Network Codes-based Multi-Source Transmission Control Protocol for Content-Centric Networks

Dongliang Xie[2,1], Xin Wang[1], Qingtao Wang[2]
[1]Department of Electrical and Computer Engineering, State University of New York at Stony Brook, USA
[2]State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, China
dongliang.xie@gmail.com, xwang@ece.sunysb.edu, poseidon_wang@bupt.edu.cn

***Abstract*—With the rapid shift from end-to-end communications to content-based data retrieval, there are increasing interests in exploiting Content-centric Networks (CCN) to deliver data. As the special characteristics of CCN, in-network caching and naming-based routing make traditional TCP-like transmission control protocol unsuitable. Although there are some existing efforts on improving the congestion control in CCN, the big issue of redundant transmissions caused by multiple sources has received little attention.**

To eliminate the redundancy and speed up the transmission, we propose a complete Network Codes-based Multi-Source Transmission Control Protocol (MSTCP), which provides an efficient and controllable multi-source content retrieval service over CCN. MSTCP takes advantage of random network coding to make full use of the coded data responded by different sources to speed up decoding and data receiving at the request side. Moreover, we design a scheduling algorithm based on a simple Expected Reception Deadline (ERD) to efficiently control the number of coded packets to send at each source. This not only effectively eliminates the redundant transmissions in CCN, but also helps to significantly speed up the information retrieval. Extensive simulations show that our mechanism greatly reduces the redundancy while speeding up the content retrievals by the network users.

I. Introduction

Internet usage has shifted from the host-centric end-to-end communication to information-based content retrieval. In order to adapt to this change, CCN have emerged. CCN decouples the location from the identity, so that the data content can be retrieved based on its name.CCN has attracted a lot of research interests in recent years. With the use of ubiquitous caching, certain piece of content may be retrieved from multiple sources. Similar to P2P network, in CCN, a node can obtain the same content from multiple nodes, and the sources can be uncertain due to the dynamic joining and departure of peer nodes. These features make the traditional TCP no longer suitable.

In light of the problems above, existing efforts mainly focus on extending the congestion control scheme in TCP to work in CCN [1][2][3][4][5]. Specifically, ICP identifies congestion based on out-of-order transmissions and timer expiration, while ICTP and CCTCP infer congestion only based on the timer expiration. Additionally, ICTP packets chunks fragmented for transmission but perform security and caching operation at the chunk level. Contug keeps multiple timeout timers and windows per flow, however, its assumption on the knowledge of content location of each chunk before transmission is infeasible in CCN. HoBHIS[6] and HR-ICP[7] control congestion hop-by-hop and maintain per-flow state on each CCN router, which would strongly affect the scalability of CCN and deployment in core Internet. Most of the above congestion control schemes fail to consider the uncertainty of sources and the data redundancy. The redundant transmissions will seriously overload the CCN network.

The two most significant features of CCN is routing-by-name and universal caching, allow for better delivery efficiency and disruption tolerance. but these features can result in the content responses from multiple sources. Generally, only the first returned content is used for quick retrieval. The redundant copies will consume a amount of network bandwidth, but this issue receives little attention.

As a parallel technique, network coding (NC) [8]has attracted much attention. The major benefit of network coding stems from its ability of integrating data across time and flows to form the coded data. As encoded packets have equal importance,as long as enough uncorrelated packets are received, the original data can be recovered. A number of solutions using network coding in current Internet architecture. However, there is very limited work in applying network coding in CCN.

To the best of our knowledge, we are the first to propose a transport protocol based on the network coding to solve the data redundancy and source uncontrollability of CCN. Our main contributions can be summarized as follows:

- We propose a novel and complete MSTCP protocol which exploit the network coding to improve the CCN controllability and speed up the content retrieval taking advantage of multi-source.
- We design an efficient scheduling algorithm which leverage a reference parameter ERD to reduce the redundant data transmissions from multiple sources.

The rest of this paper is organized as follows. We discuss our network coding design for CCN in Section II. We describe our proposed protocol and the scheduling algorithm in

Section III. Finally, we evaluate the performance of MSTCP in Section IV and conclude the work in Section V.

II. Network coding for Content Centric Network

In this section, we first introduce the basic CCN architecture to illustrate its transporting process and problem, and the network coding in CCN.

