TREND: A DYNAMIC BANDWIDTH ESTIMATION AND ADAPTATION ALGORITHM FOR REAL-TIME VIDEO CALLING

BARZUZA Tamar, BEN ZEDEFF Sagee, MODAI Ori,
VAINBRAND Leonid, WIENER Yair, YELLIN Einat

RADVISION, Tel Aviv, Israel

ABSTRACT

Video calling over the internet is becoming more and more popular in recent years. The quality of video in such applications is a major contributor to the overall user experience during the call. As received video quality degrades significantly when the network path used is either under-utilized or over-utilized, a method for estimating network bandwidth availability for adapting video bit rate in video calls is highly desirable. This paper presents TREND, a novel technique for dynamic bandwidth estimation and adaptation designed for real-time, interactive video calling. By detecting the delay in the received video frames, TREND estimates available network bandwidth in the path and adapts the encoded bitrate sent to best fit it before packets are lost. TREND uses only standard protocols for delay detection and flow control, and can therefore be easily and effectively used with traditional visual communications systems. After testing TREND under various use cases we have found that it efficiently estimates available bandwidth while outperforming competing tools in terms of detection time and accuracy. By implementing TREND in its video conferencing solutions, RADVISION was able to show a superior quality of experience for its video calls over other solutions available today.

Index Terms— Bandwidth, Bandwidth Estimation, Available Bandwidth, Video Calling, Networks, Delay Detection, Bandwidth Adaptation, Video Conferencing, Network Congestion, Flow Control

1. INTRODUCTION

Video calling has become one of the Internet's fastest growing phenomenons. Millions have discovered that reaching out and seeing someone is the next best thing to being there. Implemented in many products and form factors, from videophones to instant messaging and from video conferencing endpoints to mobile handsets, it has become a significant communications solution.

Video calling is an interactive means of communication involving the transport of high volume multimedia data in real-time, making minimal delay critical. Carried out over both managed and unmanaged networks, QoS is not always guaranteed. Most video calling is implemented over the public Internet, where network congestion often leads to transmission errors, such as packet loss, delay and jitter.

Network congestion arises at overloaded network paths where the bandwidth being sent exceeds the path's maximum available bandwidth. This results in blocks of data, or packets, being dropped at bottleneck routers of the path. Packet loss is usually perceived by the receiver as severe video artifacts, and hence can significantly affect quality of experience.

Congestion control algorithms, found in most video calling products, deal with such packet loss by lowering the bitrates, e.g. Google Talk's Rate Control mechanism [1] based on TCP Friendly Rate Control (TFRC) [2]. However, the packets that are lost before the bitrate reduction already introduce a dramatic drop in video quality. For this reason, the majority of these algorithms involve some mechanism for re-transmission of lost packets, e.g. Sony's Real-Time ARQ (Automatic Repeat reQuest) [3]. However retransmission itself degrades QoS in interactive calls due to the latency caused by sending feedback from receiver to transmitter and then waiting for the retransmitted data.

In short, one of the main challenges in real-time, interactive video calling is adapting the bitrates to the path without introducing packet loss. Obviously, the lion's share of consumed network bandwidth in such calls is the video data itself and therefore significant changes to the overall bandwidth used can only be done by adapting the bit rate produced by the video encoder.

Our proposed algorithm, TREND, combines two key functionalities:

1. Delay Detection: Over-utilization of bandwidth is detected based on real-time monitoring of the delay related to received video frames.
2. Bandwidth Adaptation: Upon delay detection, available bandwidth is estimated and the actual outgoing bitrate is adapted to best fit the path characteristics.

TREND is designed for real-time video encoders and can be used in interactive, real-time video calling. Since it is based only on standard feedback coming from either the network or the receiver, it can be integrated into any standard video calling systems, which makes it unique. TREND works with any video encoding format, without any need for multiple layers, temporal scalability or fine grain scalability (FGS) that can be found in scalable video coding (SVC) [4] based methods.

The paper is organized as follows: Section 2 discusses the characteristics of real-time, interactive video and then proceeds to describe and evaluate related work in view of these characteristics. Section 3 outlines the proposed TREND algorithm for bandwidth estimation and adaptation. Section 4 depicts some algorithm optimizations and describes measures used to evaluate the algorithm's quality. Section 5 presents actual results in various real-life test conditions and compares them to related work. Section 6 serves as a conclusion and discusses future work.

2. RELATED WORK

Before discussing related work in terms of bandwidth estimation and adaptation, we will briefly present the characteristics of real-time, interactive video. Understanding these features is essential for the comprehension of TREND.

