LLDB-TCP: An adaptive congestion control technique for Mobile Ad-hoc Networks based on link layer measurements

Sreenivas B.C [1], Dr.G.C.Bhanu Prakash[2]
[1]*Associate Prof. Department of Computer Science and Engineering*
Research Scholar, SAHE University Tumkur.
[2]*Professor Department of Computer Science and Engineering,*
Sir M Visvesvaraya Institute of Technology, Bangalore, INDIA
Email: srinivasbc@rediffmail.com

Abstract: **Congestion control is a key problem in mobile ad - hoc networks. Congestion has a severe impact on the throughput, routing and performance. Identifying the occurrence of congestion in a Mobile Ad-hoc Network (MANET) is a challenging task. The congestion control techniques provided by Transmission Control Protocol (TCP) is specially designed for wired networks. There are several approaches designed over TCP for detecting and overcoming the congestion. This paper considers design of Link-Layer congestion control for ad hoc wireless networks, where the bandwidth and delay is measured at each node along the path. Based on the cumulated values, the receiver calculates the new window size and transmits this information to the sender as feedback. The sender behavior is altered appropriately. The proposed technique is also compatible with standard TCP.**

Keywords: Congestion, TCP, Ad-hoc

1. Introduction

Mobile Ad-hoc Networks (MANET) do not have a fixed infrastructure. MANETs uses standard IEEE 802.11 MAC. In ad-hoc network each node (Mobile device) acts as a router, which helps in forwarding packets from a source to destination. MANETs are suitable in situations where fixed infrastructure is unavailable such as Military war fields, disaster relief, sensor networks, Wireless mesh network etc.,

TCP congestion control is very much suitable for Internet, whereas for MANETs the same TCP is not suitable due to some of the specific properties like node mobility and shared wireless multi-hop channel. A slow delivery and packet loss occurs due to node mobility and unreliable shared medium. The delay in the packet delivery or packet losses is due to route change should not be misread as congestion.

In Internet when congestion occurs it is normally concentrated on a single router, whereas, due to the shared medium of the MANET congestion will not overload the mobile nodes but has an effect on the entire coverage area. The changes in the routing of the packet might lead to packet losses which is not caused due to congestion in the network should not be erroneously misinterpreted as TCP congestion. This can lead to wrong reactions of TCP congestion control. Furthermore, monitoring packet losses is much harder, because of their varying transmission time and round trip time.

Many devices in ad-hoc network, sharing a common resource (i.e., media) compete for link bandwidth, which leads to network overload. When more data packet arrives at the router, the un-serviced packet gets dropped. These dropped packets would have consumed most of the network resources. The lost packets have to be retransmitted, which in turn leads to pumping of more packets into the network, resulting in degradation of network throughput and leading to congestion. To avoid congestion and network overload each sender has to adjust its data sending rate and window size.

A lot of research is being carried out in the area of congestion control, routing of packets, modification of standard TCP protocol, designing of new routing protocol, etc. in MANET.

In OSI reference model, congestion control is the responsibility of the transport layer. The combination of congestion control and reliability features in TCP, allows congestion control management without the information about congestion status of the network. A proper mechanism is to be adopted to avoid congestion collapse of the MANET, which lead to the modification of TCP congestion mechanism [1]. The modified TCP should provide error and flow control. Flow control guarantees that the sender does not flood out the receiver by sending data at a rate faster than the receiver can process. It should also provide reliable end-to-end transmission of data over MANETs. The modified TCP should be capable of providing full-duplex, reliable and byte-stream services to the application programs.

2. Related work

A suitable congestion control technique for MANET is considered as an important issue. Some of the congestion related issues like throughput degradation and flow fairness are initiated from Media Access Control (MAC), routing and transport layer as discussed in [2][3][4][5]. Several papers have addressed and provided suitable solutions to overcome these problems.

A wireless link is prone to random packet losses unlike wired network. These losses affect the transport protocols performance, if they are wrongly interpreted as congestion induced by dropped packets. The link layer provides single hop reliability in 802.11 MAC protocol. The packets are dropped by link layer, only after

maximum transmission attempts. This occurs when either a link is lost or due to packet collision. This section mainly deals with different approaches for congestion control in wireless ad-hoc network.

The performance of mobile ad-hoc network improves substantially using a small TCP congestion window as shown by Fu et al. in [17].

Dynamic congestion window limit [22]

This approach is based on the broadcast characteristics of the wireless medium proposed by Chen et al. In wireless multi-hop networks, the Bandwidth Delay Product (BDP) of the connection depends upon the congestion window limit. Further the author suggests that the value of BDP should not exceed the round trip hop count. In standard IEEE 802.11 the value of BDP is taken as 1/5 of round trip hop count.

