Title

Optimized Network Coding With Real-Time Loss Prediction for Hybrid 5G Networks

Authors

Srinivasan, Ramesh
Garcia-Luna-Aceves, J.J.

Publication Date

2022-10-01

Peer reviewed

Optimized Network Coding with Real-Time Loss Prediction for Hybrid 5G Networks

Ramesh Srinivasan[1][0000−0002−2680−7254]
J. J. Garcia-Luna-Aceves[1][0000−0001−9914−6031]

University of California, Santa Cruz, CA 95064, USA

Abstract. Two novel mechanisms are introduced to take advantage of network coding in TCP (Transmission Control Protocol), namely: TCP with Network-Coded Window Transformation (TCP-NWT) and TCP-NWT augmented with dynamic loss prediction, called Predictive-Network-Coding (TCP-PNC). TCP-NWT uses network coding to handle packet losses without retransmissions. TCP-PNC predicts the expected loss-ratio on an ongoing basis during the course of a TCP-NWT session, which in turn changes the number of network coded packets that are transmitted. These mechanisms result in a more efficient use of network-coded packet transmissions in TCP. Simulation results indicate a throughput increase of more than 22% compared to TCP in scenarios involving dynamic changes in loss ratios in the midst of a TCP session.

Keywords: TCP · Real-time · Network-Coding.

1 Introduction

Many of the current network-coding enhancements used in TCP use a predetermined loss-ratio for computing the amount of redundancy to be introduced with additional network-coded packets, which is fixed for the duration of a TCP session. This is a significant limitation, because of two key reasons. First, the proliferation of different types of mobile end devices and ubiquitous wireless last-mile access has resulted in dynamic transient fluctuations of packet-loss ratios in the midst of an ongoing TCP session that need not reflect any real network congestion. This renders the use of a predetermined loss-ratio throughout a TCP session ineffective. Second, end user applications require continuous availability of services, service providers need to attain the most efficient use of the available bandwidth over wired or wireless links, and more and more end users are mobile. Some applications need real-time reliable data delivery with predictable upper-bounds on data delivery. Hence, the static loss-ratio approach used in prior enhancements of TCP based on network coding must be revisited to account for the fact that a given TCP session may have varying loss-ratios during the course of its session. Section 2 provides a survey of related work that reveals that prior TCP variants based on network coding have relied on a static loss-ratio. The closest approach to our work is the Vegas Loss Predictor [10], which is implemented at the Network Coding layer (between Layer 4 and Layer 3) [13] to

know when the network experiences congestion; however, the RTT (round trip time) values used do not factor in the additional time incurred due to potential link-layer retransmissions in last-mile wireless-links, which we try to incorporate in our work.

This paper introduces a new approach to detect the network health in a network-coding enabled TCP session and then predicts the expected loss-ratio and adapts to it by generating network-coded data to proactively compensate for the expected data loss. The proposed approach is particularly attractive for deployments of 5G networks and beyond, because it easily accommodates the use of heterogeneous transmission media, mobile end-nodes and comes very close to guaranteed data delivery in real-time with most optimal usage of network resources.

Section 3 describes TCP with Network-Coded Window Transformation (TCP-NWT) and Section 4 describes TCP-NWT with Predictive-Loss-Ratio, namely Predictive Network Coding (TCP-PNC). TCP-NWT proactively addresses packet losses without re-transmissions, while ensuring that all TCP session metrics are suitably transformed and passed back to the original TCP stack. This is accomplished by transforming the original TCP sliding window into another sliding window comprising of Network Coded data segments. TCP-PNC improves on TCP-NWT by dynamically predicting the expected loss-ratio on an ongoing basis during the course of a TCP session. This ensures that the optimal amount of network coded packets are transmitted.

Section 5 describes the results of simulations conducted with TCP-NWT and with other deployed TCP versions including TCP-Cubic and Section 6 outlines and compares the results observed. Section 7 concludes the paper.

2 Related Work

The use of network coding (NC) in TCP has been an area of active research. A comparative study of the actual approaches can be found in [8]. We only outline some of the most salient aspects and issues with these approaches.

TCP/NC [13] uses a new interpretation of acknowledgments (ACK), the sink acknowledges every linear combination of packets that reveals one unit of new information, even if it does not reveal an original packet immediately. This scheme has the property that packet losses are essentially masked from the congestion control algorithm. Therefore, this algorithm reacts to packet drops in a smooth manner, resulting in an effective approach for congestion control over networks involving lossy links. However, packet losses due to congestion are also masked in this approach and therefore effective flow-control is inhibited.

