ENHANCING PERFORMANCE OF TCP IN MULTIHOP NETWORKS

Gnana Prakasi O.S[1] and Dr. Varalakshmi. P[2]

[1]Department of Computer Technology, MIT Campus, Anna University

prakasi_gnanam@yahoo.co.in

[2]Department of Information Technology, MIT Campus, Anna University

varanip@gmail.com

ABSTRACT

It has been observed that TCP suffers from poor bandwidth utilization and extreme unfairness in wireless environment, and the utility of TCP in the multi-hop IEEE 802.11 network has been seriously questioned. The high transmission errors and varying latency in wireless channel would have a seriously adverse effect on the performance of TCP. Thus, a novel cross-layer approach with joint congestion and contention window control scheme and a probabilistic approach for RTT measurement is proposed to improve the performance of TCP in multi-hop networks. The simulation results show that proposed design provides a more efficient solution for frequent transmission loss and enables TCP to distinguish between congestion loss and transmission errors, thus to take proper remedial actions to improve TCP performance in multi-hop network by setting optimal congestion window and updating the window only during congestion.

KEYWORDS

TCP, Multi hop network, congestion window, contention window, RTT measurement

1. INTRODUCTION

Wireless technologies eliminate the requirement of fixed cable infrastructures, thus enabling cost-effective network deployment. In recent years, wireless communication networks have been extensively deployed and are generally specified in accordance with the IEEE 802.11 [9] standard. As a consequence, a lot of relevant efforts are being devoted to the provisioning of reliable data delivery for a wide variety of applications over different wireless infrastructures.

Multi-hop network comprises of number of wireless nodes to transfer packet from source to destination. Multi-hop networks has higher available bandwidth in a multi-rate 802.11 network and the power of transmission at the edges of the 802.11 nodes can be reduced resulting in lower interference with nodes at the edges of the other 802.11 nodes.

1.1 TCP IN MULTI-HOP NETWORKS

Transmission Control Protocol (TCP) [10] is a reliable connection-oriented byte stream transport protocol for the Internet. TCP adjusts well in traditional networks comprising wired link and stationary hosts. It assumes that the congestion in the network is the primary cause for

DOI : 10.5121/ijcnc.2012.4513

packet loss and unusual delay. TCP achieves congestion control or avoidance [17] by regulating the congestion window size in accordance with the estimated network congestion status in order to adjust the sending rate. TCP performs well over such networks by adapting to end-to-end delays and congestion loss.

In multi-hop networks, TCP performance is degraded because of its two unique characteristics namely, location-dependent and spatial reuse. Packets may be dropped due to consistent link-layer contention, resulted from hidden/exposed terminal problem. Thus, packet loss in wireless network can be due to various factors.

However, when packets are lost in networks for reasons other than congestion, TCP results in an unnecessary reduction in end-to-end throughput and suboptimal performance. This problem arises in wireless environments because wireless links have different characteristics with respect to wired ones, in terms of less reliability and time-variant behaviour. Since the only reaction provided by TCP to the event of a non-successful packet delivery is the congestion control mechanism, TCP implementations will sometimes perform poorly in wireless environments.

1.2. PERFORMANCE OF IEEE 802.11 PROTOCOLS

In an 802.11-based multi-hop network, the underlying MAC coordinates the access to the shared wireless channel and provides the link abstraction to upper layer such as TCP. The performance of IEEE 802.11 protocol will be degraded when bit error rate (BER) increases in the wireless channel as well. The fundamental problem comes from the back off mechanism. Unfortunately, wireless transmission links are noisy and highly unreliable. Path loss, channel noise, fading, and interference may result in significant bit errors. What this means is that an unacknowledged frame could result from not only collisions but also frame loss. A proper approach dealing with lost but non-collided frames is to send them again, as quickly as possible. Extending the back off time has no benefit in this situation.

Based on the above observation, when a packet is lost in a wired network because of congestion or is collided in a wireless network, the sender should slow down. When a packet is lost in a wireless network because of noise, the sender should try harder. It is unnecessary for the sender to reduce its TCP congestion window or exponentially increase its back off parameter value for retransmissions. However, if the sender is unable to identify causes of packet loss, it is difficult to make the correct decision. Characteristics of error-prone wireless channel make medium accesses in wireless LANs considerably more complex than in wired networks. The IEEE 802.11 medium access control (MAC) layer provides a reliable communication link by handling packet delivery using the positive ACK and retransmission mechanisms. However, even with the retransmission mechanism in the MAC layer, packets may still be lost without being handed over to the transport layer due to spurious interference or collisions [11–13]. Although TCP can successfully recover dropped packets, recovery routines inevitably degrade the TCP performance since they involve the end-to-end retransmission of the original packet, which leads to an undesirable reduction in the TCP congestion window. Hence, the cross layer interaction between the transport layer and the wireless MAC layer has a critical effect on the detection of erratic errors and on the control of congestion for multi-hop networks.

Thus in the proposed scheme, by differentiating the packet loss due to congestion and transmission error, congestion window is not reset if the packet loss is due to transmission error. In case of congestion, instead of resetting the congestion window, an algorithm is proposed to find optimal round trip time (RTT) based on the probabilistic approach. The congestion window is set based on the RTT value calculated.

The chapters are organised in such a way that chapter II provides a literature review of various papers related to the proposed approach. Chapter III describes the proposed scheme. Chapter IV describes the skeletal framework of this project. Chapter V illustrates the implementation of proposed work with detailed explanation. Chapter VI describes the results generated by simulation. Chapter VII concludes the project for this first phase. It is followed by Reference Section where the details of various papers which are referred are illustrated.

2. Related Work

In wireless networks, channel access contentions may occur between different flows passing through the same vicinity or between different packets within the same flow, which exacerbate the channel contention problem [23, 8]. When multiple packets within the congestion window are lost in wireless links, conventional TCP schemes, such as Tahoe [21], Reno [23] and NewReno [15], etc., are only capable of recovering from single loss event per RTT time, and therefore result in high error recovery delays. In environments prone to significant loss, the performance of application is affected not only by the rate at which TCP restores its transmission, but also by its capability to recover from transmission error. Therefore, the congestion control mechanisms implemented in wired networks may not be entirely suitable for wireless environments. Accordingly, a requirement exists for a well-defined collaborative mechanism between TCP and the MAC protocol to reduce the effect of wireless interference on the TCP performance. One method for dealing with erratic errors on a wireless link is to split the wireless portion of the network from the conventional TCP connection.

