Network Coding Based Multipath TCP

Ming Li*, Andrey Lukyanenko*, Yong Cui†

*Department of Computer Science and Engineering, Aalto University, Finland
Email: {*ming.li, *andrey.lukyanenko}@aalto.fi
†Department of Computer Science and Technology, Tsinghua Unversity, China
Email: †cuiyong@tsinghua.edu.cn

***Abstract*—Multipath TCP (MPTCP) suffers from the degradation of goodput in the presence of diverse network conditions on the available subflows. The goodput can even be worse than that of one regular TCP, undermining the advantage gained by using multipath transfer.**

In this work, we propose a new multipath TCP protocol, namely NC-MPTCP, which introduces network coding (NC) to some but not all subflows traveling from source to destination. At the core of our scheme is the mixed use of regular and NC subflows. Thus, the regular subflows deliver original data while the NC subflows deliver linear combinations of the original data. The idea is to take advantage of the redundant NC data to compensate for the lost or delayed data in order to avoid receive buffer becoming full. We design a packet scheduling algorithm and a redundancy estimation algorithm to allocate data among different subflows in order to optimize the overall goodput. We also give a guideline on how to choose the NC subflows among the available subflows. We evaluate the performance of NC-MPTCP through a NS-3 network simulator. The experiments show that NC-MPTCP achieves higher goodput compared to MPTCP in the presence of different subflow qualities. And in the worst case, the performance of NC-MPTCP is close to that of one regular TCP.

I. Introduction

In recent years, the emergence of multi-interface (LAN, WiFi, 3G, WMAN, etc) devices makes ubiquitous access to the Internet possible. Users who access to the network resources anywhere and at any time also expect to take advantage of these multi-homing devices to improve the end-to-end communication performance and resilience by using several interfaces simultaneously.

To realize this expectation, both the research community and the Internet Engineering Task Force (IETF) have devoted much attention to multipath transmission during recent years. Network layer solutions such as shim6 [11] and HIP [9] have been proposed. However, they remain mainly as proof-of-concept implementations: to date, they have not been widely deployed, nor are they likely to be in the near future. Transport layer solutions have also been proposed, such as SCTP [6], [7] and some proposals for wireless networks [3], [4], [12]. However, they cannot be deployed on the Internet in practice due to either the compatibility or scalability issues.

During the last few years, the MPTCP (Multipath TCP) [1], [2], [15] working group of IETF has advocated the development of multipath extension of the traditional TCP by enabling multihomed devices to use several paths through multiple interfaces within a single TCP connection. From the performance viewpoint, MPTCP aims to improve the overall goodput by using concurrent data transmission over multiple paths. However, if the receive buffer size is finite, this requirement will incur a head-of-line blocking problem at the receiver in cases when the paths have vastly different bandwidth-delay products. Specifically, the receiver needs to buffer connection-level out-of-order segments until the lost or delayed segments arrive before forwarding them up to the upper layer and then releasing the receive buffer. Once the receive buffer becomes full because of lacking a head segment in the receive queue, the receiver will advertise its zero window to the sender. Therefore, the sender will enter the persist mode resulting in great degradation of the overall goodput. Thus, the poor quality paths may become bottlenecks to degrade the overall performance.

Head-of-line blocking is not limited to MPTCP but is a common problem for all multipath transfer schemes. In this paper, we propose a network coding based multipath TCP protocol (NC-MPTCP) to solve this problem. The main feature of our protocol is the introduction of network coding to some but not all subflows. The essential principle of this scheme is the mixed use of regular subflows and NC subflows, i.e., regular subflows transmitting original data whereas NC subflows transmitting linear combinations of original data. The benefit of our protocol comes from the fact that network coding can compensate for the lost or delayed head-of-line segments by utilizing redundant data to avoid retransmission.