A redundancy adaptation scheme for network coding in TCP Vegas [10] uses loss predictor to decide whether the network is congested based on rate estimators [2], [7], [14]. The Vegas Loss Predictor is implemented at the Network Coding layer, between the network and transport layers, to know when the network experiences congestion and to adjust accordingly as in [2]. However, the RTT values used do not factor in the additional time incurred due to potential

link-layer retransmissions in last-mile wireless-links. The effectiveness of NC has been analyzed by multiple authors [13], [5], [1]. The results indicate that NC does not provide big performance gains if it used below the transport layer in conjunction with a standard TCP implementation, as messages need to be delayed in a buffer to be able to encode them. The RTT is increased at each hop, and TCP interprets the RTT increases as a sign of congestion and reduces the transmission rate, which prevents the effective use of the transmission medium.

In summary, TCP has been augmented with a modular NC sub-layer to facilitate quick and easy adoption. However, that has resulted in many of the core intrinsic TCP session parameters and metrics like RTT and packet throughput not being accurately captured to reflect the exact status of the network along with introduction of additional delays. In this work, we ensure these metrics are accurately captured and relayed back to the transport layer and also optimal amount of network coded segments are generated with minimal additional introduction of delays.

3 TCP-NWT

TCP-NWT is a TCP congestion window transformation protocol which transforms the original TCP sliding congestion window with data segments into a new TCP congestion window comprising of network coded data segments. On the receiver side, on detecting the receipt of a TCP-NWT network coded segment, TCP-NWT window transformation protocol transforms the TCP-NWT receiver window into the corresponding original TCP receiver window comprising of the original TCP data segments generated by decoding the received group of coded TCP segments. Additionally, we propose a novel mechanism for processing of the acknowledgment packets so that the path metrics like RTT measurements by the TCP-NWT network coded segments are accurately relayed back to the original TCP. The design of TCP-NWT assumes a fixed loss-ratio and its specification consists of: (a) The TCP packet header augmentation needed to support network coding, (b) the available choice of coefficients values supported by our design and (c) the enumeration of the permitted group sizes.

We have taken an example to illustrate how TCP-NWT transformation works on group size of 1, which provides sufficient insight and clarity as to how it would work on larger group sizes. Encoding, decoding as well as processing of acknowledgments including relaying of the observed network health through RTT back to original TCP window are elaborated in great detail. A key consideration in our approach has been to keep the computation overhead of generating the network coded segments for transmission at the sender side as well as the subsequent decoding at the receiver side to bare minimal, as we are targeting mobile end-nodes which have significant computing, memory and in many cases power constraints.

We augment the TCP header with a new Boolean field indicating if its a header of an TCP-NWT segment. The group-number corresponding to this coded segment as well as the RLC coefficients used to generate this, namely CE1, CE2 and so on, also need to be included in the header. Figure 1 illustrates

the new proposed TCP-NWT packet header. The first 20 bytes of the TCP header are always used in TCP-NWT. The options field is of variable size and it starts from the 6th row and can go up to 40 bytes. We use the TCP Option Kind number 25 [6], [11]. The newly introduced TCP options field entries to support NWT are: (a) kind equal to 25 (8 bits); (b) length in bytes (8 bits); (c) network_coded_:1 (8 bits); (d) group_size equal to 1, 2, 4 or 8 (8 bits); (e) group_id: Grp Seq Num (32 bits); and (f) CEi equal to 1, 2, 4, 8, 16 /or 32 with $i = 1$ to 32 (6 unique values can be represented by 3 bits, however we have allocated 4 bits for each CE).

The group sizes permitted are 1 or 2 or 4 or 8. The permitted coefficients are one of six values namely 1, 2, 4, 8, 16 or 32, which ensures that multiplication with these coefficients is simply a bit-shifting operation and thus incurs minimal computation overhead. The group size indicates the number of segments, from the original non-coded data segments, which are combined (added) together after being multiplied by one of the random linear coefficients listed below, to generate the required number of coded segments. In the example below in Figure 2, a fixed loss-ratio is assumed and the entire set of segments 4 in the initial group from Original TCP sliding window are coded using random coefficients to generate 5 coded segments for the 15% loss-ratio scenario. These are placed in the new TCP-NWT window.

Table 1. Definitions

Abbreviations	**Definitions**
RLC	Random Linear Coefficients
Orig-Grp-size	Number of segments in a group in Original TCP Window
grp_sz	Number of segments in current group in Original TCP Window
group_id	ID corresponding to an entire group used to generate coded segments
Grp Seq Num	the group_id from where coded segment got generated
RLC grp packets	Random Linear Coded Pkts of a group
Coded-Grp-size	Number of coded segments generated from the group in Original TCP Window
WLR	Worst Case Loss Ratio
D_i	$Datagram_i$ in Original TCP window
CD_i	$Coded_Datagram_i$ in TCP-NWT window
CE1, CE2..CE16	Random Linear Coefficient (RLC)1, RLC2... RLC16
SRTT	Smoothed Round Trip Time
RTTVAR	Round Trip Time Variation
$SRTT_{8.1}$	Smoothed Round Trip Time for $CD_{8.1}$ in TCP-NWT
$RTTVAR_{8.1}$	Round Trip Time Variation for $CD_{8.1}$ in TCP-NWT
$SRTT_8$	Smoothed Round Trip Time for Group 8 in original TCP window
$RTTVAR_8$	Round Trip Time Variation for Group 8 in original TCP window
RTO	Round Trip Timeout
R	Initial Round Trip Time Measurement
R'	Next Round Trip Time Measurement

