Mitigating Intra Flow Interference
Over Multi Hop Wireless Ad Hoc Networks

RADHA NATRAJ AND KATHIRAVAN KANNAN
Department of Information Technology
Easwari Engineering College
Chennai, 600089 India
E-mail: radhupriya@yahoo.co.in; kathirraji@gmail.com

Transmission Control Protocol (TCP) is a predominant transport layer protocol for
data transmission. The burst nature of TCP congestion control mechanism has often lead
to heavy link layer contentions over multi hop ad hoc networks. Acknowledgements
(ACK) that compete with the data packets for medium access in the reverse path, further
add up to the contention levels. Such intense contention paves way to interference and
reduces the throughput. In this paper, we propose a framework that ties up rate controlled
cross layer transport solution (CLTSP) with opportunistic network coding to minimize
the intra flow interference over wireless ad hoc networks. CLTSP performs rate con-
trolled transmission to prevent self interference among data packets by spacing them out
at an appropriate interval. The time interval between the deliveries of successive data
packets is dynamically measured based on four hop propagation delay and an appropriate
correction factor that are collected from intermediate nodes. Besides, we perform net-
work coding over the data and ACK packets pertaining to the same connection to reduce
intra flow contention. We use the basic idea of piggy code and incorporate effective
buffer management which is opportunistic with CLTSP. We evaluate the performance of
our protocol with TCP New Reno, Piggy code and show significant improvement in
terms of throughput, Packet Delivery Ratio, Packet drops and Link layer collisions.

Keywords: cross layer, network coding, TCP, intra flow interference, ad hoc network

1. INTRODUCTION

Ad hoc network is a collection of dynamic, self configured, and radio equipped
nodes without any infrastructure. Ad hoc network requires every intermediate node to act
as router, receiving and forwarding data to every other node. Multi hop wireless Ad hoc
Network (MHANET) is prevalently deployed in various scenarios wherein instantaneous
connectivity becomes the need of the hour either in emergency situations like a disas-
trous evacuation situation or a casual get together for presentations.

The broadcast and shared nature of wireless medium imposes greater challenges to
higher layers in the protocol stack. TCP is a reliable transport protocol designed for
wired network wherein packet losses tend to occur due to buffer overflow. The inability
of TCP to differentiate the type of packet loss makes it inappropriate for use in multi hop
Ad hoc Network. Moreover, the burst nature of TCP causes severe link layer contentions.
Many nodes compete with each other to reserve the channel for their data transmission.
Heavy Contention leads to interference that results in instability like frequent retransmis-
sions timeouts, collisions and misinterpreted route failures [1].

There are two types of interferences from the view of TCP (i) Inter Flow Interfer-
ence and (ii) Intra Flow interference. Inter flow interference occurs between the nodes when they transmit packets belonging to adjacent or nearby TCP flows. On the other hand, intra flow interference happens between the packets of same TCP connection. Intra flow interference is further divided into two sub types, Intra flow interference between (a) data and data packets (b) data and ACK packets.

Interference always occurs between the nodes lying in the same interference range. The type of interference is not known to the underlying Medium Access Control (MAC) layer. However, transport protocol running at the source node can control and oversee the intra flow interference prevailing among the successive data packets using its rate control mechanism.

Many proactive and reactive routing protocols assign the same path for the traversal of TCP-DATA and TCP-ACK packets of a single TCP connection. ACKs propagating in the direction opposite to that of the DATA streams have to vie for occupying the medium leading to contention. Network coding is a novel technology that exploits the intrinsic broadcast nature of wireless media, to significantly reduce the number of transmissions [2]. Network coding always comes with the overhead of intermediate nodes having to buffer packets so as to successfully perform decoding. This requires the nodes to maintain large buffers.

In this paper, we propose a framework that ties up two different approaches that conjointly reduce the overall intra flow interference. In the first approach, we use a rate based cross layer transport solution (CLTSP) that prevents self interference among data packets traveling in singular path towards the destination node. This approach has been proposed by us in our previous work [3]. We have contemplated the CLTSP in such a way that it delivers a data packet only if the previous packet leaves the contention region of the source node. We also take into account the spatial reuse of the wireless channel to meet the bandwidth delay product of the used network. In the second approach, we employ a strategic implementation of a Network Coding technique to address the issues of intra flow contention that comes with the added aid of ameliorating the throughput gains.

The rest of the paper is organized as follows, Section 2 describes the related work and in Section 3, we study the impact of intra-flow interference over multi hop ad hoc networks. In section 4 we explain the key elements of our design in detail. We evaluate our performance against TCP New Reno and Piggy code in Section 5 and finally Section 6 describes the conclusion.

2. RELATED WORK

In this section, we briefly summarize the papers published to address the instabilities due to the problem of contention over multi hop Ad hoc networks. The solutions are implemented as either transport layer or cross layer or MAC layer approaches by the authors.