2.1. Real-Time Interactive Video Characteristics

There are distinct characteristics that differentiate real-time interactive video transmission from other means of data transport:

1. Frame boundaries – a video stream is made up of frames, each comprised of a set of packets that are transmitted at approximately the same time to the network and carry the same (RTP) timestamp irrespective of actual sending time. This requires monitoring packets not on a packet level, but on a frame level, aggregating the frame packets data and adjusting the data to fluctuations that may occur.
2. Hierarchical nature – encoded video is not uniform by design. For instance, I (Intra) frames are generally larger than P (Inter) frames. Moreover, video flow usually has a strongly obeyed temporal structure (GOP), and different frames in a GOP have different significance in the stream. Therefore the packet structure of all frames is not always identical.
3. High bitrate fluctuations – Intra frames, which are temporally independent, are typically used when there are modifications in the video stream, such change of scene or resolution. Intra frames are also used to deal with packet loss. As Intra frames consume more bits than other frame types, the introduction of such unexpected frames may lead to severe fluctuations in the rate control mechanisms of the encoder. This has to be modeled and compensated for a bandwidth estimation algorithm.

In additions to the above, video calling is very standardized in nature. It relies completely on existing protocols that strictly define the communication between sender and receiver. For an algorithm to be vendor and protocol agnostic it has to comply with all existing relevant standards. In particular, it needs to:

1. Provide feedback to the sender using standard flow control commands, as defined by the H.245 control channel protocol [7] (used with H.323 [8] and H.324 [9]) and/or RFC5104 [10] (used with SIP [11]). These feedback commands, sent from video receiver to video sender, dictate the desired video bitrate to be sent.
2. Rely on sequencing sent via RFC 3550 (RTCP) [6] and local system clock only.

These standard protocols define the communication means between sender and receiver and constitute all the exchanged traffic other than the bit-stream packets transmitted over the path. In principle, it is the receiver's responsibility to control the flow of packets towards it; the sender does not have any control mechanism for this. Although the sender provides the receiver with some information on the transmit time of the video frames via timestamps, there is no straightforward method to synchronize the sender and receiver clocks.

2.2. Bandwidth Estimation and Adaptation Algorithms

Common research approaches for bandwidth estimation and adaptation generally fall into three categories: 1. Algorithms that are designed for specific networks, usually with guaranteed QoS, 2. Algorithms that utilize probing packets with pre-determined spacing, and 3. Algorithms targeting video streaming, where a client-server model is assumed. We will provide hereunder several examples of each approach, and highlight the difference and novelty of TREND, our proposed algorithm.

2.2.1. Methods designed for specific networks

Algorithms that are designed for networks with guaranteed QoS, such as Ethernet Passive Optical Networks (EPON), are generally difficult to adapt for real-time interactive streaming, as they do not support the video characteristics mentioned in 2.1 and are tailored to the specific network model. Furthermore, these algorithms, such as the ones described in [12] and [13], assume a single central entity (OLT) that controls and distributes the bandwidth transmission time slots according to periodic report messages and BW grants.

QoS based algorithms also rely on queue feedback controls from the network which is not applicable for most video networks.

Algorithms that are based on TCP Friendly Rate Control (TFRC) such as [14, 15] can sometimes be adapted to real-time video calling. However, TFRC based algorithms reduce bitrate aggressively upon congestion event, often causing severe bitrate fluctuations that are well known for degrading user's quality of experience dramatically, especially in high resolutions.

Contrary to that, TREND algorithm operates in unreliable environments using the public Internet and applying UDP-based, unreliable transmission.

2.2.2. Probing Packets with Predetermined Spacing

Probing algorithms such as [16] and [17] transmit constant sized packets with different time spacing models. At the receiver side they monitor the incoming delay and apply mathematical reasoning to determine whether the available bandwidth was exceeded. PathChirp [17], for example, transmits a sequences of constant sized packets with exponentially reduced spacing, i.e., if packet p_k is

transmitted $\alpha^{k-1}T$ milliseconds after packet p_{k-1}, then packet p_{k+1} is transmitted $\alpha^{k}T$ milliseconds after p_k, with $\alpha \in [0,1]$. The incoming delay is monitored at the receiver, which indicates over-utilization of the available bandwidth once it detects a significant increase in the delay. The formal delay detection reasoning of PathChirp is quoted in the Results section (section 5) of this manuscript. Informally, a sequence of packets is detected by PathChirp as bearing a significant delay increase, if the general tendency of their incoming delay is increasing. In other words, it is still possible for several packets in the sequence to introduce temporal decreases in the delay, as long as the decreases are not too significant in magnitude.

Real-time video characteristics, such as frame hierarchy and bitrate fluctuation, rule out the possibility of using the transmission methodologies presented in such algorithms, as they are too firm and synthetic to be adjusted to video transmission.

However, the mathematical methodology of determining bandwidth over-utilization from incoming delay in these algorithms can be adapted to video. In Section 5 we compare TREND's delay analysis to that of the adapted PathChirp algorithm.

2.2.3. Algorithms for Video Streaming

As video streaming becomes more popular, bandwidth estimation methods targeting video streaming applications are becoming a very high profile research topic. These methods usually involve an end-to-end solution, comprising of a server and a client, where video is streamed one-way from server to client. This is the case of the Video Transport Protocol (VTP) [18], the Streaming and Congestion Control using Scalable Video Coding [19] and the H.264/AVC Rate Adaptation for Internet Streaming algorithm [20].

