This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Predicting User-Perceived Quality Ratings from Streaming Media Data

Amy Csizmar Dalal, David R. Musicant, Jamie Olson, Brandy McMenamy, Sami Benzaid, Ben Kazez, Erica Bolan
Department of Computer Science, Carleton College, Northfield, MN, USA
Email: {adalal,dmusican,olsonja,mcmenamb,benzaids,kazezb,bolane}@carleton.edu

***Abstract*—Media stream quality is highly dependent on underlying network conditions, but identifying scalable, unambiguous metrics to discern the user-perceived quality of a media stream in the face of network congestion is a challenging problem. User-perceived quality can be approximated through the use of carefully chosen application layer metrics, precluding the need to poll users directly. We discuss the use of data mining prediction techniques to analyze application layer metrics to determine user-perceived quality ratings on media streams. We show that several such prediction techniques are able to assign correct (within a small tolerance) quality ratings to streams with a high degree of accuracy. The time it takes to train and tune the predictors and perform the actual prediction are short enough to make such a strategy feasible to be executed in real time and on real computer networks.**

I. Introduction and Motivation

Assessing the quality of a media stream in a large-scale computer network such as the Internet is a difficult problem. While the congestion level on the network connecting a media server and its media clients affects the quality of the stream as seen by the end user, determining how exactly network congestion manifests itself in degraded stream quality is still an unsolved problem. Understanding the relationship between stream quality and network congestion is an important step to solving this problem, and can lead to better design of streaming protocols, computer networks, and content delivery systems.

In this context, we examine how network congestion affects *streaming media*, audio and video data that is sent on demand over a computer network from a single server to one or more users and watched by these users as it is sent out. This type of media traffic is most likely to be visually and audibly affected by network congestion: there is a limited window at the beginning of the stream in which congestion can be dealt with somewhat effectively, and less desirable, more limited means for dealing with network congestion once the stream is in the middle of playback.

We are interested in how network congestion in particular, among the many possible factors that affect stream quality, manifests itself in the *user-perceived* or *subjective quality* of a media stream. User-perceived stream quality indicates how a user would rank the stream relative to other media streams that he or she has seen in the past. Typically, the five-point scale Mean Opinion Score (MOS) [1] is used to collect feedback from end users on the subjective quality of a media stream. The MOS does not scale well to a large number of users, however, and it is somewhat ambiguous in that it does not provide any context for users' ratings.

A more practical approach involves discerning the user-perceived quality of a media stream through objectively-measured quantities, such as packet-level statistics, that can be easily obtained and evaluated from either the underlying computer network or the media player application itself. By taking the measurements as close to the user as possible—at the media player application—we can determine the quality of a media stream in a more scalable and accurate fashion than by polling the user directly. As we have shown previously in [2], we can utilize application layer measurements to substitute for subjective quality ratings because subjective ratings correlate highly with the usual indicators of congestion, such as lost and retransmitted application layer packets. The approach we use—instrumenting a media player application to take application layer measurements—is most similar to the approaches presented in [3]–[5]. The key difference between previous work in this area and our approach is that we utilize these objectively-obtained measurements to *predict* subjective user quality ratings. If we can predict these ratings in a fast and efficient manner, we can potentially employ this analysis in real time and work to mitigate impending network congestion conditions before they affect the user's perceived quality of the media stream.

Exploiting objectively-measured stream data to predict user-perceived stream quality ratings resembles problems that are classic data mining problems, and as such, we consider appropriate algorithms for handling it. Most of the literature on data mining of multimedia data examines a different prediction problem, however: the focus is on mining the *content* of the data, not the stream quality. A notable exception is [6], which discusses the use of random neural networks to predict the quality level of very short (ten second) streams, using aggregated data from the stream. By comparing patterns within the application layer metrics to user quality ratings for that stream, we can understand the effects of network congestion on user perception of stream quality. A simple approach

This work was sponsored by a grant from the Howard Hughes Medical Foundation and by Carleton College. Earlier portions of this work were sponsored by Hewlett-Packard Laboratories.

that we propose entails utilizing statistical information about similar streams, and assigning ratings to streams based on their similarities to past streams in these statistics. A second, more sophisticated approach, exploits the fundamental time-series nature of the streams, and thus uses a time-series similarity metric (called *dynamic time warping* [7]) to determine an appropriate rating for each stream. Both of these approaches assign ratings to a stream based on streams similar to it, and so both of these techniques are examples of *nearest neighbor predictors* [8].