Fu et al. [4] analyzed the main reason for packet drops in multi hop Mobile Ad hoc Network (MANET) and found that contention rather than buffer overflow is the primary factor for most of the packet losses. They maintain the optimal TCP congestion window size based on the link layer loss probability. Xin Ming Zhang et al. [5] proposed a cross layer solution that adapts the congestion window size based on the contention status from the network. The contention RTT is measured using queuing delay from the intermediate nodes which is a metric for reflecting contention. In [6], Zhang et al. put forth a
rate control algorithm that controls congestion window size by the factors namely, channel utilization and contention ratio. These values are measured by intermediate nodes using it’s transmit/receive time and waiting time. The sole purpose of the above mentioned approaches is to strive for the controlling of the window size and not to deal with the transmissions leading to intra flow contention.

ElRakabawy et al. [7] found a layered approach that performs rate based transmission using out of interference delay and variance of recently measured RTTs. The base idea of this paper has been the motivation for our work. However, this algorithm assumes that the queuing delay is same for both forward and reverse paths and it works on the upper bound of TCP Congestion window size. Sundaresan et al. [8] contemplated a trailblazing transport protocol, exclusively for mobile ad hoc network that performs rate based transmission using queuing delay incurred at each intermediate node. In [9] Ehsan et al. propounded a contention control approach wherein the TCP receiver monitors contention delay and achieved throughput periodically. It also measures the traffic rate and feeds it back to the sender to control the sending rate. These schemes do perform rate based transmission thereby reducing the contention, but they do not account for the spatial reuse in their transmission.

The following works concentrate on delaying the TCP acknowledgment to control contention between TCP DATA and ACK. Jubari et al. [10] calculated the number of acknowledgments to be delayed based on the inter arrival time between acknowledgments, sender TCP window size and packet drop events. In [11], authors used hop count to delay the acknowledgments. Farzaneh et al. [12] proposed a cross layer approach which reduces the number of acknowledgments by using collision probability at the intermediate nodes in a given channel condition. In [13] authors suggested to use different paths for transmitting forward data and reverse acknowledgments. They decided to select the least contended paths for the same. The process of delaying or reducing the number of acknowledgement in TCP may lead to inaccurate calculation of Round Trip Time which is crucial for the functioning of TCP congestion control.

Zhai et al. [14] proposed to mitigate the contention by prioritizing opportunities for urgent packets and by differentiating the type of packet transmissions. In [15], Shen et al. enhances the IEEE 802.11 DCF by allowing for packet transmissions when the channel condition is good. MAC layer collision probability is used as a metric in inspecting the channel status. In [16], J-CAR uses multiple interfaces on each node. One of the interface acts as a control interface and others are used for transmitting data. It dynamically selects least interfered path for data transmission based on the Channel Interference Index. The above approaches concentrate on MAC layer modifications to control contention.

Network Coding techniques are being proclaimed as the networking’s next revolution [17] and are considered as a paradigm shift in data transportation across networks. In COPE [18] wherein Medard et al. have proved the prowess of the Network Coding abilities with a practical test-bed implementation. This work looks out for effective coding opportunities to forward multiple packets in a single transmission. However, this demonstration could not produce desired levels of throughput gains in TCP end-to-end connections. Sundararajan et al. [19] proposed an intra-flow random network coding scheme with a new interpretation of ACK packets with the concept of Degrees of Freedom. But this work failed to bring out the fullest potential of the wireless network coding
as it employed coding operations only in the end hosts and not in every relay node. Chuan Qin et al. [20] proposed a system to integrate both inter and intra flow network coding operations in the wireless medium. But the suitability of this project in TCP connections is quite ambiguous. A Mac-level Network coding scheme named – Piggy code [2] had exclusive focus on the intra-flow packet coding mechanism which significantly improves TCP performance. Mario Gerla et al. used this work in their Combo Coding technique [21] with an additional timer mechanism to the Piggy code system. Both these works emphasize the need for an almost infinite buffer for effective operations.

Our proposed work, unique combination of Rate based transmission with opportunistic XOR-network coding [22] mix DATA and ACK packets and strive to achieve better throughputs and reduce the stress on the medium contention issue by restructuring buffer systems and Transport Layer mechanisms.

3. IMPACT OF INTRA FLOW INTERFERENCE OVER MULTI HOP WIRELESS AD HOC NETWORKS

Having been designed specifically for wired networks during its inception, TCP faces serious issues in the wireless world. TCP uses ACK clocking mechanism. Whenever TCP receives an ACK, it increases the congestion window size linearly and may start delivering couple of packets together. TCP does not keep track of the time at which the last packet was sent and whether the current set of packets will lead to any interference amongst them. Routing protocols such as AODV, DSR and DSDV assign the same route to the to-and-fro paths of DATA-ACK packets associated with a TCP connection. Multiple TCP connections make the scenario even bad. The effect of heavy link layer contentions is depicted in Fig. 1. It shows the impact at each layer which leads to reduction in throughput.

