# A New Architecture for Measuring and Assessing Streaming Media Quality

Amy Csizmar Dalal and Ed Perry
Hewlett-Packard Laboratories, Palo Alto, CA, USA
{amy.dalal, ed.perry}@hp.com

***Abstract*— Conducting quality assessment for streaming media services, particularly from the end user perspective, has not been widely addressed by the network research community and remains a hard problem. In this paper we discuss the general problem of assessing the quality of streaming media in a large-scale IP network. This work presents two main contributions. First, we specify a new measurement and assessment architecture that can flexibly support the needs of different classes of assessment consumers while supporting both new and existing measurements that can be correlated with user perceptions of media stream quality. Second, we demonstrate that a prototype implementation of this architecture can be used to assess a user's *perceived* quality of a media stream, by judicious choice and assessment of objective metrics. We conclude by discussing how this architecture can be used to predict future periods of stream quality degradation.**

***Index Terms*— quality assessment, analysis, measurement tool, measurement architecture, streaming media**

## I. Introduction

Much research effort over the past several years [1–5] has addressed the general problem of constructing scalable and relevant measurement infrastructures, network fault diagnosis methods, and fault prediction methods, particularly in the context of the Internet; see [6] for an overview. However, conducting quality assessment for streaming media services, particularly from the end user perspective, has not been widely addressed by the network research community and remains a hard problem.

In this paper we discuss the general problem of assessing the quality of streaming media in a large-scale system such as the Internet. In this context, "streaming media" refers to a combination of audio and/or video content that is accessed on-demand, at scheduled times, or live. On-demand and scheduled content is pre-encoded and stored at one or more media servers, while live content is created, encoded, and distributed in real-time.

In our discussion, we assume a client-server type system in which multimedia is delivered over unicast using UDP. Our system prototype uses Windows Media Server; thus the streamed media is delivered via Microsoft's Microsoft Media Server (mms) protocol over UDP.

This work presents two main contributions. First, we specify a new measurement and assessment architecture that can flexibly support the needs of different classes of assessment consumers while supporting both new and existing measurements that can be correlated with user perceptions of media stream quality. Second, we demonstrate that a prototype implementation of this architecture can be used to assess a user's *perceived* quality of a media stream, by judicious choice and assessment of objective metrics.

The rest of the document is structured as follows. In Section II, we present the case for client-side quality assessment and discuss relevant prior work in this area. In Section III, we identify indicators of poor received media quality and present our proposed methodology for conducting quality assessment of media streams. Section IV discusses a prototype implementation of this methodology and experimental results on its efficacy in measuring a client's experience of a media stream. In Section V, we discuss how this mechanism could be used in order to predict future periods of degraded media quality and what could be done with this information. We conclude in Section VI with some thoughts on future directions for this work.

## II. The case for client-side quality assessment

In this section we discuss some of the drivers for client-side media quality assessment, such as the need for appropriate metrics for prediction and analysis. In addition, we discuss the lack of appropriate assessment methods in existence today and how this impacts client-side quality assessment.

### A. Identifying appropriate metrics

Several factors make measuring media more difficult than measuring, for example, Web transactions or file transfers. Media sessions tend to be of a longer duration than file transfer sessions or Web sessions (see [7], for example, for RealAudio traffic characteristics). Media files are larger than the typical data file on the Web. Most significantly though, media metrics are very context-specific and temporal. For example, network packet loss of five percent may have a more significant effect on a stream with a high bandwidth requirement than on a stream with a low bandwidth requirement; high bandwidth streams typically involve the transfer of more packets per second than lower bandwidth streams, which means a larger number of packets have the potential to be lost.

Traditional metrics of network quality, such as average received bandwidth and average packet loss rate, are not adequate for assessing media quality. For example, one widely used metric in measuring streaming media is the instantaneous bandwidth required by the stream. While bandwidth tells us how many network resources a stream is currently consuming, it cannot tell us anything about the user-perceived quality of the stream. Variations in bandwidth may mean degraded quality, or they may be normal content encoding fluctuations for a particular stream. This example illustrates the need for observing both short-term and long-term metrics as well as the need to define appropriate metrics for the media services environment, beyond simple averages or quality scores.

The recipient of a media stream is the best authority to assess its quality. Thus, collecting data at the client-side is crucial to determine the user's perception of the stream. Subjective methods are often used to assess user-perceived received media quality, typically via a "mean opinion score", or a ranking of the quality of the viewed clip on a scale of one to five [8]. However, several issues make this method unattractive for large-scale use. For one, it requires users to actively participate in quality assessment. In turn, this requires training at least a subset of users on how to accurately assess the quality of a received video or audio stream. Additionally, the mean opinion score does not indicate what exactly is wrong with the clip, whether the clip buffered too much or was garbled in parts, without more detailed information as to how the score was achieved.