In this technique the Dynamic Source Routing (DSR) protocol is used to find the path length at source. The congestion window limit is set dynamically based on previously computed path length of a connection. The author has carried out NS-2 simulation experiments to justify the performance improvement, in comparison with TCP Reno. Further in their simulations the maximum retransmission timeout of TCP is modified to 2s instead of 240s as given in RFC 1122.

Slow Congestion Avoidance (SCA)[23]

In this technique the growth rate of TCP window size is restricted to less than one segment per round trip time, in order to bring down the number of packets in the network. After receiving the successful acknowledgement within round trip time, this technique increases the TCP window size by one segment. The cross layer information of the transport layer protocol is not used to carry on shared channel properties of MANET. The author has not explored the properties of this technique for different traffic load.

Fractional window increment (FeW).[24]

This technique mainly focuses on the manner in which TCP behaves in mobile ad-hoc network by reducing the congestion window growth rate of TCP. The congestion in wireless ad-hoc network usually occurs due to link layer losses rather than queue overflows, thus affecting the routing of packet.

To maintain low loss rate in wireless ad-hoc network, this approach modifies the TCP's operational range. Author claims that, it is evident from the mathematical analysis; the change in the TCP's operational range is accomplished by incrementing the TCP congestion window of wireless ad-hoc network slower than in standard TCP.

Non-work-conserving scheduling

Yang et al.[25] observed MANET connected to a wired backbone , suggested that by reducing the congestion window size it degrades the performance of congestion control for a larger extent. They proposed Non-work-conserving scheduling mechanism. In this mechanism a timer is set after sending a data packet. In the next step the data packets are not sent by the same Node until the timer expires. This reduces the rate at which the packets are forwarded, at each intermediate Node.

Rate-Based Congestion Control (RBCC)

Zhai et al. in [26] proposed Rate-Based Congestion Control (RBCC) which adopts leaky bucket algorithm. In this mechanism the header is added with a new feedback field which is utilized by each intermediate node along the path. These nodes furnish information about maximum rate of flow at each node. They study the channel busy ratio, i.e. the time interval at which the medium is non-idle. This information is utilized to modify the newly added feedback field. This helps the source to decide upon the sending rate. Every intermediate node maintains the details of the flow passing through it for later computation of convergence of fairness.

Cross-layer congestion control (C^3TCP)[27]

In this mechanism two network metrics, bandwidth and delay are measured between source and destination by cumulating intermediate hop measurements. This scheme is proposed by Kliazovich et al. Similar to RBCC, a feedback field is added to the link layer header. The collected information at each intermediate node is stored in the feedback field. When ACK is generated at destination node, the feedback information of the data packet is transmitted to the sender. This information is used to modify receiver advertise window field in ACK. Further more it is used to modify the windows size of the sender, which is located beyond TCP stack as an additional module. All C^3TCP logic is part of additional protocol module which performs without disturbing original TCP.

TCP with Adaptive Pacing (TCP-AP)[28]

ElRakabawy et al. proposed a technique TCP-AP. This technique adopts an end to end based approach for congestion control unlike C^3TCP and RBCC. TCP-AP is a mixture of both window and rate based approach. TCP is added with rate based mechanism to avoid large burst of packets.

In this technique the author proposes 4 hops propagation delay as a metric, measured using RTT of the packets. This is assumed as any interference if happens could be within 4 hops. The delay is the time between the transmissions of packet by source node to the receiving node 4 hops downstream. In order to estimate minimum time between successive packets an addition metric, the

coefficient of variation of RTT samples, is used along with the 4 hops propagation delay.

3. Link-Layer Delay Bandwidth (LLDB) Technique.

TCP has been predominantly used as transport protocol in the wired Internet to deliver data; consequently, numerous Internet applications have been developed to run over TCP. However, as explained earlier, TCP may not work satisfactorily in ad-hoc networks.

3.1 Concept

TCP in an ad-hoc network should be capable of handling disconnection and reconnection, packet _out_ of_ order delivery in case of route change and errors due to node mobility in addition to congestion control.

LLDB is a congestion avoidance method which enables us to obtain high performance by gathering capacity information such as bandwidth and delay at the link layer in each participating node. In this paper, we have introduced an additional module LLDB-End-System, which is used by both sender and receiver. This module contains code for

- Sending data and ACK packet
- Computation of RTT
- Modification of congestion window
- Receiving of both data and ACK packet

Sending data and ACK packet

Sender node initiates the transmission by sending a LLDB data packet to destination node over the network. Initially the send function will identify whether the packet is LLDB data packet, LLDB ACK packet or any other packets. In this function, if it is LLDB forwarding data packet, instant time at which the packet is sent along with congestion window and SRTT will be stamped onto the packet header and the packet will be forwarded. If it is LLDB ACK packet, then the packet is stored with minimum delay and bandwidth is stored in to ACK packet and sent to the sender for congestion window modification and SRTT for the subsequent packet. Other than LLDB packets are ignored.

