval of image:		
	File transfer	t_f	Dynamic
	Image retrieval overhead	I_o	Static
	Retrieval of contours:		
	Transfer shapes (xy coordinates)	t_s	Dynamic
	Contour retrieval overhead	C_o	Static
	Frame overhead	F_o	Static
a_3	**Choose structure** - This is the selection of individual structures on the anatomical slide presented. A delay is introduced by the retrieval of information about the selected structure.	G_d	Dynamic

These values $\bar{f}$ and $\bar{r}_f$ are combined to get a mean delay time,

$$t_f = \frac{\bar{f}}{\bar{r}^f}. \tag{6}$$

This presumes a linear relationship between the size of the file and the time it takes to transfer the file. Testing over a range of file sizes supports the validity of this hypothesis.

In the case of contours it is the number of contours (c_i) that is the observable to be used in constructing a metric for performance. The mean size for the contours is,

$$\bar{c} = \sum_{i=1}^{N} p(c_i)c_i. \tag{7}$$

The rates for "getting names" (r^n), and "getting shapes" (r^s), are used with the mean contour size to quantify the delays. The mean contour transfer rates are equal to the sample mean of a large number of observations,

$$\hat{r}^n = \frac{1}{N}\sum_{i=1}^{N} r_i^n \qquad \hat{r}^s = \frac{1}{N}\sum_{i=1}^{N} r_i^s. \tag{8}$$

These produce the contour related delays,

$$t_n = \frac{\bar{c}}{\bar{r}^n} \qquad t_s = \frac{\bar{c}}{\bar{r}^s}. \tag{9}$$

Static and Dynamic Delays

We divide the time delay penalty into two major categories: static delays (T^s) and dynamic delays (T^d)

$$T = T^s + T^d. \tag{10}$$

The activities that are heavily dependent on the network performance are assigned to the dynamic category, and those that depend primarily on local CPU performance are assigned to the static category. In reality each dynamic operation must have some static (local CPU) overhead in addition to the network activity but we are assuming that dynamic delays are dominated by the network performance.

Static delays

The static time delay is composed of:

1. Operating system overhead on contour retrieval (C_o) whose average value is obtained in performance tests.
2. Operating system overhead on frame acquisition (F_o) whose average value is obtained in performance tests.
3. Image overhead (I_o) whose average value is obtained in performance tests.

These delays are weighted by the probability of the "choose frame" action that initiates these functions so that,

$$T^s = (C_o + F_o + I_o)p(a_2) \qquad (11)$$

is the static delay metric.

Dynamic delays

The dynamic delay is composed of:

1. The delay due to the retrieval of structure names (t_n).
2. The delay associated with the retrieval of contour outlines (t_s).
3. The delay due to image file transfers (t_f).
4. The delay in retrieving image size and filename (I_s).
5. The time to get descriptive information for the material in the current frame (G_d).
6. The time delay (S_c) introduced by the "choose subject" activity.

The dynamic delays above are weighted with the probability for the action that precipitates the delay and then summed to calculate the overall dynamic delay,

$$\begin{aligned} T^d &= (t_n + t_s + t_f + I_s)p(a_2) \\ &+ S_c p(a_1) + G_d p(a_3). \end{aligned} \qquad (12)$$

The sum of the static delays (equation (11)) and dynamic delays (equation 12)) is a metric for performance. Separating the delay into static and dynamic components allows the relative effect of network delays to be quantified.

RESULTS

To demonstrate the performance metric just presented, we instrumented the Browser, and recorded the time for each of the values in table 1 for three different situations: (1) the Browser operating locally on a Macintosh Quadra, (2) the Browser operating at the NLM using a Macintosh Quadra, and (3) the Browser operating locally using an older Macintosh IIX. These three simple cases allow the trade off between CPU power and network distance to be discussed in a quantitative manner.

At the University of Washington the users were students in a neuroanatomy during one 10 week quarter. Users at NLM were NLM staff who were demonstrating the Browser at the NLM Teaching Learning Center. Because of the difference in user population the usage probabilities were obtained from the UW students and applied to the delays measured both locally and at NLM, in order more realistically simulate expected usage in a class situation at both sites.

