Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. VI (2011), No. 1 (March), pp. 53-62

Effective Retransmission in Network Coding for TCP

J. Chen, L.X. Liu, X.H. Hu, W. Tan

Jing Chen
Information Center
The Ministry of Science and Technology of the People's Republic of China
15#B, Fuxing Road, Beijing, 100862, P.R. China
E-mail: dongdcj@gmail.com

Lixiang Liu, Xiaohui Hu
Institute of Software, Chinese Academy of Sciences
4# South Fourth Street, Zhong Guan Cun Street, Beijing, China, 100190

Wei Tan
Baidu, Inc
E-mail: tanweildd@gmail.com

Abstract:
Incorporating network coding into TCP has the advantage of masking packet losses from the congestion control algorithm. It could make a lossy channel appear as a lossless channel for TCP, therefore the transport protocol can only focus on handling congestion. However, most schemes do not consider the decoding delay, thus are not suitable to be implemented in practical systems. We propose a novel feedback based network coding (FNC) retransmission scheme which has high throughput and quite low decoding delay without sacrificing throughput. It uses the implicit information of the seen scheme to acquire the exact number of packets the receiver needs for decoding all packets based on feedback. We also change the encoding rules of retransmission, so as to decode part of packets in advance. The scheme can work well on handling not only random losses but also bursty losses. Our scheme also keeps the end-to-end philosophy of TCP that the coding operations are only performed at the end hosts. Thus it is easier to be implemented in practical systems. Simulation results show that our scheme significantly outperforms the previous coding approach in reducing decoding delay, and obtains the throughput which is close to the scenarios where there is zero error loss. It is particularly useful for streaming applications.

Keywords: network coding, TCP, decoding delay, retransmission.

1 Introduction

It is well known that TCP suffers poor performance in lossy wireless networks. Even if a perfect congestion control algorithm can avoid congestion loss in transmission, there are still non-congestion losses (including random losses with a fixed bit error rate and bursty losses due to bad weather or signal shadowing, etc.), which necessarily degrade the transmission performance. Network coding has emerged as an important potential approach in the operation of communication networks [1]. The core idea is that the sender transmits coded packets combined with unacknowledged original packets rather than transmitting individual packets. Thus sending a packet can be seen as adding a packet to the pool and acknowledgement as removing the received packets from the pool. Each transmission is not affected by any other losses. Thus, incorporating

Copyright © 2006-2011 by CCC Publications

network coding with TCP is a natural way to enhance the robustness and effectiveness of data transmission in lossy channels.

The ARQ for network coding (ANC) scheme presented in [2] defines the seen packet as an abstraction for the case in which a packet has not yet been decoded, but can be safely removed from the sender's buffer. The expected queue size of the scheme is reduced from the traditional length $\Omega((1-\varepsilon)^{-2})$, to $\Omega((1-\varepsilon)^{-1})$ where is the erasure probability. However, the weak receivers are unlikely to recover from erasures in reasonable time, since the decoding delay becomes very large.

TCP/NC [3] uses seen scheme to mask losses from the congestion control algorithm. It aims to make a lossy channel appear as a lossless channel to TCP, so the congestion control protocol does not need to pay attention to the error losses and thus can focus solely on the congestion. In fact, masking losses from TCP was considered earlier by using link layer retransmission [4]. Yet it has been noted in [5], [6] that the interaction between link layer retransmission and TCP's retransmission is complicated and the performance may suffer due to independent retransmission protocols at different layers. TCP/NC uses a constant redundancy factor for retransmission in order to compensate for the loss rate of the channel and match TCP's sending rate. However, if it suffers bursty losses due to bad weather or signal shadowing, the algorithm must wait a long time to decode all packets. In fact, most coding schemes do not take decoding delay into consideration. The receiver has to wait for a considerable number of packets before it can decode the data. Consequently, it is hard for them to deploy in a real system in spite of the benefits in terms of throughput and robustness the network coding can bring [7], [8].

