Pipeline Network Coding-Based Multipath Data Transfer in Heterogeneous Wireless Networks

Changqiao Xu, *Senior Member, IEEE*, Peng Wang, Chunshan Xiong, Xinpeng Wei, and Gabriel-Miro Muntean, *Member, IEEE*

***Abstract*—Multipath transmission control protocol (MPTCP) has attracted significant attention from standardization bodies, industrial communities, and academic communities. However, an important performance-related aspect is that MPTCP is negatively affected by packet reordering, especially in heterogeneous wireless environments. Although to improve the protocol, the proposed updated scheduling policies and congestion control methods do not solve the problem fundamentally. In this context, by breaking the strong binding between data packets and their sequence numbers, network coding has been demonstrated to be a promising solution for end-to-end multipath transmissions. However, current network coding solutions are based on batch coding without exception, and encoding and decoding operations cannot proceed unless all packets of a group have arrived. In addition, frequent generation and transmission of coding coefficients increase the delay and waste the already limited bandwidth. This paper proposes a novel pipeline network coding-based MPTCP (MPTCP-PNC) which reduces encoding and decoding delay and saves bandwidth by using innovative economic coding coefficient rules. Based on these, a quality-based distribution scheme and a corresponding transmission management policy are introduced to further improve the performance of MPTCP-PNC. Simulation tests involving video delivery over a multi-path distribution network show how MPTCP-PNC outperforms other state-of-art network coding solutions.**

***Index Terms*—Multipath TCP, pipeline network coding, coding coefficient, heterogeneous wireless networks.**

I. Introduction

During the past few years, mobile communication terminals have evolved to smart devices with multiple interfaces and support multiple wireless networks access [1], data processing [2], and content-rich video services [3]–[6]. Multipath Transmission Control Protocol (MPTCP) [7]–[9] has been proposed to allow TCP use concurrent multipath data transfer in order to improve network resource utilization and increase redundancy. Since its introduction, MPTCP has attracted important attention from industry, academia and standardization bodies. For instance, MPTCP has already been integrated into Apple's iOS 7.0 and above for commercial use. Samsung's Galaxy S6 supports a MPTCP service called Gigapath, which enable current smartphones reach a throughput of about 800 Mbps out of a theoretical maximum of 1.17 Gbps by making use of both fast LTE and fast Wi-Fi networks. Gigapath avails from MPTCP Linux and Android kernels which have been implemented and used. Additionally, IETF has established a specific Working Group (WG) for its MPTCP research.

MPTCP targets improving overall throughput by using concurrent data transmission over multiple TCP subflows within one connection. However, there is a serious problem threatening the transmission performance when deploying MPTCP in heterogeneous wireless networks [10], [11], which is known as packet reordering issue [12]. Due to the dynamic nature and quality dissimilarities (e.g., bandwidth, delay and loss rate) of the paths between two end hosts, data packets from different paths often arrive at the receiver side out of order. In order to support reliable data packet delivery to upper layer, the receiver has to buffer the data packets already arrived and wait for the remaining packets sent over the slow paths, degrading the overall transmission performance. There is also an extreme case when the receiver buffer has no room to buffer any new data packet, and advertises its zero window value to the sender. Therefore, the sender will stop sending and enter the persist mode resulting in great degradation of the overall throughput.

Many solutions adjusting the protocol itself involving improved packet scheduling policies and transmission management schemes have been proposed to mitigate the packet reordering issue. Scheduling policies take loss, receiver buffer and network delay into consideration when scheduling packets between paths [13]–[16]. These policies try their best to make reasonable scheduling decisions and avoid reordering events in the receiver buffer in advance. Some schemes based on superior transmission management principles mainly focus on adjusting congestion control to reduce the path delay dissimilarities [14], [17], [18], or using novel retransmission policies to retransmit the packets that the receiver expects [14], [19].

Manuscript received April 19, 2016; revised June 15, 2016; accepted June 16, 2016. Date of publication August 19, 2016; date of current version June 3, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61522103 and Grant 61372112, in part by the National Science and Technology Major Project under Grant 2015ZX03003002-002, in part by the Beijing Natural Science Foundation under Grant 4142037, and in part by the Innovation Research Program of Huawei Technologies Company Ltd under Grant YB2014040008.

C. Xu and P. Wang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: cqxu@bupt.edu.cn; weaponwang@bupt.edu.cn).

C. Xiong and X. Wei are with Huawei Technologies Company Ltd., Beijing 100085, China (e-mail: sam.xiongchunshan@huawei.com; weixinpeng@huawei.com).

G.-M. Muntean is with the Performance Engineering Laboratory, School of Electronic Engineering, Dublin City University, Dublin 9, Ireland (e-mail: gabriel.muntean@dcu.ie).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBC.2016.2590819

These schemes focus mainly on the out-of-order packets, but still follow the in-order principle of packet delivery. Thus they can alleviate the reordering problem only, but cannot address the problem fundamentally, especially in lossy and dynamic heterogeneous wireless network conditions.

Recently, network coding [20]–[22] has been employed by transport layer solutions and has already been proved an efficient and effective tool to solve the problem of packet reordering for multipath data transfer [23]–[25]. By employing network coding, the receiver is no longer concerned about the arrival of each of the packets sent in a group, but the number of coded packets it receives. This is as the receiver can retrieve the original data sent by the group of packets. Besides, lost packets can be compensated by sending enough redundant packets in the group. However, state-of-art network coding solutions without exception are based on batch coding [26], [27] for multipath transport protocols, so each coded packet is a linear combination of all original packets within one group. The sender must have all packets within a group to proceed the encoding process and data transmission, and the receiver must wait for the arrival of enough packets within the group to launch the decoding process. Moreover, in current network coding solutions, in order to ensure that the coding vectors of one group are linear independent with each other, coding coefficients for each coded packet are randomly and frequently generated from the Galois Field when performing the encoding operation. All these factors in batch coding-based solutions introduce high encoding and decoding delays, and result in additional bandwidth usage due to the transmission of those coding coefficients.

