International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.2, No.1, January 2013 DOI:10.15693/ijaist/2013.v2i1.7-12

Improving Throughput in Mobile Ad hoc Networks using Receiver Assisted Congestion Control

R.Prasanna
Assistant Professor
Department of ECE,
Dhanalakshmi Srinivasan Engineering College
Perambalur, Tamilnadu, India.

Abstract*-Transmission control protocol (TCP) fails to fully utilize bandwidth due to the limitation in its conservative congestion (CC) algorithm. It assumes that every packet loss is caused by network congestion and initialises its congestion control algorithm. Receiver-assisted congestion control (RACC) mechanism, in which the sender performs loss-based control, while the receiver is performing delay-based control. The receiver measures the network bandwidth based on the packet interarrival interval and compute a congestion window size deemed appropriate for the sender. After receiving the advertised value and then feeds this information back to the sender. The sender is to adjust its congestion window according to the advertised window of the receiver. By integrating the loss-based and delay-based congestion control, we can mitigate the effect of wireless loss, timeout effect. Experimental results show that our mechanism can improve the network throughput performance.

Keywords:-Congestion control, rate estimation, transport protocol, wireless networks.

I. INTRODUCTION

Transmission Control Protocol (TCP) provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). TCP is one of the two original components of the suite, complementing the Internet Protocol (IP), and therefore the entire suite is commonly referred to as TCP/IP. Due to network congestion, traffic load balancing, or other unpredictable network behavior, IP packets can be lost, duplicated, or delivered out of order. TCP detects these problems, requests retransmission of lost packets, rearranges out-of-order packets, and even helps minimize network congestion to reduce the occurrence. TCP [8] is one of the core protocols of the Internet Protocol Suite. TCP provides the service of exchanging data directly between two network hosts, whereas IP handles addressing and routing message across one or more networks. Recently, some physics researchers want to migrate to wireless networks for its ubiquitous convenience. However, they are not satisfied with the delay-throughput performance because the TCP used in the bulk data transfer suffers from significant throughput degradation and very high interactive delay through the wireless networks. TCP traffic is accounting for about 90% of all Internet traffic nowadays. To understand the poor performance of TCP in wireless networks, understand how TCP operates. Upon a timeout or receiving three duplicate acknowledges, TCP treats these events as an indication of a packet loss and reduces its congestion window. This strategy usually works well in a traditional wired network. Unfortunately, packet loss in a wireless network may also be due to transmission problems such as a high link error probability, fading, and interference. Therefore, packet loss is no longer an appropriate indication for network congestion. With the wrong constructed information, TCP may reduce its congestion window unnecessarily, resulting in poor performance of wireless networks.

Another problem of TCP is its poor capability to utilize the network bandwidth efficiently, especially in networks with a high bandwidth delay product. When there is no packet loss detected, the congestion window (cwnd) is increased by one maximum segment size (MSS) every round-trip time (RTT). The TCP sender reduces cwnd by half if the packet loss is detected by three duplicate ACKs or reduces cwnd to one if the packet loss is detected by timeout. In a high-speed network with a large RTT, TCP requires a very large window to efficiently utilize the network resource. From the above description, one can see that standard TCP takes a very conservative approach to update its window in the congestion avoidance stage.

A new receiver-assisted congestion control mechanism (RACC) is proposed to resolve the above problems. Unlike regular TCP where the receiver only performs flow control, we allow the receiver to participate in congestion control. Sender and receiver combined and integrating the congestion control mechanism. The receiver estimates a congestion window deemed to be appropriate from the measured bandwidth and RTT [6], and then advertises the window size (feeds this information back) to the sender. The sender then adjusts its congestion window according to the advertised window of the receiver. Through this receiver-assisted method, the sender can increase the congestion window quickly to the available bandwidth, thus improving the network utilization. On the other hand, when timeout happens, the receiver can feed this

information quickly back to the sender to relieve the impact of timeout to TCP performance.

The sender then uses the additive increase and multiplicative decrease (AIMD) mechanism to compute the correct congestion window size to be used by integrating the loss-based and the delay-based congestion controls, our mechanism can mitigate the effect of wireless loss and can improve the throughput performance in lossy high-speed wireless networks. Simulation and preliminary experimental results suggest that our mechanism is a promising algorithm to achieve high link utilization while remaining friendly with regular TCP.