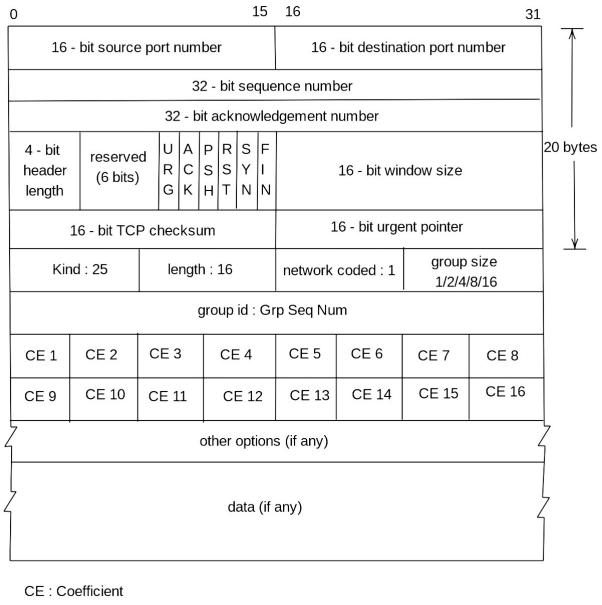

Fig. 1. TCP-NWT Packet header

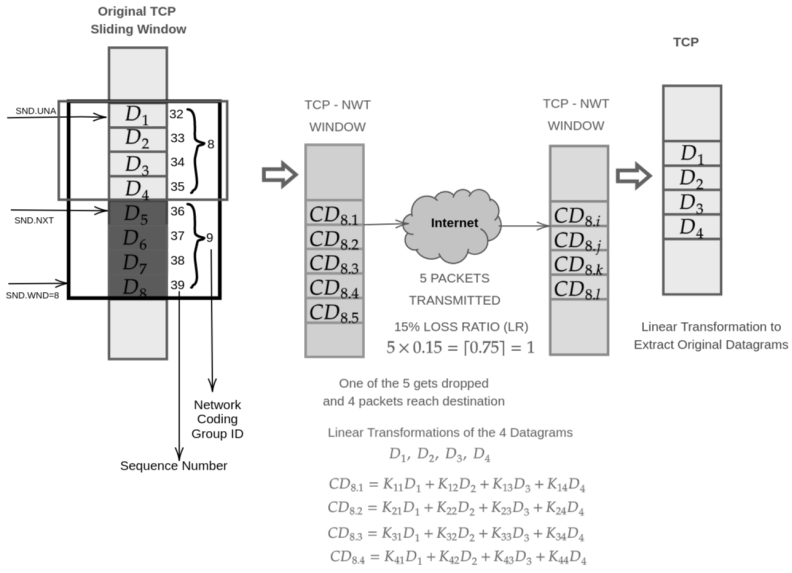

Fig. 2. Network Coding

TCP-NWT Protocol Description

Each generated RLC segment has the following additional fields:

1. Orig-Grp-size: (≤ 16) permitted values in our design: 1 or 2 or 4 or 8.
2. List of RLC coefficents: the number of these coefficients is exactly equal to the Orig-Grp-size, listed above.
3. Unique Group ID: Group Sequence number for each group (similar to the sequence number for individual packets) and is common for all members of a group. The random linear codes used for generating each of the new coded packets are always a unique tuple of dimension Orig-Grp-size (max possible is 8)
4. Coded-Grp-size = Orig-Grp-size/(1 - WLR)

Group size 1 Figure 3 depicts the scenario where a group contains just a single TCP segment.

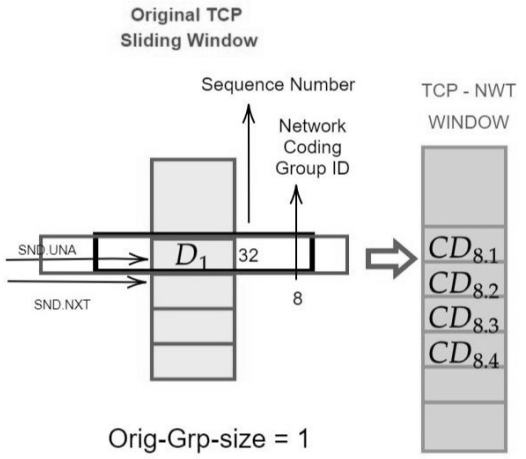

Fig. 3. Group Size 1

Sending Side When there is a single segment in the sliding window and there are no other data/segments queuing in from higher layers for this TCP session, then group size (Orig-Grp-size) is set to 1. Depending on the loss-ratio, the number of coded segments generated could range from 2 to possibly 4. In the above example, the network coding group-id is 8 and the number of coded segments generated has been chosen to 4.

