Packet Drop Rate and Round Trip Time Analysis of TCP Congestion Control Algorithm in a Cloud Based Collaborative Virtual Environment

Abdulsalam Ya’u Gital\textsuperscript{1,2}, Abdul Samad Ismail\textsuperscript{1}, Haruna Chiroma\textsuperscript{3,5} and Sanah Abdullahi Muaz\textsuperscript{4,6}

\textsuperscript{1}Department of Computer Science, Universiti Teknologi Malaysia, Malaysia
\textsuperscript{2}Mathematical Sciences Department, Abubakar Tafawa Balewa University, Bauchi, Nigeria
\textsuperscript{3}Department of Artificial Intelligence, \textsuperscript{4}Department of Software Engineering, University of Malaya, Malaysia
\textsuperscript{5}Federal College of Education (Technical), Gombe, Nigeria
\textsuperscript{6}Bayero University Kano, Nigeria

agital@acm.org, abdsamad@utm.my, hchiroma@acm.org, samaaz.csc@buk.edu.ng

Abstract—Collaborative Virtual Environment (CVE) has become popular in the last few years. In CVE the state of the virtual objects is witnessing rapid change. When a user performs an action in CVE, the information of the action needs to be transmitted to other users to maintain consistency in the cooperative work. Currently, in the design of most CVE systems, TCP is used for its reliability. When packet loss and round trip time are high during data transmission, it affects the performance of the CVE systems. This result to inconsistent state of virtual world even with the later retransmission of the lost packets. This paper implements a cloud based architecture for improving consistency and scalability in CVE systems proposed by Gital et al. (2014). We analyze and compares the drop rate and RTT of different TCP variants with CVE architecture. Our simulation results show that Vegas, Newreno and Reno can be promising TCP variants in the cloud based CVE systems compared to other variants. The Vegas, Newreno and Reno can effectively be used as an alternative to other variants of the TCP in efficient applications of the CVE.

Keywords— TCP Variants; Packet Drop Rates; Round Trip Time, Collaborative Virtual Environment; Cloud based Architectural model.

I. INTRODUCTION

The use of CVE applications minimizes cost of training thousand participants located in different geographical locations concurrently. The CVE allows these participants to share a common virtual environment, including virtual entities and resources maintained by a group of computers, in such a way that it can support effective communication between them to achieve synergistic coordination of tasks [1-3]. Applications of CVEs include military training, industrial remote training, Education, massively multiplayer online games, and collaborative engineering [4-7]. As the number of concurrent participants is becoming larger, data transmission alone the network may experience high packet drop rate because of congestion and will no longer provide the level of consistency required [1, 7, 8]. The round trip time may also rise due to network congestion, that will affect the average performance of the systems and results in inconsistency situation. TCP is the widely used transport protocol that provides reliable packet delivery over an unreliable network in the current TCP/IP network. Most virtual collaborative applications have been designed with protocols that provide timing of data transmission due to the consistency requirement in the system. Users in CVE cooperate with each other and interact with the virtual environment; the state of the virtual environment is changing fast, how to transmit the interpretation of the user level interaction in the network is a challenging task. CVE is more concern about the performance of the virtual world. Consistency is more critical since high drop rate may lead to very poor state of the virtual world at each terminal. With the increasing speed of network and readily available internet, bandwidth is no longer a limiting factor in the internet. CVE application today are built over TCP, because TCP provides sequence deliverance of data and unfailing data transmission among communicating nodes. One of the strengths of TCP is its high responsiveness toward network congestion. TCP is also a defensive protocol as it detects incident congestion as its result to try and lessen the impacts of the congestion by preventing the collapse of communication [9]. The TCP focuses on reliability, stability, and correctness of data transfer which fits well with requirements of loss sensitive applications such as web browsing and file transfer and left with the problem of delay due to high drop rate in CVE applications. Packet loss in a network affects the quality of service...
consideration if it is high. The percentage of packet loss that can be tolerated ranges between 0-0.1 percent. Any percentage loss rate above 0.1 percent may have an impact [10]. The reliability and stability comes at the cost of variable delays in data transmission that can create problems for TCP [11].