The work by Barros et al. [9] redesigns the encoding rules and stages of the ANC scheme to decrease decoding delay, which is useful for streaming applications with special delay requirements. However, it reduces delay by sacrificing some of the throughput. And its expected queue size increases from $\Omega((1-\varepsilon)^{-1})$ in ANC to $\Omega((1-\varepsilon)^{-2})$, since the sender cannot discard a packet after it is being seen by all senders.

Therefore, when incorporating the existing coding mechanisms with TCP in wireless environment, we faced two main problems: (1) retransmission schemes which are triggered by a static parameter such as a fixed delay threshold or a redundancy factor, can not balance both throughput and decoding delay. This is shown in Sec V., Fig. 3 [9], and Sec 3 of our paper. (2) The retransmission schemes require the fixed loss rate and are deeply influenced by it. As a result it cannot handle bursty losses well.

To overcome the disadvantages in existing approaches, we provide the end-to-end Feedback based Network Coding (FNC) retransmission scheme which makes use of the implicit information of the seen scheme to obtain the exact number of packets needed by the receiver to decode all data as soon as possible. We also change the encoding rules to decode part of the packets in advance. Our effective retransmission scheme can mask losses better than the previous scheme, and hence significantly improve the performance of decoding delay and throughput under both random losses and bursty losses. Simulation results show that our scheme significantly outperforms the previous coding approach in reducing decoding delay while increasing throughput. It obtains the throughput which is close to the scenarios where there is zero error loss.

The remainder of the paper is organized as follows. Section 2 introduces the terminology and describes basic ideas of network coding in TCP with the seen scheme. Section 3 proposes our new network coding retransmission scheme FNC. The corresponding simulation results are presented in Section 4. We conclude the paper in Section 5.

2 Essential Background

We treat packets as vectors over a finite field F_q of size q. The k^{th} packet generated by the source has an index k and is denoted as p_k. When the source is allowed to transmit, it sends a random linear combination of all packets instead of the original packet. We will firstly explain the seen scheme with logical description and then illustrate its implementation in the existing protocol stacks.

2.1 ARQ for Network Coding

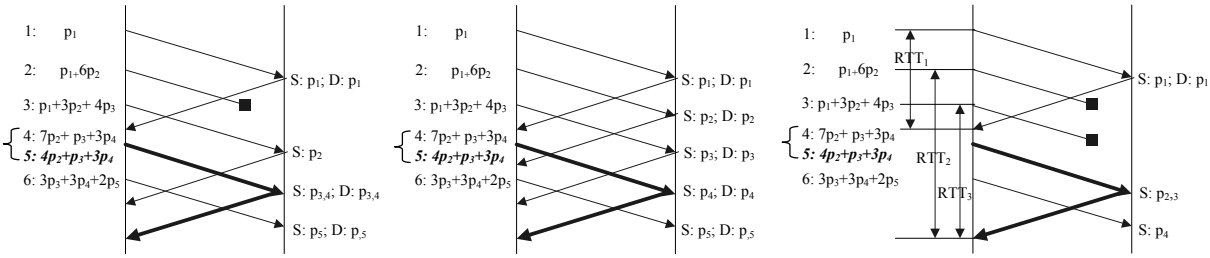
The ARQ for network coding (ANC) scheme presented in [2], is designed to handle loss in a multicast environment. In the scheme, the decision of which packets to combine relies on the concept of the seen packets. A packet p_k is said to be seen by a receiver if it has enough information to compute a linear combination in the form of p_k + q, in which q is a linear combination of packets that are newer than p_k, i.e. q = $\sum_{l>k} \alpha_l \cdot p_l$, with $\alpha_l \in F_q$ for all $l > k$. The receiver acknowledges the oldest unseen packet, so the sender always transmits a packet that is a combination of the oldest unseen packets of each receiver. A packet can be dropped from the sender queue when it is seen by all receivers. This provides an efficient method to keep the sender's queue sizes small, although the receiver may not decode all packets. It is demonstrated to be throughput optimal for the case of Poisson arrivals, perfect feedback, and identical erasure probabilities on all channels, because each reception is innovative.

