Bandwidth estimation and congestion avoidance for video streaming in mobile cloud

S.P. Tamizhselvi 1*, Vijayalakshmi Muthuswamy2

1 Department of Computer Science Engineering, KCG College of Technology, Anna University, Chennai, India
2 Department of Information Science Technology, College of Engineering, Anna University, Chennai, India

E-mail: 1tamizh8306@gmail.com, 2vijim@annauniv.edu

*Corresponding Author : S.P. Tamizhselvi

Abstract

The cloud technology has been emerged with the evolution of the mobile phone to deliver live streaming, social network, search engine, GPS navigation and email, etc. While providing the services, the smartphone faces many QoS challenges and video quality diminishes due to network busy, bandwidth, congestion, delay, and packet loss. To solve this issue, we propose a novel framework, namely, Network bandwidth estimation and congestion avoidance for video data in the mobile cloud. To manipulate the network busy, bandwidth, and congestion, we propose two algorithms such as Cloud Network Bandwidth Estimation (CNBE), Estimated Bandwidth Congestion Window (EBCW). Initially, CNBE utilizes the actual bandwidth based on different network busy and EBCW minimizes the congestion, according to the help of estimated bandwidth. Our implementation has been carried on the public cloud AWS (Amazon Web Service). CNBE achieves the packet loss rate as 10% and 70% in low, and high network. Thus, this reduces the packet loss and improves the goodput performance. CNBE are evaluated for different network which achieves high throughput compared with existing TCP variants. Next, EBCW dynamically changes the window depends on network status and bandwidth estimation to minimize the window and helps to avoid the congestion and packet loss. Thus, the proposed algorithms deliver a good quality of video to the mobile.

Keywords: Network busy; Bandwidth estimation; Congestion avoidance; Mobile Cloud; Quality of Service; Packet loss rate; Throughput; Goodput.

1. Introduction

In the upcoming era, mobile devices becomes popular to execute data-intensive applications such as real-time video streaming, YouTube and Netflix and smartphone users also expect to execute a resource-intensive task on their devices [1] itself with a quick response. Basically mobile devices have storage and processing constraints, and the battery, bandwidth, packet loss, and congestion. The limitations of mobile device are overcome by integrating the Cloud Computing (CC) with mobile devices and provide a new technology termed as Mobile Cloud Computing (MCC) [2-5]. In MCC environment, congestion occurs, and is necessary to propose efficient mechanisms to control congestion based on effective bandwidth.
estimation techniques [6]. The new bandwidth estimation and congestion control techniques are introduced in the server to receive a video request and process the heavy task of multimedia applications remotely [7]. In order to support, Transmission Control Protocol (TCP) [8] provides an end-to-end transmission protocol which applies a window-based technique and determines the packets sending rate and utilizes the available bandwidth of mobile devices to control the congestion. TCP congestion plays a significant role in the design of transport protocol [9]. However, the existing TCP based congestion techniques are not sufficient to handle the video traffic in mobile and Internet applications.

In network traffic, the cloud server plays a vital role to manage the various issues of mobile as congestion, bandwidth estimation, quality of service, and streaming video applications [10]. In networks, the available bandwidth is changed due to network loss and delay. Managing the available bandwidth of TCP packets flow is prolonged and confused and the complexity produces congestion in both the network dynamics and TCP packets. However, congestion control is a recommended TCP mechanism to detect the best bandwidth in which the packets sent and deliver at the receiver end [11]. We extended TCP behavior and the various flow control methods and the performance are improved by using the existing congestion control cloud-based algorithm. The traditional TCP congestion control algorithms are Congestion Avoidance (CA), slow start, fast retransmit, and fast recovery. The steady-state characteristics of the TCP congestion control window are satisfied only in CA. In the survey, researchers discussed the different CA algorithms such as TCP Reno, TCP Westwood (TCPW), TCP CUBIC, FAST TCP, TCP Veno, and Dynamic Transmission Control Protocol (DTCP) to support high Bandwidth-Delay product (BDP) over the cloud network [12-14] [18-20]. However, the network performance needs further improvements due to social networking and other web-based applications developed in recent years, which require high bandwidth and reduced delay for efficient communication.

Traditional packet loss-based congestion algorithms, namely, TCP Reno, New Reno, and Tahoe provides a solution to these problems. Moreover, the existing algorithms introduce a new parameter called the congestion window, which minimizes packet loss. According to mobile users, the size of the congestion window either increases or decreases rapidly the window size adjusted by the slow start, fast retransmission, and recovery algorithms [15]. However, better adjustment techniques are necessary to suit the mobile cloud-based video communication.