II. LITERATURE REVIEW

As different implementations of TCP protocols have been introduced, analysis and evaluation studies have been conducted to measure the performance of different TCP variants. For example, [12] compare the performances of different TCP variants with the routing protocols DSDV and AODV, experimented in 20 different ways and find out that TCP Tahoe has the least number of packet drops against the simulation time. Some of the other variants even though they started with a lesser number of packet drops, the TCP Tahoe variant has always the least amount of packet drops in all cases when using AODV and DSDV. [13] study the performance of TCP Vegas versus different TCP variants in homogeneous and heterogeneous wired networks are performed via simulation experiment using network simulator 2 (ns-2). The performance of TCP Vegas outperforms other TCP variants in the homogeneous wired network. However, it achieves unfair throughput in heterogeneous wired network. [14] presents a comprehensive experimental analysis of TCP variants under MPLS with emphasis on Tahoe, Reno and Vegas under different traffic load. It has been found that Reno and Tahoe fail to take advantage of MPLS features whereas Vegas has shown promising results with almost stable, constant end-to-end delay after a transient. [15] compared TCP Tahoe, NewReno, Vegas, and Sack. TCP Vegas showed better throughput than Reno. In this review, none of the studies compare the performance analysis of TCP variants in cloud based CVE architecture. In this paper, we present a performance analysis of different TCP variants with cloud based CVE architectural model [16] to determine the suitability of each variant for CVE systems. The performance metric used in this paper is packet drop rate and round trip time.

III. CVE ARCHITECTURE BASED ON CLOUD COMPUTING MODEL

The architecture of the cloud based CVE is proposed by the modification of the previous CVE. The CVE moves into the cloud instead of the conventional environment presently in used. The cloud based CVE comprises of the cloud infrastructure. This layer enables the provision of networking components, servers, storage, routers, and switches. The Cloud Computing infrastructure heavily influences application performance and throughput in a distributed computing environment [17]. It is responsible for hosting and given supportive coordination to infrastructures including the platform of cloud, repositories, computers, servers, network communication devices, storage units among other physical structures like building. The resources of the information and communication technology are distributed by the cloud infrastructure. The cloud platform provides both services and inters connections among the systems on the platforms as well as to provide an easy way for the system's hardware to operate just like the Internet [18]. Furthermore, the cloud infrastructure allowed the hardware to securely access data in a sharable platform. Data transmission between the different components of the cloud architectural model uses UDP and TCP as the transport layer protocol.

IV. TYPE OF CVE APPLICATION AND DATA TYPE

Collaborative application can be categorized according to the nature of the problem at hand. Most of these collaborative applications fall into the following six groups:

(i) Collaborative work environments (for conducting collaborative work such as military training, engineering design, visualization, documentation, etc.).
(ii) Meetings, seminars and conferences over the internet.
(iii) Simulation of face-to-face contacts where visual quality is critical (such as recruitment interviews, medical diagnoses and remote surgical operations).
(iv) Distance learning environments (for providing course materials, holding a tutorial, carrying out a team project, and conducting an examination).
(v) Networked computer games.
(vi) Leisure and entertainment (including 3D navigation and virtual embodiment) etc.

Similar to another update message in standard applications, message update in CVE have strong delay requirements. According to [19], the delay in update transmission in CVE should not exceed 100msec. [20] argue that up to 200msec delay is acceptable. This requirement is based on the real time and highly reactive multiplayer processes where users' actions are based on action of another user; therefore requiring a very low transmission delay of updates. Also, collaborative update messages are severely affected by jitter. It has been shown that a CVE session with a low 10msec delay that has jitter, results in a collaborative environment which is almost as bad as one with 200msec delay but no jitter [20]. Finally, collaborative update messages have strict reliability requirements. It is obviously pertinent that all users receive update messages or they won't be able to collaborate. In a typical CVE system, the last state of a shared object is the most crucial data. For example, if a user moves an object, generating 10 update messages each 50msec apart. If the update messages 1, 7 and 9 are lost; there is no need of retransmission because the last state of
the object is received correctly. In CVE systems, a server executes the function of: receiving the update messages from the clients; updating the whole virtual environment and transmitting updates of the virtual environment to other clients and servers to keep consistency in the virtual environment. A client also executes functions to: receiving the users input as the update message; transmitting the update message to the server and receiving the update messages from the server to keep consistency in the virtual environment. The size of the message packets transmitted for these functions in CVE are mostly uniformly sized (80, bytes).

