Implementation of TCP/NC protocol simulation based on OMNET++

Hongyun Zhang, Wanrong Yu, Chunqing Wu
School of Computer,
National University of Defense Technology,
Changsha, 410073, China
E-mail: grandcloud88@gmail.com

Miao Wang
State Key Laboratory of High Performance Computing,
National University of Defense Technology,
Changsha, 410073, China
E-mail: mercury.miao@gmail.com

Xiaoping Xu
Department of Scientific Research,
National University of Defense Technology,
Changsha, 410073, China
E-mail: xuxiaoping@nudt.cn

***Abstract*—TCP protocol has an awful performance in the wireless network because of the instability, high BER and long RTT of the wireless link. How to make wireless transmission more reliable and efficient has become a hot topic among relative researches. TCP/NC is a recently proposed protocol based on network coding and capable of achieving much higher throughput than TCP over lossy wireless Links. In this paper, network coding and TCP/NC are outlined firstly. And then simulation realization of TCP/NC protocol in OMNET++ is described. The performance evaluation of TCP/NC is conducted in OMNET++. The results show that TCP/NC offers significant better performance than TCP without affecting the fairness of data flow.**

Keywords-component; network coding; TCP/NC; OMNET++;

I. INTRODUCTION

It is well known that TCP performs poorly over lossy links prevalently existing in wireless systems [1][2][6]. It is because that each loss is interpreted as a congestion signal in TCP. Network coding has emerged as an important potential approach of operation on wireless network. TCP/NC incorporates network coding inside the TCP/IP protocol stack with some minor changes, and achieves much higher throughput compared to TCP over lossy wireless links. However, as TCP/NC is not open source, researches on TCP with network coding are limited.

This paper firstly illuminates the basic idea and characteristics of TCP/NC, then expatiate the implementation of TCP/NC in OMNET++. At last, some simulations are performed and the results indicate that TCP/NC is practicable and performs much better over lossy links in efficiency and reliability compared with TCP. Thus it is concluded that TCP/NC is significant to the theoretical research and practical applications of interfacing network coding with TCP. Based on the performance analysis of TCP/NC, its disadvantages are analyzed and summarized as well.

II. TCP/NC

Network coding has become a hot research spot since it was brought forward in 2000 [7][8]. However, it is not clear how to naturally add network coding to current network systems until TCP/NC [3]. Based on the concept of *seen(Definition 1)*, TCP/NC implements a *Random Network Coding(RNC)* [9][10] system based on byte stream oriented network protocol for the first time. It is reported that wireless transmission has been distinctly improved after using TCP/NC [4].

Definition 1 (Seeing a packet) [5]***:*** A node is said to have seen a packet P_k , if it has enough information to compute a linear combination of the form $P_k + Q$, where $Q = \sum_{l>k} \alpha_l P_l$ with $\alpha_l \in F_q$ for all $l > k$. Thus, Q is a linear combination involving packets with indices larger than k .

TCP/NC embeds the network coding operation in a separate layer between TCP and IP on the source and receiver side as shown in Figure 1.

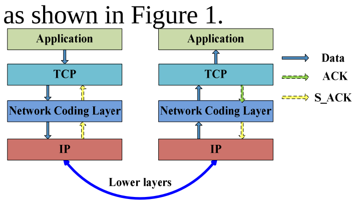

Figure 1. Protocol structure of TCP/NC

The sender side accepts packets from the TCP layer and puts them into an encoding buffer until it is ACKed by the receiver side. Every encoded packet is a random linear combination of the original packets in the encoding buffer.

The main function of the receiver side is decoding. Upon receiving a linear combination from the sender side, it first retrieves the coding coefficients from the packet header and appends them to the basis matrix of its knowledge space. Then the Gaussian elimination method is adopted to find the newly *seen* packet and decoded packet. The newly *seen* packet can be ACKed and the newly decoded packet can be submitted to TCP layer. Figure 2 illustrates an example of encoding and decoding.