This paper proposes a novel **Pipeline Network Coding-based MPTCP (MPTCP-PNC)** for increased multipath data transfer performance in heterogeneous wireless environments by taking advantage of pipeline coding [28]–[30]. In MPTCP-PNC, original packets of a group are encoded progressively at the sender side without waiting for the availability of all original packets of this group. Some redundant packets are also generated and injected to the network as a part of this group in order to ensure there are enough packets arriving at the receiver for the original packets to be retrieved successfully. Compared with batch coding-based network coding solutions, in MPTCP-PNC, a coded packet can be decoded immediately and delivered to the upper layer when all coded packets ahead of it within the same group have arrived, without waiting for the arrival of all the packets of this group. Furthermore, use of economic coding coefficient rules is also introduced to help the encoding process avoid frequent coefficient generation and transmission, reducing the encoding delay, improving the overall throughput and saving bandwidth. Based on these, a quality-based distribution scheme and a corresponding transmission management policy are proposed to further improve the performance of MPTCP-PNC.

To the best of our knowledge, MPTCP-PNC is the first to apply pipeline network coding principle to MPTCP. The main contributions of the work described in this paper are as follows:

- Introduce the novel Pipeline Network Coding principle which in conjunction with MPTCP to solve the reordering problem and reduce encoding and decoding delay.
- Employ novel economic coding coefficient rules which reduce the encoding process complexity. As the coefficients are not transmitted via the transmission process, overhead in MPTCP data packet delivery is reduced and bandwidth is saved.
- Propose a novel path-quality-based distribution scheme, a corresponding congestion control mechanism and a retransmission policy to further improve the benefits of pipeline network coding.

The proposed MPTCP-PNC is tested in a multipath heterogeneous wireless network environment for video delivery and the results show how it outperforms other similar solutions in terms of throughput, delay and overhead.

II. Related Work

A. Multipath Transmission Research

Recently, multipath transmission research has attracted attention from industry, research institutes and standardization bodies (i.e., IETF). Stream Control Transmission Protocol (SCTP) [31] and MPTCP are two typical protocols to utilize heterogeneous network interfaces. SCTP and its Concurrent Multipath Transfer extension (CMT-SCTP) [32]–[34] make use of multihoming to establish a transmission association including several paths between two end hosts. Unlike SCTP, MPTCP sets up concurrently several TCP subflows within an established connection to provide multipath transmission support and offer backward compatibility with the classic TCP. Xu *et al*. [35] presents a survey of congestion control solutions for multipath transport protocols and discusses the multipath congestion control design. The fact that most current network applications and infrastructures are using TCP, while SCTP solutions have no TCP compatibility, making them costly to be deployed in practical network environments, have encouraged the latest efforts to deploy MPTCP in practice [7]. This is also the main reason, the proposed pipeline network coding scheme is applied to MPTCP.

As already mentioned, packet reordering is critical in MPTCP, and many efforts have been made to address this problem. Solutions can be mainly classified into two categories: scheduling policies and transmission management schemes.

Trying to improve the packet scheduling policy, Sarwar *et al.* [13] proposed a delay aware packet scheduling scheme for multipath data transfer over asymmetric network paths, trying to minimize the blocking inside the receiver's buffer and solve the reordering problem. However, the solution just takes wired paths into consideration and ignores the dynamic nature of wireless networks, so it cannot be adapted to heterogeneous wireless network conditions. Oh and Lee [15] presented a new scheduling scheme which performs packet scheduling according to the receiver buffer and path delay and use a delay constraint to adjust the trade-off between throughput and delay performance. Ni *et al.* [16] also proposed a new Fine-grained Forward Prediction based Dynamic Packet Scheduling Mechanism

called F2P-DPS for MPTCP. It allocates some specific packets to under-scheduling TCP subflows by estimating the number of packets that will be transmitted on other TCP subflows simultaneously.

Regarding transmission management schemes, Zhou *et al.* [17] proposed a congestion window adaption algorithm for MPTCP (CWA-MPTCP). CWA-MPTCP dynamically adjusts the congestion window of each subflow, aiming at reducing the delay difference of different paths. Ferlin-Oliveira *et al.* [18] introduced the Multi-Path Transport Bufferbloat Mitigation (MPT-BM) to control bufferbloat, whose main idea is to capture the bufferbloat by monitoring the RTT delay difference of each flow and adjust the upper bound of the congestion window of a path. Shailendra *et al.* [19] modified fast retransmission algorithm and proposed MPSCTP as a multipath extension of SCTP. MPSCTP uses two levels of sequence numbers including Transmission Sequence Number (TSN) and Path Sequence Number (PSN) to alleviate the reordering problem.

However, the solutions mentioned above still obey the in-order packet reception requirements and suffer from reordering in the receiver buffer.

B. Network Coding-Based Multipath Transmission

Traditionally, in packet switching networks, network nodes within a path simply transmit the data packets in a store-and-forward mode from sender to receiver. By contrast, network coding allows intermediate nodes to apply a linear transformation of the original packets to coded ones before passing them on. The receiver can retrieve all the original data packets once it receives enough linearly independent coded packets. Network coding was initially proposed by Ahlswede *et al.* [36] and has evolved into linear network coding [20] and random linear network coding [37]. It is an innovative and practical way to achieve high transmission rates in multicast or broadcast networks [22], [38]. Wang *et al.* [39] proposed a network coding based broadcast scheme that allows a base station (BS) to broadcast a given number of packets to large number of users, while being able to provide a performance guarantee on the probability of successful delivery. Further, network coding has been demonstrated to be an efficient approach for multipath transmission in lossy wireless network environments [40], [41].

