



## Designing a collector overlay architecture for fault diagnosis in video networks

Mukundan Venkataraman<sup>a</sup>, Shamik Sengupta<sup>b</sup>, Mainak Chatterjee<sup>a,\*</sup>, Raja Neogi<sup>c</sup>

<sup>a</sup> Department of EECS, University of Central Florida, Orlando, FL 32816, United States

<sup>b</sup> Department of Mathematics and Computer Science, John Jay College, City University of New York, NY 10019, United States

<sup>c</sup> Radisys Inc., 15797 NW Andalusian Way, Portland, OR 97229, United States

### ARTICLE INFO

#### Article history:

Received 21 January 2010

Received in revised form 29 September 2011

Accepted 15 November 2011

Available online 25 November 2011

#### Keywords:

IPTV

VoD

Access network

Fault diagnosis

### ABSTRACT

To prevent subscriber churn, network service providers of VoD, SDV and IPTV have a pressing need to pro-actively detect, isolate and fix outages within an access network. Network induced degradations prove to be detrimental for streaming applications. This typically leads to a poor quality of experience (QoE) for subscribers. By monitoring key functional points of the access network for traces of degradation, service providers can devise mechanisms to mitigate the problem.

In this work we propose a hierarchy of exporters, collectors and ANCON (ANalysis and CONtrol) nodes that can semi-autonomously monitor, detect and isolate impairments within an access network. Exporters on the data plane gather and disseminate statistics for individual subnets, which are streamed onto “collector” nodes on an orthogonal plane. Collector nodes aggregate traffic from various exporters, and stream them onto the root of the tree (ANCON). With an even placement of exporters, root cause analysis can now take the granularity of loss rates or delay rates in individual segments or subnets of an access network. As an extension to our architecture, we show that the overlay can support instrumentations of quality evaluation for streaming video. As an example, we use a simple MOS plugin that is in part an extension of the ITU-T Erlang model to predict the quality of a video stream much before it reaches the end user. We show that our overlay can support a wide variety of quality evaluation metrics. Through extensive simulations and an implementation, we discuss issues of engineering such an overlay, isolating impairments in access networks, instrumenting MOS plugins and predicting video quality of multimedia streams in transit.

© 2011 Elsevier B.V. All rights reserved.

### 1. Introduction

Network service providers of video on demand (VoD), switched digital video (SDV) and IPTV are actively deploying triple and quadruple play networks that deliver voice, video and data services over shared and converged infrastructure. Service providers of VoD, SDV and IPTV typically revolve around an access network. This is a series of networked nodes managed by the service provider. An access network mostly consists of a source node, which acquires or generates streaming content, which is then passed through a series of other nodes until it makes it to the edge of an access network. From the edge, media is distributed to end users, mostly using fiber optics and passive interconnecting nodes. With the ever growing use of such services, detecting, isolating, and fixing problems in IP based access networks are pressing issues for service providers.

The usage of UDP/IP stack in access networks poses its own problems: (i) IP is a best effort service, and there is no guarantee that the network will not discard, duplicate, delay or mis-order packets; and (ii) UDP, on the other hand, has no provisions for detecting or recovering from packet losses, congestion, delays or jitters. Aggressive deployment of streaming content in access networks will eventually make them more sensitive to the level of service the network can provide. Such applications would then need a way to adapt their application parameters (like frame-rate, packet-injection rate, etc.) to the network status. This is something the transport stack in its present form, without explicit feedbacks, cannot guarantee. Even schemes that do utilize such explicit feedbacks rely upon round trip time (RTT) estimations and timeout upon a failure to receive an acknowledgment. Accurate RTT estimation is known to be non-trivial, and false timeouts and stalling for acknowledgments are not uncommon. Also, since new content is to be delivered continuously, round trip times may be too long to learn about or recover from an event.

The networking aspect apart, there has been a surge of interest in trying to infer the quality of a video stream in transit. Traditional quality evaluation schemes were primarily designed to test the effectiveness of transcoding schemes alone, and were never

\* Corresponding author.

E-mail addresses: [mukundan@cs.ucf.edu](mailto:mukundan@cs.ucf.edu) (M. Venkataraman), [ssengupta@jjay.cuny.edu](mailto:ssengupta@jjay.cuny.edu) (S. Sengupta), [mainak@eecs.ucf.edu](mailto:mainak@eecs.ucf.edu) (M. Chatterjee), [raja\\_neogi@ieee.org](mailto:raja_neogi@ieee.org) (R. Neogi).

designed with a networking element between the encoding and decoding processes. The process of delivering streaming content typically involves elements such as video-grooming, encoding, network transmission, decoding and playout. Since errors can potentially occur at each of these elements, there is a need to *continuously* monitor quality degradation. Quality evaluation strategies of the future shall require architectural support to perform effectively.

We propose a hierarchical overlay of decentralized *collector nodes*. A collector node is a sensor probe that captures and disseminates various statistics about a network, like number of dropped frames, network delay, jitter etc. on a given *segment* of a network. Note that most of present day access networks already have the necessary infrastructure to deploy this: compiling usage statistics for billing purposes requires snooping into the user traffic to gather statistics. We simply extend and leverage this basic infrastructure to form a hierarchical organization of such collectors with sensing elements at the leaf and a correlating platform at the root form a collector network overlay, or simply, collector overlay.

A collector overlay in converged multi-service networks plays an important role in fault diagnosis, capacity planning, dynamically assessing network optimization needs, and monitoring service usage. Collector overlays are hidden (i.e., they do not have published IP addresses) and unobtrusively gather service specific information at distributed sensing points, and then propagate them to the ANCON (ANalysis and CONtrol) root node. Based on sensitivity of observed phenomena, ANCON generates action events that are routed to appropriate provisioning elements of the network. The collector overlay thus becomes the distributed virtual sensor for the physical network that deploys services. Such a network implicitly solves three important problems: (i) it can locally identify impairment points; (ii) ANCON can provide real time feedback to source about the current network status; and, (iii) it enables a continuous quality scoring mechanism if such collectors can estimate video quality. Specifically, we make the following contributions:

- We present an architecture of a hierarchical collector overlay that gathers and disseminates network statistics. The nodes are decentralized and monitor local segments, and can control distribution of this information. We justify such an architecture with extensive experiments.
- Using our overlay, we *reverse engineer* causes of network induced degradation. Collector networks can isolate both the *cause* of degradation and *identify* impairment segments/subnets that caused the degradation. This helps service providers react to outages quickly and maintain high user satisfaction levels.
- Our architecture provisions a feedback mechanism where the source/local-segments can be informed of network statistics in real time. This would allow network aware streaming applications to better adapt to changing network conditions. This form of feedback is not possible with the conventional UDP/IP stack, which is inherently lightweight and best effort in nature.
- We create an instrumentation layer above the data plane that computes local MOS, and correlates to compute global MOS. We export to service providers an estimate of *local MOS* in a segment rather than a series of network statistics to monitor health of a segment. This, along with various other tunable parameters, allows for efficient root causes analysis where fault detection can be performed if the health of a segment goes down. Our architecture seamlessly encompasses a wide variety of quality scoring mechanisms, under a simplifying assumption that the scoring mechanism works on quantitative inputs that manifest as a unique quality score.

- Our MOS calculations at the ANCON node could *predict* the quality of video at destination much *before* playout. We compare these MOS predictions to PSNR and observe that the results corroborate with each other.

Though the main goal is to discover faulty behavior and predict the service quality of video flows, the architecture can be easily extended to perform other video and network performance related services. Some of these are performance discovery (e.g., measuring round trip times of TCP/UDP), topology discovery (e.g., part or full connectivity), service usage discovery (e.g., who is using what service), fraud discovery (e.g., illegal usage), and trend discovery (e.g., forecasting user demands and preferences).

The rest of the paper is organized as follows. Related work is presented in Section 2. In Section 3, we argue the need for collector overlay and discuss its relevant characteristic features. Our simulation testbed is described in Section 4. In Section 5, we reverse engineer playout degradation to its causes in the network, and isolate impairment points. In Section 6, we present a simple video-MOS model that considers delay and loss rate. We present a schema of calculating video-MOS continuously along the data path using collector nodes in Section 7. Conclusions are drawn in the last section.

## 2. Related research

Significant work has been done in the areas of overlay networks, video quality measurement and network level measurements. However, to the best of our knowledge, we are not aware of an overlay network architecture within an access network that can detect, isolate, diagnose impairments, and also allow implementations of video quality measurements. An overlay architecture is mostly applied at the application layer, and its deployment generally does not warrant changes to the underlying networking substrate. The concept of overlays is not new to networking: in fact, the Internet itself was designed as an overlay over the telephone network.

