Applying Loss-rate Driven Network Coding to Transmission Control Protocol

Yihjia Tsai
Dept. of Computer Science and Information Engineering
Tamkang University
Taipei, Taiwan
eplusplus@gmail.com

Chaoyuan Chiang
Dept. of Computer Science and Information Engineering
Tamkang University
Taipei, Taiwan
cory.scorpio@gmail.com

***Abstract*—Transmission control is an important issue in the Internet or other computer networks today. The retransmission scheme in TCP cannot have the best throughput in the network scenarios with more wireless links or complicated topologies. Some related works proposed the solution by network coding. Network coding is suitable for generate the redundant data for error correction. In this paper, we discussed such solutions. Then we proposed the loss-rate driven coding, LRC, for transmission control. The proposed mechanism can minimize the coding operations. Applying LRC to TCP will have lower power consumption and lower computing resource requirement.**

Keywords— linear network coding; transmission control; TCP

I. INTRODUCTION

Transport layer is an important part in the protocol stack of Internet and other modern computer networks today. The Transmission Control Protocol, TCP, is one of the transport layer protocols. TCP provides reliable communication for upper layer applications by the acknowledgment mechanism. With acknowledgment mechanism, TCP can detect the segment loss and sense the network condition. Once a segment loosed or timed-out, it represents the network congestion occurred. TCP would control the transmission rate by adjusting the congestion window size to avoid network congestion.

TCP was developed for wired networks at beginning. Wired networks are simple, if the segment loss became often, the network is in congestion. TCP will reduce the transmission rate once congestion occurs. In the modern network scenario, more wireless links, more carrier types, more complicated and larger topologies, there are more reasons caused segment loss or time-out, such as interference, fading, or temporary fault in the intermediate network device. The retransmission and congestion window adjustment policy of TCP may decrease the efficiency of transmission. In other words, TCP cannot reach the optimal usage of network throughput in some situations. For this issue, there exist some researches using network coding to improve the usage of network throughput in TCP transmission.

Network coding was first proposed in 2000 [1], which provides solution for optimizing network throughput in wireless networks, such as [5]. The major idea is combining data by XOR operations in broadcasting networks to minimize the amount of transmissions. A branch of network coding is the linear network coding [10]. Linear network coding is often applied in guaranteeing the fairness of peer-to-peer content distribution [2] or generating redundant data for error-correction [14]. The most interesting part of linear network coding is that it can distribute a large content into *n* pieces equally in logical. For peer-to-peer content distribution, each peer can get the original content by decoding the *n* received coded pieces. Consider the transmission in lossy networks, if sender divides the data into blocks and transmit in linear coding continuously, receiver can decode and get the original data after receiving any *n* blocks. The receiver doesn't have to care the order of blocks. Some researches applied the idea above to TCP with lossy networks. We discussed those works in next section below.

Linear network coding seems a good solution. However, coding takes system resources on the devices. The implementations of linear network coding use the algebraic operations on Galois Field. The operations in Galois Field are performing by bitwise XOR or bit-shift operations. The decoding procedure is more complicated than encoding procedure. Minimizing the coded data and reducing the decoding works will have better computing efficiency for the devices. In other words, that would be more friendly to embedded devices with limited system resources.

In this paper, we proposed the idea of loss-rate driven coding. We designed a transmission control mechanism, which use network coding as redundancy. The amount of redundancy is related to loss-rare sensed. In the second section, we reviewed the TCP mechanism and related works. Then, we proposed our loss-rate driven coding idea in third section, followed by the performance evaluation and conclusions.

II. TCP MECHANISM REVIEW

A. Congestion Control

The TCP data unit called segment, the data from upper layer would be divided into segments and transmit. Since the bandwidth of network links and the buffer size of network devices are limited, the packet would be dropped if the data comes faster than the bandwidth of network link. Once packets have been dropped in the lower layer, the transport layer segments would be lost or broken. TCP introduced the acknowledgment mechanism. Sender transmits the segment with sequence number and header checksum. Receiver checks

the received segment. If the segment is received correctly, receiver sends an acknowledgement, ACK, to inform the sender. The ACK contains the sequence number of next expected segment. If the segment is lost or incorrect, receiver will repeat the same ACK until received the expected segment correctly. Thus, the completeness of data can be guaranteed.

