Journal of Theoretical and Applied Information Technology
10th December 2014. Vol.70 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PERFORMANCE EVALUATION OF TCP CONGESTION CONTROL ALGORITHMS THROUGHPUT FOR CVE BASED ON CLOUD COMPUTING MODEL

[1,4]ABDULSALAM YA'U GITAL, [2]ABDUL SAMAD ISMAIL, [3,5]HARUNA CHIROMA

[1]Department of Computer Science, Universiti Teknologi Malaysia, Skudai Malaysia
[2]Prof., Department of Computer Science, Universiti Teknologi Malaysia, Skudai Malaysia.
[3] Department of Artificial Intelligence, University of Malaya, Kuala Lumpur Malaysia
[4]Department of Mathematical Sciences, Abubakar Tafawa balewa University, Bauchi Nigeria
[5]Department of Computer Science, Federal College of Education (Technical), Gombe, Nigeria
E-mail: [1]asgital@yahoo.com , [2]abdsamad@utm.my [3]hchiroma@acm.org.

ABSTRACT

Collaborative Virtual Environment (CVE) is becoming popular in the last few years; this is because CVE is designed to allow geographically distributed users to work together over the network. Currently, in the development of CVE Systems, Client server architectures with multiple servers are used with TCP as update transmitting transport protocol because of its reliability. With the increasing number of collaborators, the transport protocol is inadequate to meet the system requirements in terms of timely data transmission. The transport protocol (TCP) throughput deteriorates in the network with large delay which leads to unsatisfactory consistency requirement of the CVE systems.We proposed a cloud based architectural model for improving scalability and consistency in CVE in an earlier study. The current paper aims at evaluating and comparing the performance of different TCP variants (Tahoe, Reno, New Reno, Vegas, SACK, Fack and Linux) with the cloud based CVE architecture to determine the suitability of each TCP variant for CVE. A comparative analysis between the different TCP variants is presented in terms of throughput verses elapse time, with increasing number of users in the system. TCP Vegas with the cloud based model was found to be effective for CVE systems based on Cloud Computing .

Keywords: *CVE Architecture, Cloud Computing, TCP Variants (Congestion Control Algorithm).*

1. INTRODUCTION

Collaborative Virtual Environment (CVE) allows participant from distant geographic location 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 among the users to achieve synergistic coordination of tasks [1-3]. Applications of CVEs include Education, massively multiplayer online games (e.g., World of Warcraft), virtual worlds (e.g., Second Life), military training, industrial remote training, and collaborative engineering [4-7]. As the number of concurrent participants is becoming larger, data transmission alone the network may no longer provide the level of consistency required, typically in terms of response time [1, 7, 8].

In the TCP/IP network model, TCP is the widely used transport protocol that provides reliable packet delivery over an unreliable network. This protocol is designed to be used with the Internet Protocol (IP). 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. This is because CVE is more about the performance of the virtual world, consistency is more critical since a delay in data transmission leads to unsatisfactory results. UDP has been criticized for use as the transport layer protocol for its lack of congestion control mechanism [9-11]. With the increasing speed of network and readily available internet, bandwidth is no longer a limiting factor in the internet. The CVE application today are built over TCP. The

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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. Thereby prevent collapse of communication [12]. 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 in time-dependent applications. The reliability and stability comes at the cost of variable delays in data transmission that can create problems for TCP [13].

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, [14] 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. [15] 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. [16] 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.[17] Compared TCP Tahoe, NewReno, Vegas, and Sack overself-similar traffic. They found that NewReno did better than other TCP variants with respect to efficiency and throughput. TCP Vegas showed better throughput than Reno. However, we have not found studies that 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 [18] to determine the suitability of each variant for CVE systems.

The rest of the Organization of the paper is as follows: The Section II briefly describes the cloud based architectural model. The evaluated TCP variants are described in section III. An overview of CVE data types is described in section IV. Section V described the simulation methodology and the performance metric used. Section VI presents the simulations, and Finally, section VII presents the analysis on simulation result and conclude the paper.