Overlays have enjoyed popularity in related networking domains. Resilient overlay network (RON) [2] improves robustness and availability of Internet paths between hosts. It can discover outages and suggest workarounds: it allows a small group of distributed applications to detect and recover from outages in seconds (compared to several minutes of recovery in the Internet). Though our work is similar in philosophy (discovering outages), RON is more applicable to general traffic over the Internet. Network aware applications can make use of the CMU Remos [10], which provides a wide range of information to the application in a network independent fashion. Like our architecture, Remos gathers and controls distribution of information about individual subnets. However, the design emphasis on Remos has to been to provide a unified information interface to applications which operate over a diverse networking substrate, like the Internet. Apart from video applications, there has been some interest in designing overlay architectures to enable packet loss recovery and rapid rerouting of VoIP packets [1] over the Internet. Once again, our area of focus and design philosophies are completely different.

Extensive research has been performed by various groups towards objectively assessing quality of a video stream. Even at a time when there is little or no consensus on the exact nature of such a model, the basic idea has been to provide a subjectively interpret various events on a video stream, such as loss, delay, jitter and so on. For example, authors in [11] train a random neural network to adjust to viewer responses, and infer the quality of a video stream based on what is observed along the data path. Typical

efforts usually culminate in a mean opinion score (MOS), which on a chosen scale rates the quality of a particular sequence. MOS as a metric has been known to have its share of drawbacks [7,19]: (i) subjects tend to avoid the extreme scores, (ii) “forgiveness effect”, where users tend to give higher rating when a playout is long and smooth, is not uncommon; and (iii) quality in itself is not a very well defined notion, and has many dimensions to it.

As an alternate to asking users to provide a MOS score, certain other techniques have also been employed. Some studies, instead, ask the user to provide certain specific feedbacks. For example, users could be asked to report visibility of certain artifacts on screen [12,13], or to suggest changes to artifacts to make the sequence more appealing to them [4,15]. As can be seen, there is no one consensus on an acceptable quality assessment at the time of this writing. Thus, and to make our architecture flexible to many types of scoring schemes, we make a simplifying assumption: quality assessment shall operate on the basis of certain well defined inputs that can be drawn from the data plane. Note that this assumption is oversimplifying, since it assumes no reference frame in place for a direct comparison. These inputs could report certain specific events that happen on the data plane, like a frame loss, the type of frame that was lost, delay, jitter and so on. As an example of instrumenting a quality evaluation based, we devise a rudimentary MOS that can be calculated from such inputs. Since our goal is not to propose a new standard for MOS, we instead show that we can infer the *health* of a network segment (instead of the video stream) to successfully identify impairments. This also implies that other quality assessment techniques, once standardized or found suitable, can easily replace our model of MOS, which only reports the health of a network segment.

There have also been various proposals for protocols and techniques that mitigate network induced errors. Zhu et al. [21] present an algorithm that allows for simultaneous design of source rate control and QoS-aware congestion control for streaming video. They utilize a virtual network buffer management, which allows to capture application specific QoS requirements which can be interpreted in terms of the source rate and the sending rate. Huang et al. [6] propose JitterPath, which is a noise resilient available bandwidth estimation scheme. There has also been considerable work on packet loss resilience: Yang et al. [20] exploit the fact that not all packets or frames are of equal importance in a video stream and design a recovery scheme based on FEC. We acknowledge the vast breadth of recovery schemes possible from prior research. We argue that, with the overlay and a feedback mechanism in place, one can use any of the available scheme or devise newer ones that can avoid or react to network related outages.

### 3. Collector overlay

VoD and IPTV services are usually centered around an access network. An access network has a video source, and terminates at an edge-QAM (or E-QAM). Access networks are usually managed by service providers, and are typically run over IP. This section details the instrumentation of a collector overlay in such access networks.

#### 3.1. Topology overview

A video network along with the possible collector overlay is shown in Fig. 1. Acquired video (either by satellites or alternative sources) is pushed by the VoD or the SDV pump. Video packets pass through a “Groomer” where variable bit-rate traffic is policed to an almost constant bit-rate traffic. Video data then passes through a series of routers (in the IP domain) before it reaches “M-join” (multicast join node). The routers ( $NE_0$  and  $NE_1$ ) poten-

tially multiplex various other traffic within the access network. At Mjoin, selected flows are pulled on-demand (via control plane infrastructure not shown) through the Edge-Qam (EQAM). The access network terminates at E-QAM, and what follows next is a series of “passive” nodes. Bandwidth allocation for all video flows are done at the E-QAM and streams distributed over fiber nodes. The video signal is amplified before reaching the set-top box (STB) and the corresponding television (TV).

We begin with an overview of the overlay architecture. The topology can be loosely classified in two broad categories: (i) exporters, or elements that have information to stream, and (ii) collectors, elements that gather incoming parameters, and potentially pass them to higher nodes. Exporters snoop into user traffic that they forward, and stream flow statistics to collectors. Usage of the collector overlay calls for some judiciousness in monitoring. If too many parameters are monitored per flow, the traffic volume of the collectors will begin to show exponential growth. This problem is attributed to two clauses: (i) the number of parameters per flow, which is to identify the most pertinent information desired; and (ii) the *rate* (i.e. frequency) of information streaming, which in our architecture is a tunable parameter exported to service providers.

Extensive simulations have been conducted (see, for e.g. [18]) to test the sensitivity of subjective video quality perceptions in the face of various networking events. Of the metrics tested, namely delay, loss, jitter and error, it is well accepted that loss is the most damaging, with jitter and error being the other factors in that order. Loss correlates to missing information. Each lost packet is either a part of a given frame, or worse, part of a frame that is used as a reference frame to reconstruct other frames. Our overlay is presently designed to collect statistics on loss, delay and jitter. Every dropped packet directly increases the loss count. Delay of a packet at a collector is defined as the difference between the time a collector observes the packet to the time it was injected into the access network (timestamp). Jitter is accordingly calculated as the variance in the average delay for that given flow. These statistics are bundled into an information packet that is streamed onto the orthogonal plane. This streaming happens at a *rate*, which is roughly one statistic packet per 5 s per flow in our design.

#### 3.2. ANCON: correlation platform

While the video passes through the data plane (shown with solid arrows), probing can be done at various stages to monitor the quality of the flow. The collector overlay shown in Fig. 1 has five collector nodes. These collector nodes constantly monitor and measure various network and flow parameters. Information gathered by leaf collectors are highly localized. In other words, a given collector is completely oblivious to impairments happening in a preceding and succeeding segment. The statistics thus gathered are streamed to the ANCON node. Such a mode of operation achieves two things: (i) local collector data can be inferred as an *absolute* health of a particular segment, and (ii) global correlated data at ANCON shows *relative* fluctuations among various segments.

#### 3.3. Streaming protocol

Given the collector overlay set-up, there is a need for a streaming protocol that exports network statistics to correlation platforms. Streaming usage statistics to correlation nodes for usage, accounting and billing purposes have been popular, but they are inadequate in supporting evolving requirements from service providers. For example, RADIUS [16] is a widely deployed protocol that may be used for exporting usage information. However, it can only handle a few outstanding requests and has extensibility

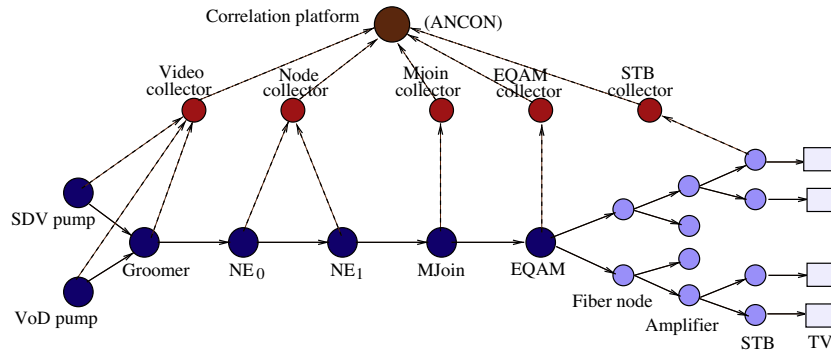


Fig. 1. Collector overlay: ANCON, collector nodes and the data plane.

challenges due to its limited command and attribute address space, complexity, and vendor specific attributes [27]. DIAMETER [3] is a new Authentication, Authorization and Accounting (AAA) protocol that retains most of the RADIUS structure but overcomes many of its shortcomings. However, it is highly optimized for streaming accounting information and its design does not stress on efficiency and performance for a large number of flows. SNMP based approaches are also inadequate, since they require a large amount of processing and bandwidth [5]. Further, the request-reply dialog of SNMP is not too well suited for such purposes.