Network coding, initially applied to intermediate nodes within multicast and broadcast networks, was used in the network layer to improve end-to-end communications. From the perspective of coding principle, batch-based network coding scheme is the most widely used in current network coding solutions. Batch coding requires coded packets to be linear combinations of all original packets in one group. At transport layer, the authors of this paper have previously applied batch network coding principle in CMT-SCTP to solve the data reordering problem, compensate for lost packets and reduce the number of retransmissions [23]. Li *et al.* [24] proposed NC-MPTCP to utilize batch network coding to boost the overall throughput when path dissimilarity exists. At the core of NC-MPTCP is the mixed use of regular and NC subflows, through which head-of-line blocking is avoided. Cui *et al.* [42] proposed FMTCP to take advantage of the fountain code [43] to flexibly transmit encoded symbols from the same or different data blocks over different subflows.

Batch-based network coding is an efficient solution to solve the reordering problem fundamentally in multipath transmission protocol at transport layer. Nevertheless, there are two critical drawbacks of batch network coding-based solutions in multipath transport protocols [24], [42]. On one hand, when using batch coding, each coded packet will encode all the original data packets within the same group. As a consequence, when receiving coded packets, the receiver cannot retrieve any original data packet before it accumulates enough coded packets of this group. Therefore the encoding and decoding latency proportionally increases with group size. On the second hand, at sender side, when generating a coded packet, coding coefficients are generated frequently from a finite field. Besides, to guarantee a coded packet is useful for decoding, the sender must ensure that every coding vector corresponding to its coded packet must be linearly independent with each other in the same group. A coding vector is a vector of coefficients that reflect the linear combination of the original data packets. This frequent coefficient generation and linear independence check make the encoding process time-consuming and complex. These two critical drawbacks mentioned above severely constrain the performance and efficiency of network coding in multipath transmissions.

In pipeline network coding [28]–[30], first coded packet of a group include only the first original packet of this group. The last coded packet and redundant coded packets of this group is a linear combination of all original packets of this group, following a one-to-all progressive principle. Making use of this scheme, encoding and decoding delay is largely reduced. Chen *et al.* [28] applied pipeline network coding scheme in a high loss, multihop wireless scenario and noted the throughput was significantly improved and the delays were largely reduced compared to a classic batch coding scheme. Li *et al.* [29], [30] proposed a reliable multicast protocol called CodePipe for lossy wireless networks which achieves simple coordination between nodes and improves the multicast throughput significantly. However, pipeline network coding has only been applied to network layer scenarios so far.

Based on the literatures discussed above, this paper innovatively applies the pipeline network coding principle to MPTCP-PNC for improved performance at transport layer.

III. MPTCP-PNC Overview

Fig. 1 illustrates MPTCP-PNC system architecture, which includes three major components: a Sender, a Receiver and a Multipath Wireless Network. The Sender is mainly composed of the following modules: **Packet Grouping**, **Pipeline Network Coder**, **Quality-based Data Distributor**, **Loss Rate Estimation** and **Transmission Management**. The Receiver's major modules include: **Receiver Buffer**, **Pipeline Network Decoder** and **Data Assembler**. Next these modules will be described in details in the context of MPTCP-PNC.

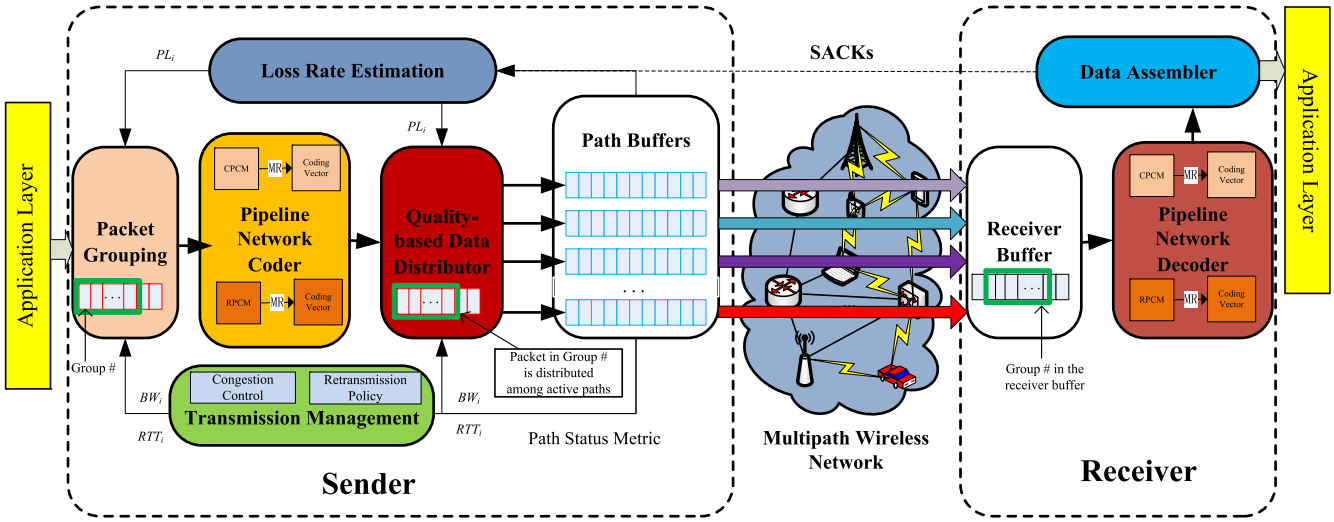

Fig. 1. Design Overview of MPTCP-PNC.