All of these algorithms include a set of pre-encoded content in a number of pre-defined, different bitrates, either layered or using a single layer, that the server (the sender) can choose from based on bandwidth estimation. Therefore, they are not designed to estimate and adapt exactly to the available bandwidth, but rather to identify which of the available pre-encoded streams is most suitable. Further constraints are made by these algorithms, which limit their applicability for specific settings: VTP is mainly focused on MPEG-4; the Streaming and Congestion Control algorithm is based on a strict GOP structure with key pictures in fixed locations; and the H.264/AVC Rate Adaptation algorithm assumes temporal scalability.

Unlike these algorithms, TREND utilizes any amount of available bandwidth up to 90% and does not require any pre-encoded content. It can work with any video compression standard, be it layered or single-layered and does not imply any restrictions on GOP structure.

The Adaptive Congestion Control Scheme for Real Time Packet Video Transport [21] is more general in nature, but is based on a feedback channel from the network (for instance, network switches) to modulate the sender's rate.

Furthermore, it assumes that packets of each frame are sent in regularly spaced intervals to the network, which is not necessarily the case in real-time video communications. Additionally, TREND requires no network feedback and relies on existing standard feedback that is part of any interactive, bidirectional, real-time communication protocol. From our experience with interactive real-time communication, any assumption on the packet sending intervals can be risky; therefore our algorithm does not take the inter-packet delay within frames into consideration.

3. THE TREND ALGORITHM

In this section we shall describe our proposed bandwidth estimation and adaptation algorithm, TREND, for live video communication. TREND converges to the available bandwidth in the path during call setup, and dynamically adjusts to mid-call changes in the available bandwidth. Its logic and decisions are executed only at the receiver side. We named this algorithm TREND, as it searches for significant increasing and/or decreasing trends in the delay of incoming video frames.

Next we describe the TREND algorithm in detail beginning with the delay detection mechanism and proceeding to the algorithm flow.

3.1. Delay Detection

The TREND algorithm detects bandwidth over-utilization by monitoring the incoming video packets delay at the receiver and searching for significant increasing and/or decreasing trends in the delay.

First we will specify the motivation for delay based bandwidth estimation, and then describe the delay modeling and detection process.

3.1.1. Delay Analysis

Understanding the effects of bandwidth over-utilization and how it is experienced at the receiver side is essential for designing a bandwidth estimation tool with best possible quality of experience. When the available bandwidth is over-utilized, packets enter the path at a rate faster than the network can process them. In this case the packets are usually stored by network router(s) in a buffer until processing is complete. This introduces a delay in packet arrival at the receiver, known as queuing delay. The maximum queuing delay is proportional to the router's buffer size and to the extent of bandwidth over-utilization. Packet loss occurs when the buffer overflows, i.e., when the buffer has no room to store further arriving packets. At routers that adopt the common RED policy [22], packet loss can occur even earlier, as these routers randomly drop packets to reduce TCP traffic bandwidth when queuing delay is accumulated.

Figure 1 demonstrates the delay and packet loss patterns obtained with nearly 50% over-utilization of the available bandwidth. In this experiment, a 2Mbps video was transmitted over an MPLS line with a 1.4Mpbs limit. The light line represents the delay in frame arrival at the receiver side. The dark dots represent packet loss events, with every dot corresponding to loss of one or more consecutive

packets. Note that the delay accumulates during 19 video frames (2/3 of a second) before the first packet is lost.
Similar experiments substantiate the expected results: the receiver perceives accumulated delay for a significant period of time before the first packet is lost. As packet loss tremendously reduces video quality [5], identifying bandwidth over-utilization at the delay accumulation stage before packets are lost and reducing the transmitted bitrate accordingly is highly desirable.

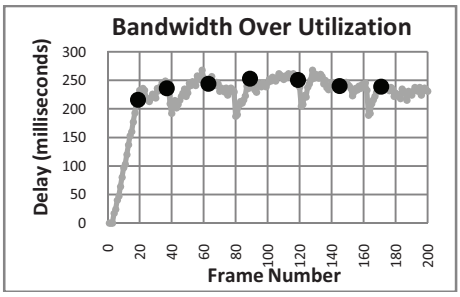


Fig. 1 The delay and packet losses as they are perceived at the receiver.

Computing the end-to-end delay of an incoming packet is not trivial since the send time and the receive time originate from different machines. The Network Time Protocol (NTP) [23] can be used to synchronize the sender and receiver machine clocks, but may entail additional implementation efforts, deployment complexity and messaging overhead. Bandwidth estimation algorithms often consider the delta delay between consecutive packets rather than the actual end-to-end delay [16, 17]. In this paper we present a new delay measure based on delay history in reference points. Our delay measure is computed per video frame, is simple to implement and requires no overheads. We found this approach highly effective for detecting bandwidth over-utilization.