The proposed work examines the algorithms and not dynamically adjust the window size among the existing algorithm and propose a new algorithm for handling the issues. Our proposed algorithm, called hybrid Estimated Bandwidth Congestion Window (EBCW), performs the dynamic window size modification precisely on the receiver side to minimize the congestion and packet loss. While changing the congestion window, in any case, there is a possibility for failure in congestion control due to massive load packets, lossy channels, or High-Bandwidth Delay Product (BDP) [16]. Hence, packet loss and delay are the parameters must be kept in mind while developing protocols for video communication as considered in other hybrid approaches [17]. The TCP congestion algorithms cannot provide an appropriate solution for loss, delay, and goodput. The challenges have motivated to propose CNBE, achieves high goodput compared with the existing congestion control techniques.

* Corresponding Author: S.P. Tamizh selvi

ISSN: 0303-6286
Contributions:

1. We proposed the CNBE algorithm to determine the busy period of mobile and dynamically estimate the bandwidth of network in the cloud.
2. We proposed a hybrid congestion avoidance algorithm EBCW to adjust the congestion window periodically as per the estimated bandwidth.
3. The proposed algorithm minimizes the packet loss, and increases the goodput, throughput compared with other existing algorithm.
4. We compared our proposed CNBE and EBCW algorithm with the existing congestion control algorithms (such as Vegas, Reno, NewReno, Tahoe, TCPW, DTCP) during different network busy period with the condition low and high.

The paper is structured as follows. Section 2 briefly covers the background on various TCP congestion control algorithms. Section 3 gives a detailed description of the proposed framework for Network bandwidth estimation and congestion avoidance for video data in the mobile cloud. Section 4 provides the experimental setup with the results and discussion for various parameters. Finally, Section 5 concludes with the salient features of the proposed work and suggests some future work.

2. Background

In computer networks, TCP congestion control algorithms helps to control the sending rate and the flow of data from both the ends like sender to the receiver network node. Then, TCP decide to send data at a given time and how much data to be sent for fast and reliable data delivery. Congestion control algorithms [21] are needed to utilize network bandwidth efficiently and provide low network data loss rates. Congestion occurs in heterogeneous networks due to bandwidth and RTT [22] [23]. The size of the window and slow start threshold manage data flow in the sender side to control the data transmission for acknowledgment. Then, the window is increased by 1 till it reaches the slow start threshold value. To avoid the congestion window (cwnd), 1/cwnd increases for every acknowledgment which has been proposed by [24]. TCP congestion algorithm improves the forwarding packets, detect the loss of packets, network delay, goodput and throughput.

Van Jacobson introduced a TCP congestion control algorithm named, Tahoe to preserve the packets and retain the congestion window, and network capacity [25]. When the packet loss happens, Tahoe will retransmit the packet after the timeout. To solve the issue of Tahoe, the Reno TCP congestion control algorithm proposed by Floyd. The algorithm follows similar principles as Tahoe, but detect the packet loss earlier [26]. Reno applies a Fast retransmit approach called Additive Increase Multiplicative Decrease (AIMD), and detects a single packet loss. The drawback of Reno is reducing the congestion window and detecting multiple packet loss does not happen.

To solve the above issues, Floyd proposed another new algorithm, namely, New Reno. The modified version of Reno is NewReno, effectively detects multiple packet loss and fastly retransmit packets after receiving the duplicate acknowledgments.

The main difference between the two algorithms is in the reduction of the congestion window and multiple retransmission of packets is allowed in New Reno whereas, it is not possible in Reno. In order to

* Corresponding Author: S.P. Tamizh Selvi
overcome the issues of Reno, the author modified the Reno algorithm to provide TCP Vegas. The algorithm adjusts the slow start mechanism to prevent congestion. The merit of Vegas is that it detects the congestion before the packet loss and utilizes the estimated bandwidth, delay, and RTT [27].

The extended version of TCPReno is TCPW and this model applies the slow start threshold (ssthresh) and BWE (Bandwidth Estimation) to control the congestion flow [28]. TCPW estimates the sender available bandwidth dynamically by monitoring and equating the flow of receiving ACKs. When fast retransmission or retransmission timeout occurs, the sender modifies the cwnd and threshold (ssthresh) values to the estimated bandwidth.