For large-scale streaming media quality assessment, an attractive solution is to use objective metrics, those which can be easily measured, in order to derive subjective quality metrics, those which relate to what the user sees. One such solution is given in [9] and [10], but it requires correlating measurements on both the sender and receiver sides. A more useful solution would assess received quality by taking a select set of measurements from the receiver and using these measurements to deduce the user's received quality.

A significant challenge is to derive assessments from these collected metrics with sufficient information to permit either the adaptation of quality of service parameters or the correct diagnostic action. For example, if an end user's quality suffers because of inadequate bandwidth for a particular media stream, how can the network and/or media source address the problem in real time? Can the network find an alternate, less-congested path with more available bandwidth? Or can the source reduce the rate at which it streams, by sending fewer encoded layers (if applicable)? Achieving this goal requires developing new test tools that can interact with client-side players and existing network measurement tools.

Ideally, quality assessments should be obtainable without need for end-user participation, under remote control. Marshaling the resources to test the impact of large-scale crowds should be possible, without requiring real end-users to consume their time in the effort. Fault isolation and diagnosis activities should be able to proceed with a minimum of coordination and cooperation of the afflicted parties. On-going infrastructure and service assessments should be possible without requiring end-user involvement. By identifying metrics that can predict future events that directly affect the user's received stream quality, we can potentially realize this goal.

### *B. The need for proper assessment tools*

Existing software tools, such as [11–14], are not adequate for large-scale assessment of streaming media quality from the client's perspective. They rely on synthetic test streams, synthetic applications, and/or arbitrary test points in the network. These tools can fail to detect application sensitivities to service quality, such as timeout and failure responses, stream start-up delay, or player stall, that are of relevance to actual media clients.

Control channel solutions such as Real-time Transport Protocol's RTCP [15] and agents in RealPlayer [16] and Windows Media Player [17] provide feedback on the quality of received streams to the media server (and, in the case of RTCP, other users as well). However, these approaches have inherent scalability issues which, in the case of RTCP, prevent it from widespread adoption and use. Also, in the case of the two commercial media players, the feedback mechanisms are limited to the media server only and have limited functionality in terms of what can be corrected and/or modified (mainly the rate at which the stream is sent to the player). For RTCP, feedback is limited to network-level metrics, such as network-level packet loss.

The approach we propose is to utilize existing client-side players in the assessment of streaming media quality via observations at the client. Previous work, such as [18, 19] has addressed this as well, although the solutions derived either require user intervention ( [18]) or entirely new media player applications ( [19]). We propose a method that does not entail the modification of the client-side media player application and that is completely transparent to the user.

## III. Methodology

In this section, we present our proposed methodology for streaming media quality assessment. We begin by discussing indicators of poor received stream quality that will be used by our methodology. We then briefly describe the system architecture, and conclude by discussing a client-side assessment tool that we have developed for the purpose of making quality observations at the client in an unobtrusive manner.

### *A. Quality indicators*

We have identified two indicators of poor received media quality. These two indicators are player buffer starvation and lost packets. We describe these in more detail below.

*a) Player buffer starvation:* Currently, media players such as RealPlayer and Windows Media Player establish a client-side receive buffer at startup. At the start of playback, the player buffers for a predetermined period of time, typically five to thirty seconds. This provides the player with a cushion of several seconds should something go wrong during the transmission of the stream and no packets arrive for a period of time. This mechanism is meant to prevent the player from having to stop and refill the buffer during playback. If the period of time over which no packets are received is long enough such that the player exhausts the receive buffer, the player will be forced to stop and refill the buffer during playback. We refer to this event as *buffer starvation*. During buffer starvation, the player must wait for new packets to arrive because it does not have any data to render. The user notices this buffer starvation; it manifests itself as "stop action" or a freezing of the last video frame rendered and the absence of an audio stream.

*b) Lost packets:* Both RealPlayer and Windows Media Player implement methods at the application level to request application-level packets that have not yet been received be retransmitted. This method allows for error correction and a degree of reliability over the unreliable UDP protocol, which does not support retransmissions at the transport layer. We refer to this method as "application-level retransmission". When an application-level packet is initially lost or delayed, the player will request for that packet to be retransmitted. If the retransmitted packet, or the original packet, arrive at the player before its scheduled playback time, the player will record this event as a successful retransmission and render the packet as usual. If, however, neither the original nor the retransmitted packet arrive before its scheduled playback time, the player will record this packet as lost. Lost application-level packets can manifest themselves as anywhere from a slight to a severe degradation in received video and/or audio quality. Examples of the effects of lost packets include the appearance of shadows or visible blocks in the video and garbled audio.

In this paper, we concentrate on player buffer starvation as the indicator of interest. A more detailed treatment of lost packets appears in [20].

### B. System architecture

We briefly sketch the proposed architecture for this system; the full architecture can be found in [20].

Figure 1 illustrates the proposed architecture. The main components of this system include assessment servers, media clients, data collection points, and report servers.

