Continuous One-Way Available Bandwidth Change Detection in High Definition Video Conferencing

Aziz Khanchi, Mehdi Semsarzadeh, Abbas Javadtalab and Shervin Shirmohammadi
Distributed and Collaborative Virtual Environment Research Lab (DISCOVER Lab)
School of Electrical Engineering and Computer Science (EECS), University of Ottawa, Canada
{akhanchi, msemsarzadeh, javadtalab, shervin} @discover.uottawa.ca

ABSTRACT
In order to deal, as fast as possible, with available bandwidth variations in High Definition Video Conferencing over best-effort networks, we propose a Bayesian instantaneous end-to-end bandwidth change discovery mechanism. Most other congestion detection mechanisms use network parameters such as packet loss probability, round trip time or jitter. On the other hand, our approach uses weighted inter-arrival time of video packets at the receiver in order to detect bandwidth variations more accurately and more quickly. Our approach is continuous, since it monitors available bandwidth with each incoming video packet, and therefore detects congestion occurrence in less than 200 ms, on average, which is significantly faster than existing RTCP-based approaches. It is also one-way, because it only takes into account the characteristics of the incoming path and not the outgoing path, as opposed to other approaches which use round trip time.

Categories and Subject Descriptors
H.4.3 [Information Systems Applications]: Communications Applications – Computer conferencing, teleconferencing, and videoconferencing.

General Terms
Algorithms, Measurement, Performance, Design.

Keywords
Congestion Detection, VBR Video, Bayesian Statistics, High Definition Video Conferencing, Inter-arrival Time.

1. INTRODUCTION
With the constant growth of ubiquitous multimedia access, the number and variety of multimedia aware devices and their corresponding applications are increasing every day. Video conferencing is one of those applications that is becoming more and more popular in the multimedia world. The market for this application is growing rapidly due to its efficiency, practicality, convenience and affordability. Typical customers of video conferencing include hotels, law firms, remote development firms, construction firms, and generally companies and organizations with geographically-distributed branches.

Video conferencing, especially High Definition Video Conferencing (HDVC), is a high bandwidth application; the higher the video resolution, the higher the needed network bandwidth. Furthermore, user satisfaction or specifically quality of service (QoS) plays the most important role in video conferencing. This is why many current video conferencing systems require dedicated networks with guaranteed quality of service, such as CISCO’s Telepresence (formerly Tandberg) or Hewlett-Packard’s Halo telepresence. However, a dedicated network increases the total cost of the solution and is also a recurring cost. Hence, a solution that takes advantage of Internet’s best-effort services is preferred. Because best effort networks do pose the problem that the available bandwidth changes dynamically over time and is not fixed, it is challenging to transmit high quality video with guaranteed QoS. A detailed discussion about HDVC requirements and challenges in its video transmission was presented in [1] and will not be repeated here, other than mentioning that one of the problems to be solved is the live detection of changes in the available bandwidth, in order to adjust the video bitrate to match that bandwidth for two purposes: 1- to ensure very few (ideally none) video packets are lost if the available bandwidth goes down, and 2- to ensure the video is transmitted at a higher quality when the available bandwidth goes up. Hence, changes in the available bandwidth have to be detected quickly in order to maintain the QoS of a video conferencing system, which is the main goal of this paper. While our proposed method detects the network congestion in a prompt way, the video bitrate won’t be adjusted drastically to ensure acceptable video quality from end user’s perspective.

Current two-way congestion detection and avoidance techniques rely on round-trip time (RTT) and/or packet loss measurements using RTCP packets that are sent from the sender side. But such existing methods have certain shortcomings, discussed in details in section 0 that make them unsuitable for video conferencing. In this paper, we will propose a one-way congestion detection method that only depends on video packets without any control information such as RTCP. We measure bandwidth changes at the receiver side, by utilizing the received video packets and implementing Bayesian analysis. Based on the network status and our network congestion detection algorithm, appropriate feedbacks can then be sent to the encoder to adapt its video encoding bitrate to network conditions in a live manner. Our primary focus is real-time end-to-end live video transmission, as is the case for HDVC. For prerecorded or on-demand video transmission, which is outside the scope of this paper, we refer the readers to [2].