2. CLOUD BASED ARCHITECTURAL MODEL

The architecture of the cloud based CVE is proposed by the modification of previous CVE. The CVE moves into the cloud instead of the conventional environment presently in used. The cloud based CVE comprised 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 [19, 20]. 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. 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.

3. OVERVIEW OF SOME SELECTED TCP VARIANTS

3.1 TCP Reno

TCP Reno [21-24], 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). When the sender receives duplicate acknowledgements (ACKs), the sender activates TCP fast retransmit and recovery algorithms and reduces its congestion window size

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

to half. It then linearly increases the congestion window as in TCP Tahoe. This is the same in the case of congestion avoidance. This increase in transmission rate is slower than in the case of a slow start and helps relieve congestion. TCP Reno fast recovery algorithm improves TCP performance in case of a single packet loss within a window of data. However, the performance of TCP Reno suffers in case of multiple packet losses within a window of data [25-27].

3.2 TCP New Reno

TCP NewReno [28] 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 [23, 29, 30]. TCP Reno overcomes the problem of reducing the congestion window size many times when there is time there is multiple packet losses. TCP NewReno maintains the slow start, congestion avoidance, and fast retransmits of TCP Reno. It exits fast recovery after receiving acknowledgement of all unacknowledged segments and then sets the congestion window size to slow start threshold and continues the congestion avoidance phase [31]. TCP New Reno retransmits the next segment after the received partial acknowledgment. (Partial acknowledgments are the acknowledgments that do not acknowledge all outstanding packets at the onset of the fast recovery.)

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. However, still there are situations when stalls can occur if packets are lost in successive windows, like all of the previous versions of TCP New Reno which infer that all lost packets are due to congestion and it may therefore unnecessarily cut the congestion window size when errors occur [29, 32].

3.3 TCP Tahoe

TCP Tahoe [21-23, 26] 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. This is because an initial burst might overpower the network and the connection might never get started. The congestion window size becomes double (Multiplicative Increase) for each transmission until there is congestion in the network. Slow start suggests that the sender sets the congestion window to 1 and then for each ACK received it increase the CWD by 1. This implies that in the first round trip time (RTT) only one packet is sent, and keep doubling after each RTT. When there is congestion, the sending rate and the congestion window are set to 1 and start over again. The important thing is that Tahoe detects packet losses by timeouts. The sender is notified that congestion has occurred based on the packet loss.

3.4 TCP Vegas

TCP Vegas [33] is a modification of TCP Reno [33]. 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 [33]. 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 [23, 27]. The three major changes induced by Vegas are: New Re-Transmission Mechanism, Congestion avoidance and Modified Slow-start. Vegas try extending on the retransmission mechanism of the standard TCP. It monitors when each segment was sent and calculates an estimate of the RTT by monitoring the time it takes for the acknowledgment to get back [23, 33]. During Congestion avoidance, TCP Vegas does not use the loss of segment to signal that there is congestion, instead, it determines congestion by a decrease in sending rate as compared to the expected rate, as a result of large queues building up in the routers. It used Tri-S scheme [23, 33].

In the case of slow-start phase, TCP Vegas differs from the other implementations. This is because at the beginning of each connection, Vegas have no idea of the available bandwidth. It is possible that the bandwidth was overshoot during the exponential increase by a big amount and thus induces congestion. Vegas increases exponentially after every RTT, and calculates the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

actual sending throughput to the expected and when the difference goes above a certain threshold, it exits slow start and enters the congestion avoidance phase [23, 33].

3.5 TCP SACK

SACK [34] 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 [21, 22, 35]. 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 [34]. 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.

The disadvantage with SACK is that currently selective acknowledgments are not provided by the receiver. To implement SACK, there is a need for full implementation of the selective acknowledgment.