A main requirement of MPTCP is that it is deployable in the current Internet without the need for changing routers and middle boxes. A consequence of this requirement is that the encoding/decoding operations incurred by NC are performed only at the end hosts. The intermediate nodes do not need to know about NC. Our main contributions are:

- Introducing network coding into a multipath TCP to compensate for the lost or delayed head-of-line packets to avoid receive buffer becoming full.
- Designing an efficient packet scheduling algorithm and a redundancy estimation algorithm.
- Showing the performance gain of the new protocol compared with MPTCP over heterogeneous path conditions.

The rest of this paper is organized as follows. Section II discusses the background and makes reference to related work. Section III looks at the architecture of our NC-MPTCP

978-1-4673-1017-8/12/$31.00 ©2012 IEEE

protocol, and Section IV describes the packet scheduling algorithm and the redundancy estimation algorithm. Section V presents our evaluation, in which we show our implementation and present the simulation results. Our conclusions as well as future work are discussed in Section VI.

II. Background and Related work

A. Problem Statement

The initial motivation of multipath transmission is to use multiple available paths to deliver data to destination. The expectation is to improve the overall goodput and reliability. However, one observation on the performance of multipath transfer is that when the receive buffer capacity is limited, the performance suffers from the degradation of the overall goodput in presence of diverse network conditions on different subflows. The overall goodput can even be worse than that of a single TCP, undermining the advantage of multipath transfer. The reason behind this phenomenon is the head-of-line blocking problem, i.e., a fast path can be blocked due to the receive buffer becoming full resulting in the overall goodput being capped at that of the slowest subflow.

B. Multipath TCP

Multipath communication is not new. Several protocols and their extensions have been proposed to support multipath transportation. Stream Control Transmission Protocol (SCTP) [13] has been designed with multi-homing support. Based on SCTP, several extensions [6], [7] enable hosts to use concurrent multipath transfer. To solve the head-of-line blocking problem, [6], [8] proposed several retransmission policies. However, these policies can only mitigate the problem caused by spurious retransmission, which is only a small part of the cause resulting in receive buffer blocking. Furthermore, the policies they employed did not take bandwidth and packet loss factors into consideration. Thus, the research on the receive buffer blocking problem based on SCTP is insufficient.

[4] proposed to allow the applications provide large aggregate buffer to handle the out-of-order problem incurred by multipath transmission. However, it did not solve the problem itself but just moved the problem from transport layer to application layer.

To solve the receive buffer blocking problem of multipath transfer in heterogeneous networks, some coding based techniques have been proposed. [12] demonstrated that using erasure codes can realize improved bandwidth gains through multiple heterogeneous, highly lossy paths. [16] proposed to use network coding techniques to minimize reordering and timeouts at the receivers. However, these papers still lacked in-depth analysis on the impact of receive buffer on the performance of multipath transfer. Moreover, these coding based solutions were designed to be used in wireless mesh networks instead of Internet. In wireless mesh networks, much information can be provided by utilizing the broadcast nature of the wireless networks. For example, a node can estimate with reasonable probability whether its neighboring nodes have received its out-sent packets or not by overhearing the transmission of its neighboring nodes. However, on Internet the nodes can not get these information freely. Due to the scalability problem, these coding based solutions cannot be extended to be used on Internet easily.

MPTCP [1], [2], [15] has been developed to simultaneously transmit data over multiple subflows in a transparent manner to application and IP layer. One objective of MPTCP is to improve the goodput by bandwidth aggregation over multiple available paths. This objective is still a challenge when different paths have different characteristics in terms of bandwidth, delay, and packet loss. [10] evaluated the performance of MPTCP in terms of throughput optimization. The measurements showed that the context of heterogeneous networks (Ethernet, Wifi, and 3G) had a great impact on the MPTCP performance due to the receive buffer blocking problem. Therefore, some intelligent packet scheduling algorithms for interface selection in MPTCP is needed.

C. Network Coding

In traditional store and forward networks, data is transmitted in packets and forwarded from one node to another. Contrary to this standard network paradigm, network coding allows nodes to encode multiple packets together to generate a new one. Sink nodes can recover the original messages when they collect enough linearly independent coded packets.