2.2 Network Coding for TCP

The reference system for our scheme is the TCP/NC protocol [3], which is designed with respect to a single source that generates a stream of packets to one sink. It incorporates the seen scheme with congestion control and introduces a new network coding layer between the transport layer and the network layer in the protocol stack, which masks packet losses from congestion control algorithm.

There are several modifications on ANC to be fit for end-to-end connection. First, the sender transmits random linear combinations of packets in the coding window, instead of combinations of the oldest unseen packets of each receiver. The receiver acknowledges the oldest unseen packet although it may not be decoded yet. There will never be any duplicate ACKs as each reception is innovative. Every ACK will cause the congestion window to advance, so it is not proper to apply fast retransmit/recovery algorithms which use three duplicate ACKs as the packet loss indication. Therefore TCP/NC chooses TCP Vegas [12] as the congestion control approach and introduces a novel RTT estimation algorithm. It matches the newly arrived ACK with the transmission that occurred after the one that triggered the previous ACK, rather than the transmission that triggered this ACK. For example, in Figure 1(c), RTT_2 matches the 2^{nd} transmission, rather than the 4^{th} transmission.

In the implementation, TCP/NC embeds the network coding operations in a separate layer below TCP and above IP on two end nodes in order to naturally add network coding to the current protocol stack. In the source side, when a packet arrives at the coding layer from the transport layer, the coding layer generates a random linear combination of the packets in the coding window and sends it to the sink. In order to compensate for the loss rate of the channel and to match TCP's sending rate, for every packet from TCP, R linear combinations are sent to IP on average, where R is the constant redundancy factor equal to the reciprocal of the probability of successful reception. As an example shown in Figure 1(a), assuming packet loss rate e to be 20%, hence R is 1/0.8=1.25, which means that the sender makes a retransmission after sending four combinations triggered by TCP. The retransmission does not include any new



(a) proper retransmission (b) redundant retransmission (c)insufficient retransmission

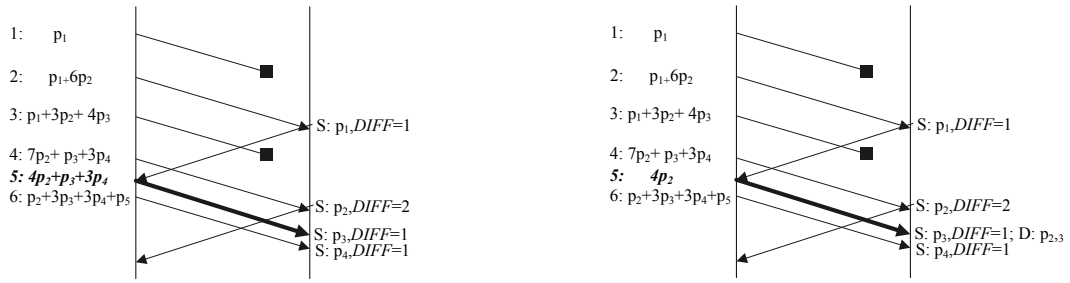
Figure 1: Example of the transmission with a redundancy factor

original packet from TCP. In Figure 1(a), the 5^{th} transmission is a retransmission which only encodes p_2, p_3 and p_4, without p_5. It compensates the loss of the 2^{nd} transmission, makes p_3, p_4 to be seen and p_3, p_4, p_5 decodable. However, the retransmission with a fixed rate may result in the following two problems: 1) the retransmission may be useless. For example, in Figure 1(b), the 5^{th} retransmission is useless since no losses happened before it. We call this redundancy or redundant retransmission. 2) The retransmission may be helpless. In Figure 1(c), the 5^{th} retransmission is valid but still can not help the receiver to decode all packets as there are two combinations lost before. Therefore, it is natural to think about using a feedback based retransmission scheme to replace the scheme with a fixed retransmission rate.