*1) Assessment servers:* The assessment servers together form the distributed control center of the architecture. They control the collection of data at the media clients and the distribution of this data to one or more data collection points. Assessment servers configure media clients to (1) schedule and execute tests independent of user activity; (2) detect user-requested media streams and selectively assess their quality; and (3) deliver assessment results to one of the data collection points, with appropriate load-balancing across the various points. They direct the analysis of the collected data, defining which analyses should be performed on which data by which collection point.

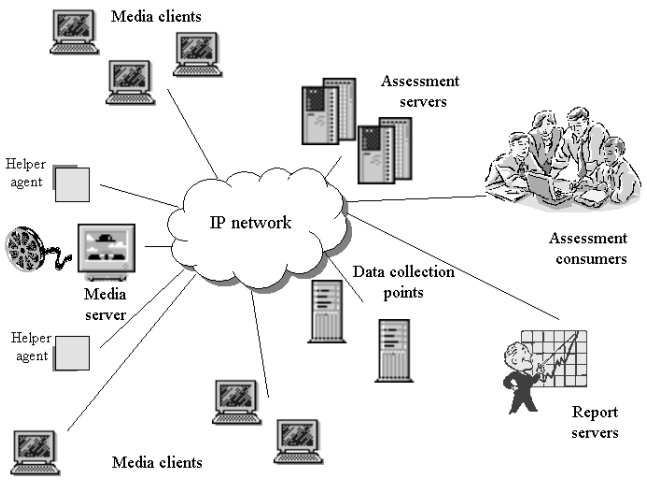


Fig. 1. Proposed architecture diagram

*2) Media clients:* The clients in this architecture are the end users whose computers host the streaming media assessment tool. The assessment tool collects data about the media stream as it plays out at the client and returns this data to one or more data collection points in the architecture.

*3) Data analysis:* Data analysis is accomplished by the data collection points and the report servers. The data collection points sample, reduce, and analyze the data received from the clients and send this analyzed data to the appropriate report server(s). The data collection points may also turn data collection at particular clients on or off during a test period or at any other time that a client-side assessment tool is collecting data. In the ideal scenario, the data collection points will be located "close" (in a topological sense) to clusters of clients; for example, a collection point may reside at the edge of a small ISP. The report servers aggregate analyzed data from one or more collection points and package this data for various subsets of assessment consumers. Note that an assessment "consumer" may in fact be a subsystem, such as an SLA verification system. Additionally, helper agents assist in the execution and collection of additional on-demand network-level measurements, such as traceroute, to supplement the data from the media clients. Such measurements lend additional supporting data to the primary analysis of client-generated data in order to more completely identify and correct faults in the system.

### C. Assessment tool

One of the main contributions of this work is a client-side assessment tool whose purpose is to collect information about

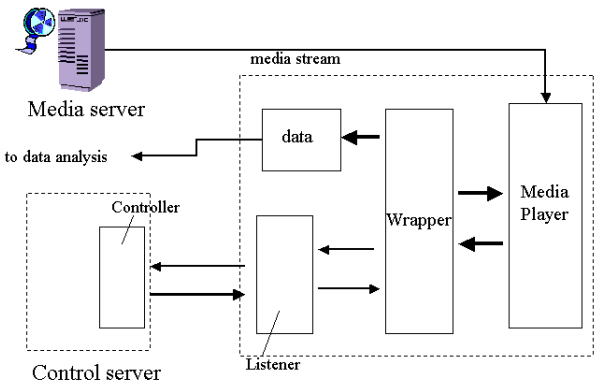


Fig. 2. Diagram of the client-side quality assessment tool.

the stream as it is playing out, *from the player itself*, and return this data to *multiple interested parties*.

The assessment tool is a standalone application that resides on each client machine within the system. It consists of three parts, two of which reside on the client machine and one which resides on a separate control server. We describe the tool in more detail below. Its operation is illustrated in Figure 2.

The main part of the assessment tool is a media player "wrapper". This wrapper is a standalone software application that interacts with the installed software media player's API. The wrapper polls the media player at predetermined intervals for playback metrics, such as the number of lost, recovered, and received packets; the current received bandwidth of the stream; and whether or not the player is currently experiencing buffer starvation.

It is important to note that the wrapper is independent of the media player software. Thus, the assessment tool can be developed independently of the media player software and does not require the modification of the existing media player, nor the installation of a new player onto the media clients.

Working in concert with the wrapper is an additional standalone application that resides at the client, the Listener. The Listener runs in the background on each media client. It acts as a liaison between the wrapper/player and the control server. It detects both user-initiated activity and control server-initiated activity, and reacts correspondingly. (The different modes in which this system operates will be explained shortly.) The Listener also monitors the wrapper during runtime for indications of success and failure conditions, such as the premature shutting down of the media player before stream playback ends.