A. CCN Network Model

Different from the current end-to-end IP network, the special characteristics of CCN such as in-network caching and name-based routing help to decouple transmissions from senders and receivers. To access a piece of content, called chunk, a user sends an interest to the network. The router then forwards the interest based on the content name to the sources indicated in the Forwarding Interest Base (FIB) and records the interest into the Pending Interest Table (PIT). Because contents can be cached or stored anywhere, the requested data chunk may be returned from multiple sources denoted as a set S=$S_1, S_2, \ldots, S_n$. A source segments the chunk into k packets and send them along the reverse path of the interest to the users. When multiple chunks returned from different sources arrive at the router, only the first arriving one will be accepted, and the remaining ones are all discarded.

In order to effectively control the redundant transmissions and better adapt to the other characteristics of CCN, we introduce random network coding into CCN so that randomly coded packets sent from different sources can be effectively reduced. We intend the changes of the CCN protocol stack to incorporate network coding. Figure 1 shows the network model for network coding in CCN.Then two key problems need to be addressed: the selection of the coding method and the determination of the chunk size.

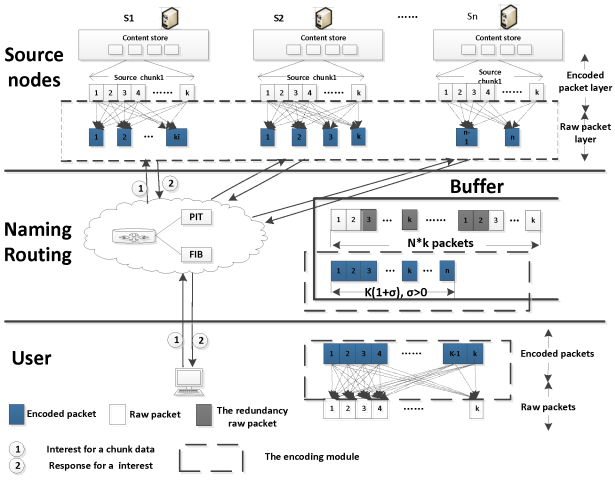


Fig. 1: Network model for Network Coding in CCN

B. Coding for CCN

RLNC [9] is a popular coding scheme which is more robust to the joining and departure of source nodes and implemented distributedly. For RLNC to work, in order to reliably transmit k_b symbols, it needs to generate a block that contains $(1+\beta)k_b$ coded symbols. A requester can recover the raw symbols once obtaining any k_b linearly independent encoded symbols.

C. Determination of Chunk Size

Chunk is the basic data unit of caching and security in CCN. If the chunk is too small, signature and authentication for each chunk will cause high computation and transmission overhead. On the other hand, too large a chunk will cause cache oscillation, if large data blocks need to be frequently inserted or deleted from a node that has limited caching space. In addition, only a part of chunk is lost or received in errors, retransmission of the whole chunk will be triggered.

With the quick reduction of storage cost, the cache size is generally not a major concern. Thus a more realistic option is to increase the chunk size. The default chunk size in CCNx [10] is set to 4 KB. The work in [1] shows the throughput improves when the chunk size changes from 4k bytes to 32k bytes. Thus increasing the chunk size is the trend of the future. And the default chunk size will be given in the simulation section.

III. Protocol Design

In this section, we first introduce the protocol architecture of MSTCP, and then propose an effective multi-source scheduling algorithm.

A. Protocol Architecture of MSTCP

In the protocol architecture of MSTCP shown in Figure 2, application nodes are divided into two sides: interest requesters and content sources. Different from conventional TCP end-to-end transmissions, CCN transmissions are pulled by interest requesters. In MSTCP, there are mainly several parts on the request side, which can be divided into three function modules: *Scheduling Control*, *Congestion Control*, and *Decoding*.

