Improving TCP Performance in Multi-Hop coded Wireless Networks

Khaled Alferaidi[1] , Robert Piechocki [2] and F. Alfordy [3]

Abstract— **Wireless Multi-Hop networks often rely on the use of the Transport Control Protocol (TCP) which provides high reliability and throughput performance at Transport Layer. Xor network coding does not benefit greatly from TCP's attributes. TCP takes into consideration the nature of networks; however, TCP does not acknowledge the network coding which can ease congestion by performing coding at intermediates nodes as in COPE. Meanwhile, the perspective of network coding has a more rigid approach that depends on relay nodes to perform coding rather than depending on the sender and the receiver exchange messages. In this paper, we investigate the different perspective to bridge the gap, as our understating is moved from throughput to Ratio of Loss Packet (RLP), delay and jitters. So we propose WeNC-TCP to appreciate the network coding nature functionality and incorporate better interaction between TCP at Transport layer, and network coding functionality which relies on between network and Data link layer, as a result, the average ratio of loss packets has improved up to 72%, and end-to-end Delays have reduced to up to 48%.**

I. Introduction

Network coding is a new transmission paradigm, that was introduced by Ahlswede et al. [1]. The beauty of Ahlswede novel concept is bandwidth efficient, and it achieves higher throughput than traditional transmission theorem [2][5]. The basic idea about network coding is that packets are intelligently mixed (coded) together at an intermediate node into one coded packet. Consequently, it is broadcasted to the network. This process opens the door to reducing the number of transmissions, also gives the flow of rich information per transmission. As a result, bandwidth becomes available for new data to be transmitted [7]. Many researchers have been attracted to wireless networks as the fact that, the nature of wireless is a broadcast environment, and network coding benefits from such feature.

Network coding divides into three different categories: random network coding [3],[6],[7], physical layer network coding [8],[9], and Xor network coding [7],[10],[11]. In the first case, encoding coefficients are randomly chosen from a set of coefficients of a finite field. A linear combination of packets data is processed to generate a coded packet. At the receiver, the decoding process is done if and only if the receiver can construct a full rank transfer matrix made of coefficients extracted from received coded packets [6]. It means that if there are K encoding coefficients in a coded packet, the receiver will be able to decode all encoded packets. On the other hand, Physical layer network coding allows for simultaneous reception of packets at the nodes air interface [8] which are aggregated. Before broadcasting frames, a node encoded frames from a different source using Xor operation. Lastly, Xor network coding, a native packet in the node's queue is encoded from different sources using Xor operation. After encoding, the packet is transmitted to destination nodes. At the receiver, it decodes encoded packets with its own sent packets in the queue to retrieve native packets using the same Xor operation [4]. A reception report is used by all nodes to coordinate the process of "who has what packets". Reception reports help nodes to decode packets and track the native and encoded packets on medium [4]. Xor network coding is regarded as a special case of random network coding where the Galois field is of size 2 [8].

This paper has been inspired by the work of Katti et al. [4]. COPE was the first attempt to test the practicality of network coding. However, TCP protocol is not as expected to enjoy the benefit of Xor network coding. Even though TCP is a resilient and network friendly protocol that eases the congestion and guarantee smooth delivery of data packets, it does not guarantee to give the same expectations when network coding is being used in the wireless networks. The rest of this paper is organised as follows. Section II describes state of the art, section III problem statement. The proposed scheme is presented in section IV. Simulation results and discussion are presented in Section IV, and finally, we conclude our work in Section VII.

II. Related Work

Katti et al. [4] proposed a new architecture in wireless mesh networks that put the concept of network coding at the test. The work in [4] succeeded to present COPE. They considered unicast traffic, dynamic and potentially bursty flows in which COPE demonstrated significant performance improvement. However, COPE showed little or no significant improvement in TCP, unlike UDP improvements using COPE. Authors in [4] claimed that TCP did not show any significant throughput improvement (the average gain was 2-3%), their explanation formed based on reaction to collision-related losses. Katti argued that many nodes were sending packets to the bottleneck nodes, but they were not within carrier sense range of each other. This was due to the classic hidden terminals problem. Therefore, many collision-related losses cannot be mended with the maximum number of MAC retries. This causes TCP flows experience 14% loss. As a result, the TCP flows suffered timeouts and excessive increase in back-off values, and could not utilise the medium efficiently.