The action probabilities are used with the file sizes and contour sizes to produce mean file and contour size values shown in equations (3) and (9). In the test data set there are 5058 accesses of images, each of which retrieved one of the 105 images available.

The numerical values for the static and dynamic delay components of the Browser as developed in the previous sections are shown in table 2. Table 2 presents the static and dynamic time delays for the individual actions unweighted by the probability of that action. The second column of table 2 indicates the action with which the delay is associated. The third column lists the values for the probability of the actions $p(a_i)$ that precipitate the delays. The static (T^S), dynamic (T^D), and overall (T) delay metric for the two sites and two CPU's are shown in bold in table 2.

For two Macintosh Quadra CPU's of essentially the same speed located at different sites on the internet the static delays (T^S) are nearly equal but the dynamic delays (T^D) are vastly different. The static delays on the slower Macintosh IIX computer are a factor of four larger than those on the faster CPU. However, the dynamic delays are also larger for the slower computer at the UW site. This is behavior is expected since it is difficult, in the case of the dynamic delays, to separate the CPU speed effect from the network performance.

DISCUSSION

The value for the overall metric provides a quantitative comparison of the performance of the Browser when operating on different platforms and at different locations. The values from Table 2 suggest that the overall performance on a slower local machine is similar to a faster CPU that uses the Internet to obtain the structural information from the other side of the country. This result demonstrated to us that the local machine configuration is very important in determining the performance delay. For that reason we recently completed a new version of the Browser, written in C rather than Supercard, that is much faster than the version evaluated in this study. We are

Table 2: Delay results

DelayType		a_i	$p(a_i)$	Location/Computer		
				UW/Quadra	NLM/Quadra	UW/Mac IIX
Static Delays						
Contour Overhead	C_o	a_2	0.14	1.42	1.38	6.32
Frame Overhead	F_o	a_2	0.14	1.09	1.18	2.94
Image Overhead	I_o	a_2	0.14	0.46	0.80	5.45
Static Delay	T^S			**0.416**	**0.475**	**2.06**
Dynamic Delays						
Get Descriptions	G_d	a_3	0.83	0.74	1.50	1.69
Choose Subject	S_c	a_1	0.025	0.45	1.07	1.12
File Transfer	t_f	a_2	0.14	1.37	29.81	6.9
Get Names	t_n	a_2	0.14	0.34	0.48	0.48
Get Shapes	t_s	a_2	0.14	1.27	2.94	2.06
Image Setup	I_s	a_2	0.14	0.45	1.21	1.50
Dynamic Delay	T^D			**1.10**	**6.08**	**2.96**
Delay Metric	T			**1.52**	**6.55**	**4.96**

currently collecting performance data on the new version, and expect to see much smaller overall delay because of the reduced static component.

We plan to use the methodology embodied in the Browser metric to quantitatively compare Browser performance at different remote sites. A questionnaire we are developing will ask students to qualitatively rate the acceptability of the perceived delay. Correlations between these qualitative assessments and the measured metric will allow us to determine an acceptable threshold value for Browser performance. We also plan to incorporate a network model, described at last year's SCAMC [3], that will allow us to predict the Browser metric without actually installing the Browser client. Such a methodology should be very helpful in determining the potential acceptability of the Browser at remote sites, and in analyzing the tradeoff between network upgrades and faster local machines.

The same methodology is applicable to other non-deterministic client-server applications that are becoming increasingly prevalent on the Internet. The advantage of this kind of performance evaluation is that it allows the tradeoffs between network and machine upgrades to be analyzed in terms of working applications, rather than more abstract measurements such as network bandwidth.

Acknowledgments

This work was funded by National Library of Medicine Contract N01-LM-1-3506.

References

[1] E. Braun. *The Internet Directory*. Fawcett Columbine, New York, 1 edition, 1994.

[2] J.F. Brinkley, K. Eno, and J.W. Sundsten. Knowledge-based client-server approach to structural information retrieval: the digital anatomist browser. *Computer Methods and Programs in Biomedicine*, 40:131–145, 1993.

[3] D.J. Dailey, K.E. Eno, G.L. Zick, and J.F. Brinkley. A network model for wide area access to structural information. In *17th Symposium on Computer Applications in Medical Care*, pages 497–501. SCAMC, 1993.