For higher bandwidth usage, TCP will try to transmit a group of segments continuously, known as congestion window. The amount of segments in the group called window size. The window size will be increased when there's no transmission timed-out or network congestion. TCP will repeat a congestion window until received next correct ACK. Once the transmission timed-out or congestion occurs, TCP will reduce the congestion window size, slow down the transmission rate, to make the segments transmitted correctly.

B. Selective ACK

In the TCP congestion window Go-Back-N mechanism, any segment loss or fault will make sender retransmit all the rest segments in the window, some segments will be transmitted more than once, which is not quietly efficiently. The selective ACK, SACK, specified in RFC2018 [9], which allows receiver to ask sender retransmit specified segments.

The SACK scheme solved the redundant retransmission problem. Receiver specified the SACK options in the header of ACK. But there still exist extra costs. Sender takes time and computing resources to process the SACK and retransmit the specified segments.

C. Network Coding Issues

Network coding can help to recover the lost segments. There exist some researches applying network coding for peer-to-peer content distribution, such as [2]. The most interesting part of network coding in this field is that data can divide into some pieces uniformly in logical. That is, if the original divided in to n blocks, any peer can decode and get the original data after collect n coded blocks. This characteristic also can be applied on error correction, such as [14]. The network coded TCP, TCP/NC, was proposed by Sundararajan et al, [11]-[13]. TCP/NC adds a coding layer between IP layer and TCP layer. The coding layer performance the linear network coding operations to encode or decode the segments. Kim et al, [7] and [8], analyzed TCP/NC and concluded that TCP/NC may have better throughput and better efficiency in lossy networks. Chan et al. [3] proposed the adaptive network coded TCP. They focus on adjusting the size of coding window according to the loss-rate. That is a quiet good idea, but there is no discuss about the compatibility issue.

In this paper, we use this characteristic of network coding to recover the lost segments in TCP. Our goal is to minimize the coding operations and get optimal performance.

III. Loss-rate Driven Coding

In this section, we proposed the loss-rate driven coding, LRC, and combined it with TCP, expressed as TCP/LRC below. The basic idea of our loss-rate driven coding, LRC, is sensing the segment loss rate and using coded segment as redundancy to recover the lost segments. We described our method in the following sub-sections.

A. Transmission Model

The sender and receiver both maintain their own coding buffer. The coding buffer is a cyclic queue, called sender queue, Q_S, and receiver queue, Q_R, in sender side and receiver side, respectively. The data came from upper layer would be packed into segments in sender side. Then, the sender would transmit the segment and put a copy into sender queue, Q_S. After received the segment, receiver would put a copy into receiver queue, Q_R, and process the data in the segment for upper layer. For both of Q_S and Q_R, the eldest segment is stored in the first element while the latest segment stored in the last element. In normal condition, Q_S and Q_R will rotate simultaneously with a little delay, like a tape, shown in Fig. 1.

When the network congestion, segment loss or fault occurs, Q_S will rotate faster than Q_R. And some segments in Q_R may not store in right order, shown as Fig. 2. This condition should be fixed. For Q_R, there should be at least one segment in right order. The segments stored in Q_R with the right order are the candidates of coding head. Once sender detects the loos-rate greater than the threshold, sender will pause the processing of new data. Then encode the segments in Q_S into coded segments and transmit. The first segment in Q_S is the coding head. Receiver will receive the coded segments and put them into Q_R until the first segment in Q_R is coding head. That is, the segments stored in the first element of Q_S and Q_R have the same sequence number, shown as Fig. 3. Then, receiver can decode the coded segments in Q_R and get the original data. Thus, Q_S and Q_R become synchronized again and the problem has been fixed.