[1] K. Alferaidi is with Faculty of Engineering, University of Bristol & Yanbu University College Khalidmalaya at gmail

[2] R.Piechocki is s with Faculty of Engineering, University of Bristol, Woodland Road,BS8 1BU, UK R.J.Piechocki at Bristol

[3] F. Alfordy University of Hail, K.S.A. F.Alfordy at hail

978-1-7281-0108-8/19/$31.00 ©2019 IEEE

Authors in [5], proposed TCP/NC and argued that current approaches such as rateless codes and batch-based coding are not compatible with TCPs retransmission and sliding-window mechanisms. TCP/NC incorporated network coding into TCP with only minor changes. The main aim of [5] is to mask losses from TCP using random linear coding. The sender adapts TCP source and buffers packet into an encoding buffer which represents and used as the coding window. The receiver must acknowledge them. The sender generates and sends random linear combinations of the packets in the coding window. The coefficients used in the linear combination are also included in the header. The decoder acknowledges the last seen packet by requesting the byte sequence number of the first byte of the first unseen packet, by using a regular TCP ACK. Moreover, the port and IP address information for sending ACK to the sender, they could be obtained from the SYN packet at the beginning of the connection. Any ACKs generated by the receiver TCP are not sent to the sender. They are instead used to update the receive window field that is used in the TCP ACKs generated by the decoder [5].

Authors in [8] introduced a robust and resilient TCP aware network coding with opportunistic scheduling in wireless mobile Adhoc networks. More specifically, they considered a TCP parameter, congestion window size, and wireless channel conditions to improve TCP throughput performance. Their Simulation results showed that using traditional network coding and combined with opportunistic scheduling could improve the performance of TCP up to 35% in wireless mobile Adhoc networks[8].

Meanwhile, Authors in [12] investigated the benefit of network coding for TCP traffic in a wireless mesh network. They implemented their network coding in 802.11a wireless mesh network and measured TCP throughput. Unlike previous implementations of network coding, they used off-the-shelf hardware and software and did not modify TCP or the underlying MAC protocol. They showed that network coding not only reduces the number of transmissions by sending multiple packets via a single transmission but also results in a smaller loss probability due to reduced contention on the wireless medium. Moreover, due to asynchronous packet transmissions, there was often little opportunity for code resulting in small throughput gains. Coding opportunity could be increased by inducing small delays at intermediate nodes. However, this extra delay at intermediate nodes resulted in longer round-trip-times that adversely affect TCP throughput. They argued that a delay in the range of 1 ms to 2 ms could help to maximise TCP throughput. For the topologies considered in this paper, network coding improves TCP throughput by 10% to 85%.