Of particular interest is the recent Internet Protocol Detail Record, or IPDR [26] which is especially geared towards high volumes of reporting traffic. IPDR specifies the mode of exchange of such streaming statistics, and is generally classified in two parts: (i) exchanging information in XDR format, and (ii) an IPDR streaming protocol (IPDR/SP [27]). The streaming protocol, though not freely available for download, identifies key requirements that such a protocol must possess: (i) reliable, in-order delivery of data, (ii) connection oriented, (iii) session level authentication, (iv) message based delivery and (v) fast connection failure detection. A recommended transport layer for such requirements is the SCTP [17] (though TCP [14] possesses features (i) and (ii), it does not support either of (iii), (iv) and (v) as requirements). We use a network stack (with SCTP) for the collector overlay that unconditionally streams statistics. The streaming can be controlled by a “rate” visible to the network provider to better adapt to variations in the overlay.

#### 4. Testbed set-up

We set up a collector at nodes 1, 3, 4 and 5 (Fig. 2). Collectors monitor their individual subnets and make statistics within their subnet visible to the root node. For example, the collector at node-1 (collector-1) makes visible events that happen between the video source and node-1, while the collector node-2 (attached to node 3) makes visible events between node-1 and node-3.

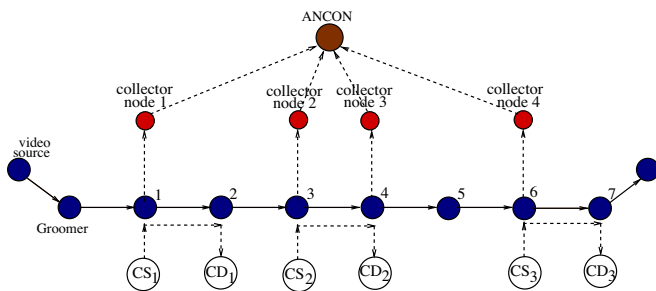


Fig. 2. Simulation topology.

The set-up was designed to establish feasibility of collecting and analyzing data via placement of collector nodes. We use the MPEG-4 [9] specification to encode the source video stream and the network simulator (ns-2) [25] to simulate the network. We use packages such as Evalvid [8] and MSU Video Evaluation Toolkit [28] to better model the network as close to reality as possible. Ns-2 has well tested simulation suites to model various transport protocols, buffering strategies, links and network dynamics.

We perform the following sequence of events. First, we take a raw video in a YUV format and perform MPEG-4 encoding to generate frames. These frames are then converted to IP packets with a fragmentation limit, and transported over the conventional UDP/IP stack. At the destination, IP packets are reassembled to form the MPEG-4 frames which are then played out.

Video grooming is used to inject frames into the network at a constant rate instead of overwhelming the network with a naive UDP blasting strategy. This is achieved by creating a custom traffic in ns-2 that injects video packets at a pre-defined interval into the network. The path taken by the video stream is as follows: the source node (node “video source”), MPEG-4 “encoder”, a video “groomer”, the intermediate IP hops (nodes 1 through 7) which potentially multiplex traffic from various sources, MPEG-4 “decoder”, and a destination node (node “destination”) where decoding and playout occur. Collector nodes (marked “CN” 1 thorough 4) are attached to data-plane nodes (nodes 1, 3, 4 and 6). The collectors form a simple hierarchy (a tree of depth 1) and correlate data at the ANCON node. The streaming behavior mimics the IPDR streaming protocol, which works by disseminating control information periodically for a specified “rate” of streaming. We also set up a competing traffic originating at nodes with names prefixed with CS (competing source) and terminating at nodes prefixed with CD (competing destination) respectively, marked 1 through 3. The rate and duration of this competing traffic, in combination with a choice of various other simulation parameters (shared buffer/queue length, link speeds etc.) are tuned to create various events in the network like delay, jitter, and loss. Other simulation parameters are shown in Table 1. This simulation topology is consistently used for the rest of the paper.

#### 5. Isolating impairment points

We show how our collector overlay can isolate impairment points within an access network when degradation occurs. Though traditional quality evaluation metrics like PSNR can provide a measure of degradation, they cannot pinpoint the exact cause of it. On the networking side, though lightweight protocol stacks such as UDP/IP are highly suited for streaming applications, it is not possible to isolate impairment points in a network because of the lack of feedback between networking entities. Collector networks can, however, unobtrusively achieve these with relative ease.

**Table 1**  
Simulation parameters.

Video parameters	Value
Resolution	352 × 288 pixels
Frame rate	30 frames/s
Video color mode	Y, U, V (4 – 2 – 0 scheme)
GOP length	30 Frames
Sequence length	10 s of playout
Network parameters	Value
Unshared links	1 Mbps
Link delay	1 ms
Buffer length	50 packets
Queue behavior	Droptail
Packet size	1052 bytes
Max. fragmented size	1024 bytes

In this section, we *reverse engineer* the causes of degradation. When a respective degradation happens during playout, we corroborate with statistics from the collector nodes to establish their ability in isolating impairments. For example, we seek data from collector nodes when the playout (i) is smooth and without distortion; (ii) has frame freezing in it; and, (iii) playout has garbled images. With collector nodes, we can identify both the *cause of the impairment* as well as *isolate the segment* that caused the impairment. Feedback received from the collectors make way for a plethora of corrective actions possible at the source host or with the choice of protocols used.

### 5.1. Delay in a segment: frame freezing

To simulate excessive delay in a segment, we introduced a series of ephemeral VBR flows from the 3rd second of playout between  $CS_2$  and  $CD_2$  in Fig. 2. This was done in conjunction to increasing the buffer size of node-5 to ensure no packets were lost. With an increase in competing traffic and large queue build up at node-5, the segment between node 3 to node 5 is at fault.

A direct consequence of large delays is frame “freezing” during playout. Freezing is when the motion picture gets stuck at one particular frame during playout. Since successful decoding relies upon a constant arrival of frames for consumption at the destination buffer, a delayed frame is as good as a lost frame. In this case, the decoder presents the *last* successfully constructed frame until another frame arrives at the destination buffer. Since our competing traffic starts at the third second of playout, the first part of the playout is smooth while the later part has frozen frames due to network delays. We present a snapshot of freezing during playout and reverse engineer its cause. While playout at source changes from Fig. 3(a) and (b) for frames during the 7th and 9th second, the playout at destination is stuck at one frame as seen in Fig. 3(c) and (d). Note, however, that no frames were lost as a congestion drop because the queue size at node-5 was chosen to be sufficiently large.

The cumulative delay plots as seen across the four collector nodes for this scenario is presented. While the first two collector nodes before the faulty segment register an even increase in delay (Fig. 4(a) and (b)), the collectors at and after the faulty segment clearly show increasing delay in the segment between node-3 and node-4 (Fig. 4(c) and (d)).

### 5.2. Loss impairment

Competing traffic was introduced in the first segment (node 1 to node 2 region) between  $CS_1$  and  $CD_1$  for the first three seconds of playout. VBR flows were introduced with default queue sizes at node 1.

With IP fragmentation, each frame is broken into many packets (an I-Frame was usually broken to 16 packets with a fragmentation limit of 1024 bytes). A lost packet is hence a part of a frame. Since successful decoding depends upon the reference frames being intact, a lost packet corrupts a frame, or worse, hampers reconstruction of other frames. Loss is undoubtedly the most damaging event to video quality. Depending upon the relative importance of a packet lost in the network, the effect on playout varies from the appearance of minor “blocks” in the video (corruption of P-Frames) to a complete “whiteout” when nothing appears on screen (loss of an I-Frame, especially the first I-Frame of a temporal sequence) [18].

We reverse engineered from a damaged playout when only edges were visible on screen (Fig. 7(b)) compared to playout at source (Fig. 7(a)).

The loss statistics gathered from the collector nodes are as shown in Fig. 5, which clearly shows 86 packets lost in the first segment with no losses at any other segment. The delay statistics are also interesting (Fig. 6(a)–(d)), which show an increase in cumulative delay at and after the second collector. When drops occur at a node, the queue sizes naturally tend to increase to a maximum. Packets which see a full queue are dropped, while the ones enqueued have to wait their turn before being services which accounts for the increase in delay.