When the application delivers the original data to the sender, the sender encapsulates it into data segments and attaches the MPTCP common header to form data packets. On the basis of path-related information about estimated end-to-end delay, available bandwidth and loss rate received from the **Loss Rate Estimation** module, MPTCP-PNC decides how many data packets can be grouped together for network coding operation in the **Packet Grouping** module. Other types of packets (i.e., control) will not be involved in the network coding process and will be sent out as usual.

The **Pipeline Network Coder** performs pipeline network coding operations, linearly combining these original packets of a group to form coded packets. Notably, the proposed innovative pipeline network coding scheme and the new economic coding coefficient rules make the encoding process quite different from the multipath transmission solutions based on batch network coding. Then the **Quality-based Data Distributor** enables data distribution among all active paths, adaptively choosing the path(s) with better quality to distribute reasonable number of coded packets to the appropriate paths. This is crucial in order to obtain a good performance of pipeline network coding. To ensure robustness to transmissions over lossy wireless environments and to enable faster decoding operation at the receiver side, redundant coded packets will be appended by the **Pipeline Network Coder** following each group. The number of the redundant packets is based on real-time network conditions, in order to accelerate the decoding process and avoid retransmissions as many as possible. Meanwhile, the **Loss Rate Estimation** module monitors the path buffers and accepts SACK messages, learning about the packet reception status. The **Transmission Management** module maintains the congestion control information, manages the sending behavior and retransmits data if necessary. The transmission control is based at both group and packet levels.

When arriving at the receiver side, coded packets are stored in the **Receiver Buffer** then handed over to the **Pipeline Network Decoder** for original packet retrieval using the Gaussian Elimination method. Next, the decoded original data packets are assembled and delivered to the upper layer by the **Data Assembler**. At last the receiver replies to the sender with a SACK message to report data reception.

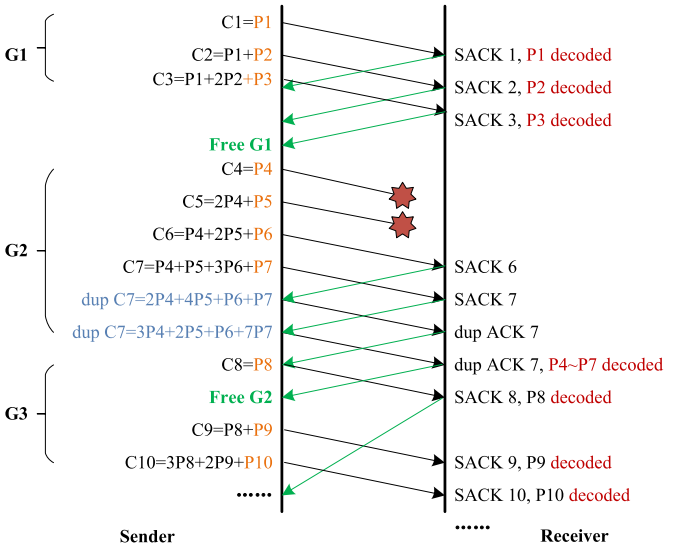

Fig. 2. Principle of Pipeline Network Coding in MPTCP-PNC.

The functionality and algorithms behind the modules mentioned above will be described in details in the following section.

IV. MPTCP-PNC Detailed Design

A. Principle of MPTCP-PNC

The principle of pipeline network coding in MPTCP-PNC is illustrated in an example in Fig. 2. The sender divides the original packets P1∼P10 with continuous Data Sequence Numbers (DSN) into three groups, namely G1, G2 and G3. Original packets within the same group are combined together to form coded packets. However, unlikely batch coding, original packet(s) included in each coded packet of a group follows

the one-to-all progressive approach. For instance, in G1, the first original packet P1 is encoded to C1 with coefficient 1, and the second coded packet C2 is a linear combination of original packets P2 and P1, using a random coefficient from the finite field for P1 and coefficient 1 for P2. The third coded packet C3 in G1 contains P1, P2 and P3 with coding coefficients 1, 2 and 1 for them respectively. The m-th coded packet C_m of a group can be obtained using a linear combination of original packets P1∼Pm, as is shown in eq. (1):

$$C_m = \begin{cases} P_1, m = 1 \\ \sum_{j=1}^{m-1} e_j P_j + P_m, m \in (1, N) \end{cases} \tag{1}$$

In eq. (1) e_j is the coding coefficient for original packet P_j and N is the group size, indicating how many original packets can be coded together. In batch coding, the sender buffer has to wait for the arrival of all data packets of a group before encoding process begins. However, in pipeline network coding-based MPTCP-PNC, upon receiving a new data packet from the upper layer, the sender will instantly trigger the encoding process based on currently received packets. The receiver can progressively construct the following decoding matrix upon the received packets within this group.

$$\begin{bmatrix} C_1 \\ \vdots \\ C_N \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ e_{2,1} & 1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ e_{N,1} & e_{N,2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} P_1 \\ \vdots \\ P_N \end{bmatrix} \tag{2}$$