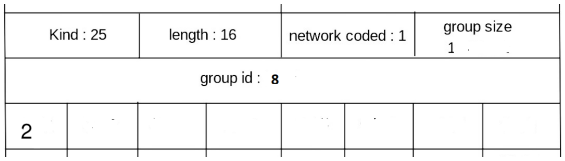

Fig. 4. Group Size 1: Coded Datagram 8.1's TCP Header

The relevant portions of the modified TCP header for network coded segment using a coefficient of 2 is depictedin Figure 5.

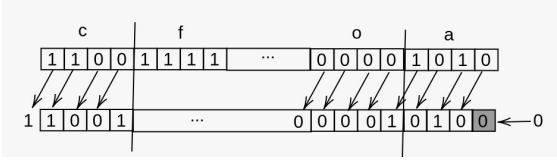

Fig. 5. Group Size 2: CE1-2 Coded Datagram 8.1's Computation by bit shifting

As can be seen a simple left shift of all the contents by 1 bit results in generation of the coded data. Similarly simple left shifts of all the contents by 2/3/4 bits results in generation of the corresponding coded data for coefficients of 4/8/16.

Receiving Side Group Size 1 The figure 6 depicts the receive mechanism when there is a single segment.

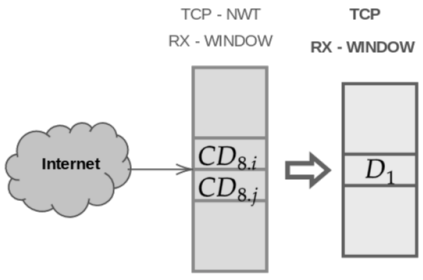

Fig. 6. Group Size 1 RX

Receipt of any one coded segment suffices to recompute the original segment, by a simple bit shift operation to the right according the value of the CE1. If

CE1=2, right shift by 1 bit, if CE1=4, right shift by 2 bits, if CE1=8, right shift by 3 bits and if CE1=16, right shift by 4 bits to generate original Segment D1.

Acknowledgement for Group Size 1 The receipt of an ACK for any one coded segment in a group of size of one confirms receipt of the data for that group. One of the contributions of this work is ensuring that the health of the network is captured accurately and relayed back as-is to the higher layer by the combined TCP stack. To compute the current RTO, a TCP sender maintains two state variables, SRTT and RTTVAR. We compute the RTO at the end of receipt of acknowledgement for each of the four coded segments transmitted using the exact method outlined in RFC-2988 [9]. When the first RTT measurement R is made, the host updates SRRT, SRRT and RTO as follows:
SRTT $\leftarrow$ R; RTTVAR $\leftarrow$ R/2
RTO $\leftarrow$ SRTT + max (G, K*RTTVAR); where K = 4

For each subsequent RTT measurement R' in a given NC group, the sender updates RTTVAR and SRTT for TCP-NWT window, as follows till measurements for all coded segments are completed.

$RTTVAR_{8.1} \leftarrow$(1-beta)*$RTTVAR_8$ + beta* | SRTT - $R_{8.1}$' |
$SRTT_{8.1} \leftarrow$ (1 - alpha) * $SRTT_8$ + alpha * $R_{8.1}$'
$RTTVAR_{8.2} \leftarrow$(1 - beta)*$RTTVAR_{8.1}$ +
beta * | $SRTT_{8.1}$ - $R_{8.2}$' |
$SRTT_{8.2} \leftarrow$ (1 - alpha) * $SRTT_{8.1}$ + alpha * $R_{8.2}$'
$RTTVAR_{8.3} \leftarrow$ (1 - beta) * $RTTVAR_{8.2}$ + beta * | $SRTT_{8.2}$ - $R_{8.3}$' |
$SRTT_{8.3} \leftarrow$ (1 - alpha) * $SRTT_{8.2}$ + alpha * $R_{8.3}$'
$RTTVAR_{8.4} \leftarrow$ (1 - beta) * $RTTVAR_{8.3}$ + beta * | $SRTT_{8.3}$ - $R_{8.4}$' |
$SRTT_{8.4} \leftarrow$ (1 - alpha) * $SRTT_{8.3}$ + alpha * $R_{8.4}$'
$RTTVAR_9 \leftarrow RTTVAR_{8.4}$
$SRTT_9 \leftarrow SRTT_{8.4}$

The RTTVAR and SRRT corresponding to $CD_{8.4}$ from TCP-NWT window are then assigned to the updated RTTVAR and SRRT corresponding to completion of successful transmission of D1 and receipt of ACK. The computation mechanism for other group sizes on both sending and receiving side are similar to that for group size 1.