## 6. Measuring quality of video

While streaming loss and delay statistics does provide some insight into the network, the large bulk of unprocessed data eventually adds little interpretative insight. Network service providers are already realizing the need to understand the health of their infrastructure in terms of *Quality of Experience*, which tells them how an end user perceives their service. In this section, we devise a simple video MOS as an example of a plugin that can process the gathered statistics and provide interpretative data to service providers. While our MOS may not be the most extensive, the purpose of this exercise is to demonstrate that MOS instrumentation can be easily plugged into our architecture.

The quality of reconstructed video frames are often measured by the mean opinion score (MOS). MOS is a subjective quality score that ranges from 1 (worst) to 5 (best) and is obtained by conducting subjective surveys. To continuously monitor the quality of a video stream, we need a way of calculating MOS from the statistics provided by the collector nodes. This section presents one such model with the following goals: (i) establish feasibility of performing this calculation locally at collector nodes; (ii) prove that continuous monitoring is possible *without* actually decoding the video frames. Our model is in part a mutation of the Erlang model recommended by the ITU-T [23,24]; (iii) create a model that can translate link layer statistics (like delay, loss, jitter, etc.) into a more meaningful subjective MOS. This would help service providers draw meaningful inferences about an access network’s health using the collector overlay; and, (iv) provide a way to aggregate traffic from exporters. Statistics from exporters can be conveniently converted to a representative MOS by intermediate collectors which could dramatically reduce the traffic volume in the overlay flowing towards ANCON.

### 6.1. Video-MOS

We use the video-MOS characterized by the ITU-T E-Model [23,24] (for audio, video and audio-visual). The adaptation of video-MOS can be expressed with the help of parametric estimations that combines different aspects of quality impairment. For this



**Fig. 3.** An example of frame freezing observed when large delays or loss of intermediate reference frames are lost. While the motion picture changes from scene at (a) to scene (b) at the 7th and 9th second, playout remains stuck as seen in (c) and (d).

purpose, we first define  $Q$ -factor, that considers the effect of delay and loss impairments on the quality of a video stream. It is given by

$$Q_{der} = Q_{original} - I_e - I_d \quad (1)$$

where  $Q_{original}$  is the quality factor of the original video and  $Q_{der}$  is the derived quality factor of the captured video at the receiver's end.  $I_e$  is an equipment impairment factor associated with the losses due to the codecs and network and  $I_d$  represents the impairment caused by the delay.

For all parameters used in the algorithm of the E-model, the default values are listed in Table 2. We use these default values for all parameters which are not varied during planning calculation to find the maximum value of quality factor, which eventually can be considered as  $Q_{original}$ . If all parameters are set to the default values, the calculation results in a very high quality with a quality factor of  $Q_{original} = 93.2$ .

### 6.2. Effect of delay impairment $I_d$

For streaming video, total delay is composed of three components: codec delay ( $d_{codec}$ ), playout delay ( $d_{playout}$ ), and network delay ( $d_{network}$ ). Codec delay represents the algorithmic and packetization delay associated with the codec and varies from codec to codec. Playout delay is the delay associated with the receiver side buffer required to smooth out the delay for the arriving packet

streams. Network delay is the one-way transit delay across the IP transport network from one gateway to another. Thus the total delay is

$$d = d_{codec} + d_{playout} + d_{network} \quad (2)$$

The delay impairment  $I_d$  depends on the one way delay encountered. The effect of this delay can be modeled as [22],

$$I_d = \alpha_1 d + \alpha_2 (d - d_{threshold}) \mathbf{H}(d - d_{threshold}) \quad (3)$$

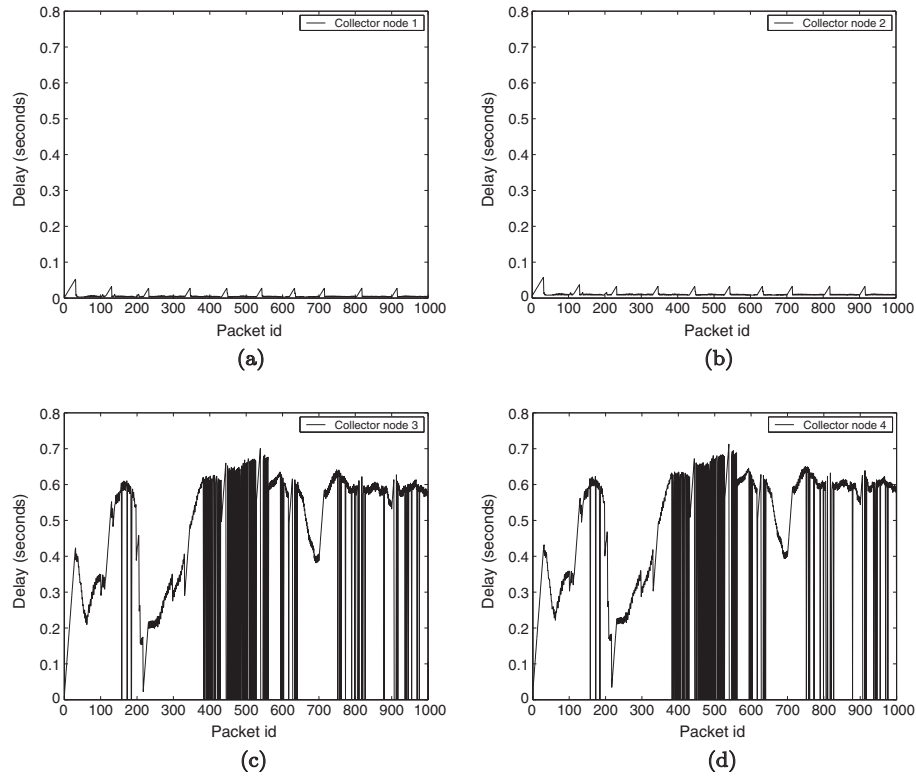
where  $\mathbf{H}(x)$  is an indicator function:  $\mathbf{H}(x) = 0$  if  $x < 0$ , and 1 otherwise.  $d_{threshold}$  is some delay threshold exceeding which might degrade the video quality excessively.  $\alpha_1$  and  $\alpha_2$  are two experimental parametric estimators to measure the effect of delay on quality factor.

### 6.3. Effect of loss impairment $I_e$

Similarly, we model the loss ( $e$ ), which includes all kinds of losses. The modeling can be done as

$$e = e_{network} + (1 - e_{network})e_{playout} \quad (4)$$

where  $e_{network}$  is the loss rate due to the loss in the network and  $e_{playout}$  is loss rate due to the playout loss at the receiver side.



**Fig. 4.** Cumulatively delay at the four collectors in the presence of competing traffic causing large delays in the second segment: (a) CN1 shows uniform delays, (b) CN2 also registers a uniform delay, (c) CN3, however, registers large delays, and (d) CN4 registers similar cumulative delays. This identifies the second segment with large delays.

We believe that as the collector nodes are probing the video stream at the packet level, no playout of the video is necessary at this scenario thus making  $e_{\text{playout}} = 0$ .

Then the loss impairment factor can be given as

$$I_e = \beta_1 \ln(1 + \beta_2 e) \quad (5)$$

$\beta_1$  and  $\beta_2$  are two experimental parametric estimators to measure the effect of loss on quality factor.

#### 6.4. Video-MOS calculation

We rewrite Eq. (1) using Eqs. (3) and (5) as

$$Q_{\text{der}} = 93.2 - \beta_1 \ln(1 + \beta_2 e) - (\alpha_1 d + \alpha_2 (d - d_{\text{threshold}}) \mathbf{H} \times (d - d_{\text{threshold}})) \quad (6)$$

where  $Q_{\text{original}}$  is assumed as 93.2.