This work presents several key contributions. First, we evaluate several prediction techniques on data obtained from 38 members of a small college community and determine that these prediction techniques assign correct quality ratings, within a small tolerance, to streams with a very high degree of accuracy, typically in the 70-90% range. Second, we demonstrate that there are multiple strategies for selecting the quality rating estimate that work well in this context, and that there are clear strategies that work well in specific circumstances that are directly related to real-world scenarios such as video-on-demand services. Finally, we show that the time it takes to train and tune the predictor and to perform the actual prediction are short enough (in the worst case, proportional to the length of the stream) to make such a strategy feasible to be executed in real time.

The measurement architecture and data collection mechanisms are briefly described in Section II. Section III introduces the strategies that our predictors use, along with the techniques the predictors use to select the quality rating for the stream in question. Section IV explains how we obtained our data sets; the analysis of this data is described in Section V; and future research directions are outlined in Section VI.

II. Measurement Architecture

Collecting measurements at the network level gives us a clear picture of current congestion conditions, either on a global scale or on a particular network segment. However, network-level metrics may not clearly indicate a user's experience with a media stream. In particular, streaming media applications often employ mechanisms such as aggressive retransmissions or temporarily increasing the transmission rate to several times higher than the normal transmission rate to mitigate the effects of network congestion [9] . An alternative is to take measurements from the media player application itself. Doing so allows us to utilize objectively-obtained measurements taken as close to the user as possible, without the inherent scalability and ambiguity problems associated with traditional subjective measurements.

In previous work [2], [10], we have identified a set of application layer metrics that characterize the user-perceived quality of a media stream. Two of these metrics, lost application layer packets and retransmitted application layer packets, form the basis for our analysis here. Lost application layer packets are data packets that either never arrive at the media player application, or arrive after their play-out window has expired. Retransmitted application layer packets are data packets that arrive in time to be rendered by the media player, but do not arrive within their initial delivery window. For simplicity, we refer to these metrics as "lost packets" and "retransmitted packets" throughout the rest of this paper. Lost packets are *instantaneous indicators* of stream quality, while retransmitted packets are *leading indicators* of stream quality. The former appears at the exact moment that the quality of a media stream degrades, while the latter precedes periods of reduced stream quality.

Collecting data at the application layer requires us to poll the media player application for current state information about a stream. In [10], we describe a measurement tool that leverages the existing installed media player software on a user's machine, without requiring the modification of the media player. Our tool consists of a plug-in that interfaces directly to the installed media player on the client—in this case, Windows Media Player. The plug-in uses ActiveX hooks to query the media player at uniform (one second) intervals about the current state of a stream. Polling the player at one second intervals allows us to receive timely information about a media stream without overwhelming the data collection mechanism. The amount of data, for instance, collected for a thirty second stream is about 1200 bytes. The interval duration was chosen via experimentation. The collected data is then logged for later off-line analysis.

Figure 1 is an example of the data collected by our tool for a stream several minutes in duration that experiences a moderate level of network congestion. In this plot, the "received packets" are the data packets that arrive at the media player application within their initial delivery window. The plot shows the number of retransmitted packets increasing well before the number of lost packets increases. This demonstrates that the media player's first response to network congestion is to request that the media server resend missing packets, and that packets are only declared lost once the player has unsuccessfully received these packets.

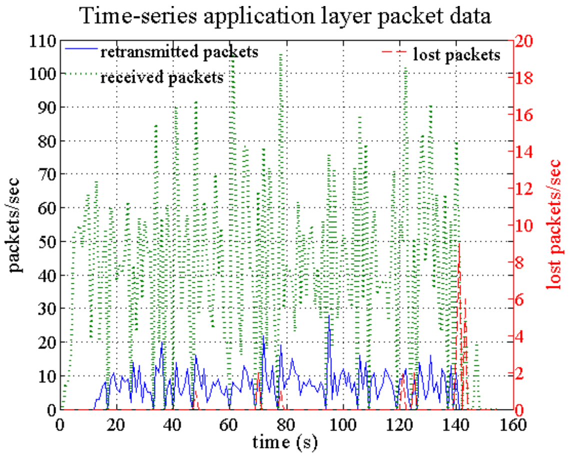

Fig. 1. Time-series packet data collected by the instrumented media player application.

III. Methods for Predicting and Evaluating Stream Quality

A major goal of this study is to determine the user-perceived quality of media streams solely on the basis of state information gathered from a media player application and on knowledge of how similar streams were rated by past users under similar conditions. This is precisely the kind of problem addressed by the field of data mining. Given that data mining as a field focuses heavily on bringing techniques from computer science and statistics together for the purposes of understanding patterns in data, it is only natural to leverage tools from the data mining community for handling this task.