3.6 TCP FACK

FACK TCP (TCP with forward acknowledgement) [36] was developed to decouple the congestion control algorithms from the data recovery algorithms. It uses the additional information provided by the SACK option to maintain an explicit measure of the total amount of outstanding data in the network. However, standard TCP Reno and TC SACK both attempt to estimate this by assuming that each drawback received represents one segment that has left the network. The basic concept of Fack mechanism is by considering the greatest sequence number of forward selective acknowledgement as a mark that completely previous segments which unselectively acknowledged were lost. This method improved the recovery process of packets losses significantly. Fack algorithm takes an aggressive method, and considers all unacknowledged holes as lost packets and Sack blocks. FACK implementation improved TCP performance than the traditional approach; it is excessively aggressive if packets have been rearranged in the pipeline, due to these holes between the blocks of Sack, FACK does not designate packet loss in this state.

4. OVERVIEW OF CVE DATA TYPE

Similar to another update message in standard applications, message update in CVE have strong delay requirements. According to [37], the delay in update transmission in CVE should not exceed 100msec. [38] argue that up to 200msec delay is acceptable. This requirement is based on the real time and highly reactive multiuser 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 10 msec delay that has jitter, results in a collaborative environment which is almost as bad as one with 200 msec delay but no jitter [38]. 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, 7and 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 user's 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).

5. SIMULATION METHODOLOGY AND PERFORMANCE METRIC

5.1 Simulation Methodology

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. NS2 is an object

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

oriented, event driven network simulator developed at UC Berkley, it is written in C++ and OTcl programming languages. NS2 simulator is a very good simulator for simulating both local and wide area networks. The network topology used in this 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).

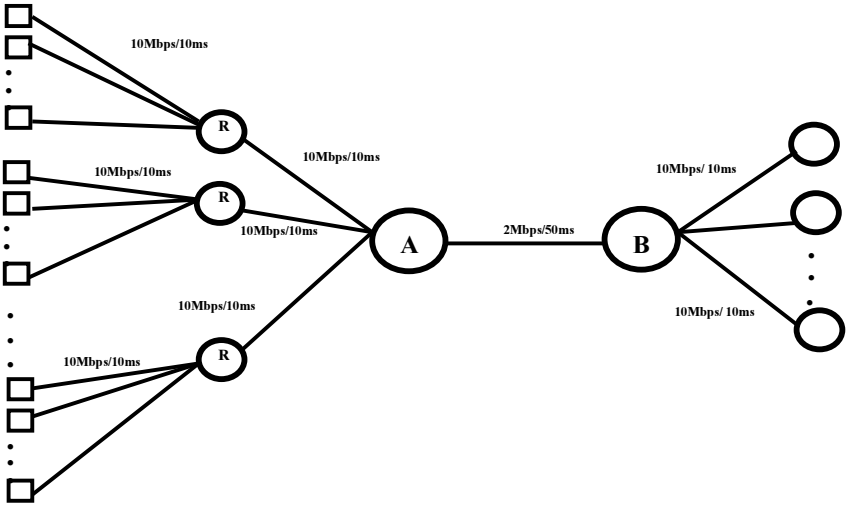


Figure 1 Simulation topology

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 in six different simulations as follows: 200 with 2 receivers' node, 400 with 4 receivers' node, 600 with 6 receivers' node, 800 with 8 receivers' node, and 1000 with 10 receivers' node respectively. This setting represents a virtual environment with ten partitions each hadling 100 users, 100 users is the expected threshold for each server (Receiver node) in the system. The simulation parameters of the network topology are shown in Table I. In this simulation, the throughput of Tahoe, Reno, NewReno, Vegas , Fack, SACK and Linux TCP in a cloud setting is evaluated.