The rest of this paper is organized as follows. The related works are reviewed in section 2. Our proposed Bayesian congestion detection method is discussed in Section 2. Simulation results are explained in Section 4, and finally the paper concludes in section 5.

2. RELATED WORK
There is much literature focusing on the detection and/or estimation of the available bandwidth [3]-[9] or on rate adaptation techniques [10]-[14]. In general, there are two stages for adapting the video bitrate in response to network fluctuations. First, certain network
parameters need to be measured. Exactly which parameters, depends on the specific technique, but typically parameters such as RTT or packet loss rate are measured. Second, the video bit rate needs to be adjusted according to the measured parameters. There is a major agreement in the scientific community that the video bitrate should be adjusted in a manner that makes its transmission TCP-friendly.

In [10], an equation is introduced in order to adapt the sending rate to network fluctuations for a unicast application, by relying on packet loss probability and RTT measurements. This equation replaces the AIMD approach in regular TCP and is shown to be TCP compatible. The throughput equation is shown in formula (1), as provided in [11].

\[
X_{pps} = \frac{1}{RTT \left( \frac{2p}{3} + 12 \sqrt{\frac{3p}{8}} p(1 + 32p^2) \right)},
\]

where \(X_{pps}\) is the sending rate in packets per second, \(RTT\) is round trip time in seconds and \(p\) is the loss event rate, between 0 and 1, of the number of loss events as a fraction of the number of packets transmitted. As the \(RTT\) and/or \(p\) increase, \(X_{pps}\) decreases. Eq. (1) not only performs congestion detection through monitoring \(RTT\) and \(p\), but also provides a sending rate in response to detected bandwidth fluctuations. In Section 0, approaches using Eq. (1) outcomes are compared to our proposed approach. Moreover, our commercial partner currently utilizes this equation as part of their HDVC package.

Many researchers have extended the TCP-friendly scheme for various applications and network topologies. For example, equation (1) is extended to multicast applications in [12]. In [13], a prediction method is developed based on packet loss probability and is shown to be TCP-friendly. This method also uses RTT predictions. The network performance evaluation is based on comparing the true current state with the forecasted state at the end of the previous control period. Each control period has three different cases: unloaded, loaded and congested. Crossing forecasts with actual cases leads to different sending rates. For example, if a control period is forecasted to be unloaded and its current status is also unloaded, then the sending rate is increased aggressively. Finlay, 0 provides a survey of TCP-compatible congestion control methods.

The above techniques have three main shortcomings, especially when it comes to HDVC.

First, they are all based on round trip time, which means they are unable to perform measurements for actual one-way traffic. In fact, it is well known that RTT latencies provide misleading information for unidirectional delays [21]. In video conferencing, if users A and B are teleconferencing, the network path from A to B is not necessarily the same as from B to A. More importantly, the traffic from A to B is not the same as that from B to A. Given the high volume of data transmitted by this application, it becomes crucial to measure the one-way characteristics accurately. In other words, if we detect a bandwidth reduction from A to B, we need to reduce the video bit rate from A to B, but the video bit rate from B to A must remain the same if no changes are detected from point B to A. Existing techniques do not make such distinction, whereas our approach has been specifically designed to be a one-way measurement technique. Enhanced approaches like TCP Vegas also depend on RTT.

Second, existing techniques take too long to detect and report changes, typically in few second intervals. They assume available bandwidth to be mostly stationary and consequently measure the available bandwidth over (possibly) few RTTs. This is not suitable for video conferencing which requires bandwidth changes to be detected as fast as possible, in order to maintain the video quality as high as possible. Our approach detects changes much faster and is therefore more suitable for video conferencing.

Third, existing techniques mostly rely on control packets, which adds data overhead to the network and processing overhead to the application. In our technique, we use only the incoming video packets themselves, without any control messages, and we analyze them to detect changes in the available bandwidth. As such, our approach has less overhead as well.

