A

User-perceived Quality Assessment of Streaming Media Using Reduced Feature Sets

AMY CSIZMAR DALAL, Carleton College

While subjective measurements are the most natural for assessing the user-perceived quality of a media stream, there are issues with their scalability and their context accuracy. We explore techniques to select application-layer measurements, collected by an instrumented media player, that most accurately predict the subjective quality rating that a user would assign to a stream. We consider three *feature subset selection* techniques, that reduce the number of features (measurements) under consideration to ones most relevant to user-perceived stream quality. Two of the three techniques mathematically consider stream characteristics when selecting measurements, while the third is based on observation. We apply the reduced feature sets to two *nearest-neighbor* algorithms for predicting user-perceived stream quality. Our results demonstrate that there are clear strategies for estimating the quality rating that work well in specific circumstances such as video-on-demand services. The results also demonstrate that neither of the mathematically-based feature subset selection techniques identify a single set of features that is unambiguously influential on user-perceived stream quality, but that ultimately a combination of retransmitted and/or lost application-layer packets is most accurate for predicting stream quality.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Applications (SMTP, FTP, etc.)*; C.4 [**Performance of Systems**]: Measurement techniques; Performance attributes

General Terms: Measurement, Performance, Reliability

Additional Key Words and Phrases: Quality of Service (QoS), Quality of Experience (QoE), video quality assessment (VQA), multimedia applications, multimedia measurement, subjective quality, streaming media

ACM Reference Format:
Csizmar Dalal, A. 2012. User-perceived Quality Assessment of Streaming Media Using Reduced Feature Sets ACM Trans. Internet Technol. V, N, Article A (January YYYY), 32 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Determining how network congestion manifests itself in degraded stream quality, as seen by the end user, can lead to better design of streaming protocols, computer networks, and content delivery systems. Network congestion is caused by a bottleneck somewhere on the path between the sender and the recipient; for instance, due to queueing delays at one of the routers along the path. *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 received, is likely to be visually and audibly affected by network congestion. Within a streaming media application, network congestion is evidenced by reduced available bandwidth, delays and/or loss

This work is supported by grants from the Howard Hughes Medical Foundation, and Carleton College. An earlier version of this work was sponsored by Hewlett-Packard Laboratories.
Author's address: A. Csizmar Dalal, Department of Computer Science, Carleton College.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

of video frames, jitter, and play-out buffer underflows. From the user's perspective, these stream pathologies may appear as skips or pauses in the video stream (in extreme cases, resulting in a "slide-show" like effect), loss of or garbled audio, and player buffering. 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.

As network researchers, we are most interested in the effects of network congestive events on stream quality. While small and isolated congestive events may have minimal impact on stream quality, cumulative events over a period of time will manifest themselves in subjective stream quality. As such, we examine ways to discern the subjective quality effects on streaming media of network pathologies in a scalable, efficient, and effective way.

The factors that affect the user-perceived, or subjective, quality of a video stream are varied and complex. They include such things as the user's personal preference for the video content, the context in which the user views the video (i.e., a work training video versus music videos playing in the background), the user's past experience with video (whether he or she is a heavy or casual consumer of Internet video), and even the amount of motion and scene complexity in the stream.

The five-point scale Mean Opinion Score (MOS) [P.910] is the gold standard in assessing subjective video quality, and can be used to assess streaming video quality as well. Realistically, because collecting MOS ratings requires active user participation, it is difficult to marshall enough participants to make MOS measurements feasible for large user populations, making the MOS less attractive and less accurate for large-scale stream quality assessment. The MOS is also somewhat ambiguous in that it does not provide any context for users' ratings.

It is difficult to control for all of the above factors when assessing subjective video quality. However, determining the subjective quality of streamed video is a crucial component for ensuring that video content is delivered with acceptable quality the majority of the time. In addition, understanding the factors that affect subjective video quality is important in designing new and more effective protocols, delivery mechanisms, streaming servers, and network resources and architectures to support video delivery. As the amount of video streamed online increases at a rapid clip, and as the demand for higher-quality and longer-duration video increases, driven by services such as Hulu [Hulu, LLC] and Netflix [Netflix, Inc.], the ability to support such video through improved protocols, networks, etc. becomes ever more crucial.

To aid in this goal of understanding the factors that affect subjective video quality without the trappings of the MOS, we need to identify stream quality measurement and evaluation mechanisms that are not only more scalable, but can also determine the quality rating quickly and efficiently with a high degree of accuracy. Past and present approaches utilize objective measurements to infer, or predict, the subjective quality rating of a streamed video. Objective measurements are attractive because they are easily obtained, more scalable that the MOS, and easier to analyze and interpret. The challenge lies in identifying appropriate objective measurements to stand in for subjective quality ratings.

We require such measurements to be fast, accurate, and efficiently collected, as close to the user experience as possible. Network measurements, at first glance, are a good candidate because they are easy to interpret and process. Such measurements, however, may not accurately assess the user's experience with a media stream, and indeed may not be collected anywhere near the user population. In addition, 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 [Nichols et al. 2004].

Another approach is to use partial reference [Serral-Gracia et al. 2010] or full-reference [Wolf and Pinson 1999; Ashmawi et al. 2001] prediction of stream quality by comparing part of all of the source stream with part or all of the received stream. Such an approach is highly attractive because we have access to the source stream, and thus the degradations will be clear and can be exactly pinpointed. The drawback, of course, is that we require knowledge about and data from the source stream. This increases the amount of data we have to retain and analyze, and increases the amount of resources that stream quality prediction requires. It may also not be feasible to obtain the source stream—if a video has been very recently uploaded to a service like YouTube [YouTube, LLC], it may take a while for copies to propagate to all of the prediction points in the system.

A more practical approach, and the one we examine here, involves no-reference prediction of the user-perceived quality of a media stream through objectively-measured quantities, such as packet-level statistics, that can be easily and quickly obtained and evaluated from the media player *application*. 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. In the absence of knowledge about the exact nature of recovery techniques used by the media server, media player, and/or encoder, and/or about the types of packets lost (I-frames versus B-frames, for instance), which is the case for proprietary media players and servers, application-layer measurements provide perhaps the only available insight into the occurrence of network pathologies and serve as a first-order approximation of these pathologies. Once these pathologies are identified, further analysis can be completed, if necessary, on their exact causes. If we can predict stream quality ratings in a fast and efficient manner using only these measurements, we can potentially employ this analysis in real time and mitigate impending network congestion conditions before they affect the user's perceived quality of the media stream. If we can do all of this without a reference stream, we can deploy such a system without having to recalibrate every time another video is added to the system, and without control over both the sender and receiver sides.

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. 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 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 distance measure (called *dynamic time warping* [Keogh and Ratanamahatana 2005]) 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* [Dunham 2002].

The predictor algorithms described above take as input measurements collected from an instrumented media player application. We are interested in determining whether an "ideal" set of measurements can be identified, using a systematic identification technique, that yields higher or comparable hit rates than those obtained by our predictor algorithms when using all available measurements. To this end, we employ several *feature subset selection techniques* to reduce the number of measurements, or *features*, utilized by our predictor algorithms. The first feature subset selection technique that we consider chooses features by observation [Csizmar Dalal and Perry 2003; Csizmar Dalal et al. 2007]. The second technique selects the measurements that show

the highest correlations with subjective quality ratings [Csizmar Dalal and Purrington 2005]. The third technique is based on a set of heuristics that mathematically analyze the data using *principal component analysis (PCA)* to determine which measurements most strongly influence the user quality ratings [Yoon et al. 2005].

The main contributions of this work are as follows. Expanding on the work we presented in Csizmar Dalal and Olson [2007], we evaluate three feature subset selection techniques to determine which one selects the application-layer measurements that most accurately reflect user-perceived stream quality as measured using a modified MOS. Further, expanding on the work we presented in Csizmar Dalal et al. [2007], we evaluate two different stream quality prediction algorithms, both of which use nearest-neighbors: a *dynamic time warping* algorithm, and a *summary-statistic* algorithm. We evaluate the accuracy of each prediction algorithm when using the full set of measurements and when using each of the reduced measurement sets identified by the feature subset selection techniques, using data obtained from 38 members of a small college community. Our results demonstrate that there are clear strategies for estimating the quality rating that work well in specific circumstances such as video-on-demand services, but that the strategies differ slightly depending on the nature of the available streams. The results also demonstrate that neither of the mathematically-based feature subset selection techniques identify a single set of features that is unambiguously influential on user-perceived stream quality, but that ultimately a combination of retransmitted and/or lost application-layer packets is most accurate for predicting stream quality, particularly when used as input to the more complex dynamic time warping predictor.

The purpose of this particular paper is to lay out the *foundation* of a subjective stream quality measurement system *framework* that utilizes *only* objective measurements, and to demonstrate that there exist application-layer objective measurements that reflect a given set of network congestion conditions for a particular combination of media player and stream transport protocol. The experiments described in this paper (the ones in which we collect data from an instrumented media player application, as described in Section 6) were designed in such a way as to produce a particular subset of effects in the viewed stream, roughly corresponding to “mild”, “moderate”, or “severe” stream pathologies. As such, our results cannot and should not be viewed as representative of Internet traffic patterns. In the Discussion section (Section 9), we further discuss the benefits and limitations of this approach, and outline a mechanism for testing and verifying our results using a more Internet-realistic testbed.

In the following section (Section 2), we survey prior approaches to the problem of evaluating subjective stream quality. Section 3 makes the case for using application-layer measurements to discern subjective stream quality. In Section 4, we describe the three feature subset selection techniques we consider in this work. Section 5 presents the two nearest-neighbor algorithms we use to predict user-perceived stream quality. The data and experiments are described in Section 6, along with a description of how we applied the feature subset selection techniques and the prediction algorithms to the data. In Section 7, we present the feature sets selected by the various feature subset selection algorithms from our data. Section 8 presents the results of applying the prediction algorithms to both the full and reduced feature sets and identifies the most accurate stream quality predictors from this set. We discuss the implications of our results on the design of content distribution and measurement systems in Section 9, and summarize our results in Section 10.

2. RELATED WORK

A common approach to discerning stream quality is to use an instrumented media player application to collect measurements from the application layer, similar to the

approach we take here. Examples include RealTracer [Wang and Claypool 2005; Wang et al. 2001] and MediaTracker [Li et al. 2002; Nichols et al. 2004; Li et al. 2005] and Keynote's Streaming Perspective software [Keynote Systems, Inc.]. Keynote's measurements are collected from inside the network (at ISPs and peering points), which does not accurately represent the experience of the typical end-user. In addition, Keynote only collects measurements over a portion of the stream, which means that key information about stream pathologies may be missed for longer streams. RealTracer and MediaTracker focus on discerning information about frame rate and jitter, connecting the collected measurements directly to network measurements as opposed to tying them strongly to user quality ratings as we do here.