The third component of the assessment tool is a Controller, which resides on a control server. The Controller directs the collection of data from the media player on a client via the wrapper. It determines which data should be collected from which client and on what streams, and where this data should be sent for analysis. It is a standalone software application that also runs in the background on the control server.

There are two modes in which this tool can operate. We term these two modes "test mode" and "user mode". We define each of these modes here and describe how the assessment tool operates in each mode.

In test mode, the assessment tool collects data from the media player independently of user activity. That is, data is collected on streams which users are not currently watching. Test mode can be used in large-scale testing scenarios, such as verifying the correct amount of network resources are available to support an upcoming webcast, as well as smaller-scale testing scenarios, such as troubleshooting quality problems on a particular subnet.

In user mode, the assessment tool collects data from actual user activity. That is, data is collected on streams that users are currently watching. User mode can be used to monitor the health of a network in terms of its support for streaming media, determine if media server capacity is adequate, and so on.

The assessment tool operates in test mode as follows. The control server predetermines which streams to test on which clients at what times, and then schedules these at the Controller. The Controller, at the specified times, sends a message to the correct client-side Listener containing information about the stream to play, the data to collect, for how long to monitor the stream, how often to poll the player, and where to send the data (either at the end of the stream or during playback). Details about the data to collect can be specified via the wrapper command line or in a configuration file, for example. The Listener verifies that the message came from a reputable control server. It then starts the wrapper, which in turn starts the player with the relevant parameters, such as the URL of the media stream. The player then contacts the specified server and begins playing back the stream. As the stream plays out, the wrapper polls the player at intervals specified by the Controller, and logs this information, sending it to the data collection point immediately or storing it for later transmission. At the end of the data collection period, or at the end of the stream playback, the wrapper shuts down the player, sends any remaining data to the specified analysis point, and logs success or failure conditions to the Listener. The Listener then indicates to the Controller that data collection has ceased, along with any other conditions that should be reported, such as error conditions.

The assessment tool operates in user mode in a similar fashion, only in this case action is initiated by the media player when the user starts the player on hisor her machine. The Listener detects this activity, determines if possible the name of the stream which the user is accessing, and then contacts the Controller with this information. The Controller decides if it wants to collect data about this particular stream for this particular user, and if so, which data to collect. It then sends a message to the Listener indicating if data collection should commence. If so, the Listener starts the wrapper, passing along details as to how often to poll the media player and what data to collect.

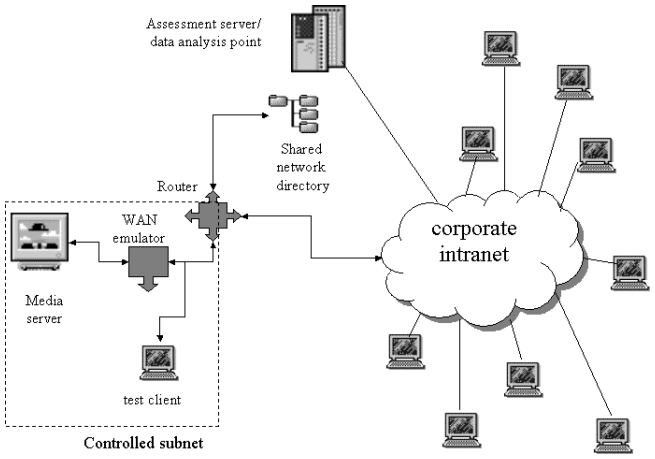


Fig. 3. The implementation of the test system architecture.

The wrapper then operates as in test mode, polling the player and gathering the requested data. At the end of stream playback, the player shuts down, initiating the same exit process as in test mode.

## IV. Experimental results

In this section, we discuss an implementation of the architecture and assessment tool and present some experimental results gathered with this prototype.

### A. Prototype implementation and testbed network

Figure 3 illustrates the prototype implementation and testbed network. The test network consists of two portions, a controlled subnet hosting the media server, and a population of test clients that reside on the corporate network. The controlled subnet is connected to but isolated from the corporate network. The controlled subnet hosts a WAN emulation tool, which emulates network congestion by dropping or delaying packets or reducing the available bandwidth to individuals or groups of users. The controlled subnet also hosts the media server used in the evaluation of the prototype.

The media server is a Windows 2000 server running Windows Media Services version 4.1. The WAN emulation tool is a Linux machine running NIST Net.

Our measurements on the corporate network show negligible packet loss, bandwidth limitation, and delay except in a few rare cases. This allows us to assume that any perturbations applied to the controlled portion of the network are the only perturbations of significance in our evaluation.

The test client population consists of twelve PCs each running Windows 2000 or Windows NT and Windows Media Player version 7.1 or 6.4. The clients are dispersed geographically over the United States and Canada. Via periodic measurements, we determined that latency was not a factor of significance in our measurements, and that all clients showed similar behavior in the presence of similar network perturbations regardless of geographic distance from the server. The clients all have access to a shared network mount point to which they have read and write access that serves as the data collection point.