V. OVERVIEW OF SOME TCP VARIANTS

TCP Tahoe [21-24] by Van Jacobson is a method based on the principle of packet conservation. Packets get into the network only when there is bandwidth available. This principle is implemented by using acknowledgement. By sending acknowledgement, it means that a packet has reached its destination, leaving available bandwidth it occupies for sending another packet. It also maintains a congestion window CWD to reflect the network capacity. TCP Tahoe suggests that whenever a TCP connection starts or re-starts after a packet loss it should go through a procedure called slow-start. Slow start suggests that the sender sets the congestion window to 1 and then for each ACK received it increase the CWD by 1. The important thing is that Tahoe detects packet losses by timeouts. The sender is notified that congestion has occurred based on the packet loss. TCP Reno [21, 23, 24], also known as standard TCP is the most widely adopted Internet TCP protocol. The method uses the four phases of transmission: slow start, congestion avoidance, fast retransmit, and fast recovery. Link congestion is indicated by either receives of duplicate acknowledgment or expiration of retransmission time out (RTO). However, the performance of TCP Reno suffers in case of multiple packet losses within a window of data [22, 25, 26]. TCP NewReno [27] is a modification of TCP Reno. It improves the retransmission process during the fast recovery phase of TCP Reno. TCP NewReno can detect multiple packet losses. It does not exit the fast recovery phase until all unacknowledged segments at the time of fast recovery are acknowledged [21, 28, 29], [30]. The critical issue in TCP New Reno is that it is capable of handling multiple packet losses in a single window. It is limited to detecting and responding only one packet loss per RTT. This insufficiency becomes more distinct as the delay-bandwidth becomes greater. [28, 31].

TCP Vegas [32] is a modification of TCP Reno [32]. It builds on the fact that proactive measures to encounter congestion is much more efficient than reactive measures. Vegas tried to get around the problem of coarse grain timeouts by suggesting an algorithm which checks for timeouts at a very efficient schedule [32]. Also, it overcomes the problem of requiring enough duplicate acknowledgements to detect a packet loss, and suggests a modified slow start algorithm which prevents it from congesting the network [21, 26]. The three major changes induced by Vegas are: New Re-Transmission Mechanism, Congestion avoidance and Modified Slow-start [21, 32]. SACK [33] algorithm is an extension of the standard TCP. It allows a TCP receiver to acknowledge out-of-order segments selectively rather than cumulatively by acknowledging the last correctly in order received segment. The sender retransmits only missing segment after receiving acknowledgement of out of order packets from the receiver. It does not send the entire unacknowledged segment. A TCP SACK behavior is similar to that of TCP Tahoe and TCP Reno, which are robust in case of out of order packet arrivals [23, 24, 34]. However, it improves the performance of the TCP Reno when there are multiple packet losses. TCP SACK maintains a variable called pipe that represents the estimated number of outstanding packets during fast recovery phase [17]. The sender only sends new or retransmitted data when the estimated number of packets in a router is smaller than the congestion window. The pipe variable is incremented by 1 when the sender either sends a new segment or retransmits old segment. It decreases by 1 when the sender receives the duplicate ACK.

VI. SIMULATION

Ns2 simulator is used for the simulation of our experimental setup on a machine with the following configurations: Intel (R) Core (TM) i5-2410m processor, 2.30GHz speed, 4.00GB RAM with Obuntu operating system. The network topology used in the simulation is flex bell topology shown in Fig. 1. The topology consists of TCP senders, TCP receivers and a pair of routers. The link between the sender’s nodes and routers is termed sender’s link and it is connected to different router because the users are formed from different subnet and each subnet is connected to a router A, while the link between the receivers and router B is called the receiver link. The sender and receiver links represent a local area network (LAN). The link between router A and router B represent the bottleneck link within the cloud in the form of a wide area network (WAN).
The links between the sender’s nodes and the Cloud link are full wired duplex link. The bandwidth of the sender’s links is set to 10Mbps with 10ms delay. The bandwidth of the receiver links that represent the cloud is also set to 10Mbps with 10ms delay. The speed of the cloud and that of local area network of the senders are assumed to be equal. The bottleneck link is set to 2Mbps with 50ms delay to represent a connection to cloud infrastructures. The number of sender’s node which is equivalent to the number of concurrent collaborators, is set to 200 with 2 receivers’ node in the first simulation. In the second simulation, the sender’s nodes increase to 600 with 6 receivers’ node. This setting represents a virtual environment with two partitions and six partitions respectively, with 100 users as server threshold. In this simulation, the throughput of Tahoe, Reno, NewReno, Vegas, Fack, SACK and Linux TCP in a cloud setting is evaluate.