3. PROPOSED BAYESIAN CONGESTION DETECTION MODEL

In this section, we describe the weighted inter-arrival process and introduce our statistical model and Bayesian approach. This will provide us with the technical tools necessary for congestion detection.

For simplicity, we will consider only one queue of video packets that travel from a single sender to a single receiver through an unreliable network. Once the packets are received by the receiver, they are time-stamped. We define the random process of inter-arrival times, \(D_i\), as the difference in packet spacing at the receiver side for a pair of subsequent packets. If \(R_i\) is the arrival time in timestamp units for packet \(i\), then:

\[
D_i = R_i - R_{i-1}
\]

Note that

\[
D_i = R_i - R_{i-1} = (OTT_i + T_i) - (OTT_{i-1} + T_{i-1})
\]

\[
= (OTT_i - OTT_{i-1}) - (T_i - T_{i-1})
\]

where \(OTT_i\) and \(T_i\) are one-way transit time and sending time of the \(i\)th packet, respectively. Because the video packets are sent by RTP, the receiver has no information about the sending times. However, equation (3) shows how bandwidth fluctuations can affect the inter-arrival process. Obviously, out of order delivery is a problem here, but we will use statistical inference to minimize this issue. The basic idea in our method is that the \(OTT\) shows an increasing trend when the available bandwidth drops for any reason.

Due to variable packet sizes in a video stream, we will engage the weighted inter-arrival time, \(J_i\), as defined in formula (4).

\[
J_i = \frac{D_i}{\text{Size of packet } i}
\]

In line with Internet data analysis 0 [15], and as confirmed by our measurements of HDVC packets, the random variable \(J_i\) is heavy-tailed and approximately fits a Pareto distribution.

The probability density function (pdf) of a Pareto random variable with parameters \(\rho\) and \(x_M\) is:

\[
f(x | \rho, x_M) = \frac{\rho x^\rho}{x_M^{\rho+1}}, \ x \geq x_M > 0, \rho > 0,
\]

where \(x_M\) is the minimum possible value of the random variable and \(\rho\) is the shape parameter.

For the random variable \(J_i\) of HDVC video packets, we may assume a Pareto distribution with fixed or stochastic shape parameter, \(\rho\).

Due to the complex nature of Internet communications and traffic, it is reasonable to think of the shape parameter itself as a random variable that varies over time. This time-varying perspective will harness the unpredictable behavior of bandwidth variations. In other words, if we agree that \(J_i\) follows a Pareto distribution, there is no
reason to accept universal values for the shape parameter because this parameter itself may change over time, even within a single HDVC session.

Let us assume that we have a reasonable estimate of the minimum parameter \(x_M\). This information can be obtained from previous experiences or it can simply be a sensible number, although it should not be much smaller than the actual minimum. From this point, the only unknown factor in the Pareto distribution is the shape parameter, \(\rho\), which in turn is our new random variable. As a random variable, the shape parameter needs a distribution by itself. In Bayesian statistics, it is well-known that the conjugate distribution for the shape parameter of a Pareto distribution with known minimum, \(x_M\), is the gamma distribution [2]. The choice of the gamma distribution guarantees that as we dynamically update the information according to weighted inter-arrival times, the distribution of \(\rho\) does not alter from the gamma distribution. We will assume that \(\rho \sim G(a, b)\) where \(G(a, b)\) is the gamma distribution with mean and variance \(ab\) and \(ab^2\), respectively.

In the following subsection, we will first show how the weighted inter-arrival times can be used to update the distribution of \(\rho\). Then, we will introduce our control scheme.

### 3.1 Updating the distribution of \(\rho\)

At the beginning of the session, we assume that the shape parameter denoted by \(\rho_0\) follows the gamma distribution \(G(a_0, b_0)\), where \(a_0\) and \(b_0\) are predetermined and fixed. These fixed numbers can be chosen based on our previous information or we can choose some logical fixed positive numbers with the hope that the system will correct itself in a short period of time. Note that initially the mathematical expectation of \(\rho = \rho_0 = a_0b_0\).