In eq. (2), C_1∼C_N represent the coded packets of this group, the elements in the matrix are coding coefficients and $P_1 \sim P_N$ are the original packets of this group. At receiver side, in relative stable network conditions, the decoding matrix shown above can be solved progressively, without the need for the arrival of all coded packets of the group. For example, as shown in Fig. 2, the **Pipeline Network Decoder** can immediately retrieve $P8$ after receiving coded packet $C8$. Afterwards, $P9$ and $P10$ can also be retrieved once $C9$ and $C10$ are received in succession, and so on. Besides, when the wireless network environment becomes dynamically unstable, the decoding process can also be performed in a regular network coding way. For in stance, for G2 in Fig. 2, coded packets $C4$ and $C5$ are lost due to severely lossy heterogeneous wireless network conditions. Although the receiver cannot retrieve coded packets progressively, MPTCP-PNC can add proper number of redundant coded packets following a group, in order to guarantee enough coded packets arrive at receiver. As shown in Fig. 2, following G2 are two redundant coded packets (in light blue color) to compensate for the loss of $C4$ and $C5$. Once the receiver gathers enough coded packets (4 in this group), the decoding process can be performed. Following the process described above, MPTCP-PNC makes the encoding and decoding process time efficient, eliminating the reordering problem.

B. Economic Coding Coefficient Rules-Based Encoding and Decoding Process

When determining coding coefficients for a coded packet, present solutions [23]–[25] have to generate each coefficient randomly from a finite field $GF(2^8)$ in order to form a coding vector for this coded packet. Also, a coded packet is useful for decoding only when its coding vector is linearly independent with those of other coded packets within this group, so a linear independence check is indispensable. Frequent coded packet generation leads to a huge amount of coefficient generating operations and linear independence checks, which means the encoding latency is relatively large.

Additionally, both in SCTP and MPTCP, when integrating network coding [23]–[25], a coded packet includes both coded data and its corresponding coding vector. For example, NC-MPTCP attaches coefficients to the MPTCP packet header [24]. In conclusion, employing a variable group size N means there is a variable number of coefficients, which makes the packet header length-variable and bandwidth-wasting.

In MPTCP-PNC, at the beginning of establishing a connection, the sender and receiver negotiate and agree to maintain three structures: a Coded Packet Coefficients Matrix (**CPCM**), a Redundant Packet Coefficients Matrix (**RPCM**) and a Mapping Rule (**MR**), which are shown in eq. (3), eq. (4) and eq. (5).

$$\mathbf{CPCM} = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 \\ e_{2,1} & 1 & \ddots & \ddots & 0 \\ e_{3,1} & e_{3,2} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ e_{20,1} & e_{20,2} & \ldots & e_{20,19} & 1 \end{bmatrix} \tag{3}$$

$$\mathbf{RPCM} = \begin{bmatrix} e_{1,1} & \ldots & \ldots & e_{1,19} & e_{1,20} \\ e_{2,1} & e_{2,2} & \ldots & \ldots & e_{2,20} \\ e_{3,1} & e_{3,2} & e_{3,3} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ e_{10,1} & e_{10,2} & \ldots & e_{10,19} & e_{10,20} \end{bmatrix} \tag{4}$$

$$MR \ : \ (S_1, S_N, DSN, flag) \rightarrow \mathbf{Coding\ Vector}, \mathbf{CV} \tag{5}$$

CPCM and **RPCM** are coefficients matrices for coded packet and redundant coded packets, respectively. **MR** is a mapping rule from a tuple information of packet to its corresponding coding vector. Elements in the two matrices are generated from the finite field $GF(2^8)$ and linear independence checks among vectors have been performed beforehand. The details of **MR** are as follows:

S_1: The smallest DSN of original packets within the group.

S_N: The largest DSN of original packets within the group.

DSN: The DSN of the coded packet.

flag: An identifier determines the use of either *CPCM* or *RPCM*.

The **CPCM** and **RPCM** are generated at the beginning of establishing a connection and will be used throughout the life of this connection. As observed experimentally (detailed later), group size is around 10 and never exceeds 20 and the number of redundant packets never exceeds 10. We take the size of **CPCM** as 20×20 and size of **RPCM** as 10×20 as an upper bound. Note that **CPCM** and **RPCM** are extensible. The economic coding coefficient rules are embedded in encoding and decoding process to avoid frequent coefficient generation

Algorithm 1 Mapping Rule

/*when the Packet Grouping module finishes grouping and passes the original data packet to the Pipeline Network Coder */
Compute and obtain $(S_1, S_N, DSN, flag)$**;**
$GroupSize = S_N - S_1 + 1$; //calculate group size
$RowNumber = DSN - S_1 + 1$; //calculate row number
/*GroupSize determines used range of CPCM and RPCM, while RowNumber determines which coding vector will be selected */
//Select coding vector based on *GroupSize* and *RowNumber*
if$(flag == 0)${//this is a coded packet, and CPCM is used
Select coding vector CV for coded packet from CPCM;
}**else{**//this is a redundant packet, and RPCM is used
Select coding vector CV for coded packet from RPCM;
}

Algorithm 2 Pipeline Network Encoding Process

/*When data are delivered from application layer to transport layer */
Package proper amount of data in one data segment;
//based on the *MSSi*
if$(cn == N)${//means the end of a group
$cn = 1$ and start a new group; //cn records the $cn - th$ packet in this group
Empty the present encoding matrix;
}**else{**//means the packet still belongs to the current group
$cn++$;
}
Choose the coding vector $(a_1, a_2, \ldots, a_N)$ from *CPCM* or *RPCM* based on the Mapping Rule;
Add the vector $(a_1, a_2, \ldots, a_N)$ to the encoding matrix;
Use the coding vector $(a_1, a_2, \ldots, a_N)$ to form the coded data;

and transmission, in order to reduce the encoding complexity and save bandwidth, and improve overall throughput.