Our particular problem calls for using knowledge of pre-labeled data to predict labels on new data [8], [11], [12]. The goal is to produce a *predictor* by "training," i.e. running a data mining algorithm, on a set of labeled data referred to as a training set. The predictor that is then produced can be tested on unlabeled data. We focus on "nearest neighbor" algorithms [8], [11], [12], which are simple and straightforward. Despite their simplicity, however, nearest neighbor techniques are remarkably effective at finding accurate predictions, and are often used in the data mining community as a barometer of success to which to compare proposed new prediction algorithms.

The idea behind a nearest neighbor predictor is as follows. For an unlabeled example (stream), locate all of the examples in the training set which are the "nearest neighbors" to the unlabeled example, subject to some distance metric. To assign a label to the unlabeled example, a single label is produced from the set of labels for the nearest neighbors.

In our data, the distance metric is computed from one or more measurements collected from the media player. The labels in this case are the user quality ratings assigned to each stream. Our predictors work as follows: given an "unrated" stream, determine this stream's nearest neighbor using only data collected from the media player. Once a nearest neighbor stream has been identified, assign that stream's rating to the unrated stream.

"Nearest neighbor" is technically a family of algorithms, because a number of factors can vary: the distance metric used; the tolerance within which samples are considered to be "nearest neighbors"; and the method of reconciling nearest neighbors with conflicting labels. In this work, we focus on two distinct distance metrics. The first distance metric simply focuses on a *summary statistic* collected from the media player. For the second distance metric, we use the more complex technique of *dynamic time warping* [7].

A. Distance Metric #1: Summary Statistics

One simple approach for measuring similarity between streams is to produce a small set of characteristic numbers that describe each stream, then observe how similar these numbers are between streams. We have demonstrated previously [2] that the percentage of lost packets and the percentage of retransmitted packets provide important clues as to how users rate the quality of a particular stream. Combined, these two metrics give us a close approximation of the amount of packet loss on the underlying computer network. Therefore, for each stream under consideration, we calculate the sum of lost packets and retransmitted packets and divide it by the total number of packets for that stream. We refer to this percentage as the *summary statistic* for that stream. Given an unrated stream with its own summary statistic, then, its nearest neighbor is the stream in the training set with the closest summary statistic. In the case of ties, which are fairly common since we round the summary statistic to the nearest integer, we include all of these streams in the set of nearest neighbors. For example, if our training set consists of streams with summary statistics of $\{10\%, 10\%, 13\%, 25\%, \text{and } 25\%\}$, and the summary statistic of the unrated stream is 11%, then the nearest neighbors are the two streams with summary statistics of 10%. This approach is an example of a *one-nearest neighbor* predictor in which all ties are counted as a single neighbor.

Having now produced a distance metric and a technique for determining which metric is close enough, we determine a single rating (label) for a stream based on the ratings (labels) contributed by the set of nearest neighbors. We consider three reconciliation techniques. The *mean predictor* averages all of the ratings given to streams in the set of nearest neighbors to produce a rating for the unlabeled stream. The *median predictor* does the same thing as the mean predictor, except that it computes the median rating. Finally, the *mode predictor* chooses the most commonly-occurring rating in the set of nearest neighbors. If there are multiple modes, the first occurrence is chosen; if no mode exists, the smallest rating is selected.

B. Distance Metric #2: Dynamic Time Warping

The state information that is gathered from the media player is an example of a *time series*, a collection of observations made sequentially in time [13]. The above summary statistics methodology aggregates data over the entire stream, removing all time characteristics from it. The technique we present here exploits the time-series nature of the media streams.

Dynamic time warping (DTW) is a generalization of Euclidean distance designed for use with time series data. It has been shown to result in a highly effective predictor for time series when coupled with nearest neighbor predictors [7]. DTW is based on the assumption that two time series may be quite similar, even if the precise timing between the two series is misaligned. Instead of using a direct point-to-point (second-to-second) matching as Euclidean distance does, DTW allows time along both series to shift. While DTW aligns the start and end points of each stream, it allows points in mid-stream to align with the closest appropriate point. This fluidity often results in more accurate predictions and pattern identifications. The approach we use involves an extension to DTW [14] that facilitates its use on multidimensional time series, similar to our data. In this experiment, our data set has two dimensions: lost packets and retransmitted packets.