As a core of MSTCP, scheduling control module consists of network *monitoring unit*, *source prediction unit*, *parameter control unit* and *interest scheduling unit*. The main function of this module is to determine a reference control parameter, ERD, based on the information from the received packets. This parameter serves as a reference for each source to *independently* determine its number of encoded packets to transmit. Once receiving a coded packet, the *network monitoring unit* extracts the information. These are sent to the *source prediction unit* and the *parameter control unit*. We could estimate the source information from the feedback of the first round of transmissions in a fully distributed manner. With repetitive chunk requests from different users, chunks may be cached in different parts of the CCN network. And some continuous chunks for a file may be cached at the same

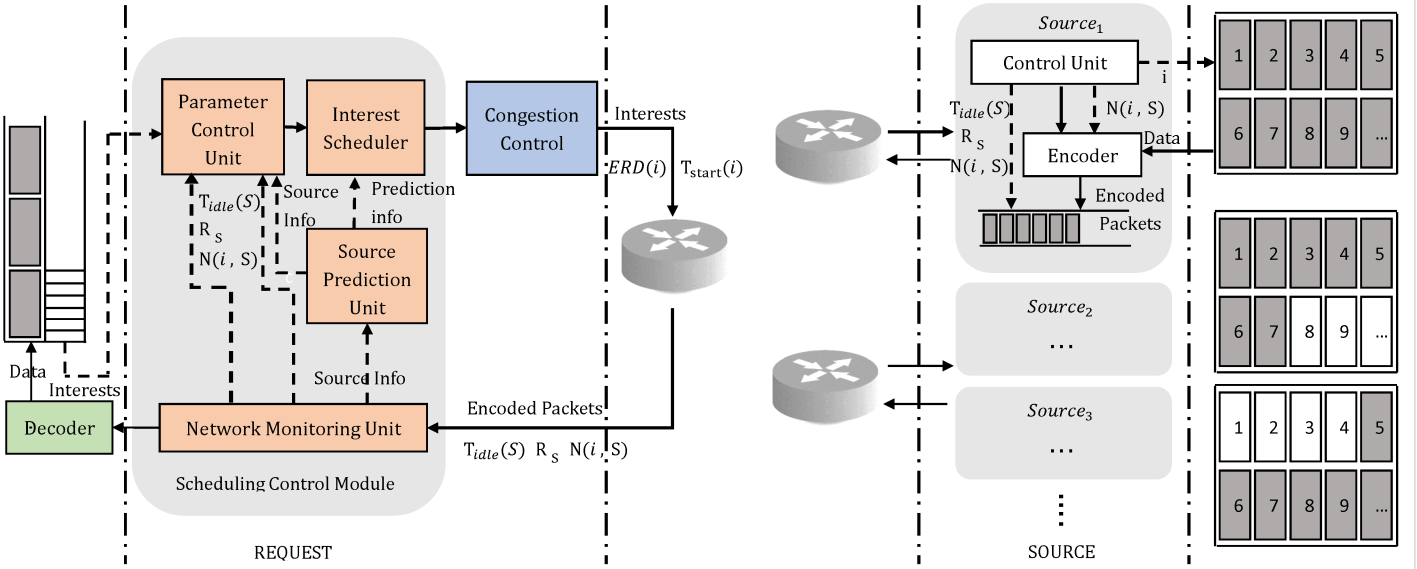


Fig. 2: MSTCP Architecture

node. Thus we can predict the source of next chunk from the feedback of a source.In case of the prediction error and data loss, we make sure the number of coded packets received is sufficient to recover a chunk.

The main function of Congestion control module is to adjust the window size according to the congestion level of the network, so as to make full use of the network bandwidth. With the use of network coding, the out-of-order transmission is no longer a problem in MSTCP. This allows us to take the same congestion control scheme as that of the traditional TCP, i.e., by using the timeout as the indicator of congestion, so that MSTCP can work friendly with the traditional TCP. On the other hand, since the congestion control module is separate, MSTCP can also be compatible with other CCN protocols. This Decoding module is responsible for recovering the requested chunk from the set of encoded packets received.

B. Multi-source Scheduling

In CCN, an interest packet will be sent to retrieve a content chunk, which may be received by multiple sources. It will consume a lot of bandwidth resources in the CCN network. In order to alleviate the redundancy, we propose a scheduling algorithm to facilitate cooperative transmissions from multiple sources.

1) Scheduling Overview and Design Considerations: There are many uncertain factors and information limitation in the CCN network. A source generally has no knowledge of other sources, which makes it difficult for multiple sources to coordinate their transmissions, while a router only knows which interface to forward an interest or an encoded packet. If an interest is broadcast in CCN, the same interest packet of a chunk will reach all the corresponding content sources.