After negotiation in the connection establishment stage, the sender and receiver maintain the same **CPCM**, **RPCM** and **MR**. An improved protocol designed to support multipath transmission in MPTCP-PNC, is shown in Fig. 3, where the fields marked by yellow play an important role in the multipath data transmission. At sender side, the **Pipeline Network Coder** can use this information to select coding vector and perform encoding operation. At the receiver side, when determining coding coefficients for a coded packet, the **Pipeline Network Decoder** directly selects the coding vector from **CPCM** or **RPCM**, which is enabled by the **Mapping Rule** from $(S_1, S_N, DSN, flag)$ to the corresponding coding vector. The tuple $(S_1, S_N, DSN, flag)$ together with coded data is then able to reconstruct the coded packet. Thereafter, the sender can perform pipeline encoding and avoid frequent coefficient generating operations. This results in further reduction of encoding latency. The **Mapping Rule** is specified in **Algorithm 1**. The economic coding coefficient rules are embedded into the Economic Coding Coefficient Rules-based Encoding and Decoding Process. **Algorithm 2** describes the encoding process, in which the coding vector is selected from either **CPCM** or **RPCM** and is assigned to create the coded data.

Moreover, in comparison with other network coding based multipath transmission schemes, the coding coefficients will not be appended to packet headers and coded data only is transmitted through the network. The reason behind this is that the receiver can recover the corresponding coding vector

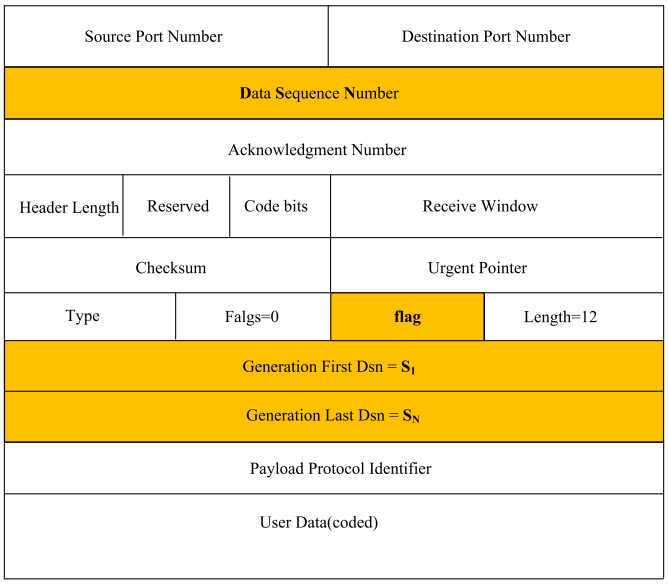

Fig. 3. Design of Packet Format in MPTCP-PNC.

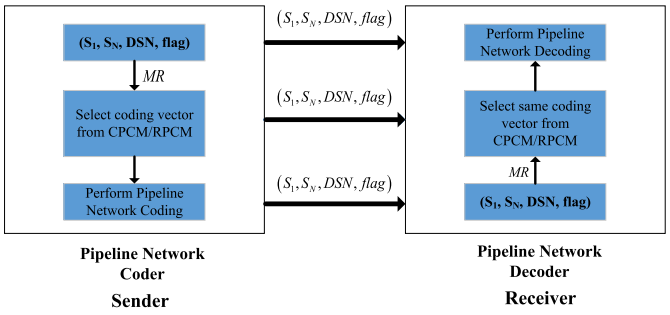

Fig. 4. Transmission Mechanism of MPTCP-PNC.

based on the tuple $(S_1, S_N, DSN, flag)$ which is transmitted in the data packet from source to destination, as shown in Fig. 3. The novel transmission mechanism is based on our novel protocol design whose principle is illustrated in Fig. 4.

Note that coefficient generating operations and linear independence checks are enforced at the beginning of a connection only and will be avoided during the life of one connection. Consequently, encoding is less complex and causes less delay. At the same time, transmission is bandwidth-efficient because the coefficients are not transmitted via the network.

C. Group Size and Redundancy Calculation

The original packets from applications are first delivered to **Packet Grouping** to determine how many of them can be coded together, i.e., group size N. N should be appropriately set for following two reasons: on one hand, a too small group size makes transmission fall back to the one without network coding, on the other hand, a too large group size can still lead to low reordering tolerance. N is determined by path metrics, considering bandwidth and delay as described next. When retransmission events happen, N is updated, keeping track of real time status of the multipath wireless environment.

The Round Trip Time (RTT_i) is employed to reflect the delay of path i. Since MPTCP requires replying subflow-level ACKs on the same path over which the corresponding data was sent [9], the Smoothed RTT (SRTT) measurement of each path is the same to that of regular TCP and is accurate enough for correct delay estimation.

The calculation of bandwidth share over path i is also crucial for group size estimation. As MPTCP is based on regular TCP, we apply and adapt the TCP bandwidth estimation scheme described in [44] to MPTCP-PNC for bandwidth estimation of each path, as shown in eq. (6) and eq. (7):

$$BW_{i,k} = \frac{D_{i,k}}{t_{i,k} - t_{i,k-1}} = \frac{D_{i,k}}{\Delta t_{i,k}} \tag{6}$$

In eq. (6), $BW_{i,k}$ is the k-th bandwidth sample of path i, $t_{i,k-1}$ is the previous SACK reception time and $D_{i,k}$ is the number of data bytes acknowledged during the interval of current SACK and previous SACK. For the purpose of smoothing the bandwidth sample and refraining to collect excessive bandwidth samples during packet bursts, we adapt the filtered method used in TCP Westwood [44] to MPTCP-PNC to obtain the bandwidth estimation of each subflow.

$$\widehat{BW_{i,k}} = \alpha_k \widehat{BW_{i,k-1}} + (1 - \alpha_k)\frac{BW_{i,k} + BW_{i,k-1}}{2} \tag{7}$$

In eq. (7), α_k is a smooth factor at time $t_{i,k}$ and its value is computed as in eq. (8).

$$\alpha_k = \frac{2\tau - \Delta t_{i,k}}{2\tau + \Delta t_{i,k}} \tag{8}$$

In eq. (8) τ is the filter parameter with $\tau > RTT_i$.