DTW is an appropriate distance metric for use in conjunction with a multimedia stream quality predictor. When presented with a stream of unknown quality, the goal is to compare it to other streams of known quality to find examples of similar behavior. A stream of unknown quality that exhibits packet loss on a periodic basis, for example, is expected to have similar quality to another stream that also loses packets periodically. However, it should not be a requirement for similarity between such streams that the packet losses occur *at precisely identical times*. DTW allows fluidity in comparing two streams so as to match similar behavior between streams that may not occur simultaneously in reality. We do not claim that DTW is theoretically optimal, but we demonstrate later in this paper that it has a positive impact on prediction rates.

One drawback of dynamic time warping is its running time: its complexity is quadratic in the length of the time series. We can reduce the complexity by limiting the warping window, or the amount of "backtracking" allowed between time series. We use the popular Sakoe-Chiba band [7], [15], which limits the distance that one time series can shift relative to the other to a maximum of w time units, where w, the "warping window size", is a user-defined parameter. Additionally, practice has shown [16] that a warping window of appropriate size can improve accuracy in some cases because it can help avoid "pathological warpings," [7] when most of one time series matches up with a very small portion of the other time series.

To integrate DTW into a nearest neighbor predictor, we define the set of nearest neighbors to be the K neighbors which are closest, where K is determined experimentally (see Section IV). Choosing nearest neighbors in this fashion results in a *K-nearest neighbor* predictor [8], [11], [12].

Since DTW is relatively slow to calculate, even with the use of a warping window, we consider the use of optimization techniques for quickly determining a candidate set of nearest neighbors. We therefore use Keogh minimum bounds [7], which provide a fast approximation of the lower bound of the DTW distance between two streams. These minimum bounds are then used to determine for each stream whether or not it is a candidate nearest neighbor. If the Keogh minimum bound for a candidate nearest neighbor is greater than the largest distance for a collection of K streams where the actual DTW distance is already known, the DTW distance for the candidate is not calculated.

Once the set of candidate streams has been selected, the predictor produces a single rating by calculating the mean of the ratings given to the candidate streams. For the rest of this paper, we refer to this predictor as the *DTW predictor*.

IV. Experimental Setup

In this section, we describe the experiments in which we collected our data. The experiments are a larger version, with slight modifications, of those presented in [2]. In addition, we define *hit rate* as the metric by which we evaluate the effectiveness of the predictors under consideration. Finally, we describe the experimental procedure used to select appropriate parameters for the predictors.

A. Measurements

Our data collection testbed consists of a set of 14 client machines on a subnet of a small campus network, and a media server on a separate, isolated subnet. The media server is separated from the rest of the campus network by two routers; the router closest to the server runs NIST Net [17] software. NIST Net was used to apply randomly-distributed packet losses, over the duration of each stream, at the percentages indicated in Table I; there was no additional delay or delay jitter applied to the network by NIST Net. The media server is a 2.4 GHz Pentium processor machine with 512 MB of RAM, running Windows Server 2003 and Windows Media Server 2003 software. The media server sends streams using RTSP over UDP. The NIST Net router is a 700 MHz processor machine with 512 MB of RAM, running Linux kernel 2.4.21-27 and NIST Net version 2.0.12. The client machines have 3.4 GHz Pentium processors and 1 GB of RAM and run Windows XP SP2 and Windows Media Player version 10. Before and during the experiments, we took periodic measurements on the campus network of network packet loss rates, network packet delays, and throughput. Based on these measurements, we found negligible levels of loss and delay on the campus network. Thus, it was not necessary to isolate the client machines in addition to isolating the media server. In addition, we performed a series of tests on the router to verify that it was not inadvertently serving as a bottleneck on our testbed network, and that it could easily support the amount of traffic necessary to send media streams to the set of clients.

Table I lists the streams used in this study and the network congestion levels used in the experiments. The congestion levels, which we determined experimentally, are higher than those typically seen in computer networks, for two reasons. First, we had to overcome the mechanisms that Windows Media Player uses to mitigate the effects of network congestion [9]. Second, we designed the loss levels to affect the media experience in an obvious fashion that influences the streams in the same manner each time. For instance, mild loss is characterized by infrequent short audio and/or visual jitters or freezes; moderate loss causes occasional audio and/or picture freezes, usually one to five seconds in length; and severe loss results in frequent audio and/or visual freezes of one to ten seconds in duration.