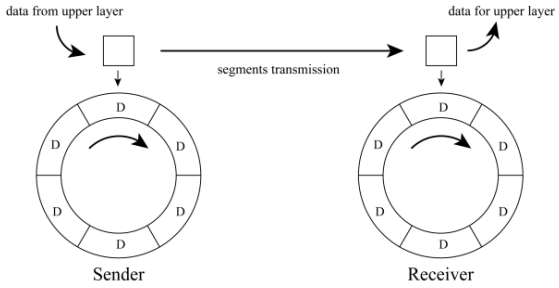

Fig. 1. Both sender and receiver have a cylic queue with same size, when transmitting under normal condition, the access pointers of two cylic queue shoud rotate simutaneously.

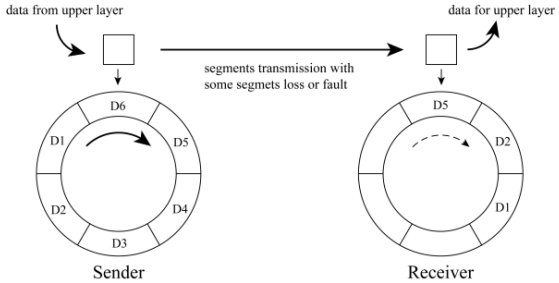

Fig. 2. When segment loss or fault occurs, the access pointer rotation will incompatable, and there may have out-of-ordered segments is receiver's queue.

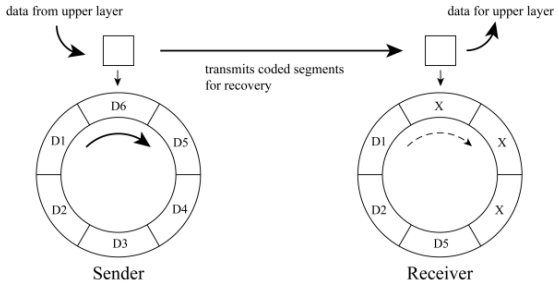

Fig. 3. The amount of lost segments recovered by coded blocks, performing decode operations can get the original segments.

Fig. 4 is the state diagram of proposed TCP/LRC mechanism. The transmission begins from the starting state, similar to the slow start procedure of original TCP. The segments will be transmitted in minimal transmission rate, which will be increased each next round. When Q_S is fully-filled, it switches to normal transmission state. Sender will detect segment loss in this state. Once the segment loss exceeded the threshold, it switches to the coding recovery state. In the coding recovery state, sender has paused processing new data, and start to send the linear combination of the segments in Q_S. Receiver collects the coded segments and performs the decoding procedure. After recovered the lost segments, it switches back to normal transmission state.

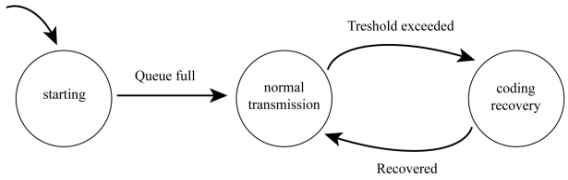

Fig. 4. State diagram of proposed TCP/LRC mechanism.

B. Sensing Segment Loss-rate

The ACK in original TCP sends the sequence number of next expected segment. In TCP/LRC, ACK sends the amount of correctly-received segments instead. Generally, the problem can be fixed by coding if the amount of lost segment is lower than the buffer size. As we discussed above, there should be at least one right-ordered segment in Q_R. But there may have segments not in the right order, which will push the right-ordered segments out of queue. For this condition, the ACK should contain two indicators, the amount of right-ordered segments, R_c, and the amount of out-of-ordered segments, R_g.