TCP is designed to provide a reliable, byte stream service. To achieve this goal, TCP uses feedback to acknowledge received segments arriving in order. The concept of sequence order is critical for several sub functions of TCP, such as reordering at the receiver and feedback signal for the congestion control loop. When introducing network coding into MPTCP, we face a few challenges. First, the concept of an ordered sequence of byte stream used by TCP is missing. We need to find a method to reorder the received segments. Secondly, a TCP sink node may receive a linear combination of some original segments but it cannot reveal any original segment immediately before it accumulates enough encoded segments. Thirdly, because the concept of sequence order is missing, a mechanism to trigger the congestion control event is needed.

In recent years, some efforts [5], [14] have been made to combine TCP and network coding together. [5] showed that network coding could compensate for the lost packets to improve the throughput. [14] demonstrated that a special TCP implementation (TCP-Vegas) could incorporate network coding with minor changes in the current protocol stack. However, they all focuse on a single connection TCP. While in this paper, our aim is to integrate network coding into multipath TCP to boost the overall goodput.

III. Architecture

In this section, we present the architecture of NC-MPTCP, a new multipath TCP protocol aiming to improve the overall goodput by mitigating the head-of-line problem with the help of network coding techniques.

The architecture of our protocol is shown in Figure 1 where it consists of three types of entities. The first type is **Main**

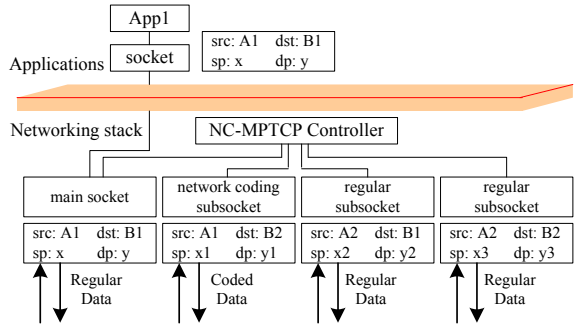

Fig. 1. Network Coding Assisted MPTCP

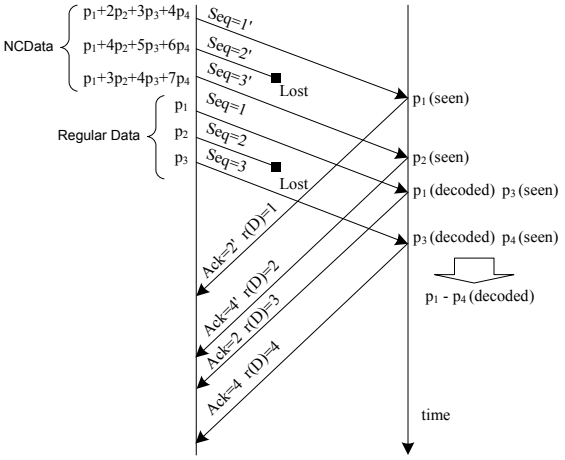

Fig. 2. Example of coding and ACK

Socket (MS), a standard socket created by an application. It runs on a regular TCP and provides standard interfaces between the application and the kernel for TCP connection. If the peer does not support multipath TCP, only the main socket will be created and executed. The second type entity is **Network Coding Subsocket and Regular Subsocket**. Each of them is also a standard TCP socket but running on a network coding based TCP and a regular TCP respectively. They provide interfaces between an NC-MPTCP controller and the kernel for TCP connection. In order to be backward compatible with the traditional TCP, only the main socket is known to the application. These subsockets are initiated, managed, and closed by the NC-MPTCP controller, which is the third entity. **NC-MPTCP Controller** includes several key control modules, such as scheduling module for allocating data among different subflows, redundancy estimating module for predicting how many redundant packets should be sent out to compensate for the lost or delayed data, and encoding/decoding modules for NC data.

In the following content of this section, we first introduce some definitions mainly related to the network codign theory. And then we briefly explain an example of sending and acknowledgement (ACK) procedure of a single generation.