We collected stream data and quality assessments using a larger version of the procedure described in [2], in a set of three experiments carried out over a period of two days in May, 2006. In each experiment, we showed a group of college-age (and several slightly older) participants each of three media streams twice, once with no loss and once with either mild, moderate, or severe loss. The participants were not aware of the loss levels shown to them for a particular stream; they also were not told which version of the stream had no loss introduced. The participants rated the audio, video, and overall quality of each stream on a seven-point scale, which allows for slightly finer granularity in participant responses, with one being the worst possible quality and seven being the best possible quality. The measurement tool collected data from each stream simultaneously. We randomized the network congestion patterns introduced to each participant such that at least six participants saw the exact same loss levels for the exact same streams, though not necessarily at the same time. From these experiments, we collected data from a total of 38 participants and their respective client machines.

B. Performance Evaluation

Because user quality ratings are subjective by nature, and because there is significant variation in how particular users rate streams, we measure prediction accuracy by a *hit rate* metric, where hit rate is the percentage of time a prediction falls within 0.8 standard deviations of the user's *normalized* rating for that stream. This corresponds to approximately plus or minus one point on the seven-point scale.[1] A normalized rating is calculated by the formula $r_{\text{norm},s} = \frac{r_s - \bar{r}}{\sigma_r}$, where r_s is the user's quality rating for stream s, $\bar{r}$ is the average of the user's quality ratings on all streams viewed, and σ_r is the standard deviation of the user's quality ratings on all streams viewed [18]. Figure 2 plots the cumulative distribution function of the normalized user quality ratings for all users for overall stream quality, the quality rating (label) used by the predictors. The plot illustrates the distribution of user ratings in terms of how the ratings for streams exposed to the different congestion levels deviate from the users' "average" quality ratings for all streams. The plot shows that our user population was clearly able to distinguish between the different loss levels in the streams, even though they had no knowledge of what loss level the streams they were viewing were experiencing.

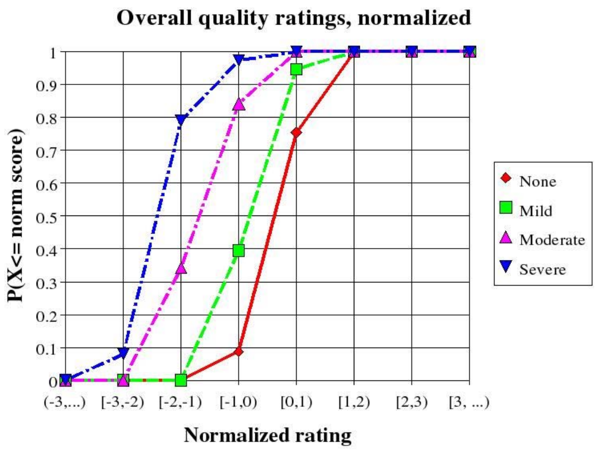

Fig. 2. CDFs of normalized user quality ratings for all viewed streams, broken down by network congestion level.

C. Tuning Predictor Parameters

Preparing predictors to be used on a set of test data is a two step process. The first step, *training*, consists of reading in and storing the data appropriately. In this context, the training phase consists of reading in the state information for the entire stream, and then either storing the lost and retransmitted packets for each second of the stream (the DTW predictor) or calculating and storing the summary statistic (the mean, median, and mode predictors). The second step, *tuning*, consists of selecting the proper parameters for the predictor. For the DTW predictor, the tuning phase consists of setting two parameters: K, the number of neighbors to use for predicting the quality of a stream, and w, the width of the Sakoe-Chiba warping window. The other predictors—mean, median, and mode—are *fixed parameter* predictors, and thus do not undergo a tuning phase. For the rest of this paper, we use "training" to refer to the combined "training and tuning" steps associated with a predictor.

To tune the DTW predictor, we determined optimal values for K and w experimentally via a leave-one-out cross-validation procedure on each training set. In a training set containing n streams, we removed one stream and predicted a rating for it using the remaining $n-1$ training streams as a proxy training set, varying K and w between 1 and 20 and 0 and 30, respectively, in the process.[2] For each value of K and w, we recorded whether or not the stream was rated correctly within a tolerance of 0.8, as discussed in Section IV-B. We repeated this procedure for each stream in the training set and selected the K and w values that yielded the highest hit rate overall. Ties were broken by selecting the K and w values with the smallest execution times, and then by selecting the smallest w value. These K and w values were then used as predictor inputs for the test sets. Since this entire cross-validation process is done purely on the training data, it does not in any way use information from the streams whose quality we wish to predict.