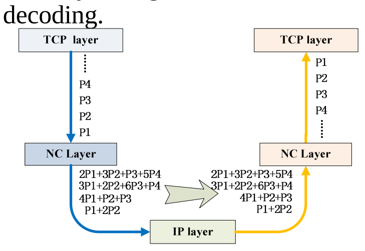

Figure 2. Example of encoding and decoding

III. The Design and Implement of TCP/NC in OMNET++

In OMNET++, designing and developing of protocol is hierarchical. The work in this paper involves TCP layer, IP layer and the network coding Layer between them.

The implement of TCP/NC in OMNET++ mainly includes 7 modules as follows:

1、One packet module. This module is implemented in NcPacket.msg defining the coding packet: NcPacket.

2、 Two simple modules: EncoderNcLayer and DecoderNcLayer. Their main functions are encoding and decoding respectively. They are implemented in HostNcLayer.ned and SinkNcLayer.ned.

3、 Two compound modules: StandardNcServer and StandardNcClient. They are standard encoding node and decoding node using TCP/NC protocol.

4、 Two network modules: SingleFlow and DoubleFlow. These two modules are defined in test.ned.

Two compound modules generate the standard encoding and decoding nodes, they interface network coding with TCP and IP. Two network modules are used to describe the network topology. Coding packet module and two simple modules are main modules which implement encoding and decoding operation and they will be described in detail next.

A. *Coding Packet Module*

The payload part of a coding packet is a random linear combination of several TCP segments. Due to the introduction of a new feedback mechanism of receiving state based on network coding, the header of the coding packet should be redesigned. The new structure of the coding packet header is shown in Figure 3.

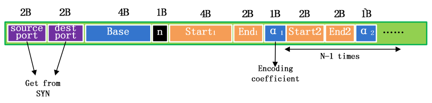

Figure 3. The header of coding packet

TABLE I. MEANING OF EVERY FIELD IN CODING PACKET HEADER

Field	**Meaning**
Source and destination port	Identify the coded packet's session
Base	The first byte that has not been ACKed
n	The number of segment involved in the NcPacket
Start$_i$	The starting byte of the ith segment
End$_i$	The ending byte of the ith segment
α $_i$	The coefficient used for the ith segment

The typical sizes (in bytes) of the various fields are signed above them. The meanings of the various fields are listed in TABLE I. The new designed coding packet is defined in NcPacket.msg in OMNET++ as shown in TABLE II.

TABLE II. NcPacket in OMNET++

```
Packet NcPacket ()
    {
            unsigned short SrcPort;
            unsigned short DestPort;
            uigned int Base;
            char n;
            char a[10];
            unsigned int start[10];
            unsigned int end[10];
    }
```

B. *EncoderNcLayer Module*

EncoderNcLayer module is the main module of sender side. It mainly contains four submodules: E_Initialization module, E_UpPacket Handling module, E_DownPacket Handling module and Encoding Buffer module. The structure of EncoderNcLayer is described in Figure 4.

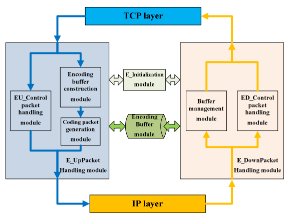
TCP layer
EU_Control packet handling module
Encoding buffer construction module
Coding packet generation module
E_UpPacket Handling module
E_Initialization module
Encoding Buffer module
Buffer management module
ED_Control packet handling module
E_DownPacket Handling module
IP layer

Figure 4. EncoderNcLayer module

1) *E_Initialization module:*

The main function of E_Initialization module is initializing finite field (F), coding window (W), encoding buffer (EB), redundancy factor(R) and some other parameters.

2) *E_UpPacket Handling module:*

This module handle packet form TCP layer. If the packet from TCP is a control packet for connection control, it was distribute to EU_Control packet handling module. And this submodule deliver control packet to IP layer without any treatment.

Encoding buffer construction module pre-processes any incoming TCP segments before adding it to the encoding buffer. The purpose of the pre-processing procedure is that the encoding and decoding operations can be performed at the granularity of packets instead of individual symbols [11], and the computational complexity and space required can also be reduced. Figure 5 shows a typical state of the coding buffer after pre-processing.