Another approach entails emulating media streams and/or media player applications as a mechanism to discern stream quality and/or to troubleshoot network issues relating to stream quality. Examples include Ixia's IxChariot software [Ixiacom] and H.323 Beacon [Calyam et al. 2004]. The architecture of both of these tools requires control over a large portion of the network for effective operation, which limits their applicability. In addition, emulating a media stream or media player application may miss certain nuances of actual stream and/or player behavior. By utilizing an existing media player application and actual video streams, we can come closer to capturing the actual user experience.

Other options exist for discerning subjective quality ratings from objective measurements. Some, like Wolf and Pinson [1999] and Ashmawi et al. [2001], correlate measurements on both the sender and receiver sides. Others, such as Clark [2001] and Calyam et al. [2004], utilize passive monitoring tools based on the use of the Emodel [G.107 1998], an objective mechanism for assessing audio quality based on measureable (or estimateable) call transmission parameters to generate an objectively-computed MOS. Both approaches require a level of control over either the sender and receiver (in the former case) and/or the network connecting sender and receiver (in the latter case). Our approach, by contrast, assumes control only over the receiver side, which allows us more freedom in collecting the measurements.

We are most concerned with determining objective measurements that can predict user-perceived stream quality. In previous work [Csizmar Dalal and Purrington 2005; Csizmar Dalal and Olson 2007], we have demonstrated that application packet-level statistics are very influential in predicting subjective stream quality. Similar studies have concentrated on identifying network-layer and link-layer measurements to predict subjective stream quality. Examples include Li et al. [2005], which identifies received signal strength and average link capacity as the most accurate metrics for predicting stream quality, indicated by the stream's average frame rate, over a wireless link; Calyam et al. [2004], which finds that user-perceived stream quality is very sensitive to network loss and jitter, while being fairly impervious to increases in network delays, on a wired network; and Wang and Claypool [2005] and Wang et al. [2001], which also identify jitter as the most influential network-level indicator of user quality ratings. These studies suffer from several drawbacks. First, collecting network measurements presumes some control over at least a portion of the network for probe placement. Such permission may be difficult to obtain, particularly if the measurer is independent of the network operator, and does not ensure that probes can be placed geographically close to the user. This leads to the second drawback: as mentioned previously in this section, network- and link-layer measures can be easily hidden by application-layer mechanisms, such as aggressive retransmissions, and thus may not fully represent what is happening at the application layer. Placing our measurement probes at the application layer, essentially at the interface between the network and the user, allows us to measure exactly what is happening at the application layer, and

allows us to directly relate the observed measurements to user-perceived stream quality.

Most of the literature from the data mining community on predicting stream quality examines the problem of mining the *content* of the multimedia data, not the stream quality. A notable exception is found in Mohamed et al. [2002], 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, with all timing and sequence information removed. We take a similar approach when we examine predictor algorithms that utilize statistical information about streams, as described in Section 5.1, although by contrast we also explore predictor algorithms that retain the timing and sequence information in the streams.

3. INSTRUMENTED MEDIA PLAYER

One strategy for discerning the user-perceived quality of a media stream without relying on subjective quality ratings is to take objective measurements as close to the user as possible. In this context, "close" can mean geographically close or conceptually close. Both goals can be achieved by taking measurements at the user's desktop, from the application itself.

In Csizmar Dalal and Perry [2003], we describe a measurement tool that leverages the existing installed media player software on a user's machine. 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 poll 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.

3.1. Stream State Information

The state information we collect from the instrumented media player application consists of packet-level information, throughput information, and player state information. Packet-level information includes the number of lost, retransmitted, and "correctly-received" packets received by the player each second. Correctly-received packets, referred to as *received packets* throughout this paper, are the application-layer packets that arrive at the media player application within their initial delivery window.[1] Retransmitted packets are the packets that are not received correctly when they are initially sent, but that eventually arrive at the player within their playout time window. Lost packets are the packets that do not arrive at the player within their playout time window.

Throughput measurements collected by the instrumented media player application include the bandwidth, as reported by the player, in kilobits per second; and the frame rate, as reported by the player, in frames per second. Player state information consists of the *buffer starvation state* of the player, where the player does not have enough data in its play-out buffer to render and play out the stream, and instead waits for the play-out buffer to fill so that playback can resume. *Buffer count* records the number of times the player entered buffer starvation mode during stream playback, including the initial startup buffering period.

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. The plot shows

[1]Throughout this paper, we use the term "packets" to refer to "application-layer packets".

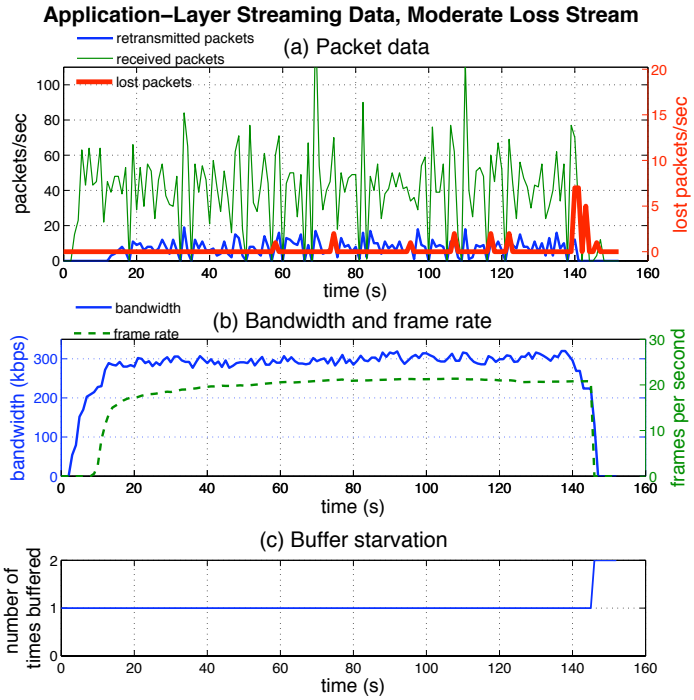

Fig. 1. Time-series data collected by the instrumented media player application. (a) Packet-level data: retransmitted packets and received packets on the left y-axis, lost packets on the right y-axis. (b) Bandwidth and frame rate, on the left y-axis and right y-axis, respectively. (c) Buffer count, including the initial startup buffering period

that most of the reaction to network-level congestion is reflected in the packet-level data, and that the number of retransmitted packets increases well before the number of lost packets. 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 in time to be played out. The buffer starvation event that occurs at the end of the stream, and the spike of lost packets at the end of the stream, are examples of how the effects of network congestion accumulate throughout the stream: the player's response of last resort, if it does not receive the transmitted or retransmitted packets in time, is to report the packets as lost, and if doing so means that the player does not have enough packets to render the current scene, it will report a buffer starvation event at the end of a stream.

Our particular measurement tool provides hooks into a specific media player (Windows Media Player), and as a result our interpretation of the results, as presented in this paper, is specific to the information obtained from that player. For instance, Windows Media Player averages bandwidth and frame rate over some window, rather than reporting the instantaneous bandwidth or frame rate as some other players, like Quicktime [Apple Inc.], do. Thus, in our discussion of bandwidth and frame rate, we assume averaged quantities for these values, and interpret them accordingly. Other results, such as packet information, can be generalized because other players collect similar information to Windows Media Player. Throughout this paper, we will present

our results as they relate to the information we obtain from Windows Media Player, but note, where applicable, where and how our results can be generalized to media players in general.

We can divide the measurements collected by our instrumented media player into *lagging indicators*, *leading indicators*, and *instantaneous indicators* [Csizmar Dalal and Purrington 2005]. **Leading indicators** are the first to change in response to network congestion, and are the precursors of degraded stream quality. In the discussion above, retransmitted packets and received packets are leading indicators; this is true for media players in general. In addition, media players that report instantaneous bandwidth might also classify bandwidth as a leading indicator. **Lagging indicators** change in response to an episode of network congestion, appearing well after the onset of degraded stream quality. Examples of lagging indicators from the discussion above are frame rate and bandwidth, since they are averaged over some window of time. **Instantaneous indicators** change at the exact moment, or as close as possible, at which stream quality degrades. Examples of instantaneous indicators from the discussion above are lost packets and buffer count. In addition, media players that report instantaneous frame rate might also classify frame rate as an instantaneous indicator. Categorizing the measurements in such a way allows us to consider their utility in predicting user quality ratings, the focus of our next section.

3.2. Using Application-Layer Measurements to Predict User Quality Ratings

Streams that have been exposed to similar levels of network congestion will show similar patterns of retransmitted packets, lost packets, etc., even if the exact occurrences and durations do not match exactly. Streams that exhibit these similar characteristics will also exhibit similar user quality ratings, particularly once individual user biases have been accounted for.

One way to discern the similarity between two streams is to compare the application-layer measurements collected at each second from one stream to those collected from another stream, calculating the distance between the two streams. The closer the two streams, the more similar they are. The challenge in this approach, besides identifying an appropriate distance measure, lies in the complexity of the distance calculation. If we are measuring six pieces of state information from a stream, our distance calculation must operate in six dimensions. If timing information is important, as it is in the data under consideration here, then we must repeat this calculation for every measurement at every second. Hence, from a time and resource perspective, it behooves us to determine if we can reduce the dimension of the measurement space without sacrificing accuracy in calculating the distances between streams. From a practical standpoint, this reduces the number of computational resources we require to determine the similarity between two streams.

Our goal is to explore systematic and accurate ways to reduce the number of measurements (here, the six pieces of state information collected by our instrumented media player) that our distance calculation requires. We discuss techniques to address this problem in Section 4. We are interested in determining if measurements from one particular category, such as packet-level measurements or throughput measurements, are more influential to stream quality than measurements from other categories. We are also interested in determining if reducing the number of measurements to, say, one measurement from each category will yield sufficiently accurate stream quality ratings. We utilize this information to predict user quality ratings, as described in Section 5. Finally, we are interested in determining if the reduced set of measurements is constant across streams, or if it is stream-dependent.

4. FEATURE SUBSET SELECTION ALGORITHMS

In this section, we discuss three techniques for feature subset selection. As described in Section 3, each feature is a measurement of one aspect of the state of a stream at a certain time snapshot. All three techniques considered here exploit some aspect of our data in selecting the features that best characterize the data. We first discuss the observation-based feature selection technique from Csizmar Dalal et al. [2007]. We then consider two additional techniques: one in which we utilize the correlations between the application-layer measurements and subjective user quality ratings; and one in which we use principal component analysis (PCA). The feature set selected by any of these techniques will be used as input into the stream quality predictor. Further information on the stream quality predictor can be found in Section 5.