IEEE 802.11 is generally used as a MAC layer protocol for multi hop ad hoc networks. It uses RTS/CTS mechanism to reserve the channel for data transmission. Unfortunately, increased contention causes interferences to RTS, CTS packets due to hidden/

![Diagram of network interference impact](image)

Fig. 1. Effect of Intra flow interference over wireless ad hoc network.
exposed terminals. When the RTS retry count exceeds the limits, it drops the data packet and reports link failure to the network layer which unnecessarily initiates the route discovery. This is a typical instance of a misinterpreted route failure that occurs many times during transmissions. All this add up to the cost of overheads and reduces the throughput undesirably. These types of MAC retry and retransmission occurs at many points all over the forward and reverse path while transmitting TCP Data and TCP ACK packets. It also causes frequent timeouts followed by retransmissions at TCP source. Unnecessary retransmissions due to lost ACK will again lead to increased contention. To validate the effect of intra flow interference, we simulate a static multi hop horizontal chain network with eight nodes using ns-2.35 simulator. It has a single TCP connection originating at node 0 and ending at node 7.

We do not consider the wireless transmission error. The results are plotted in Fig. 2. The variation in throughput and the corresponding congestion window size are shown in Figs. 2 (a) and (b) respectively. It is clear that the congestion window size resets to one in many points due to timeout events which can only be the impact of intra flow interference.

![Fig. 2. Simulation result: Effect of Intra flow interference over wireless ad hoc network.](image)

The points of time at which congestion window cut happens can be seen in Fig. 2 (c). We have shown the number of misinterpreted route failures, retransmissions and packet drops through Fig. 2 (d). The total number of TCP packets dropped due to no route (NRTE), Retry Count Exceed (RTE), Call back (CBK) and Collision (COL) is 600 which is around 25% of total packets transmitted. 15% of the total delivery is for retransmissions. We can imagine the control overhead involved for 78 misinterpreted route
failures. We can conclude that addressing the intra flow contentions can minimize the interference and resolve many drawbacks. Being the controller of end-to-end traffic, the transport layer holds the responsibility of controlling the intra flow contention thereby intra flow interference.

4. SYSTEM ARCHITECTURE

In this section, we describe a framework that ties up two different approaches which can minimize the intra flow interference. The architecture of the system is shown in Fig. 3. Our first approach named CLTSP [1] is implemented at the transport layer of source node. In our second approach, opportunistic network coding is placed as a thin layer between MAC and network layer of every node. Its design elements are explained in the following section.

Fig. 3. Cross layer Framework.

4.1 CLTSP – A Cross Layer Transport Solution Rate Control

The function of any transport layer congestion control algorithm is to control the amount of data delivery so that it cannot exceed the bandwidth delay product of the used network. In multi hop wireless ad hoc network, the bandwidth delay product cannot be as high as wired network due to the shared broadcast nature of the medium [5]. Spatial reuse allows two nodes to transmit packets simultaneously through the same channel without any interference among them. CLTSP is designed by considering the above factors. CLTSP is responsible for controlling data rate to reduce the intra flow interference among forward data packets and also to improve spatial reuse for meeting the bandwidth delay product of the network.
CLTSP ensures that source can deliver a packet only when the previous packet leaves the interference range. Time duration between the deliveries of successive data packets at CLTSP source is named as Inter Packet Delay (IPD). IPD should be calculated such that it should not be large which may otherwise under utilize the bandwidth of the network and it should not also be small which can otherwise lead to self interference among the packets.

In order to understand IPD calculation, we brief about spatial reuse with an example shown in Fig. 4 (a). We assume IEEE 802.11 as MAC protocol with the transmission range of 250m and interference range of 550m. We consider node 0 to be the transmitter and node 1 to be the receiver. The distance between the nodes is 200m. The interference range of node 0 covers two hops till node 2 and for node 1, it is up to node 3. When node 0 transmits to node 1, the nodes which are two hops away from the sender and receiver should not be involved in transmission/reception to avoid interference. It means that node 4 can be allowed to transmit to its upstream node while node 0 is communicating with node 1 to accommodate spatial reuse. We can say that the time interval between the deliveries of successive packets at the transport layer can be the four hop propagation delay which is named as out of interference delay. So, IPD should be equal to the out of interference delay to prevent collision between two successive packets and to improve spatial reuse. We may also argue that the out of interference delay cannot be always represented as four hop propagation delay. The hop count that approximates the interference region will get increased when the nodes are closely placed. For example, we consider Fig. 4 (b), where the distance between the nodes is taken as 100 m (closely located). The transmission range of node F covers node G, H and its interference range covers up to node K. The routing protocol like AODV chooses shortest path which is based on hop count to reach the destination. The next hop for node F is chosen as node H not the node G. The path may be chosen as node F->node H->node J …. So, we assume that four hop propagation delays can represent the out of interference delay in most cases.