III. Problem statement

Impact of network coding on TCP was studied by [12], authors in [12] argued that TCP function of the AIMD (Additive Increase/Multiplicative Decrease), had a significant impact on the benefits of network coding. TCP's congestion control mechanism continuously adapts TCP sending rate to network conditions and the current available capacity of the network. Specifically, the AIMD mechanism is incredibly sensitive to packet losses and translates them as signs of congestion. Furthermore, upon detecting a loss, TCP halves its sending rate by reducing its congestion window size to half. In the early simulation test, it was clear. There was AIMD impact on Xor coding network as COPE [4], the TCP's AIMD has not been designed to accept interaction of intermediate nodes role in coded networks, TCP is suitable for classical transmission paradigm. Furthermore, slashing TCP upon detection of congestion or lost packet can restrict the utilisation of medium and holding back any attempts to make use of unused resources, for example COPE can reduce traffic congestion on intermediate nodes by finding the right packet to encode and broadcast, and as a result bandwidth and network resources could be utilised for another transmission opportunity. That is not all; Wireless is situated on the principal of contention on the MAC layer as in IEEE 802.11, there are many packet losses due to wireless channel errors and contention on the medium. Again TCP reacts to all such packet losses by reducing its transmission rate to half, and that results in low throughput and less medium utilisation. Therefore, TCP throughput is fundamentally limited by the end-to-end loss probability rather than the available end-to-end capacity [12]. Hence, TCP mighty does not gain significant benefit from the increased capacity due to network coding process of encoding when it compares to UDP packets. TCP is not enjoying directly from network coding, it is rather seen as a packet damage factor, or it incurs delays in network overall end-to-end delay. Work in [15] used test topology considered the proximity of wireless nodes. Therefore contention on the wireless medium was likely high. High contention leads to high end-to-end packet loss probability which denies TCP from fully utilising the medium. TCP is not even able to utilise the channel capacity available to it without coding, and hence increasing channel capacity with coding does not significantly benefit TCP. Instead, TCP benefits from coding indirectly. It can be interpreted as following: coding reduces the number of transmissions which results in lower contention on the wireless medium than non-coding approach. The need to revisit the TCP design functionality could befit current TCP to enjoy the network coding ability to reduce contention and number of needed transmissions.

IV. System Model

A. TCP Tahoe

First simple TCP implementation was Tahoe, and it was based on the go-back-n model with cumulative positive acknowledgements and the expiration of the retransmission timer (RTO), that was required before the flow could be able to transmit any lost packets. TCP Tahoe facilitated with the slow-start, congestion avoidance, and fast recovery algorithms [8]. When a packet is lost, fast transmit is triggered to reduce waiting time of transmission instead of waiting for the retransmission timer to expire, if TCP rexmtthresh (usually three) duplicate ACKs are received, the sender infers a packet

loss and retransmits the lost packet. The sender now sets its ssthresh to half the current value of cwnd (maintained in bytes) and begins again in the slow-start mode with an initial window of 1. The slow start phase lasts till the cwnd reaches ssthresh and then congestion avoidance takes over. In this phase, the sender increases its cwnd linearly for every new ACK it receives by $cwnd = \frac{MSS \times MSS}{cwnd}$ (2).
Note that with TCP Tahoe the sender might retransmit packets which have been received correctly. Timeouts are used as the means of last resort to recover lost packets [6]. Mostly, the incident of three DupAcks considers the main reason behind low throughput, in our simulation, we came across, continues DupAcks incident which causes considerable delays, and slows the process of smooth transmission of TCP flows. Slow start and congestion avoidance mechanism restrain the overall throughput. The slow start carve plunges sharply to the congestion avoidance algorithm which increments one 1 MSS per Round Trip Time (RTT). As a result it causes more delays and degradation in throughput.

B. TCP New Reno

TCP New Reno improvements considered both the fast retransmit and fast recovery algorithms. The TCP sender can conjure from the arrival of duplicate acknowledgements, whether multiple losses in the same window of data would occur, and it can avoid taking a retransmit timeout or making multiple congestion window reductions due to such an event. The New Reno improvements apply to the fast recovery procedure that begins when three duplicate ACKs are received, and it ends when either a retransmission timeout occurs, or an ACK arrives, which that acknowledges all of the data that previously sent, besides it includes the data that was outstanding when the fast recovery procedure began [13].

Surprisingly in New Reno, neither full Ack nor Partial Ack affects the overall throughput, besides they did not incur any delays in packets or affect the slow start. The slow start could be increased to lift the restrains of permitted data on the medium. However, it can result in duplicate ACKs. Only congestion avoidance and DupAck are evidenced to cause a disturbance within New Reno, and they decrease total throughput in two or more hops. Congestion avoidance was counted to be the first calibre of 77% New Reno components. It was the most called function of the simulation time. The second called function was Duplicated Ack in fast recovery mode, it is unlike Tahoe where Duplicated Ack itself was the least called function, it is only counted for 2% of the whole simulation. In New Reno, Duplicated Ack goes to a quick remedy of fast retransmission to save time for waiting for the retransmission timer to expire. Duplicated Ack in new Reno was accounted to be 16% of simulation time.