3 Network Coding Retransmission Scheme

To reduce the decoding delay and redundancy, we propose the feedback based network coding (FNC) retransmission scheme. It acknowledges the total number of coding packets required to repair a loss and then decodes all packets. In the receiver, the difference between the number of seen packets and the largest packet index in the coefficient matrix implies the number of packets that the source needs to retransmit. So we maintain two variables in the receiver side: one is the number of seen packets $SEEN\_CNT$; the other is the largest index of the received packets MAX_SEQ. For each ACK, the receiver embeds in the header not only the sequence number that equals the oldest unseen packet, but also the difference between MAX_SEQ and $SEEN\_CNT$, which is called $DIFF$. For example, suppose the source transmits the following linear combinations: $x = p_1, y = (p_1 + p_2)$ and $z = (p_1 + p_2 + p_3)$. The second transmission y is lost. So the sink only receives the linear combinations x and z. As p_1 and p_2 have been seen, $SEEN\_CNT$ is 2, the largest index MAX_SEQ is 3, and hence $DIFF = 3 - 2 = 1$. Thus $DIFF$ indicates the number of packets the receiver needs to change into the decodable state which means the receiver can decode all packets. In the sender side, when an ACK arrives, if the $DIFF$ is larger than zero, the sender uses this value to decide to retransmit in the following two steps: First, it checks to see if the difference between the current time T_{now} and the time of the last retransmission T_{last} is greater than the timeout value. If it is, the sender retransmits $DIFF$ linear combinations of first $DIFF$ packets in the coding window. This avoids retransmission more than once for losses that occurred during one RTT interval. If it is not, the sender compares the $DIFF$ received this time with the previous retransmission $LAST\_DIFF$. If the current $DIFF$ is greater than the $LAST\_DIFF$, there is new packet loss and it retransmits $(DIFF - LAST\_DIFF)$ random linear combinations of the first $(DIFF - LAST\_DIFF)$ packets in coding window.

If we try to retransmit the combinations of all packets in the coding windows, as in TCP/NC [3], then the average decoding delay of our scheme is indeed lower than NC. However, it is still



(a) coding with all packets (b) coding with the first several packets

Figure 2: Retransmission based on feedback

much higher than our expectation. Given an example in Figure 2(a), if the sender retransmits the coded packet combined with p_2, p_3 and p_4 in the current coding window in the 5^{th} transmission, the receiver is still in undecodable state and cannot decode any packets, because the $DIFF$ has changed to 2 due to the loss of the 3^{rd} transmission. Therefore, we improve the encoding rule in which retransmission packets are only combined with first $DIFF$ (or $DIFF - LAST\_DIFF$) packets in the coding window in order to make a part of packets decodable. Although we do not decode all packets together, we significantly reduce the number of undecodable packets as well as the decoding delay. Additionally, encoding less packets can reduce the overhead and complexity of encoding and decoding operations. Consequently, in the example of FNC in Figure 2(b), we only retransmit the combination containing p_2 (as $DIFF$ equals to 1) in the $5th$ transmission, and thus p_1 and p_2 are decodable.

Additionally, TCP/NC simply retransmits packets through redundancy factor to compensate for error losses, rather than for congestion losses. The retransmission in the coding layer does not consume the congestion window. However, our previously mentioned algorithm treats congestion losses and error losses the same, which may bring or exacerbate congestion. We can solve this problem by limiting the current total number of retransmission packets in the coding layer with no more than the product of the total number of transmitted packets N and the loss rate e. In contrast with TCP/NC that retransmits dispersedly, our policy can retransmit the appropriate number of packets together while they are required, and does not mask congestion losses at the same time. In other words, NC (i.e. TCP/NC) is a static retransmission scheme whereas FNC is a dynamic one. However, regarding those near-zero congestion loss algorithms such as VCP [10] and MLCP [11], it is unnecessary to adopt this policy. Thus, it can handle such losses much better, when it encounters burst or inconstant loss rate conditions.