Traditional congestion control algorithms such as Tahoe, Reno, New Reno and TCPW suffer different issues on network traffic, fixed bandwidth, congestion window based on thresholds, packet loss, queue delay, reduction in goodput, and throughput. Therefore, these issues motivated us to provide the solution of hybrid TCP variants. Hence, we propose a new framework called a framework of Network bandwidth estimation and congestion window for video data in the mobile cloud to deliver a better quality of video with the estimated bandwidth and avoid the congestion.

3. Network bandwidth estimation and congestion avoidance for video data in the mobile Cloud

In this work, Figure 1 proposes a Network bandwidth estimation and congestion window for video data in the mobile cloud, which comprises two major components (1) Mobile client (2) Cloud server. The Mobile client casts the various network Quality of Service (QoS) issues such as network failure, signal strength, heavy traffic, low quality of video in live streaming, battery, power consumption, and low bandwidth. The cloud server must be combined with the mobile to overcome the issues of the mobile client.

3.1 Mobile Client

This component is constituted with two parts, as follows: (I). Adaptive decoder and (II). Video player. The mobile sends the video request to the cloud server. The server processes the video request, and it downloads the video file without buffering, packet loss, and good quality of the video. The requirement satisfies network traffic status with Low and High monitored by the Cloud, depending on the mobile's actual bandwidth. The cloud server maintains the history of mobile bandwidth to deliver a good quality of video. Hence, the mobile client acts as a front end to send the request and receives the video. Later, it decodes by an adaptive decoder and finally viewed in the video player format as H.265 of mobile.

3.2. Cloud server

The cloud server consists of three components (1) Device Profile Manager (DPM) (2) Network traffic Cloud estimator and (3) Stream engine. A detailed description of components as follows:

3.2.1. Device Profile Manager (DPM)

The cloud monitors the mobile user-activated services, and the device manager maintains the various QoS parameters such as bandwidth, link capacity, network status, signal strength, buffering stream, and

* Corresponding Author: S.P. Tamizhselvi1
energy. Whenever a device state change occurs, the device information updates in the cloud server for the registered mobile users. The Device Profile Manager (DPM) saves a copy of the mobile user profile preferences and mobile details.

The following steps give the aspect of job processing for DPM are as follows:

1. DPM manages and synchronizes the device profile with the cloud server.

2. DPM maintains the mobile history, and it explores the capabilities of the device to adapt to the display services of mobile. The DPM supports data about several new methods in the container connected to the server and ensures the database's consistency.

![Diagram of Network bandwidth estimation and congestion avoidance for video data in the mobile cloud](image)

**Figure 1**: A framework of Network bandwidth estimation and congestion avoidance for video data in the mobile cloud

### 3.2.2. Proposed Mobile Network Cloud Estimator

The proposed mobile network cloud estimator performs two tasks: (1) Cloud Network Bandwidth Estimation (CNBE), and (2) Estimate Bandwidth Congestion Window (EBCW). The working principle of the two tasks given below: The cloud predicts the network bandwidth using a mobile profile for different network traffics. The available bandwidth of the mobile decides the data transfer and the time required to send the packets. The algorithm estimates the bandwidth to utilize it to the maximum, which reduces packet loss and buffer. After the estimation, the sender’s side fit the congestion window in cloud using EBCW.

* Corresponding Author: S.P. Tamizhselvi¹
**Algorithm 1:** Cloud Network Bandwidth Estimation (CNBE) algorithm

**Input:**
- \(N_T\) - Network Busy, \(NT_L\) - Network Busy Limit
- \(AT_j\) - Arrival Time, \(T_d\) - Delay
- \(NV_p\) - Number of video packets
- \(\Delta_{\text{max}}\) - Maximum inter arrival time, \(Q_L\) - Length of Queue,
- \(CV_p\) - Current Video Packet, \(B\) - Busy period

**Output:** Estimate Bandwidth (\(E_{bw}\))

```plaintext
function CNBE(\(AT_j\), \(T_d\), \(NV_p\), \(NT_L\), \(CV_p\))
    Initialize \(N_T \leftarrow 0\);
    \(B \leftarrow 0\);
    for \(j \leftarrow 1\) to \(NV_p - 1\) do
        if \(NT_L < A_{\text{CB}_u}\) then // Actual bandwidth
            if \(AT_{j+1} - AT_j = T_d \&\& N_T = 0\) then
                \(Q_L \leftarrow 0\);
                \(N_T \leftarrow 1\);
            else if \(AT_{j+1} - AT_j > \Delta_{\text{max}} \&\& N_T = 1\) then
                \(N_T \leftarrow 0\);
                \(B \leftarrow 0\);
            else if \(N_T = 1\) then
                \(Q_L \leftarrow AT_{j+1} - AT_j - T_d + Q_L\);
                if \(Q_L \geq NT_L \&\& B = 0\) then
                    \(CV_p \leftarrow AT_j\);
                    \(B \leftarrow 1\);
                else if \(Q_L < NT_L \&\& B = 1\) then
                    \(T_p = \frac{\sum(CV_p, AT_j)}{AT_j - CV_p}\)
                    \(B \leftarrow 0\);
        end
    end
    return \(E_{bw} \leftarrow \text{Avg}(T_p)\);
```