C. TCP Westwood and TCP Westwood+

TCP Westwood [14] is considered a sender side modification based on TCP Reno protocol stack, which improved the performance congestion control (CWND) of TCP in both wired and wireless networks. Furthermore, TCP Westwood is designated on end-to-end bandwidth estimation to set congestion window and slow start threshold after a congestion occurrence, or after three duplicate acknowledgements or a timeout. The author in [18] noticed that low-pass filtering estimated the bandwidth for the rate of returning acknowledgement packets. Unlike TCP Reno, which hastily halves the congestion window after three duplicate ACKs, TCP Westwood strategically configures a slow start threshold and a congestion window by taking into account the bandwidth used at the time of previous congestion experience. The author in [14] claimed that TCP Westwood significantly increased throughput over wireless links and fairness compared to TCP Reno, and New Reno in wired networks [14]. TCP Westwood+ is the upgraded version of TCP Westwood, which is an end-to-end bandwidth estimation for setting control windows after congestion. The novelty of Westwood+ is its sole propriety of the available end-to-end bandwidth estimation algorithm. While TCP Westwood bandwidth estimation algorithm does not work well in the presence of reverse traffic due to ACK compression [14]. In our work, we do not use any ACK compression for reverse traffic; therefore, we use Westwood solely.

D. WeNC-TCP

Our proposed WeNC-TCP (Westwood based COPE Network Coding- TCP) is depicted in Fig.1. It has the framework of TCP Westwood. However, it is designated to accommodated Xor network coding such as COPE [4]. WeNC-TCP starts when sender begins sending its data packets to potential recipients, and receiving correspondent acknowledges data packets. One and halftime increase the congestion window (cwnd), and it is based on theoretical throughput gain of network coding in the classical two-hop network. If threshold exceeds the current cwnd (Slow Start), it enters congestion avoidance mode. As a result, cwnd assigns one segment size. It helps to rectify any congestion and reduce the injection of a new segment into the network. As in original Westwood implementation, the case of fast recovery mode of received new Ack, it sets cwnd equal to ssThresh. In our simulation, it is increased by $ssThresh \times \frac{1+\sqrt{5}}{2}$. It assists TCP sender to recover from three duplicated acknowledgement incident. As a result, it allows better utilisation of available resources. It breaks the rigid design of original Westwood that sets the cwnd to current bandwidth estimation, as in case of one or two delayed Ack, in our simulation, the case two delayed Ack is more likely to happen in different TCP variants. Therefore, it is increased by $ssThresh \times \frac{1+\sqrt{5}}{2}$.

On another hand, threshold is set by bandwidth estimation when duplicate Acked is detected and not in fast recovery mode, it is calculated based on Westwood estimation function that extracts information from recently received ACK [14], the following equation shows estimated bandwidth is multiplied by minimum Round Trip Time (RTT) $ssThresh = BW \times minRTT$ (3).

In case of three duplicated acknowledgement (DupAck) is received at sender side and DupAck is not in fast recovery mode, cwnd is calculated differently by taking estimated

threshold and multiple by $\frac{1+\sqrt{5}}{2}$ as follow : $cwnd = ssThresh \times \frac{1+\sqrt{5}}{2}$ (4). However, in the case of three duplicated acknowledgement and it is in fast recovery mode, the cwnd is multiplied by two segment size as follow: $cwnd = SegmentSize \times 2$ (5). At retransmission stage, the sender cwnd is set by one segment size for every successful transmitted of the lost packet, while multiplier function increases round Trip Time (RTT). The multiplier doubles estimated RTT for the next Transmission Time-Out Timer (RTO).