Computation of RTT*)*

This function is used to calculate The Smoothened Round Trip Sime(SRTT). Initially for the first packet SRTT will be set to delay, for subsequent packets SRTT will be obtained by considering the 86.5% of previous delay and 13.5% of current delay ,further the calculated value of SRTT is used for computation of congestion window size. This function is also maintains the history of smooth RTT to compute window threshold.

```
void LLDBEndsys::rtt_update(double tao)
{
        Int i;
        if(!rtt_inited){
                SRTT = tao;
                rtt_inited = 1;
                for(i=0;i<256;i++)
                        wnd_thru[i] = 0.0;
        }
        else
                SRTT = SRTT *0.865 + tao*0.135;
        srtt_estimate_= SRTT;
        if(wnd_thru[tcp_->window()] == 0)
                wnd_thru[tcp_->window()] = tao;
        else
          wnd_thru[tcp_->window()] = wnd_thru[tcp_->window()]
                                        * 0.865 + tao*0.135;
        return;
}
```

Modification of congestion window

In this module the modification of congestion window is based on old congestion window, SRTT, last congestion throughput and current throughput. If the TCP congestion window is greater than zero, then the current throughput is set to the value obtained by dividing product of old congestion window and LLDB packet size by old congestion window throughput.

If current throughput is less than the last congestion window, then the LLDB congestion window is decremented by one value of the old congestion window. If the decremented old congestion window is greater than 0, then the last congestion throughput is changed otherwise, it is set to zero.

If current throughput is less than last window throughput, then the congestion window size is set to old congestion window. Otherwise, the congestion window size will be increased; here rate at which packets should be sent through the network is calculated. Thus limit based congestion control window calculation takes place.

Receiving data and ACK packet

At the receiver side, upon the reception of the packet the receiver has to identify the type of packet, i.e. whether it is a LLDB data packet, LLDB ACK packet or any other packet.

Upon the reception of LLDB data packet the function will retrieve the required information such as bandwidth and delay, stored in the structure DLBW for computation of the congestion window, which in turn takes the minimum bandwidth and also the sum of the delay .If an ACK packet is encountered, the control flow is sent to Modicwnd( ) where the new congestion window size is calculated. This calculated window size is updated in LLDB ACK packet. No modifications are performed on any other packets other LLDB packets. In order to store the obtained cumulative data, a structure is added in the data frame of the link layer.

```
struct DLBW {
        double Mydelay_;
        double st_time_;
```



```
        double Mybw_;
        intMynode_;
        inline double&Mydelay() { return(Mydelay_); }
        inline double&Mybw() { return(Mybw_); }
        inline int&Mynode() { return(Mynode_); }
        inline double&st_time() { return(st_time_); }
};
struct MYDLBW  {
        struct DLBW *my_opt;
};
```

The gathered information will be stored in the structure DLBW which is defined in common header in packet.h, which will be accessible by link layer during the estimation of the delay and bandwidth without any restriction.

Once the packet is sent up to the agent from the link layer, the information about the packet will not be available to the link layer for any computation , thus to overcome this problem the control flow in the link layer while sending the packets up, will be interrupted and made to compute the delay.

```
void LL::sendUp(Packet* p)
{
        Int bh;
        .
        .
        .
        else if(ch->ptype()==59 ) {    // 59 refers to LLDB packect
                Compute_delay(p,&bh);
                s.schedule(uptarget_, p, delay_); }
        else
                s.schedule(uptarget_, p, delay_);
}
```

Initially compute_delay() proceeds with the computation if and only if the packet is a TCP packet, else the control flow is resumed. If compute_delay() is invoked by destination node ,then it calculates the delay : difference between packet receive time(rctime) and instance of arrival of packet(tin) and the minimum bandwidth. Thus the calculated values are stored in destination structure DLBW.

```
void LL::Compute_delay(Packet* p,int *flag)
{
   if (mac_->addr() == Mydst&&ch->ptype()==59 )  choice =-1;
   else if(mac_->addr() == src)  choice = 0;
   else   choice = mac_->addr();
   double rctime;
   if(ch->direction() == hdr_cmn::UP
        && mac_->addr() == Mydst) Myflg=1;
   else if(ch->direction() == hdr_cmn::DOWN) Myflg=1;
   if( ch->ptype()==59 &&ch->ack_pkt==0 &&Myflg)  {
  // TCP Packet
    switch (choice) {
      case -1://destination node
           ch->Myflag=1;
           rctime =  ch->BW_info.my_opt[mac_->addr()].st_time();
            ch->BW_info.my_opt[mac_->addr()].st_time()=tin;
            ch->BW_info.my_opt[mac_->addr()].Mydelay()=
                                  tin-rctime;
            ch->BW_info.my_opt[mac_->addr()].Mybw()
                                  =ch->size()/(tin -rctime);
            *flag=1;
            filewrite(p,mac_->addr());
                        break;
      case 0: // start node
            int nn,i;
            nn = LL::num_nodes;
            ch->BW_info.my_opt = new struct DLBW[nn];
            for(i=0;i<nn;i++){
              ch->BW_info.my_opt[i].Mydelay()=0;
              ch->BW_info.my_opt[i].Mybw()=0;
            }
            ch->BW_info.my_opt[ch->next_hop()].st_time()=tin;
            ch->BW_info.my_opt[src].Mydelay()=0;
            ch->BW_info.my_opt[src].Mybw()=0;
            *flag=0;
            ch->Myflag=0;
            prev_src = src;
            filewrite(p,ch->next_hop());
                        break;
     default:// intermediate nodes
            rctime = ch->BW_info.my_opt[mac_->addr()].st_time();
            prev_src = mac_->addr();
            ch->BW_info.my_opt[ch->next_hop()].st_time()=tin;
            ch->BW_info.my_opt[mac_->addr()].Mydelay()=tin-rctime;
            ch->BW_info.my_opt[mac_->addr()].Mybw()
                        =ch->size()/(tin-rctime);
            *flag=0;
            ch->Myflag=0;
            filewrite(p,ch->next_hop());
                        break;
   }   }  }
```