3.1.2. Delay Modeling

TREND models the one-way delay between the sender and the receiver by comparing the receive time and the RTP timestamp of every incoming packet, as demonstrated below. Since TREND delay detection is based on several dozen consecutive packets, clock drifts are irrelevant. Since the delay is measured in milliseconds, clock drifts may only cause isolated increases or decreases of a single millisecond. Intra frames on the other hand do affect the TREND algorithm, due to Intra frame encoder rate control compensation. Most of the encoders consume significant bandwidth for encoding an Intra frame, and compensate in the subsequent Inter frame and during the interval between these frames. The TREND algorithm resets the delay modeling parameters upon Intra frame events, formally known as reference modification. Formal TREND delay modeling is defined as follows:

Let $\{p_1, p_2, p_3, \dots\}$ denote the sequence of incoming packets. Denote by Rx_i the reception time of packet i (in millisecond precision) and by TS_i the RTP timestamp of packet i. Let $\{ref_1, ref_2, \dots\} \subseteq \{1,2,3, \dots\}$ be an ordered list of reference packet indices, and define

$$Ref(i) = ref_j \ \ s.t. \ ref_j \leq i \ and \ ref_{j+1} > i.$$

$Ref(i)$ is the latest reference packet index before i. Then the effective packet delay is computed as follows:

$$D_i = Rx_i - (Rx_{Ref(i)} + \frac{(TS_i - TS_{Ref(i)})}{90}).$$

The delay measure D_i is calculated as the difference between the receive time and the timestamp of packet i, normalized by the delay of a previous reference packet.

The list of reference packet indices is obtained as follows: Let $\{f_1, f_2, f_3, \dots\}$ denote the sequence of incoming frames and $P(f_i)$ denote the group of indices of the packets composing frame f_i. In other words $f_i = \bigcup_{l \in P(f_i)} p_l$. Note that $P(f_i)$ consists of consecutive indices and that $p_{minP(f_i)}$ is the first packet of frame f_i. Label $C(f_i) = I$ if f_i is an Intra frame and $C(f_i) = P$ if f_i is Inter frame. Then the list of reference packet indices is defined as follows:

$$\{minP(f_k) | C(f_k) = P, C(f_{k-1}) = P, C(f_{k-2}) = I\}.$$

In other words, the reference packet is updated according to the second P-frame after every I-frame. We chose the second P-frame, as usually the encoder rate control mechanism compensates for the excessive bandwidth allocated for the Intra frame in the subsequent P frame and during the interval between these frames.

Next we smooth the delay measure D_i into a per-frame delay estimator S that will be used for bandwidth estimation. The delay estimator $S(f_i)$ of frame f_i is computed by averaging the delay of all f_i packets:

$$D(f_i) = avg_{j \in P(f_j)} D_i$$

and deducing an exponential moving average as follows:

$$S(f_i) = \alpha * D(f_i) + (1 - \alpha) * S(f_{i-1}) \quad \text{for } \alpha \in [0,1]$$

3.1.3. Delay Detection

Delay detection is in the core of the TREND algorithm. Our goal is to identify two possible events:

1. *Delay UP* – A significant delay increase, indicating over-utilization of the available bandwidth
2. *Delay DOWN* – A significant delay decrease, indicating that over-utilization was fixed to match available bandwidth limitations.

Delay detection is performed over a fixed-sized window of recently received frames. The window slides with every receipt of a new frame.

Let K define the number of frames in the window. Then the delay events are defined as follows:

UP Event

For every f_i invoke an *UP event* if $\forall (i - K + 2) \leq l \leq i$,

$$S(f_l) > S(f_{l-1}).$$

In other words, S is strictly increasing over K consecutive frames once reaching f_i.

DOWN Event

DOWN events are detected only after an *UP event* was detected. Suppose that we detected *UP event* at frame f_k. Set $Step = S(f_k)$, indicating the highest point in the initial

delay increase. Invoke the *DOWN event* for frame f_i if $\forall(i-K+2) \le l \le i$,

$$S(f_l) < S(f_{l-1}) \ and \ S(f_i) < \sigma * Step,$$

for $\sigma \in [0,1]$. In other words, S is strictly decreasing over K consecutive frames once reaching f_i, and finally drops under σ fraction of the initial step.

Note that different window sizes can be selected for *UP* detection and *DOWN* detection.

3.2. Algorithm Flow

The TREND algorithm has two modes of action:

- *Scan Mode*: gradual attempts to increase utilized bandwidth without exceeding available bandwidth.
- *Steady Mode*: constant rate transmission while monitoring for changes in available bandwidth.

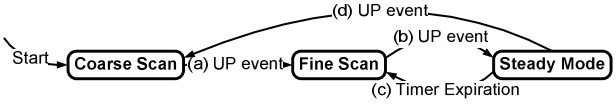


Fig. 2 The TREND algorithm flow

3.2.1. Scan Mode

A scan is a series of increasing bitrate steps. Let B_0 be the initial bitrate of the scan. Denote by b the step size and by T the time interval between steps. Then the transmitted bitrate after nT seconds is $B_0 + nb$. TREND toggles between two scan steps $\bar{\bar{b}} \gg \bar{b}$: a scan with large $\bar{\bar{b}}$ step is called *coarse scan*, and a scan with small $\bar{b}$ step is called *fine scan*.