Definition 1 (Generation)**.** *A generation is a batch of segments whose sequence numbers are continuous. Only the segments from the same generation are encoded.*

When we introduce network coding to TCP, each segment is treated as a vector over a finite field $\mathbb{F}_q^n$. A generation can be treated as a matrix. We use $r(G)$ to denote the rank of G, such that $r(G) = |G|$. Throughout the paper, we use segment and vector interchangeably.

Definition 2 (Decoding Matrix)**.** *When a sink receives vectors, it constructs a matrix for each set of linear encoded segments from the same generation. We define this matrix as a decoding matrix. Let D denote a decoding matrix and let r(D) represent its rank.*

Each decoding matrix maps to a generation. A destination uses decoding matrix transmission to decode the original segments when it accumulates enough linearly independent segments, i.e., $r(D) \geq r(G)$. In this paper, we do not distinguish between matrix and generation.

As shown in Figure 2, the sender wants to send a generation of segments through multiple subflows. We assume $|G| = 4$ and $G = \{s_1, \cdots, s_4\}$, where s_k is the k^{th} segment that the sender generates. $\sum_{k=1}^{4} \alpha_k s_k$ is a linear coded segment calculated over G. We suppose the statistic loss probability of each subflow is $1/3$. We assume that three coded segments and three original segments are sent out through NC subflow and regular subflow respectively within the same time interval. We explain each transmission by following the time line. $r(D) = 1$ because of the 1^{st} transmission. The 2^{nd} transmission is lost, and the 3^{rd} transmission causes $r(D) = 2$. When the 4^{th} transmission arrives, $r(D) = 3$. The 5^{th} transmission is lost. The 6^{th} transmission increments $r(D)$ to 4 which leads to r(D)=r(G). Then, all the packets can be decoded and forwarded up to the upper layer. Note that $r(D)$ is carried in each ACK packet to let the source know how many innovative segments have arrived at the peer.

IV. Algorithm and Model

The example shown in Figure 2 has two problems to solve: 1) How to coordinate the transmission of segments on different subflows; 2) How to determine the number of redundant segments sent through each subflow to guarantee the sink can decode the generation. There is a correlation between these two problems. In order to solve the latter, we should solve the former first, i.e., finding an efficient scheduling algorithm to allocate data among subflows. In this section, we will abstract and formulate the packet scheduling algorithm and redundancy estimation algorithm as well as some involved parameters.

A. Scheduling Algorithm

The scheduling algorithm allocates data among different subflows. It aims to coordinate the data transmission among all the subflows to optimize the overall goodput.

We use an intelligent packet selection strategy to choose packets from the shared send buffer to individual subflows.

Specifically, when the sender receives a new ACK and is allowed to send extra packets over an individual subflow according to the congestion window. The scheduling algorithm picks a packet based on subflow condition parameters (end-to-end delay, packet loss rate, and packet sending speed). We use p and R to represent the estimated packet loss rate and round trip time (RTT). Let MSS denote the minimum value of the maximum segment sizes of all the subflows. We use the following equation to estimate the transfer rate:

$$X \approx \frac{MSS}{R\sqrt{p}} \tag{1}$$

We assume the sequence numbers of the first unsent packet and the selected packet in the shared send buffer are S_f and S_s respectively. p_i and R_i represent the estimated packet loss rate and round trip time (RTT) on the ith subflow. Then we can get the following equation.

$$S_s = S_f + \sum_{R_j > R_i} X_j * (R_j - R_i)/2 \tag{2}$$

We take (1) into (2), such that

$$S_s = S_f + \sum_{R_j > R_i} \frac{MSS}{R_j\sqrt{p_j}} * (R_j - R_i)/2 \tag{3}$$

B. Redundancy Estimation

Consider a generalized example of Figure 2. Let G denote a generation, C be the set of coded segments calculated over G, and D represent the corresponding decoding matrix. We use p_r and p_n to represent the statistic packet loss probability on regular subflows and NC subflows respectively. By using the packet scheduling algorithm described in section IV-A, different subsockets can finish the transmission of the segments from the same generation within the same time interval. The NC subsockets deliver the set of C and the regular subsockets deliver a subset of G. We use G' to denote this subset, such that $G' \subseteq G$ and $|G'| \leq |G|$. Now the problem becomes as how to determine the size of C and G' to ensure that the destination can decode the generation with the constraints of the average packet loss probability p_r and p_n on regular subflows and NC subflows respectively.