If the Compute_delay() is invoked by source node, the function will allocate memory for the DLBW structure for participating nodes, then the time at which the packet is sent and other required information will be written in the next hop's structure for further computation. Otherwise if it is invoked by the intermediate nodes, then the calculated delay and bandwidth will be written in to next hop's structure for further computation.

Finally, control flow returns from compute_delay() to send_up() function .By the addition of the function compute_delay() without any changes to the normal flow, the required details of a TCP packet is obtained and stored in the structure. This whole action is done for LLDB data packet only.

Simulation Parameters

The network consists of 5 nodes arranged as string topology in 500m x 500m square field and two nodes are used for cross traffic. The MAC layer is configured to IEEE 802.11. Interface queue at MAC layer is set to default number of packets. The nominal bit rate is 2 Mbps and maximum transmission range is 250m. The TwoRayGround model is used with maximum node speed of 4m/s. DSR is used as a routing protocol. The simulation time is 50s. Constant Bit Rate (CBR) traffic is introduced at a rate of 1Mbps between node(6) and node(7). FTP traffic is introduced between node (0) and node (5) with default packet size and LLDB as TCP agent.

Simulation Analysis

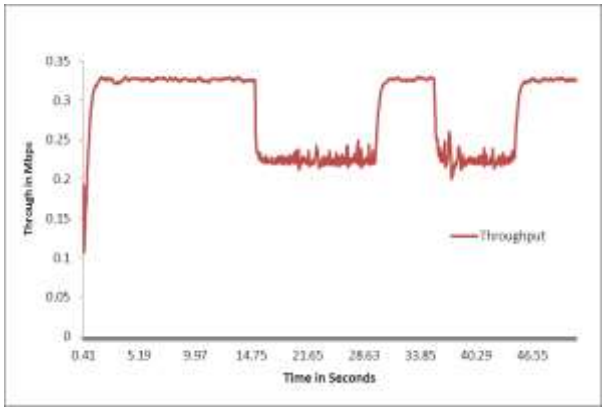

Fig. 1 Throughput Analysis

Fig 1 presents simulation results of throughput. The graph clearly indicates during the interval 0.41s to 15s the LLDB flow is not disturbed by any traffic and the achieved throughput is at 0.327Mpbs. Between the intervals 15.00s to 30.00s due to cross-traffic the throughput is varied from 0.217Mpbs to 0.243Mpbs. Again, an uninterrupted LLDB flow is started between the interval 30.00s to 35.00s where the throughput is 0.315Mpbs. When the cross-traffic UDP flow starts to take a part of bandwidth from the LLDB flow during the interval 35.00s to 45.00s, the throughput varies from 0.201Mpbs to 0.260Mpbs which can be observed in the above graph.

Congestion Window Analysis.

The Fig 2 depicts the analysis of congestion window. Initially the window size is set to be 1. When the LLDB flow is initiated, the congestion window is varied from 1.00 to 1.23.

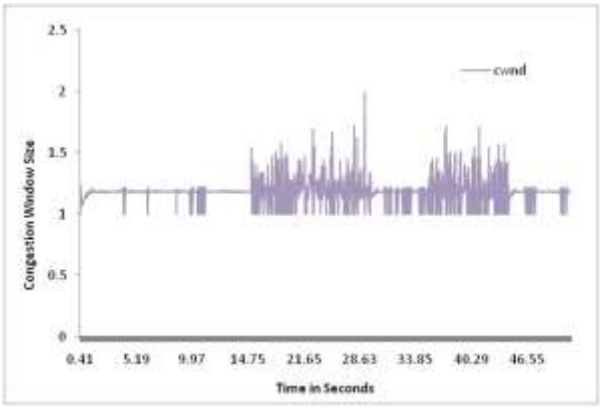

Fig. 2 Congestion window

During the interval 15.00 s to 30.00 s, the LLDB increases the congestion window size based on the feedback received because of which the packet loss occurs. And also due to the cross-traffic at the same interval the window size is varied from 1.00 to 1.98. The same variation can be observed during the interval 35.00s to 45.00s.