In our simulation we noticed that DupAck is more likely to have happened when Xor network coding is being used, it frequently happens in Tahoe, New Reno and Westwood. It results in degrading of throughput and causing RTO (Retransmissions Time Out) timer to expire. It causes TCP to inter re-transmission mode. As a result, RLP ratio is higher in Tahoe, New Reno and Westwood. What makes it difficult to tackle is the concept of interaction of intermediate nodes and intersession network coding. It is the inefficiency of both congestion avoidance and DupAck.

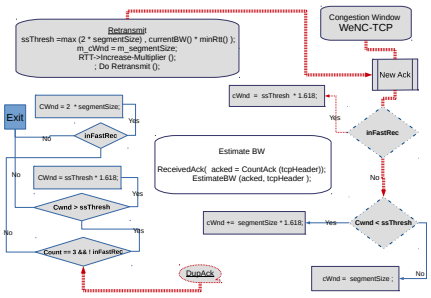

Fig. 1. Proposed WeCN-TCP Model.

V. Performance Metric and Simulation

In order to evaluate our algorithm either positive or negative impact on TCP performance, the following metrics must be considered:

1. Ratio of lost Packet: It is the ratio of the number of packets lost and the total number of packets sent on the medium.

$$RLP = \frac{The\ number\ of\ packets\ lost}{The\ total\ number\ of\ packets\ sent}$$

2. Throughput: It is the number of packets successfully received by destination with respect to time (bits/s)

$$T = \frac{Number\ of Received\ packets \times PacketSize \times 8}{Simulation\ time\ (Seconds)}$$

3. Delay: refers to the amount of time it takes a packet to be transmitted from source to destination.

4. Jitters: is defined to be a variation in the delay of received packets.

The Simulation is based on NS3 and chain topology is used. We test our algorithm against the increased number of hops. The size of TCP's queue at receiver and sender is limited to be 3276800 bytes. Furthermore, TCP layer queue does not have a maximum delay limit, and it follows the queue discipline of FIFO. The MAC layer sets to the default setting, for example, the maximum queue capacity is 10 packet with the maximum delay of 0.5 seconds. TCP Tahoe, New Reno and Westwood are set to original settings with no modification or alteration.

VI. Results and Discussion

In a Multi-hop scenario, all three TCP implementations were tested on a different number of hops. We noticed that all TCPs behaved similarly, Fig.3 depicted all tree TCP throughput. The only exception is TCP Westwood in the red line. It has the slightly less throughput. Moreover, the average throughput for all 13 hops was around 1.57 Mbps. Also, the maximum throughput for all hops was seen in TCP Tahoe with 1.64 Mbps. Whereas, TCP New Reno was the second highest with a throughput of 1.62 Mbps. Despite the mild differences in throughput, All TCP variants are identical to a degree with a smooth decline over the increased number of hops. However, the RLP for TCP Westwood over the increased number of hops was the highest, and it was around 17%. Westwood was not as the author in [14] claimed to be better than Reno in term of throughput and fairness over wireless links compared to TCP Reno. Moreover, the author claimed that TCP Westwood would choose better congestion window value based on bandwidth estimation after congestion occurred. It is clear it does not fit well when Xor network coding is applied.

However, Tahoe and new Reno are having at least 10 to 11 % loss packet ratio. Tahoe is considered premature less complicated for lacking the mechanism of fast recovery. Although, it has similar results as well as New Reno in term of throughput and RLP as in Fig.3 and Fig.4. Average delays of New Reno is seen as second highest after Westwood. It was around 0.73 second. While Westwood in red line was 0.74 second. Westwood's delay line has fluctuated rigidly at 8 and 11 hops as shown in Fig.5. Unlike jitters in Fig.6 Westwood is much smoother and it as same as other TCP variants. Moreover, jitters of TCP Tahoe, New Reno and Westwood are rising gradually as the number of hops increases, both Tahoe and new Reno recorded longest jitters when hop increases, The difference between Tahoe and New Reno is small in term of variance number of hops increase Fig.6 depicts. The smallest variance of jitters was recorded for Westwood with 0.133 scored for bidirectional flows. However, the maximum average jitters in our simulation were proved to for New Reno, and it was around 0.136 second. TCP Tahoe was second highest at 0.135 seconds.