After the arrival of the first two packets, \(J_1\) is calculated with packet numbering starting from zero. Therefore, the posterior distribution of the shape parameter based on the observed data, \(J_1\), will be modified as shown in (6).

\[
G \left( a_0 + 1, \frac{1}{b_0} + \ln \frac{J_1}{x_M} \right)^{-1}
\]  
(6)

As a result, based on the numerical value of \(J_1\), \(\rho\) follows a gamma distribution with average represented in (7).

\[
E(\rho) = (a_0 + 1) \left( \frac{1}{b_0} + \ln \frac{J_1}{x_M} \right)^{-1}
\]  
(7)

Once the next (third) packet arrives and \(J_2\) is calculated, it will be used to obtain a new estimation of the Pareto shape parameter, and its corresponding average, shown in equations (8) and (9), respectively.

\[
E(\rho) = (a_1 + 1) \left( \frac{1}{b_1} + \ln \frac{J_2}{x_M} \right)^{-1}
\]  
(8)

\[
E(\rho) = (a_1 + 1) \left( \frac{1}{b_1} + \ln \frac{J_2}{x_M} \right)^{-1}
\]  
(9)

This scheme will be continued and after each packet arrival the expectation of the shape parameter is calculated, which will provide us with a sequence of mathematical expectations of the shape parameter. Note that for simplicity, we can write:

\[
b_n^{-1} = \frac{1}{b_0} + \ln \left( \frac{\prod_{i=1}^{n} J_i}{x_M^2} \right)
\]  
(10)

\[
E_n = E(\rho) = (a_0 + n)b_n
\]  
(11)

Our data shows that the sequence \(E_i\) is highly associated with the reliability of the network. In 03.2, we will show that the bandwidth variations can be detected by examining the moving average of \(E(\rho)\).

In order to express the relation between \(E_n\) and network status, we have extracted \(E(\rho)\) based on the arrival time of video packets of an actual HDVC trace. The video trace we used is an actual recording of an HDVC session using Major Communication’s Telecollaboration HDVC system, at 1080p resolution, 30 frames per second, lasting for 12 seconds, and with its video bitrate varying between 0.75 and 2 Mbps. The resulting \(E(\rho)\) is shown in Figure 1. As the figure shows, \(E(\rho)\) follows the bandwidth condition closely with an apparent linear trend; i.e., the slope of \(E(\rho)\) changes whenever the bandwidth changes.

Although \(E(\rho)\) and network bandwidth follow the same trend, in the small scale (zoomed-in section in Figure 1) the non-linearity of \(E(\rho)\) makes it challenging to have any regression type justification about the network condition, while the bandwidth variation has to be detected with an accuracy in the order of millisecond. Hence, we propose to use the moving average of \(E(\rho)\) instead of its instant values, as described in the next subsection.

### 3.2 Control Scheme

As explained before, our analysis of actual HDVC traces shows that variations of \(E(\rho)\) highly depend on bandwidth fluctuations. Therefore, we propose a congestion control technique based on comparing moving averages of \(E(\rho)\). Comparing moving averages is a well-known technique used for the prediction of stock markets and financial applications.

We propose to generate a numerical sequence of differences of the last moving averages of orders \(n_1\) and \(n_2\) \((n_1 < n_2)\) following each packet arrival. When the \(i^{th}\) packet arrives \((i > n_2)\), the difference is determined as shown in formula (12).

\[
\text{Diff}_{i} = MA_{i}(n_1) - MA_{i}(n_2),
\]  
(12)

where

\[
MA_i(n) = \frac{\sum_{l=n+1}^{i} E_l}{n}, i \geq n
\]  
(13)

After determining the value of \(\text{Diff}\), we can detect network congestion easily. A sharp decline in the value of \(\text{Diff}\) indicates a decrease in the available bandwidth and, analogously, a rise in \(\text{Diff}\) shows an increase. The steps of List demonstrate the overall scheme. As shown in List , after extracting the \(\text{Diff}_{i}\) value, we can adjust the sender’s video rate by comparing \(\text{Diff}_{i}\) fluctuations with
lower and upper bounds, represented by $\text{Diff}_{i-\Delta_2} - \Delta_1$ and $\text{Diff}_{i-\Delta_2} + \Delta_1$, respectively. Here, $\Delta_2$ is the time interval that the bounds are kept fixed. Once $\text{Diff}_i$ crosses the lower (upper) bound it indicates a decrease (an increase) in the bandwidth.