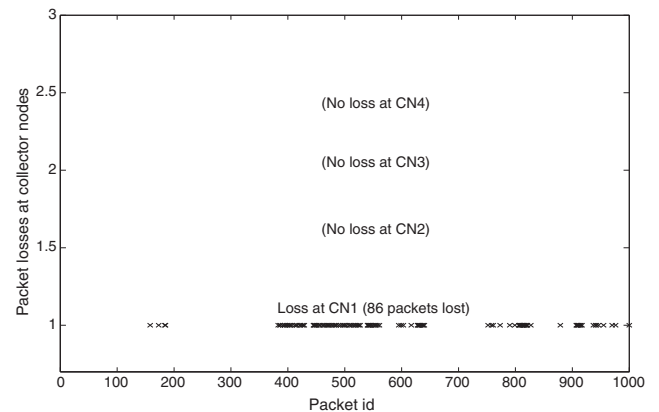
The quality-factor can be then related to MOS through the following non-linear mapping provided by ITU-T [22,24],

$$\text{MOS} = 1 + 0.035Q_{\text{der}} + 7 \times 10^{-6}Q_{\text{der}}(Q_{\text{der}} - 60)(100 - Q_{\text{der}}) \quad (7)$$

We use this definition of MOS for calculating quality of video stream in Section 7.

### 7. Correlating MOS with collectors

Using the expression for MOS, we proceed to demonstrate the effectiveness of the collector overlay in locally computing MOS, and corroborating the global MOS at the ANCON node. Simulations in this section assume a full fledged network traffic with persistent competing traffic (traffic with varying sending rates at all three competing traffic segments). The collectors observe delay and loss at various segments and calculate local MOS. Using these statistics, the ANCON is able to compute global MOS of the stream and predict

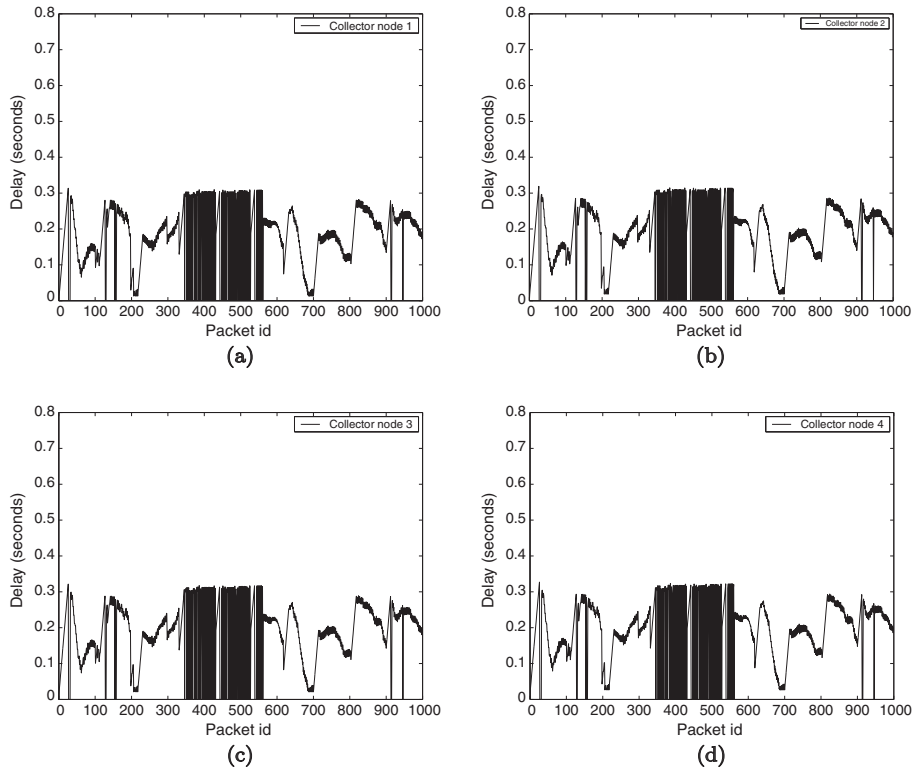


**Fig. 5.** Loss statistics at four collectors identifying, with the first collector registering 86 losses. This correlates to an impairment at segment-1.

a subjective rating of the video much before decoding at destination happens. We consider each event as follows.

We implement our collector overlay with MOS instrumentation on standard Linux terminals running Fedora Core 6. The exporter code is written using the libpcap library.<sup>1</sup> Statistics are streamed onto collector nodes using the IPDR/SP [27] protocol. The free version of IPDR/SP available for download makes the exporter code visible but comes with a pre-compiled image of the collector code. Collectors dump received information onto a text file, which is further processed using scripts. Using these statistics, we instrument our MOS calculation, and stream these calculations towards ANCON.

<sup>1</sup> Code available at <http://www.cs.ucf.edu/mukundan/netdoctor>.



**Fig. 6.** Delay statistics at the four collectors when loss happens at the first segment: (a) CN1 varying delay patterns, since the queue at node-1 fills to maximum leading to drops; (b) CN2, (c) CN3 and (d) CN4. Observe cumulative delays after the first segment since there are no competing flows in subsequent segments.



**Fig. 7.** Screen shots of video playback at (a) source, and (b) destination. Notice the intangible output caused by losses of reference frames in the very first second of playback.

**Table 2**  
Default values and permitted ranges for the parameters [22,24].

Parameters	Default value	Permitted range
Mean one-way delay of echo path	0 ms	0–500 ms
Absolute delay in echo-free connection	0 ms	0–500 ms
Equipment impairment factor ( $I_e$ )	0	0–40
Random packet-loss probability	0%	0–20%

Our topology is similar to the simulation testbed (Fig. 2), with the exclusion of nodes 2, 5 and 7.

7.1. Computing local MOS

To compute MOS, we calculate delay and loss rates, measured as the average delay and loss per epoch of time. We introduce a live

video stream through our testbed, and calculate MOS in the presence of a VBR competing traffic, which causes delay and loss in the intermediate nodes. In Fig. 8(a)–(d), we show the video-MOS (as calculated from Eq. 7) at the four collector nodes.

It is interesting to note the health of individual segments in terms of video MOS. Segment-3 intuitively provides a smooth and consistent MOS<sup>2</sup> while segment-1 (Fig. 8(a)) and segment-4 (Fig. 8(d)) register the largest fluctuations in values. It is easy to observe from the MOS calculations that loss has a large effect on MOS: the peaks in loss rates have a direct impact in the MOS calculations.

<sup>2</sup> Though the plot in Fig. 8(c) seems to be a straight line, there are very slight variations of amplitude 0.01 within the line. This is largely because of delay variations in segment-3 as seen in Fig. 8.

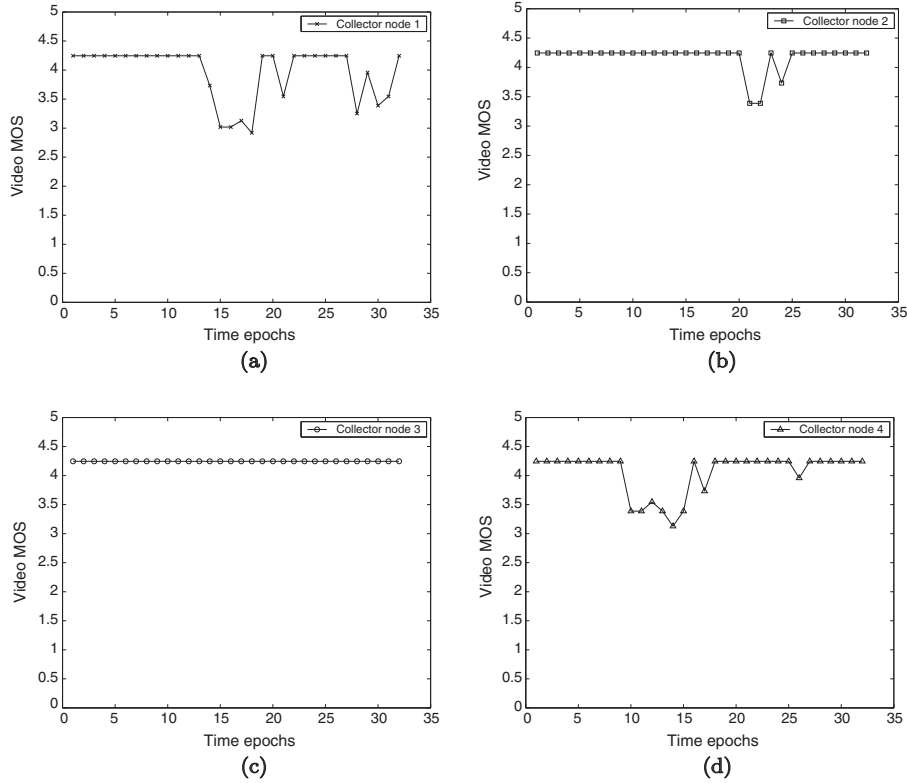


Fig. 8. Local MOS calculations at the four collectors: (a) CN1 shows maximum fluctuations, especially when loss rates are high, (b) CN2 shows large fluctuations at high loss rates around the 22nd epoch, (c) CN3 registers no large fluctuations and poses a consistent MOS rating, and (d) CN4 analogously registers fluctuations at high loss rates.