VII. RESULTS AND DISCUSSION

In this simulation, the drop rate and round trip time of TCP Variants (Tahoe, Reno, NewReno, Vegas, Fack, SACK, and Linux) were measured in two different simulations as described in the previous section. The simulation period is set to 100sec in each case. The drop rate and round trip time analysis results of the simulation are shown in Figures 2 and 3.

The average drop rate of the variants with 200 users and 2 servers, and with 600 users and 6 servers are presented in Table II. In the first simulation with 200 users and 2 servers, TCP Vegas has the minimum average number of packet drop, which is 91.2 packet per second. This is followed by TCP Newreno with 91.2 packet drop per second. TCP Linus has the maximum average packet drop per second. When relate to the consistency requirement in CVE systems and with few number of users, TCP Vegas and TCP Newreno are found to be suitable compared to the other variants. When the number of users increases to 600, the behaviour of the variants in terms of packet drop changes which clearly shows an indications of effect of of increase in number of concurrent users. TCP Fack has a better average packet drop with 126 packet per second. This analysis prove that modification in TCP to be suitable for CVE systems is highly required, especially with the current exponential growth in the number of concurrent users. Other metric such as Round Trip Time, Congestion window behavior, link utilization are also required. This paper only analyze...
packet drop rate and round trip time. The analysis of the round trip time is describe in the subsequent paragraph.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Tahoe</th>
<th>Reno</th>
<th>New Reno</th>
<th>Vegas</th>
<th>Sack1</th>
<th>Fack</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>96.46</td>
<td>94.87</td>
<td>92.7</td>
<td>91.2</td>
<td>106.7</td>
<td>68.64</td>
<td>111.5</td>
</tr>
<tr>
<td>600</td>
<td>142.4</td>
<td>141.8</td>
<td>135.7</td>
<td>169.1</td>
<td>174.8</td>
<td>126.2</td>
<td>168.3</td>
</tr>
</tbody>
</table>

In the round trip time analysis, with 200 users, TCP Newreno is better with 169.83 as average round trip time. TCP Tahoe, Reno, Vegas, Sack1, Fack, and Linux gain 376.11, 375.83, 380.9, 357.6, 664.3 and 384.1 ms respectively as average RTT. When the number of users increases from 200 to six hundred, the results shows the effect of the increase in the number of users. Reno gain the minimum RTT with 176.82 ms, followed by Newreno with 244.64 ms (see Table III). Fack gain the highest RTT, this is clear evidence that TCP Fack is not suitable for CVE systems implementation. Tahoe, Vegas, and Linux show a decrease in the RTT with the increase in the number of users. Sack1 RTT increase to 374.53 ms. The overall results show that Newreno and Reno can be promising in CVE systems implementation. Figure 3 illustrates the behaviors of the different TCP variants in the two scenarios. Since the protocols are not designed with collaborative features, there is need for modification to obtain a collaborative TCP variant.

VIII. CONCLUSIONS

The paper analyzed and compares the drop rate and round trip time behaviors of seven different TCP variants to determine the suitability of the protocols for Cloud-based CVE systems implementation. The TCP variants evaluated are: Tahoe, Reno, New Reno, Vegas, Sack1, Fack, and Linux. The variants are evaluated in a Cloud-Based CVE system in the study using ns2 simulator. Comparing the results obtained from the simulation, TCP Vegas, Newreno and Reno are more close to the Cloud-based CVE systems requirement. Better performance can be achieved with little modification to conform to the collaborative features of the Cloud-based CVE systems. Our feature work is to improve the performance of Tahoe, Newreno and Reno to come up with a suitable collaborative TCP variant considering the general Cloud-based CVE system requirements.

REFERENCES