In order to ensure better transmission control, our scheduling algorithm is designed with the following considerations:

- *Reducing the redundancy*: In the multi-source scenario, the use of network coding do not need differentiate coded packets from different sources. This will alleviate the problem of transmission redundancy. Taking advantage of the network coding, our scheduling algorithm can control the total transmissions so that the randomly coded packets from all sources cooperatively contribute to the decoding of the requested content chunk.
- *Source selection with low chunk transmission delay:* The scheduling algorithm should make the right sources to send appropriate number of encoded packets and ensure the total number of packets received to be enough for decoding to reduce the chance of receiving timeout.

Generally, transmissions in CCN are carried between a content requester and some uncertain sources. In our scheduling, we control the total number of encoded packets from all possible sources through a reference parameter, and we expect a content requester can receive enough linearly independent packets to successfully decode all packets in the requested data chunk.

2) Determination of Reference Control Parameter: As discussed earlier, we propose a receiver-driven mechanism to control the number of encoded packets sent by individual sources. The determination of the reference parameter is critical for the control efficiency.

Definition 1. *Expected Reception Deadline* (ERD) *is defined as the expected time instant that the CCN interest request side receives enough encoded packets to decode a chunk.*

ERD is sent by the content requester to all possible sources along with the interest broadcast. For a chunk i, a source should stop sending packets if they couldn't arrive at

the request side before $ERD(i)$. In order to achieve this, each source calculates the number of encoded packets to send based on $ERD(i)$ received and its own transmission parameters, such as RTT, loss rate, sending rate, and its existing tasks.

In the following, we introduce our method for deriving $ERD(i)$. We use $T_{start}(i)$ to represent the time instant that the request side sends the interest, and $RTT_S/2$ as the approximate transmission delay of each source to the request side. As the paths between different sources and the request side are different, the interest may arrive at each source S at a different time instant $T_{receive}(i,S)$:

$$T_{receive}(i,S) = T_{start}(i) + RTT_S/2. \quad (1)$$

Intuitively sources will send back the encoded packets upon receiving the interest. However, some sources may be occupied by other tasks and can only start transmissions of packets of the requested chunk after $T_{idle}(S)$. The time instant $T_{send}(i,S)$ is thus

$$T_{send}(i,S) = max(T_{idle}(S), T_{receive}(i,S)). \quad (2)$$

To avoid unnecessary transmissions, a source S needs to stop sending packets if they cannot arrive at the requester before $ERD(i)$. The time instant $T_{stop}(i,S)$ is estimated as

$$T_{stop}(i,s) = ERD(i) - RTT_S/2. \quad (3)$$

For the sake of packets arriving at the requester in time for decoding, the sending time $T_{send}(i,S)$ can not be later than $T_{stop}(i,S)$. The number of encoded packets to send by a source S with the sending rate R_s can be estimated as:

$$\begin{aligned} N(i,S) = max[(T_{stop}(i,S) - T_{send}(i,S)), 0] \times R_S \\ /Size_{seg} = max[(ERD(i) - RTT_S/2 - max(\\ T_{idle}(S), (T_{start}(i) + RTT_S/2)), 0] \times R_s/Size_{seg} \end{aligned} \quad (4)$$

where $Size_{seg}$ stands for the size of encoded packet.

The total number of packet received for chunk i can be represented as $N(i) = \sum N(i,S)$, and $ERD(i)$ can be derived based on the total number of packets needed at the requester side. As the above calculations are based on the parameters feedbacked by different sources, some parameters may change over time which makes the estimated $ERD(i)$ deviate from the correct one.

Some encoded packets may get lost, and each path may have a different loss rate. The total number of encoded packets received at the request side for $chunk\ i$ is

$$N(i) = \sum N(i,S) \times (1 - p_S) \quad (5)$$

where p_S represents the lost rate on the path from source S to the request side. p_S can be calculated by the number of encoded packets sent by $source_S$ and the actual number of encoded packets received by the request side.

Figure 3 shows parameters of different sources under different conditions. We will describe our scheduling algorithm in the following section.

3) Scheduling Algorithm: Our scheduling algorithm needs the coordinative actions from both the request side and the source side.