Theorem 1. *The decoding matrix D can decode the original segments of a generation G, only when the following two conditions are satisfied.*

$$|C|(1-p_n) + |G'|(1-p_r) \geq |G| \tag{4}$$

$$|C| \geq |C|p_n + |G'|p_r \tag{5}$$

Proof:

The first condition guarantees that the receiver is able to receive enough number of segments to decode the original generation. In order to avoid accidental rank reduction due to a poor random choice of encoding coefficients, we assume all the encoding coefficients are linearly independent; this guarantees that each received segment is an innovative segment. This assumption is feasible because we could either use fixed linearly independent coefficients or drop the coefficient vector which is not linearly independent on the previously generated ones. Thus, every received segment increases the rank of D by one, such that we have $r(D) = |D| = |C|(1-p_n) + |G'|(1-p_r)$. In order to decode G, the decoding matrix D must have a rank no less than $|G|$, such that (4) must be satisfied.

The second condition guarantees that the number of NC segments is enough to compensate for the lost segments on all subflows. The number of lost segments on regular subflows and NC subflows can be estimated as $|C|p_n$ and $|G'|p_r$ respectively, such that (5) must be satisfied.

■

$|G'|$ and $|C|$ are the numbers of segments sent through regular subflows and NC subflows respectively for the same generation within the same time interval. Thus, they can be modeled as the TCP transfer rates on the sending side over different subflows. Let R_n and R_r denote the average RTT on NC subflows and regular subflows respectively. We still use (1) to model the transfer rate, such that

$$\frac{|C|}{|G'|} = \frac{R_r\sqrt{p_r}}{R_n\sqrt{p_n}} \tag{6}$$

We take (6) into (4). Then, we get

$$\frac{|G'|}{|G|}\left[(1-p_r) + \frac{R_r\sqrt{p_r}}{R_n\sqrt{p_n}}(1-p_n)\right] \geq 1 \tag{7}$$

Let $r = \frac{R_r}{R_n}$. Then we have

$$\frac{|G'|}{|G|}\left[(1-p_r) + r\frac{\sqrt{p_r}}{\sqrt{p_n}}(1-p_n)\right] \geq 1 \tag{8}$$

Our objective is to get the minimum $|G'|$ and $|C|$ to satisfy Equation (8) when p_r and p_n change within reasonable range. We assume p_r and p_n change dynamically within the range of (p_r^{min}, p_r^{max}) and (p_n^{min}, p_n^{max}) respectively. Then we get the following theorem.

Theorem 2. *Let c denote* $\frac{r(1-p_n^{max})}{2\sqrt{p_r^{min}p_n^{max}}}$ *and g denote the minimum value of* $|G'|$ *which can satisfy Equation 4 and* (5)*, such that*

$$g = \begin{cases} \frac{|G|}{(1-p_r^{min}) + r\frac{\sqrt{p_r^{min}}}{\sqrt{p_n^{max}}}(1-p_n^{max})} & \text{if } c \geq 1 \\ \frac{|G|}{(1-p_r^{min}) + r\frac{\sqrt{p_r^{max}}}{\sqrt{p_n^{max}}}(1-p_n^{max})} & \text{if } c < 1 \end{cases} \tag{9}$$

Proof:

Due to the limited space, we ignore the mathematical proof here.

■

When we get $|G'|$ through Theorem 2, we can get $|C|$ by taking $|G'|$ into Equation (6).

In this section, we use estimated packet loss probability to calculate how many redundant vectors should be sent.