The paper is organized as follows. Section II reviews the related work. Section III describes our new RACC mechanism in detail. Section IV presents simulation results and we conclude the paper in Section V.

II. RELATED WORK

Ever since the poor performance of TCP under high-speed and high link error probability environment was pointed out, numerous solutions have been proposed to deal with it. These approaches can be classified into several categories based on the location of the control. They are the sender-centric methods [3], the receiver-centric methods [10], and the mobile-host-centric methods [11].

It can be further subclassified into delay-based methods and loss-based methods. Delay-based approaches modify the transmission rate based on RTT variations during congestion. It executes the multiplicative increase operation if the buffer occupancy at the bottleneck router is far less than some predefined thresholds, and then switches to a linear increase operation if it is close to the threshold. FAST TCP [7] tries to maintain the buffer occupancy around the threshold and reduces the sending rate if the measured delay increases further. Theoretical analysis and experiments show that delay based approaches have better properties than pure loss-based approaches. They result in better performance such as higher bandwidth utilization, less self-induced packet losses, faster convergence speed, better RTT fairness and better stabilization. However, previous work also reveals that delay-based approaches may not be able to obtain a fair share of the network resource when they are competing with loss-based approaches like standard TCP. To resolve this problem, Compound TCP [9] uses a combination of delay-based and rate-based method; however, this method also inherits the problem of inaccurate bandwidth estimation.

Loss-based approaches modify the increment/decrement parameters of a congestion avoidance algorithm to make TCP more aggressive in reaction to packet loss. TCP Westwood [3] is an end-to-end transport layer solution that uses the ACK arrival rate to estimate the bandwidth, and then sets a proper congestion window according to the estimated bandwidth when three duplicate ACKs are received. Although this operation alleviates the influence of wireless drops and improves the channel utilization, the main drawback is the oscillation in the estimation of the available bandwidth. This is because ACK interarrival time instead of DATA packet interarrival time is used to measure data bandwidth. Therefore, the ACKs in Westwood's bandwidth estimation can affect such estimation in two ways: 1) wrong estimation if the ACKs use a path different from the DATA packets; 2) the receiver may use delayed ACKs. BIC-TCP [4] adopts a binary search method to aggressively increase the congestion window when the difference between the current congestion window and the maximum window is large. However, it is difficult to estimate the maximum allowed window. API-RCP [5] proposes an adaptive proportional integral (PI) rate controller for the active queue management (AQM) router to support best-effort service traffic in the Internet. The problem is that it has to modify both the TCP function and the router's function.

We can see from the above analysis that the TCP problem could not be resolved only by loss-based mechanism. Although achieving some performance improvement [3], [6], [7], sender centric delay-based schemes have been criticized for their accuracy and the effectiveness of congestion prediction at the sender. We present a receiver-centric transport protocol called RCP (Reception Control Protocol) that is a TCP, but allows for better congestion control, loss recovery, and power management mechanisms compared to sender-centric approaches. More importantly, in the context of recent trends where mobile hosts are increasingly being equipped with multiple interfaces providing access to heterogeneous wireless networks, we show that a receiver-centric protocol such as RCP enable a powerful and comprehensive transport layer solution for such multi-homed hosts.

A Mobile-host-Centric Protocol (MCP) [11] is proposed by moving the control to the mobile host. MCP can make better use of the wireless channel status (such as contention and bit error) to improve the accuracy of congestion control. Although RCP and MCP can obtain better throughput performance in some scenarios, they face the problem that the receiver centric control may block the data sending. For example, if the receiver requests the data when the sender has no data to send, the receiver may assume that the data packets were lost in the network and hence perform an unnecessary congestion window reduction. On the other hand, if the data is available on the sender side, but the request of the receiver does not arrive in time, the sender will remain in a waiting state. As the sender controls the data, it is better to perform congestion control at the sender side.

Unlike the three types of congestion control mechanisms classified above, we propose a hybrid congestion control method by combining sender-centric and receiver-centric methods as well as combining both delay-based control and loss-based control. We can improve TCP the sender will decrease the congestion window to one in consideration that the network is in congestion, performance in lossy high-speed networks using the following two approaches. The first one is to reduce the waiting time of a sender to alleviate the impact of timeout. The second one is to combine the window-based control of the sender and the delay-based control of the receiver in order to lower the impact of packet loss to TCP performance and to improve the bandwidth utilization.