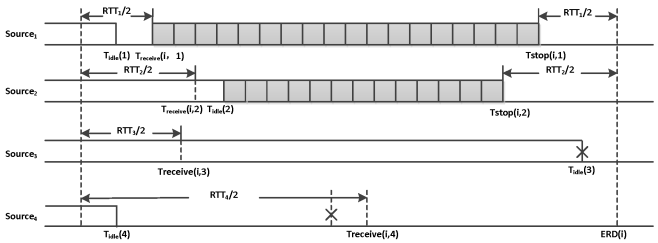


Fig. 3: Time instants of the scheduling algorithm

*a) **The request side**:* To calculate $ERD(i)$ for a chunk i, the request side needs to collect and estimate the parameters of sources. When a new encoded packet is received from source S, the request side obtains the parameters $T_{idle}(S)$, $N(i,S)$, R_S, and it can further calculate some other parameters needed to determine $ERD(i)$. When the request side wants to send an interest packet for a new content chunk, based on $T_{start}(i)$ and $RTT_S/2$, the request side estimates the time instant $T_{receive}(i,S)$ for the interest to arrive at the source based on Eq. (1).

Algorithm 1 Actions at Requester for Requested Chunk i

```
INPUT: T_idle(S), p_S, R_S, T_start(i), RTT_S/2;
for S = 1 to n do
    if Source_S ∈ Sources which hold the content then
        T_receive(i,S) ← T_start(i) + RTT_S/2
        estimate T_send(i,S) using Eq.(2)
    end if
end for
calculate ERD(i) using Eq.(3) and Eq.(4)
```

The request side can further estimate $T_{send}(i,S)$ using Eq.(2). Based on these information, a content requester further estimates the total number of encoded packets to be sent by all sources, $N(i)$. Then the request side inserts $ERD(i)$ and $T_{start}(i)$ into the interest packet of $chunk\ i$ before sending it to the CCN network.

Algorithm 2 Actions at Source S for Requested Chunk i

```
Abstract T_start(i), ERD(i) from the interest packet;
Calculate T_stop(i,S) using Eq.(3);
Calculate T_send(i,S) using Eq.(2)
if T_send(i,S) > T_stop(i,S) then
    Drop the interest;
else
    Calculate N(i,S) using Eq.(4)
end if
Send N(i,S) encoded packets to the request side;
```

*b) **The source side**:* A source first determines the number $N(i,S)$ of the encoded packets to send for $chunk\ i$ based on $ERD(i)$. It then sends the packets along with its local parameters that are needed for the requester to estimate $ERD(i)$, $T_{start}(i)$ and the sequence number of the coded packet.

To make sure that packets transmitted in the CCN network are useful for the request side to decode, not all the sources holding the requested data should send back the encoded packets. In our scheme, a source makes the decision in four different cases, as shown in Figure 3.

Case 1: The sources are idle and close enough to the request side. $Source_1$ first calculates $T_{stop}(i,S)$ using $Eq.(3)$ when the interest arrives with the information $ERD(i)$. If $T_{receive}(i,S)$ is later than $T_{idle}(S)$, $Source_1$ will start to send encoded packets at $T_{receive}(i,S)$. Then we can calculate $N(i,S)$ using $Eq.(4)$.

The source encodes the data of chunk i, and sends $N(i,S)$ encoded packets to the request side along with the parameters $T_{idle}(S)$, $N(i,S)$, and R_S. Then source sends the encoded packets to the queue to be waited to send.

Case 2: Like $Source_2$, the source is occupied but is close enough to the request side. In this case, $T_{idle}(S)$ is later than $T_{receive}(i,S)$, so the source can only start to send its encoded packets at $T_{idle}(S)$.

Case 3: Like $Source_3$, the source is occupied by too many previous tasks. When a source ends its former tasks, it would be too late to send the encoded packets for chunk i.

Case 4: Like $Source_4$, the source is too far away from the request side. In this case, it is also impossible to send encoded packets.

By using this Scheduling Algorithm, source selection can be performed indirectly and determined by the source itself. With our algorithm, the total number of encoded packets only a little bit higher than needed to ensure timely decoding of the chunk by the request side. Therefore, MSTCP can minimize the redundant transmissions while speeding up the information retrieval.

IV. Simulation

We evaluate the performance of MSTCP using ndnSIM over NS-3 network simulator[11]. We setup with one application user and two content sources as in Figure 4. More sophisticated network setting will be studied for the impact of various factors on our scheduling algorithm. All communication links are set to 50 Mbps with 10 ms delay while the maximum sending rate of each source is 8Mbps and the size of the transfer file is set to 10MB. The simulations are carried out in two parts. First, the chunk size discussed in Section II is evaluated. Second, we compare the performance of our proposed MSTCP with ICTP.