5.2 Performance Metric

The metric use in this simulation is throughput. The throughput is defined as the size of data transmitted from one node to another in a given time. The data transfer rate for network is measured in terms of kilo bits per second, mega bit per second. This paper uses kilo bit per second as for the measurement as shown in the simulation output in figure 2. The expression of the throughput is as follows.

Throughput = (Transfer rate)/ (Transfer time).

Journal of Theoretical and Applied Information Technology
10th December 2014. Vol.70 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TABLE I. SIMULATION PARAMETERS

Link	Bandwidth	Delay	Queue Limit	Window Size	Packet Size	Traffic Types
Link to Cloud (Bottleneck)	2Mbps	50ms	100	8000kb	552B/200B	-
Senders Link	10Mbps	10ms	-	-		Telnet/ CRB
Link to Cloud Infrastructures (Receivers) within the Cloud	10Mbps	10ms	-	-		Telnet/ CRB

6. RESULTS AND DISCUSSION

During this simulation, the throughput of TCP Variants (Tahoe, Reno, NewReno, Vegas, Fack, SACK, and Linux) was measured in six different simulations as described in the previous section. The simulation period is set to 40sec in each case. The throughput analysis results of the simulation are shown in Figure 2-6.

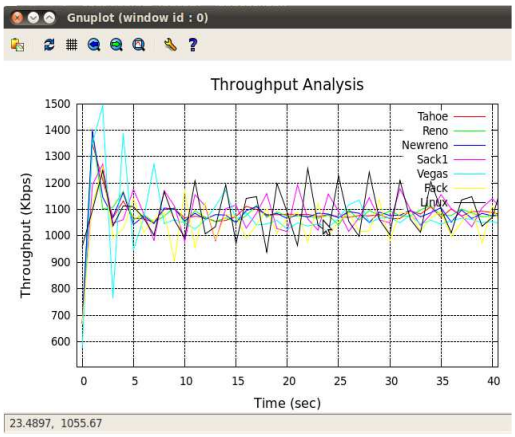
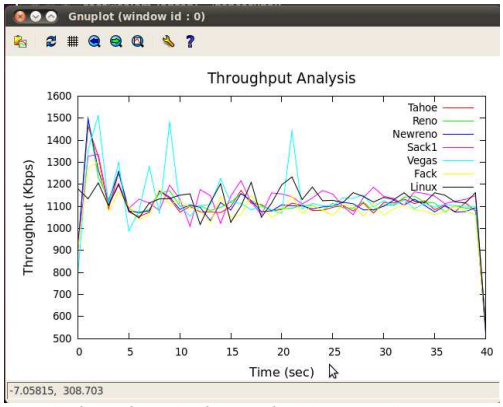