III. RACC MECHANISM

In this mechanism, the receiver not only performs the function of flow control, but also participates in the congestion control. Congestion control concerns controlling traffic entry into a telecommunications network, so as to avoid congestive collapse by attempting to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending packets. It should not be confused with flow control, which prevents the sender from overwhelming the receiver.

There are many ways to classify congestion control algorithms:

1. By the type and amount of feedback received from the network: Loss; delay; single-bit or multi-bit explicit signals.
2. By incremental deploy ability on the current Internet: Only sender needs modification; sender and receiver need modification; only router needs modification; sender, receiver and routers need modification.
3. By the aspect of performance it aims to improve: high bandwidth-delay product networks; lossy links; fairness; advantage to short flows; variable-rate links.
4. By the fairness criterion it uses: max-min, proportional, "minimum potential delay".

A. *Congestion Window*

In TCP, the congestion window determines the number of bytes that can be outstanding at any time. This is a means of stopping the link between two places from getting overloaded with too much traffic. The size of this window is calculated by estimating how much congestion there is between the two places. The sender maintains the congestion window. When a connection is set up, the congestion window is set to the maximum segment size (MSS) allowed on that connection. Further variance in the collision window is dictated by an Additive Increase/Multiplicative Decrease approach. This means that if all segments are received and the acknowledgments reach the sender on time, some constant is added to the window size. The window keeps growing linearly until a timeout occurs or the receiver reaches its limit. If a timeout occurs, the window size is halved.

The main task of TCP congestion control is to adjust the sending rate of the source in accordance with the state of the network. TCP limits the amount of outstanding data. The congestion window (cwnd) represents the maximum amount of data a sender can have sent and for which no acknowledgment was yet received. In particular, when the source starts sending data, TCP conservatively initializes cwnd to one packet and then doubles cwnd for every window worth of ACKs received until congestion is detected. When loss is detected, cwnd is reduced to react to congestion and to reduce the risk of losing more packets. The congestion avoidance algorithm is then used to slowly increase cwnd in a linear manner by one packet for every window worth of packets that are acknowledged. If a loss event occurs, TCP assumes this it is due to network congestion and takes steps to reduce the offered load on the network. Once a loss event has occurred or the threshold has been reached, TCP enters the linear growth phase.

B. *Congestion Avoidance*

As long as non-duplicate ACKs are received, the congestion window is additively increased by one MSS every round trip time. When a packet is lost, the likelihood of duplicate ACKs being received is very high (it possible though unlikely that the stream just underwent extreme packet reordering, which would also prompt duplicate ACKs). The TCP congestion window defines the number of packets to be sent at every round-trip time, and TCP is designed to adjust its congestion window according to the bandwidth-delay product of the network. Without loss of generality, we assume the same size for all packets so that the bandwidth delay product can be expressed in terms of the number of packets.

Congestion avoidance phase is to use a faster rate to achieve maximum capacity of the network, and then slow down the sending rate, making the process of sending stability at a higher rate. Congestion avoidance phase is to use a faster rate to achieve maximum capacity of the network, and then slow down the sending rate, making the process of sending stability at a higher rate. Use the ratio of the number of queuing packets to the buffer size in bottleneck link as a parameter, and use four sub-phases to change the congestion window.

C. Retransmit Timer

TCP, a sender that uses a too aggressive retransmit timer has to pay the price (i.e. slow down) after a spurious timeout. Presumably this discourages developing too aggressive retransmission timers and preserves the network from duplicate retransmissions that do no useful work. Therefore, some modification to the retransmit timer that makes it more conservative after a spurious timeout is needed. Another important factor to be considered is the value of retransmission timeout (RTO). Although different RTT values result in an out-of-order delivery of packets at the destination, this is better than losing packets or increasing traffic due to duplicate acknowledgments. It is less aggressive than RTO computed without timestamps due to using the delayed segments for RTT sampling. Immediately after a timeout when original ACKs are arriving, the RTO becomes very high. This section discusses various approaches to adapting the retransmit timer after a spurious timeout. Note that in order to increase conservativeness of the retransmit timer, the TCP sender must be robust to packet losses. A conservative RTO such as suggested in with timestamps provides a sufficient protection against excessive spurious timeouts in many cases. Further adapting the timer may include the following options:

1. Re-seed the RTO after a spurious timeout
2. Reset the back-off counter only on a genuine timeout
3. Increase the minimum RTO

The first option is to use an RTT sample obtained with timestamps from delayed segments to re-initialise RTT and RTTVAR variables and restart the timer. The second option

is to keep the back-off counter at the level set during a spurious timeout and reset it only on a genuine timeout. The third option is to perform additive increase of the minimum RTO value on each spurious timeout and reset it to the default value on a genuine timeout. Further possible option could be to increase the minimum RTO value based on a length of a delay or its experientially smoothed average. It is also possible to exploit different options for reducing the minimum RTO value, such as halve it on every genuine timeout instead of simply resetting it to the default. When multiple paths with different RTT values are present, it is better to maintain different RTO values for each of these paths. This ensures a guaranteed delivery of the packets to their destination and also reduces the number of duplicate acknowledgments.

D.RACC Algorithm

The receiver measures the RTT based on the DATA packet arrival time. We use a variable rcv.rtt to record the RTT. The algorithm can be described as follows.

1. When an ACK is sent, if rcv.rtt is zero, let rcv.rtt equal 1 and record the corresponding sequence (rtseq) of data packet (equals to the sum of the ACK sequence and the current congestion window). Otherwise, just send the ACK.

2. If a data packet with a larger sequence number than "rtseq" is arrived in order and the new measured packet interarrival interval is larger than two times of pre-estimated packet interarrival interval, set the new measured RTT to the value of rcv.rtt, and let rcv.rtt be zero.

3. On each clock cycle, if rcv.rtt is not zero, rcv.rtt is added by 1. Then, the retransmission timer is set based on the measured RTT.

$$rwnd = Bw * \text{RTT} \qquad (1)$$

The timer is managed according to the procedures in RFC 2988. Next, the receiver uses RTT to convert the bandwidth to the receiver congestion window value. Then, it compares the rwnd with the available receiver buffer, and deposits the lesser value in the advertised window field of an ACK going back to the sender.

$$\text{adv\_wnd} = \min(r\_\text{abuf}, rnwb) \qquad (2)$$

In otherwords, the receiver advertised window not only has the original flow control function, but also takes on the congestion control function. When the receiver detects a timeout, it will send an ACK to and inform the sender that the network is congested. In this ACK, the receiver advertised window is set to one packet, meaning that the sender should retransmit the dropped packet. As the receiver must have detected this timeout earlier than the sender, it will help the sender to reduce the waiting time and confirm the packet loss.

I . *Sender's Function*

As the receiver can timely help the sender to increase the congestion window according to the instantaneous available bandwidth, the sender only needs to maintain the AIMD mechanism in the congestion avoidance stage, and the slow start stage can be eliminated. Upon a timeout event (whether it is detected by the senders timer or informed by the receivers ACK), the sender will decrease the congestion window to one in consideration that the network is in congestion, and it will have some time to recover. If congestion is mitigated after one RTT, the sender will recover/adjust the congestion window in the next window by using the receiver advertised window.

During fast retransmission, the sender sets the congestion window size to the lesser value of the receiver advertised window size and the current size. Since the packet loss may also indicate congestion, we should reduce the congestion window. On the other hand, if the congestion window is less than the receiver advertised window, the network may only be in mild congestion. Therefore, it is unnecessary for the sender to reduce the congestion window. In a normal state, when the sender receives an ACK, it will compare the current congestion window with the receiver advertised congestion window. The sender then adjusts its congestion window according to the advertised window of the receiver. If the receiver advertised window is larger than the sender's congestion window, and the difference is larger than a predefined threshold, the sender will set the congestion window size to the receiver advertised window. Otherwise, the sender will ignore the receiver advertised congestion window by just performing the additive increase mechanisms.

II. *Receiver's Function*

The receiver measures the bandwidth according to the packet inter-arrival interval. This method can remedy the oscillation in the estimation of bandwidth of TCP Westwood as mentioned earlier.

Let *Bw* be the measured bandwidth, *L* be the data packet size, and *tint* be the packet inter-arrival interval. Then, one can be estimate the available bandwidth by Bw = L/tint for each packet arrival. Let *Bwi* be the *ith* measured bandwidth. Then, the bandwidth can be continuously updated by

$$Bw = \alpha\, Bw + (1 - \alpha)\, Bwi\,, \qquad (3)$$

where α is an exponential filter coefficient. From our experiments, $\alpha = 0.9$ is a good value, for the following reason that the former averaged values should have a higher weight to lower the measure variations.