The improved algorithm is specified below using pseudo-code:

Source side:

Initialization:

Set $LAST\_DIFF$ and T_{last} to 0.

ACK arrives from receiver:

The source side algorithm removes the seen packet from the buffer and retrieves $DIFF$ from ACK header. If this is the fourth uninterrupted time that $DIFF$ is larger than 0:

1) If ($T_{now} - T_{last} > RTT$)
   a) Set $LAST\_DIFF$=0;
   b) goto 3);
2) If $DIFF <= LAST\_DIFF$ goto 5).
3) Repeat the following ($DIFF - LAST\_DIFF$) times:
   a) Generate a random linear combination of the packets in the coding window.
   b) Deliver the packet to the IP layer.

4) Update T_{last} to the current time;
5) Update $LAST\_DIFF$ to the $DIFF$ of the new arrival.

Receiver side:

Initialization:

Set $SEEN\_CNT$ and MAX_SEQ to 0.

Packet arrives from source side:

1) Performs Gaussian elimination to update the set of seen packets.
2) Update $SEEN\_CNT$ and MAX_SEQ.
3) Add the network coding ACK header to TCP ACK, consisting of the value of $DIFF$ which is the difference between MAX_SEQ and $SEEN\_CNT$.

The algorithm not only avoids the redundant transmissions when the receiver is in the decodable state, but also retransmits the appropriate number of packets so as to make a part of packets decoded in advance. It significantly reduces the decoding delay, which is particularly useful for streaming applications with stringent delay requirement. Also, as we do not change the acknowledgment scheme of seen packet, the expected queue size for our scheme remains $\Omega((1-\varepsilon)^{-1})$ rather than $\Omega((1-\varepsilon)^{-2})$ in SNC [9] and TCP.

In contrast with traditional retransmission schemes, such as SACK etc., our scheme obtains the total number of packets for decoding. It significantly reduces overheads in the ACK header. It is also more robust and effective. Compared with TCP/NC [3], FNC has a relative small decoding delay. It can handle both random losses and unknown bursty losses. Furthermore, our scheme respects the end-to-end philosophy of TCP that coding operations are only performed at the end hosts while achieving the aim of masking losses from congestion control algorithm at the same time.

4 Simulation Results

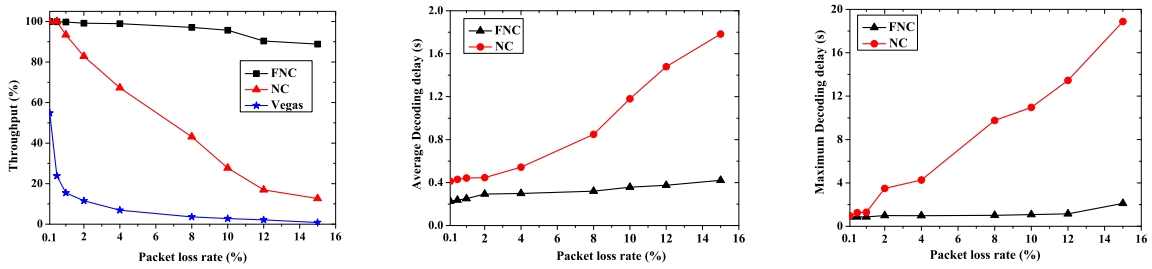
We evaluate the performance of different coding algorithms by means of the network simulator "ns-2" [13]. The basic setting is a tandem network consisting of 4 hops. The source and sink nodes are at opposite ends of the chain. The packet size is 1000 bytes. We incorporate network coding with TCP Vegas. The Vegas parameters are set as $\alpha = 28, \beta = 30, \gamma = 2$. The performance metrics are the throughput, the average decoding delay and the maximum decoding delay. The throughput with network coding is calculated as the total number of seen packets, rather than the decoded packets, divided by the simulation time. All simulations are run for at least 200s to ensure that the system reaches its steady state.