* Corresponding Author: S.P. Tamizh selvi

**ISSN:** 0303-6286

**Vol 41, 2021**
CNBE algorithm is mainly designed to identify the network busy based on the queue length. The Cloud applies the queue to handle the requesting packets of mobile up to the maximum limit. When the queue is full, the downloading time difference is also identified in the given algorithm. Different notations are defined initially in the proposed algorithm namely $T_d$, $NT_L$, and the target delay is determined between the arrival of requested video packets, and the Network threshold identifies the network busy limit and the $\Delta_{max}$ finds the maximum in the queue respectively. Moreover, $AT_j$ is the requested video packet time of the $j^{th}$ packet which is received in the smartphone, $AT_{j+1}$ is the time taken for the video packet arrival as $(j + 1)$ the packet and $NV_p$ specify the number of video packets in the network busy limit. The Boolean variables $NT$ and $B$ indicates the active network busy and the estimated bandwidth.

The step 4 checks the network traffic limit is less than the actual bandwidth, then the condition is satisfied means, step 5 analyse the arrival packet and the delay to assign the queue length and network busy limit. The maximum inter arrival time is processed in step 8 and the bandwidth becomes 0. Thus, step 6 to 10 determines the network traffic limit. Step 12 -15 determines the queue length and current video packet. Based on queue length and network traffic limit, the total video packets to be streamed is calculated by applying step 17. Finally, estimated bandwidth is returned by finding the average of total packets in step 22.

**Algorithm 2: Estimate Bandwidth Congestion Window (EBCW)**

**Input:** Estimate Bandwidth ($Eb_w$)

**Output:** Target Congestion Window ($T_{cwnd}$)

1. Initialize $Max_{cwnd} \leftarrow 64$ Kbps
2. if 3 DUPACKs are received then
3. $QL \leftarrow (Eb_w \times \Delta_{max}) / NV_p$
4. end
5. /* congestion avoid. */
6. if $Max_{cwnd} > Q_L$ then
7. $T_{cwnd} = Q_L$
8. end

EBCW Description: EBCW algorithm takes the estimated bandwidth as input from the algorithm. The maximum mobile window size initializes to 64 kbps. The cloud server receives three duplicate acknowledgments to calculate $Q_L$ in step 3. The cloud fits the dynamic congestion window to avoid congestion by comparing the $Max_{cwnd}$ with $Q_L$. Then, assign the queue length of the target congestion window $T_{cwnd}$ in step 6. The cloud server minimizes the packet loss as per the estimated bandwidth and congestion window. After estimation, the cloud server process the video packets in stream engine and transcode it. Then the result is passed to adaptive decoder and good quality of video is delivered in the mobile without any congestion and packet loss.

* Corresponding Author: S.P. Tamizhselvi1
4. Experimental setup and Results Discussion

This section provides a detailed description of the experimental setup, implementation of mobile client and public cloud in the Table 1.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Simulation</th>
<th>Public Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel ® Core ™ i7-930 CPU with 2.8 GHz</td>
<td>Intel® Xeon® CPU E5 2686 v4 2.3 GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>8 GB</td>
<td>4 GB</td>
</tr>
<tr>
<td>Storage</td>
<td>500 GB</td>
<td>300 GB</td>
</tr>
<tr>
<td>Operating System</td>
<td>Windows 10 64-bit</td>
<td>Ubuntu 14.04.5 LTS Server</td>
</tr>
<tr>
<td>Video Standard</td>
<td>H.265</td>
<td>H.265</td>
</tr>
<tr>
<td>Software</td>
<td>NS3, Netbeans 7.0, CloudSim</td>
<td>AWS Device Farm, S3, Amazon CloudFront, Amazon EBS</td>
</tr>
</tbody>
</table>

Next, the network QoS parameters such as Bandwidth estimation, congestion window, Packet loss rate, goodput and throughput are experimentally evaluated and the results are discussed in the following section.