The summary-statistics predictors were all trained in the same manner: For each stream in the training set, the summary statistic, as defined in Section III-A, was computed. The summary statistics and associated stream ratings were compiled into a lookup table. In the testing phase, the prediction consists of a simple lookup in this table.

V. Results

In evaluating the effectiveness of the predictors under consideration, we are first and foremost interested in the hit rate of each predictor, as defined in Section IV-B. However, if we are to utilize these predictors in real time, which is our ultimate goal, then we need to also evaluate the time requirements for each predictor. We discuss each of these performance issues below.

A. Predictor Hit Rates

The first six columns of Table II show the hit rates for the different predictors when assigning overall user quality ratings. The bold values in the table show the highest hit rate achieved for the set of training and testing data among the predictors.

[1]The factors that affect user ratings can vary significantly. Users may be more or less sensitive to encoding differences between streams, or they may view a stream that experiences more apparent loss due to which frames are dropped by the router. Normalizing user ratings helps mitigate these factors, by basing ratings on the biases of the particular user in question.

[2]Note that $w = 0$ is equivalent to traditional Euclidean distance.

TABLE I
DESCRIPTION OF TEST STREAMS AND CONGESTION PATTERNS

Stream information				**Network congestion level (loss)**		
Name	**Time (mm:ss)**	**Description**	$\overline{BW}$ **(kbps)**	**Mild**	**Moderate**	**Severe**
Commercial	0:30	Moderate action	273	5%	15%	25%
Movie trailer	2:22	High action	273	5%	15%	25%
News	4:09	Low/moderate action	331	5%	15%	25%

The values along the diagonal for each predictor indicate the accuracy via cross-validation, as described in Section IV-C. Because the mean predictor produced consistently poor results, with hit rates typically below 60%, we omit its results here.

The cross-validated results show how each predictor would perform when rating an unknown stream of similar length and characteristics, or an unlabeled stream identical to the streams in the training set. This is useful in a scenario such as a video-on-demand system, in which the set of available streams is known completely in advance by the content provider. In such a system, predictors could be trained on each stream in advance, and the appropriate training set selected by the predictor once the user makes his or her selection. In this scenario, the DTW predictor achieves very high hit rates of 88-90%, with the median predictor coming in second with hit rates of 82-83%. The mode predictor does not do as well under cross-validation as the other two predictors. When rating streams that are very similar to the streams in the training set, then, the DTW predictor is clearly the strongest choice.

The results for the off-diagonal experiments are more varied. In these cases, the DTW predictor has the highest hit rate when the shortest stream (commercial) is used as the training stream to predict ratings for the longer streams (trailer and news); the worst performance occurs when either of the longer streams (trailer or news) is used to predict the rating for the shortest stream (commercial). This is an artifact of DTW: we in essence "compact" the longer stream to match the shorter stream. A short stream with the same congestion level as a longer stream will often show very different behavior: for instance, packet losses and retransmissions are concentrated into smaller portions of a shorter stream, whereas in a longer stream the packet losses would be spread out over the duration of the stream. Thus, key cues may be missed by the predictor. Using a shorter training stream, such as the commercial, mitigates this performance issue.

Both the median and mode predictors clearly outperform the DTW predictor when either the movie trailer or news clip is used to predict the ratings for the commercial stream. For the other off-diagonal experiments, results are mixed. However, if we rank each of these predictors from 1-3, with 1 indicating the predictor that yields the highest hit rate and 3 the lowest, for each experiment, we find that the DTW predictor still ranks first the majority of the time, in five of the nine experiments. The median predictor is a good alternate choice: it ranks first in two experiments, and second in five experiments. The mode predictor is the least desirable alternative, since it ranks third in five of the nine experiments and ranks first in only two experiments.

In summary, the DTW predictor is the clear choice under cross-validation and when the shortest stream is used to predict the ratings of the two longer streams. The median is a good alternative to the DTW predictor, particularly in the remaining off-diagonal streams. Because of the mode predictor's variability and low ranking compared to the other two predictors, it is less desirable in this environment.