4.1 Random Losses

We first evaluate the performance of FNC under the case of a fixed packet loss rate. The basic setting is a 10Mbps link capacity, an 80ms round-trip time, and a 5% packet loss rate. We compare the throughput of NC, FNC and Vegas under a fixed loss rate with the standard Vegas under no loss rate to evaluate their performance on masking losses. Each scenario runs 20 times and we get the mean value.

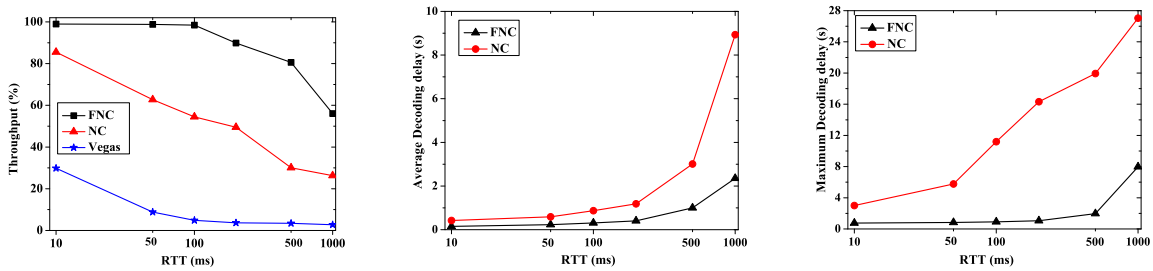
4.2 Impact of Packet Loss Rate

We first study the variation of throughput with loss rate from 0% to 15%. Figure 3(a) shows that the throughput of Vegas falls rapidly as losses increase. NC performs better than Vegas but it is worse than FNC since it only successfully masks part of losses. In contrast, FNC is very robust to losses. It maintains over 88% throughput as if there is no error loss. Our



(a) Throughput (b) Average decoding delay (c) Maximum decoding delay

Figure 3: Performance as a function of the loss rate variation



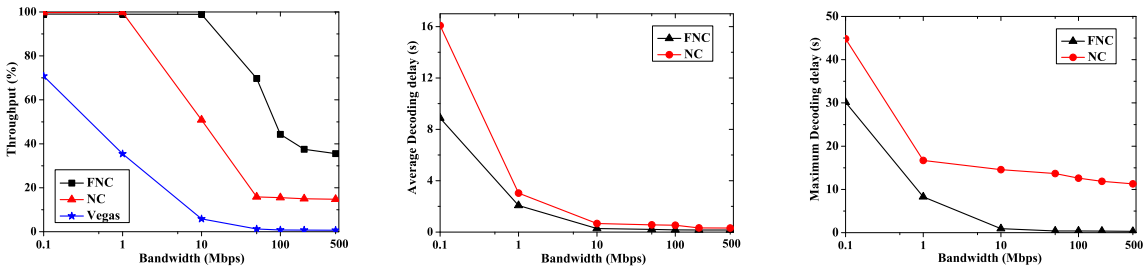
(a) Throughput (b) Average decoding delay (c) Maximum decoding delay

Figure 4: Performance as a function of RTT variation

scheme also keeps quite a low decoding delay and the average value does not increase as the loss rate increases. This is because FNC is according to the actual losses to retransmit packets dynamically. In contrast, NC is deeply affected by the loss rate since its retransmission is static. In some cases, the average decoding delay of NC is even higher than the maximum decoding delay of FNC.

