International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 3, Number 9 (2013), pp. 939-946
© International Research Publications House
http://www. irphouse.com /ijict.htm

Performance Analysis of Selective Acknowledgement with Fractional Window Increment (SACK-FeW)

Taranvir Kaur

CSE/IT, CTIEMT, CT Institutes, Shahpur, Jalandhar, India.

Abstract

In this work we investigated the effect of congestion over the average throughput, average packet loss, average packet retransmission and average energy consumption of the various nodes of the 802.11 ad hoc networks. Due to lack of coordination and sharing in these networks a number of problems has been encountered. One among the various problems that lowers the performance of ad hoc networks is the window mechanism of Transmission Control Protocol (TCP). To handle this problem TCP with fractional window (FeW) mechanism has been proposed earlier that has given very good results as compared to original TCP. Here in our work we implemented the same mechanism with another variant of TCP known as TCP with selective acknowledgement (TCP-SACK) and named the modified version as SACK-FeW. Our simulation results have shown that the proposed scheme improves SACK's performance in 802.11 adhoc networks.

Keywords: TCP, Ad Hoc networks, congestion, disconnections, Ack, SACK, Newreno, lite, DYMO, LANMAR, OLSRv2.

1. Introduction

In Ad hoc networks[1] the nodes move independent of each other and are not dependent on any centralized authority due to this reason there is lack of coordination in these nodes that causes overreaction of routing protocol due to TCP[2]. One among the several sources that affects the quality of end to end connection is in the window size measuring mechanism of TCP. In order to address this problem fractional window mechanism in TCP is implemented earlier that lowers the over reaction of TCP on reactive routing protocol[3] and limits the TCP's aggressiveness.

In wireless Ad hoc networks various available resources are shared by all the nodes present in the network. Due to its nature TCP tries to maximize the usage of available natural resources and leaves little resources for the lower layer usage, the lack of available resources for lower layers affects the quality of the end to end connections and lowers the TCP performance. By using TCP with fractional window increment it has been proved experimentally that simply by reducing the network overload the overall performance of TCP and reactive routing protocol can be improved[4][5].

Network overload is a major factor causing packet loss packet loss, packet loss increases with increase in network load, analysis in [6][7][8][9] has proved that that it is the growth rate of the TCP congestion window that effectively controls the loss rate and, consequently, the network overload.

To keep the network load at a reasonable level, a fractional window increment scheme, called FeW has been proposed, that allows a fractional increment of the TCP congestion window. The FeW scheme can be validated mathematically by the well-known TCP-friendly equation[1]. If we modify the window increment scheme it will shift the TCP operation to a new range of applications that can give better results with lesser available bandwidth. It has been proved by using simulations that TCP-FeW dramatically improved the transport layer performance in 802.11 multihop networks.

In our study we have used this fractional window mechanism with another TCP variant known as TCP-Selective Acknowledgement (SACK)[9][11] and named this new variant as SACK-FeW.

Traditional TCP makes use of cumulative acknowledgement scheme[2]. In this scheme those received segments are not acknowledged that are on the left side of the received window due to which the sender is forced to either wait for a round trip time to get knowledge about each lost packet or the sender unnecessarily retransmits segments. Multiple dropped segments in cumulative acknowledgement scheme leads to loss of acknowledgement based clock in TCP reducing overall throughput and SACK (selective acknowledgement) is the solution to this problem. SACK is a technique in which the receiver informs the sender about all the segments that has been received successfully, so the sender needs to retransmit only those segments that has been received successfully, so the sender needs to retransmit only those segments whose acknowledgements has not yet been received or that are actually lost. After retransmitting these lost packets the sender updates the size of the congestion window either by one maximum segment size or if the size of congestion window is greater than slow start threshold then by-:

$$\text{Increment} = \frac{\text{Increment} * \text{Increment}}{\text{Size of congestion window}}$$

Here, Increment = maximum segment size[1]

Here instead of using these mechanisms for updating the size of the congestion window we made use of fractional window update mechanism by using same formula for updating the window in both the cases i.e when the congestion window is less as well as greater than the slow start threshold. It has been seen via simulations that

SACK-FeW gives higher throughput, lesser loss and lesser retransmission with nearly similar energy consumption.

Rest of the paper consists of three sections 2, 3 and 4. In section 2 formula used to calculate the size of the congestion window and algorithm demonstrating the use of the formula are discussed, The methodology adopted and the performance of the proposed scheme are discussed in section 3 and the concluding remarks and future scope are given in section 4.