The Indirect-TCP scheme (commonly referred as I-TCP) in [1-3] is one example of such scheme. I-TCP splits an end-to-end TCP flow into two separate TCP connections, i.e. a regular TCP connection over the wired network and a wireless TCP connection over the wireless link. Under this approach, any corrupted packets will be retransmitted directly through base stations on the wireless part of the path, and the wired connection is unaware of the wireless losses. Hence, the transport layer is isolated from the erratic behavior of the wireless link. However, a major drawback of this split-connection approach is that it fails to preserve the TCP end-to-end semantics because an ACK originating from the base station may reach the sender side before the corresponding data packet reaches its destination. One method for dealing with cross layer interaction between MAC and TCP is [7] in which a scheme to differentiate loss due to congestion and transmission error is proposed. In this method the contention window size is doubled in case of collision. In this scheme, The resetting of congestion window size due to collision will not be optimized one. So, the performance degradation will be there.

Another approach [18] is based on the decision of congestion window size based on RTT and measured channel bandwidth. Bandwidth and RTT can be measured periodically, if there is a predetermined deviation in the bandwidth or RTT values, we can change the maximum congestion window size to a new one. This deviation value can be determined by considering the trade-off between system overhead and TCP performance. In [22], TCP- DAA approach is described in detail. The technique used for minimizing unnecessary retransmissions by timeout consists of two adjustments:1) the number of duplicate ACKs for triggering a retransmission by the fast retransmit mechanism is decreased from 3 to 2 packets, which is in line with [20] in the sense that we work with a small cwnd limit.2) the regular retransmission timeout interval is increased fivefold for compensating the maximum of four delayed ACKs. Highly noisy scenarios degrade the performance in the TCP-DAA algorithm.

Existing works introduces optimal congestion window and reduces number of control packets. Setting the delay window cause unnecessary delay in the network that reduces TCP performance and hence an optimal delay window is needed. . In this research work, TCP performance in multihop wireless network is proposed to be improved by increasing end-to end throughput and to stabilize channel utility by optimal window and by reducing number of control packets.

3. System Architecture

The Proposed system architecture, in Figure 1, shows the data transfer of packets across a hybrid network from TCP source to TCP receiver with the wireless network being a multi-hop network. A snoop agent, inside the Base Station (BS) monitors every packet that passes through it in either direction. It maintains a cache of TCP packets sent from the TCP source that have not yet been acknowledged by the mobile host. Besides, the snoop agent also keeps track of the last acknowledgment sent from the mobile host.

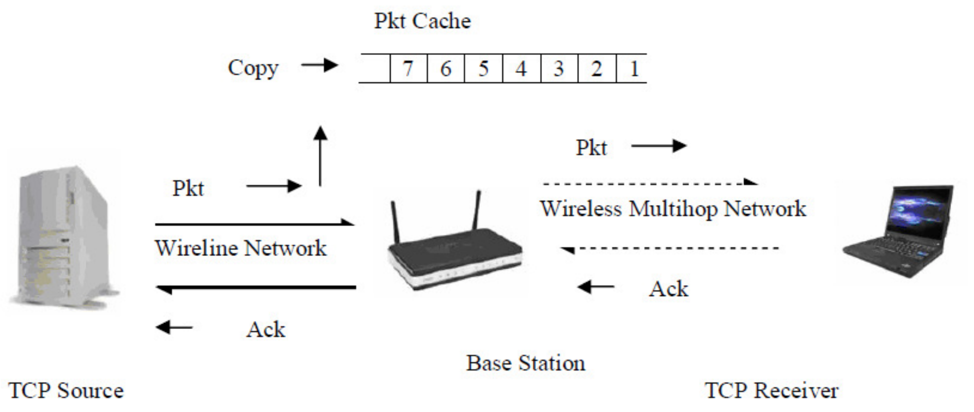

Figure1. System Architecture

4. Proposed Framework

4.1 Estimate Channel Status

In order to exploit the information about the actual channel status, we define the probability of transmission failure (*pf*), to be the probability that a frame transmitted by the station of interest fails. It is noted that busy period in the channel status includes collision, frame loss, and successful transmission.

Since the absence of an immediate positive acknowledgement following each data frame will be regarded as a failed transmission, *pf* can be obtained by counting the number of observed transmission failures, divided by the total number of transmission attempts on which the measurement is taken. Therefore, the probability of transmission failure can be defined as given in equation (1) [7] as,

$$pf = \frac{\text{Number of transmission failures}}{\text{Number of transmission attempts.}} \qquad \text{-- (1)}$$

Now let us try to estimate the probability of transmission collision (pc), which is given as,

$$pc = 1 - \frac{1-pf}{1-\text{FER}} \qquad \text{-- (2)}$$

Where pf defines Probability of Transmission failure and FER defines Frame Error Rate, which is set as 0.2 in our case.

4.2 Differentiating Congestion and Noise

4.2.1 Packet/ACK Arrival from TCP Source/Destination:

When a TCP data packet is received from wire-line network, the snoop agent will cache the TCP data packet in base station and monitor packet transfer at the base station. If the packet is lost in the wireless medium, the base station will automatically retransmit the TCP data packet and avoids TCP senders to retransmit again. When the base station forwards a packet to a TCP receiver, the snoop agent will evaluate the probability of packet collision. If the packet is lost, the snoop agent will use the previous evaluation to adjust the contention window of the MAC layer. Also, a report stating the reason of packet lost will be sent to the TCP sender to help the TCP sender with resetting its congestion window size.

When snoop agent receives a TCP data packet, it will determine if the packet has been received and cached in the base station. If the packet has not been received before, then Snoop agent will temporarily cache the packet into the buffer in base station and if this packet is in-order the snoop agent will forward the packet to the TCP receiver (case 1: normal case). But if in out-of-order (e.g. the packet arrival sequence is 1, 2, and then 5), then snoop agent will assume that the loss is in the wire-line TCP connection. Now the snoop agent will forward this data packet to the TCP receiver and uses traditional congestion control mechanisms to retransmit the lost packet (case 2: packet lost in wired connection).

