Impact of window decrement rate on TCP performance in an adhoc network

Suherman[1,*], Arief T. W. Hutasuhut[1], Khaldun Badra[1], Marwan Al-Akaidi[2]
[1]Electrical Engineering Department, Universitas Sumatera Utara, Kampus USU Medan 20155, Indonesia
[2]Arab Open University, Kuwait

* suherman@usu.ac.id

Abstract. Transmission control protocol (TCP) is a reliable transport protocol handling end to end connection in TCP/IP stack. It works well in copper or optical fibre link, but experiences increasing delay in wireless network. Further, TCP experiences multiple retransmissions due to higher collision probability within wireless network. The situation may get worsen in an ad hoc network. This paper examines the impact half window or window reduction rate to the overall TCP performances. The evaluation using NS-2 simulator shows that the smaller the window decrement rate results the smaller end to end delay. Delay is reduced to 17.05% in average when window decrement rate decreases. Average jitter also decreases 4.15%, while packet loss is not affected.

1. Introduction

Transmission control protocol (TCP) is one part of protocol stacks building computer network [1]. Its connection is reliable but sensitive to network condition. TCP adjust transmission rate by reducing or increasing its window. Window decrement rate is usually assigned by half window (0.5 x initial window). Window decrement and increment strategies are the reason TCP has many variants, such as TCP Reno and TCP Texas [2].

On the other hand, one of the networks that always give TCP difficulties is an ad hoc network [3]. The ad hoc network is a network configuration of the IEEE 802.11x wireless network which is known as Wireless Fidelity (WiFi). High demands on WiFi network make TCP performance evaluations within ad hoc network gets researcher attentions such as in [4-7]. This paper adjusts half window rates (0.5Win) on standard TCP [8] from 0.1x initial window (0.1Win) to 0.9 x initial window (0.9Win).

2. Research method

In order to evaluate window decrement rates from 0.1Win to 0.9Win, NS-2 simulator is employed. NS-2 is an event-based simulator that is flexible to make changes to some networks and protocols [9]. The simulator is employed for examining the impact of window decrement rates by directly changes the related code within 802.11 scripts. The installed NS-2 runs on Ubuntu 14.04. The evalvid from [10] is employed to evaluate network parameters such as delay, jitter and packet loss.

The evaluated 802.11 network was set by using request to send (RTS) and clear to send (CTS) multi access method, transmission power is set to cover 1000 m distance, which means whole stations are able to listen other stations; no middle temporary router is required. Network is fully in an ad hoc configuration. Modulation is fixed; 64 QAM with two-ray ground propagation model.

Number of nodes is set from 2 nodes to 20 nodes. Each station is assessed in both conditions: transmitting and receiving. The evaluated traffics are video with 300 kbps rate. TCP packet size is set to 1052 bytes.

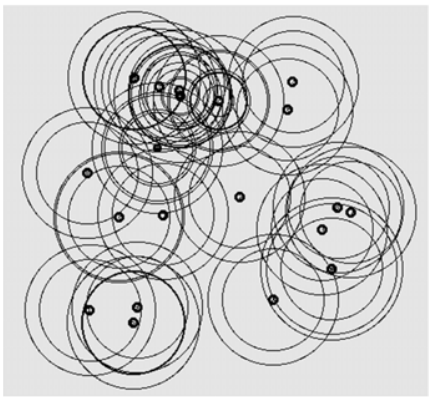

Figure 1. Snapsot of simulation configuration

Figure 1 shows the snapshot of the simulation configuration. Nodes are free moving randomly within the square area 1000x1000 m^2. Each decrement window is tested for 2 to 20 nodes. Each number of nodes is tested 20 times with duration 90 second transmissions.

3. Simulation results

3.1. Impact on end to end delay

End to end delay increases to number of nodes. By changing the value of half window from0.1to 0.9 and recording delay for each node in each simulation, the average delay is obtained as depicted in Figure 2.

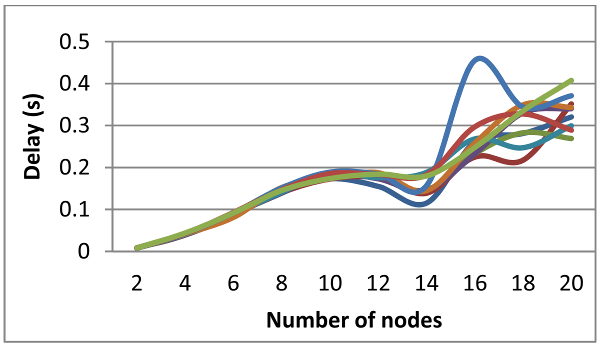

Figure 2. Delay variation for different number of nodes

Up to 10 nodes, each window decrement rate has little differences, however up to 20 nodes, delay varies. This means that window decrement rate be influential if number of the connected node is large. After the delay values for all number of nodes are averaged, the delay pattern to the decrement rate is shown in Figure 3.