7.2. MOS at correlation platform

Since statistics are streamed to the ANCON node, it is able to globally correlate various events at all four segments of the data path. Using this, we calculate a global MOS at ANCON. Unlike collectors which only see local events and compute local MOS, ANCON can *cumulate* effects of one collector to those of another. In the sense, if packets are lost at a collector segment, its effects can be easily perceived at the successive segment since quality of the video has already degraded. Also, equipped with individual MOS patterns at various collectors, the ANCON is able to identify impairments in term of local MOS.

It is interesting to observe how ANCON's projection of MOS compares with more established metrics like PSNR. Note that ANCON calculates MOS using network events like loss and delay.

PSNR, however, decodes the received frame and compares that to its corresponding frame at source. Such a direct comparison can directly identify differences in frame content, and this returns a score of quality. We plot ANCON's projection of MOS in various epochs in Fig. 9(a), and directly compare this to PSNR values calculated with source and reference frames. For the PSNR plot, we compute the *average* PSNR values in various epochs (to allow a direct comparison). The plots are shown in Fig. 9(a) and (b). Notice the striking similarity in projections. The ANCON was hence successfully able to project degradations, and this happens much before actual decoding or playout occurs at destination.

We present two screen-shots of the video at two different instances. Fig. 10(a) shows the 75th frame of the video. This frame falls in the 9th time epoch which has a MOS of 4.2575. Similarly, Fig. 10(b) shows the 261th frame. This frame falls in the 28th time

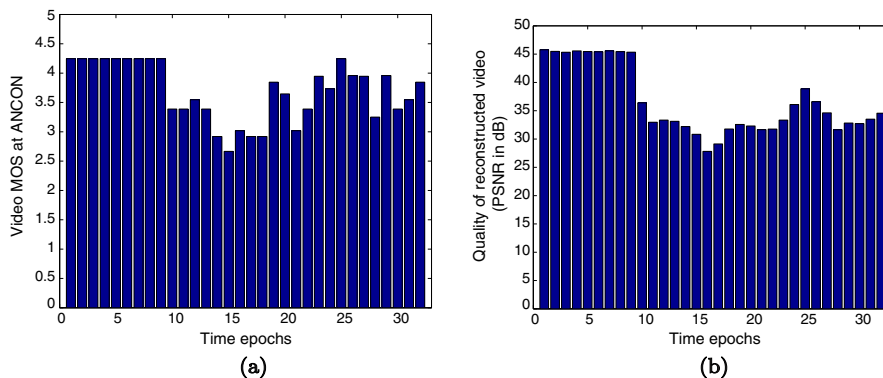


Fig. 9. Global MOS for the entire stream as processed at ANCON plotted against PSNR: (a) ANCON correlates from various collectors, and (b) PSNR values averaged to various epochs for direct comparison with ANCON's projections.



**Fig. 10.** Processed frames at destination during playback. ANCON could successfully predict playback quality at: (a) Frame-75, when MOS projection was 4.257 (higher quality), and (b) Frame-261 with slight distortions, when MOS projection was 3.25 (mediocre quality).

epoch which has a MOS of 3.250206. As expected, there is a distinct difference in the quality of the video as predicted by the ANCON.

**8. Overlay stability**

Streaming monitorables on an overlay plane orthogonal to the data-path is emerging as a promising way to monitor, detect and isolate network impairments and outages. However, keeping the overlay stable and efficient poses acute challenges, since the overlay network itself shall not be monitored for outages. A linear mapping of exporter elements onto collectors often leads to a sub-optimum resource allocation, frequently saturating a few collectors leading to gaps in monitored data. Such gaps could prove costly for service providers, and a most common reaction to this is expensive hardware upgrades. In times of such outages, however, it is observed that not all collectors are saturated. More often than not, balancing the load and efficiently distributing available resources was all that was needed to prevent saturation related outages, and increase overall capacity of the overlay. We discuss ways to construct a correct, robust, and resilient overlay that can function autonomously given the constraints of a typical real world access network.

Serious questions on the feasibility of such an architecture arise:

- How scalable is the architecture with number of data-flows?
- What is the impact of increasing number of exporters on monitoring?
- What is the ideal number of collectors for a given set-up?
- What is the service level of this architecture given a blocking probability?
- And finally, how efficiently is traffic within the overlay engineered?

**8.1. Load allocation and deallocation**

We begin with the load allocation and deallocation algorithm that polices traffic within the overlay. The resources in an access network are typically governed by a central entity popularly known as the *resource manager* (RM). As a new flow is initiated from the source, the RM makes provisions of bandwidth and other resources for the new flow. Likewise, we vest the load allocation and deallocation to a centralized entity (ANCON), which initiates the load allocation algorithm once a new flow is introduced in the network by RM. The load allocation algorithm specifies which exporter shall stream statistics about this new flow to which avail-

able collector. This allocation is static and the mapping holds until the flow is deallocated from the network.

We abstract the loose bunch of exporters as a set  $E$ , and the collecting bunch of collector nodes as a set  $S$  (abstraction in Fig. 11). The exporter pool shows a variation in number captured by the following equation:

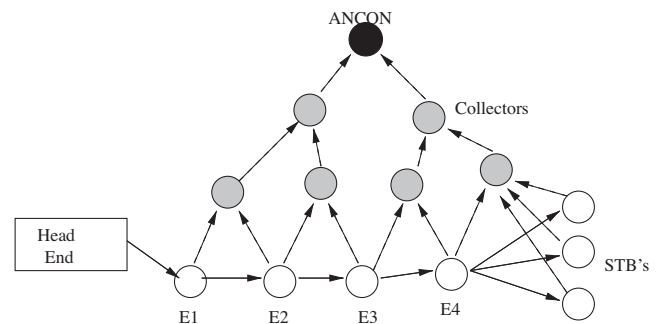
$$E = E + s_i$$

which states that with an inception of a new flow, a new set-top-box ( $s_i$ ) is initiated as an exporter, and is subsequently added to the list  $E$ . Similarly, the set  $S$  denotes the set of all collector nodes in the topology. The elements of set  $S$  make up two other sets,  $S_{saturated}$  and  $S_{candidate}$ , the set of collectors that are saturated and the set of collectors which are potential candidates for traffic mapping. Consequently, the following holds:

$$S = S_{saturated} + S_{candidate}$$

The set  $S_{candidate}$  is equivalent to the set  $S$  when the network boots up, since no collector is saturated. The RM initiates requests for new flows, and the ANCON fires the mapping algorithm. The exporter list is updated to reflect the set-top-box associated with the new flow.

The task of the allocation algorithm is to arrive at a mapping between exporters and collectors for every new flow. The algorithm scans every exporter, and determines a collector for the given exporter. The list of collectors are scanned to ensure that a collector has minimum requirements for bandwidth and CPU for a maximum of  $p$  parameters. Note that, to be selected, *all* three axes must be resolved, while saturation of *any* one moves it to the saturation list. In case the loading is heavy and no suitable collectors are found, the algorithm is recursively run for reduced number of parameters until the number reaches  $p_{min}$ . For each selected



**Fig. 11.** An abstraction of the topology.

$N$	Number of data flows in the data path
$S$	Set which contains a list of all collectors
$S_{sat}$	Set of collectors that are saturated
$S_{can}$	Set of collectors that are candidates
$E$	Set of active exporters
$e$	Number of data flow exporters (pre-E-QAM, static)
$c_k$	The $k$ th collector, from the superset $S$
$p$	Number of parameters per flow
$p_{min}$	Min parameters required, $ p  -  p_{min}  > 0$
$n_i$	Ingress flows at collector node $i$ , $i \leq k$
$e_{CPU}$	% of CPU at exporters for extracting parameters
$b_{CPU}$	% of CPU at STB for extracting parameters
$s_{CPU}$	% of CPU at collector for processing <i>one</i> flow
$b_{uj}$	Bandwidth required for streaming $p$ params
$B_{ui}$	Uplink bandwidth (egress) available at Collector $i$
$b_{di}$	Downlink (ingress) bandwidth for $p$ params at node $i$
$B_{di}$	Downlink (ingress) bandwidth available at collector $i$

Fig. 12. Tabular column of parameters used in modeling.

collector, the status is updated in terms of new bandwidth and CPU availability. The set of variables used is tabulated in Fig. 12. The allocation algorithms is shown as pseudo-code in Fig. 13.