4.1. Observation-Based Feature Subset Selection

In Section 3, we described the application-layer measurements that our instrumented media player application collects about each stream. The leading indicators—retransmitted packets and received packets—determine the probability that stream quality will degrade in the near future, and determine the extent to which stream quality degrades. The instantaneous indicators—lost packets and buffer count—determine exactly which events lead to lower or higher user-perceived stream quality, and indicate the exact moment at which network congestion events are manifested in stream quality degradation. The lagging indicators—bandwidth and frame rate—verify that stream quality degraded at a certain point in the stream. The observation-based technique for feature subset selection exploits this knowledge about lagging, leading, and instantaneous indicators to guide the reduction of the feature set. Since stream quality is impacted by the amount of congestion on the underlying network, it follows that leading and instantaneous indicators that directly reflect network congestion will be good candidate features for our predictor. In particular, a leading indicator foreshadows potential future stream quality degradation, while the instantaneous indicator confirms it. As Figure 1 shows, lost packets and retransmitted packets are accurate second-order and first-order approximations of network congestion, respectively. Thus, under this technique we select lost and retransmitted packets to form our reduced feature set.

4.2. Correlation-Based Feature Subset Selection

Correlation coefficients indicate the strength of the relationship between two variables. Determining the correlation coefficients between each of the application-layer measurements and the stream quality ratings should thus indicate which measurements (features) influence user-perceived stream quality most strongly, as we describe in Csizmar Dalal and Purrington [2005]. The issue with the data here is that we have measurements at multiple time instances per stream, yet only one user rating per stream. To mitigate this, we remove all timing information from the stream data by calculating *summary statistics* for the stream. Specifically, we calculate totals or percentages for the packet measurements, averages for the throughput measurements, and a total for the player state measurement. Pre-calculating the summary statistics makes the correlation coefficients fast and easy to compute. We then take the measurements that have the highest correlation coefficients and use these features as input to our stream quality predictor.

4.3. PCA-Based Feature Subset Selection

The measurements collected from the instrumented media player application forms a *multivariate time series*. We can consider the state information over the length of a sin-

gle stream to be a series of points in n-space, where n is the number of pieces of state information we collect each second. We can preprocess this time series to remove irrelevant and/or redundant features in such a way as to retain the relevant information from the original data. Mathematically, we can reduce the dimension of the n-space to a lower dimension by projecting the original measurements onto a smaller basis, and then selecting the measurements (features) that contribute most strongly to these new basis vectors.

As described in Csizmar Dalal and Olson [2007], we use a family of techniques that performs basis reduction and that is well-suited to multivariate time series data, named CLeVer [Yoon et al. 2005]. The principle behind each of these techniques involves a mathematical procedure called *principal component analysis*, or PCA. The purpose of PCA is to reduce the number of dimensions in each item in a set of data by finding a lower-dimension basis that retains the most relevant information from the original data. In our case, an item is a single stream exposed to a certain level of network congestion that is watched and rated by a single user. PCA is performed separately for each item in the set of data, after which the CLeVer techniques find a set of basis vectors that are common to *all* items in the data set. These basis vectors are called the *descriptive common principal components*. At this stage, we project all of the original data points for all of the items into this common subspace, and then prune the features so that only those that contribute most heavily to the new basis are left.

CLeVer uses three different heuristics to reduce the number of features. *CLeVer-Rank* sorts the features by decreasing magnitude and selects the top features from this ranking. *CLeVer-Cluster* and *CLeVer-Hybrid* both perform k-Means clustering [Dunham 2002] on the basis vectors. CLeVer-Cluster selects the features that are the closest to the cluster centers. CLeVer-Hybrid uses the same ranking mechanism as CLeVer-Rank to rank the features within each cluster, and then selects the highest ranking feature from each cluster.

5. 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 [Hand et al. 2001; Dunham 2002; Tan et al. 2005]. 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 [Hand et al. 2001; Dunham 2002; Tan et al. 2005], which are simple and straightforward. Despite their simplicity, nearest neighbor algorithms 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, locate all of the examples in the training set which are the closest in space to the unlabeled example, subject to some distance measure. 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 measure 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 measure 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 measures [Csizmar Dalal and Olson 2007]. The first distance measure simply focuses on a *summary statistic* collected from the media player. For the second distance measure, we use the more complex technique of *dynamic time warping* [Keogh and Ratanamahatana 2005].

5.1. Distance Measure #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. Each time-series measurement collected by the instrumented media player application can be used to produce one of these characteristic numbers. For instance, we can reduce the time-series measurements of lost packets to the percentage of lost packets over the duration of the stream; we can reduce the instantaneous frame rate to the average frame rate over the duration of the stream; etc. We refer to these numbers as the *summary statistics* for that stream.

Given an unrated stream with its own summary statistic, or set of summary statistics, then, its nearest neighbor is the stream in the training set with the closest summary statistic(s). 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.

Consider a simple example. Suppose our distance measure is computed using the Euclidean distance between two streams using only the lost packets measurement. We first compute the summary statistic for lost packets on every stream in our training set, which is the percentage of packets lost over the duration of the stream. If our training set consists of streams with summary statistics of {10%, 10%, 13%, 25%, 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 is an example of a *one-nearest neighbor* predictor in which all ties are counted as a single neighbor.

Having now produced a distance measure and a technique for determining the nearest stream, 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, each as a separate predictor mechanism. 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* assigns the median rating of the nearest neighbors to the unlabeled stream. 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.

5.2. Distance Measure #2: Dynamic Time Warping

The state information that is gathered from the media player is an example of a time series. 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 [Keogh and Ratanamahatana 2005]. DTW is based on the assumption that two time series may be quite

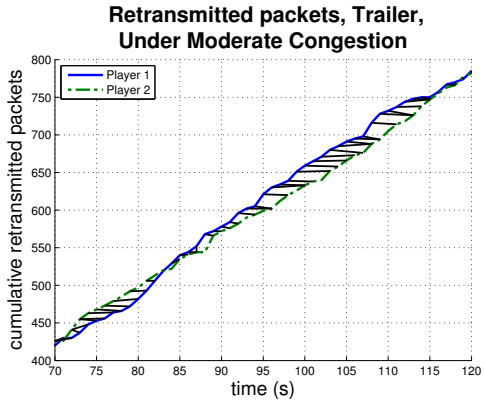

Fig. 2. The cumulative number of retransmitted packets per second for the Trailer stream, measured at two different clients under moderate congestion. The plot is zoomed in to show sufficient detail. The solid black lines connecting the two plots show the point-to-point matching performed by DTW.

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. DTW typically aligns the start and end points of each stream (although this requirement may be relaxed, as Fu et al. [2008] points out), allowing 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 [Vlachos et al. 2003] that facilitates its use on multidimensional time series, similar to our data. Our original data has six dimensions: lost packets, received packets, retransmitted packets, bandwidth, frame rate, and buffer count.

DTW is an appropriate distance measure 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 retransmissions on a periodic basis, for example, is expected to have similar quality to another stream that also retransmits packets periodically. However, it should not be a requirement for similarity between such streams that the packet retransmissions occur *at precisely identical times*, as Figure 2 illustrates. As this figure shows, the shapes of the plots of the cumulative number of retransmitted packets for this stream are very similar, though not identical. The solid black lines connecting the two plots show the point-to-point matching performed by DTW. Note that the "peaks" and "valleys" in packet retransmissions occur slightly later in the stream at the second client than they do in the stream at the first client: the lull at 98 seconds in the first stream is mirrored in the second stream at 102 seconds, for instance. DTW is able to accurately assess which portions of each stream show the most commonalities, and matches them up accordingly. Such matching results in a more accurate characterization for both streams.

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 [Keogh and Ratanamahatana 2005; Sakoe and Chiba 1978], 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 [Ratanamahatana and Keogh 2004] that a warping window of appropriate size can improve accuracy in some cases because it can help avoid "pathological warpings," [Keogh and Ratanamahatana 2005] 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 6.3 for more details on this process). Choosing nearest neighbors in this fashion results in a *K-nearest neighbor* predictor [Hand et al. 2001; Dunham 2002; Tan et al. 2005].

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 use Keogh minimum bounds [Keogh and Ratanamahatana 2005], which provide a fast approximation of the lower bound of the DTW distance between two streams. 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*.

6. EXPERIMENTS

To evaluate our stream quality predictors and our feature subset selection techniques, we collected data in a similar manner to that presented in Csizmar Dalal and Purrington [2005]. We describe the data collection in more detail in this section, as well as explain how we applied the feature subset selection techniques to the data. We also explain how this reduced data was ultimately utilized by the stream quality predictors. Finally, we present two performance evaluation metrics: one to measure the effectiveness of each stream quality predictor (*hit rate*) and one to measure the effectiveness of each feature subset selection technique (*efficacy*).

Figure 3 illustrates the stages that are used to process the data and predict subjective stream quality ratings from the data. These stages are further described throughout this section; we provide a brief overview of the system operation here. The system operates in two stages: *pre-rating* and *rating*. The pre-rating stages, due to their time-consuming nature, execute well in advance of and independently of the actual rating of unlabeled streams. In a production version of this system, the pre-rating stage would occur offline, while the rating stage could occur online, in real time. The pre-rating stage begins with the data collection phase, which is described in Section 6.1. After the data is collected, the state information from all streams is processed using one of the feature subset selection techniques, as described in Section 6.2. The stream quality ratings from the collected data, meanwhile, are sent ahead to the predictor's training phase (Section 6.3); they are also stored to be later used in the testing phase (Section 5). The reduced stream state information, output from the applied feature subset selection technique, is also sent to the training phase and is stored for later use in the testing phase. The training phase outputs a stream quality rating heuristic, or a method of applying a label to an unlabeled stream. This information is also stored for later use in the testing phase. In the rating portion of the system, state information from the stream to be rated is sent to the testing phase, along with the stored reduced stream state information and stream quality heuristic. This data is then used to produce a predicted quality rating for that stream.

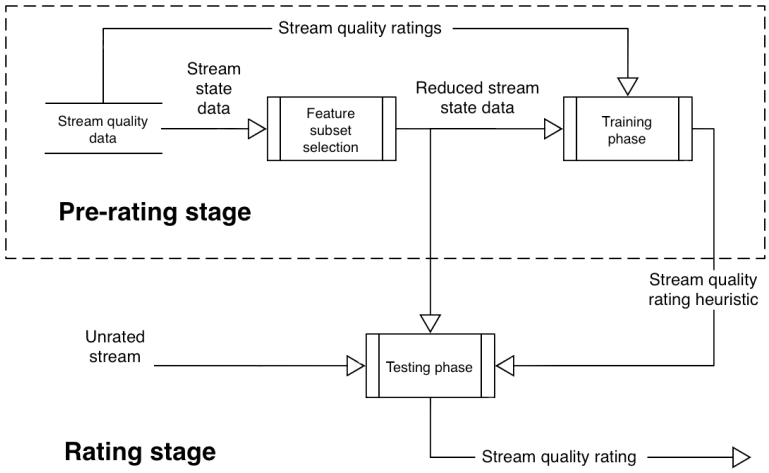

Fig. 3. Lifecycle diagram of the stream quality prediction system.