If the TCP data packet has been received by snoop agent but has not been received by the TCP receiver then this represents that the packet was lost in the wireless connection. In this case, snoop agent will evaluate the collision rate (*pc*) in the wireless channel. If *pc* is greater than the Frame Error rate, snoop agent will determine that the packet loss is due to collisions, hence Snoop agent compute Round Trip Time (RTT) based on probabilistic approach on sent packets and acknowledgements received. Congestion window and contention window is calculated based on the optimal RTT. Congestion window size that is set is transmitted to the TCP sender. (Case 3: packet lost in wireless connection). If the *pc* value is less than the frame error rate the congestion window size will not be changed since the loss due to transmission error and hence retransmit the packet. (Case 4: packet lost in wireless connection due to transmission error).

If the packet has been received (i.e., is a retransmitted packet) and the last ACK number cached on the snoop agent is greater than or equal to the sequence number of the received TCP data packet. This represents that the TCP receiver has received the packet. Thus, snoop agent will directly drop the packet and send the last ACK number to the TCP source.(case 5: ACK lost in wired TCP connection).

When it receives ACK from the snoop agent will inspect the arrived ACK. If the ACK is new (i.e., ACK number > last ACK number) the snoop agent will forward the ACK to the TCP source and subsequently clear the packet cached in the base station (case 1 and case 2). If the ACK is a duplicate, and if the packet was lost in the wire-line network (i.e. snoop agent did not receive the next packet predetermined by the TCP receiver), the ACK packet will be forwarded directly to the TCP source (case 3).

Conversely, if the packet was lost in the wireless connection, snoop agent will compute round trip time (RTT) based on probabilistic approach on sent packets and acknowledgements received. Congestion window and contention window is calculated based on the optimal RTT. A negative acknowledgement (NAK) option [29] and congestion window size will be added in the ACK (associate with the same TCP connection) and the ACK will be forwarded to the TCP source at a later time (case 4).

Hence, by inspecting the previous ACK sequence number and historical NAK information, the TCP sender clearly identifies corruption losses in the wireless links and therefore determines whether invoking congestion control mechanisms is necessary. In sum, to reduce the effect of contention on the transmission efficiency of the wireless network, this study proposed a method of calculating the collision rate by observing the access status of the channel, and thus adjusting the contention window size.

Moreover, if the transmission error on the wireless link causes packet loss continuously, the Snoop agent would generate NAK by cross-layer, and use the messages sent by NAK to notify the TCP sender end that packets are dropped on the wireless link (cross-layer mechanism). This cross-layer support from the MAC layer protocol ensures that the transport layer protocol is aware of the transmission error in the link layer. It is expected to improve the transmission efficiency of TCP on wireless network.

5. IMPLEMENTATION

The packet from TCP source (wired) to TCP receiver (wireless) transfers through snoop agent. At snoop agent, based on the packet sequence number and acknowledgement it received, it differentiates the loss due to congestion and transmission error.

5.1 DIFFERENTIATING CONGESTION LOSS AND TRANSMISSION ERROR

This describes a mechanism to differentiate congestion loss and transmission error at the snoop agent. Cross layer approach between MAC and TCP is used to invoke this mechanism. The overall flow chart for module 1 is given in figure 2. It calculates the followings.

Successful transmission in MAC layer.

Calculate Probability of collision.

MAC layer transmission of collision probability to snoop agent.

Decision Making

5. 1.1 Successful transmission in MAC layer: In the MAC layer of the intermediate nodes in the multi-hop network, check the channel status to see if it is idle. The numbers of attempts it had failed to provide successful transmissions are calculated in each node whenever it attempts to make a transmission.

$$pf = \frac{\text{Number of transmission failures}}{\text{Number of transmission attempts.}} \qquad \text{-- (1)}$$

5.1.2 Calculate Probability of collision: Probability of transmission failure (*pf*) is calculated at each node based on the number of transmission failures and number of attempts it made to make a transmission as in equation (1). Probability of collision (*pc*) is found at various nodes by equation (2).

$$pc = 1 - \frac{1-pf}{1-FER} \qquad \text{-- (2)}$$

5.1.3 MAC layer transmission of collision probability to snoop agent: Each node whenever it sends acknowledgement for packet it received in MAC layer, it sends its collision probability value. The collision probability value which is received in the acknowledgement is compared with its own probability of collision and sends the maximum value to its next hop. The maximum collision probability that is found in the entire multi-hop link throughout its transmission from receiver to snoop agent is transferred to the snoop agent.