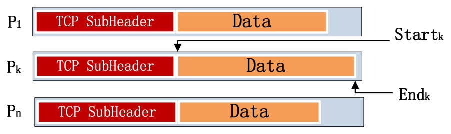

Figure 5. Typical state of coding buffer

Coding packet generation module generates and sends random linear combinations of the packets which have been pre-processed by encoding buffer construction module. The coding packet generation module is described as TABLE III.

TABLE III. NCPACKET GENERATION

```
EncoderNcLayer:: SendNcPacket ()
{
    NUM +=R;
    int k = NUM/1;
    do
    begin
        create a new NcPacket;
        generate a random combination of the packet in coding window;
        encapsulate this combination into the new NcPacket;
        setup the header of NcPacket;
        setup IPv4ControlInfo of NcPacket;
        send this NcPacket to IP layer;
        k = k-1;
    end
    until(k < = 0)
}
```

R is the redundancy parameter for the sender side, it is necessary in order to compensate for the loss rate of the wireless link.

3) E_DownPacket Handling module:

This module handle packet form IP layer. It has two submodules: Buffer management module and ED_Control packet handling module. Just like EU_Control handling packet module, ED_Control packet handling module deliver control packet to TCP layer without any treatment.

Buffer management module accomplishes the update, maintenance and management of coding buffer based on the last byte of the latest *seen* packet. A packet should be removed from the coding buffer if an ACK has arrived requesting a byte beyond the last byte of that packet.

C. DecoderNcLayer Module

As shown in Figure @@, DecoderNcLayer module is composed of three main modules: D_Initialization module, D_UpPacket Handling module, D_DownPacket Handling module and Decoding Buffer module.

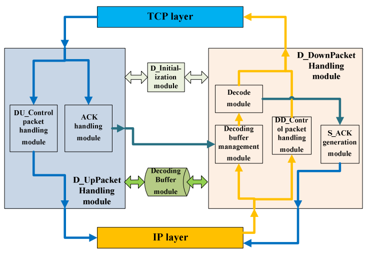
TCP layer
D_Initial-ization module
D_DownPacket Handling module
Decode module
DU_Control packet handling module
ACK handling module
Decoding buffer management module
DD_Control packet handling module
S_ACK generation module
D_UpPacket Handling module
Decoding Buffer module
IP layer

Figure 6. DecoderNcLayer module

1) D_Initialization module:

D_Initialization module initializes some parameters of receiver side, such as finite field (F), the size of decoding buffer (EB) and some other parameters of decoder.

2) D_UpPacket Handling module:

This module handle pcket form recever sider TCP layer.it has two submodules: DU_Control packet handling module and ACK handling module.

DU_Control packet handling module handle packet like EU_Control packet handling module. The ACK packet from TCP layer is distribute to ACK handling module. The value of ACK_no is send to Decoding buffer management module by ACK handling module.

3) D_DownPacket Handling module:

This module handle packet form IP layer. It consist of four submodules: Decoding buffer management module, DD_Control packet handling module, S_ACK generation module and Decode module.

DD_Control packet handling module deliver control packet for connection control to TCP layer without any change.

Decoding buffer management module is in charge of the management of decoding buffer. The Decoding buffer management module can be understood using Figure 7. It show the receiver side windows in a typical situation. A packet can be dropped if its last byte is smaller than *Base*, and has been delivered to receiver TCP layer, that is, the packet in area A. The packet containing bytes lager than *Base* still will be involved in incoming coding packet.

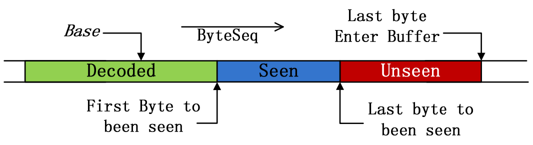

Figure 7. Decoding buffer management

Decode module performs Gaussian elimination. Before decoding, decode module firstly retrieves the coding coefficients from the packet-headers and appends them to the basis matrix of its *knowledge space* [5]. After Gaussian elimination, the oldest *unseen* byte identified. And S_ACK generate module will generate a new S_ACK according to this value. When a new packet is decoded, dummy zero symbols are pruned and a TCP segment is created to deliver to the receiver TCP layer.