In practice, it is infeasible to precisely predict the loss rate because the losses are bursty. In our protocol, the knowledge of the loss distribution is used to estimate the redundancy. And the loss distribution is updated over time. However, it is still possible that the actual loss rate is underestimated in the algorithm, in which case the receiver cannot decode the packets without retransmission. To avoid this problem, we allow the ACK on fast subflows to carry the feedback of the decoding matrix status to the sender. The sender is able to decide whether to retransmit the lost packet or continue to transmit the next unsent one. Thus, the performance achieved by NC-MPTCP is no less than that of MPTCP.

C. Network Coding Subflow Selection

In this section, we give a guideline on how to choose the NC subflows among the available ones. Consider the condition in Equation (9) where $c \geq 1$ can be satisfied at most cases when $r \geq 1$. For example, we assume $p_r \in (0.0001, 0.5)$ and $p_n \in (0.0001, 0.5)$ which are the most common range of packet loss rate in practice. When we take these two values to c, we get $r \geq 0.03$, i.e.,

$$\frac{R_r}{R_n} \geq 0.03 \tag{10}$$

Thus, the equation (10) is the determination condition to select the NC subflow. However, this theoretical value does not always work in practice due to the dynamic change of the path properties. According to our simulation experience, selecting the relatively fast subflow as a NC subflow is always a better choice.

V. Evaluation

In this section we evaluate NC-MPTCP and compare its performance compared with that of TCP and MPTCP.

A. Implementation

We implemented the protocols using Network Simulator NS-3 (version 3-10)[1] and use a typical multipath transfer topology having two disjoint paths between the source and the destination. We fix the bandwidth of each path to 5 Mbps. As the task for protocols MPTCP, TCP and NC-MPTCP, we transmit a file of 21 MB from the source to the destination. We use f_1 and f_2 to represent the two subflows respectively.

For every set of parameters we consider, we repeated the measurements 100 times with global random seeds, and based on received statistics we construct 95% percent confidence intervals to guarantee the accuracy of the measurement.

B. Results

We first study the effect of packet loss rate on the goodput of MPTCP and NC-MPTCP. The RTT of both subflows is set to $120ms$. f_1 has a fixed packet loss rate of 0.01%, and the receiving buffer size is set at $385KB$ which can accommodate 282 segments each of which has 1400 bytes. The packet loss rate of f_2 is varied between 0% and 10%. Figure 3 shows the goodput of MPTCP and NC-MPTCP, as a function of the packet loss rate of f_2. On the plot we may observe that when the loss rate of f_2 grows, the goodput of MPTCP decreases much more dramatically than that of NC-MPTCP. In addition, we compare both of them against a regular TCP. When subflows have very different network conditions, almost immediately MPTCP starts to perform worse compared to regular TCP, while NC-MPTCP remains higher performance than regular TCP.

[1]http://www.nsnam.org

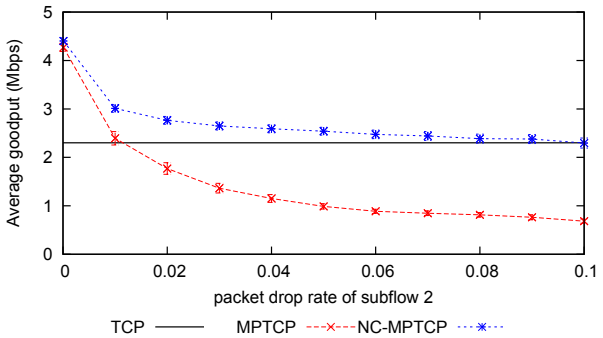

Fig. 3. Goodput comparison result with heterogeneous packet loss rate

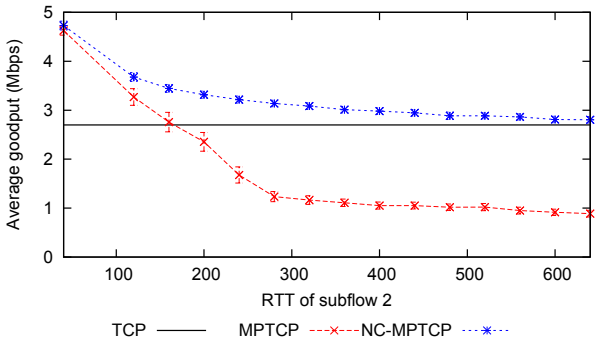

Fig. 4. Goodput comparison result with heterogeneous RTT (flow 1: 40ms)