Table II also shows the K and w parameters that were determined in the training stage of the predictor algorithm, as described in Section IV-C. Note that a window size of ∞ indicates that the best parameter choice for this experiment was to allow the points to match up freely, with no warping window imposed. There was no clear relationship between the choices for K and w and the resulting hit rate for the DTW predictor in the training phase. The hit rate across all values of K and w was highest, in general, for the cross-validated data sets: the range was about 75% to about 88% for all three experiments. To determine the range of hit rates for the non-cross validated data, we ran the DTW predictor on all of the test data for all possible values of K and w, not just the chosen parameter values. In the off-diagonal experiments, the hit rate percentages over all K and w values typically ranged from the low 60s to the low 80s. These results demonstrate that there is a large range of K and w parameter values that would yield acceptably high hit rates when used by the DTW predictor.[3]

B. Predictor Running Time

An important consideration in evaluating the efficacy of these predictors in a real-time environment is the running time of each predictor. The running time of a predictor is composed of its *training time* and its *execution time*. The training time is the time required for the predictor to execute its training phase as described in Section IV-C. The execution time is the time required for the predictor to assign a quality rating to a stream of unknown quality level. Because of the nature of the training phase, we expect the training time to dominate the predictor's running time, particularly for the DTW predictor.

We ran our timing experiments on a 3.0 GHz Pentium 4 machine with 1 GB of RAM, running RedHat Enterprise Linux AS 4 and Java 1.5.0_04-b05. The last four columns of Table II list the training times and execution times for each predictor on each of the training sets and test sets. The training times for each of the summary statistics predictors are identical, since training these predictors entails constructing a table of summary statistics and quality ratings. As expected, training time increases with the duration of the training stream for all

[3]We emphasize that the results presented in Table II use values of K and w chosen without knowledge of the streams used for testing.

TABLE II
USER QUALITY RATING HIT RATES (PERCENTAGE OF PREDICTIONS WITHIN ± 0.8 OF ACTUAL NORMALIZED USER QUALITY RATING), TRAINING TIMES, AND TESTING TIMES FOR EACH PREDICTOR. THE BEST QUALITY RATING HIT RATES FOR EACH TRAINING/TEST SET ARE IN **BOLD**.

		Hit Rate				**Timing**			
		Testing					**Testing time (ns)**		
Predictor	**Training**	**Commercial**	**Trailer**	**News**	**Details**	**Training**	**Commercial**	**Trailer**	**News**
DTW	**Commercial**	**88.2**	**79.0**	**77.6**	$K = 6, w = 1$	24 s	2000	2000	4000
	Trailer	64.5	**89.5**	81.6	$K = 12, w = 3$	3 min 35 s	4000	3000	4000
	News	67.1	76.3	**88.2**	$K = 14, w = \infty$	14 min 45 s	3000	3000	4000
Median	**Commercial**	81.6	75.0	73.7		294 ms	106	359	601
	Trailer	77.6	82.9	**84.2**	n/a	494 ms	80	302	489
	News	**76.3**	71.1	81.6		835 ms	84	297	486
Mode	**Commercial**	68.4	73.7	76.3		294 ms	106	361	609
	Trailer	**78.9**	75.0	80.3	n/a	494 ms	79	304	494
	News	71.1	**80.3**	68.4		835 ms	84	318	507

of the predictors under consideration; also as expected, the training time for the DTW predictor is considerably longer than that of the summary-statistics predictors. However, even in the worst case, the training time for the DTW predictor is under fifteen minutes, or roughly four times the duration of the longest training stream. As we saw earlier, we get highly accurate results with the DTW predictor by training on shorter streams, cutting down the training time for the DTW predictor considerably—in the case of the movie trailer, to about one and a half times the length of the stream. Since the training phase would still occur off-line, a delay of this size is acceptable even in a real-time environment.

The execution time, as measured by our timing experiments, is extremely fast—on the order of hundreds or thousands of nanoseconds. The negligibly small execution times, combined with off-line training times on the order of minutes at worst, mean that all of the predictors described here are potentially good candidates for use in a real-time analysis environment.

By monitoring the user-perceived quality rating in real time, network providers can determine whether quality-of-service (QoS) guarantees are being met for different customers, and adjust network and possibly server resources accordingly, perhaps even mitigating problems before they fully develop. Content providers can utilize such measurements to determine if their content is being seen by the users as intended, and whether their content is being effectively transmitted over computer networks. Finally, content distributors can utilize such information to deploy servers and repeaters in the locations where they are most needed, switch content serving to less-loaded areas of the network, and employ other strategies to ensure that the end users receive media streams at an acceptable quality level. We are currently studying the feasibility of such approaches in real-world computer networks.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have demonstrated that several types of predictors that employ several different distance metrics—summary statistics and dynamic time warping—based on objectively-measured application-layer metrics, are able to predict with a high rate of accuracy the quality rating that a user would assign to a media stream. In addition, we have demonstrated that one particular predictor that employs dynamic time warping as a distance metric is highly accurate in predicting the quality rating assigned to an unlabeled stream when its training set consists of the same stream—which would, for instance, be particularly useful in a video-on-demand system. In other cases, there are several predictors—employing dynamic time warping, and also employing summary statistics and breaking ties by calculating the median of all the nearest neighbor ratings—that can predict user ratings on our data with a high degree of accuracy. This demonstrates the potential for this type of prediction system to be used in a more general streaming environment, in which users may access streams from sources that are not known *a priori*. We have shown on a preliminary basis that all of the predictors studied are potentially viable for use in a real-time prediction environment, because their training times are at worst on the order of minutes and the actual prediction time is on the order of at worst thousands of nanoseconds.