- **Step 1-Collect Raw network information:**
  
  Determine arrival time, $D_1$, and size of packets upon arrival at the receiver's side and extract $f_i$ using (4).

- **Step 2- Update $E(\rho)$:**
  
  Use (11) to update the expected value of $\rho$
  
  \[ E_n = E(\rho) = (a_0 + n)b_n \]

- **Step 3- Evaluate Diff:**

  Evaluate Diff using (12):
  
  \[ \text{Diff}_i = MA_n(n_1) - MA_n(n_2) \]

- **Step 4- Generate lower and upper bound:**

  Generate lower and upper bounds every fraction of a second.

- **Step 5- Monitor bound crossings:**

  Send the information to the sender to decrease or increase the video bit rate according to bound crossings.

**List 1. Proposed method for adjusting the video bitrate**

Considering steps 1, 2, and 3, once Diff is sampled; the lower and upper bounds will be generated according to the following rules:

- If the first nonzero decimal of Diff is of order $10^{-2}$, the bounds are proposed to be $\text{Diff} - 0.02$ and $\text{Diff} + 0.05$. We have used a closer lower bound to increase sensitivity.
- Otherwise, if the first nonzero decimal of Diff is of order $10^{-n}$ where $n \geq 3$, then bounds should be $\text{Diff} \pm 1.5(10^{-n+1})$.
- In order, to avoid computation complications, we may choose a lower bound on $1.5(10^{-n+1})$.

It should be noted that the numbers $0.02, 0.05$, and $1.5$ in the above rules have been determined experimentally using dozens of HDVC recording samples, where it was found that these numbers do not depend on the specific video but on the network behavior. These numbers are quite important and their choice affects the sensitivity of our congestion detection method. Hence, more future work is needed to determine these numbers in a systematic manner.

## 4. SIMULATION RESULTS

In this section, we show the speed and accuracy of our bandwidth measurement method and compare it with the most common method, which is the congestion detection algorithm of RTT-based schemes, such as those used in [11] and [19]. In order to examine the performance of the proposed method, we have used the x264 reference software [17] as the H.264/AVC video encoder and the NS2 software [18] as the network simulator platform.

The NS-2 simulation setup is depicted in Figure 2. Here, S1 is the RTP sender which is responsible to send a video stream. The stream goes through the Routers (R1 and R2). The whole throughput of the system is determined by the link capacity (L1) and S4 is the receiver of RTP stream. S2 and S3 are used to generate traffic other than video stream. The Sender (S1) sends video packets and the Receiver (S4) collects video packets and stores it in a file. The packet is transmitted with RTP/RTCP protocol and all network parameters are stored in the log files.

In our simulation scenario, various videos (see Table 2) are coded and sent through the NS2 software.

For this experiment, $n_1 = 20$ and $n_2 = 50$. We use a dynamic $\Delta_2$ and update the bounds every 50 packet arrivals, based on the update rules discussed in the previous section.

In order to examine the performance of our method for different network conditions, we have defined four cases for HD videos and 2 cases for SD videos as shown in Table. For all of the cases in Table, the video encoder bitrate is fixed and is equal to the Initial bandwidth of each case. The network then changes its bandwidth at frame 60 (second 2 of the video) to the Secondary bandwidth indicated in the table.

Table 2 and Table 3 show the response time of the tested methods for various HD and SD video sequences, respectively. We should mention that the recommended time spacing between RTCP packets is 5 seconds [19] [20]. However, because one of our goals is to illustrate the speed of our approach, we have decreased the RTCP transmission interval to 1 second (in Table 2) and 0.1 second (in Table 3) to be fairer to RTCP. As the results of Table 2 and Table 3 show, our proposed method outperforms the RTCP-based scheme in all of the simulation cases.