Algorithm – TCP-NWT We state the algorithm for encoding a group of segments and the algorithm for decoding a group of segments below. Once the receiver has received sufficient number of coded segments for a group, equal to the size of the group, the decoding steps are initiated. We use the Gaussian-elimination [4] procedure for solving a system of linear equations to decode and arrive at the original data sent.

Algorithm 1 encode(group)

Determine the group (group_id) of the set of packets to be encoded.
Size of the group, grp_sz;
Initialize the group_seq_num for each individual packet within the group to the group_id of this group.
Based on LR (Loss_Ratio), determine the number of encoded packets to be generated: numEncoded
for i = 1; i ≤ numEncoded; i ++ **do**
    Determine the unique set of NC Coefficients Tuples: CE[1], CE[2], ⋯, CE[numEncoded]
    /* (based on the group_id_ and group_seq_num for each encoded packet to be generated.) */
    Clear RLC_PAC[i];
    Compute the contents of the coded packet.
    for j =1 ; j ≤ grp_sz; j++ **do**
        RLC_PAC[i] = RLC_PAC[i] + CE[i][j] × PAC[j]
    end for
    Populate the packet hdr of RLC_PAC[i] with the the NC Coefficients used to generate it;
    Upload RLC_PAC[i] the newly generated coded packet into new TCP_NWT window.
end for

Algorithm 2 decode(group)

while TCP Session is still active **do**
    Wait for the receipt of a packet. if times out waiting, quit;
    Determine the group (group_id_) of the received segment.
    Determine the group Size of the group, grp_sz;
    Determine if there is already a sink created to gather all segments of this group.
    if not, create a new sink for this group and initialize grp_rcv_cnt, group receive count to 1;
    Check if grp_sz for this group equals grp_rcv_cnt for this sink
    if yes pass the set of packets to the GaussianElimination function, which will return the original segments of this group.
end while

4 TCP-PNC

Based on the current dynamically estimated loss-ratio of the network at a given point in time and using a new enhanced approach based off [15], the number of network coded data packets (n) to be generated from an initial dataset (m) of data packets is computed. We evaluate the dynamic loss ratio as indicated below:

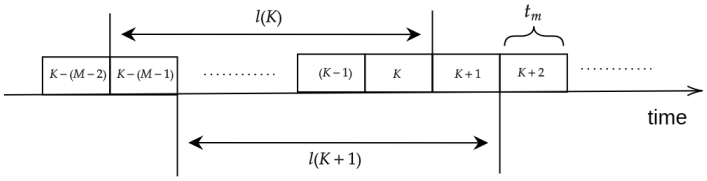

Fig. 7. Dynamic Loss-Ratio Prediction

Loss-Ratio for different range of the time periods "M" starting from 2 to about 32 is computed in every measurement period tm . Let $l_M(k)$ be the packet loss ratio of the k-th measurement, which is calculated as the number of dropped packets over the total number of packets arrived during the latest M periods (see equation 1); where $N_d(k)$ is the number of packets dropped in the k-th measurement period, and $N_a(k)$ is the number of packets arrived in the k-th measurement period.

$$l_M(k) = \frac{\left(\sum_{i=0}^{M-1} N_d(k-i)\right)}{\left(\sum_{i=0}^{M-1} N_a(k-i)\right)}, \quad M = 2, 3, 4, \cdots, 32 \tag{1}$$

In real scenario M can be any value ≥ 0:

$$M = 0,\ LR(k) = l_0(k) \tag{2}$$

$$M = 1,\ LR(k) = \frac{l_0(k) + 2^{-1} l_1(k)}{2^0 + 2^{-1}} \tag{3}$$

$$M = 2,\ LR(k) = \frac{l_0(k) + 2^{-1} l_1(k) + 2^{-2} l_2(k)}{2^0 + 2^{-1} + 2^{-2}} \tag{4}$$

$$LR(k) = \frac{\begin{array}{c} 2^0 \times l_0(k) + 2^{-1} \times l_1(k) + 2^{-2} \times l_2(k) + \cdots \\ +2^{-n} \times l_n(k) \end{array}}{2^0 + 2^{-1} + 2^{-2} + \cdots + 2^{-n}} \tag{5}$$