TABLE IV. NCPACKET DECODING

```
DecoderNcLayer:: DecodePacket (NcPacket)
{
    remove the NcPacket header and generate a new coefficients vector;
    add this new coefficients vector to the existing coefficients matrix as a new row;
    perform the first step of Gaussian elimination to update the set of seen packets;
    activate S_ACK generation module;
    if this NcPacket make a segment to been seen
    {
        add the payload this NcPacket to the payload matrix;
        perform the second step of Gaussian elimination to update the set of decoded packets;
        if some TCP segment is decoded, deliver it to TCP layer;
    }
}
```

S_ACK generate module generate the newly S_ACK packet based on the oldest *unseen* byte, S_ACK is identical with TCP ACK packet. The sink thus pretends to have received the packet even if it cannot be decoded yet.

IV. SIMULATION AND RESULT ANALYSIS

The realization of TCP/NC is base on discrete event simulation environment OMNET++ and the open source TCP/IP protocol framework INET. The simulation environment is built on the Windows operating system. The version of OMNET++ is 4.2.2 and the version of INET is 2.0.

A. Simulation Environment Setup

The experiment is performed over IEEE 802.11b with a bit-rate of 1 Mbps, the queue type of wireless interface is DropTailQueue which the first item stored is the first item output. The frame capacity of DropTailQueue is 150. NOAH (No Ad-Hoc Routing Agent)is used as the static routes. We use FTP protocol in application layer, the apptype of sender side is TCPSessionApp and receiver side is TCPSinkApp. The file size of transmission is 40MB.Figure@@ show SingleFlow network topology and DoubleFlow network topology used to analysis and test TCP/NC protocol.

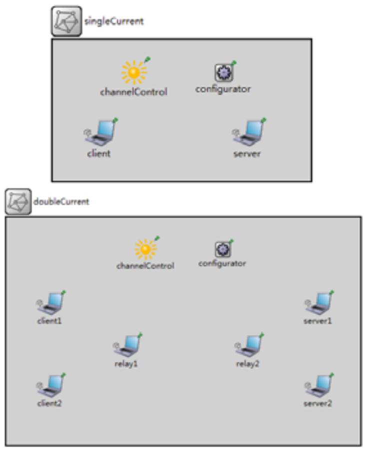

Figure 8. Network topology

Two networks (shown as Figure 8) represent two kinds of situation case. Single data stream represent the situation where only one TCP flow running in a lossy links, the send side is client and the receiver side is server. Double data stream represent the situation where two TCP flow compete with each other, one flow is start form client1and end with server1, and the second flow is from client2 to server2.

B. Result Analysis

First we study the variation of goodput with loss rate for bost TCP/NC and TCP. For TCP/NC the values of R and W have been chosen by trial and error, to be the one that maximizes the goodput. The goodput is measured using outputhook(a kind of measure class in INET framework). Each point is averaged over 4 or more iterations of such session, depending on the variability.

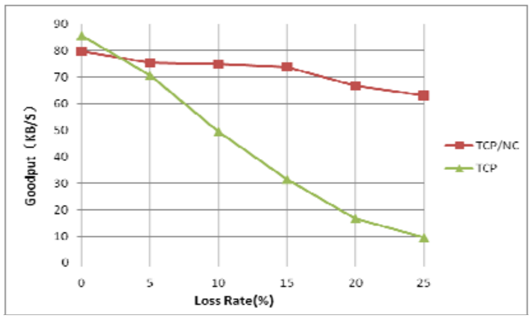

Figure 9. Goodput versus loss rate

Figure 9 shows that when the losses is very low, TCP performs better than TCP/NC, this could be because of the computational overhead of coding and decoding operation and the coding header overhead. However TCP/NC is very robust to losses and reaches a goodput that is close to capacity compared to TCP's goodput fall rapidly as losses increase.