In the second test, we study the goodput difference of MPTCP and NC-MPTCP by varying RTT deviation between f_1 and f_2. The packet loss rate of both subflows is set to 0.5%. In this test we fix the RTT of f_1 to $40ms$. The RTT of f_2 is varied between $40ms$ and $640ms$. Figure 4 shows the results that, the goodput of MPTCP falls rapidly as the RTT of f_2 increases and again as in the previous example it achieves even worse goodput compared to a regular TCP. NC-MPTCP is able to work up to three times better than MPTCP.

The above result shows that the performance of MPTCP degrades dramatically when the subflows have different network quality. As we know that MPTCP uses a connection level receive buffer as an aggregate buffer to accommodate out-of-order segments from each subflow. In the context of multipath transfer, the aggregate buffer has to be large enough to accommodate all received segments until the lost or delayed segments arrive. When the subflows have very different flow quality in terms of bandwidth, RTT, and packet loss rate, the required size of the aggregate buffer becomes too large to be possible in practice.

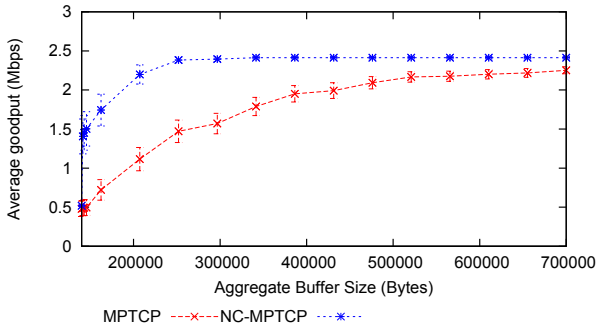

Fig. 5. Goodput comparison result with different aggregate buffer size

In the third test, we study the impact of aggregate buffer on goodput. The simulation parameters are: The RTT of both subflows are set to $120ms$, f_1 and f_2 have the fixed packet loss rate of 0.1% and 4% respectively, the default subflow-level receive buffer is set as $128K$ and the aggregate buffer size is varied between $147KB$ ($150000B$) and $684KB$ ($700000B$). The result of the test is shown in Figure 5, from which we get that MPTCP requires larger aggregation buffer size in order to achieve the stable goodput than NC-MPTCP. That is due to the NC property that any linear combination received increase the rank of matrix by one, and thus all missing gaps are very fast being filled with redundant data.

C. Discussion

The evaluation results suggest that MPTCP and NC-MPTCP behave very similarly when all subflows have similar network conditions. In this case, both protocols gain advantages over a regular TCP (one channel over the best subflow). However, the situation starts to change whenever the discrepancy between subflows starts to grow. Our simulation shows that the performance of MPTCP degrades much worse than that of only one regular TCP. Whereas under the same situation, NC-MPTCP behaves much better in terms of goodput. And in the worst case, its performance is close to that of one regular TCP. Another advantage is that the NC-MPTCP requires smaller aggregation buffer in order to achieve stable goodput. It reduces the physical requirements for the end terminals, especially in case of hand-held devices.

However, the discussed advantages appear with trade of the NC overhead in computations. End hosts have to perform higher computations in order to encode/decode the data. However, as we use network coding ideas only as end-to-end communications, both peers can agree on some predefined coding algorithms in such way that decoding becomes faster and less CPU-intensive.