There are several natural extensions to this study that we are currently actively pursuing. Employing these predictors in the real-time analysis of stream data is of paramount interest to us. To do so, a closer analysis of training and execution times of the predictors is necessary. Studying additional refinements to the predictors in general, and to the predictor that employs dynamic time warping in particular, may unearth strategies that the predictor can use to either increase hit rate or decrease training time, or both. For instance, varying the number of streams in the training set may have an effect on the hit rates of the DTW predictor; in particular, perhaps increasing the number of streams in the training set may improve the hit rate of the DTW predictor when using a long stream to predict the quality rating on a much shorter stream. In moving towards realistic real-time prediction of media stream quality, we are studying the prediction of quality ratings from partial stream data. In particular, we are studying how representative a quality rating assigned to an entire stream is of a piece of that stream, and determining if there are portions of a stream that are more representative than others (the middle, for instance, as opposed to closer to the start or end of the stream). Finally, we have demonstrated the viability of this approach using a tool developed for one particular media player application (Windows Media Player). A natural extension to this work would entail expanding this measurement tool to different media player applications, such as Apple's QuickTime player and Real Network's Real Player.

References

[1] ITU-T Recommendation P.910, "Subjective video quality assessment methods for multimedia applications," Recommendations of the ITU, Telecommunications Sector.

[2] A. Csizmar Dalal and K. Purrington, "Discerning user-perceived media stream quality through application-layer measurements," in *Proceedings of the First International Conference on Multimedia Services Access Networks*, Orlando, Florida, June 2005.

[3] Y. Wang and M. Claypool, "RealTracer - tools for measuring the performance of RealVideo on the internet," *Kluwer Multimedia Tools and Applications*, vol. 27, no. 3, December 2005.

[4] D. Loguinov and H. Radha, "Measurement study of low-bitrate Internet video streaming," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, November 2001.

[5] P. Calyam, M. Sridharan, W. Mandrawa, and P. Schopis, "Performance measurement and analysis of H.323 traffic," in *Proceedings of the 2004 Passive and Active Measurement Workshop*, Antibes Juan-les-Pins, France, April 2004.

[6] S. Mohamed and G. Rubino, "A study of real-time packet video quality using random neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1071–1083, 2002.

[7] E. Keogh and C. Ratanamahatana, "Exact indexing of dynamic time warping," in *Proceedings of the 28th International Conference on Very Large Databases*, Hong Kong, China, 2002, pp. 406–417.

[8] M. H. Dunham, *Data Mining: Introductory and Advanced Topics*, Prentice Hall, August 2002.

[9] J. Nichols, M. Claypool, R. Kinicki, and M. Li, "Measurement of the congestion responsiveness of Windows streaming media," in *Proceedings of NOSSDAV*, Kinsdale, Ireland, June 2004.

[10] A. Csizmar Dalal and E. Perry, "A new architecture for measuring and assessing streaming media quality," in *Proceedings of the Workshop on Passive and Active Measurements (PAM 2003)*, La Jolla, CA, April 2003.

[11] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, MIT Press, Cambridge, MA, 2001.

[12] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Addison-Wesley, May 2005.

[13] E. Keogh, "Data mining and machine learning in time series databases," Tutorial presented at KDD 2004, Seattle, WA.

[14] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures," in *KDD '03*, New York, NY, USA, 2003, pp. 216–225, ACM Press.

[15] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans Acoustics Speech Signal Process*, vol. 26, pp. 43–49, 1978.

[16] C. A. Ratanamahatana and E. Keogh, "Everything you know about dynamic time warping is wrong," in *Third Workshop on Mining Temporal and Sequential Data, in conjunction with KDD'04*, Seattle, WA, August 2004.

[17] M. Carson and D. Santay, "NIST Net: a Linux-based network emulation tool," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 111–126, 2003.

[18] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, July 1998, pp. 43–52.