A. Impact of Different Chunk Sizes

As shown in Figure 5(a), the goodput increases until the encoded chunk size reaches 32k. As the chunk size grows, the overhead due to the signature and authentication becomes smaller, and the number of interests for retrieving the content reduces considerably. However when the chunk size grows to a certain level, the benefits reduce while the delay of encoding and decoding increases significantly. As a result, the goodput decrease dramatically when the chunk size increases beyond 64k.

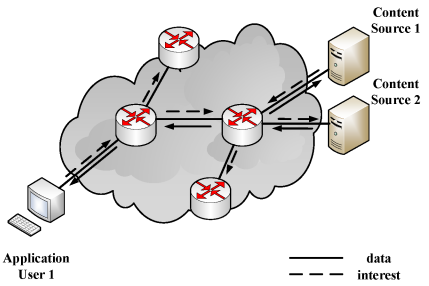
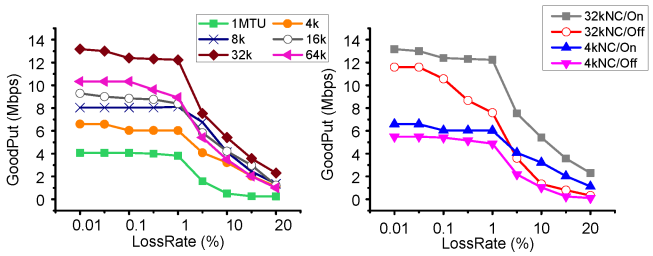


Fig. 4: Network topology



(a) Goodput with different chunk size

(b) Goodput with network coding on and off

Fig. 5: Total goodput of MSTCP with various chunk size

In Figure 5(b), the goodput performs better when the network coding is used. Without differentiating between coded data packets from different sources, upon packet loss, MSTCP only requires sources to retransmit additional number of coded packets instead of the whole chunk. The data can be decoded successfully as soon as the receiver retrieves enough coded packets from different sources. Thus, MSTCP takes advantage of multi-source transmission to improve the transmission reliability while mitigating the transmission overhead.

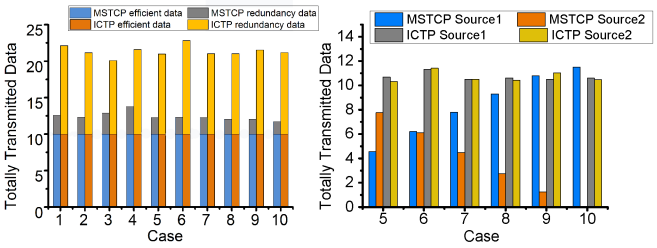
B. Transmission Efficiency

We further compare the performance between MSTCP and ICTP on the transmission redundancy and goodput in different scenarios. We set the loss rate in TABLE I. The redundancy is reflected from the total amount of data transmitted.

TABLE I: Cases of simulation

Test Case	1	2	3	4	5	6	7	8	9	10
Loss Rate (%)	0.1	1	10	20	1	1	1	1	1	1
Delay (ms)	10	10	10	10	5	10	15	20	25	30

Figure 6(a) shows the total amount of data transmitted by MSTCP and ICTP respectively, both of which are composed of the effective data and the redundant data. MSTCP is shown to have significant lower redundant transmission compared to that of ICTP in all cases. The redundant transmissions from ICTP can be five times that of MSTCP. In ICTP,

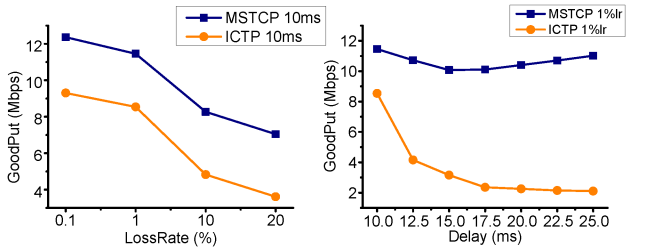


(a) Total redundancy with various link quality

(b) Redundancy of each source with various link quality

Fig. 6: Redundancy of MSTCP and ICTP

each data source receiving the interest will send a response, which causes a large number of redundant transmissions in the CCN network. Taking advantage of network coding, our proposed scheduling algorithm facilitate the the coordinative transmissions from all available sources to reach efficiently response the request. To better analyze the redundancy of MSTCP and ICTP, Figure 6(b) shows the amount of data sent by each source. Compared with ICTP, the data transmitted by the two sources in MSTCP vary in different cases which have different link conditions. The transmissions from $source_2$ reduce as its delay increases. When the delay surges to 30ms, the $source_2$ is prevented from sending packets completely. This indicates that our scheduling algorithm can adaptively select the better source to transmit packets according to the network condition.