We designed a discrete event system simulator in Java which closely models an access network as well as the collector overlay. Our model of simulation works as follows. Flows are introduced to the source at the access network, and the flow arrival is modeled as a Poisson process. As a flow is admitted into the access network (which in conventional networks is done by the RM), exporters are configured to stream statistics for the given flow in question. The ANCON, in turn, begins the mapping algorithm for the given flow. In particular, it scans the list of candidate collectors to find a suitable mapping for all exporters to suitable collectors for that given flow.

### 8.2. Effect of increasing exporters

As the number of active exporters increase, we seek to understand packet delivery performance of the overlay. We simulate the case of 25 exporters, and 1000 active flows. Each flow generates a monitorable every 5 s. In other words, this means that each exporter injects a monitorable every 5 ms into the overlay. If packets are sent to ANCON faster than it can receive and process, the input buffer gets saturated. Ingress traffic beyond this point is tail dropped, until the buffer begins to free. We measure performance in terms of delivery ratio. This is defined as the ratio of the number of packets received and processed at ANCON, to the total number of monitorables generated at *all* the exporters for all the flows through the access network. As can be seen (Fig. 14), the delivery ratio (DR) drops with an increase in the number of active exporters. The nature of the drop in DR values is also noteworthy: while the drop is steep with an initial increase in exporters, the value begins to stabilize after a certain point in time. Also shown in the figure are the effects of varying the ANCON buffer size. The input queue size was varied from a capacity of 50 packets to a 100 packets in steps of 25. The plot shows two things: (i) the nature of DR variation remains the same, and (ii) the values of DR for various buffer sizes converge.

### 8.3. Role of collectors

Collectors are an intermediate hop between the exporters and the ANCON. The role of collectors as an intermediate is hence two-

### ALGORITHM FOR FLOW MAPPING

```

STEP 1. Resource manager grants requests for data flow  $N + 1$ 
STEP 2. ANCON initiates collector node mapping
STEP 3. Initialize export list  $E \leftarrow e + set - top_{new}$ 
STEP 4. Identify candidate collector for  $p_i$  parameters
for  $k = 1$  to  $|E|$  do
  select  $e_k$  from  $E$ 
  for  $j = 1$  to  $|S_{candidate}|$  do
    Select candidate collector, such that
     $B_{uj} > \frac{p_i}{t}$  and  $B_{dj} > \frac{p_i}{t}$  and  $E_{k,CPU} > e_{k,CPU}$ 
    if Candidate found then
      Map  $E_k \rightarrow c_k$ 
      Update  $S_{can}$ , do
         $B_{uj} \leftarrow B_{uj} - b_{uj}$ 
         $B_{dj} \leftarrow B_{dj} - b_{dj}$ 
         $E_{k,CPU} \leftarrow E_{k,CPU} - e_{k,CPU}$ 
        if  $B_{uj} < b_u$  or  $B_{dj} < b_d$  or  $E_{j,CPU} < e_{CPU}$  then
          move  $c_k \rightarrow S_{saturated}$ 
        end if
      end if
    else
      Decrement parameter:  $p_i \leftarrow p_i - 1$ 
      if  $p_i < p_{min}$  then
        Throw No collector match exception
      else
        Go back to STEP 4, with  $p_i$  as input.
      end if
    end if
  end for
end for

```

Fig. 13. Algorithm for flow allocation, which runs in  $O(n^2)$ .

fold: (i) they can quantitatively aggregate traffic towards ANCON by suppressing self-similar messages (for example, if an exporter reports the same statistic consecutively, the collector might decide to send just one packet by suppressing redundant ones), and (ii) collectors can perform qualitative suppression: by extracting various parameters from the packets, like delay, loss and jitter, they can compute interpretative health scores.

A natural question with such a design is: how many collectors are ideally required for a given set of exporters? In our experiment, we chose 50 exporters, 1000 flows and a monitoring rate of 1 packet every 5 s per flow. The variation in delivery ratio with an increasing number of collectors (Fig. 15(a)) interesting. Delivery ratio improves with rising number of collectors to a maxima, and

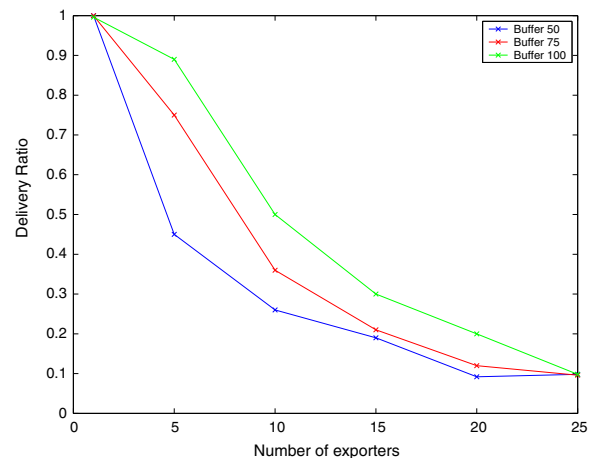
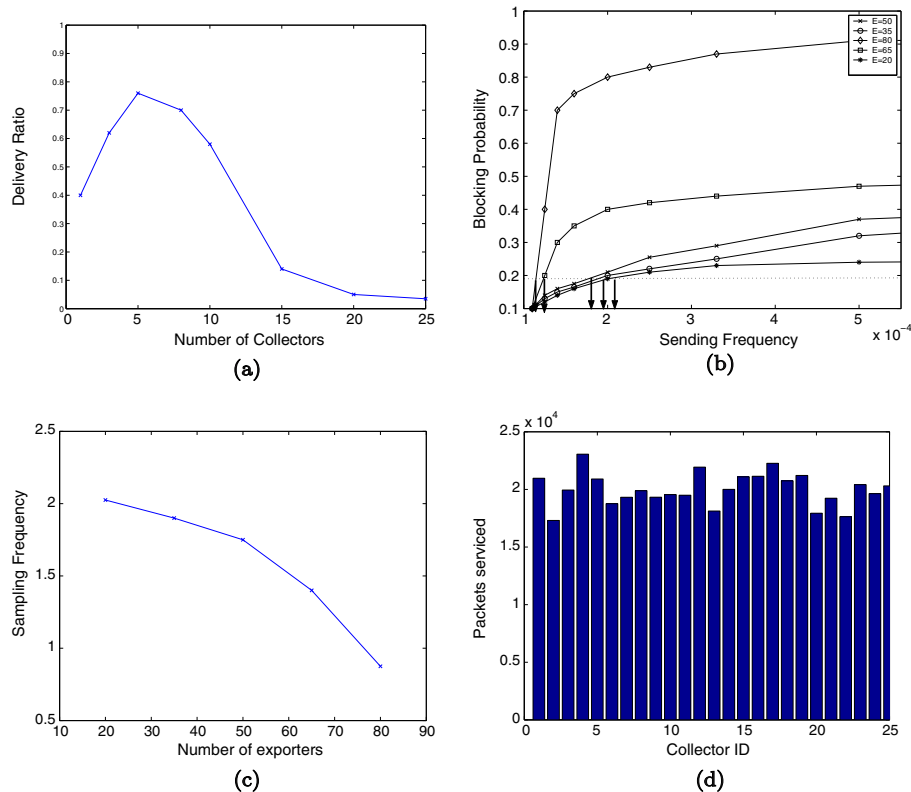


Fig. 14. Delivery ratio versus number of exporters, for a given set of 10 collectors.



**Fig. 15.** (a) Delivery ratio versus number of collectors, for a given set of 50 exporters; (b) Blocking probability against source rate and number of exporters; (c) Sampling rate against number of active exporters, for a blocking probability of less than 0.2; (d) Distribution of number of packets serviced by a collector, for a case of 25 collectors.

then drops in a non-increasing fashion. This clearly establishes that there is a particular number of collectors that are indeed optimum for a given set of exporters. Upon extensive investigation with various other values, we offer this following guideline as a simple rule of thumb. The optimum number of collectors is roughly one-tenth the number of active exporters within an access network.

#### 8.4. Monitoring gaps: specifying minimum requirements

Blocking probability is defined as the probability that the demanded service is blocked, averaged over a sufficiently long interval of time. In our case, it is the probability that a monitored information is lost in the overlay before it makes it to ANCON. What is ideally required is a form of quality assurance: given a requirement that the blocking probability be less than a certain value, what is the right number of exporters for such a set-up.