### B. Prototype client-side assessment tool

Our prototype client assessment tool is a Java application that interacts with the Windows Media Player ActiveX Control, version 6.4, which is included with both the 7.1 and 6.4 versions of Windows Media Player. The assessment tool is completely independent of the media player software, meaning that installation of the tool does not require modifying the existing media player software, and that the media player can be used independently of the assessment tool with no difference in functionality. Thus, it can be deployed non-invasively on the media clients.

The remote control structure that resides on the assessment server to schedule and execute tests and collect data from clients is also implemented in Java. In the current implementation, only the remote testing capability is included; the control structure does not interact with the player outside of scheduled testing, although this functionality is currently being implemented.

### C. Experimental validation

To verify our prototype architecture, we conducted experiments on our testbed network between September 2001 and December 2001. Table I describes the media clips used in the experiments. All three clips are streamed via unicast over UDP, and all are streamed using Microsoft's mms format.

In our experiments, we introduce packet loss into the test network by two different methods and observe the results.[1] The first method introduces random packet loss into the network of one, five, or ten percent. The second method introduces deterministic periods of bursty packet loss on the network of one to three seconds over a sixty-second period. These bursts yield average network packet losses over the stream of 1.67%, 3.33%, and 5% respectively.

In the experiments presented here, the assessment tool polls the media player once every second. The metrics measured include received bandwidth; the number of packets lost, recovered, and received; and the start and end times of periods of buffer starvation, which the player reports as "buffering".

### D. Results

Figure 4 presents the CDF of the length of time over which a player reports that it is buffering (buffer starvation). The top plot shows all buffering periods as reported by the player. Notice that there is a rather large spike at five seconds. Upon

[1] In this discussion, "packets" refer to "IP packets". Throughout most of the rest of this paper, "packets" refer to "application-level packets". We will reiterate the distinction when necessary.

TABLE I

LIST OF MEDIA FILES USED IN EXPERIMENTS

| Stream | Duration | Size (MB) | Average BW (kbps) | Description |
|---|---|---|---|---|
| A | 2 min 4 sec | 16.5 | 457.5 | movie trailer |
| B | 11 min 26 sec | 15.5 | 107.2 | CEO message |
| C | 30 min (truncated) | 147.8 | 84.4 | presentation |

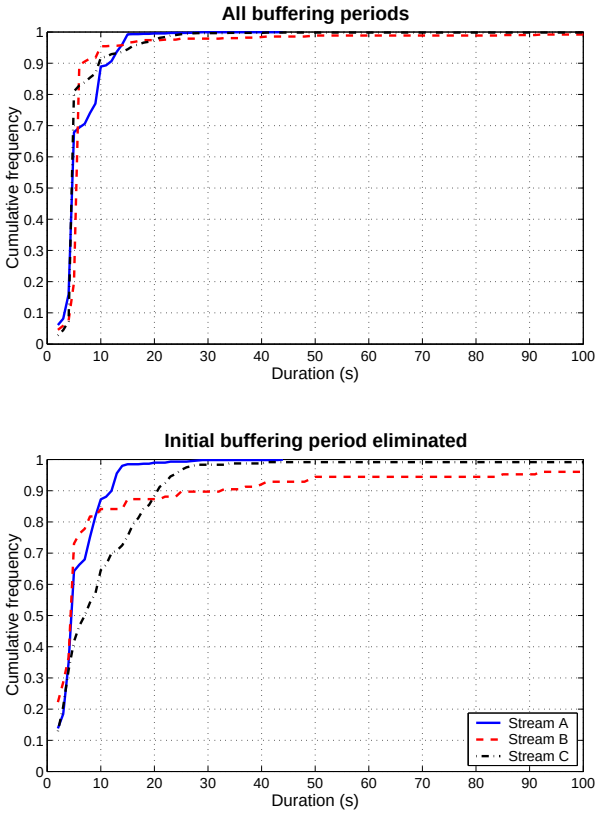


Fig. 4. CDF of the duration of "buffering periods", as reported by the player. The top plot shows all buffering periods measured in the data, while the bottom plot shows only the buffering periods that did not occur at the beginning of a stream (startup buffering).

further inspection, it was discovered that the majority of these buffering periods corresponded to the startup buffering period. We filtered these periods out; the CDF of the remaining buffering periods, which are periods of true buffer starvation during stream playback, is shown in the bottom plot. This plot shows that, while the majority of buffer starvation periods are less than ten seconds, there are a nontrivial number that are larger than ten seconds, and in a few cases larger than 100 seconds (Stream B).

Figure 5 shows the CDF of the duration of periods over which no new packets arrive at the player's receive buffer. Over 80% of these periods, among all streams, last for less than ten seconds; for Streams B and C over 95% of these periods last less than ten seconds. Because Stream A is a higher bandwidth stream, it is more susceptible to random network losses, which explains its longer tail as compared to the CDFs of Streams B and C.