In addition, Figure 3 shows the graph of Diff with the generated upper and lower bounds for the Rush hour video sequence, while network condition is set to case 4 of Table. As we can see from the graph, the sudden decline in Diff is triggered right after the sudden decrease in bandwidth from 2 Mbps to 1 Mbps. However, the receiver reports the detected congestion at 2.2 sec when Diff crosses the lower bound.

Finally, in order to show the efficiency of our method, we have considered an unrealistic case of RTCP transmission interval of 0.2 second for HD size videos. The simulation results for this case are shown in Table 4. As the table shows, even in this unrealistic case our proposed method outperforms the RTCP-based scheme, in all simulation cases. It can also be seen from our results that, although in our approach the receiver still has to inform the sender of the bandwidth change, meaning that the sender will get the information in a round trip time similar to existing systems, our approach leads to a faster and more accurate congestion information at the sender, while the proposed method in [11] and [19] detects the congestion much later or even sometimes it fails to detect the congestion. Moreover, the receiver should delay its congestion detection until the sender adjusts its sending rate.

In addition to these ns-2 simulations and results, our preliminary implementation and evaluations on our partner's testbed show that our proposed method works, and bandwidth changes are detected immediately in many instances.

There is one other performance metric that we did not measure, and that is the fact that our approach measures one-way network changes, whereas the RTT methods cannot measure unidirectional network characteristics. Since this is obvious and intuitive, there is no need to conduct simulations to show it.
Table 1. Network condition cases

<table>
<thead>
<tr>
<th>Case Number</th>
<th>Network Bandwidth (Mbps)</th>
<th>HD Videos</th>
<th>SD Videos</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Initial</td>
<td>Secondary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Initial</td>
<td>Secondary</td>
</tr>
<tr>
<td>1</td>
<td>2.0</td>
<td>3.0</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>2.0</td>
<td>2.5</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>2.0</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>1.0</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Response time comparison for the proposed and RTCP-Based schemes, with 1 sec RTCP interval – HD videos

<table>
<thead>
<tr>
<th>Case#</th>
<th>Sequence name</th>
<th>Response time (Sec)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Proposed method</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>Tractor</td>
<td>0.0990</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Sunflower</td>
<td>0.1224</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Station 2</td>
<td>0.1201</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Rush hour</td>
<td>0.1382</td>
<td>No detection</td>
</tr>
<tr>
<td>2</td>
<td>Tractor</td>
<td>0.1216</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Sunflower</td>
<td>0.1616</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Station 2</td>
<td>0.1438</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Rush hour</td>
<td>0.1647</td>
<td>No detection</td>
</tr>
<tr>
<td>3</td>
<td>Tractor</td>
<td>0.1995</td>
<td>1.1970</td>
</tr>
<tr>
<td></td>
<td>Sunflower</td>
<td>0.1649</td>
<td>1.1970</td>
</tr>
<tr>
<td></td>
<td>Station 2</td>
<td>0.2262</td>
<td>1.5966</td>
</tr>
<tr>
<td></td>
<td>Rush hour</td>
<td>0.1867</td>
<td>1.5966</td>
</tr>
<tr>
<td>4</td>
<td>Tractor</td>
<td>0.1836</td>
<td>1.5966</td>
</tr>
<tr>
<td></td>
<td>Sunflower</td>
<td>0.1541</td>
<td>1.5966</td>
</tr>
<tr>
<td></td>
<td>Station 2</td>
<td>0.1994</td>
<td>1.3973</td>
</tr>
<tr>
<td></td>
<td>Rush hour</td>
<td>0.1683</td>
<td>1.3973</td>
</tr>
<tr>
<td></td>
<td>Average time ahead when both methods have detection</td>
<td>0.5115</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Response time comparison for the proposed and RTCP-Based schemes, with 0.1 sec RTCP interval – SD videos