RTT Analysis

RTT is the total time it takes for a signal to be sent and the total time it takes for an acknowledgment of that signal to be received. The Fig 3 shows the analysis of RTT. Simulation is started at 0.41s at which the measured RTT is found to be 0.108.

At the interval 0.51 the RTT is at 0.032 which is maintained till the interval 15.00s. during the interval 15.00s to 30.00s, it is observed that the graph is varying from 0.085 to 0.031. This is due to the cross-traffic introduced by UDP flow because of which the delay increases and in turn affects RTT. The same variation can be observed at the intervals 35.00s to 45.00s

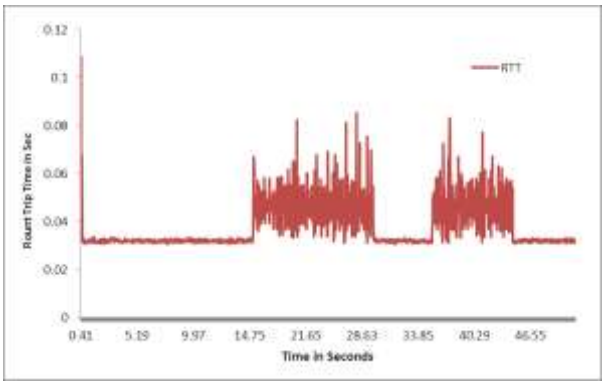

Fig. 3 RTT Analysis

Smoothened RTT Analysis

The Fig 4 presents the simulation graph for SRTT. When the LLDB flow is initiated the SRTT is uniformly maintained i.e. between the intervals 3s to 15s the value of SRTT is 0.03. Due to the cross-traffic, SRTT varies from 0.44 to 0.31 during the interval 15s to 30s and it varies from 0.46 to 0.33 during the interval 35.00s to 45.00s.

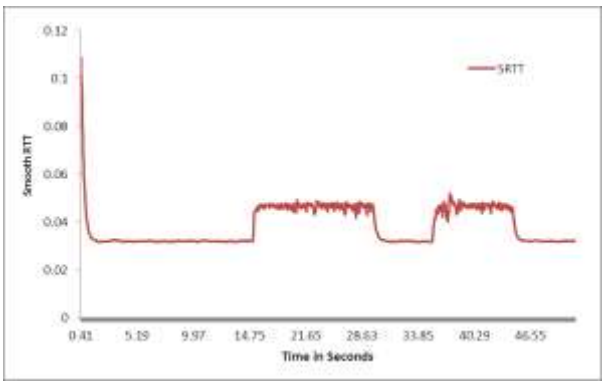

Fig. 4 Analysis of SRTT

Analysis at node N1

Fig. 5 represents the delay experienced by the packets at node N1. The X-axis represents the packet number and Y-axis represents the delay.

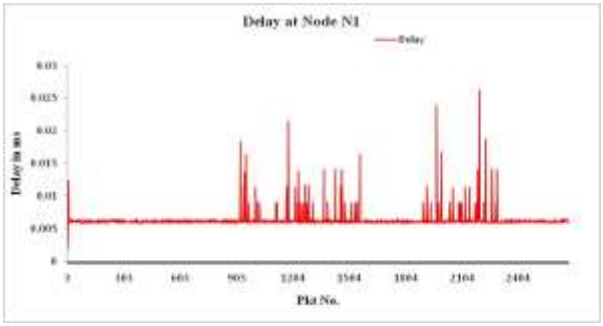

Fig. 5 Delay at node N1

A variation in delay with the peak value of 0.0239ms is observed between the packet numbers 903 and 1504. Another such variation is observed between the packet numbers1900 and 2350 with the peak value of delay being 0 .0262ms. These variations are basically due to cross-traffic which is pretty less at the source node.