4.1 TCP-PNC Predictive Network Coding - Protocol Description by Example

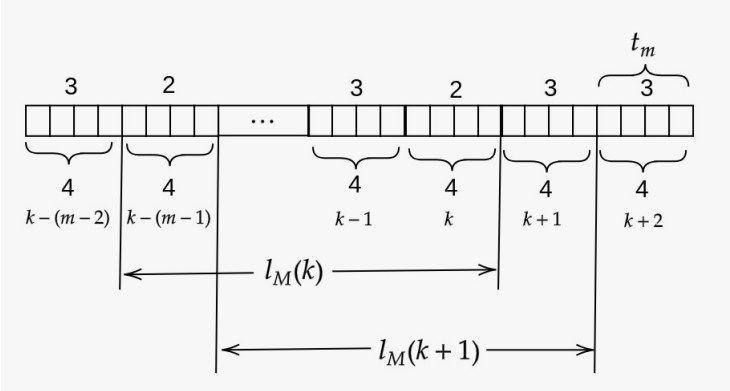

Fig. 8. An Example of Dynamic Loss-Ratio Prediction

We are taking an actual example in order to succinctly illustrate our proposed mechanism for dynamically arriving at the predicted loss at the next upcoming time interval. Value of M determines the number of time periods over which the loss ratio is computed. We are proposing here of assigning a weight of 1 for loss ratio $l_0(k)$, 2^{-1} for $l_1(k)$, 2^{-2} for $l_2(k)$ and so on, to ensure the data comprising just the immediate past is given a higher importance compared to the data corresponding to a slightly larger duration from the past. In the above example, we have taken the actual data which indicates that in the k-th measurement, out of 4 packets sent, 2 are successfully received and acknowledged. In the (k+1)th measurement, out of 4 packets sent, 3 are successfully received and acknowledged.

$$M = 0; \quad l_0(k-1) = \frac{3}{4} = 0.75; \quad LR(k-1) = \frac{1 * l_0(k-1)}{1} = 0.75 \tag{6}$$

$$M = 0; \quad l_0(k) = \frac{2}{4} = 0.5 \tag{7}$$

$$M = 1; \quad l_1(k) = \frac{3+2}{4+4} = \frac{5}{8} = 0.625 \tag{8}$$

$$LR(k) = \frac{1 * l_0(k) + (1/2) * l_1(k)}{1 + 1/2} = 0.54 \tag{9}$$

$$M = 0; \quad l_0(k+1) = \frac{3}{4} = 0.75 \tag{10}$$

$$M = 1; \quad l_1(k+1) = \frac{3+2}{4+4} = \frac{5}{8} = 0.625 \tag{11}$$

$$M = 2; \quad l_2(k+1) = \frac{3+2+3}{4+4+4} = \frac{8}{12} = 0.67 \tag{12}$$

$$LR(k+1) = \frac{1 * l_0(k+1) + (1/2) * l_1(k+1) + (1/4) * l_2(k+1)}{1 + 1/2 + 1/4} = 0.70 \tag{13}$$

Expected Dynamic Loss Ratio: Using $LR(k-1)$,LR(k), and $LR(k+1)$ we try to predict the PLR Predicted loss Ratio at the next three time intervals PLR(k+2), PLR(K+3), PLR(k+4) using following simple mechanisms. We do a linear extrapolation of the Observed loss ratio values at (k) and (k+1) to arrive at PLR(k+2). LR(K) is 0.54 and LR(K+1) is 0.70 and therefore initial estimate for PLR(k+2) is 0.90. However since in our example we are sending 4 segments in a timeslot, the actual possible values for $l_0(k+2)$ are 0, 0.25, 0.5, 0.75 and 1. Since our initial estimate of 0.90 is between 0.75 and 1, we would take the lower of the two namely 0.75 as the PLR(k+2). Similarly taking the values of OLR(K+1) and PLR(k+2) and doing a similar linear extrapolation we estimate PLR(K+3), which in our example turns out to be 0.75. Similarly taking PLR(k+2) and PLR(K+3) we estimate PLR(k+4), which also turns out to be 0.75. Next we try to predict the Worst case loss-ratio by taking the minimum of the observed loss ratio in the last two measurement periods and the predicted loss ratio in the upcoming two measurement periods:

MIN($LR(k), LR(k+1), PLR(k=2)PLR(K=3)PLR(k=4)$), namely MIN(0.54, 0.70, 0.75, 0.75, 0.75), which is 0.54. This is closest to 0.5, which would be the worst case loss-ratio in the above example. For a given session, at every 2 secs interval the Observed Loss Ratio (OLR) is computed and saved in a LossRatioTable for last hour (array size is 3600/2 = 1800). Using the past saved values of the observed loss ratio – along with currently observed loss ratio extrapolation of the gradient/trend and prediction of the PLR (Predicted Loss Ratio) values for the next 3 time periods is done. Based on this trend, the MIN (OLR(t0-4), OLR(t0-2), PLR(t0), PLR(t0+2), PLR(t0+4)) is chosen as the WLR(t0) potential Worst-case Loss Ratio scenario to be addressed while deriving the number of RLC (Random Linear Coded) TCP datagrams. Coded_Grp_size = Orig_Grp_size/(WLR).