The scan is controlled by the receiver. The receiver signals the sender to increase the transmitted bitrate according to the scan steps, using standard flow control commands, e.g. H.245 for H.323 or RFC5104 for SIP. The receiver begins with a *coarse scan*, during which it constantly monitors the incoming delay looking for *UP events*. An *UP event* indicates that the available bandwidth barrier was reached by the *coarse scan*; therefore the receiver drops the requested bitrate and enters a *fine scan* (Figure 2(a)). The bitrate is dropped to a safe value from earlier steps of the *coarse scan*. The receiver continues with the *fine scan* until another *UP event* is reached, which causes it to enter the steady mode (Figure 2(b)). The receiver chooses a bitrate for the steady mode based on the incoming bitrate. At the moment of choice, the link is assumed to be utilized completely with no packet losses; therefore the incoming bitrate at the receiver provides a good estimation of the available bandwidth. We call this bitrate the effective bitrate.

Figure 3 demonstrates a scan mode that gradually climbs towards the available bandwidth. Once the available bandwidth is exceeded, an *UP event* is detected and the mode switches to steady. Figure 3.a demonstrates a *fine scan* with initial bitrate $B_0 = 256$Kbps, step size $b = 20$Kbps, and time interval $T = 6$ seconds. The available bandwidth is exceeded around frame 1775, where the sender transmits 575Kbs. When the available bandwidth is exceeded, as Figure 3.b shows, an increasing trend in the delay is observed at the receiver. The delay increase is detected and an *UP event* is invoked. The receiver sends a flow control command to the sender with bitrate 535Kbs and moves to the steady mode.

3.2.2. Steady Mode

The steady mode begins with a flow control command from the receiver specifying the steady bitrate to transmit. The receiver will monitor the incoming delay looking for *UP events*. As long as no *UP event* occurs, there will be no flow control commands from the receiver. If an *UP event* is detected, then the available bandwidth at the link is assumed to be changed and thus adjustment is required. The receiver will instruct to drop the bitrate according to the effective bitrate and initiate a *coarse scan* (Figure 2.d). If no *UP events* are detected, the receiver remains in steady mode until a safe period of time elapses. Then it reenters scan mode, seeking to improve bandwidth utilization (Figure 2.c).

a

b

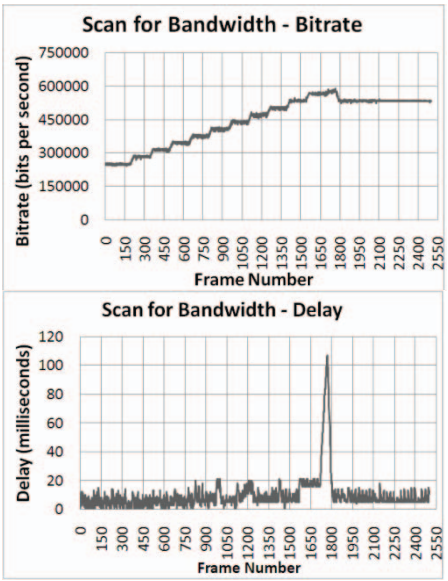


Fig. 3 Convergence to the available bandwidth of 560Kbs.

Figure 4 demonstrates a steady mode that encountered an *UP event* and dropped the bitrate accordingly: The 1st call performs a short scan towards the application limited bitrate of 560Kbs. The 560Kbs destination is available at the link, and the call enters steady mode at around frame 450. At frame 635 a 2nd call joins the same network link and begins a scan process. The new call results in over-utilization of available bandwidth, and consequently the 1st call drops its bitrate (frame 770). After two attempts of the 1st call to improve its bandwidth utilization, the two calls converge (frame 1270): The 1st call converges to 376Kbs and the 2nd call converges to 441Kbs. This test was performed over an MPLS line with significant roundtrip delay of 700-800 milliseconds, showing the stability and practicality of the algorithm.

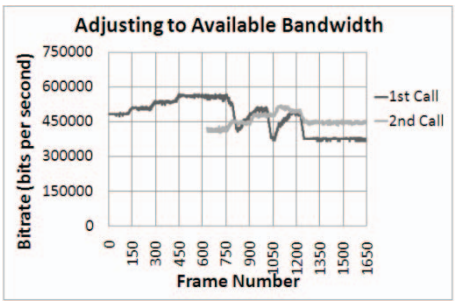


Fig. 4 Adjusting the bitrate to mid-call change in the available bandwidth.

4. ALGORITHM OPTIMIZATION

In this section we will describe various methods used for evaluating TREND's performance and how these methods were used for optimally calibrating the algorithm's parameters.

4.1. Performance Evaluation

The ultimate goal of the TREND algorithm is to improve the quality of experience for call participants by avoiding packet loss, maximizing available bandwidth utilization and reducing delay.

With standard methodology for video quality assessment [5], packet loss, bitrate and delay measures can be quantified into a single score representing the quality of experience of call participants (the lower the score the worse the experience). Being able to analyze performances with a single score is important for algorithm optimization, parameter calibration and comparison with other methods.