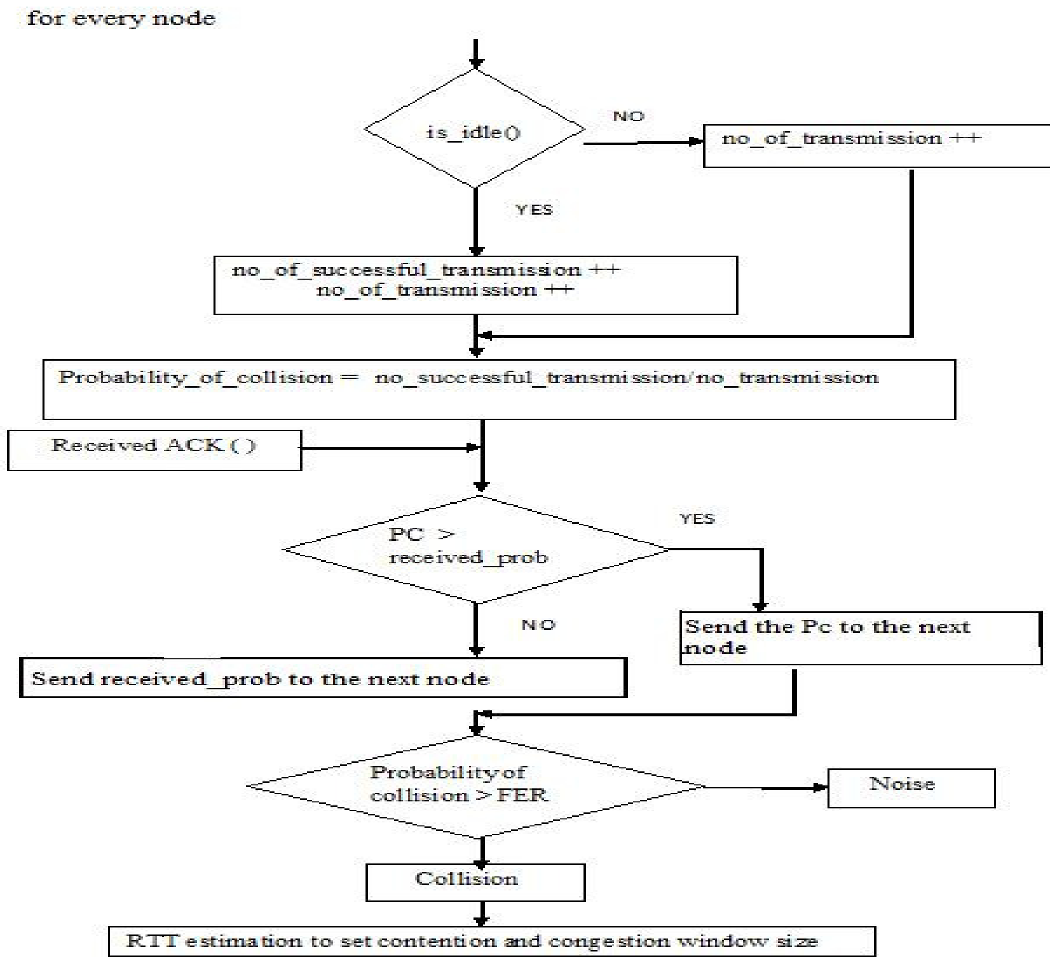
for every node
is_idle()
NO
no_of_transmission ++
YES
no_of_successful_transmission ++
no_of_transmission ++
Probability_of_collision = no_successful_transmission/no_transmission
Received ACK ( )
PC > received_prob
YES
NO
Send the Pc to the next node
Send received_prob to the next node
Probability of collision > FER
Noise
Collision
RTT estimation to set contention and congestion window size

Figure 2 Flow Chart for differentiating loss due to congestion and noise

5. 1.4 Decision Making: In the snoop agent, the value of collision probability is compared with Frame Error Rate (FER) which is set to 0.2 in our case. If collision probability is greater than FER, the loss is assumed to be based on collision which invokes PA-RTT approach else it is due to noise and just retransmits the packet.

5.2. CALCULATE THE OPTIMAL RTT VALUE

Calculation of round trip time (RTT) is based on the history of sent packets and acknowledgements received at any instant of time. Given the two sequences $S = \{s_1, s_2, ., s_n\}$ and $D = \{a_1, a_2, ., a_m\}$,where $s_1, s_2, ., s_n$ are send packets and $e_1, e_2, ., e_m$ are ACK packets corresponding to the send packets in S. If a packet s_i in S is acknowledged by a packet a_j in D, we denote $s_i \alpha a_j$.

If all the packets are captured on a host in a connection chain at a period of time, the following conditions must be satisfied:

(1) Any send packet in S must be acknowledged by one or more packets in D; similarly, any ACK packet in D must acknowledge one or more send packets in S;

(2) Packets both in S and E are stored in chronological order;

(3) For any two packets s_i, s_j in S and a_p, a_q in D,

if s_i α a_p, s_j α a_q,and $i < j$, then we have $p \leq q$.

Condition (1) indicates that the relationships between send and its corresponding ACK packets may be one-to-one, manyto-one, or one-to-many. RTT of a send packet can be defined as the gap between the timestamp of a send packet and that of its corresponding ACK packet if the relationship between them is one-to-one.

However, if the relationship is many-to-one or one-to-many, the gap is not unique. If there is k send packets from s_i to s_{i+k-1} acknowledged by a_j, the RTT of those send packets is defined as the gap between the timestamp of s_{i+k-1} and that of a_j, i.e., the smallest gap. Similarly, if a send packet s_i is acknowledged by k packets a_j to a_{j+k-1}, only packets s_i and a_j are involved in the RTT definition of s_i.

Conditions (2) and (3) guarantee that send packets must be replied sequentially. Each send packet must be acknowledged by one or more packets successfully at one time, and the value of a send packet RTT must be positive.

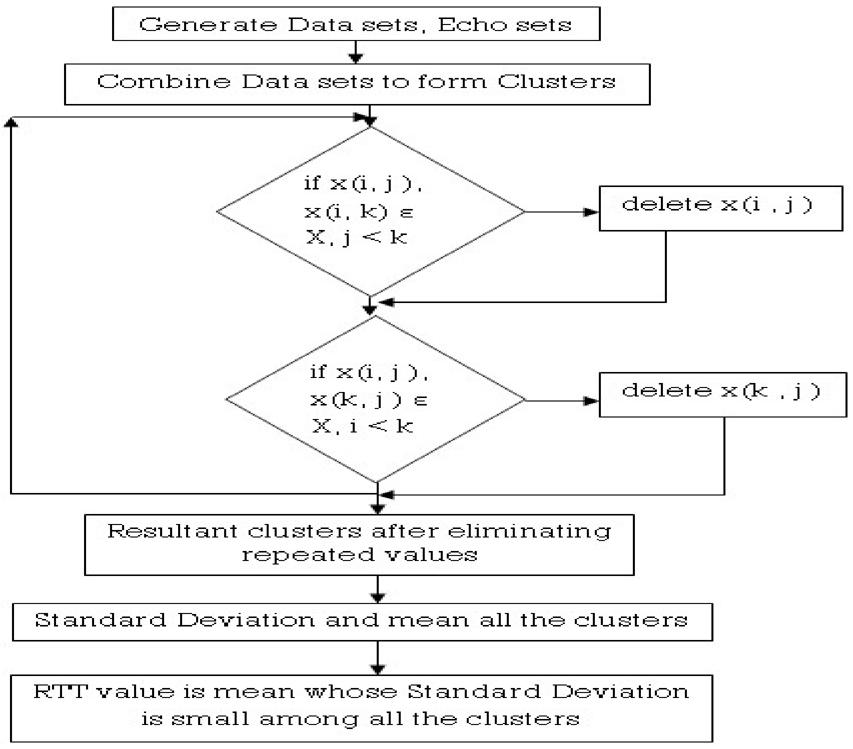
Generate Data sets, Echo sets
Combine Data sets to form Clusters
if x(i, j), x(i, k) ε X, j < k
delete x(i , j)
if x(i, j), x(k, j) ε X, i < k
delete x(k , j)
Resultant clusters after eliminating repeated values
Standard Deviation and mean all the clusters
RTT value is mean whose Standard Deviation is small among all the clusters

Figure 3 RTT Estimation

Implementation is done as
Formation of Data set
Cluster Formation
Compute optimal RTT

5.2.1Formation of Data set: To form data set compute the gaps between the timestamp of each ACK packet in D and that of all the send packets in S. Eliminate the negative values since RTT must be positive. Now group these differences in sets according to each ACK packet in D, forming data sets D_1, $D_{2,..,}$ D_m for ACK packets a1, a2, .,am, respectively.