Sender monitors R_c and R_g by receiving the ACKs. If R_c is lower than the last segment of Q_S or R_g is greater than buffer size, the problem cannot be fixed by coding. Such situation should be avoided. So we defined two thresholds for R_c and R_g, the T_c and T_g. If R_c lower than T_c or R_g greater than T_g, sender will pause processing new data and start sending coded segment.

C. Segment Format

Considering the compatibility with original TCP, we use the regular TCP segment format. The header is briefly shown as TABLE I. . In original TCP, when the ACK flag is set, the sequence number of next expected segment will be filled into ACK number field. For the proposed TCP/LRC mechanism, when ACK flag is set, R_c will be filled into ACK number filled and R_g will be filled into URG pointer field. URG mechanism is for urgent data in TCP. When the URG flag is set, it means the data need process quickly, and the URG pointer field will be the position of the urgent data. When the segment with URG specified is received, receiver will process the segment with higher priority. For the segment lost condition, the urgency is also expired. So here we use the URG point field to transfer additional information in TCP/LRC. And URG mechanism is still available in normal state.

There are nine bits for flags in the TCP header. The three reserved bits generally set to zero in original TCP. For identification of LRC, we use two of them as flags. First one is the LRC flag, which will be set if LRC mechanism is applied in current transmission. The other is COD flag, which will be set if the segment is TCP/LRC coded segment.

TABLE I. TCP SEGMENT HEADER

Bit	**Fields**				**Bit**
0	Source port			Destination port	31
32	Sequence number				63
64	Acknowledgment number				95
96	*1	*2	Flags	Window size	127
128	Checksum			URG pointer	159

*1. Data offset
*2. Reserved

D. Encoding Procedure

Linear network coding performs the algebraic operations in Galois Field. Some researches applied the random linear network coding, which will choose coding coefficients randomly. Random linear network coding can ensure that the linear combination has a solution. But it will take bandwidth to transmit the coefficients to receivers. In LRC, we use a hash function, H, to generate the coding coefficients. With the hash function, we only need to put the hash seed in the header. We made LRC perform the coding operations in $GF(2^8)$. Many researches use $GF(2^8)$ because each number in $GF(2^8)$ is a byte. This makes it easy for implementation.

When entered the coding recovery state, sender would pick a hash seed, k, for different coded segment. Then get the n code coefficients for n segments, as (1). And put the linear combination, X, in the payload of coded segment, as (2). The hash seed, k, will filled in the URG pointer field in the segment header. The sequence number of coding head, D_1, will be put in the sequence number field of coded segment. Thus, receiver can identify the coded segments as same group.

$$\{C_1, C_2, \ldots, C_n\} = H(k) \quad (1)$$

$$X = C_1D_1 + C_2D_2 + \cdots + C_nD_n = \sum_{i=1}^{n}(C_iD_i) \quad (2)$$

E. Decoding Procedure

In linear network coding, *n* coded segments can be decode and get the original data by the operations in (3). In LRC, we hope the amount of coded segments is minimized. When received a coded segment, receiver will unpack it in the buffer and determine whether it has the parts of the coded segment by the coding head number. If receiver already has the u^{th} segment of coded segment *X*, it will do the operation as (4) to remove the u^{th} segment from the coded segment. *X'* denotes the coded segment without the u^{th} segment and C_u denotes the coding coefficient of *X*. Because the addition operation in Galois field can be performed by XOR, adding C_uD_u equals to remove it from *X*. The coding coefficients can be extracted from the hash function, *H*, with the hash seed in the URG pointer field of the segment header. Receiver also prepares a coding coefficient mask, *M*, once it received the coded block. The mask, *M*, is a binary array, or vector, with all zeros. When receiver find out it already has the u^{th} segment, it will also set the u^{th} bit of *M* to one.

$$\begin{bmatrix} D_1 \\ \vdots \\ D_n \end{bmatrix} = \begin{bmatrix} C_1^1 & \cdots & C_n^1 \\ \vdots & \ddots & \vdots \\ C_1^n & \cdots & C_n^n \end{bmatrix}^{-1} \times \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} \quad (3)$$

$$X' = X + C_uD_u \quad (4)$$

When there are *m* zeros in *M* and receiver has received *m* coded segments with same coding head, receiver can decode and get the *m* original segments. Receiver will re-order the coding coefficients, only use the C_i with *M(i)* is zero. And the size of coefficient matrix in (3) will be reduced. Then, receiver can do the decoding operations and get the *m* original segments.