Delay detection is the core of the TREND algorithm, thereby critically affecting its performance. Delay detection on its own is a significant target for optimization, parameter calibration and comparison with other related algorithms. Hence, there is great value in representing delay detection with a single score. We will define three delay detection correctness measures and quantify them into a single performance score which will be used to calibrate the delay detection parameters.

The evaluation and calibration processes are based on a large database of tests, which demonstrate the TREND delay detector operation in diverse conditions, including numerous examples of bandwidth over-utilization at different points in time and with different magnitudes.

4.1.1. Correctness Measures

Let N be the number of tests in the database. For every test $1 \le i \le N$, denote by ϑ_i the group of frames that indicate the exact points in time where the available bandwidth was exceeded (up to frame precision), and by φ_i the group of frames that indicate the exact points in time where the bitrate dropped back below the available bandwidth. Denote by $\bar{\vartheta}_i$ the group of frames for which TREND invoked an *UP event*. We define the following correctness measures:

False Positive

We say that frame $\bar{f} \in \bar{\vartheta}_i$ *refers* to $f \in \vartheta_i$ and denote $\bar{f} \to f$ if $\bar{f} > f$ and $\nexists \ddot{f} \in \varphi_i \, s.t. \bar{f} > \ddot{f} > f$, i.e. the available bandwidth was exceeded at frame f and detected at $\bar{f}$, while there was no decrease in bitrate below the available bandwidth in between. An event in $\bar{f} \in \bar{\vartheta}_i$ is considered to be a false positive event, if it does not *refer* to any event from ϑ_i. More formally, the number of false positive events in test i is defined as follows:

$FPu_i = \left|\left\{\bar{f} \in \bar{\vartheta}_i \middle| \nexists f \in \vartheta_i \, s.t. \bar{f} \to f \right\}\right|$.

False Negative

Test i is considered to have a false negative event if an actual bandwidth over-utilization was not detected by an *UP event* for a reasonable amount of time. More formally, the number of false negative events in test i is defined as follows:

$FNu_i = |\{f \in \vartheta_i | \nexists \bar{f} \in \bar{\vartheta}_i \, s.t. \bar{f} \to f \text{ and } (\bar{f} - f) < l\}|$.

Where $0 < l$ indicates the maximal acceptable detection time in frames. After this time, TREND is assumed to have missed the delay event.

Detection Time

Let $\bar{f} \in \bar{\vartheta}_i$ be a detection of an actual over-utilization event $f \in \vartheta_i$, that was detected within the reasonable time limitation defined by l. Then the detection time of $\bar{f}$ is its frame distance from f. Formally:

$$T_i^{\vartheta}(\bar{f}) = \begin{cases} 0 & if \nexists f \in \vartheta_i \, s.t. \bar{f} \to f \, and \, (\bar{f} - f) < l \\ min_{f \in \vartheta_i, \bar{f} \to f \wedge (\bar{f} - f) < l} (\bar{f} - f) & O/W \end{cases}.$$

Define the detection time of test i by $Tu_i(\bar{\vartheta}_i) = \sum_{\bar{f} \in \bar{\vartheta}_i} T_i^{\vartheta}(\bar{f})$, i.e., the sum of all time intervals required for the delay detector to indicate bandwidth over-utilization.

We define FPd_i, FNd_i and Td_i similarly to FPu_i, FNu_i and Tu_i but for *DOWN events*.

4.1.2. Single Penalty Measure

We define a new penalty function that receives the correctness measure values, namely *FPu/d*, *FNu/d* and *Tu/d*, and produces a single penalty score. In this penalty function, packet loss is indicated by FNu_i and FPd_i events, non-utilized bandwidth is indicated by FPu_i and FNd events and detection delay is represented by Td_i and Tu_i.

The penalty score correlates to the quality of experience at the receiver; in other words, the penalty function is influenced by *FPu/d*, *FNu/d* and *Tu/d* according to their affect on the video quality at the receiver. The penalty function can be used to compare two different delay detectors, as long as the link and algorithm flow policy are equivalent (including the state machine, step sizes b, step intervals T, etc.). The penalty function is defined as follows:

For every test $1 \le i \le N$, the penalty measure $\mathrm{P}(i)$ is defined by:

$$\mathrm{P}(i) = \alpha_1 * FPu_i + \alpha_2 * FPd_i + \beta_1 * (1+\rho)^{Tu_i} + \beta_2 * (1+\sigma)^{Td_i} + \gamma_1 * FNu_i + \gamma_2 * FNd_i$$

With $\beta_2 < \beta_1 < \gamma_2 < \alpha_1 < \alpha_2 \ll \gamma_1$ and $\sigma < \rho \ll 1$

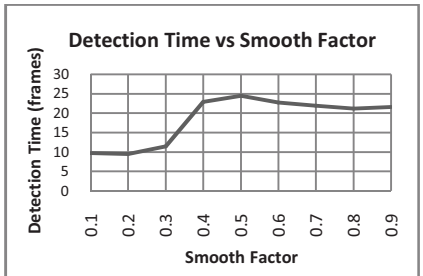
The penalty measure is then used for calibrating TREND parameters.