5 Testing and Simulation

We evaluated the performance of TCP-WSC using discrete-event simulation. The NS-2 simulator [12] was used. NS-2 [12] provides substantial support for simulation of TCP, Routing, and Multicast Protocols over wired and wireless (local and satellite) networks. The TCP implementation was modified to support the new proposed protocols.

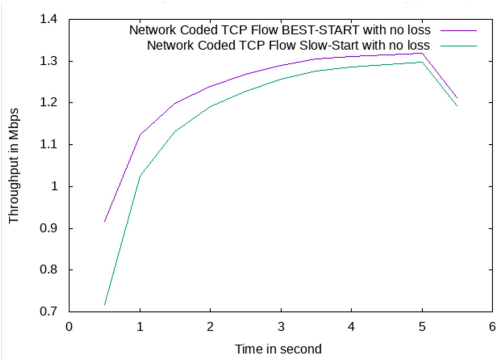

Fig. 9. TCP-NWT with No-Loss

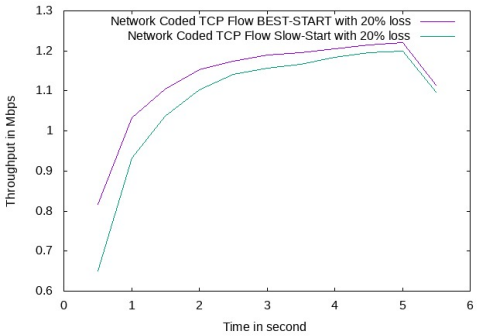

Fig. 10. TCP-NWT with 20%Loss

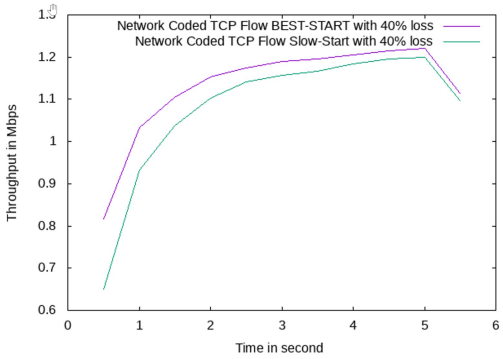

Fig. 11. TCP-NWT with 40%Loss

Numerous scenarios and options were tried out to truly validate the gains and benefits of the proposed approach here.

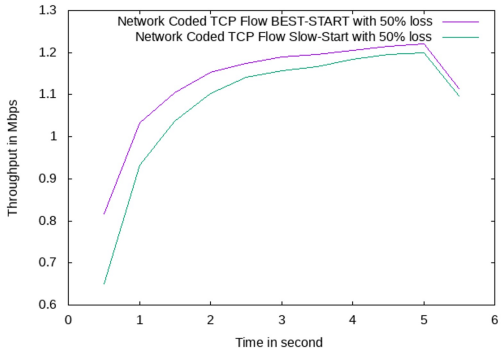

Fig. 12. TCP-NWT with 50%Loss

As very succinctly evident in the results of the simulation, the performance has been maintained at the same level despite varying levels of errors, all the way from 20% loss in figure 10, 40% loss in figure 11 and finally even with 50% loss as seen in the figure 12. Comparing these with the loss-less scenario in figure 9, clearly shows we can guarantee performance and throughput despite level of errors/losses with one big CAVEAT to remember, namely: these errors are ONLY due to wireless link-layer errors and NOT due to a true congestion per-se in the network. The above results were with TCP-NWT Only, without the predictive dynamic Loss-ratio incorporated.

The Simulation Scenario with TCP-PNC, which comprises TCP-NWT and additionally incorporates the Predictive loss Ratio.

This section describes simulations from 4 scenarios – 2 each with

a. standard TCP ns-2 [12] new Reno
   (i) Using a wireless topology with an almost lossless wireless link
   (ii) Using the same wireless topology with a substantial lossy wireless link at both the wireless end-nodes
b. standard TCP ns-2 [12] new Reno implementation modified with our proposed enhancements for networks with wireless end-nodes.
   (i) Using the same wireless topology with an almost lossless wireless link
   (ii) Using the same wireless topology with a substantial lossy wireless link (10% and subsequently 20%)

6 Results

There was an improvement in overall throughput observed with the new implementation – especially as transmission errors (link-layer losses) increase. Comparative results with TCP Cubic as well as TCP newReno [3] show that our Predictive Network Coding TCP-PNC provides a significantly higher throughput of about 22% . The results clearly demonstrate that dynamic adjustment of