Four hop propagation delay (FHPD) is calculated by receiving cross layer feedback from the intermediate nodes. This feedback is sent to the source through the CLTSP ACK. FHPD will fail to reflect the out of interference delay between upcoming data packets when there is a growth/shrink in traffic during the journey of CLTSP ACK in the reverse path. To cope with this scenario, we add a novel correction factor along with FHPD to compute IPD for upcoming data packets. IPD is dynamically updated on re-
ceiving CLTSP ACK. IPD will dynamically shrink/grow according to the congestion status. In presence of multiple heterogeneous traffic flows, the four hop propagation delay or correction factor period will be large that leads to reduction in data rate. During light traffic, IPD will be less thereby increasing the data rate.

(A) Cross Layer Feedback from Intermediate Node

Intermediate nodes provide cross layer feedback through the packets (Data & ACK) exchanged between source and destination. Every node stamps its average hop delay \( (hd) \) information at the header of CLTSP data packet. The interval between the time at which the packet enters the queue and leaves the current node for transmission is measured by every node. The average of this time interval is accumulated with packet transmission duration to represent the one hop delay \( (hd) \). One hop delay information is stamped by every intermediate node through the data packet. It is diagrammatically shown in Fig. 5. Here \( hd_i \) represents the average one hop delay of node \( i \).

\[
\begin{align*}
& (hd_0, d_0, q_0) \quad (hd_1, d_1, q_1) \quad (hd_2, d_2, q_2) \quad (hd_3, d_3, q_3) \quad (hd_4, d_4, q_4) \\
& (d_0, q_0') \quad (d_1, q_1') \quad (d_2, q_2') \quad (d_3, q_3') \quad (d_4, q_4')
\end{align*}
\]

Fig. 5. Cross layer feedback.

In order to calculate the correction factor period, every node stores its interface queue length and average contention delay at each data and ACK packet when the packet leaves the node. \( q_0, q_1, q_2, ..., q_h \) represents the queue length when the current data packet is dequeued from interface queue for transmission at node 0, node 1, node 2, ..., node \( h \) respectively and \( q_0', q_1', q_2', ..., q_h' \) represents the queue length when the current ACK packet is dequeued from interface queue for transmission at node 0, node 1, node 2, ..., node \( h \) respectively.

Contention delay (CD) is the duration between the time at which the packet enters the head of the interface queue and the time at which it gets the medium for transmission. \( CD_i \) represents the average contention delay at \( i \)th node.

(B) CLTSP Destination

The function of CLTSP destination is similar to TCP Sink. It is responsible for sending Cumulative ACK for every data packet. When the data packet is received out of order, CLTSP destination will be transmitting three duplicate ACK mentioning the sequence number of missing packet.

In addition CLTSP destination extracts the one hop delay timestamps \( \{hd_0, hd_1, hd_2, ..., hd_h\} \), Queue Length \( \{q_0, q_1, q_2, ..., q_h\} \) and contention delay from the received data packet. These values are to be conveyed to source to calculate IPD, But, to reduce
the packet transmission overhead and size, destination can perform some computation on behalf of CLTSP source using collected time delay. Destination calculates \( fhpd_1 \) by summing the one hop delays of node 0, 1, 2 and 3. Similarly \( fhpd_2 \) is calculated by adding the next four one hop delays. Finally maximum four hop delay is measured as given in Eq. (1).

\[
fhpd(i) = \max(fhpd_1, fhpd_2, \ldots, fhpd_{i-4})
\]  

Where, \( fhpd \) represents the time taken to propagate through the first four hop nodes from the source. \( fhpd(i) \) represents the maximum four hop propagation delay of \( i \)th data packet in the forward path. Destination node puts the value of \( fhpd(i) \) and \( \max(q_0, q_1, q_2, \ldots, q_h) \) in CLTSP ACK header and transmits to source node.

(C) CLTSP Source

Source node is responsible for delivering out the data packets at an appropriate interval thereby controlling the data rate. This interval is named as Inter Packet Delay (IPD).

Four hop Propagation delay \( (fhpd(i)) \) represents the congestion status and the spatial reuse of the network during the journey of \( i \)th data packet in the forward path. During heavy traffic, \( fhpd(i) \) will also be more. The congestion in the network does not occur suddenly but it grows or shrinks from time to time. So, we take into account the four hop delay of recent \( n \) packets for calculating the exponential mean as shown in Eq. (2). Here \( k \) represents the weight age factor which has the value of 0.4.

\[
\text{expmean}(fhpd(i)) = (k \times \text{expmean}(fhpd(i-1))) + ((1 - k) \times fhpd(i))
\]  

Inter packet delivery (IPD) for the next two packets is measured based on this exponential mean. The value of IPD is dynamically updated while receiving each acknowledgement. Four hop propagation delay may not reflect the exact IPD for the next two packets when there is growth (new connections) / shrink in traffic during the journey of ACK in the reverse path. To cope with this, we add a novel correction factor that handles growth or decrease in traffic as shown in Eq. (3). The source node extracts the queue length and contention delay from the received ACK and computes IPD as shown in Eq. (3).

\[
cfactor(i) = (\max(q_1, q_2, \ldots, q_h) - \max(q'_0, q'_1, q'_2, \ldots, q'_h)) \times CD_i
\]

\[
\text{ipdi} = \text{expmean}(fhpd(i)) + cfactor(i)
\]  

(i) Probing or Initialization Phase

This phase is helpful for finding the Initial value of inter packet delivery period during connection establishment. CLTSP sends out a control packet and wait for an acknowledgment to calculate the initial IPD among the two packets. Probing is also useful for determining the Inter Packet Delivery Period of newly discovered path after route failure. CLTSP source starts delivering data packet based on the Initial IPD value found in this phase.