Our solution WeNC-TCP has overtaken both Tahoe, and New Reno Fig.3 depicts, what makes WeNC-TCP superior is the ability to achieve better throughput when packet size is less than 500 bytes. In Fig.2, it is clear the throughput is slightly improved up to 15% in two hop network. However, WeNC-TCP can manage smaller packets as well as bigger packets size. WeNC-TCP can improve throughput up to 8% when packet size reaches 1500 bytes. In Multi-hop case,

WeNC-TCP has slightly better throughput over the increased number of hops as in Fig.3. Throughput can get up to 14% improvement of throughput. The line in black represents the throughput performance of WeNC-TCP among other tested TCP variants Fig.3 shown. Although throughput is not significantly enhanced, RLP has enhanced considerably. WeNC-TCP has the lowest RLP, which is much reliable than other TCPs shown in Fig.4. Again, in term of PLR, WeNC-TCP minimises the ratio of packet loss down to 95% when it compares to Tahoe, New Reno and Westwood. For instance, 89% RLP improvements are achieved against New Reno at the same time 87% achieved when compared to TCP Tahoe in two hops. Moreover, Fig.4 illustrates RLP vs increased number of hops for WeNC-TCP against Tahoe, New Reno and Westwood. The average RLP of WeNC-TCP reaches 2.0% loss ratio in two hops with increased packets size. In Multi-hop, the average improvement of WeNC-TCP's RLP was 72% overall hops, and the maximum RLP was less than 10%. Whereas RLP for Tahoe, New Reno and were a bit around 10 % or above, Westwood showed a much higher ratio of RLP in 8th and 11th hops, and such incident is depicted in Fig.4.

Delay of WeNC-TCP in black is the lowest among other TCPs as Fig.5 illustrates. WeNC-TCP has up to 70% improvement concerns delay, with average delay improvement of 38% when it compares against New Reno's delay over the increased number of hops. If WeNC-TCP puts against Tahoe and Westwood, WeNC-TCP can improve delay up to 72% with average delay improvement of 38%. Jitters of WeNC-TCP has matching behaviour as other TCP variants. It fluctuates (in Blackline) at 6th hop to 14 hops. WeNC-TCP has relatively average jitters of 0.127 second which it is similar to Westwood (0.133 seconds). However, both Tahoe and New Reno were recording the highest average jitters which was around 0.136 seconds. Fig.6 shows jitters of Tahoe, New Reno, Westwood and WeNC-TCP, jitters for all TCP variants have levelled up after 9th hop as well as WeNC-TCP. To summarise all WeNC-TCP attained improvements, the table I represents the number of hops and achieved improvements (%) in terms of RLP, delay, jitters and throughput. WeNC-TCP addressed the issue of DupAck, and reduced its impact on TCP flows, and Fig.8 shows the total of DupAck incident over time, it is clear it is a short period of fewer than 50 indents of whole simulation time in less than 3.1 to 3.35 a second of whole 80 seconds. In Fig.7 shows the WeNC-TCP congestion window, the cwnd has much smoother and regulated behaviour, it can reach 65000 (Maximum), and later it got to the range of 10000 and stayed around it with our rigid fluctuations.

VII. Conclusion

The current TCP functionality causes drain in the network throughput when network coding is being applied. The congestion avoidance algorithm affects the throughput badly. It cripples the ability of the sender to injects packets based on its mechanism of preventing congestion, wherein wireless could be misinterpreted due to link quality. However, the network Xor coding process is crippled by restricting the number of available native packets for finding the matching pair. In our work, we try to gap the difference between

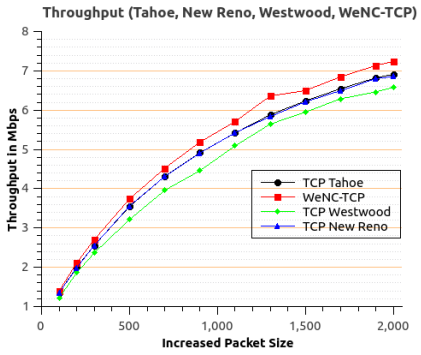

Fig. 2. Throughput of WeNC-TCP in Two hops.