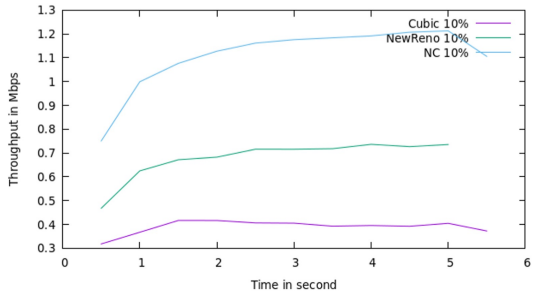

Fig. 13. Cubic vs New Reno vs TCP-PNC - 10% loss

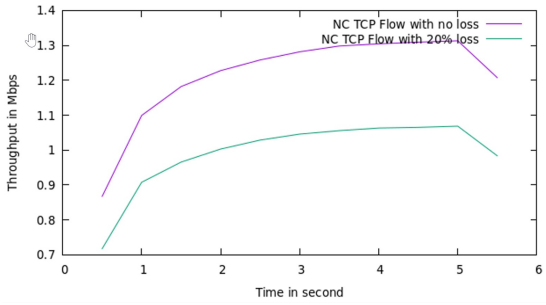

Fig. 14. TCP-PNC - Throughput Comparison with no loss vs 20% loss

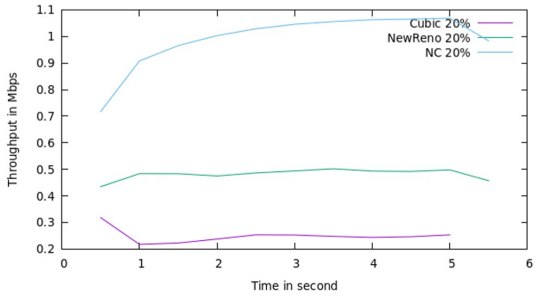

Fig. 15. Cubic vs New Reno vs TCP-PNC- 20% loss

amount of additional network coded segments being generated based on accurate prediction of the loss-ratio results in a much more optimal effective usage of the network resources as well as ensuring minimizing retransmission for lost segments, thus significantly improving throughput.

7 Conclusion and Future Work

The prediction of the loss ratio proposed in this paper constitutes a solution based on rudimentary machine learning. This is a nascent area with the potential for much more innovations based on proactive response based on machine-learning techniques, and there could be many more ways to predict the loss-ratio more accurately. Saving past TCP sessions metrics and parameters is another promising way to predict the current expected network behaviour for the same destinations.

References

1. Alferaidi, K., Piechocki, R.: Tcp-mac cross layer integration for xor network coding. In: Science and Information Conference. pp. 860–875. Springer (2018)
2. Brakmo, L.S., Peterson, L.L.: Tcp vegas: End to end congestion avoidance on a global internet. IEEE Journal on selected Areas in communications **13**(8), 1465–1480 (1995)
3. Floyd, S., Henderson, T., Gurtov, A.: Rfc3782: The newreno modification to tcp's fast recovery algorithm (2004)
4. Gauss, C.F.: Gaussian elimination method solving matrix equations. https://mathworld.wolfram.com/GaussianElimination.html (1850)
5. Huang, Y., Ghaderi, M., Towsley, D., Gong, W.: Tcp performance in coded wireless mesh networks. In: 2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. pp. 179–187. IEEE (2008)
6. IANA: 5g network deployment scenarios. https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml (2022)
7. Martignon, F., Fratta, L.: Loss differentiation schemes for tcp over wireless networks. In: International Workshop on Quality of Service in Multiservice IP Networks. pp. 586–599. Springer (2004)
8. Matsuda, T., Noguchi, T., Takine, T.: Survey of network coding and its applications. IEICE transactions on communications **94**(3), 698–717 (2011)
9. Paxson, V., Allman, M.: Rfc 2988: Computing tcp's retransmission timer (2000)
10. Ruiz, H.M., Kieffer, M., Pesquet-Popescu, B.: Redundancy adaptation scheme for network coding with tcp. In: 2012 International Symposium on Network Coding (NetCod). pp. 49–54. IEEE (2012)
11. S. Bradner, V.P.: Rfc2780: Iana allocation guidelines for values in the internet protocol and related headers (2000)
12. S. McCanne, S.F., Fall, K.: Network simulator. Public domain software (1995)
13. Sundararajan, J.K., Shah, D., Médard, M., Mitzenmacher, M., Barros, J.: Network coding meets tcp. In: IEEE INFOCOM 2009. pp. 280–288. IEEE (2009)
14. Tian, Y., Xu, K., Ansari, N.: Tcp in wireless environments: problems and solutions. IEEE Communications Magazine **43**(3), S27–S32 (2005)
15. Wang, C., Liu, J., Li, B., Sohraby, K., Hou, Y.T.: Lred: a robust and responsive aqm algorithm using packet loss ratio measurement. IEEE Transactions on Parallel and Distributed Systems **18**(1), 29–43 (2006)