(ii) Detecting Route Failure

On detecting route failure, CLTSP source stops sending data temporarily. Route failure is detected with the help of timeout events. A retransmission timer is set for each
data packet based on Round Trip Time. When three timeout events occur continuously, it is interpreted as route failure and CL-TSP source is temporarily stopped from sending data till it recovers from failure. Source can resume sending data after it receives an acknowledgment from the end receiver. Calculation of timeout period for each data packet is similar to the TCP timeout algorithm.

(iii) Timers Used

Two types of timers are employed in CL-TSP source. They are periodic timer and an array of retransmission timers. The Periodic timer is scheduled for the inter packet deliver period (IPD). It is immediately started after delivering a packet. The expiry of the timer sends the next packet. Retransmission timer is set for each packet for the timeout period measured on RTT and its variance. It is cancelled after receiving ACK for this or higher packets. The algorithm for updating dynamic RTT and RTO are used from TCP.

(iv) Reliability

Reliability mechanism in CLTSP is independent of the congestion control or rate control. Reliability ensures that every data packet reaches the final destination by receiving ACK from destination. A retransmission timer is set for each data packet based on Round Trip Time. If source does not receive ACK within the timeout period, it prioritizes the packets to be retransmitted by storing it at the front of the buffer. These packets will be retransmitted after the expiry of periodic sending timer. Exponential Back off algorithm is executed for finding timeout period of retransmitted packets which is similar to TCP. When the CL-TSP receives duplicate acknowledgment for a packet, the corresponding data packet will also be put into the front of the buffer.

4.2 Opportunistic Network Coding

In this section we describe the opportunistic network coding that performs XOR between CLTSP data and ACK packets of the same connection to minimize the number of transmissions. We use the idea of Piggy code and incorporate effective buffer management [22] and pseudo broadcast techniques to reduce the buffer overhead.

The basic idea of Network Coding is to combine multiple packets together for transmission, thereby reducing the number of packet transmissions at the MAC layer, and consequently the number of times a node contends for the medium.

We implement piggy code network coding [2] which mixes data and acknowledgement in a single packet transmission to two different receivers, assuming that CLTSP DATA and ACK packets travel in the same path but in opposite directions. Here, piggy code makes use of decoding buffers of larger size to store a copy of all the sent packets. But, our scheme requires only the last delivered packet, in the decoding buffer.

(a) Encoding Process

Before we remove a packet from the head of the interface queue for transmission, we check if there are coding opportunities, by searching for a packet of opposite type. We are said to arrive at a coding opportunity when a packet of opposite type with interchanged source-destination pair addresses and belonging to the same flow are identified in the queue. If the queue lookup is successful, the two packets to be exored are chained
together. The chained packets are dequeued together from the Link Layer interface Queue and transmitted to the MAC layer for the actual Network Coding Operation.

We EXOR the payload fields of the MAC layer and put them up in the new EXOR-ed packet. The packets arriving at the MAC layer to be passed on to the Wireless Physical Layer are stored in the Packet Pool for decoding purposes. We then set the binary CODED field in the Network Coding header (NC Header) of the new packet to indicate that the packet is coded. We also insert the sequence ids of the two packets that have been coded. The new packet is now sent out from MAC for transmission.

(b) Decoding Process

When a coded packet is received, we check if one of the packets inside the coded packet has already been buffered in the pool. This lookup is performed using the packet identifiers as the search key. If the un-coded packet is found, it is exored with the received packet in order to retrieve the original packet.

4.3 Network Coding – A Stress on the Buffer?

Network Coding mandates the maintenance of larger decoding buffer to retain the packets that have already delivered, in order to perform successful decoding. This comes as a huge overhead in terms of the buffer space required. Also, this introduces a significant delay in the decoding process. Therefore, we resort to a technique which retains only the last packet delivered, in the buffer. This takes the stress off the buffer to a large extent. However the above mentioned technique is opportunistic with CLTSP, to ensure that packets belonging to the same connection do not get clogged at a specific node.

When we use CLTSP as a transport layer agent, it spaces out the packets with sufficient interval i.e., four hop transmission delay. When there is sufficient interval between the delivery of two packets of same flow, Each node need not store all delivered packets. Instead, it can carry only the recently delivered packet of current flow.