Analysis of Stream A's tail led to the discovery that fragmentation of the application-level packets was occurring at the media server, at rates as high as 5 fragments per application-level packet. The encoder created packets that were much larger than the network MTU of 1500 bytes, forcing the media server to fragment the packets and increasing the apparent packet loss at the player. This is an example of how the client-side assessment tool can be used in the diagnosis of unexpected player behavior.

## V. PREDICTING DEGRADED MEDIA STREAM QUALITY

In this section, we discuss indicators that could be used to predict future degraded stream quality, as seen by the user. We define and present one particular indicator, discuss how knowing when a stream's quality will degrade would be useful and what could be done given this information, and present experimental results that demonstrate how useful this indicator would be and how realistic of a predictor it is.

### A. Defining the predictor

In a previous section, we defined player buffer starvation as an indicator of degraded stream quality. When a player is undergoing buffer starvation, no packets can be rendered, and the condition is visible to the user in the form of "freeze frame" video and no audio.

Periods of buffer starvation are typically preceded by a period of time over which no new packets arrive at the player's receive buffer. Thus, the player is forced to render packets that are resident in the buffer until the buffer runs out of packets, in which case buffer starvation occurs. We will use these periods over which no new packets arrive to predict when buffer starvation will occur. We define these periods as *packet reception pauses*.

A more rigorous definition of packet reception pauses follows and is illustrated in Figure 6. Let $A(t)$ be the arrival process, in packets per second, of the application-level pack-

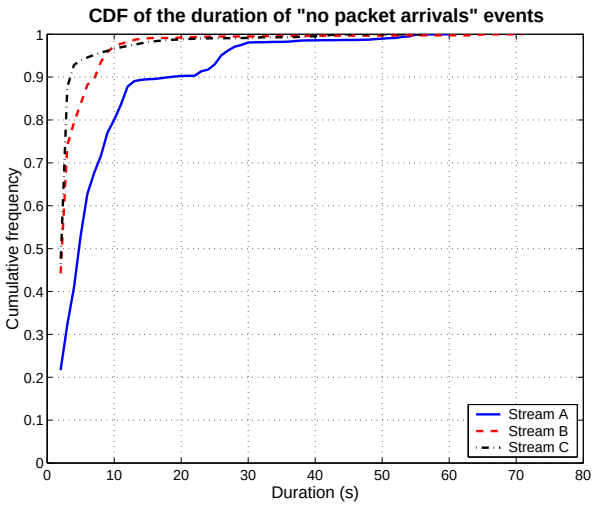


Fig. 5. CDF of the duration of periods where no packets arrive at the receive buffer, as reported by the player.

ets at the receiver buffer, a FIFO queue $B$ of length $L$.[2] $L$ is in units of seconds worth of playback data. Define $D(t)$ as the departure process from queue $B$ in packets per second. The departure process is defined by the rate at which the media player removes application-level packets from the buffer to be rendered (and thus the rendering speed of the client software). $A(t)$ is dependent on the state of the network and also the state of the media server from which it is drawing content. Because the media player software is proprietary, both $L$ and $D(t)$ are unknown. We know, however, that the startup buffering time is five seconds for the media player, so $L$ contains at least five seconds worth of audio and/or video data. By observing $A(t)$, we wish to determine the current length of the buffer, $l(t)$. We assume that $D(t)$ does not depend on either $A(t)$ or $l(t)$.

Given the above assumptions, a packet reception pause is defined as a period $t \in [T_{p1}, T_{p2}]$ such that $A(t) = 0$.

We are interested in determining which periods $[T_{p1}, T_{p2}]$ are

[2]We assume that any reordering of out-of-order packets occurs at the network layer.

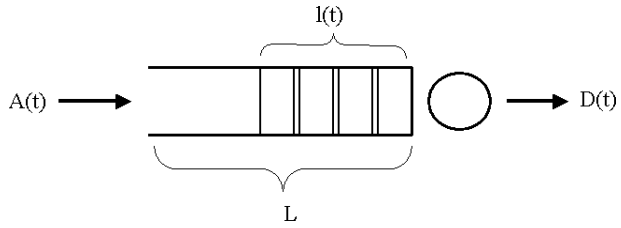


Fig. 6. he receive buffer as a FIFO queueing system, to illustrate packet reception pauses.