Because we separate out the training and testing portions of the system architecture, allowing the former to operate off-line and the latter to operate in real-time, the ongoing collection of subjective quality ratings from a user population is not required for the system to operate. Rather, such an architecture allows for *periodic* collection of subjective quality information from a user population. In a closed video content distribution system, for instance, this may be accomplished by allowing volunteer users of the system to press a "Rate Now" button as they are normally viewing content; this rating, along with stream state measurements, can be added to the training database and used to periodically update the stream quality rating heuristic. It is also not necessary for the data from specific users to be present in the training set, since it is only important that the testing phase find the closest match(es) in the training set, and not that it will predict a specific rating by a specific user.

6.1. Experimental Data

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 software [Carson and Santay 2003]. 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 RTP over UDP. We chose UDP as our transport protocol rather than the more ubiquitous TCP so that network congestive effects would be more readily reflected in the application-layer data. We discuss the repercussions of this design choice, as well as indicate how our experimental setup and analysis might change were we to study TCP streams instead, in Section 9. 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

Table I. Description of test streams and congestion patterns.

Stream information				% Network Packet Loss		
Name	**Time (mm:ss)**	**Action Level**	$\overline{BW}$ **(kbps)**	**Mild**	**Mod-erate**	**Severe**
Ad	0:30	Moderate	273	5%	15%	25%
Trailer	2:22	High	273	5%	15%	25%
News	4:09	Moderate	331	5%	15%	25%

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 streams were selected to provide some variety in duration, style, content, and amount of action. The duration of the videos are not atypical of the durations of videos found on sites like YouTube, and are close to the average duration of an Internet video (three minutes) as of October 2008 [comScore, Inc. 2008]. We do not claim that the videos used in these experiments are representative of Internet video in general. Because we required our test subjects to be physically present in the lab during the user experiments, described below, we felt it was important to limit the number of videos and the duration of each video such that the entire experiment could be completed in less than an hour. The content, and amount of action, was selected to be appealing to our chosen demographic of college-age students.

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 [Nichols et al. 2004]. Second, to facilitate evaluating our stream quality prediction system as a proof of concept, we designed the congestion levels to affect the media experience in an obvious fashion that influences the streams in the same manner each time. For instance, mild congestion is characterized by infrequent short audio and/or visual jitters or freezes; moderate congestion causes occasional audio and/or picture freezes, usually one to five seconds in length; and severe congestion results in frequent audio and/or visual freezes of one to ten seconds in duration. In our current work, we are reducing the congestion levels used in our testbed to emulate more realistic network conditions.

We collected stream data and quality assessments using an expanded version of the procedure described in Csizmar Dalal and Purrington [2005], 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 under no network congestion and once with either mild, moderate, or severe network congestion introduced. The participants were not aware of the congestion levels shown to them for a particular stream; they also were not told which version of the stream had no congestion introduced. The participants rated the audio, video, and overall quality of each stream using seven-point scales, which allows for slightly finer granularity in participant responses [Krosnick and Fabrigar 1997; Tang et al. 1999], with one being the worst possible quality and seven being the best possible quality. The end and midpoint scale values were labeled as shown in Table II. 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 congestion levels for the exact same streams,

Table II. Text labels used for the rating scale points, as presented to the participants in our experiments.

Scale Point	Text Label		
	Audio	**Video**	**Overall**
1	completely garbled	unwatchable	poor
2			
3			
4	some degradation	some degradation	fair
5			
6			
7	no degradation	no degradation	excellent

though not necessarily at the same time. From these experiments, we collected data from a total of 38 participants and their respective client machines.

6.2. Applying the Feature Subset Selection Techniques to the Data

Our data set consists of $N = 228$ items. An item is a single stream watched by a single user under a given congestion level for which we have collected both objective measurements and a subjective stream quality rating. Each item contains T observations, where T is the number of seconds in the stream, with $M = 6$ features at each observation, corresponding to the state information collected by the instrumented media player at that time. The feature selection techniques each reduce M.

With the exception of the observation-based feature selection method, feature subset selection is a two step process. In the first step, we run the feature selection technique to determine the features that should be extracted from the data before it is sent to the predictor. This is the *feature selection phase*. In the second stage, the selected features are extracted from the data. This is the *feature reduction phase*.

Observation-based feature subset selection only requires a feature reduction phase, in which we extract the number of lost and retransmitted packets from the stream at each time observation T. This phase yields N Tx2 items to be input into the predictor.

In correlation-based feature subset selection, we reduce M from six to one in the feature selection phase. First, we calculate the summary statistic for each feature in each item, as described in Section 4.2. Second, we group the items by stream, so that the state information from all of the Ad streams, for instance, is included in a single matrix. For each such stream matrix, we calculate the correlation coefficient between the normalized user quality rating assigned to that item and each summary statistic from that item, selecting the feature that yields the highest correlation coefficient for the stream. Then, in the feature reduction phase, we extract the selected feature from each of the original N data items, and calculate its summary statistic. This results in N 1x1 items to be input into the predictor.

Using the CLeVer family of techniques, we reduce M to either two or three in the feature selection phase. We determine the p basis vectors that contribute 80% of the information to the original basis vectors for each item. We then separate the items by stream, as in the correlation-based feature selection method, and calculate the descriptive principle common components separately for each stream, as described in Section 4.3. To these basis vectors, we apply each of the three heuristics (CLeVer-Rank, CLeVer-Cluster, and CLeVer-Hybrid) to determine the top two or three features for each stream. Then, in the feature reduction phase, we extract these features from the original N data items, resulting in N Tx2 or Tx3 items to be input into the predictor.

6.3. Predictor Tuning Phase

The training phase in Figure 3, which prepares predictors to be used on a set of test data, is actually a two step process. The first step, *training*, consists of reading in the state information for the entire stream, and then either storing the chosen features for each second of the stream (the DTW predictor) or calculating and storing the summary statistics corresponding to the chosen features (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 determine the best 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 remove one stream and predict 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] We also test the predictor using no warping window at all ($w = \infty$), meaning that the algorithm is allowed to shift freely along the time axis. For each value of K and w, we record whether or not the stream was rated correctly within a tolerance of 0.8, as we discuss in Section 6.4. We repeat this procedure for each stream in the training set and select the K and w values that yield the highest hit rate overall. Originally, ties were broken by selecting the K and w values with the smallest execution times, and then by selecting the smallest w value [Csizmar Dalal et al. 2007]. However, we discovered that the differences in execution times were negligible between K and w values. Thus, we break ties by selecting the smallest w value, and then by selecting the smallest K value. Choosing a smaller warping window and/or smaller number of neighbors under consideration reduces the amount of computation required by the DTW predictor, which is an important consideration if the predictor is to run in real time. These K and w values are then used as predictor parameters 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 are trained as follows: For each stream in the training set, compute the summary statistic, as defined in Section 5.1. Compile the summary statistics and associated stream ratings into a lookup table. In the testing phase, the prediction consists of a simple lookup in this table.

6.4. Performance Evaluation

In order to evaluate how well our feature subset selection techniques and our stream quality prediction algorithms work, we use two separate performance evaluation metrics.

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 (I-frames versus P-frames, for instance) are dropped by the router. Normalizing user ratings helps mitigate these factors, by basing ratings on the biases of the particular user in question. Here, we use a z-score to normalize ratings: $z_s = \frac{r_s - \bar{r}}{\sigma_r}$, 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

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

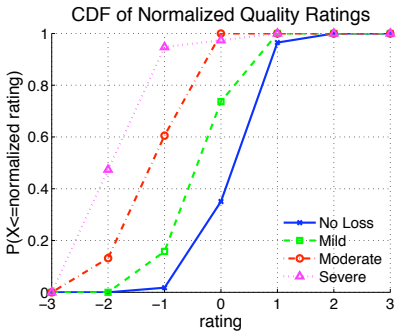

Fig. 4. CDFs of normalized user quality ratings for all viewed streams, broken down by network congestion level.

the standard deviation of the user's quality ratings on all streams viewed. 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 z-score for that stream. This corresponds to approximately plus or minus one point on the raw seven-point scale.

To demonstrate how this performance evaluation metric works on our data, Figure 4 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 congestion levels in the streams, even though they had no knowledge of what congestion level the streams they were viewing were experiencing.

In order to compare the different predictors using different feature sets fairly, we use a single numerical rating, which we call *predictor efficacy*. Given a predictor algorithm (one of DTW, Mean, Median, or Mode) and a possibly reduced feature set, the efficacy is determined by the following equation:

$$\text{efficacy} = \sum_{i=0}^{N} \left(\text{HR}_i - \text{HR}_{\text{ref},i} \right)$$

where i is an item in the dataset, N is the number of items in the dataset, HR_i is the hit rate the predictor obtains for stream i given the input feature set, and $\text{HR}_{\text{ref},i}$ is the hit rate of the *reference predictor*. The reference predictor, identified in Section 8, rates unmodified streams, that is streams that do not undergo feature subset reduction. Briefly, predictor efficacy is a measure of the distance between the hit rates obtained by the reference predictor and those obtained by one of the reduced feature set predictors. A positive efficacy indicates that this particular predictor, when rating streams that have undergone feature subset reduction, is more accurate at predicting stream quality than the reference predictor. The greater the efficacy, the more accurate the predictor.

7. RESULTS: FEATURE SUBSET SELECTION TECHNIQUES

In this section, we present the results of applying the feature subset selection techniques, described in Section 4, to the stream quality data we collected as described in Section 6. Recall that one set of features—lost packets and retransmitted packets—

Table III. Correlation coefficients between the summary statistics of the measurements obtained by the instrumented media player and the normalized user quality ratings. Features that are strongly correlated with the ratings (i.e., a correlation coefficient magnitude above 0.8) are in **bold**.

	Stream		
Feature	**Ad**	**Trailer**	**News**
Lost packets (LP)	-0.5762	-0.7315	-0.7533
Received packets (TP)	0.1264	**0.8402**	0.7453
Retransmitted packets (RP)	-0.1137	**-0.8386**	-0.7622
Bandwidth (BW)	0.2107	-0.7973	-0.7440
Frame rate (FR)	0.6612	0.7592	**0.8145**
Buffer count (BC)	-0.7323	-0.6779	-0.7772

was selected by simple observation. Here, we examine how the other two techniques—correlation and principal component analysis—select features from the set of measurements. We are interested in how much overlap, if any, there is between the two mathematically-based feature subset selection techniques, and whether the features selected by these techniques agree with the ones we selected by simple observation of the data.