Broadcasting of exored packets might lead to collisions in the medium due to lack of RTS-CTS exchange in 802.11. This results in the TCP source having to retransmit the packet. Hence we resort to pseudo-broadcast of the coded packet. The coded packet is now unicast to only one of the two intended recipients by including the receiver address in the MAC header. The other receiver’s address is stored in our new Header. Since any node within the interference range of the sender can receive packets that are not destined for it, the node receives the packet and checks the new Header to see if it is the other intended recipient. If yes, it performs de-exoring of the packet. Pseudo broadcast decreases the collision rate to a considerable extent.

5. PERFORMANCE EVALUATION

We evaluate the performance of our protocol against (i) TCP New Reno without coding and (ii) TCP New Reno with piggy code using ns 2.35 simulator. Piggy code [2] is a Mac-level Network coding scheme that had exclusive focus on the intra-flow packet coding mechanism which significantly improves TCP performance. In the performance evaluation diagrams, we have used the following names.
• “TCP New reno” represents TCP New reno at the transport layer, IEEE 802.11 at MAC layer and no coding is done between the packets.
• “TCP with piggy code” represents TCP New reno at the transport layer, IEEE 802.11 at MAC layer and MAC level coding is done between the packets using piggy code.
• “CLTSP” represents CLTSP at the transport layer, IEEE 802.11 at MAC layer and MAC level network coding is done between the packets using effective buffer management technique.

We have considered two scenarios for performance evaluation. A horizontal static chain topology of 15 nodes with varying number of hops is taken in the first scenario and the second scenario takes a grid topology with 100 nodes in a 10*10 scenario. We run the simulation for the basic data rate of 1Mbps and tested each scenario with AODV as routing protocol.

The objective of CLTSP is to control intra flow interference over multi hop ad hoc networks. To prove the effectiveness of CLTSP in reducing the intra flow interference, we tested the performance on a single CLTSP flow with variable number of hops ranging from 1 to 15 and show our results in section 5.1.

Although our paper focuses on intra flow interference, we still need to face multiple cross traffic flows in real time. In section 5.2, we evaluate our performance in presence of multiple flows. We varied the number of CLTSP flows from 2 to 12. In addition, we added 5 UDP flows with every simulation. Each CLTSP and UDP flow is multi hop in nature. In section 5.3, we compared the performance with mobility among nodes. We introduced 1 m/sec and 5 m/sec speed with pause time of 0.1 second.

5.1 Performance Analysis of Single Flow with Variable Hops

(A) Throughput

Throughput is defined as the number of bits received per second by the destination. In Fig. 6, we measure throughput in terms of kilo bits per second and represent it in Y axis. Hop count of the flow is given as X axis. When the hop count increases, the contention region is getting enlarged. Due to this reason, the throughput of TCP and Piggycode+TCP comes down rapidly where as our method is able to perform better than TCP New reno. Initially throughput of CL-TSP is slightly less for one hop and two hops connection. It is because, when the number of hops is less than four, CLTSP sends a packet for every one end to end delay. When it is more than four, out of interference delay comes into picture. CL-TSP started giving higher throughput from hop count 3 onwards.
(B) Factors affected due to Intra flow interference.

Intra flow interference occurs when two packets belonging to same flow collide during transmission. This interference can happen for either control packets (like RTS, CTS, ACK) or data packets. We have analyzed both control and data packet drops in this section.

(i) Number of RTS, CTS collisions

Number of collisions is a factor to measure congestion in the network. When there are more RTS, CTS collisions, it means that many nodes compete with each other to reserve the medium at the same time and collides apart from back off algorithm. We represent number of collisions as Y axis in Fig. 7.

![Fig. 7. Number of collisions for varying number of hops.](image)

It denotes the interferences happened while transmitting two packets at the same time. It is calculated using the number of collision drops while transmitting Request To Send (RTS), Clear To Send (CTS) and Data packets. Piggy code+TCP performs broadcast for transmitting the exored packets which do not have RTS, CTS reservation. When the numbers of coding opportunities are more, collisions among the data packets are also more than TCP. Since CLTSP uses rate based transmission based on congestion status, it reduces the collisions significantly. When the hop count is less than 5, the collision is nil because of four hop propagation delay being treated as inter packet delivery period. The number of interferences is less than the other two methods as shown.

(ii) Number of Packet Drops

Packet drops refer to the number of data packets lost or dropped due to various reasons like COL, NRTE, CBK, RET. We analyze about these drops later in this section. In spite of absence of mobility, Bit errors and multiple connections, the increasing number of packet drops occurs only due to the intra flow interferences and congestion in the network. Our effort of reducing contention has largely reduced the packet drops as shown in Fig. 8.

(C) Single TCP Flow Analysis

In this section, we have analyzed the performance of a single flow in detail. We have considered a single flow from node 0 as source and node 15 as destination. We measure the throughput and study the various types of packet drops in detail.
Fig. 8. Number of Packet Drops for varying number of hops.

Fig. 9. Throughput over time for a single flow.

(i) Throughput of a single TCP flow.