IV.SIMULATION RESULTS

The analysis of the work is carried out in the NS2 simulator under LINUX platform. NS2 is a discrete event simulator for wired and wireless networks. It is not a polished and finished product, but the result of an on-going effort of research and development. NS is mainly used for simulating local and wide area networks and it simulates a wide variety of IP networks.

Fig.1. represents the example scenario for node placement. Both end-link are wired, while the intermediate bottleneck link is wireless. S1, S2, S3, S4 represents a TCP sender and D1, D2, D3, D4 represents a TCP receiver. Node 6 and 7 is the two routers with a finite buffer capacity. In wired link bandwidth is 100mb/s and wireless channel bandwidth is 54mb/s. The packets are transferring from node 3 to node 6. In wireless, packet drops from wired drops; it can reduce/eliminate the unnecessary throughput reduction.

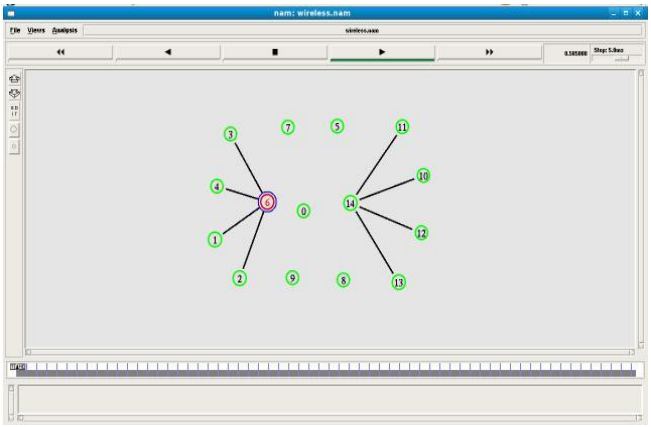

Fig.1. Example Scenario for Node Placement

Fig.2. represents the comparison of bandwidth and interarrival time. It shows that the arrival rate is to estimate the bandwidth, and then set a proper congestion window. The receiver as measure the network bandwidth based on packet interarrival interval. In X axis is Bandwidth in bytes and Y axis is arrival time in milliseconds. It is seen that if the bandwidth increases the arrival time decreases.

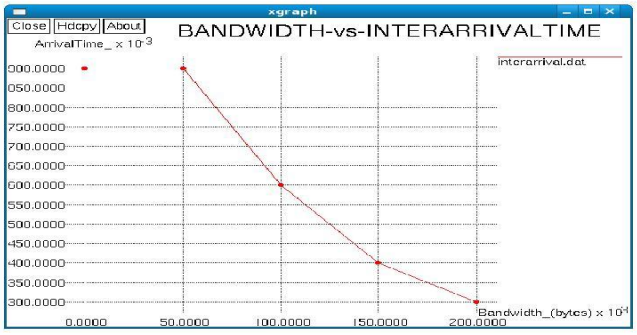

Fig. 2. Bandwidth Vs Interarrival Time

Fig.3. represents the comparison of bandwidth and rwnd. This comparison it shows that receiving window is larger than the sender congestion window. Its difference is larger than predefined threshold value, the sender as set the congestion window size. In X axis represents Bandwidth in bytes and Y axis represents Receiving window in milliseconds. If Bandwidth increases, the receiving window also increases.

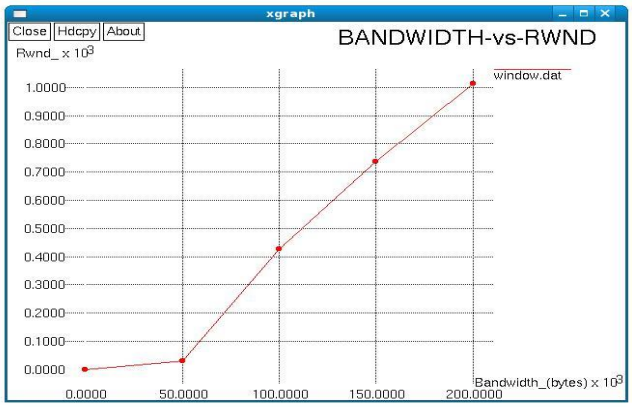

Fig. 3. Bandwidth Vs Rwnd

Fig.4. represents the throughput and this comparison shows that the receiver advertised window is set to one packet, meaning that the sender should retransmit the dropped packet. As the receiver must have detected this timeout earlier than the sender, it helps the sender to waiting time and confirms the packet loss. In shows that the throughput of RACC is high compared to New Reno. In X axis represents packet error probability and Y axis represents throughput.