F. Congestion Control

Although network coding can improve the throughput, there still has the bandwidth limit of network link. Transmit the segments too fast will cause network congestion. Original TCP use the congestion window and ACK for congestion control. When transmitted an amount of segments, TCP will wait for the correct ACK before transmit other segments. TCP/NC adds a new layer between TCP layer and IP layer, so the congestion control is still handling by TCP.

In this paper, TCP/LRC also maintains the congestion window mechanism of TCP. After sender transmitted an amount of segments, it would wait for the ACK before continues. When entered the coding recovery state, the congestion window size will be reduced. We are studying for advanced in this issue, to adjust the transmission more accurate by R_c and R_g. This may become our future work.

IV. Performance Evaluation

A. Theoretical Induction

The first indicator of performance is throughput, which represents the amount of data can be transmitted per time period. For TCP without SACK, we assumed the loss probability of each segment is *q*, and the mean value of window size is *w*. The Z denotes the expected amount of actual transmitted segments, that is, considered the retransmitted or redundant segments. We calculate the usage of network link like (5). According to the Go-Back-N retransmission scheme of TCP, *Z* will be the equation in (6). Our proposed TCP/LRC mechanism can reduce *Z* to *Z'* shown in (7).

$$\frac{amount\ of\ original\ data}{expected\ amount\ of\ transmission} = \frac{w}{Z} \quad (5)$$

$$\mathrm{Z} = \mathrm{w} + \left(\frac{1+w}{2}\right)(1-(1-q)^w) \quad (6)$$

$$Z' = w(1+q) \quad (7)$$

Comparing with TCP/NC, our TCP/LRC mechanism has no transmission overhead because we use a hash function to generate the coding coefficients. The seed of hash function can be filled in the URG pointer of TCP header. TCP/LRC doesn't need extra transmission for coded segments. Moreover, TCP/LRC do the coding operations only in needed condition, the computing overhead can be minimized.

For the computing complexity, according to (2), the complexity of encoding *n* segments is $O(n^2)$. This complexity level is similar to some searching and sorting algorithms. So encoding operation is acceptable for most of systems. The decoding operations in (3) have a complexity of $O(n^3)$. This is more complicated than most of other operations and is possible to solve by hardware decoding in the future. The proposed LRC mechanism minimized the coding operations, also minimized the extra costs of network coding.

B. Packet Loss Model

For the accuracy of simulation, we studied the packet loss model of real computer networks with wireless media. The packet losses we discuss here are caused by wireless interference or fading, or sporadic fault in the network device, not caused by link failed or network device failed. Hohlfeld et al, [6] and [4], analyzed the packet loss model by the Gilbert-Elliott Model in Fig. 5, which is inspired by Markov Model. Each of the network links is either in good (G) or bad (B) state. The probability of a correctly-transmitted bit in good state is *k*, while in bad state is *h*. In other words, the bit error rates in the two states are *1-k* and *1-h*, respectively. In the good state, the probability of switching to bad state is *p*, staying in good state is *1-p*. In the bad state, the probability of switching to good state is *r*, staying in bad state is *1-r*. These works discussed the packet loss in networks including wireless network and mobile network. They focus on the quality of media streams, the UDP-like traffics, through the networks. The UDP-like traffics are quietly different from TCP traffics. Without transmission

control and the retransmission mechanism, UDP-like protocols cannot provide reliable transmission, but may have higher throughput because they do not have to wait for ACK. Studying the packet loss-model of UDP-like traffics can help to improve the performance of UDP-like protocol. For TCP with network coding, understanding the loss-model also can help to optimize the amount of retransmission and improve the performance.