7.1. Features Selected by Correlation

Table III lists the correlation coefficients between the stream quality ratings assigned by the study participants and the summary statistics of the measurements collected from our instrumented media player. The abbreviations we use for the features in this table are the same abbreviations that we use in the rest of the tables and figures in this article. We define a correlation to be strong if its magnitude is greater than 0.8. Using this criteria, there is not a single feature or set of features that is strongly correlated with user quality ratings for all three streams. In fact, for the Ad stream none of the features are strongly correlated with user quality ratings. For the Trailer stream, both received and retransmitted packets are strongly correlated with user quality ratings, while for the News stream, only frame rate is strongly correlated with user quality ratings.

The correlations reflect the different strategies that a media server uses to smooth out congestive effects in streams of different lengths. In short streams, it is difficult for the server to mitigate congestion because it simply doesn't have the time to do so. For longer streams, the server has a chance to mitigate some of the congestive effects using a variety of strategies: retransmitting packets, reducing the frame rate, or temporarily increasing the sending rate, particularly at the beginning of the stream. The last of these will increase the average bandwidth over the lifetime of the stream, which explains the negative correlation between bandwidth and stream quality ratings for the Trailer and News streams. When these measurements are then aggregated and/or averaged over the lifetime of a stream, it is less clear which measurements influenced the stream at which point of the stream playback. Hence, in some cases (such as the Trailer stream) we get several measurements that are highly correlated with user-perceived stream quality, while in other cases (such as the Ad stream), we do not have any measurements that are highly correlated with quality ratings.

7.2. Features Selected by Principal Component Analysis

Recall from Section 4.3 that there are three heuristics within the PCA feature selection technique that we use here, CLeVer, that we use to reduce the feature set. For each heuristic, we select the best two and best three features from the set of features, in

Table IV. The features selected by each of the three CLeVer heuristics, and the clusters selected by CLeVer-Cluster and CLeVer-Hybrid.

Algorithm	**Ad**	**Trailer**	**News**
Rank	TP, RP, BW	LP, TP, RP	LP, BW, TP
Cluster (2)	{**TP**, ...} {**LP**}	{**LP**, ...} {**BC**}	{**BW**, ...} {**FR**}
Hybrid (2)	{**FR**, ...} {**LP**}	{**FR**, ...} {**BC**}	{**BC**, ...} {**FR**}
Cluster (3)	{**TP**, ...} {**LP**} {**FR**}	{**LP**, ...} {**FR**} {**BC**}	{**BW**, ...} {**RP**} {**FR**}
Hybrid (3)	{**BC**, ...} {**LP**} {**FR**}	{**BW**, ...} {**FR**} {**BC**}	{**BC**, ...} {**RP**} {**FR**}

part to determine the influence of the number of features input into the predictor on predictor hit rate.

7.2.1. CLeVer-Rank. The first row of Table IV lists the top three features, in order, obtained by running CLeVer-Rank on the data from each stream, as described in Section 6. All three streams are heavily influenced by the packet-level data: at least two of the top ranked features are packet-based measures. Packet-level data is the most immediate indicator of the presence of congestion over time, and in the case of received and retransmitted packets, is the first indicator of the presence of congestion. Lost packets tend to congregate at the end of short streams, but are spread throughout longer streams; this explains why lost packets, the third-strongest correlated measurement for the Ad stream, does not appear in the top three ranked features for Ad here. Of the remaining measurements, bandwidth appears in the top ranked features in two of the streams. While we consider bandwidth to be a lagging indicator of the presence of network congestion for Windows Media Player, it does respond strongly to the presence of congestion, so its appearance here is not surprising.

7.2.2. CLeVer-Cluster and CLeVer-Hybrid. The remaining rows in Table IV list the clusters formed by the CLeVer-Cluster and CLeVer-Hybrid heuristics on our data, along with the feature selected from each cluster by each of these heuristics (in boldface). The clustering portion of each heuristic, rather than evenly distributing the features among the clusters, creates completely unbalanced clusters. We found this to be true regardless of whether we partition the data into two clusters or three clusters. Part of our rationale for choosing a cluster size of three was to determine if the features would cluster by lagging, leading, and instantaneous indicators. This did not occur. In fact, the results show that the clustering portion of the CLeVer heuristics found the features to be more similar than not. Even though CLeVer-Cluster and CLeVer-Hybrid cluster the features identically by design, they never select the same feature from the larger cluster. This demonstrates that the most highly-ranked feature is often not the one at the center of the chosen cluster.

7.3. Summary

In applying various feature subset selection techniques to our data, it was our hope that one or a few features would consistently emerge as having the greatest influence over user quality ratings in our study. Instead, what the results show us is that different feature subset selection techniques interpret the contribution of features quite differently, and often identify multiple candidate feature sets depending on the heuristic chosen. While some features, such as lost packets, appear frequently in the results, there is no set of features that appears the majority of the time among the feature sub-

Table V. User quality rating hit rates (percentage of predictions within ± 0.8 of actual normalized user quality rating) for each predictor using all six possible features as input.

Predictor	Training Stream	Test Stream: Ad	Trailer	News	Params {K, w}
DTW	**Ad**	85.5	59.2	61.8	9, 6
	Trailer	65.8	86.8	82.9	17, 13
	News	65.8	73.7	85.5	7, 2
Mean	**Ad**	57.9	31.6	57.9	
	Trailer	59.2	47.4	57.9	n/a
	News	57.9	39.5	65.8	
Median	**Ad**	60.5	30.3	48.7	
	Trailer	60.5	75.0	60.5	n/a
	News	28.9	31.6	80.3	
Mode	**Ad**	43.4	31.6	60.5	
	Trailer	36.8	40.8	31.6	n/a
	News	30.3	31.6	51.3	

set selection techniques. To determine what, if any, reduced feature sets are adequate for use by our stream quality predictors, we must rely on the measures of predictor hit rates and predictor efficacies.

8. RESULTS: APPLYING SELECTED FEATURES TO STREAM QUALITY PREDICTORS

To identify a reference predictor, we examine the hit rates of each of the stream quality predictors (DTW, Mean, Median, and Mode) when evaluating streams with all six features present, as listed in Table V. Recall that we classify a "hit" as a stream quality prediction that falls within ± 0.8 of the *normalized* quality rating for that stream, which corresponds to approximately ± 1 point on the raw rating scale.

The cross-validated results (the hit rates along the diagonals) show how each predictor would perform when rating an unknown stream of similar length and characteristics to the training stream, or an unrated 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. The off-diagonal results show how each predictor would perform when rating an unknown stream with different length and characteristics than the training stream. This is useful in a scenario such as a content provider that serves up many different types of content, or a content provider whose content rapidly changes and may not have time to run the training algorithm on all possible streams served, such as YouTube.

The DTW predictor's hit rates are the highest of the four predictors for every single combination of training stream and test stream, and consistently predicts the correct quality rating over 59% of the time for all train/test stream scenarios. The three summary statistics predictors are often inaccurate; in particular, the Mode predictor only generates hit rates higher than 50% in two instances (Ad/News and News/News). Of the three summary statistics predictors, the Median predictor generates hit rates over 60% in five of the nine instances, although the range of hit rates is quite extensive, from 29% all the way up to 80%. This is not a huge surprise, as median is a more robust calculation statistically than mean or mode. If we compute the efficacies for the summary statistics predictors as compared to the DTW predictor, we see that they are all worse than the DTW predictor by a substantial amount: -191 for the Median predictor, -192 for the Mean predictor, and -309 for the Mode predictor. By removing all timing information from the stream state data, we smooth out the peaks and valleys in

Table VI. Predictor efficacies using features selected by correlation or by observation. Positive efficacies are listed in **bold**.

Feature(s)	Predictor Efficacy			
	DTW	Median	Mean	Mode
RP	**50.0**	**21.1**	-148.4	-1.2
LP,RP	**44.7**	**21.1**	-148.4	-1.2
LP	**32.9**	-47.2	-182.8	-127.5
BC	-32.9	-2.5	-201.1	-102.4
TP	-32.9	-18.4	-141.9	-265.6
FR	-40.8	-163.1	-178.8	-272.3
BW	-182.9	-250.0	-209.0	-285.4

the stream state information, thus removing valuable information about the stream's behavior in the presence of congestion.

Because the DTW predictor produces the highest hit rate in every scenario, we select it as our reference predictor. In the remainder of this paper, we will compare the hit rates of every predictor evaluating streams with reduced feature sets, to the DTW predictor evaluating streams with all six features present.

8.1. Predictor Efficacies for Features Selected by Observation and by Correlation

Because the correlation technique for reducing the feature set of the stream data failed to identify a single feature or set of features to be used across all streams, we consider all possible features taken one at a time as input into our stream quality predictor algorithms. For space reasons, we combine those results with the results for the features selected by observation. Table VI lists the predictor efficacies for each of the four predictors—DTW, Mean, Median, and Mode—when evaluating streams with a single feature and when evaluating streams with lost and retransmitted packets.

The table shows that only five predictors are more effective than the reference predictor. These are listed in boldface in the table. In all five instances, either lost or retransmitted packets, or a combination of the two, is involved. This matches our initial observation that these two features would be most influential on overall stream quality. For the Mean predictor, and in most cases for the Mode predictor, the choice of features is irrelevant: these predictors are much less effective than the reference predictor.

Interestingly, the DTW predictor's efficacy decreases slightly when lost packets are included with retransmitted packets as opposed to using retransmitted packets alone. Similarly, adding lost packets to retransmitted packets for the Median predictor has no effect on that predictor's efficacy. Clearly, in this case retransmitted packets is the more effective predictor of overall stream quality, when all training and test stream combinations are taken into consideration. Adding lost packets adds noise to the system. We will see later on that this is not universally true: lost packets and retransmitted packets in combination will be a more accurate predictor of stream quality than retransmitted packets alone for some combinations of training and test streams.

8.2. Predictor Efficacies for Features Selected by CLeVer Techniques

8.2.1. Predictor Efficacies for Features Selected by CLeVer-Rank. Table VII lists the predictor efficacies for each predictor when operating on streams with the features selected by CLeVer-Rank as input. Recall that the CLeVer algorithms provide a careful mathematical examination of the stream state data to select the most influential features, so we expect the predictor efficacies to increase. We do not see this in our results. Only the Median predictor, using the reduced feature set of received and retransmitted packets, is more accurate than the reference predictor, but only by the smallest of margins.

Table VII. Predictor efficacies for each predictor using the features selected by CLeVer-Rank. Positive efficacies are listed in **bold**.