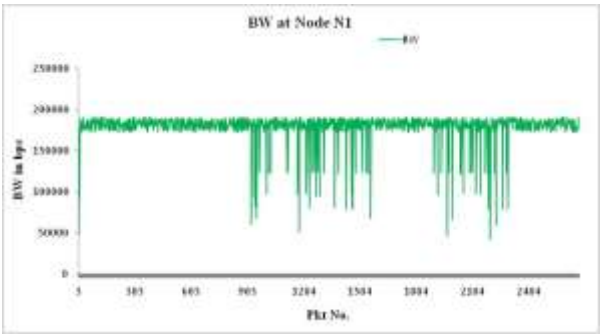

Fig. 6 Bandwidth at Node N1

Fig. 6 represents the bandwidth per packet at node N1. The X-axis shows the packet number and Y-axis represents the bandwidth in Mbps. A drastic dip in the bandwidth with the lowest value of 0.51Mbps is between the packet numbers 903 and 1504. Another such variation is between the packet numbers 1900 and 2350 with the least value of bandwidth being 0.46Mbps. These variations are basically due to cross-traffic which is pretty less at the source node.

Analysis at node N2

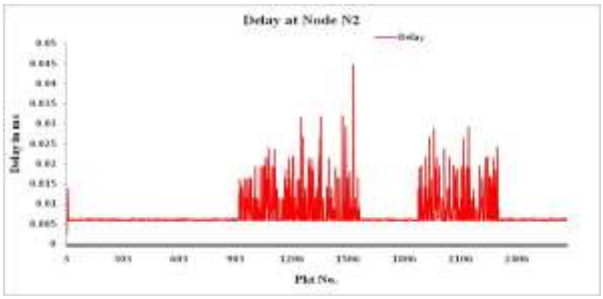

Fig.7 Delay at Node N2

Fig. 7 represents the delay experienced by each packet at node N2. The X-axis represents the packet no. and Y-axis represents the delay. A variation in delay with the peak value of 0.0447ms is observed between the packet numbers 903 and 1504. Another such variation is observed between the packet numbers1900 and 2350 with the peak value of delay being 0 .02917ms. The variation in delay is high in node N2 when compared to node N1 due to the presence of large cross-traffic.

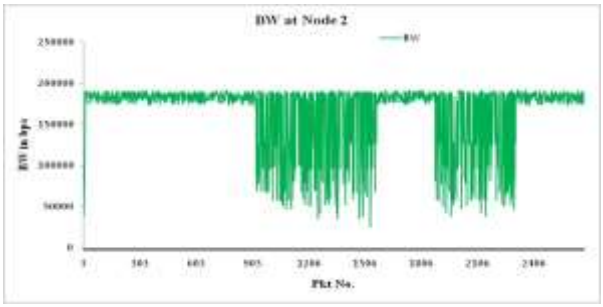

Fig. 8 Bandwidth at Node N2

Fig. 8 represents the bandwidth per packet at node N2. The X-axis shows the packet number and Y-axis represents the bandwidth in Mbps. A drastic dip in the bandwidth with the lowest value of 0.35Mbps is between the packet numbers 903 and 1504. Another such variation is between the packet numbers 1900 and 2350 with the least value of bandwidth being 0.44Mbps.The decrease in bandwidth of node N2 is greater when compared to node N1 due to the presence of large cross-traffic.

Analysis at node N3

Fig.9 represents the delay experienced by the packets at node N3.

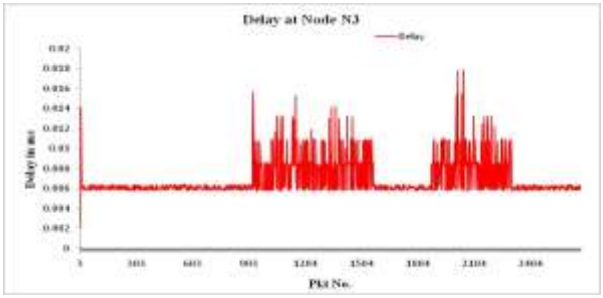

Fig. 9 Delay at Node N3

The variation in delay is high in node N3 when compared to node N1 due to the presence of large cross-traffic. Fig. 10 represents the bandwidth per packet at node N3.

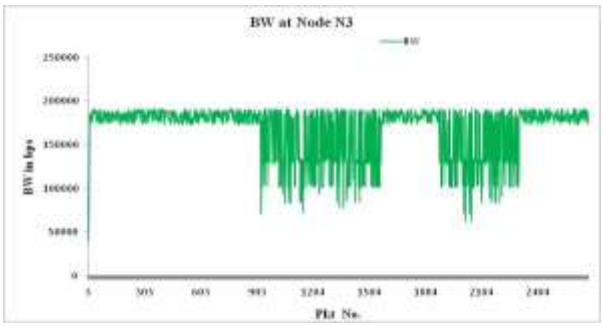

Fig. 10 Bandwidth at Node N3

A drastic dip in the bandwidth with the lowest value of 0.72Mbps is between the packet numbers 903 and 1504. Another such variation is between the packet numbers 1900 and 2350 with the least value of bandwidth being 0.62Mbps. The decrease in bandwidth of node N3 is greater when compared to node N1 due to the presence of large cross-traffic.