Fig.2 Throughput Analysis with 200 User, 2 Servers In 40 *Seconds*



Fog. 3 Throughput Analysis with 400 User, 4 Servers In 40 Seconds

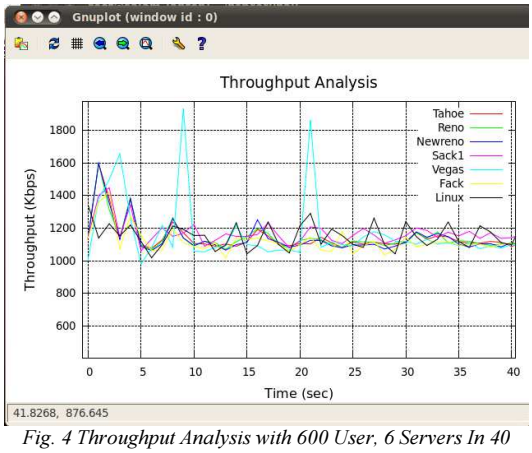


Fig. 4 Throughput Analysis with 600 User, 6 Servers In 40 Seconds

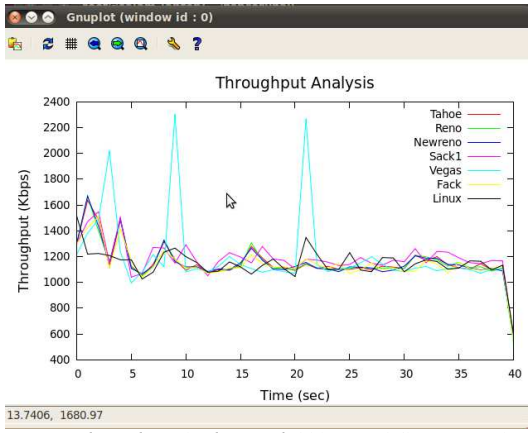


Fig. 5Throughput Analysis with 800 User, 8 Servers In 40 Seconds

Journal of Theoretical and Applied Information Technology
10th December 2014. Vol.70 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