4.3 Impact of Feedback Delay

Next, we evaluate the performance across a wide range of propagation delay from 10ms to 1000ms. As illustrated in Figure 4(a), our scheme performs much better than NC and Vegas in every case. The throughput in FNC is at least 2 times more than in NC. However, the FNC's throughput degrades significantly when RTT increases to 1000ms. This is because our retransmission scheme is based on feedback. NC may perform better than FNC when RTT becomes much larger. Thus our retransmission scheme can not be applied in deep space communication of which RTT may be hundreds of seconds, whereas in most real cases, RTT is no more than 1000ms. Figure 4(b) shows that the average decoding delay of NC is 2 to 10 times of FNC. The decoding delay for both schemes increases quickly when RTT grows beyond 500ms. When RTT is 500 ms, FNC's average decoding delay is 1.00165s, and when RTT is 1000ms, its average decoding delay is 2.35303s. They are approximately 2 times that of RTT, which is in accordance with our expectations. The average decoding delay of NC is 3.0087s and 8.93286s respectively, much larger than FNC.



(a) Throughput (b) Average decoding delay (c) Maximum decoding delay

Figure 5: Performance as a function of the bandwidth variation

4.4 Impact of Link Capacity

We fix the round-trip time to 100ms with a packet loss rate of 5%, and vary the link capacity from 0.1Mbps to 1000Mbps. As shown in Figure 5(b) and (c), the maximum decoding delay for NC is very high in every case, whereas both the maximum and the average decoding delay for FNC are quite low. When the bandwidth is lower than 1Mbps, the congestion control algorithm itself leads to congestion or even loss, which increases the delay. If the bandwidth is very high, one loss can lead to great throughput decrease. It is shown in Figure 5(a) that due to the limitation of Vegas itself, the performance of all three schemes is not very high when the bandwidth is larger than 100Mbps. Since our scheme can mask packet losses more effectively, it has the highest throughput in contrast with NC and Vegas without incorporating with network coding.

Overall, the decoding delay for FNC depends on the RTT, the overhead of operation of encoding and decoding rules, and the congestion control algorithm, but it is not deeply affected by the loss rate. It seldom has a long undecodable chain thus its maximum decoding delay is endurable. Furthermore, FNC masks random losses more effectively than NC does, and hence obtains higher throughput in most cases.

4.5 Bursty Loss

All previous simulations focus on the behavior of FNC under the fixed loss rate. Now, we investigate its performance in an unknown environment with unfixed loss rate. The setting of this scenario is a 10Mbps link capacity with 80 ms RTT, where the background loss rate is 0.1%. At t=40s, the loss rate is changed to 50%, and lasts for 5 seconds until t=45s. At t=60s, the loss rate is changed to 100% which means the signal is fully shadowed by obstacles, and it lasts for 2 seconds until t=62s. The tested protocols Vegas and FNC do not know these loss rate changes. Figure 6 clearly shows that FNC can quickly and effectively handle the sudden bursty losses. Before t=40s, FNC almost masks all random losses whereas Vegas without network coding is seriously affected by random losses. When the loss rate is changed to 50%, Vegas could hardly work whereas FNC still has a relatively high throughput. FNC does not mask all losses because the retransmission packet also has a 50% probability to be lost. After around t=63s, the sending rate of FNC has a sudden increase because the sender retransmits all lost packets together. Due to the limitation of the congestion control algorithm Vegas itself, the stable sending rate after this burst does not recover as before. The maximum decoding delay of FNC is 3.33053s, which contains 2 seconds during which the connection is broken. The average decoding delay is 0.0839042s, a little more than one RTT. We do not test NC because its retransmission scheme depends on the estimation of the loss rate, and it is hard to give an effective estimation algorithm.