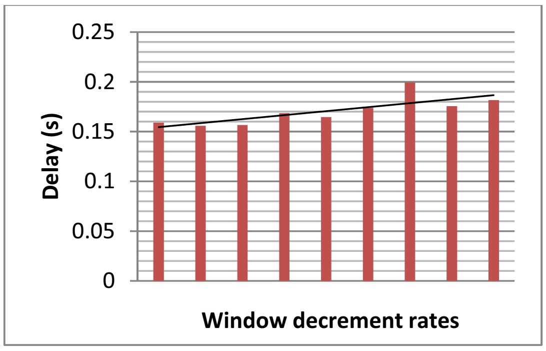

Figure 3. Average delay for each decrement rate

Figure 3 depicts the average delay of all number of nodes snapshot for increasing window decrement rate from 0.1 to 0.9. Increment causes delay increases linearly. This figure suggested that reducing window decrement rate will avoid increasing delay. Reducing decrement rate results 17.05% average delay reduction.

Suppose the duration of TCP timer waiting for the acknowledgement packet is *rto*, the flow size of packet *i*is l_i and *p* is the probability of dropped packet determined by the physical and medium access layers, then the expected response time for the acknowledgement is [11]:

$$r_i = \frac{l_i.rto_i}{1-p}$$

For window decrement rate *k*, then the average expected response time is:

$$r_i' = l_i.rto_i.2^k$$

This means, the smaller value *k*, the smaller *ri'* which causes smaller packet delay. It confirms the simulation results. Figure 4 depicts the example average response time increment to value *k* for packet flow 1024 bytes and *rto* 100 ms.

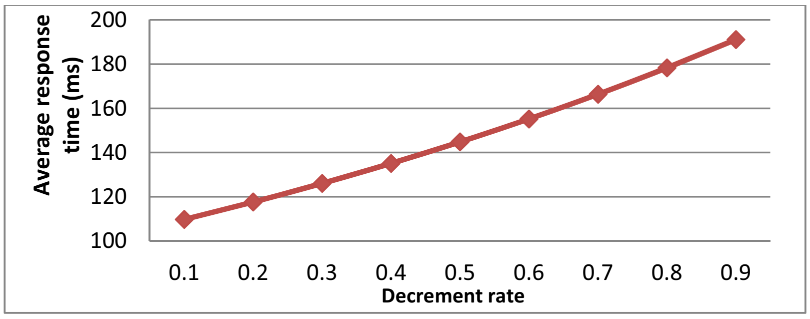

Figure 4. Average acknowledgment response time

However, since small*k* causes significant window reduction as depicted in Figure 5, value *k* should be carefully chosen to avoid high overall delay. Parameter *k* should fit the following equation for given server rate C [11]:

$$k \geq \log_2 \frac{\sum_i^n {}^1/_{rto_i}}{C}$$

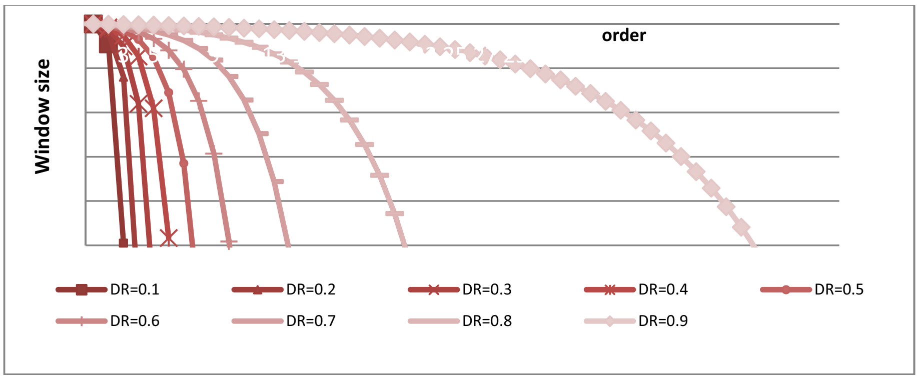

Figure 5. Decrement rate impact to window value

3.2. Impact on jitter

Impact decrement rate on jitter is the same as delay, which is influential if number of nodes large (Figure 6). Jitter pattern to decrement rate varies (Figure 7). Average jitter rises slightly if decrement rate increases. As stated in previous section, lower *k* results lower l_i. Smaller acknowledgement reponse time tends to reduce jitter, although in many cases, subsequent packets may experience different network conditions.