Feature	Predictor Efficacy			
	DTW	Median	Mean	Mode
TP,RP	-19.7	**0.2**	-174.8	-274.8
TP,RP,BW	-9.2	-231.6	-182.7	-344.5
LP,TP	-30.3	-18.4	-141.9	-265.6
LP,TP,RP	-17.1	-2.5	-176.1	-270.8
LP,BW	-180.3	-250.0	-209.0	-285.4
LP,TP,BW	-89.5	-243.2	-190.7	-320.9

Table VIII. Predictor efficacies for each predictor using the features selected by CLeVer-Cluster (above the double line) and CLeVer-Hybrid (below the double line). Positive efficacies are listed in **bold**.

Feature	Predictor Efficacy			
	DTW	Median	Mean	Mode
LP,BC	**28.9**	**1.5**	-199.8	-101.1
BW,FR	-189.5	-250.0	-212.9	-289.3
LP,TP,FR	-26.3	**9.3**	-143.3	-111.7
LP,FR,BC	-25.0	-163.1	-178.8	-272.3
RP,BW,FR	-117.1	-253.9	-209.0	-288.0
LP,FR	-38.2	-163.1	-178.8	-272.3
FR,BC	-48.7	-163.1	-178.8	-272.3
RP,FR,BC	**42.1**	-328.7	-209.1	-398.6
BW,FR,BC	-189.5	-250.0	-212.9	-289.3

CLeVer-Rank provides us with a unique opportunity to directly compare the effects of the number of features input into a predictor on predictor hit rate. Every other line in Table VII adds a feature to an existing set of two features. In all three cases here, the DTW predictor generates a higher hit rate when the additional feature is included, although the additional feature does not improve predictor efficacy over the reference predictor. Another way to look at this is that each line adds another feature to a single, highly-influential feature from Table VI: adding transmitted packets to either retransmitted packets or lost packets, and then adding either bandwidth or retransmitted packets to this set. The addition of transmitted packets to either lost or retransmitted packets actually decreases predictor efficacy substantially over using either one of these features alone. Adding a third feature on top of these helps matters a bit, but the overall effect is that the predictor is still less accurate than when using each of these features individually.

8.2.2. Predictor Efficacies for Features Selected by CLeVer-Cluster and CLeVer-Hybrid. Table VIII lists the predictor efficacies for each predictor operating on streams with the reduced feature sets selected by CLeVer-Cluster and CLeVer-Hybrid. Note that the feature set of lost packets, frame rate, and buffer count is selected by both CLeVer-Cluster and CLeVer-Hybrid, so it is only included once in the table results. The table shows that CLeVer-Cluster does slightly better than CLeVer-Rank in selecting "good" feature sets: three predictor/feature set combinations fare better in prediction accuracy than the reference predictor (the entries in bold in the table). CLeVer-Hybrid fares about as well as CLeVer-Rank, only finding one predictor/feature set combination that produces more accurate results than the reference predictor. As seen with the observation and correlation results, all four of the predictor/feature sets with positive efficacies utilize either lost packets or retransmitted packets.

Table IX. The top five scenarios of predictors and reduced feature sets that yield higher hit rates than the reference predictor.

Efficacy	Predictor	Feature Set	Selected By
50.0	DTW	RP	Correlation
44.7	DTW	LP,RP	Observation
42.1	DTW	RP,FR,BC	CLeVer-Hybrid
32.9	DTW	LP	Correlation
28.9	DTW	LP,BC	CLeVer-Cluster

8.3. Examining the Best Predictors

Our analysis identifies ten scenarios in which stream quality predictors, predicting user quality ratings on streams with a reduced number of features, are more effective than the reference predictor. At first glance, this result seems counterintuitive: shouldn't a stream quality predictor that utilizes *more* information about a stream produce more accurate ratings than one using less information? In fact, it is true for the majority of our scenarios that the reference predictor, which does not undergo feature reduction, generates more accurate ratings than predictors that use a reduced feature set. The existence of exceptions to this rule indicate that there is a key collection of features under which the predictor operates quite accurately, but that including additional features actually introduces the equivalent of noise into the system and results in less accurate ratings assignments—a "law of diminishing returns".

The top five of these scenarios are listed in Table IX, in decreasing order of efficacy. The top five most accurate predictors identified in this study are all DTW predictors, and three of these use retransmitted packets as one of the input features. All of the DTW predictors on this list use either retransmitted packets or lost packets, verifying our observation that these two features yield the most information about stream quality. In general, the use of a more complex predictor that retains time-series information yields more accurate stream quality predictions than simpler, summary statistic predictors that remove all time-series information from the data.

It is clear from the results that, with few exceptions and contrary to our expectations, the CLeVer feature subset selection mechanism did no better than simple observation and/or correlation at selecting the best features to input into the stream quality predictor. While the CLeVer algorithms did take stream state characteristics into account, and while they preserved the time-series nature of the measurements, the features deemed redundant may in fact contribute useful information when discerning subjective stream quality. In fact, looking at the feature sets for the top predictors shows that the presence of lost packets and/or retransmitted packets in the feature set is the one constant. The results also demonstrate that adding features to either retransmitted packets or lost packets actually decreases predictor efficacy, when taking all combinations of training and test streams into account. As we mentioned previously, in essence we are adding noise to the system and thus diluting the efficacy of our predictors.

We now take a closer look at the most effective predictors, as listed in Table IX. In particular, we examine their hit rates as compared to the hit rates of the reference predictor, to see exactly where and by how much the reduced feature sets improve upon the reference predictor's hit rates. The hit rates of the top five predictors, as well as the hit rates of the reference predictor, are plotted in Figure 5 for each combination of training and test streams. Table X lists the K and w parameters used by each of the top predictors.

The figure shows that, when training and testing on the same stream, hit rates are high regardless of the choice of features. In fact, even the reference predictor is extraordinarily accurate in this scenario. It is easier to find an exact match in the training set when the test streams are identical to the training streams, so we have

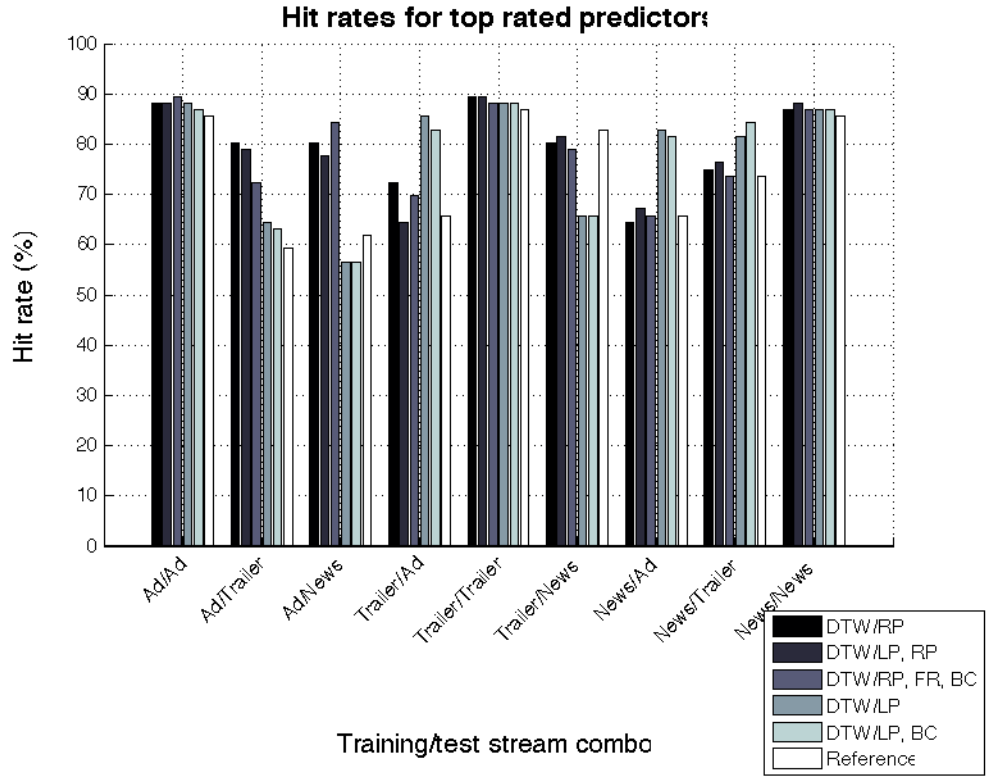

Fig. 5. The top five scenarios of predictors and reduced feature sets that yield higher hit rates than the reference predictor. The figure shows the hit rates for each training and test stream combination

Table X. Parameters (K and w values) used by the top-rated predictors.

Predictor/Features	**Training Stream**		
	Ad	**Trailer**	**News**
DTW/RP	3, 1	6, 0	8, 2
DTW/LP,RP	6, 1	12, 3	14, ∞
DTW/RP,FR,BC	4, 4	8, 0	8, 3
DTW/LP	9, 5	7, 5	5, 10
DTW/LP,BC	7, 4	7, 7	5, 10

more leeway in this situation in terms of choosing appropriate features to yield high predictor hit rates.

The results are more varied for the off-diagonal train/test scenarios. For instance, when the training stream is much longer than the test stream, as in Trailer/Ad and News/Ad, the predictor with the highest hit rate is DTW using lost packets alone. However, if Ad is the training stream and either Trailer or News is the test stream, lost packets is one of the weaker features. Lost packets tend to occur in bursts. In shorter streams, there is less time for negotiation over missing packets between server and client, and often the end of the stream is reached before all missing packets can be retransmitted, leading to bursts clustering at the end of such streams. An artifact of DTW is that we in essence "compact" longer streams to match shorter streams. Such

compacting means it is easier to match up lost packets in a compacted training stream with a short test stream than to do the reverse.

When the training stream is shorter than the test stream, the predictors that utilize retransmitted packets, either alone or in combination with other features, fare better than those who use lost packets alone. As a first-order indicator of network congestion, retransmitted packets tend to be spread out over the course of the stream, making it easier to find matches among streams that are mismatched in length.

Interestingly, the reference predictor fares better than any of the top predictors for the Trailer/News combination, although the predictors utilizing retransmitted packets in some combination of features are a close second. This is the only train/test stream combination that exhibits this tendency, and may just be an artifact of the characteristics of this particular train/test combination.

While Figure 5 shows that there is no single feature that is universally the strongest for all train/test stream combinations, it does show the conditions under which retransmitted packet-based feature sets perform especially well and the conditions under which lost packet-based feature sets perform especially well. In fact, given that retransmitted packets by itself is the top-performing feature for only two of the nine train/test scenarios, it may seem counterintuitive that its efficacy is the highest of the five best feature set combinations. On closer inspection, it is clear that in most of the train/test scenarios, retransmitted packets either generates higher hit rates than the reference predictor or hit rates fairly close to the reference predictor, which is what yields its high overall efficacy. What gives the DTW predictor using retransmitted packets alone the edge over, say lost and retransmitted packets in combination is the train/test set Trailer/Ad: it is ten percent more accurate than the DTW predictor using both lost and retransmitted packets. Thus, retransmitted packets is the most consistent feature set predictor input overall for all train/test combinations.