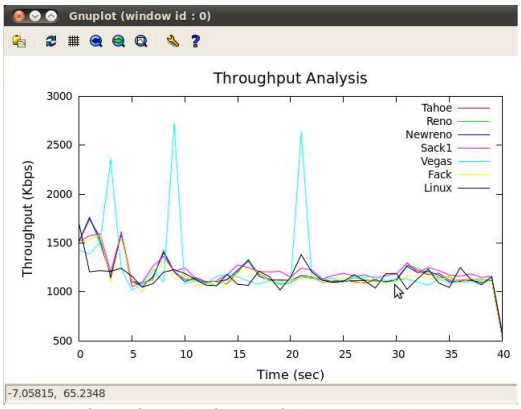


Fig. 6 Throughput Analysis with 1000 User, 10 servers In 40 Seconds

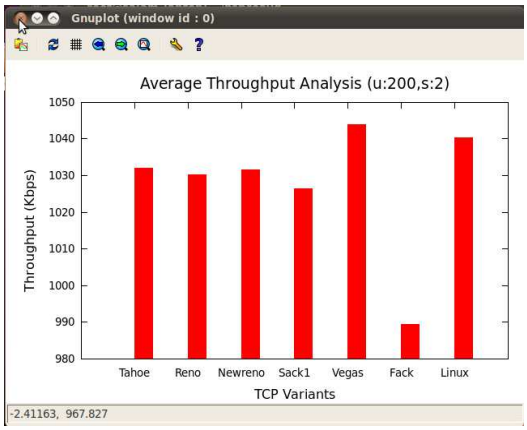


Fig. 7 Average Throughput Analysis with 200 User and 2 Servers

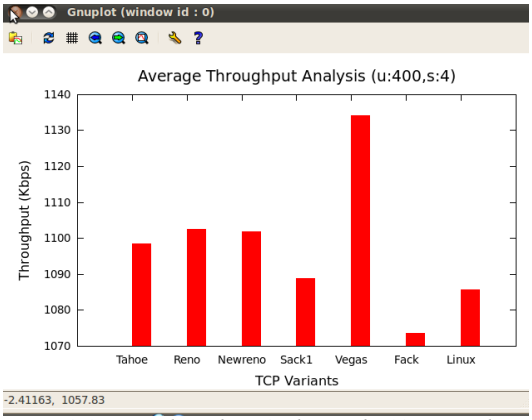


Fig. 8 Average Throughput Analysis with 400 User and 4 Servers

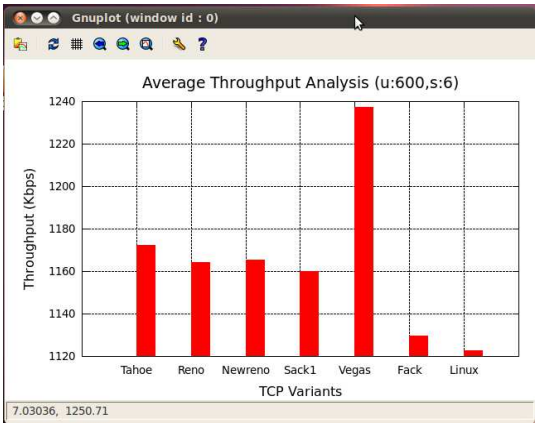


Fig. 9 Average Throughput Analysis with 600 User and 6 Servers

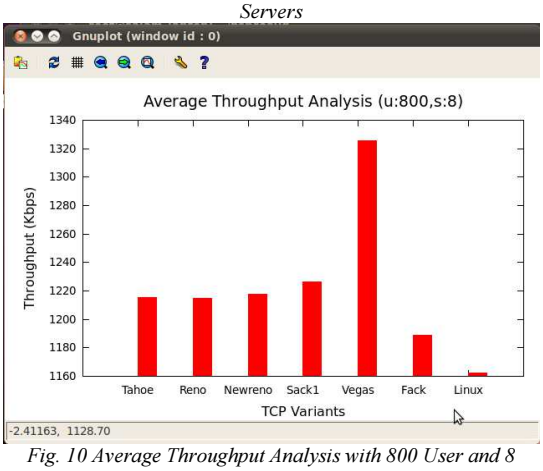


Fig. 10 Average Throughput Analysis with 800 User and 8 Servers

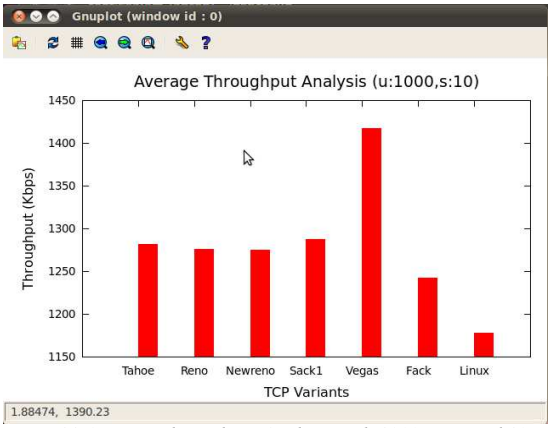


Fig. 11 Average Throughput Analysis with 1000 User and 10 Servers

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TABLE II AVERAGE THROUGHPUT OF THE VARIANTS FOR THE FIVE SIMULATIONS

TCP Variant	200 Users	400 Users	600 Users	800 Users	1000 Users
Tahoe	1032	1199	1175	1218	1288
Reno	1030	1104	1165	1218	1275
NewReno	1031	1103	1166	1220	1275
Sack1	1026	1089	1160	1225	1297
Vegas	1045	1135	1238	1330	1425
Fack	989	1075	1229	1190	1249
Linux	1040	1040	1222	1162	1130

7. CONCLUSION

This paper evaluates and compares the performance of different TCP variants (Tahoe, Reno, New Reno, Vegas, SACK, Fack and Linux) with the cloud based CVE architecture, to determine the suitability of each TCP variant for CVE. A comparative analysis between the different TCP variants is presented in terms of average throughput. With increasing number of users in the system, TCP Vegas with the cloud based model was found to be more effective. The results show that the performance of the TCP Vegas with 200, 400, 600, 800, and 1000 users in cloud based architecture is better than the other TCP variants. This is followed by TCP Linux, Tahoe, NewReno, Reno, Sack1, and Fack, in the first simulation with 200 user. In the second simulation, Vegas performance is followed by that of Reno, NewReno, Tahoe, Sack1, Linux, and latly Fack. In the third simulation, Vegas shows good performance followed by Fack, Linux, Tahoe, NewReno, reno, and Sack1. Other simulation are the forth and fifth. In the fourth simulation it is observed that the performance of Vegas is still at a high point followed by Sack1, NewReno, Reno and Tahoe, Fack, and Linux. Lastly, the last simulation which was performed with 1000 users and 10 severs gernerate throughtput shows that the performance of Vegas is more effective, followed by Sack1, Tahoe, Reno and NewReno, Fack, and Linux.