4.2. Parameter Calibration

The TREND algorithm requires that two variables be determined: K –window size and α - smooth factor.

These values have a strong effect on algorithm successfulness and choosing the right one is critical. Too small a window size may cause the algorithm to be overly sensitive to temporal delays or glitches not caused by bandwidth over-utilization. On the other hand, detection time is at least the window size in time as the TREND algorithm requires an increasing trend in the delay of K frames. Therefore, a large window would result in late detection of delay events. Additionally, smoothing the delay too strongly or too weakly for a chosen window size may improve detection time but lead to a false positive event.

a



b

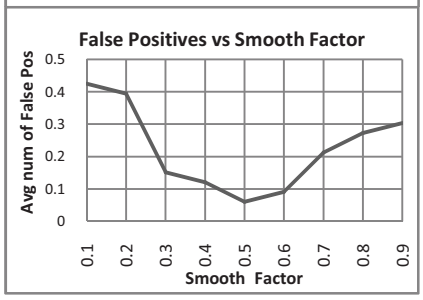


Fig. 5 The affect of smooth factor on delay detection, over a window of size 11.

Figure 5 demonstrates the effect of smooth factor on delay detection over a window of size 11. Figure 5.a displays the average number of frames required for delay detection. It can be observed that the delay detection time is significantly lower for small smooth factors. Figure 5.b shows the average number of false positives per two-second call. It can be observed that both small and large values increase the number of false positives. Note that the false positives graph and detection time graph show opposite behavior, i.e. one increases as the other decreases and vice versa.

Since the correctness measures as well as the penalty measure can be computed offline on a database of prerecorded tests, the penalty function can be used to calibrate delay detector parameters. The calibration process chooses values for delay detector parameters that minimize function P.

5. RESULTS

The TREND algorithm was tested with more than 60 video calls, with test setup as depicted in Figure 6. The transmitters were real-life video conferencing endpoints from leading vendors, such as LifeSize (Logitech), Polycom and Tandberg (Cisco). TREND algorithm was implemented at the receiver, utilizing standard flow control commands for bitrate control. The calls were performed over different network links, such as MPLS, ADSL and Cable Modem.

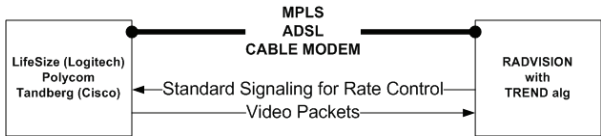
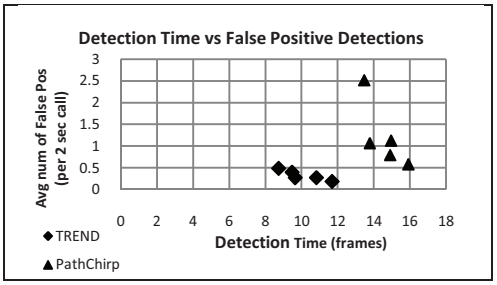


Fig. 6 Test setup.

a



b

$procedure\ excursion(q, i, F, L)\{$
$j = i + 1;$
$max\_q = 0;$
$while(j \leq N\ and\ (q(j) - q(i)) > \max\_q/F\ )\{$
$\max\_q = maximum(max\_q, q(j) - q(i));$
$j = j + 1;\}$
$if\ (j \geq N) return\ j;$
$if\ (j - i \geq L) return\ j;$
$else\ return\ i;\}$

Fig. 7 Detection time versus detection errors in comparison to the PathChirp delay detector.

We first tested convergence to the available bandwidth upon call initiation, as well as mid-call changes in the available bandwidth. The latter was carried out by allowing a call to converge to the available bandwidth, and after a while introducing a new call on the same link. We learnt that the TREND algorithm converges to 90-95% of the available bandwidth in up to 15 seconds with no packet loss.

Next we tested the TREND delay detector according to the correctness measures defined in 4.1.1. Each call was initiated with more than 90% of the available bitrate and lasted precisely 2 seconds. In 75% of the calls we modified either the call setup or the link to obtain overutilization of the bandwidth during the first second of the call. We noticed that an optimal choice of TREND parameters leads to an average detection time of 9.65 frames. In other words once the bandwidth is over-utilized it takes 9.65 frames on average to detect it. The average number of false positive events for this set of parameters is 0.27 events per 2 second call. Figure 7a demonstrates the TREND algorithm detection time versus false positive events as defined in 4.1.1. Every data point was obtained from optimizing the algorithm for different window size K. The tradeoff between detection time and the number of false positive events is

demonstrated. TREND algorithm users can configure it to favor either the detection time or the number of false positive events, according to their application's requirements.

Next we compared the TREND delay detector with that of a leading probing algorithm: We repeated the above tests while using the TREND algorithm for all aspects related to video communication, but determining *UP* (or *DOWN*) *events* using PathChirp methodology [17]. The results show that TREND has a significant advantage for both detection time and the number of false positive events, as seen in Figure 7.a. Figure 7.b displays the PathChirp delay detector algorithm as described in [17]. q is the set of delta delays, N is the size of the delta delay set and i is the first frame in the tested window. L and F were optimized with the parameter calibration process as defined in section 4. If the returned value satisfies $i < j \leq N$ then an *UP event* should be declared. On average, PathChirp's detection time is between 13.47 to 17.3 frames, and the number of false positive events per call is 0.42 to 2.51. Both the detection time and the false positive events of PathChirp are noticeably higher than those of TREND.