The throughput is measured for every 5 seconds over time and depicted in Fig. 9. The result shows that the throughput of TCP and piggy code with TCP reaches zero at many points due to timeouts or congestion. Timeout results in slow start that leads to zero throughput. Our method performs consistently high throughout the simulation.

(ii) Packet drops of a single TCP flow.

Packet drops are important factors to depict intra flow interference. We have described the different types of packet drops in Fig. 10. COL drops indicate that data packets have been dropped due to collision. After drop, MAC layer tries to retransmit and recover from the drop. The maximum retry limit for data packets is three. RET indicates the drops when maximum retry limit is exceeded. During this event, MAC layer calls back the routing layer to report link failure which in turn initiates route discovery procedure. This RET drop increases MAC and routing overhead. If packet is dropped because of COL, MAC will try to fix it by retransmissions, But, RET drops indicate that congestion is heavy. CBK drops happen at routing layer after RET. Router discards the buffered data packets destined to the failed link and records CBK.

NRTE drop occurs at routing layer when there is no specific route to destination. The total number of Data packets dropped due to no route (NRTE), Retry Count Exceed (RET), Call back (CBK) and Collision (COL) is clearly shown in the Fig. 10. Our method has very less number of NRTE, CBK and RETS drops.

In Piggy code, coded packets are to be broadcasted which is actually intended for
two receivers. Broadcasting cannot have RTS/CTS mechanism for medium reservation. Due to this reason, the collision loss is very high for piggy code.

Our method uses pseudo broadcast to manage collision where we unicast the coded packet to one of the two receivers and we specify the second receiver’s address in the header. The second receiver overhears the coded packet and performs decoding. But, the second receiver cannot acknowledge the coded packet. Due to this reason, our method also suffers from collision.

5.2 Performance Analysis of Multiple Flows with Cross Traffic

In this section, we have taken the scenario of grid topology with 100 (10*10) nodes. We created multiple CLTSP flows each having the path length of more than four. In addition to CLTSP flows, we add five multi hop UDP flows with every simulation. X axis represents the number of flows. 2+5 means two CLTSP or TCP flows and Five UDP flows. The starting time of each flow is also randomly chosen. All the flows run till the end of simulation.

The throughput is shown in Fig. 11. Throughput increases when the number of traffic flows increase. The throughput of CLTSP is more than the other two. Every CLTSP flow controls its rate based on the out of interference delay of its own packets. This delay incorporates the contention delay against the cross traffic flows. Obviously, the inter packet delay of every CLTSP flow will be large when there is more cross traffic. In this
way CLTSP is able to respond for cross traffic. We can conclude that controlling the intra flow interference from the end source can spread the traffic in uniform manner.

During this scenario, Piggycode suffers from the need for larger buffer. Broadcasting makes the scenario even worse by increasing the number of collision drops. Packet delivery ratio is also very less for Piggy code with TCP comparing the other two as shown in Fig. 12. Packet delivery ratio of CLTSP drops when the number of flows increases. This is due to the increase in drops and retransmissions for the lost packets. PDR of CLTSP is better comparing other two as shown in Fig. 12.

In absence of Mobility and transmission errors, the packet drops represent the interference among packets. The details of packet drops are shown in Table 1. Number of drops are less for CLTSP comparing the other two methods. Number of mis interpreted route failure is almost equal to RET drops. Since the network is dynamic, the number of RET drops are more than single flow in section 5.1. Still, CLTSP is able to perform better than other two methods due to its rate control mechanism. The total packet drops faced by CLTSP is 30% lesser than the other two methods. RET drops are significantly lesser.

<table>
<thead>
<tr>
<th>Packet Drops</th>
<th>Number of flows</th>
<th>2+5</th>
<th>4+5</th>
<th>6+5</th>
<th>8+5</th>
<th>10</th>
<th>12+5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP NewReno</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP with PiggyCode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLTSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packets lost due to no route (NRTE)</td>
<td>TCP NewReno</td>
<td>71</td>
<td>105</td>
<td>143</td>
<td>182</td>
<td>174</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>TCP with PiggyCode</td>
<td>155</td>
<td>68</td>
<td>102</td>
<td>171</td>
<td>217</td>
<td>259</td>
</tr>
<tr>
<td></td>
<td>CLTSP</td>
<td>33</td>
<td>49</td>
<td>86</td>
<td>113</td>
<td>139</td>
<td>151</td>
</tr>
<tr>
<td>Packets lost due to Call back (CBK)</td>
<td>TCP NewReno</td>
<td>314</td>
<td>645</td>
<td>708</td>
<td>871</td>
<td>1260</td>
<td>1438</td>
</tr>
<tr>
<td></td>
<td>TCP with PiggyCode</td>
<td>198</td>
<td>284</td>
<td>688</td>
<td>832</td>
<td>1398</td>
<td>1808</td>
</tr>
<tr>
<td></td>
<td>CLTSP</td>
<td>139</td>
<td>160</td>
<td>400</td>
<td>505</td>
<td>580</td>
<td>723</td>
</tr>
<tr>
<td>Packets lost due to Retry exceeded (RET)</td>
<td>TCP NewReno</td>
<td>140</td>
<td>219</td>
<td>232</td>
<td>267</td>
<td>330</td>
<td>415</td>
</tr>
<tr>
<td></td>
<td>TCP with PiggyCode</td>
<td>127</td>
<td>264</td>
<td>201</td>
<td>295</td>
<td>429</td>
<td>525</td>
</tr>
<tr>
<td></td>
<td>CLTSP</td>
<td>91</td>
<td>162</td>
<td>190</td>
<td>219</td>
<td>264</td>
<td>308</td>
</tr>
<tr>
<td>Packets lost due to collision (COL)</td>
<td>TCP NewReno</td>
<td>205</td>
<td>408</td>
<td>501</td>
<td>605</td>
<td>721</td>
<td>970</td>
</tr>
<tr>
<td></td>
<td>TCP with PiggyCode</td>
<td>136</td>
<td>237</td>
<td>481</td>
<td>585</td>
<td>860</td>
<td>996</td>
</tr>
<tr>
<td></td>
<td>CLTSP</td>
<td>114</td>
<td>180</td>
<td>322</td>
<td>470</td>
<td>567</td>
<td>918</td>
</tr>
</tbody>
</table>