Sent Packets History

packet_id	time_sent	corresponding_ack_id
0	1	0
1	2	0
2	3	0
3	6	2
4	7	2

Acknowledgement Packets History

Ack_id	time _sent	sent_ids
0	4	0 1 2
1	5	------
2	8	3 4

Figure 4 History of Send and ACK Packets

$D_1 = \{s_1\ a_1, s_2\ a_1, ., s_n\ a_1\}$,

$D_2 = \{s_1\ a_2, s_2\ a_2, ., s_n\ a_2\}$,

.

.

.

$Dm = \{s_1\ a_m, s_2\ a_m, ., s_n\ a_m\}$, where element $s_i\ a_j$ in D_j represents the gap $a_j – s_i$ between timestamp of the jth ACK packet in D and that of the ith send packet in S, where $1 \leq i \leq n$ and $1 \leq j \leq m$.

From this history form the Data set based on ACK Packets is given in figure 5. The Data set is formed by grouping all data packets that has been sent before receiving an ACK packet. In this example ACK0 is used acknowledge data packets 0, 1, 2, which forms the first Data Set. ACK1 acknowledges packets 0, 1, 2 which forms second Data Set and so on.

Form the data set find the Cluster by grouping Data Set, take one element from each Data Set and store them as a cluster. In the given scenario select first element in data set 1, second element in data set 2, third element in data set 3 to form Cluster1. To construct cluster C_u, take one element from each data set D_j $(1 \leq j \leq m)$, and store them to C_u according to the chronological order of the ACK packets in D. Now find all possible combinations, there will be up to nm possible clusters, but only one of them represents the correct RTTs for all ACK packets.

5.2.2 Formation of clusters: Each send packet can be acknowledged by one or more packets successfully at one time, which indicates that in each data set D_j there is only one element to represent the real RTT of that send packet. To construct cluster C_u, take one element from each data set D_j ($1 \leq j \leq m$), and store them to C_u according to the chronological order of the ACK packets in D. Now find all possible combinations, there will be up to nm possible clusters, but only one of them represents the correct RTTs for all ACK packets.

$C_1=\{s_1a_1, s_2a_2, \ldots, s_na_m / (s_1 \varepsilon D_1) \infty (s_2 \varepsilon D_2) \ldots (s_n \varepsilon D_m) \infty (s_1 < s_2 \ldots < s_n)\}$

$C_2=\{s_2a_1, s_3a_2, \ldots, s_na_m / (s_2 \varepsilon D_1) \infty (s_3 \varepsilon D_2) \ldots (s_n \varepsilon D_m) \infty (s_1 < s_2 \ldots < s_n)\}$

.

. .

$D_n^m=\{s_na_1, s_{n+1}a_2, \ldots, s_{n+m}a_m / (s_n \varepsilon D_1) \infty (s_{n+1} \varepsilon D_2) \ldots (s_{n+m} \varepsilon D_m) \infty (s_1 < s_2 \ldots < s_n)\}$

0th DATA set

sent_packet_id	Acknowledgement_id	TimeStamp
0	0	3
1	0	2
2	0	1

1th DATA set

sent_packet_id	Acknowledgement_id	TimeStamp
0	1	4
1	1	3
2	1	2

2nd DATA set

sent_packet_id	Acknowledgement_id	TimeStamp
0	2	7
1	2	6
2	2	5
3	2	2
4	2	1

Figure 5 Data Set

Each cluster C_u ($1 \leq u \leq n^m$) has m elements while some of them may share a same ACK or send packet. Remove the send (ACK) packets which share the same ACK (send) packets but keep the one with smaller gap. Eventually we can get C_u ($1 \leq u\ s \leq n^m$) with each element relating to unique send and ACK packets.

5.2.3 Real Time Scenario: Figure 4 shows the history of acknowledgement packets for the set of sample sent packets. Each packet sent is recorded with the fields: sent packet id, time at which the packet is sent (taken as integers) and corresponding acknowledgement id. Each acknowledgement packet is also marked with the fields: acknowledgement id, the time at which the acknowledgement is sent (taken as integers), the sequence of sent packets that it acknowledges. Figure 5 shows the data set generated for each acknowledgement packet with the timestamp being the difference between the time at which the acknowledgement packet was sent to the corresponding sent packet time.

```
.......................................................................
0th CLUSTER set
.......................................................................
sent_packet_id  Ack_id  TimeStamp
.......................................................................
  0              0              3
  1              1              3
  2              2              5
.......................................................................
1th CLUSTER set
.......................................................................
sent_packet_id  Ack_id  TimeStamp
.......................................................................
  1              0              2
  2              1              2
  3              2              2
.......................................................................
2th CLUSTER set
.......................................................................
sent_packet_id  Ack_id  TimeStamp
.................................................
  2              0              1
  3              2              2
.......................................................................
3th CLUSTER set
.................................................... ......................
sent_packet_id  Ack_id  TimeStamp
................................................ .....................
  0              1              4
  1              2              6
................................................ .......................
4th CLUSTER set
......................................... ...........................
sent_packet_id  Ack_id  TimeStamp
.................................................
  1              1              3
  2              2              5
.................................................
```

```
5th CLUSTER set
...........................................
sent_packet_id  Ack_id  TimeStamp
................................................
  2              1              2
  3              2              2
..............................................
6th CLUSTER set
...........................................
sent_packet_id  Ack_id  TimeStamp
................................................
  0              2              7
...............................................
7th CLUSTER set
...........................................
sent_packet_id  Ack_id  TimeStamp
................................................
  1              2              6
...............................................
8th CLUSTER set
...........................................
sent_packet_id  Ack_id  TimeStamp
................................................
  2              2              5
...............................................
9th CLUSTER set
...........................................
sent_packet_id  Ack_id  TimeStamp
................................................
  3              2          2
...............................................
10th CLUSTER set
...........................................
sent_packet_id  Ack_id  TimeStamp
................................................
  4              2          1
```

Figure 6 Initial Cluster Set

Figure 6 and 7 shows initial and final cluster set for the given scenario. Final cluster ensures that the redundant data which are either 1)the same sent packet acknowledged by 2 acknowledgement packets where in the 1st acknowledgement is being removed. 2) the acknowledgement packet acknowledging 2 sent packets wherein the 1st sent packet is being taken. In this way, the final cluster has the unique sent-acknowledgement packets combinations.