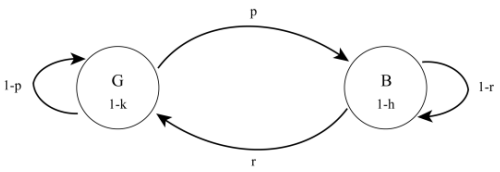

Fig. 5. The Gilber-Elliot Model defines the network link with two states, good (G) state and bad (B) state.

C. Simulation Results

For simulation, we emulate the lossy network environment by the Gilbert-Elliot Model. The probability from the good state to the bad state is 0.1, while the probability from bad state to good state is 0.3. And the bit error rate in the good state and bad state are 0.000001 and 0.000002 respectively. The content length is 240,000 bytes while each segment carrying 1200 bytes. There are 200 original segments in each round. We simulated 20 rounds, after each round, the bit error rate in both good state and bad state increased 0.000003. We compared the proposed mechanism, labeled as TCP/LRC here, with TCP/NC and TCP Reno. Fig. 6 shows the relationship between dropped segments and actual transmitted segments. As the growth of bit error rate, the amount of dropped segments also increased after each round. TCP Reno detects segment loss by time-out or triple-duplicated ACK and discarded the out-of-ordered segments, so the number of actual transmitted segments was much more than the dropped segments plus the original segments. TCP/NC and TCP/LRC transmit the linear combination for the error recovery condition, so the number of actual transmitted segments equal to the dropped segments plus the original segment.

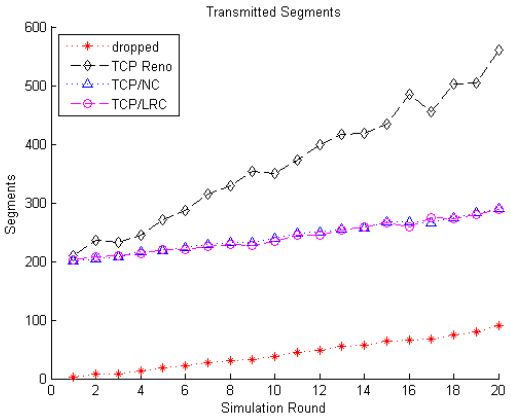

Fig. 6. Comparison of the exact transmitted segments with the bit error rate increased after each round..

We also compared the total amount of transmitted data in transport layer, shown as Fig. 7. TCP/NC has additional information for network coding in the segment. As a result, TCP/NC has to transmit more data. Our TCP/LRC just uses the original TCP header, so it will be more efficient when transmitting large content.

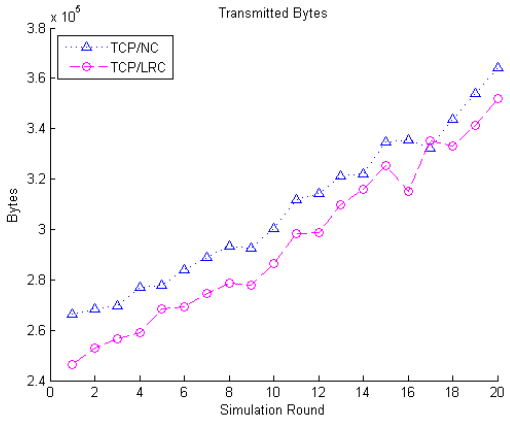

Fig. 7. Comparison of TCP/LRC and TCP/NC by the amout of transmitted data, in bytes.

For the computing complexity, Fig. 8 showed that our TCP/LRC has less coded segments than TCP/NC. The encoding and decoding procedure in TCP/LRC is minimized. This let TCP/LRC can have lower power consumption and lower computing resource requirement.