6. CONCLUSION

We have presented TREND, a bandwidth estimation and adaptation algorithm for interactive, real-time video calling. TREND converges to the available bandwidth during call setup, and dynamically adjusts the bandwidth in mid-call when changes occur. TREND operates solely on the receiver side, and utilizes standard protocols for delay detection and bandwidth adaptation.

Different methods used for evaluating TREND's performance were discussed and comparisons made with other algorithms. These methods were used to optimize TREND and our results show that it consistently outperformed existing tools in terms of detection time and number of false positive events.

Future work includes investigating methods to estimate and adapt bandwidth in networks where there is no delay before packet loss (no buffering). In these cases the challenge is to react to packet loss resulting from bandwidth over-utilization, while ignoring packet loss caused by other network events.

7. ACKNOWLEDGEMENTS

The work presented in this paper has been carried out by RADVISION as part of the EU-FP7 3DPresence project (215269) framework (http://www.3dpresence.eu).

8. REFERENCES

[1] Google Talk Call Signaling, Google Talk for Developers, Google Code, http://code.google.com/intl/iw-IL/apis/talk/call_signaling.html#Video_Rate_Control

[2] RFC 3448: TCP Friendly Rate Control (TFRC), http://www.ietf.org/rfc/rfc3448.txt

[3] Quality of Service (QoS) White Paper for SONY PCS-Series Videoconferencing, www.sony.co.uk/res/attachment/file/20/1187079500620.pdf

[4] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video*, Vol. 17, Issue 9, pp. 1103-1120, Sep 2007.

[5] G 1070: Opinion model for video-telephony applications, http://www.itu.int/rec/T-REC-G.1070-200704-I

[6] RFC 3550: RTP: A Transport Protocol for Real-Time Applications, http://www.ietf.org/rfc/rfc3550.txt

[7] H.245: Control Protocol for Multimedia Communication, http://www.itu.int/rec/T-REC-H.245/en

[8] H.323: Packet-based Multimedia Communication Systems, http://www.itu.int/rec/T-REC-H.323/en/

[9] H.324: Terminal for Low Bit-rate Multimedia Communication, http://www.itu.int/rec/T-REC-H.324/en

[10] RFC 5104: Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF), http://tools.ietf.org/html/rfc5104

[11] SIP: Session Initiation Protocol (RFC 3261), http://tools.ietf.org/html/rfc3261

[12] C.M. Assi, Y. Ye, S. Dixit and M.A. Ali, "Dynamic Bandwidth Allocation for Quality-of-Service over Ethernet PONs," *IEEE Journal on Selected Areas in Communications*, Vol. 21, Issue 9, pp. 1467–1477, Nov 2003.

[13] M.P. McGarry, M. Maier and M. Reisslein, "Ethernet PONs: A Survey of Dynamic BW Allocation (DBA) algorithms," *IEEE communications magazine*, Vol. 42, Issue 8, pp. S8–S15, Aug 2004.

[14] Reza Rejaie, Mark Handley and Deborah Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", *Proceedings of IEEE INFOCOM*, pp. 1337-1345, March 1999.

[15] Tansu Alpcan and Tamer Bas, "A Utility-Based Congestion Control Scheme for Internet-Style Networks with Delay", *Proceedings of IEEE INFOCOM*, vol.3, pp. 2039-2048, Apr 2003.

[16] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," *ACM SIGCOMM Computer Communication Review*, Vol. 32, Issue 4, pp. 295 – 308, Oct 2002.

[17] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," *Passive and Active Measurement Workshop*, CA, USA, Apr 2003.

[18] A. Balk, M. Gerla, and M. Sanadidi, "Adaptive MPEG-4 Video Streaming with Bandwidth Estimation: Journal Version," *Workshop on QoS with Multiservice IP*, Italy, Feb 2003.

[19] D.T Nguyen, J. Ostermann , "Streaming and Congestion Control using Scalable Video Coding based on H.264/AVC," *Journal of Zhejiang University SCIENCE A*, Dec 2005.

[20] T. Schierl and T. Wiegand, "H.264 Rate Adaptation for Internet Streaming," *Packet Video*, California USA, Dec 2004.

[21] H. Kanakia, P. P. Mishra, and A. R. Reibman, "An Adaptive Congestion Control Scheme for Real Time Packet Video Transport," *IEEE/ACM Transactions on Networking*, Vol. 3, Issue 6, pp. 671 – 682, Dec 1995.

[22] RFC 2309: Recommendations on Queue Management and Congestion Avoidance in the Internet, http://www.ietf.org/rfc/rfc2309.txt

[23] RFC1305: Network Time Protocol (Version 3) Specification, Implementation and Analysis, http://www.faqs.org/rfcs/rfc1305.html