The performance of other variants different from Vegas keep fluctuating in all the simulations, this indicates an instability in their performance. The performance metrics used in this study are throughput and time. The average throughputs used for evaluating the performance are showm in table II, and the comparison is illustrated in Figure 7-11.

REFERENCES:

[1] X. Hu, L. Liu, and T. Yu, "A hierarchical architecture for improving scalability and consistency in CVE systems," *International Journal of Parallel, Emergent and Distributed Systems,* vol. 26, pp. 179-205, 2011.

[2] X. M. Hu, H. X. Cai, and T. Yu, "A Self-Adaptive Filtering Algorithm Based on Consistency QoS in CVE Systems," *Advanced Materials Research,* vol. 225, pp. 301-306, 2011.

[3] W. Shao-Qing, C. Ling, and C. Gen-Cai, "A framework for Java 3D based collaborative virtual environment," in *Computer Supported Cooperative Work in Design, 2004. Proceedings. The 8th International Conference on*, 2004, pp. 34-39.

[4] S. Benford, C. Greenhalgh, T. Rodden, and J. Pycock, "Collaborative virtual environments," *Communications of the ACM,* vol. 44, pp. 79-85, 2001.

[5] C. Greenhalgh and S. Benford, "MASSIVE: a collaborative virtual environment for teleconferencing," *ACM Transactions on Computer-Human Interaction (TOCHI),* vol. 2, pp. 239-261, 1995.

[6] A. S. Haji-Ismail, M. Chen, and P. W. Grant, "JACIE–an authoring language for WWW-based collaborative applications," *Annals of Software Engineering,* vol. 12, pp. 47-75, 1999.

[7] A. Y. Gital, A. S. bn Ismail, and S. Subramaniam, "On consistency and security issues in collaborative virtual environment systems," *International Journal of Physical Sciences,* vol. 8, pp. 1646-1654, 2013.

[8] Y. Deng and R. W. Lau, "On Delay Adjustment for Dynamic Load Balancing in Distributed Virtual Environments," *Visualization and Computer Graphics, IEEE Transactions on,* vol. 18, pp. 529-537, 2012.

[9] A. Petlund, P. Beskow, J. Pedersen, E. S. Paaby, C. Griwodz, and P. Halvorsen, "Improving SCTP retransmission delays for time-dependent thin streams," *Multimedia Tools and Applications,* vol. 45, pp. 33-60, 2009.

[10] M. N. Khalid, "Simulation Based Comparison of SCTP, DCCP and UDP Using MPEG-4 Traffic Over Mobile

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

WiMAX/IEEE 802.16 e," Thesis no: MSEE-2010-xx May, 2010.

[11] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *ACM SIGCOMM Computer Communication Review*, 1990, pp. 200-208.

[12] M. Kazmi, M. Y. Javed, and M. K. Afzal, "An Overview of Performance Comparison of Different TCP Variants in IP and MPLS Networks," in *Networked Digital Technologies*, ed: Springer, 2011, pp. 120-127.

[13] A. Gurtov, "Effect of delays on TCP performance," in *In Proceedings of IFIP Personal Wireless Communications*, 2001.

[14] A. Pradeep, N. Dhinakaran, and P. Angelin, "Comparison of Drop Rates in Different TCP Variants against Various Routing Protocols," *International Journal of Computer Applications,* vol. 20, 2011.

[15] B. Yew, B. Ong, and R. Ahmad, "Performance Evaluation of TCP Vegas versus Different TCP Variants in Homogeneous and Heterogeneous Networks by Using Network Simulator 2," *International Journal of Electrical & Computer Sciences,* vol. 11, 2011.

[16] M. S. Akbar, S. Z. Ahmed, and M. A. Qadir, "Quantitative Analytical Performance of TCP Variants in IP and MPLS Networks," in *Multitopic Conference, 2006. INMIC'06. IEEE*, 2006, pp. 331-336.