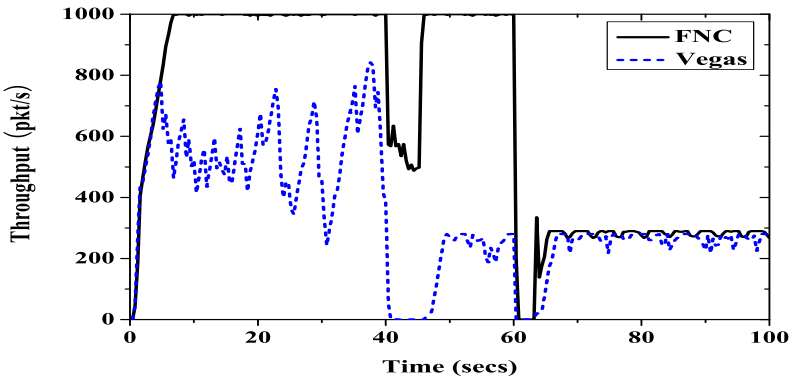


Figure 6: Performance under unknown bursty losses

5 Conclusions

Network coding is a powerful tool in fighting against non-congestion losses. However, its redundancy and decoding delay can significantly impair transmission performance so that most schemes cannot be implemented in practical systems. In our work, we propose a novel dynamic network coding retransmission scheme which makes use of the information implied in the seen scheme to acquire the exact number of packets the receiver wants instantly. As we do not retransmit packets with a stable rate or a constant redundancy factor, our approach can handle not only random losses, but also unknown bursty losses. Simulation results show that our scheme significantly outperforms the previous coding approach in reducing decoding delay and masking losses. It obtains the throughput which is close to the scenarios where there is zero error loss.

The remaining issue in our research is to evaluate the performance of incorporating network coding with other congestion control algorithms, such as those load factor-based algorithms, VCP and MLCP, etc. Furthermore, we intend to implement our algorithm in a Linux protocol stack to asses its strengths and limitations in practice.

Bibliography

[1] T. Ho, Networking from a network coding perspective, PhD Thesis, Massachusetts Institute of Technology, Dept. of EECS, May 2004.

[2] J. K. Sundararajan, D. Shah, M. Medard, ARQ for network coding, *in IEEE ISIT 2008*, Toronto, Canada, Jul, 2008.

[3] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros, Network Coding Meets TCP, *in IEEE INFOCOM, 2009*, San Francisco, USA, Apr 2009.

[4] S. Paul, E. Ayanoglu, T. F. L. Porta, K.-W. H. Chen, K. E. Sabnani, and R. D. Gitlin, An asymmetric protocol for digital cellular communications, *in Proceedings of INFOCOM, 1995.*

[5] A. DeSimone, M. C. Chuah, and O.-C. Yue, Throughput performance of transport-layer protocols over wireless LANs, *IEEE Global Telecommunications Conference (GLOBECOM '93)*, pp. 542-549 Vol. 1, 1993.

[6] H. Balakrishnan, S. Seshan, and R. H. Katz, Improving reliable transport and handoff performance in cellular wireless networks, *ACM Wireless Networks*, vol. 1, no. 4, pp. 469-481, 1995.

[7] S. Katti, H. Rahul, W Hu, Databi, M. Mcdard, and J. Crowcrofg, XORs in the Air: Practical Wireless Network Coding, *in IEEE/ACM Transactions on Networking*, 16(3): 497-510, 2008.

[8] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, Network coding: Aninstant primer, *ACM Computer Communication Review*, Jan. 2006.

[9] J. Barros, R. A. Costa, D. Munuaretto, and J. Widmer, Effective Delay Control in Online Network Coding, *in IEEE INFOCOM, 2009*, San Francisco, USA, Apr 2009.

[10] Yong Xia, L. Subramanian, I. Stoica, S. Kalyanaraman, One More Bit is Enough, *in IEEE/ACM Trans. Networking*, 16(6):1281-1294, Dec 2008.

[11] I. A. Qazi, and T. Zuai, On the Design of Load Factor based Congestion Control Protocols for Next-Generation Networks, *in IEEE INFOCOM*, Apr 2008.

[12] L. S. Bramko, S. W. O'Malley, and L. L. Peterson, TCP Vegas: New Technichques for Congestion Detection and Avoidance, *in Proceedings of the SIGCOMM '94 Symposium*, August 1994.

[13] ns-2 Network Simulator, http://www.isi.edu/nsnam/ns/