2.3Finding optimal RTT based on standard deviation*:* Each cluster C_u is a candidate of the RTTs for the send packets in S while only one of them is the optimal one or close to the real one with high probability. We select the one with smallest standard deviation among all clusters C_u ($1 \leq u \leq n^m$) to be the RTTs of the packets in S.

```
..................................................................
0th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
...............................................
     0            0          3
     1            1          3
     2            2          5
..................................................................
..................................................................
1th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
..................................................................
     1            0          2
     2            1          2
     3            2          2
..................................................................
2th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
..................................................................
     2            0          1
     3            2          2
..................................................................
3th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
...............................................
     0            1          4
     1            2          6
..................................................................
4th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
..................................................................
     1            1          3
     2            2          5
```

```
..................................................................
5th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
..................................................................
     2            1          2
     3            2          2
...............................................
6th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
..................................................................
     0            2          7
..................................................................
7th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
..................................................................
     1            2          6
..................................................................
8th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
..................................................................
     2            2          5
..................................................................
9th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
..................................................................
     3            2      2
..................................................................
10th FINALCLUSTER set
..................................................................
sent_packet_id  Ack_id  TimeStamp
..................................................................
     4            2          1
```

Figure 7 Final Cluster Set

From data set we calculate the RTT from all clusters.

$$\text{Mean}_u=\sum t(i,j)_u/N \text{ for } (1\le u\le n^m), \qquad \text{- - (3)}$$

N is number of elements in $C_{u.}$

$$\text{Standard\_deviation}_u = \sqrt{(x-\text{mean}_u)^2/N}, \text{ where } C \,\varepsilon\, C_u . \qquad \text{- - (4)}$$

$$\text{Optimal RTT}= \text{mean}_u \text{ with smallest } SD_{u.} \qquad \text{- - (5)}$$

```
Standard Deviation
            ............................
            cluster id      SDvalue
            ..............................
            0                    0.942809
            2                    0.5
            3                    1
            4                    1
            5                    0
            6                    0
            7                    0
            8                    0
            9                    0
            10                   0
            ..............................
................................................................................
RTT                      Standard Deviation value
...............................................................................
3.66667             0.942809
1.5                 0.5
5                   1
4                   1
.............................................................................
Minimum RTT value is1.5
```

5. 3. SET CONTENTION WINDOW AND CONGESTION WINDOW

Contention window size is determined based on the RTT value calculated. Contention window size is increased or reduced relative to the RTT calculated. The contention window size is sent to the nodes in the wireless channel along the TCP packets sent. Contention window value received is updated in the MAC layer for every node.

Retransmission timer is set based on the RTT calculated. When the timer fails, the congestion window is set as per Additive increase Multiplicative decrease mechanism. In case of successful transfer, congestion window is decreased. Congestion window size determined is sent to the TCP sender to adjust the Cwnd size of sender.

6. SIMULATION RESULTS

The simulation is done in NS2 simulator. The simulation is carried out in the multi-hop network with wired and wireless nodes which forms the multi-hop network. The simulation is carried out with a wired node being the TCP source and a wireless node as the TCP receiver. A wired node acts as a base station in which snoop agent is implemented. The topology diagram for the simulation is shown in figure 8. In this simulation we assume the frame error rate as 0.2 and it is compared with 2 existing techniques.

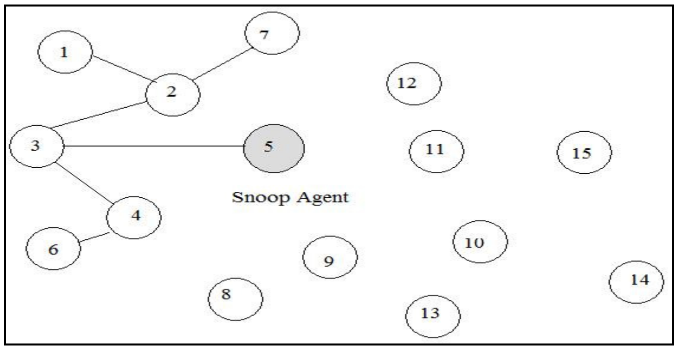

Figure 8 Topology Diagram

The graph shows packet losses due to noise, congestion and total number of packets lost in our proposed scheme, which is compared with 2 existing techniques based on bandwidth measurement [18] and DAA [24] approach.

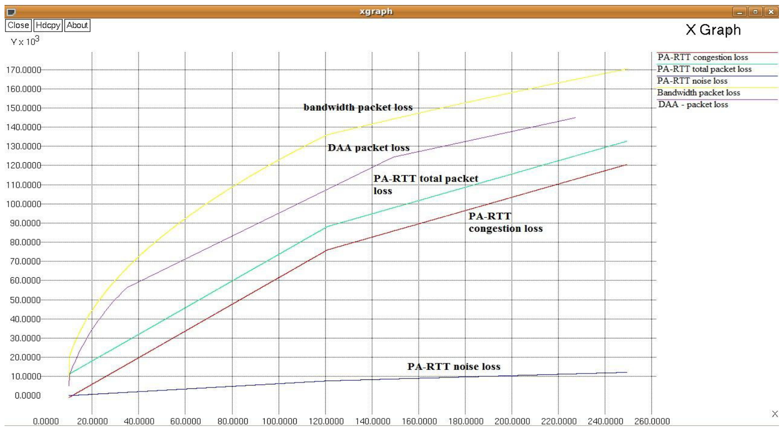

Figure9 Packet Loss (Time Vs Loss)

The graph for Cnu [18] depicts that the amount of packets lost is much higher than DAA and the proposed scheme. As the proposed scheme adjusts the congestion window size only in the case of loss due to congestion and also the RTT calculated is an optimal one, the amount of packet lost is reduced.