9. DISCUSSION

The stream quality prediction system we describe in this paper, illustrated in Figure 3, consists of three phases: feature subset selection; an off-line training phase, in which the system learns how to assign stream quality ratings by studying ratings assigned to previously measured streams; and a testing phase, in which the system uses the information learned in the training phase to assign quality ratings to unrated streams, possibly in real time. The results presented in the previous section can be used to inform the design of such a system.

The lack of overlap in the features selected by each feature subset selection technique, and the failure of the more mathematically-based feature selection techniques to ultimately select better feature sets than simple observation, has several implications for the design of stream quality prediction systems. The original intent for applying principle component analysis and correlation coefficient calculations was to provide a more systematic way of narrowing down the measurement space from six dimensions to between one and three dimensions. In reality, our results show that, since the number of dimensions is still relatively small, exhaustive search could discern the best measurements in the feature subset selection phase. Or, as our results have demonstrated, if we have the opportunity to observe stream trends and know a bit about the types of streams the user will most likely encounter, we can use simple observation and reduce the time, processing resources, and storage needs in the feature subset selection phase considerably. The tradeoff, of course, is that more time must be spent pre-processing and observing streams to discern these trends, but if we can identify a family of streams with similar characteristics (length, encoding rate, level of action), we may be able to apply the observations from one stream to this family of streams.

If we design our stream quality predictor for a video on demand system, this is analogous to the scenario in which the streams used in the training and testing phases are identical. In such a system, a stream quality predictor that utilizes dynamic time warping to assign user ratings based on objective stream measurements is most accurate, particularly when the input streams contain information about lost and/or retransmitted packets, potentially in combination with buffer count and/or frame rate.

If we instead design our stream quality predictor for a system in which the available videos were not necessarily the same as the training streams (for instance, a system in which new content was continually uploaded, such as YouTube), the picture changes somewhat. Here, there is no clear choice for all possible combinations of training and test streams, and we have to consider the length of the training and test streams as well when selecting a predictor. In the discussion below, a "short" stream refers to a stream that is less than 40 seconds in length. A "long" stream refers to a stream that is longer than two minutes but less than five minutes in duration. In this particular example, the Ad stream is a short stream, while the Trailer and News streams are long streams.

If our training stream is a short stream, then the ideal predictor is the DTW predictor using input streams that only contain information on retransmitted packets. The hit rate for this predictor in these scenarios is 80%. A good alternative would be the DTW predictor using both lost and retransmitted packets as input; this predictor achieves hit rates of about 78%.

If our training stream is a long stream, then hit rates are also influenced by the duration of the test streams. If the test stream is short, then the best option is the DTW predictor using input streams that only retain information about lost packets. In this scenario, the hit rates achieved by the predictor are above 82%. Another good option is the DTW predictor that uses both lost packets and buffer count as input; in this case, the hit rates achieved are between 81 and 83%. If the test stream is also long, the best predictor is the DTW predictor using both lost and retransmitted packets as input; here, the hit rates are between 76% and 80%. Another good option is the DTW predictor using only retransmitted packets (75% to 82% hit rates). In long streams, retransmitted packets tend to be a better indicator of reduced stream quality than lost packets, because the server and client have more time to attempt packet retransmissions before giving up and reporting a packet as "lost".

Recently, sites like Hulu have appeared on the scene, housing content that is much longer on average (22 minutes) than the average length of Internet video (3 minutes). In sitcom-length and/or feature film-length video streams, the server and client will have more time to negotiate connection parameters during times of congestion and/or respond to periods of congestion. More study is needed to determine if this is true over the entire video or just over smaller portions of such video. We plan on addressing this issue in future work.

As mentioned in the Introduction, we designed the system described in this paper as a framework for inferring stream quality measurements from objectively-obtained data. We developed the framework with the end goal in mind of predicting stream quality from different media players, different transport and application protocols, etc. The framework would accomplish this by swapping variations of the feature subset selection, training, and testing phases in and out depending on the current transport protocol, media player, etc. In the testing of the viability of this system, we necessarily had to make some implementation decisions for the prototype. The three main decisions—using UDP instead of TCP streams, the choice of network packet loss levels on the testbed, collecting data from a single proprietary media player—all unquestionably impact the exact results presented in this paper. Verifying that our proposed framework is more generally applicable in the way we envision requires modifying

these three pieces of the implementation and repeating the experiments presented here. Below are some thoughts as to the implications of these design decisions.

In Section 6.1, we discuss the experimental design choice to use UDP streams in our testbed rather than TCP streams, even though the majority of video streams are transported over TCP in actual networks, at least for residential customers. The experimental setup and data analysis described in this paper could be adapted to instead collect stream quality and stream state measurements from TCP streams. UDP does not adjust its transmission rate based on network congestion conditions; the application protocol handles this task (if at all), which makes network congestion easy to measure at the application layer. TCP adjusts its sliding window in response to packet losses on the network, which means that network congestion will be reflected differently in the application-layer measurements. For instance, a TCP stream will not report any packets as lost or retransmitted when network congestion is present, but it will report a lower number of transmitted packets and reduced bandwidth and frame rate. The feature space for TCP is thus reduced to four: transmitted packets, bandwidth, frame rate, and buffer count. Exhaustive search can be used as a viable feature subset selection mechanism, reducing the complexity of this portion of the pre-rating phase.

Originally, we suspected that a stream quality predictor for TCP streams may require limited information about the source stream: for instance, the expected number of retransmitted packets. Other quantities, such as bandwidth and frame rate, could still be measured solely at the receiver as we do currently. The training phase would require some additional pre-processing of the stream state data in the cases where source stream data is needed, which would occur prior to feature subset selection. On a smaller set of experiments conducted over TCP in July, 2010 [Csizmar Dalal et al. 2010], we found that in fact this was not required, and that we could continue to use no-reference prediction techniques with TCP streams as we did with UDP streams. Beyond this stage the same prediction principles apply: extract the features most relevant to user quality ratings, train the predictor using the reduced stream state information and the user quality ratings, and then use the resulting stream quality rating heuristic to obtain ratings for unrated streams. The prediction algorithms themselves will remain largely unchanged: only the selected feature inputs will change. In our recent TCP experiments, for instance, bandwidth and frame rate were the strongest predictors of stream quality ratings, and acted more like leading predictors than lagging and instantaneous predictors, respectively.

Because of the different nature of TCP and UDP streams, and the different congestive recovery mechanisms they employ, TCP and UDP streams will react differently to the same levels of congestion within a network. The second piece for verifying the generalizability of our framework entails modifying the testbed parameters to not only be more realistic in terms of typical Internet congestion, but also to determine how exactly UDP and TCP streams react in the presence of congestion. UDP is much better suited to video traffic by design, so it is likely that TCP streams will show signs of degradation at lower network packet loss levels. Our recent TCP experiments indicate that “severe” quality degradations manifest at 10-15% packet loss, and sometimes at even lower levels of packet loss. Additionally, introducing some delay into the system is required to more closely model Internet conditions—delays alone may also introduce slightly different manifestations of quality degradations.

The system proposed in this paper obtains measurements from a particular media player (Windows Media Player), and as discussed in Section 3, to some extent our results are influenced both by the type of measurements that this media player makes available via its API and by the format in which these measurements are presented (average bandwidth versus instantaneous bandwidth, for instance). We can easily ex-

pand the proposed measurement system to handle data from a variety of media player types. The first step in this process is to develop instrumented versions of other media players, as we did for Windows Media Player. The second step is to collect a sufficient amount of measurements from these instrumented players to form player-specific training sets. Once these two pieces are in place, and after making minor changes to our predictor algorithms to handle the data formats from these different players, our system could detect the type of player from which data was arriving, and employ the appropriate version of the predictor algorithm using the appropriate training set, in the same manner explained in this paper. Preliminary results from our recent work implementing a Flash player [Adobe Systems, Inc.] plugin [French et al. 2011] demonstrate that this is indeed possible, with minimal changes to the existing framework.

Finally, the question remains as to the applications of such a system. Given that we can detect stream quality degradation from application-layer measurements, how would we utilize such information in a real-world system? Ideally, such information would be fed back to interested parties along the critical path between media server and media client, and the parties could use this information to mitigate or lessen the congestive effects. This of course would be easier in a more tightly-controlled environment, such as an educational institution or corporation, where it is easier to make architectural changes and where the user population is smaller. For instance, within an organization, a local access point might serve the content from a local store temporarily until network conditions improve. A content provider might switch the user over to another, less-congested media server, or even play a commercial or two until the problem subsides. In the worst-case scenario, the organizational access point or the external content provider may tell the user to come back later, so as to avoid a poor experience by the end-user. Even within a controlled environment, these applications pose some potential scalability and privacy problems. In particular, keeping track of individual users becomes less scalable as the number of users increases. A more scalable solution might entail monitoring stream quality for a sample of representative users and making system-wide decisions, such as switching a subset of users over to a local store, rather than individual decisions. Each of these scenarios poses its own design and architectural challenges, as we discuss further in Csizmar Dalal [2009].

10. CONCLUSIONS

Practical, scalable assessment of user-perceived quality of media streams is possible using application-layer measurements and by exploiting the fact that similar streams will show similar trends in the state information collected from each stream. The similarity of two streams is the distance in n-space between their state measurements. Since distance calculations increase in complexity with the number of dimensions, we employ three different feature subset selection techniques to reduce the number of measurements collected by an instrumented streaming media application used as input to a stream quality predictor algorithm. The three feature subset selection techniques were chosen to compare observational feature subset selection techniques (selecting features based on knowledge of how network congestion is reflected in application-layer measurements) with more mathematical feature subset selection techniques (correlation and principal component analysis), to determine which type yields feature sets that are more reflective of user-perceived stream quality. We present two sets of prediction algorithms which use different forms of statistical matching to assign ratings to unrated streams: a K-nearest-neighbor predictor using dynamic time warping as its distance measure, and one-nearest-neighbor predictors that use summary statistics as their distance measures, with different strategies for dealing with ties. Using data collected in May 2006 from 38 college students, each of whom rated six streams under a variety of congestion scenarios, to yield a total of 228 rated streams

(with stream state measurements), we evaluated the accuracy of the stream quality prediction algorithms using all available measurements and using the reduced measurement sets, output from the feature subset selection techniques.

Our results from our proof-of-concept experiments demonstrate that neither of the mathematically-based feature subset selection techniques identify a single set of features that is unambiguously influential on user-perceived stream quality. In the absence of a predictor algorithm, given just the objective measurements and the subsequent subjective stream quality measurement, all of our feature subset selection techniques found elements of each objective measurement that influenced the overall observed stream quality. It is only when these features are ultimately used as input into a more complex predictor algorithm, with knowledge across multiple streams, that the differences in the influences of various features become apparent.