4.1. Bandwidth Estimation

Figure 2 (a) (b) depicts the network busy load different TCP variants for investigating the of the status as low, and high. In the figure, in low bandwidth network status, we noticed the actual bandwidth of mobile fixed as 9.0 Mbps. Our proposed CNBE utilizes the most available bandwidth compare with other TCP variants. Because the estimated busy depends on network status. In the meanwhile figure (b) shows the fluctuation in estimated bandwidth. CNBE consumes the nearby actual bandwidth as many times more than other variants. In some scenario, TCPW, Reno overestimates the actual bandwidth and incurs packet loss. To solve problem CNBE estimates the best bandwidth of the mobile in cloud environment which reduces the packet loss.
4.2. Estimating Congestion window

Figure 3(a), (b) shows the dynamic variation of congestion window of packets with a different time interval. We found that congestion window is detected for TCPW and CNBE as per the estimation of bandwidth in the fast recovery algorithm. Therefore, the maximum congestion window size varies according to the network status low, high as 2500 and 1600 between the time intervals 100 to 700 seconds. The analysis clearly depicts, in both cases, the Vegas congestion window is minimal, and thereby exponential increase in packet loss incurs heavy congestion. In contrast, TCPW congestion window is maximized as 1/cwnd, wherever the new acknowledgement is received compared with other existing algorithm. Our EBCW dynamically changes the window as per the new acknowledgement and it purely depends on busy status and bandwidth estimation to minimize the window to avoid the congestion and packet loss.

* Corresponding Author: S.P. Tamizhselvi¹
4.3. Packet Loss Rate

Figure 4 shows the PLR and average throughput. PLR varies from 0.00 to 0.12 with a maximum delay of 80 ms. We noticed in the figure CNBE achieves similar throughput other than the existing algorithm. The reason behind this, the network busy varies based on the estimated bandwidth and congestion window of the mobile. The two parameters reduce PLR on the receiver side. The reduction of PLR increases the average throughput.
4.4. Goodput

Figure 5 (a), (b), depicts the goodput versus RTT for different TCP variants. Goodput is defined as sent the total number of packets to the receiver successfully within a particular time interval. Therefore, the packet loss ratio must be minimal for excellent goodput. The quality concern of the busy network rate determines the packet loss. Figure 5(a), (b), achieves the packet loss rate is 10%, 70% in the network busy low, and high. We noticed the PLR was only 10% in the low network because bandwidth increased infers the high goodput. But for the high network, the PLR varies by 70%, which drastically reduces the goodput 1.3 Mbps in the 400 seconds RTT. The reason behind this goodput is PLR depends on network busy and bandwidth. The estimated bandwidth and the congestion window is dynamically fixed to evaluate the goodput with different RTT compare with existing algorithms.
4.5. Throughput

Figure 6(a) (b) shows the throughput for random fluctuation with a different RTT of various TCP algorithms on low and high network. Tahoe, New Reno, Reno, is an existing algorithm that increases the congestion window using an exponential slow start mechanism. New Reno and Reno fix the window size as stable, and the packet loss occurrence ratio is high. So there is a possibility to recover the lost packets. Due to this fact, the two variants do not produce high throughput. In New Reno, packet loss occurs not only based on cwnd but also the bandwidth's link. Even though in the network fluctuation, the Tahoe provide considerable throughput. Our CNBE provides high throughput on low network and vice versa compare with other earlier TCP variants because it does not apply any exponential slow start to increase the count and randomly sets the queue length based on the network busy.

* Corresponding Author: S.P. Tamizhselvi¹
This paper presented a framework of Network bandwidth estimation and congestion window for video data in the mobile cloud. CNBE and EBCW are the two proposed algorithms used to estimate network traffic and fit the congestion window. Our experimental results of CNBE employs the best available bandwidth on a low and high network. EBCW sends the maximum number of packets according to estimated bandwidth and queue length to minimize congestion and packet loss. Our proposed algorithm improve the goodput, throughput and reduce the PLR to deliver the best video quality on mobile. Our work extends in a cloud router to improve TCP variant’s performance in the mobile cloud environment.

* Corresponding Author: S.P. Tamizhselvi

5. Conclusion

Figure 6: Comparison of Throughput for TCP Variants on (a) Low and (b) High network
References


* Corresponding Author: S.P. Tamizhselvi


* Corresponding Author: S.P. Tamizhselvi1