Fig. 4. Throughput

V.CONCLUSION

In a mobile ad hoc network, a sender with loss-based control and receiver with delay-based control are integrated in order to control the congestion . The receiver always detects a packe t drop earlier than the sender and it also feedback the information quickly

back to the sender. The receiver measures and estimates the network bandwidth, based on the packet interarrival time and moreover it sends the network bandwidth to the sender . The sender can adjust its congestion window according to the advertised sending rate and additive increase and multiplicative decrease (AIMD) mechanism. By combining the loss-based and the delay-based congestion controls, our mechanism can reduce the effect of wireless loss and timeout effect. Simulation results under various scenario have validated that this mechanism can improve the network throughput performance and make better use of the network resources.This system can again be improved in the case of a lossy wireless network such as a wireless LAN (WLAN) system, in multi-hop scenarios using I E E E 8 0 2 . 1 1 medium access control (MAC) layer protocol in the place of the TCP congestion control. The network utilization can be further improved by using this MAC layer protocol.

REFERENCES

[1]Y. Shu, L. Zhang, W. Zhao, H. Chen, and J. Luo,"P2P- based data system for the EAST experiment,"IEEE Trans. Nucl. Sci., vol. 53, no. 3, pp. 694–699, n. 2006.

[2]W. Stevens, "TCP slow start, congestion avoidance, fast retransmit," IETF, RFC 2001, Jan. 1997.

[3]C. Casetti et al., "TCPWestwood: Bandwidth estimation for enhanced transport over wireless links," in Proc. 7th ACM Annu. Int. Conf. Mobile Comput. Netw., Rome, Italy, 2001, pp. 287–297.

[4]L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in Proc. 23rd IEEE INFOCOM, Hong Kong, China, 2004, pp. 2514–2524.

[5]Y. Hong and O.Yang, "Design of adaptive PI rate controller for best-effort traffic in the internet based on phase margin," IEEE Trans. Parallel Distrib. Syst., vol. 18, no. 4, pp. 550–561, Apr. 2007.

[6]L. S. Brakmo and L. L. Peterson, "TCP vegas: End to end congestion avoidance on a global internet," IEEE J. Sel. Areas Commun., vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[7]D. X. Wei, C. Jin, S. H. Low, and S. Hedge, "FASTTCP: Motivation, architecture, algorithm and performance," IEEE/ACM Trans. Netw., vol. 14, no. 6, pp. 1246–1259, Dec. 2006.

[8]J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan,"Evaluation of TCP Vegas: Emulation and experiment," Comput. Commun. Rev., vol. 25, no. 4, pp. 185–195, Oct. 1995.

[9]K. Tan and J. Song, "A compound TCP approach for high-speed and long distance networks," in Proc. 25th IEEE INFOCOM, Barcelona, Spain, Apr. 2006, pp. 1–12.

[10]H. Hsieh et al., "A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces," in Proc. 9th ACM Annu. Int. Conf. Mobile Comput. Netw., San Diego, CA, 2003, pp. 1–15.

[11]Y. Shu et al., "Mobile-host-centric transport protocol for EAST experiment, IEEE Trans. Nucl. Sci., vol. 55, no. 1, pp. 209–216, Feb. 2008.

[12]V. Paxson and M. Allman, "Computing TCP's retransmission timer," IETF, RFC 2988, 2000.

AUTHOR PROFILE

R. Prasanna **received** the **B.E (ECE)** degree from the V.R.S. college of engineering and technology, Arasur in 2008 and **M.E** from the G.K.M College of Engineering and Technology, Perungulathur, India, in 2011. As my research interests include Communication Engineering, Ad hoc Networks, Design issues includes **OFDM MIMO** and noise Suppression in **MAI** Systems, **ASIC** design, Control systems, Fuzzy logic and Networks, **AI**, Sensor Networks