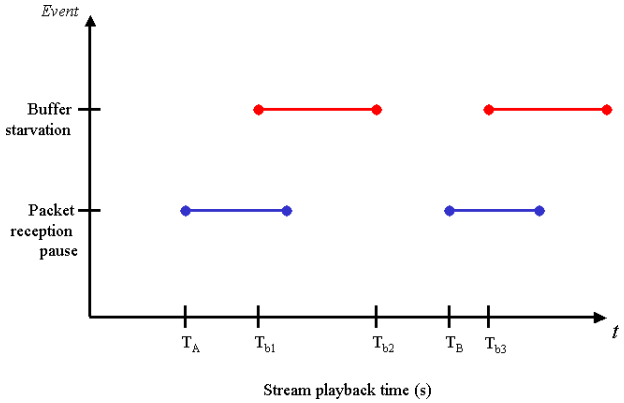


Fig. 7. Example of a buffer starvation period that follows a packet reception pause period.

followed by a period $t \in [T_{b1}, T_{b2}]$ such that $D(t) = 0$. In other words, periods of buffer starvation are indicated by no packets departing the queue. We define the interval $[T_{b1}, T_{b2}]$ to follow the interval $[T_{p1}, T_{p2}]$ in the following instance. Let $T_A$ and $T_B$ be the start times of two distinct periods of packet reception pauses. Then the buffer starvation period indicated by the interval $[T_{b1}, T_{b2}]$ follows the packet reception pause that starts at time $T_A$ if $T_A \leq T_{b1} \leq T_B$. Alternately, if $T_A$ indicates the start time of the last packet reception pause event in a stream, and $0 < T_{b1} - T_A < \epsilon$, where $\epsilon$ is a threshold value, then we say that the buffer starvation event $[T_{b1}, T_{b2}]$ follows the packet reception pause started at time $T_A$. As an example, Figure 7 illustrates a buffer starvation period that follows a period of no packet arrivals.

Because there is always a startup buffering period that occurs at the start of a stream, we ignore these in the following analysis. We also ignore any packet reception pauses that occur at the very end of a stream, as these are also very common in the data and simply mean that the player finishes playing out the stream from its receive buffer. Among our experiments, 64% of the streams ended with such a packet reception pause, comprising just under 11% of all packet reception pauses seen in the data.

### B. *Uses of prediction*

It should be noted that to some extent, the lack of newly-received packets is information that is already communicated

to the media server by the player (in the case of RealPlayer and Windows Media Player) as part of the player's loss recovery mechanism. We do not propose replacing this functionality. Rather, we propose to share this and other information with interested parties along the path of the media stream, such as network operators and ISPs. In doing so, it is entirely possible for these parties to use this information to either immediately adapt to the impending buffer starvation or to utilize this information in off-line analysis of the media stream. We focus here on the former scenario.

Given that third parties have access to this predictive information and could use it to mitigate the degraded quality at a media player, what could be done? There are several possibilities. First, a local ISP could serve local content to players that are about to enter buffer starvation. This content could be third party content, such as ads, or could be locally cached or "prefetched" content from either the same media server or another media server that is serving the same content. Second, networks could utilize this information to determine if the problem is originating with them, and if so either choose another path on which to route the data or require the user to access another server (if possible). If all else fails, the user could receive a message indicating impending failure of the delivery of the stream and give the user the option to either "wait it out" or come back at a later time.

### C. Experimental results

The percentage of packet reception pauses that are followed by a buffering event over all experiments is 7.4%. Figure 8 shows the percentage of packet reception pauses that are followed by buffering events, as a function of the duration of the packet reception pause in seconds. The packet reception pauses are binned into five-second intervals. The overall percentage is skewed lower by the presence of many small packet reception pauses of less than five seconds in duration, of which less than 3% result in a buffering event immediately following. Beyond ten seconds, this percentage rises significantly, with a few exceptions.[3]

Because there are so many small packet reception pauses that do not lead directly to buffering events, the question arises as to the proper threshold at which to consider packet reception pauses as likely indicators of buffering events. Figure 9 illustrates the percentage of packet reception pauses that are followed by a buffering event for all packet reception pauses greater than $x$ seconds. The point at which 50% of the packet reception pauses are followed by buffering events is seven seconds. This percentage rises to 60% if we raise the threshold value to ten seconds.

For the data presented here, by observing $A(t) = 0$ for five seconds we can accurately predict that a period of buffer starvation will follow 50% of the time. Whether or not there is enough time to feed this information to interested parties in time for them to take corrective action depends on the lag time between the onset of the packet reception pause and the onset of the period of buffer starvation. We are currently studying the lag times between these two events to determine if corrective action is possible in some or most cases of degraded stream quality.

[3]It is unclear why there is a significant drop in the 30-35 second range. The 55-60 second range contains less than ten data points, so we omit this bin from the plot.

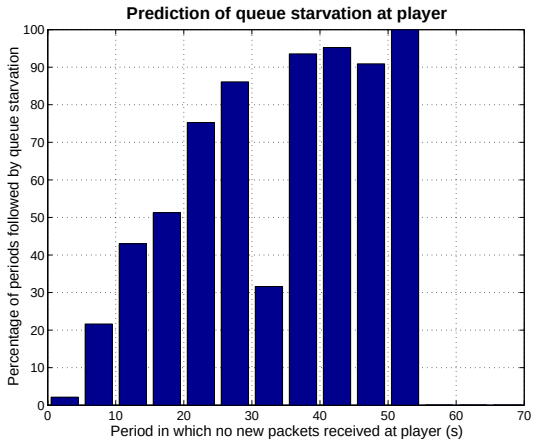


Fig. 8. Prediction of queue starvation by observing periods of time over which no new packets arrive. The periods are binned into five second intervals.