5.3 Performance Evaluation with Mobility

Although the protocol is developed for static multi hop wireless ad hoc networks, we need to evaluate the performance for mobility. We have taken the scenario of grid topology with 100 (10*10) nodes. We varied the number of flows from 5 to 25 and the pause time is set as 0.1 sec. We have tested two scenarios. In the first scenario, we have set the velocity as 1 m/second.

In the second scenario, velocity is set as 5 m/sec. The throughput is shown in Figs. 13 (a) and (b). In Fig. 13 (a), the velocity is high, so the disconnections happen faster. It results in reduced throughput than Fig. 13 (b). The Mobility leads to frequent route failures. Due to this reason, CLTSP executes probing phase after every route failure to find
out appropriate inter packet delay period for the newly discovered path. It increases the control overhead. Still, CLTSP spreads the traffic uniformly thereby controlling the collision drops and retransmissions. Packet drops that occur due to route failure is same for TCP and CLTSP. But, TCP suffers more from frequent timeouts and retransmissions due to congestion. Piggy code with TCP suffers worst than TCP due to its inability of decoding. The decoding opportunity becomes very less during mobility that leads to poor throughput. CLTSP is able to perform slightly better than TCP and Piggy code with TCP in presence of mobility as shown in Fig. 13.

![Fig. 13. Throughput for multiple flows with mobility of (a) 5 m/sec (b) 1 m/sec.](image)

### 6. CONCLUSIONS

TCP protocol is inappropriate for multi hop ad hoc networks due to its burst nature. It follows ACK clocking mechanism. Whenever it receives ACK, it increases the window size linearly and delivers a group of packets without considering the time interval between packet deliveries. This behavior of TCP leads to severe contentions that unnecessarily create route failures and collisions. We proposed a new cross layer transport solution that delivers data packets such that it reduces the contention among them and improves spatial reuse. Four hop out of interference delay is used as a primary metric for finding the inter packet delivery time period. In addition, we reduce the number of packet transmissions by combining DATA and ACK packets travelling in opposite directions, performing a pseudo broadcast which paves way for added reliability. The Network Coding overhead is very minimal as we go for Opportunistic Coding instead of timed coding of packets in a buffer. We evaluate the performance of our work against TCP New Reno, Piggy code and it shows that our solution outperforms them in terms of throughput, end to end delay, control overhead, collisions and wrong route failures significantly.

### REFERENCES


MITIGATING INTRA FLOW INTERFERENCE OVER MULTI HOP WIRELESS AD HOC NETWORKS

2009, pp. 1706-1715.


Radha Natraj is working as Assistant Professor in the Department of Information Technology, Easwari Engineering College, Chennai, India. She received her Master degree in Computer Science from Anna University in 2007. Her research focuses on mobile ad hoc networks, wireless sensor networks, cross layer design and fuzzy logic system.

Kathiravan Kannan received his M.Tech. degree in Electronics and Communication Engineering from Pondicherry University and Ph.D. in Mobile Ad Hoc Networks from Anna University, Chennai, India in 2001 and 2007 respectively. He is currently the Dean of Easwari Engineering College, Chennai, India. He was an Incharge and Project Coordinator for two All India Council for Technical Education funded research projects. He has nearly 20 years of Teaching Experience in various reputed institutions in India. He is a Convener of IEEE International Conference on Technological Innovations in ICT for Agriculture and Rural Development in the year 2015. He is a lifetime member of IEEE, Institution of Engineering and Technology and Indian Society for Technical Education professional societies. His main research interest includes mobile ad hoc networks, wireless sensor networks, RFID, UAV and robotics.