We define N as in eq. (9) to ensure different paths can finish the transmission of coded packets belonging to the same group within the same time interval:

$$N = \left\lceil \sum_{i \in S} \frac{BW_i}{MSS_i} * \frac{RTT_{\max} - RTT_i}{2} \right\rceil \tag{9}$$

In eq. (9), MSS_i is the Maximum Segment Size of path i which can differ between subflows. The MSS of each path is dynamically adapted based on the contribution of the subflow to the connection's throughput. BW_i/MSS_i represents the number of packets that can be sent on path i per unit of time. $RTT_{\max}$ is the largest RTT of all active paths. S is the set of all active paths within the current connection. Appropriate N value guarantees that packets of a group can arrive at the receiver within a same time round as well as making best efforts to avoid data reordering among groups. **Algorithm 3** describes the calculation and update of N. When retransmission event is detected on a path i, i's RTT and bandwidth will be obtained. Finally, the group size N is calculated and updated according to eq. (9) by the **Packet Grouping** module.

Redundancy is used to compensate the probable packet loss and redundant packets follow the group transmission. Every redundant packet is a linear combination all original packets, thus any coded packets in the same group can be substituted by the redundant packets following the group. However, there should neither be too many redundant packets, adding excessive unnecessary data to the network, nor too few, causing retransmissions and long decoding delay. The number of redundant packets belonging to a group is defined to guarantee enough number of packets reaching the receiver as is shown in eq. (10).

Algorithm 3 Calculation and Update of Group Size N

/*when retransmission event happens in an active path i*/
/*RTT estimation*/
Calculate the RTT of path i according to the approach used in common
TCP as RTT_i;
Find out the maximum RTT of all active paths as $RTT_{\max}$;
/*Bandwidth estimation*/
Record the arrival time of every acknowledgement at sender side;
Calculate the amount of data during two successive SACKs;
Update bandwidth of path i according to eq. (6), eq. (7) and eq. (8);
Update group number N according to eq. (9)

$$R = \left\lceil \sum_{i \in S} \left[N_i * \frac{PL_i(1 + \sigma_i)}{1 - PL_i(1 + \sigma_i)} \right] \right\rceil \tag{10}$$

In eq. (10), σ_i is the mean square deviation of packet loss of path i and PL_i is the loss rate of path i which will be detailed later and N_i is the number of coded packets sent on path i within a group. Obviously, the sum of N_i should be N.

D. Quality-Based Data Distribution

After the packet grouping and pipeline encoding process, coded packets will be distributed among all active paths. As mentioned before, coded packets within a group arriving in-order at receiver could be decoded progressively, without waiting for the reception of all the group code packets. In order to avail from the advantages brought by the pipeline network coding scheme in MPTCP-PNC, best-effort in-order packet transmission is needed.

For instance, consider a case when there are two paths A and B, both with $cwnd = 5$ and RTT of 200ms and 20ms, respectively. At a certain time, there are 5 outstanding packets on path B. Now there is a 6th packet ready to be sent. MPTCP default policy will schedule this packet to path A because there is no available $cwnd$ for path B. The average delivery delay for this scheme is calculated as 100ms. However, if the 6th packet is scheduled on path B, the total delivery delay would be 30ms. This situation is depicted in [45]. However, this example did not take packet loss into consideration which means the solution is not suitable for heterogeneous wireless networks. We adapt it to accommodate delivery in lossy network environments. Q_i is defined as the interval between the time the packet is scheduled to path i and the packet arrival at the receiver side. When distributing packets, the path with the least Q_i value is selected to schedule the packets. Note that the packet scheduled on path i will not necessarily be sent out immediately. Q_i is defined in eq. (11):

$$Q_i = \left(\frac{P_{wait_i}}{smooth\_cwnd_i} * \frac{1}{1 - PL_i} + \frac{1}{2} \right) * RTT_i \tag{11}$$

In eq. (11), P_{wait_i} is the data amount waiting in the send buffer, but not in the congestion window of path i,

$smooth\_cwnd_i$ is the smooth value of $cwnd_i$ which will be detailed later and $P_{wait_i}/(1-PL_i)$ is the estimated number of packets waiting in the sending buffer and outside the congestion window when we take the path loss rate into considerations. $P_{wait_i}/[(1-PL_i)*smooth\_cwnd_i]$ refers to how many *RTT* intervals this scheduled packet has to wait before entering into the congestion window. 1/2 RTT_i represents the one-way delay of the scheduled packet from sender to receiver.

We define $Q_{i\_optimal}$ as the lowest Q_i among all paths, which represents a path with optimal quality. $Q_{i\_suboptimal}$ is defined as the second lowest Q_i, which denotes the path with suboptimal quality. $Q_{i\_optimal}$ and $Q_{i\_suboptimal}$ are used to calculate the data amount D_i that would be scheduled to the path with quality value $Q_{i\_optimal}$, as shown in eq. (12):

$$D_i = \left(Q_{i\_suboptimal} - Q_{i\_optimal}\right) * BW_i \tag{12}$$

In eq. (12), BW_i represents the current bandwidth of path i. The data amount D_i computed above guarantees that the packets can arrive earlier at receiver if sent through the path with best quality Q_i (namely path with quality value $Q_{i\_optimal}$) than sent on other paths. However, if distributing packets more than D_i on the path with quality value $Q_{i\_optimal}$, some packets (in the tail of sender buffer) will arrive later than if sent on the path with quality $Q_{i\_suboptimal}$. Considering this, we iterate the process of data allocation to paths in order to enable best-effort in-order data transmission.