(a) Goodput with different LossRate

(b) Goodput with different Delay

Fig. 7: Goodput of MSTCP and ICTP

In Figure 7(a), the goodput of MSTCP outperforms that of ICTP in all the loss rates studied, and the gain increases as the loss rate becomes larger. When the loss rate reaches 20%, the goodput of MSTCP is more than triple that of ICTP. With the incorporation of random network coding, MSTCP effectively alleviates the impact of packet loss while making full use of coded packets received. Figure 7(b) shows that MSTCP also has better performance on goodput while varying the delay. When the delay of $source_2$ increases, the out-of-order receiving in ICTP becomes serious and only the better source can send back the requested data in time. However, with the introduction of random network coding, MSTCP can make better use of packets received from all sources regardless of their order. In the worst delay case, the goodput of MSTCP is almost 5 times that of ICTP.

V. CONCLUSION

In this paper, we propose MSTCP, a novel complete transmission control protocol based on random network coding to effectively address the problem of redundant transmission in CCN caused by name-based routing and in-network caching. We employ RLNC to encode the chunks of contents in CCN, while MSTCP can work with other types of random network coding scheme. We propose an efficient scheduling algorithm which provides a simple reference control parameter from the request side to facilitate distributed determination of the number of packets to transmit by each resource node. Our proposed MSTCP not only effectively eliminate the transmission redundancy from multiple sources but also take advantage of network coding to fully utilize the coded packets from all responding sources to significantly speed up the information retrieval.

VI. ACKNOWLEDGEMENT

This work was supported in part by National Science Foundation (NSF) under grant CNS- 1218597, CNS-1526425, and CNS-1421578China 863 Program 2015AA01A705, and NSFC of China 61271185.

REFERENCES

[1] Stefano Salsano, Andrea Detti, Matteo Cancellieri, Matteo Pomposini, and Nicola Blefari-Melazzi. Transport-layer issues in information centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 19–24. ACM, 2012.

[2] Giovanna Carofiglio, Massimo Gallo, and Luca Muscariello. Icp: Design and evaluation of an interest control protocol for content-centric networking. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 304–309. IEEE, 2012.

[3] Somaya Arianfar, L Eggert, P Nikander, Jörg Ott, and Walter Wong. Contug: A receiver-driven transport protocol for content-centric networks. *Under submission*, 2010.

[4] Lorenzo Saino, Cosmin Cocora, and George Pavlou. Cctcp: A scalable receiver-driven congestion control protocol for content centric networking. In *Communications (ICC), 2013 IEEE International Conference on*, pages 3775–3780. IEEE, 2013.

[5] Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, Michele Papalini, and Sen Wang. Optimal multipath congestion control and request forwarding in information-centric networks. In *ICNP*, pages 1–10, 2013.

[6] Natalya Rozhnova and Serge Fdida. An effective hop-by-hop interest shaping mechanism for ccn communications. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 322–327. IEEE, 2012.

[7] Giovanna Carofiglio, Massimo Gallo, and Luca Muscariello. Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 37–42. ACM, 2012.

[8] Tracey Ho, Ralf Koetter, Muriel Medard, David R Karger, and Michelle Effros. The benefits of coding over routing in a randomized setting. 2003.

[9] S-YR Li, Raymond W Yeung, and Ning Cai. Linear network coding. *Information Theory, IEEE Transactions on*, 49(2):371–381, 2003.

[10] Ccnx project. http://www.ccnx.org.

[11] Alexander Afanasyev, Ilya Moiseenko, Lixia Zhang, et al. ndnsim: Ndn simulator for ns-3. *Named Data Networking (NDN) Project, Tech. Rep. NDN-0005, Rev*, 2, 2012.