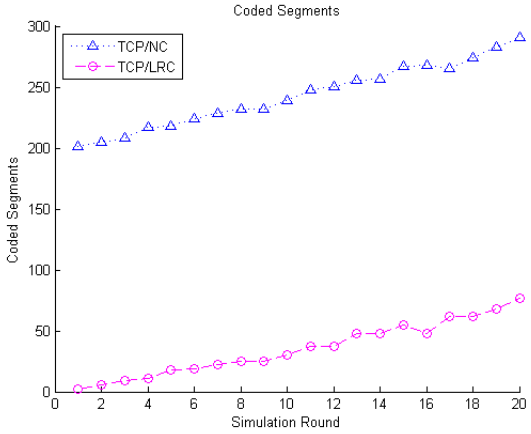

Fig. 8. Comparison of TCP/LRC and TCP/NC by the amount of coded segments.

V. Conclusions

In this paper, we proposed the loss-rate driven coding, LRC, and the TCP/LRC mechanism. With this mechanism, the amount of encoding operations and decoding operations can be minimized. The coded segment will be used only if needed. And the amount of decoding operations is also reduced to the actual needed amount. Thus, this mechanism will be easier to implement in the embedded systems with limited system resources. We also used the original TCP segment header in

the proposed mechanism. This makes the implement of TCP/LRC compatible with the original TCP. In normal status, the features of original TCP, such as URG, is still available. For the future works, we are studying about the congestion control issue. With a more accurate congestion control scheme, we hope the throughput could be optimized in the future.

REFERENCES

[1] R. Ahlswede, N. Cai, S. Y. Li, and R. W. Yeung, "Network information flow," *Information Theory, IEEE Transactions on,* vol. 46, pp. 1204-1216, 2000

[2] Gkantsidis, C.; Rodriguez, P.R., "Network coding for large scale content distribution," *INFOCOM 2005*, *24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol.4, no., pp.2235,2245 vol. 4, 13-17 March 2005

[3] Chan, Yi-Cheng, and Ya-Yi Hu, "Adaptive Network Coding Scheme for TCP over Wireless Sensor Networks," *International Journal of Computers*, *Communications and Control 8.6*, 2013

[4] Hasslinger, Gerhard; Hohlfeld, Oliver, "The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet," *Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2008 14th GI/ITG Conference* - , vol., no., pp.1,15, March 31 2008-April 2 2008

[5] 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. 16*, 3 (June 2008), pp. 497-510, 2008

[6] Hohlfeld, Oliver. "Stochastic packet loss model to evaluate QoE impairments "*PIK-Praxis der Informationsverarbeitung und Kommunikation* 32.1 (2009) pp. 53-56, 2009

[7] M. Kim, T. Klein, E. Soljanin, J. Barros, and M. Medard, "Modeling Network Coded TCP: Analysis of Throughput and Energy Cost," *arXiv preprint arXiv:1208.3212,* 2012

[8] M. Kim, M. Medard, and J. o. Barros, "Modeling network coded TCP throughput: A simple model and its validation," 2011

[9] Mathis, M. et al., "RFC 2018," *Internet Engineering Task Force (IETF)*, 1996

[10] S. Y. Li, R. W. Yeung, and N. Cai, "Linear network coding," *Information Theory, IEEE Transactions on,* vol. 49, pp. 371-381, 2003

[11] J. K. Sundararajan, S. Jakubczak, M. Medard, M. Mitzenmacher, and J. Barros, "Interfacing network coding with TCP: an implementation," *arXiv preprint arXiv:0908.1564,* 2009

[12] J. K. Sundararajan, D. Shah, M. Medard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets TCP: Theory and implementation," *Proceedings of the IEEE,* vol. 99, pp. 490-512, 2011

[13] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros, "Network coding meets TCP," *INFOCOM 2009, IEEE*, vol., no., pp.280,288, 19-25 April 2009

[14] Zhen Zhang, "Linear Network Error Correction Codes in Packet Networks," *Information Theory*, *IEEE Transactions on*, vol.54, no.1, pp.209,218, Jan. 2008