VI. Conclusion

In this paper we present the architecture and algorithms of NC-MPTCP, a novel multipath TCP protocol. It utilizes network coding to boost the overall goodput in case of highly dissimilar subflow conditions. Together with designing and modeling it, we evaluate the protocol in a NS-3 network simulator by comparing the performance of NC-MPTCP, MPTCP and regular TCP. The result shows that NC-MPTCP is able to outperform MPTCP under widely diverse subflow conditions with limited aggregate buffer. And in the worst case, the performance of NC-MPTCP is close to that of one regular TCP.

Looking to the future, it would be useful to study a more sophisticated scheduler, one which can gain more goodput using a slow (in the sense of RTT) but reliable supplementary channel, in which network coding takes place. In addition, it is important to have practical implementation in order to compute the NC CPU overheads. Finally, some very sophisticated scenarios, in which a mixture of subflow conditions are met, should be discussed.

VII. Acknowledgement

This work was supported in part by the Academy of Finland (no. 135230), NSFC (60911130511, 60873252) and 973 Program of China (2009CB320501, 2009CB320503).

References

[1] S. Barré, C. Paasch, and O. Bonaventure. MultiPath TCP: From Theory to Practice. *In proc. of the 10th international IFIP TC 6 conference on Networking - Volume Part I*, pages 444–457, 2011.

[2] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar. Architectural Guidelines for Multipath TCP Development. RFC 6182, Internet Engineering Task Force, Mar. 2011.

[3] H. Hsieh and R. Sivakumar. pTCP: An end-to-end transport layer protocol for striped connections. In *In proc. of the 10th IEEE International Conference on Network Protocols*, pages 24–33, 2002.

[4] H.-Y. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. *Wireless Networks*, 11:99–114, January 2005.

[5] Y. Huang, M. Ghaderi, D. Towsley, and W. Gong. TCP performance in coded wireless mesh networks. In *In Proc. of Sensor, Mesh and Ad Hoc Communications and Networks.*, pages 179–187. IEEE, 2008.

[6] J. Iyengar, P. Amer, and R. Stewart. Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths. *In IEEE/ACM Transactions on Networking*, 14(5):951 –964, October 2006.

[7] J. Liao, J. Wang, and X. Zhu. cmpSCTP: An Extension of SCTP to Support Concurrent Multi-Path Transfer. In *Proc. of IEEE International Conference on Communications*, pages 5762 –5766, May 2008.

[8] J. Liu, H. Zou, J. Dou, and Y. Gao. Rethinking Retransmission Policy In Concurrent Multipath Transfer. In *Proc. of Intelligent Information Hiding and Multimedia Signal Processing*, pages 1005 –1008, aug. 2008.

[9] R. Moskowitz, P. Nikander, E. P. Jokela, and T. Henderson. Host Identity Protocol. RFC 5201, Internet Engineering Task Force, Apr. 2008.

[10] S. C. Nguyen, X. Zhang, T. M. T. Nguyen, and G. Pujolle. Evaluation of throughput optimization and load sharing of multipath TCP in heterogeneous networks. In *Proc. of Wireless and Optical Communications Networks (WOCN)*, pages 1 –5, may 2011.

[11] E. Nordmark and M. Bagnulo. Shim6: Level 3 Multihoming Shim Protocol for IPv6. RFC 5533, Internet Engineering Task Force, June 2009.

[12] V. Sharma, S. Kalyanaraman, K. Kar, K. Ramakrishnan, and V. Subramanian. MPLOT: A transport protocol exploiting multipath diversity using erasure codes. In *Proc. of INFOCOM*, pages 121–125, 2008.

[13] R. Stewart. Stream control transmission protocol. *IETF RFC 4960*, Sep. 2007.

[14] J. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros. Network Coding Meets TCP. In *Proc. of INFOCOM*, pages 280 –288, April 2009.

[15] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath TCP. In *Proc. of Usenix NSDI*, 2011.

[16] X. Zhuoqun, C. Zhigang, Y. Hui, and Z. Ming. An improved MPTCP in coded wireless mesh networks. In *Proc. of Broadband Network & Multimedia Technology*, pages 795–799. IEEE, 2009.