2. Function Used and Algorithm

2.1 SACK – Fractional Window Increment Function

SACK – FeW allows the TCP congestion window to grow by a fractional rate instead of growing by a fixed value always. The congestion window in SACK - FeW grows by a fractional rate α (packets) after retransmitting the packets. The value of α can vary from 0 to 1 mathematically it can be expressed as $0 \leq \alpha \leq 1$. The formula used is given below [5]-:

$$cwnd_{new} = cwnd_{current} + \frac{\alpha}{cwnd_{current}}$$

Here, $cwnd_{new}$ is the new size of congestion window

$cwnd_{current}$ is the current size of congestion window

α is a constant such that $0 \leq \alpha \leq 1$

2.2 SACK - Fractional Window Increment Algorithm

The modified algorithm of TCP – SACK with fractional window increment is as shown below-:

```
// cwnd_new is the size of congestion window after increment.
// cwnd_current is the size of the congestion window before increment.
// α is any constant such that 0 ≤ α ≤ 1.
// Increment is inversely proportional to current size of congestion window and depends upon the value of α.
// ssthresh is the slow start threshold value
1. Initialize α = 0.01
2. Initialize Increment = α / cwnd_current
3. On receiving selective acknowledgements from the receiver if loss of packets
   have taken place then retransmit the packets
4. Set cwnd_current = minimum(cwnd_current, ssthresh)
5. cwnd_new = cwnd_current + Increment
```

3. Results and Analysis

3.1 Methodology

The performance of modified variant was evaluated via simulations. The simulation was carried on Qualnet Network Simulator version 5.0 developed by Scalable Network

Technologies. The main interest of the project was to test the performance of SACK-FeW over DYMO[8] and compare it with SACK over DYMO with changing network topology. Furthermore the focus was set on varying number of nodes and area sizes. We experimented with 300 seconds of simulated time over a square field for each scenario. The various modified parameters are shown in Table 1.

Table 1: Parameters Used for Analysis.

Parameters	**Values**
TCP Variants	SACK, SACK-FeW
Routing Protocols	DYMO
Fading Model	Rayleigh
Shadowing Model	Constant
Pathloss Model	Two-Ray
Energy Model	Mica Motes
Battery Model	Simple linear
Mobility Model	Random Waypoint
Mobility Speed	2 to 10 meter/sec
Network Size- 10 nodes	Area Considered- 500 X 500
Network size- 20 nodes	Area Considered- 700 X 700
Network Size- 50 nodes	Area Considered- 1000 X 1000
Network size- 100 nodes	Area Considered- 1500 X 1500
Network Size- 200 nodes	Area Considered- 2000 X 2000
Node Placement	Random node placement under seed-1

3.2 Performance Analysis

Four parameters namely average throughput, percentage of average packet loss, percentage of average packet retransmission and average energy consumption were used to analyse the performance of SACK-FeW and compare it with the performance of SACK over same scenarios. The results are as shown below-:

3.2.1 Average Throughput-: In communication networks throughput is the rate of successful message delivery over a communication channel. It can be seen in the Table 2 that as the number of nodes are increasing the throughput in both the cases that is in SACK as well as SACK-FeW is decreasing this is because of the reason that as the number of nodes are increasing the amount of data to be transmitted is increasing leading to increase in the amount of network load leading to decrease in throughput. Table 1 clearly shows that as the number of nodes are increasing SACK-FeW is giving higher throughput as compared to SACK.

Table 2: Average Throughput for SACK and SACK-FeW.

Average Throughput(bits/sec)		
Number of Nodes	**SACK**	**SACK-FeW**
10	169467.5	167358.5
20	51384.75	50651.5
50	22783	28890.7
100	15782.45	16428.72
200	2714.658	4314.042

3.2.2 Percentage of Average Packet Loss-: Transport layer divides the data into small packets and sender sends these packets to the destination via physical channel due to certain problems in these physical channels, congestion or due to other constraints loss of these packet may take place. It can be seen in the Table 3 that as the number of nodes are increasing the percentage of average packet loss is increasing this is because of the reason that as the number of nodes are increasing the network faces more load due to which more loss of packets takes place. Table clearly depicts that on an average percentage of average packet loss is less in case of SACK-FeW as compared to SACK.

Table 3: Percentage of Average Packet Loss for SACK and SACK-FeW.