<table>
<thead>
<tr>
<th>Case#</th>
<th>Sequence name</th>
<th>Response time (Sec)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Proposed method</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>City</td>
<td>0.4035</td>
<td>0.1708</td>
</tr>
<tr>
<td></td>
<td>Crew</td>
<td>0.2042</td>
<td>0.2734</td>
</tr>
<tr>
<td></td>
<td>Harbour</td>
<td>0.4035</td>
<td>0.3283</td>
</tr>
<tr>
<td></td>
<td>Ice</td>
<td>0.4035</td>
<td>0.1982</td>
</tr>
<tr>
<td>2</td>
<td>City</td>
<td>0.5027</td>
<td>0.1755</td>
</tr>
<tr>
<td></td>
<td>Crew</td>
<td>0.2042</td>
<td>0.2745</td>
</tr>
<tr>
<td></td>
<td>Harbour</td>
<td>0.6022</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Ice</td>
<td>0.4035</td>
<td>0.2043</td>
</tr>
<tr>
<td></td>
<td>Average time ahead when both methods have detection</td>
<td>0.128586</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Response time comparison for the proposed and RTCP-Based schemes, with 0.2 sec RTCP interval – HD videos

<table>
<thead>
<tr>
<th>Case#</th>
<th>Sequence name</th>
<th>Response time (Sec)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Proposed method</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>Tractor</td>
<td>0.0943</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Sunflower</td>
<td>0.1302</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Station 2</td>
<td>0.0936</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Rush hour</td>
<td>0.1332</td>
<td>No detection</td>
</tr>
<tr>
<td>2</td>
<td>Tractor</td>
<td>0.1056</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Sunflower</td>
<td>0.1872</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Station 2</td>
<td>0.0936</td>
<td>No detection</td>
</tr>
<tr>
<td></td>
<td>Rush hour</td>
<td>0.1653</td>
<td>No detection</td>
</tr>
<tr>
<td>3</td>
<td>Tractor</td>
<td>0.2110</td>
<td>3.0552</td>
</tr>
<tr>
<td></td>
<td>Sunflower</td>
<td>0.1496</td>
<td>3.0552</td>
</tr>
<tr>
<td></td>
<td>Station 2</td>
<td>0.4107</td>
<td>1.9919</td>
</tr>
<tr>
<td></td>
<td>Rush hour</td>
<td>0.2005</td>
<td>1.9919</td>
</tr>
<tr>
<td>4</td>
<td>Tractor</td>
<td>0.1975</td>
<td>0.9321</td>
</tr>
<tr>
<td></td>
<td>Sunflower</td>
<td>0.1520</td>
<td>0.9321</td>
</tr>
<tr>
<td></td>
<td>Station 2</td>
<td>0.2537</td>
<td>0.9321</td>
</tr>
<tr>
<td></td>
<td>Rush hour</td>
<td>0.2082</td>
<td>0.9321</td>
</tr>
<tr>
<td></td>
<td>Average time ahead when both methods have detection</td>
<td>1.5049</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. Graph for our proposed approach for the Rush hour video decreasing the bandwidth from 2 Mbps to 1 Mbps. The vertical and horizontal axes represent time in seconds and Diff values, respectively. The curve shows the values of Diff upon every packet arrival and the straight lines are the generated lower and upper bounds.

5. CONCLUSION

In this paper, we described a one-way and fast congestion detection mechanism. Among the benefits of our work is the fact that our mechanism is receiver-based and depends only on the path connecting the sender to the receiver and hence does not rely on RTT. Our proposed Bayesian mechanism improves upon currently available control methods, and can also be implemented in other (non-HD) real time applications if required. Based on our simulation results, the proposed method predicts the congestion occurrence well ahead of the popular RTCP-Based scheme, in all of simulation cases.
Our method can be used either on its own, or in conjunction with other methods, even RTCP based methods, in order to provide more information for the rate adaptation decision engine. For our future work, the mathematical model can be improved by considering a stochastic minimum value for the Pareto distribution.

6. REFERENCES