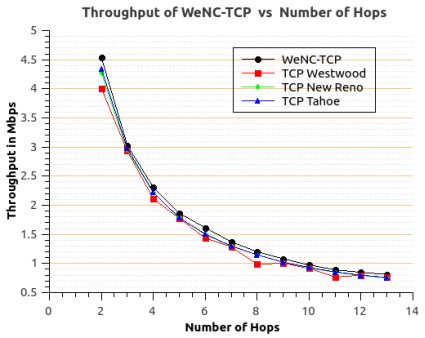

Fig. 3. Number Hops Vs WeNC-TCP Throughput.

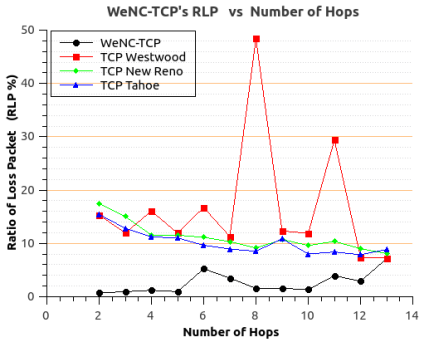

Fig. 4. Number Hops Vs WeNC-TCP RLP.

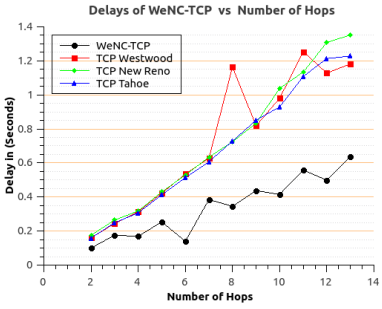

Fig. 5. Delays of WeNC-TCP Vs Number of Hops.

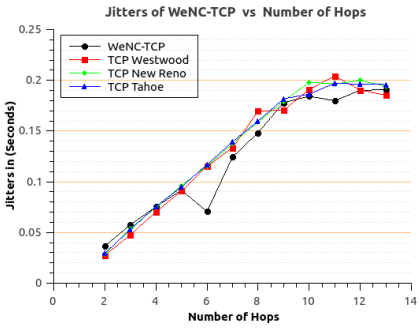

Fig. 6. jitters of WeNC-TCP Vs Number of Hops.

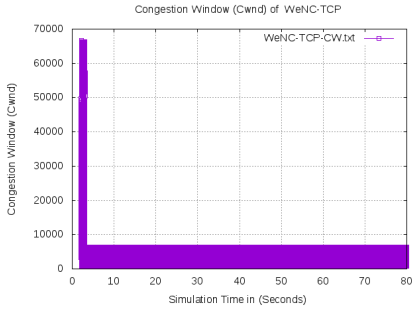

Fig. 7. Congestion Window in TCP WeNC-TC.

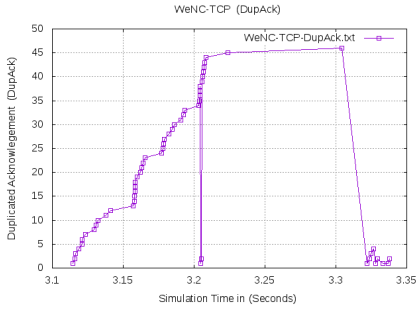

Fig. 8. DupAck in WeNC-TCP.

two different aspects TCP reliability and Network coding process. Principality's of current TCP cannot accept the role of intermediate node which it finds the best match of packet pair. Our WeNC-TCP achieves slightly better throughput than other TCP variants, although, the best of WeNC-TCP can be seen in RLP and delay in two or more hops. Table I has a summary of all achieved improvements over the increased number of hops. The need for better interaction between two layers such Transport and MAC layer becomes a necessity when network coding is applied in the wireless networks. WeNC-TCP is still the first version to accommodate the need for network coding at the transport layer. It is far from perfection. We could anticipate that the MAC layer could add a significant contribution to TCP flows when the network experience different circumstances of such packet losses, limited bandwidth or limited access to the medium in high contention scenario. Our future work will focus on the interaction between TCP and MAC layer using Xor network coding such as COPE[4].