To understand the nature of sampling on the number of exporters for a given threshold of, we first understand the behavior of blocking itself for a given streaming rate. We conducted an experiment where we measured the blocking probability for a rising rate of monitoring at exporters. We choose a sufficiently large number of packets (5,00,000) over a 6 h long simulation period to understand the nature of blocking within the overlay. We run each experiment for different set of collectors at least 10 times to smoothen out variations in performance. Shown in Fig. 15(b), blocking probability for any given set of exporters increases sharply with a slight increase in streaming rate. The value, however, stabilizes with further increase in the streaming rate, almost hitting a saturation beyond a certain point.

Given such a behavior, we now apply quality assurance. We ask: what is the sampling frequency at ANCON if a service provider demands a blocking probability of less than 0.2 (i.e., at least 80% of monitorables should be received and processed at ANCON). We

draw a horizontal line across the plot in Fig. 15(b) at around the value of 0.2 for blocking probability, and subsequently measure the correct sampling frequency given the maximum blocking probability.

The sampling frequency for such a case for various values of exporters is as shown in Fig. 15(c). The curve exhibits a parabolic trend, with the rate falling more steeply beyond a certain point (in this case, for  $E > 50$ ). This study implies a few properties of this overlay: (i) it is possible to specify design with maximum blocking probability in mind, since there is an optimum number of exporters all blocking probabilities, (ii) there is an optimum number of collectors given a set of exporters; and, (iii) buffer sizes, and increasing them to improve capacity, does not necessarily work very well.

#### 8.5. Comparing load allocation to randomized allocation

Having studied the nature and dynamics of the overlay, we finally turn our attention to ensuring a balanced load distribution given the following: (i) a maximum blocking probability demanded, or in other words, a minimum quality assurance in monitoring, (ii) a correct set of exporters for such a case, and (iii) the correct set of collectors for this set-up. Having established the first three requirements in the previous sections, we now run our load-balancing algorithm to study the variations in load distribution across various collectors for a given set of 250 exporters, 25 collectors, a monitoring rate of 1 packet every 5 s per flow, for a set of 1000 flows. The simulation runs reported here run for a total of approximately 5,00,000 packets, for a simulation time of 6 h. For 25 collectors, this means that a perfectly distributed load would result in approximately 20,000 packets serviced per collector. The plot for this experiment is as shown in Fig. 15(d). We run our load distribution algorithm as flows arrive and depart (Fig. 13), and measure the

actual number of packets serviced by every collector. As can be seen, the variance of load distribution is nearly minimum. While it is theoretically impossible to achieve a perfectly balanced system, our algorithm outperforms other schemes (like random allocation or a round-robin allocation), both in terms of load distribution as well as maximizing delivery ratio.

## 9. Conclusions

Given the stringent delivery bounds of streaming applications, there is a need to *arrest* network induced errors as they occur. This would naturally be the fastest way to isolate, react, and adapt to changing network conditions. In this paper, we successfully design and evaluate an architecture of a hierarchical overlay of “collector” nodes that can gather and disseminate network statistics. The collectors are decentralized, and can monitor the health of local segments. The statistics are routed to a root node, called ANCON, that correlates and computes various properties of the video flow.

Using such a network of collectors, we are able to: (i) isolate network induced causes of a degradation; (ii) isolate segments/subnets in the network that caused the impairment, (iii) provision local feedbacks, and inform source in real time of network statistics, (iv) create an instrumentation layer above the data plane that can correlate network statistics and compute MOS, (v) export local MOS to service providers instead of overwhelming them with network statistics, to better facilitate root cause analysis, (vi) show that quality can be scored continuously along the data path, and, (vii) predict the quality of a video much before the playout actually occurs at destination. Our evaluation of the design with extensive experiments establishes feasibility of the architecture. This study also identifies key knobs that should be exported to service providers to help them self tune the architecture, as well as key parameters that a collector nodes should monitor.

## References

- [1] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, A. Terzis, An overlay architecture for high quality VoIP streams, *IEEE Trans. Multimedia* 8 (6) (2006) 1250–1262.
- [2] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, R. Morris, Resilient overlay networks, in: *Proc. 18th ACM SOSP*, 2001.
- [3] P. Calhoun, DIAMETER Base Protocol, IETF FRC 3588, 2003.
- [4] D. Chandler, S.S. Hemami, Effects of natural images on the detectability of simple and compound wavelet subband quantization distortion, *J. Opt. Soc. Am.* 20 (7) (2003) 1164–1180.
- [5] P. Calhoun, DIAMETER Base Protocol, IETF FRC 3588, 2003.
- [6] Y.C. Huang, C.S. Lu, H.K. Wu, JitterPath: probing noise resilient one-way delay jitter-based available bandwidth estimation, *IEEE Trans. Multimedia* 9 (4) (2007) 798–812.
- [7] S. Kanumuri, P.C. Cosman, A.R. Reibman, V.A. Vaishampayan, Modeling packet-loss visibility in MPEG-2 video, *IEEE Trans. Multimedia* 8 (2006) 341–355.
- [8] J. Klaue, B. Rathke, A. Wolisz, EvalVid-A framework for video transmission and quality evaluation, in: *Proceedings of the 13th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, IL, September 2003.
- [9] R. Koenen, Overview of the MPEG-4 Standard, ISO IEC JTCL/SC29/WG11 M4030, 2001.
- [10] N. Miller, P. Steenkiste, Collecting network status information for network-aware applications, *IEEE Infocom* (2000).
- [11] S. Mohamed, G. Rubino, A study of realtime packet video quality using random neural networks, *IEEE Trans. Circ. Syst. Video Technol.* 12 (12) (2002) 1071–1083.
- [12] M.S. Moore, J.M. Foley, S.K. Mitra, Detectability and annoyance value of MPEG-2 artifacts inserted into uncompressed video sequences, in: *Proceedings of SPIE, Human Vision and Electronic Imaging V*, vol. 3959, San Jose, CA, January 2000, pp. 99–110.
- [13] M.S. Moore, S.K. Mitra, J.M. Foley, Defect visibility and content importance implications for the design of an objective video fidelity metric, *Proc. IEEE ICIP* 3 (2002) 45–48.
- [14] J. Postel, Transmission Control Protocol, IETF RFC 793, September 1981.
- [15] M.G. Ramos, S.S. Hemami, Suprathreshold wavelet co-efficient quantization in complex stimuli: psychophysical evaluation and analysis, *J. Opt. Soc. Am.* 20 (7) (2003) 1164–1180.
- [16] C. Rigney, S. Willens, A. Rubens, W. Simpson, Remote Authentication Dial In User Services (RADIUS), IETF RFC 2865, 2000.
- [17] R. Stewart, Stream Control Transmission Protocol, IETF RFC 2960, 2000.
- [18] M. Venkataraman, S. Sengupta, M. Chatterjee, R. Neogi, Towards a Video-QoE definition in converged networks, in: *Second International Conference on Digital Telecommunications, ICDT*, Silicon Valley, CA, July 2007.
- [19] A. Watson, M.A. Sasse, Measuring perceived quality of speech and video in multimedia conferencing applications, *Proc. ACM Int. Conf. Multimedia* (1998) 55–60.
- [20] X. Yang, C. Zhu, Z.G. Li, X. Lin, N. Ling, An unequal packet loss resilience scheme for video over the Internet, *IEEE Trans. Multimedia* 7 (4) (2005) 753–765.
- [21] P. Zhu, W. Zhen, C. Li, Joint design of source rate control and QoS-aware congestion control for video streaming over the internet, *IEEE Trans. Multimedia* 9 (2) (2007) 366–376.
- [22] ITU-T Recommendation G.107, The E-Model, a computational model for use in transmission planning, December 1998, Subjective video quality assessment methods for multimedia applications, September 1999.
- [23] ITU-R BT.500-8, Methodology for the subjective assessment of the quality of television pictures, 1998.
- [24] ITU-T Recommendation P.910, Subjective video quality assessment methods for multimedia applications, September 1999.
- [25] The Network Simulator v2.30. <<http://www.isi.edu/nsnam/ns>>.
- [26] Internet Protocol Detail Record. <<http://www.ipdr.org>>.
- [27] The IPDR/SP (Streaming Protocol) Specifications. <<http://ipdr.org/download-docs/protocol.html>>.
- [28] Video Evaluation Toolkit, Moscow State University, Moscow, Russia. <<http://graphics.cs.msu.ru/>>. Available for free download.