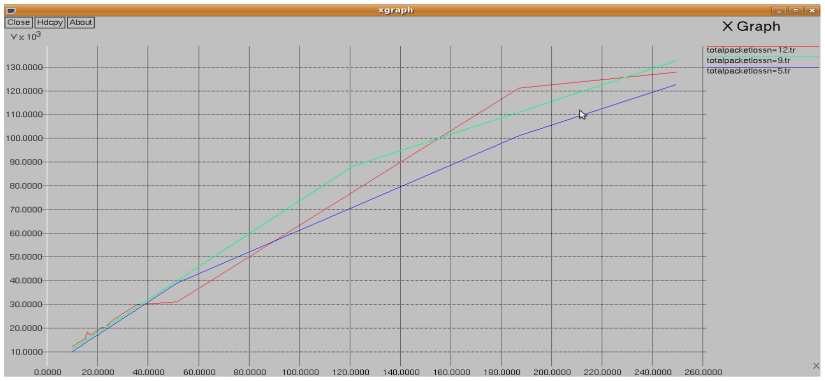

Figure 10 Increasing number of nodes (Number Of nodes Vs loss)

The graph signifies that the amount of packets lost due to noise increases as time increases and it clearly demonstrates that the congestion window need not be reset in case of noise loss. The optimal congestion window and contention window size is set based on the calculated optimal RTT. The graph in figure 11 describes that the proposed scheme implementation prevents the sudden increase and decrease in the congestion window size in the multi-hop network by setting the congestion window based on the RTT calculated in probabilistic manner.

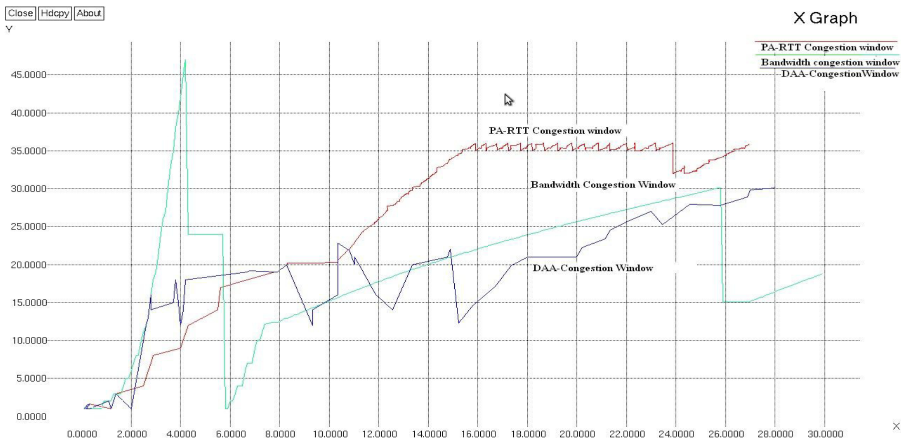

Figure 11 Congestion window Size(Time vs Congestion Window)

As the RTT calculated is an optimal one, the congestion window size changes gradually. As the maximum congestion window alone changes in the Cnu[18], the peek value of the congestion window size changes abruptly which increases the packet loss in the previous case.

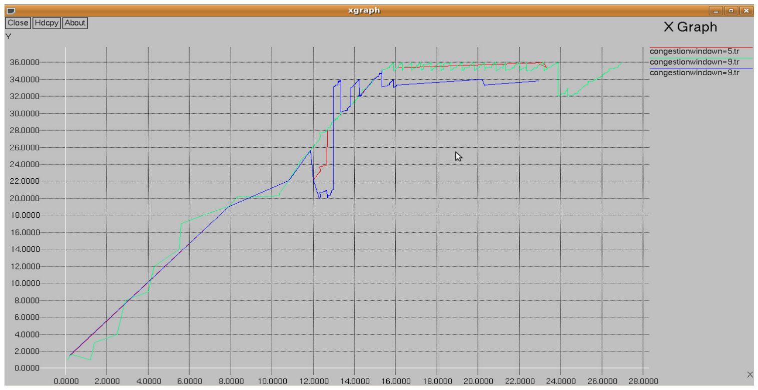

Figure 12: Increasing Number of nodes (number of nodes vs congestion window)

From the graph in figure12 it is clear that the throughput is maintained even in the noisy environment in the proposed scheme. But in the Cnu[18], the throughput decreases after attaining the peek value when the traffic in the flow of data increases. It is clear from the graph that the throughput is increased in the proposed scheme.

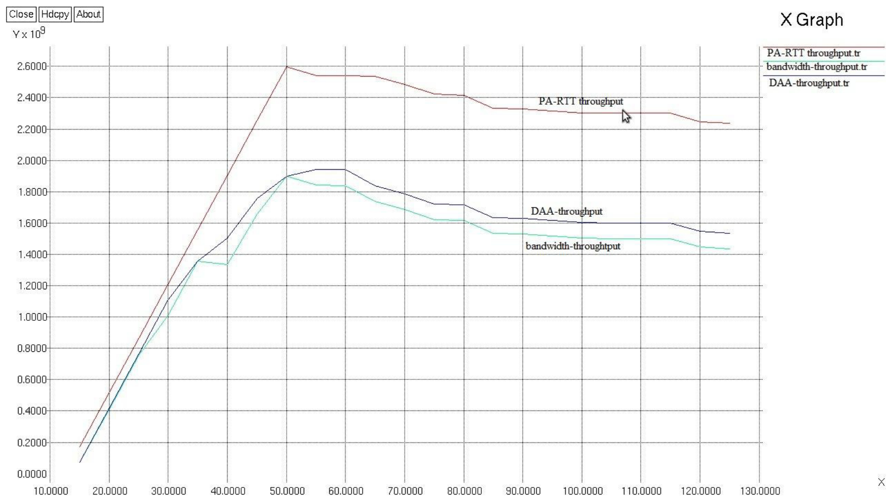

Figure 13: Throughput Measurement (Time Vs Throughput)