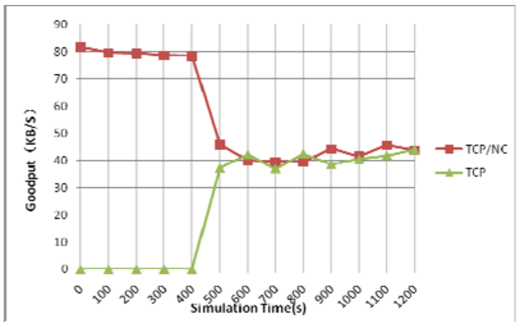

Figure 10. Fairness between TCP/NC and TCP

In order to evaluate the effect of our medication of our simulate protocol on fairness, we test the fairness of TCP/NC and TCP In the DoubleFlow network. The loss rate is set to 0% and the redundancy parameter is set to 1 for a fare comparison. In first case, TCP/NC flow start an t=0.1s, TCP flow start at t=500s. The simulation is all over in 1200s. The evaluation result is shown in Figure10. For the second case, TCP/NC flow competes with each other in one lossy link, the plot is similar to Figure 10. Both the two cases show that the effect of introducing the coding layer does not affect fairness.

V. CONCLUSION AND FUTURE WORK

How to combine network coding and TCP to reach their full potential in lossy link especially wireless link is a hot and difficult issue. This paper first introduces the problem of the TCP in the wireless environment and network coding technology. Then TCP/NC protocol is reviewed briefly, which first design a coding system that is compatible with standards TCP. we also elaborate the implementation details of TCP/NC model in OMNET++ platform. Finally, we analyze and testing the effectiveness and fairness of TCP/NC in different network scenarios compared with standard TCP. The work of this paper provides a new method for researching how to naturally add network coding to current network system, the future work could be to study how to automatic adjust R and W to adapt to the dynamic change of lossy rate of wireless link.

REFERENCES

[1] S. R Li, R. W. Yeung, and N. Cai. Linear Network Coding[C]. IEEE Transactions on Information Theory, 2003, 49, pp:371-381.

[2] George C. Polyzos, George Xylomenos. Internet Protocols over Wireless Networks [M]. MULTIMEDIA COMMUNICATIONS: DIRECTIONS AND INNOVATIONS, JERRY D. GIBSON (ED.), ACADEMIC PRESS, 2000.

[3] J. K. Sundararajan, D. Shah, M. M´edard, M. Mitzenmacher, and J. Barros, “Network coding meets TCP” in Proceedings of IEEE INFOCOM, April 2009, pp. 280–288.

[4] J. K. Sundararajan, Szymon Jakubcza, M. M´edard, M. Mitzenmacher, and J. Barros, “Interfacing network coding with TCP: an implementation” in Proceedings of IEEE INFOCOM, April 2009, pp. 280–288.

[5] 4 J. K. Sundararajan, D. Shah, and M. M´edard, “ARQ for network coding,” in Proc. of IEEE International Symposium on Info. Theory (ISIT), 2008.

[6] 4Fabienne LEFEVRE, Guillaume VIBIER. Understanding TCP’s behavior over wireless links [C]. Proc. IEEE Symposium on Computers and Communications, June 2000.

[7] 8R Ahlswede, N Cai, S Y R Li, et al. Network information [J]. IEEE Trans on Information Theory, 2000, 46(4), pp:1204-1216.

[8] 9R K Ahuja, T L Magnanti, J B Orilin. Network Flows: Theory, Algorithms, and Applications [M]. Englewood Cliffs, NJ: Prentice Hall, 1993.

[9] 10Ho T, Karger D, Medard M, et al. The benefits of coding over routing in a randomized setting [C]. Yokohama, Japan: IEEE International Symposium on Information Theory, 2003, pp: 442.

[10] 14Gkantsidis C, Rodriguez P R. Network coding for large scale content distribution [Z]. Microsoft Research, 2004.

[11] W. Richard Stevens. TCP/IP Illustrated Volume 1:The protocol. 2000.