Percentage of Average Packet Loss (%)		
Number of Nodes	**SACK**	**SACK-FeW**
10	0.09	0.074
20	0.94	1.048
50	1.53	1.4
100	1.64	1.48
200	3.97	2.77

3.3.3 Percentage of Average Packet Retransmission-: Whenever loss of packets takes place the the receiver on not receiving the packets start sending the old acknowledgements i.e already sent acknowledgements again and again which belongs to the last successfully received packets and these acknowledgements are termed as duplicate acknowledgements whenever the sender receives the duplicate acknowledgements it retransmits the lost packets by sending the packet with next sequence number next to the sequence number present in the duplicate acknowledgement. Table 3 depicts the percentage of average packet loss, as the number of nodes are increasing the percentage of average packet retransmission is increasing this is because due to increase in number of nodes the amount of data present in the network that needs to be transferred from one node to another increases and hence overloads the network due to which more loss takes place that leads to more packet retransmission. And also it can be seen in the Table 4 that by using SACK-FeW the percentage of average packet retransmission has been decreased in certain cases like with 10 and 200 nodes as compare to SACK.

Table 4: Percentage of Average Packet Retransmission for SACK and SACK-FeW.

Percentage of Average Packet Retransmission(%)		
Number of Nodes	**SACK**	**SACK-FeW**
10	0.028	0.0128
20	1.4075	1.43
50	2.43	2.57
100	2.72	2.98
200	4.94	4.28

3.3.4 Energy Consumption-: Here energy consumption is the measure of the average energy consumed in transmitting and receiving the packets. Table 5 shows the average energy consumption by varying the number of nodes. As the number of nodes are increasing the average energy consumption is decreasing and also the values of average energy consumption are same in the case of 10, 100 and 200 nodes. With 20 nodes SACK-FeW is giving better performance and with 50 node SACK is giving better performance.

Table 5: Average Energy Consumption for SACK and SACK-FeW.

Average Energy Consumption (mJ)		
Number of Nodes	**SACK**	**SACK-FeW**
10	3.16	3.16
20	2.71	2.68
50	2.41	2.43
100	2.26	2.26
200	2.1	2.1

4. Conclusions

In this work we analysed the performance of SACK-FeW. All the results were abtained via simulations. The various contributions of this work include-:

- Setting up the various scenarios for simulation.
- Modifying the existing TCP variant SACK to SACK-FeW.
- Evaluating the proposed scheme over five different scenarios.

- Comparing the performance of proposed algorithm with the already existing variant (SACK).

Even though 802.11 multihop networks impose many technical challenges on TCP, it can be concluded from the current study that the TCP window mechanism actually provides a good solution to 802.11 networks with a proper value of α.

References

[1] Ahmed Helmy, C.-C.Jay Kuo, Kitae Nahm, "*TCP over multihop 802.11 networks: issues and performance enhancement*," in Proceedings of ACM MobiHoc, Urbana-Champaign, IL, May 2005.

[2] J.Postel, "*Transmission Control Protocol*", RFC 793, 1981.

[3] B.S. Manoj and C. Siva Ram Murthy, "*Ad Hoc Wireless Networks*", Pearson Education, 2005.

[4] Bhaskar Sardar and Debashis Saha, "*A Survey of TCP Enhancements for Last-Hop Wireless Networks*", IEEE, Communications Surveys & Tutorials, 3rd Quarter 2006, Vol 8, Issue. 3, pp-20-34.

[5] Xinbing Wang , Yun Han and Youyun Xu, "*APS-FeW: Improving TCP throughput over multihop adhoc networks*", Computer Communications, Elsevier, September 11 2008, pp-19-24.

[6] Dan Keun Sung, Nana Li, Wenbo Zhu, and Xinming Zhang, "*TCP Transmission Rate Control Mechanism Based on Channel Utilization and Contention Ratio in Ad hoc Networks*" , IEEE Communication Letters, APRIL 2009, VOL. 13, NO. 4.

[7] C. Siva Ram Murthy and Venkataramana Badarla, "*Learning-TCP: A stochastic approach for efficient update in TCP congestion window in ad hoc wireless networks*", Journals Parallel and Distributed Computing, Elsevier, January 20 2011, pp- 863–878.

[8] Tilman Wolf, Weibo Gong and Yan Cai, "*Delaying Transmissions in Data Communication Networks to Improve Transport-Layer Performance*", IEEE Journal on Selected Area in Communications, MAY 2011 , VOL. 29, NO. 5.

[9] A. Romanow, J. Mahdavi, M. Mathis and S. Floyd , "*TCP Selective Acknowledgment Options*", RFC 2018, October 1996.

[10] C. Perkins and I. Chakeres, "*Dynamic MANET On-demand (AODVv2) Routing*", March 12, 2012.

[11] Mandakini Tayade and Sanjeev Sharma, "*Review of Different TCP Variants in Ad- Hoc Networks*", International Journal of Engineering Science and Technology, March 2011, Vol. 3, Issue. 3, pp-1906-1913.