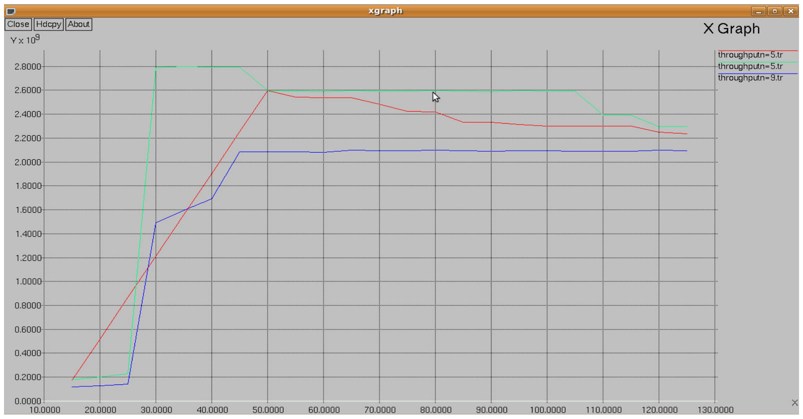

Figure 14: Increasing number of nodes (Number of nodes vs throughput)

Figure 10, 12, 14 shows the measurement of packet loss, congestion window and throughput for increase in the number of nodes.

7. CONCLUSION

The shared channel contention and erratic packet loss usually lead to a curbing of window size on the TCP, and thus limit the performance of TCP in multi-hop networks. This paper proposed a scheme to differentiate the loss due to congestion and transmission error and a probabilistic approach to find the optimal RTT to set congestion window size. The results show that implementation prevents the sudden increase and decrease in the congestion window size in the multi-hop network and avoids unnecessary resetting of congestion window. Thus the TCP performance improvement occurs in noisy scenarios which are prevalent by which it outsmarts other techniques. The future work includes extending this work in dynamic environment.

REFERENCES

[1] Bakre, A. V., & Badrinath, B. R. (1997). "Implementation and performance evaluation of indirect TCP". *IEEE Transactions on computers*, *46*(3), 260–278.

[2] Balakrishnan, H., Seshan, S., Amir, E., & Katz, R. H. (1995). I"mproving TCP/IP performance over wireless networks". ACM MOBICOM, pp. 2–15.

[3] Balakrishnan, H., Padmanabhan, V. N., Seshan, S., & Katz, R. H. (1997). "A comparison of mechanisms for improving TCP performance over wireless links". *IEEE/ACM Transactions on Networking*, *5*(6), 756–769.

[4] Balakrishnan, H., & Katz, R. H. (1998). "Explicit loss notification and wireless web performance". IEEE GLOBECOM, pp. 172–179.

[5] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithm, performance," *IEEE/ACM Trans. Networking*, vol. 14, no. 6, pp. 1246-1259, Dec. 2006.

[6] Chen et al *2003* “On setting TCP’s congestion window limit in mobile ad hoc networks,” [7] Cheng, R. S., & Lin, H. T. (2008). “A cross-layer design for TCP end-to-end performance improvement in multi-hop wireless” networks.*Journal of Computer Communications*, *31*, 3145–3152.

[8] Colandairaj, J., Irwin, G. W., & Scanlon, W. G. (2007). “Wireless networked control systems with QoS-based sampling”. *IET Control Theory & Applications*, *1*(1), 430–438.

[9] Der-Jiunn Deng, Rung-Shiang Cheng,Heng-Jia Chang,Hui-Tang Lin, Ruay-Shiung Chang (2009) “A cross-layer congestion and contention window control scheme for TCP performance improvement in wireless LANs” *Telecommunication systems*.Springerlink.

[10] Dimitri, B., & Robert, G., (1992). *Data networks*, 2nd edn. pp. 1–556. New York: Prentice Hall.

[11] Elkhart, D. A., & Steenkiste, P. (1996).” Measurement and analysis of the error characteristics of an in-building wireless network”.ACM SIGCOMM, pp. 243–254.

[12] Ehsan Hamadani, Veselin Rakocevic, “Evaluating and Improving TCP Performance against Contention Losses in Multi-hop Ad Hoc Networks”.

[13] F. Ge, L. Tan, and M. Zukerman, “Throughput of FAST TCP in asymmetric network,” *IEEE Commun. Lett.*, vol. 12, no. 2, pp. 158-160.

[14] First A. N.Shanthi, Second B. Dr.LGanesan and Third C. Dr.K.Ramar**, “**Performance Improvement Of Reliable TCP Using Cross - Layer Interaction In Multi-hop Wireless Networks**”**, IEEE.

[15] Floyd, S., & Henderson, T. (1999). “The NewReno modification to TCP’s fast recovery algorithm”. *RFC*, *2582*, 1–12.

[16] Floyd, S. (1994). “TCP and explicit congestion notification”. *ACM Computer Communication Review*, *24*(5), 10–23.

[17] F. Ge and L. Tan, “A partial super fast recovery algorithm for FAST TCP,” in *Proc. 2nd International Conference on Wireless Broadband and Ultra Wideband Communications*, pp. 57-61, Aug. 2007.

[18] In Huh , Jae Yong Lee and Byung Chul Kim **, “**Decision of Maximum Congestion Window size for TCP performance Improvement by Bandwidth and RTT measurement in Wireless Multi-hop Networks**” ,***International Journal of Information Processing Systems , March 2006*

[19] Kliazovich, D., & Granelli, F. (2006). “Cross-layer congestion control in ad hoc wireless networks”. *Ad Hoc Networks*, *4*(6), 687–708.

[20] K.-C. Leung and V. O. K. Li, “Transmission control protocol (TCP) in wireless networks: issues, approaches, and challenges,” *IEEE Commun. Surveys & Tutorials*, pp. 64-79, 4th Quarter 2006.

[21] Postel, J. B. (1981). “Transmission Control Protocol”. RFC 793, pp.1–85.

[22] Ruy de Oliveira and Torsten Braun, “A Dynamic Adaptive Acknowledgment Strategy for TCP over Multihop Wireless Networks”, 18th International Conference on Distributed Computing Systems (ICDCS*).*

[23] Yang, X., & Nitin, H. V. (2006).” A wireless MAC protocol using implicit pipelining”``````````. *IEEE Transactions on Mobile Computing*, *5*(3), 258–273.