Our results also demonstrate that the most accurate predictor in our proof-of-concept experiments, in terms of predicting user-perceived stream quality within a tolerance of the actual stream quality rating, is the DTW predictor utilizing some combination of lost and retransmitted packets from the stream data. The specific features chosen have a much more powerful impact on predictor hit rates than the number of features used as input.

The implications for the design of a stream quality prediction system are as follows. If we are to design such a system to support video-on-demand, our best option is to use a K-nearest neighbor predictor that uses DTW as its distance measure and whose input streams contain information about lost and/or retransmitted packets. For systems in which the content served varies or changes quickly in time or does not resemble the training streams, there are several candidate predictors. If the training stream is short, a K-nearest neighbor predictor that uses DTW as its distance measure and retransmitted packets as stream input is the best option. Otherwise, a K-nearest neighbor predictor using DTW as its distance measure and either lost packets or lost and retransmitted packets is the best option. Regardless of the type of content provider for which we are designing this system, our results demonstrate that using simple techniques for reducing the number of features to be input into the predictor is our best option, which is beneficial because using simple techniques reduces the time and the resources required for feature subset selection (and preparing the data for prediction).

A limitation of this particular proof-of-concept study is that our testbed network utilizes network settings that do not reflect realistic levels of congestion for the Internet on a typical day. As we discuss in Section 6.1, we made this particular trade-off so that we would ensure a particular and consistent "look" to the streamed video during our user experiments. We are currently reworking our experiments using more realistic settings in our testbed network.

There are several areas that we are interested in studying in the future, and/or are currently expanding upon. First, we have chosen to limit the number of videos in our original study to three, for the reasons mentioned in Section 6. In follow-up experiments, we are expanding the set of videos to include a broader representation of video content, as well as include streams with more variety in the amount of motion, content, and resolution. Second, we are modifying our experimental design, so that our testbed network introduces congestion in a manner closer to what happens on the Internet. For instance, our current setup does not emulate packet loss burstiness, queueing delays, and contending traffic, which may adversely affect measurements such as buffer count and frame rate. Third, our prediction mechanism considers stream-level packet measurements without considering the context of these measurements, i.e. the location of packet losses within the stream as compared to the portions of high or low motion. Theoretically, packet losses will have more of a detrimental effect if they appear at "crucial" points in the stream, corresponding roughly to the location of I-frames. While our

media format is proprietary, meaning we can glean limited information about stream encoding (such as the location of I-frames in the stream), we can, and are, taking a closer look at where in the stream packet losses occur, and determining the resulting effect on stream quality. Fourth, we have restricted our analysis in these experiments to predicting overall stream quality, even though we have collected ratings for audio and video quality in addition to the overall ratings. We have preliminary results examining which of the two ratings, audio or video, has a bigger influence on the user's overall quality rating, and whether we can obtain more accurate predictions using either the audio or the video rating instead of the overall quality rating; we continue work in this area as well. Fifth, we are studying the effect that reducing the epsilon, and thus attempting to predict a rating closer to the actual z-score for each stream, has on the stream prediction accuracy. Finally, we are studying how to apply our findings to the design of a real-world stream quality prediction system. In particular, we are examining the feasibility of performing stream prediction in real time, after an off-line training phase for the predictor.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

The author would like to thank Sami Benzaid, Erica Bolan, Ben Kazez, Brandy McMenamy, Dave Musicant, Jamie Olson, Keith Purrington, and Anna Sallstrom, who contributed to earlier phases of this research. The author would also like to thank Mike Tie for his technical assistance in setting up and maintaining the testbed network. Finally, the author would like to thank the anonymous reviewers for their constructive feedback on earlier versions of this article.

REFERENCES

ADOBE SYSTEMS, INC. Adobe Flash software. http://www.adobe.com/products/flash/.

APPLE INC. QuickTime Player. http://www.apple.com/quicktime/.

ASHMAWI, W., GUERIN, R., WOLF, S., AND PINSON, M. H. 2001. On the impact of policing and rate guarantees in Diff-Serv networks: A video streaming application perspective. In *Proc SIGCOMM*. San Diego, CA.

CALYAM, P., SRIDHARAN, M., MANDRAWA, W., AND SCHOPIS, P. 2004. Performance measurement and analysis of H.323 traffic. In *Proc. PAM 2004*. Antibes Juan-les-Pins, France.

CARSON, M. AND SANTAY, D. 2003. NIST Net: a Linux-based network emulation tool. *SIGCOMM Comput. Commun. Rev. 33,* 3, 111–126.

CLARK, A. D. 2001. Modeling the effects of burst packet loss and recency on subjective voice quality. In *Proceedings of the IP Telephony Workshop*. New York, New York.

COMSCORE, INC. 2008. YouTube attracts 100 million U.S. online video viewers in October 2008. http://www.comscore.com/press/release.asp?press=2616.

CSIZMAR DALAL, A. 2009. Revisiting a QoE assessment architecture six years later: Lessons learned and remaining challenges. In *Proceedings of the Third International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA 2009)*. Las Palmas de Gran Canaria, Spain.

CSIZMAR DALAL, A., BOUCHARD, A., CANTOR, S., GUO, Y., AND JOHNSON, A. 2010. Predicting video stream quality in real time using objective stream state measurements. Tech. rep., Carleton College. October.

CSIZMAR DALAL, A., MUSICANT, D. R., OLSON, J., MCMENAMY, B., BENZAID, S., KAZEZ, B., AND BOLAN, E. 2007. Predicting user-perceived quality ratings from streaming media data. In *Proc ICC*. Glasgow, Scotland.

CSIZMAR DALAL, A. AND OLSON, J. 2007. Feature selection for prediction of user-perceived streaming media quality. In *Proc SPECTS*. San Diego, CA.

CSIZMAR DALAL, A. AND PERRY, E. 2003. A new architecture for measuring and assessing streaming media quality. In *Proceedings of PAM 2003*. La Jolla, CA.

CSIZMAR DALAL, A. AND PURRINGTON, K. 2005. Discerning user-perceived media stream quality through application-layer measurements. In *Proc MSAN*. Orlando, Florida.

DUNHAM, M. H. 2002. *Data Mining: Introductory and Advanced Topics*. Prentice Hall.

FRENCH, H., LIN, J., PHAN, T., AND CSIZMAR DALAL, A. 2011. Real time video QoE analysis of RTMP streams. In *Proceedings of the 30th IEEE International Performance Computing and Communications Conference (IPCCC)*. Orlando, FL.

FU, A., KEOUGH, E., LAU, L. Y. H., RATANAMAHATANA, C. A., AND WONG, R. 2008. Scaling and time warping in time series querying. *VLDB Journal 17,* 4, 899–921.

G.107, I.-T. R. 1998. The Emodel, a computational model for use in transmission planning. Recommendations of the ITU, Telecommunications Sector.

HAND, D., MANNILA, H., AND SMYTH, P. 2001. *Principles of Data Mining*. MIT Press, Cambridge, MA.

HULU, LLC. Hulu. `http://www.hulu.com`.

IXIACOM. IxChariot software. `http://www.ixiacom.com/products/ixchariot/`.

KEOGH, E. AND RATANAMAHATANA, C. 2005. Exact indexing of dynamic time warping. *Knowledge and Information Systems 7,* 3, 358–386.

KEYNOTE SYSTEMS, INC. Keynote Streaming Perspective. `http://www.keynote.com/products/voip_and_streaming/streaming_performance/streaming_perspective.html`.

KROSNICK, J. A. AND FABRIGAR, L. R. 1997. *Survey Measurement and Process Quality*. Wiley-Interscience, Chapter Designing rating scales for effective measurement in surveys, 141–165.

LI, M., CLAYPOOL, M., AND KINICKI, R. 2002. MediaPlayer versus RealPlayer - a comparison of network turbulence. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*. Marseille, France.

LI, M., LI, F., CLAYPOOL, M., AND KINICKI, R. 2005. Weather forecasting - predicting performance for streaming video over wireless LANs. In *Proceeedings of NOSSDAV*. Stevenson, Washington.

MOHAMED, S., RUBINO, G., MOHAMED, S., AND RUBINO, G. 2002. A study of real-time packet video quality using random neural networks. *IEEE Transactions on Circuits and Systems for Video Technology 12,* 12, 1071–1083.

NETFLIX, INC. Netflix. `http://www.netflix.com`.

NICHOLS, J., CLAYPOOL, M., KINICKI, R., AND LI, M. 2004. Measurement of the congestion responsiveness of Windows streaming media. In *Proc NOSSDAV*. Kinsdale, Ireland.

P.910, I.-T. R. Subjective video quality assessment methods for multimedia applications. Recommendations of the ITU, Telecommunications Sector.

RATANAMAHATANA, C. A. AND KEOGH, E. 2004. 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.

SAKOE, H. AND CHIBA, S. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans Acoustics Speech Signal Process 26*, 43–49.

SERRAL-GRACIA, R., YANNUZZI, M., MARIN-TORDERA, E., MASIP-BRUIN, X., AND SNCHEZ, S. 2010. Quality of experience enforcement in wireless networks. In *Wired/Wireless Internet Communications*, E. Osipov, A. Kassler, T. Bohnert, and X. Masip-Bruin, Eds. Lecture Notes in Computer Science Series, vol. 6074. Springer Berlin / Heidelberg, 180–191.

TAN, P., STEINBACH, M., AND KUMAR, V. 2005. *Introduction to Data Mining*. Addison-Wesley.

TANG, R., WILLIAM M. SHAW, J., AND VEVEA, J. L. 1999. Towards the identification of the optimal number of relevance categories. *Journal of the American Society for Information Science 50,* 3, 254–264.

VLACHOS, M., HADJIELEFTHERIOU, M., GUNOPULOS, D., AND KEOGH, E. 2003. Indexing multidimensional time-series with support for multiple distance measures. In *Proc KDD*. ACM Press, New York, NY, USA, 216–225.

WANG, Y. AND CLAYPOOL, M. 2005. RealTracer - tools for measuring the performance of RealVideo on the internet. *Kluwer Multimedia Tools and Applications 27,* 3.

WANG, Y., CLAYPOOL, M., AND ZUO, Z. 2001. An empirical study of RealVideo performance across the Internet. In *Proceedings of IMW 2001*. San Francisco, CA.

WOLF, S. AND PINSON, M. H. 1999. Spatial-temporal distortion metrics for in-service quality monitoring of any digital video system. In *Proceedings of SPIE International Symposium on Voice, Video, and Data Communications*. Boston, MA.

YOON, H., YANG, K., AND SHAHABI, C. 2005. Feature subset selection and feature ranking for multivariate time series. *IEEE Trans. Knowledge Data Eng. 17,* 9.

YOUTUBE, LLC. YouTube. `http://www.youtube.com`.

Received ; revised ; accepted