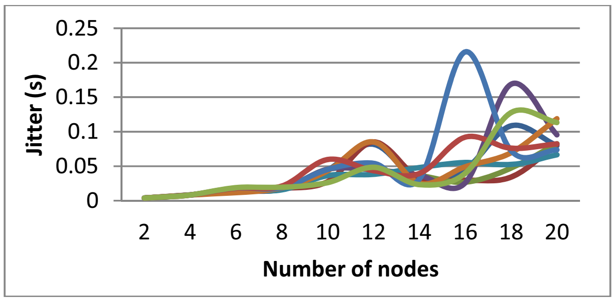

Figure 6. Jitter variation for different number of nodes

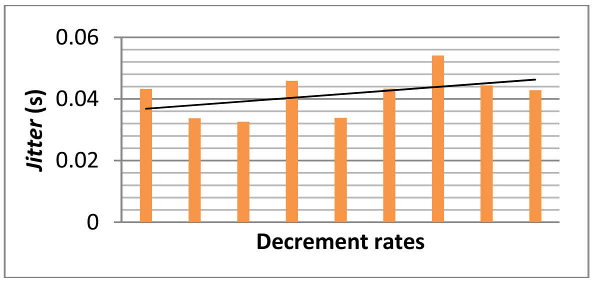

Figure 7. Jitter variation for different decrement rates

3.3. Impact on packet loss

Packet loss is not affected by the change of window as TCP keeps resending the lost packets. The average loss is zero for all conditions.

4. Conclusion

This paper has evaluated impact of changes on window decrement rate. Normally window decrement rate is 0.5 or known as half window. The assessment changes decrement from 0.1 to 0.9. As result, delay reduces in average 17.05% as window decrement rate reduced from 0.9 down to 0.1. Jitter has the same pattern, but smaller average decrement, just about 4.15%. Packet loss is not influenced by the changes as TCP keeps retransmit the lost packets.

Since half window is utilized in all TCP variants, the finding in this paper could be applied to other types of TCP. That would be of future work.

References

[1] Forouzan, Behrouz A., 2000, "TCP/IP Protocol Suite". Singapore: McGraw-Hill International Edition.
[2] Stallings W, 2007, Data and computer communications. Pearson/Prentice Hall.
[3] Suherman, Marwan Al-Akaidi, and Naemah Mubarakah, 2015, "A Transport Layer Protocol for uplink WiMAX video streaming." International Journal of Multimedia & Ubiquitous Engineering, Volume 10, Number 1.
[4] Deshpande, A. and Kaushal, A., September 2016, Feedback-based adaptive speedy transmission (FAST) control protocol to improve the performance of TCP over Ad-Hoc networks. In Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on (pp. 2034-2041). IEEE.
[5] Deshpande, Aniket, and Ashok Kaushal, 2015, "MX-TCP and HS-TCP as Possible Options to Overcome TCP Limitations in Multi-Hop Ad-Hoc Networks." Int. J. Adv. Res. Comput. Commun. Eng. 4.12, pp. 505-508.
[6] Liaqat HB, Xia F, Yang Q, Liu L, Chen Z, Qiu T. August 2014, Poster: reliable TCP for popular data in socially-aware ad-hoc networks. InProceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing pp. 429-430, ACM.
[7] Luo, Ying, et al. 2014, "An improved congestion avoidance control model for TCP Vegas based on Ad Hoc networks." Control and Decision Conference (2014 CCDC), The 26th Chinese. IEEE.
[8] Allman, Mark, Vern Paxson, and Ethan Blanton, 2009, "TCP congestion control". No. RFC 5681.
[9] Issariyakul, Terawat, and Ekram Hossain, 2011, "Introduction to network simulator NS2". Springer Science & Business Media.
[10] Klaue, Jirka, Berthold Rathke, and Adam Wolisz, 2003, "Evalvid–A framework for video transmission and quality evaluation." International Conference on Modelling Techniques and Tools for Computer Performance Evaluation. Springer Berlin Heidelberg.
[11] Mondal, Amit, and Aleksandar Kuzmanovic, 2008, "Removing exponential backoff from TCP." ACM SIGCOMM Computer Communication Review 38, no. 5, pp. 17-28.