[17] M. Salleh, A. Bakar, and A. Zaki, "Comparison of TCP Variants Over Self-Similar Traffic," 2005.

[18] A. Y. Gital and A. S. Ismail, "A Framework for the Design of Cloud Based Collaborative Virtual Environment Architecture," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2014.

[19] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications,* vol. 1, pp. 7-18, 2010.

[20] A. Y. Gital and A. S. Ismail, "An Alternative Design Of Collaborative Virtual Environment Architecture Based On Cloud Computing," *Journal of Theoretical & Applied Information Technology,* vol. 61, 2014.

[21] Z. F, M. Z, A. O. M, and Z. Zhao, "Throughput Analysis of TCP SACK in comparison to TCP Tahoe, Reno, and New Reno against Constant Rate Assignment (CRA) of 2500 and 4500 bps," *Journal of Computer Science and Computational Mathematics,* vol. 2, pp. 35-41, 2012.

[22] H. Paul, A. K. Saha, P. P. Deb, and P. S. Bhattacharjee, "Comparative Analysis of Different TCP Variants in Mobile Ad-Hoc Network," *International Journal of Computer Applications,* vol. 52, 2012.

[23] B. Yuvaraju and N. N. Chiplunkar, "Scenario Based Performance Analysis of Variants of TCP using NS2-Simulator," *International Journal of Advancements in Technology,* vol. 1, pp. 223-233, 2010.

[24] H. Jamal and K. Sultan, "Performance analysis of tcp congestion control algorithms," *International Journal of Computers and Communications,* vol. 2, 2008.

[25] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *ACM SIGCOMM Computer Communication Review,* vol. 26, pp. 5-21, 1996.

[26] L. Subedi, M. Najiminaini, and L. Trajkovi, "Performance Evaluation of TCP Tahoe, Reno, Reno with SACK, and NewReno Using OPNET Modeler," *Communication Networks Laboratory http://www. ensc. sfu. ca/research/c nl OPNET technologies,* 2008.

[27] M. Podlesny and C. Williamson, "Providing fairness between TCP NewReno and TCP Vegas with RD network services," in *Quality of Service (IWQoS), 2010 18th International Workshop on*, 2010, pp. 1-9.

[28] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm," RFC 2582, April1999.

[29] D. Bisen and S. Sharma, "IMPROVE PERFORMANCE OF TCP NEW RENO OVER MOBILE AD-HOC NETWORK USING ABRA," *International Journal of Wireless & Mobile Networks,* vol. 3, 2011.

[30] B. Qureshi, M. Othman, and N. Hamid, "Progress in various TCP variants (February 2009)," in *Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*, 2009, pp. 1-6.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[31] H. Lee, S.-h. Lee, and Y. Choi, "The influence of the large bandwidth-delay product on TCP Reno, NewReno, and SACK," in *Information Networking, 2001. Proceedings. 15th International Conference on*, 2001, pp. 327-334.

[32] B. Qureshi, M. Othman, and N. Hamid, "Progress in various TCP variants," in *2nd International Conference on Computer, Control and Communication*, 2009, p. 1.

[33] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *Selected Areas in Communications, IEEE Journal on,* vol. 13, pp. 1465-1480, 1995.

[34] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," RFc 2018, October1996.

[35] M. Zorzi and R. R. Rao, "Effect of correlated errors on TCP," in *Proc. 1997 CISS*, 1997, pp. 666-671.

[36] M. Mathis and J. Mahdavi, "TCP rate-halving with bounding parameters," ed, 1996.

[37] M. M. Wloka, "Lag in multiprocessor virtual reality," *Presence: Teleoperators and Virtual Environments,* vol. 4, pp. 50-63, 1995.

[38] K. S. Park and R. V. Kenyon, "Effects of network characteristics on human performance in a collaborative virtual environment," in *Virtual Reality, 1999. Proceedings., IEEE*, 1999, pp. 104-111.