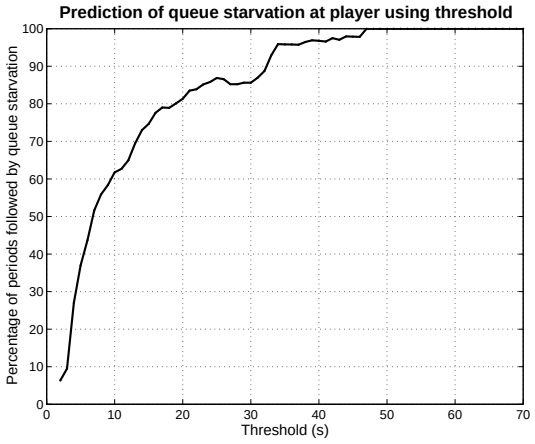


Fig. 9. Prediction of queue starvation using a threshold value for periods of no new packet arrivals.

## VI. Conclusions

In this paper, we have presented a new measurement and assessment architecture for determining streaming media quality at the client. This architecture differs from previous and current methods in that it is flexible, operating in both test mode and user mode; it supports multiple consumers of assessment data, such as ISPs, media servers, and content providers; and it utilizes existing client-side media players without requiring the modification of these players. We have implemented a prototype of this architecture, which we have used to demonstrate

the type of measurements it is capable of collecting and how these can relate to a user's perceived quality of a media stream. Finally, we discuss how one of these measurements can be used to predict future periods of degraded stream quality and present several scenarios in which this prediction would be useful.

There are several areas of future work that we are pursuing. We are currently assessing the scalability of such a system, in particular the scalability of the data collection and assessment mechanism. Also, our assessment tool has uncovered some unanticipated behavior in the way Windows Media Player and Windows Media Server interoperate, particularly in the size of the receive buffer as the stream plays out. We are utilizing this architecture in order to explore these discrepancies more fully. Finally, we are upgrading the measurement tool itself to fully support both user mode and test mode.

## ACKNOWLEDGMENTS

The authors would like to thank Puneet Sharma, Sujata Banerjee, and Jack Brassil for their assistance and feedback during this project.

## REFERENCES

[1] S. Kalidindi and M. Zekauskas, "Surveyor: An infrastructure for Internet performance measurements," in *Proceedings of INET '99*, San Jose, CA, June 1999.

[2] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, "An architecture for large-scale Internet measurement," *IEEE Communications*, vol. 36, no. 8, pp. 48–54, 1998.

[3] "NLANR's Network Analysis Infrastructure," `http://moat.nlanr.net/NAI/`.

[4] "Stanford Linear Accelerator Center's IEPM project," `http://www-iepm.slac.stanford.edu/`.

[5] D. Thaler and C.V. Ravishankar, "An architecture for inter-domain troubleshooting," in *Proceedings of the Sixth International Conference on Computers Communications and Networks*, Las Vegas, NV, September 1997.

[6] M. Murray and kc claffy, "Measuring the immeasurable: global Internet measurement infrastructure," in *PAM 2001 – A Workshop on Passive and Active Measurements*, Amsterdam, Netherlands, April 2001.

[7] A. Mena and J. Heidemann, "An empirical study of Internet audio traffic," in *Proceedings of the IEEE INFOCOM'00*, Tel Aviv, Israel, March 2000, pp. 101–110.

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

[9] S. Wolf and M. H. Pinson, "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, September 1999.

[10] W. Ashmawi, R. Guerin, S. Wolf, and M. H. Pinson, "On the impact of policing and rate guarantees in Diff-Serv networks: A video streaming application perspective," in *Proceedings of SIGCOMM 2001*, San Diego, CA, August 2001.

[11] "NetIQ's Chariot software," `http://www.netiq.com`.

[12] "Broadstream," `http://www.broadstream.com`.

[13] "Streamcheck," `http://www.streamcheck.com`.

[14] "Keynote Streaming Perspective," `http://www.keynote.com/solutions/html/streaming_perspective1.html`.

[15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," IETF RFC 1889, January 1996.

[16] "RealNetworks' RealPlayer," `http://www.real.com`.

[17] "Windows Media Player," `http://www.microsoft.com/windows/windowsmedia/players.asp`.

[18] Y. Wang, M. Claypool, and Z. Zuo, "An empirical study of RealVideo performance across the Internet," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, November 2001.

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

[20] A. C. Dalal and E. Perry, "An architecture for client-side streaming media quality assessment," Tech. Rep. HPL-2002-90, Hewlett-Packard Labs, April 2002.