Analysis at node N4

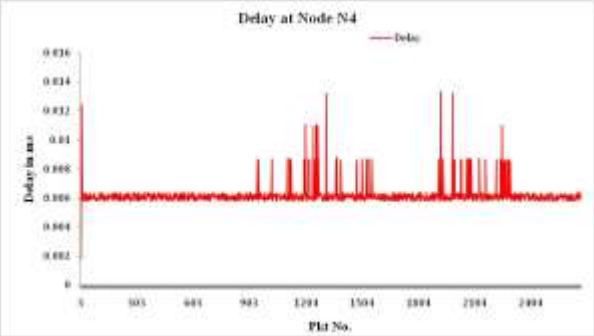

Fig. 11 Delay at Node N4

Fig. 11 represents the delay experienced by the packets at node N4. A variation in delay with the peak value of 0.0131ms is observed between the packet numbers 903 and 1504. Another such variation is observed between the packet numbers1900 and 2350 with the peak value of delay being 0.0133ms. These variations are basically due to cross-traffic which is pretty less at the destination node.

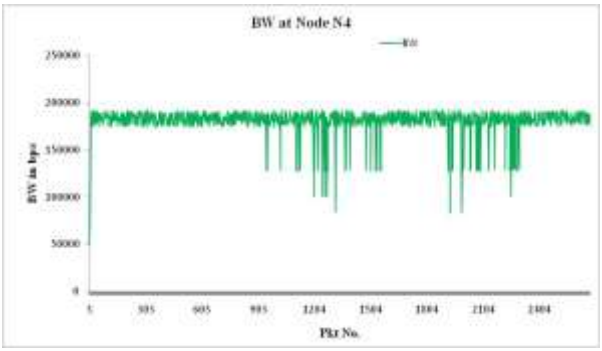

Fig. 12 Bandwidth at Node N4

Fig. 12 represents the bandwidth per packet at node N4. A drastic dip in bandwidth with the lowest value of 0.84Mbps is between the packet numbers 903 and 1504. Another such variation is between the packet numbers 1900 and 2350 with the least value of bandwidth being 0.83Mbps. These variations are basically due to cross-traffic which is pretty less at the destination node.

5 CONCLUSION

A congestion control algorithm using LLDB for a Wireless Ad-Hoc Access Network is developed and tested using the NS2 simulator. The reasons of throughput degradation and unfairness among flows, when using TCP in a Wireless Ad-hoc Network, were studied and simulations were conducted to verify the performance of TCP.

LLDB introduces several recent transport layer protocols, and current bandwidth estimation algorithms, which is the basis for designing the congestion control algorithm in this project. The proposed congestion control algorithm LLDB is able to obtain higher performance by gathering capacity information such as bandwidth and delay at the link layer. This method requires the introduction of an additional module within the protocol stack of the mobile node, which is capable of adjusting the outgoing data stream based on capacity measurements.

To support this designed congestion control module i.e. LLDB an additional proposal has been made which provides an optional field to support the existing IEEE 802.11 protocol stack to store the information obtained from the link layer.

Acknowledgement

The author acknowledges SAHE University Tumkur, for the encouragement and permission to publish this paper based on the research work carried out by the author towards his Ph.d work.

The author thanks the principal of Sir MVIT, Prof.K R Kini for his constant en-couragement and The author thanks Prof. Dilip K. Sen, Head of Department Computer Science Engineering for his invaluable guidance and suggestions from time to time.

References

[1] S. M. ElRakabawy, A. Klemm, and C. Lindemann. TCP with Adaptive Pacing for Multihop Wireless Networks. In MobiHoc ’05: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pages 288–299, May 2005.

[2] K. Chen, Y. Xue, and K. Nahrstedt. On Setting TCP’s Congestion Window Limit in Mobile Ad Hoc Networks. In ICC ’03: Proceedings of the IEEE International Conference on Communications, Anchorage, Alaska, May 2003.

[3] V. Raghunathan and P. R. Kumar. A Counterexample in Congestion Control of Wireless Networks. In MSWiM ’05: Proceedings of the 8th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pages 290–297, Oct. 2005. DOI: http://doi.acm.org/10.1145/1089444.1089496.

[4] K. Tang and M. Gerla. Fair Sharing of MAC under TCP in Wireless Ad Hoc Networks. In MMT ’99: Proceedings of the Workshop on Multi-access, Mobility and Tele-traffic for Wireless Communications, Oct. 1999.

[5] H. Zhai, X. Chen, and Y. Fang. Alleviating Intra-Flow and Inter-Flow Contentions for Reliable Service in Mobile Ad Hoc Networks. In MILCOM ’04: Proceedings of the IEEE Military Communications Conference, volume 3, pages 1640–1646,Oct. 2004. DOI: 10.1109 / MILCOM. 2004 1495184.

[6] D. Kliazovich and F. Granelli. Cross-layer congestion control in ad hoc wireless networks. Ad Hoc Networks, 4(6):687–708, Nov. 2006.