Improvement in				
Number of Hops	RLP %	Delay	Jitters	Throughput %
2	95	37	-24	5
3	93	29	-9.	2
4	89	45	0.41	4
5	91	39	3	4
6	46	73	40	7
7	61	37	11	6
8	82	53	7	5
9	86	49	2	6
10	83	55	1	5
11	53	50	9	4
12	63	59	3	6
13	19	48	3	7

TABLE I

IMPROVEMENT FOR INCREASED HOPS.

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, *Network Information Flow* , IEEE Trans. Inform. Theory, vol. 46, pp. 12041216, Jul. 2000.

[2] D. Katabi, S. Katti, W. Hu, H. Rahul, and M. Medard, *On Practical Network Coding for Wireless Environments,*, in Proc. Intl Zurich Seminar on Communications, pp. 8485, 2006.

[3] J. Jin and B. Li, *Adaptive Random Network Coding in WiMAX,*, in Proc. IEEE ICC08, pp. 25762580, May 2008.

[4] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, *XORs in the Air: Practical Wireless Network Coding,* , IEEE/ACM Trans. Netw., vol. 16, pp. 497510, June 2008.

[5] J. K. Sundararajan, D. Shah, M. Medard, S. Jakubczak, M. Mitzenmacher and J. Barros, *"Network Coding Meets TCP: Theory and Implementation,"*, in Proceedings of the IEEE, vol. 99, no. 3, pp. 490-512, March 2011.

[6] B. Sikdar, S. Kalyanaraman and K. S. Vastola, *"Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno, and SACK,"*, in IEEE/ACM Transactions on Networking, vol. 11, no. 6, pp. 959-971, Dec. 2003.

[7] A. Campo and A. Grant, *Robustness of Random Network Coding to Interfering Sources,*, in Proc. 7th Australian Communications Theory Workshop, pp. 120124, Feb. 2006.

[8] T. Nage, F. R. Yu and M. St-Hilaire, *"TCP-Aware Network Coding with Opportunistic Scheduling in Wireless Mobile Ad Hoc Networks,"* , 2010 7th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, 2010, pp. 1-5.

[9] Swastik Brahma and Mainak Chatterjee, *Congestion control and fairness in wireless sensor networks*, Pervasive Computing and Communications Workshops (PERCOM), 2010 8th IEEE International Conference, March 29 2010-April 2 2010, Pp. 413-418.

[10] Puneet Mehra, Avideh Zakhor, Christophe De Vleeschouwer, *Receiver-Driven Bandwidth Sharing for TCP* , INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications., San Francisco, CA, March 30 2003-April 3 2003.

[11] Hamamreh, Rushdi A., and Mohammed J. Bawatna. *"Protocol for Dynamic Avoiding End-to-End Congestion in MANETs."*, Journal of Wireless Networking and Communications 4.3 (2014): 67-75.

[12] Y. Huang, M. Ghaderi, D. Towsley and W. Gong, *"TCP Performance in Coded Wireless Mesh Networks,"*, 2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, San Francisco, CA, 2008, pp. 179-187.

[13] T. Henderson, A. Gurtov, and Y. Nishida , *The NewReno Modification to TCP's Fast Recovery Algorithm,* , IETF, RFC 6582, April 2012. [Online]. Available: http://https://tools.ietf.org/html/rfc6582.

[14] M. Gerla, M. Y. Sanadidi, Ren Wang, A. Zanella, C. Casetti and S. Mascolo, *"TCP Westwood: congestion window control using bandwidth estimation,"*, Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE, San Antonio, TX, 2001, pp. 1698-1702 vol.3.