[7] H. Zhai, X. Chen, and Y. Fang. Rate-Based Transport Control for Mobile Ad Hoc Networks. In WCNC ’05:

Proceedings of the IEEE Wireless Communications and Networking Conference, volume 4, pages 2264–2269, Mar. 2005.

[8] M. G¨unes¸ and D. Vlahovic. The Performance of the TCP/RCWE Enhancement for Ad-Hoc Networks. In ISCC ’02: Proceedings of the 7th IEEE International Symposium on Computers and Communication, pages 43–48, 2002.

[9] Z. Fu, B. Greenstein, X. Meng, and S. Lu. Design and Implementation of a TCP-Friendly Transport Protocol for Ad Hoc Wireless Networks. In ICNP ’02: Proceedings of the 10th IEEE International Conference on Network Protocols, pages 216–225, Nov. 2002.

[10] R. de Oliveira and T. Braun. A Delay-based Approach Using Fuzzy Logic to Improve TCP Error Detection in Ad Hoc Networks. In WCNC ’04: Proceedings of the IEEE Wireless Communications and Networking Conference, volume 3, pages 1666–1671, Mar. 2004. 24

[11] R. de Oliveira, T. Braun, and M. Heissenb¨uttel. An Edge-based Approach for Improving TCP in Wireless Mobile Ad Hoc Networks. In DADS ’03: Proceedings of the Conference on Design, Analysis and Simulation of Distributed Systems, Orlando, USA, Mar. 2003.

[12] B.C Sreenivasa, G. C. Bhanu Prakash, K. V. Ramakrishnan. Survey on congestion control techniques in AD-HOC network. Published in Elixir Adoc.Net. 32 (2011) 2061-2067 March 2011.

[13] Christian Lochert,Bj¨orn Scheuermann, Martin Mauve. A Survey on Congestion Control for Mobile Ad-Hoc Networks article published in Wiley Wireless Communications and Mobile Computing 7 (5), pp. 655–676, June 2007. http://www.interscience.wiley.com.

[14] S. Biaz and N.H. Vaidya, ”Distinguishing congestion losses from wireless transmission losses” IEEE 7th Int. Conf. on Computer Communications and Networks, October 1998.

[15] Sachin Gajjar and Hari M Gupta. Improving performance of Ad-hoc TCP in Mobile Ad-hoc Network. Article is published in IEEE 2008, 987-1-4244-2746-8/08

[16] Miss Ashwini D Bhople and P.A Tijare. An analysis of ADTCP,I-ADTCP and Cross-Layer Based Protocol for Improving Performance of TCP in Mobile Adhoc Network. Published by IJCSA, Vol. No 4, No 2 June-July 2011. pp 56-70

[17] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The Impact of Multihop Wireless Channel on TCP Throughput and Loss. In INFOCOM ’03: Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, volume 3, pages 1744–1753, Apr. 2003.

[22] K. Chen, Y. Xue, and K. Nahrstedt. On Setting TCP’s Congestion Window Limit in Mobile Ad Hoc Networks. In ICC ’03: Proceedings of the IEEE International Conference on Communications, Anchorage, Alaska, May 2003.

[23] S. Papanastasiou and M. Ould-Khaoua. TCP congestion window evolution and spatial reuse in MANETs. Journal of Wireless Communications and Mobile Computing, 4(6):669–682, Sept. 2004

[24] K. Nahm, A. Helmy, and C.-C. J. Kuo. TCP over Multihop 802.11 Networks: Issues and Performance Enhancement. In MobiHoc ’05: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pages 277–287, 2005.

[25] L. Yang, W. K. G. Seah, and Q. Yin. Improving Fairness among TCP Flows crossing Wireless Ad Hoc and Wired Networks. In MobiHoc ’03: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing, pages 57–63, New York, NY, USA, June 2003. ACM Press.

[26] H. Zhai, X. Chen, and Y. Fang. Rate-Based Transport Control for Mobile Ad Hoc Networks. In WCNC ’05: Proceedings of the IEEE Wireless Communications and Networking Conference, volume 4, pages 2264–2269, Mar. 2005.

[27] D. Kliazovich and F. Granelli. Cross-layer congestion control in ad hoc wireless networks. Ad Hoc Networks, 4(6):687–708, Nov. 2006.

[28] S. M. ElRakabawy, A. Klemm, and C. Lindemann. TCP with Adaptive Pacing for Multihop Wireless Networks. In MobiHoc ’05: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pages 288–299, May 2005.

[29] B.C Sreenivasa, G. C. Bhanu Prakash, K. V. Ramakrishnan. Comparative analysis of ADTCP and M-ADTCP: Congestion Control Techniques for improving TCP performance over Ad-hoc Networks in International Journal of Mobile Network Communications & Telematics (IJMNCT) Vol.2, No.4, August 2012