As we know, $cwnd_i$ decreases when packet loss happens. Thus, after scheduling packets on a path, if packet loss happens during the period those scheduled packets are waiting in the path buffer, their waiting time Q_i is actually larger than the calculated value beforehand using the initial congestion window size when scheduling. So we take packet loss into consideration to get a more accurate and smoothed congestion window size of this path. When scheduling a packet on path i, the number of packets in the sender buffer (including in and outside the congestion window) is $P_{wait_i} + cwnd_i$, and we define β as the number of possible packet lost among those packets, as shown in eq. (13):

$$\beta = \left\lfloor \frac{\left(P_{wait_i} + cwnd_i\right) * PL_i}{MSS_i} \right\rfloor \tag{13}$$

We know that every time packet loss happens, *cwnd* halves. When taking loss into consideration, we use the smooth *cwnd* value $smooth\_cwnd_i$ as in eq. (14):

$$smooth\_cwnd_i = \lambda * cwnd_i \tag{14}$$

In eq. (14) λ is a smooth factor and is defined in eq. (15).

$$\lambda = \frac{\sum_{i=0}^{\beta} \left(\frac{1}{2}\right)^i}{\beta + 1} \tag{15}$$

By using eq. (13), eq. (14) and eq. (15), we can get the smoothed congestion window size $smooth\_cwnd_i$ of path i, which can be used in eq. (11) to estimate the path quality.

MPTCP-PNC uses the mean square deviation of path loss rate to estimate the number of lost packets, for the purpose of compensating the long-time burst losses, which happen with high probability in wireless network environments.

TABLE I
CONFIDENCE LEVEL AND CORRESPONDING Z VALUE

$1-\alpha$	$Z_{\alpha/2}$
0.70	1.04
0.75	1.15
0.80	1.28
0.85	1.44
0.90	1.645
0.92	1.85
0.95	1.96
0.96	2.05
0.98	2.33
0.99	2.58

In the **Loss Rate Estimation** module, for each path, we record the number of packets that have been sent, including coded ones, redundant ones and retransmitted ones within one group. Also, the number of successfully acknowledged packets in this group is also recorded. When a group is decoded completely, the sender can get the unacknowledged percentage per path as a sample of loss rate sample pl_sample_i, using 1 minus the acknowledged percentage per path, as shown in eq. (16).

$$pl\_sample_i = 1 - \frac{Acked\_Packets}{Total\_Packets\ Sent\ Out} \tag{16}$$

We notice that some groups may have no packet loss. In this situation, the no-loss group is combined with the next group until packet loss happens.

After collecting samples of loss rate, we can get the mean value of path loss rate as $\overline{PL_i}$. Assuming that $\overline{PL_i[M]}$ is the previous mean path loss rate of path i, which is calculated with the previous M sample, e.g., pl_sample_i [1], pl_sample_i [2],..., pl_sample_i [M]. When new sample of path loss rate pl_sample_i [$M+1$] is collected, the mean value of path i's loss rate is calculated using eq. (17).

$$\overline{PL_i[N+1]} = \frac{\overline{PL_i[N]} * N + pl\_sample_i[N+1]}{N+1} \tag{17}$$

Similarly, the mean square deviation of path i's loss rate can be obtained using eq. (18).

$$\sigma_i[M+1] = \sqrt{\frac{{\sigma_i}^2[M] * (M-1)}{M} + \frac{\left(pl\_sample_i[M+1] - \overline{PL_i[M]}\right)^2}{M+1}} \tag{18}$$

In eq. (18), $\sigma_i[M]$ is the previous mean square deviation of path i's loss rate. Next, we use the confidence interval to determine the value of PL_i, as shown in eq. (19).

$$\Pr\left\{\overline{PL_i} - Z_{\alpha/2} * \frac{\sigma_i}{\sqrt{M}} < PL_i < \overline{PL_i} + Z_{\alpha/2} * \frac{\sigma_i}{\sqrt{M}}\right\} = 1 - \alpha \tag{19}$$

M is the number of samples and $1-\alpha$ is the confidence level. PL_i and σ_i are the mean value of path i's loss rate and its mean square deviation, respectively. $Z_{\alpha/2}$ is a function of $\alpha/2$. The relation between $1-\alpha$ and $Z_{\alpha/2}$ is shown in Table I. When updating, PL_i can be randomly chosen from the interval according to different confidence levels. PL_i is used in eq. (10) for estimating the number of redundant coded packets following a group and is also used for calculating the path quality Q_i

Algorithm 4 Quality-Based Data Distribution

```
/*Iterate the process to distribute packets among subflows*/
for (every active path i ∈ current connection){
  Update loss rate and its mean square deviation of this path
  according to eq. (16)~(19);
  Calculate Q_i according to eq. (11);  // path i's quality
  i++;  //move to next active path
}
Sort all active paths in ascending order according to their
path quality values;
//Quality-based Distribution
Find out Q_i_suboptimal and Q_i_optimal;
Calculate the amount of data that will be distributed according
to eq. (12);
Distribute the data packets to the path with best quality;
```

in quality-based distribution scheme which is implemented in **Quality-based Data Distributor** module. The quality-based distribution scheme is detailed in **Algorithm 4**.

V. Congestion Control and Retransmission Policy

After pipeline network coding process, each coded packet is a linear combination of the original packets within a group, being also correlated with each other. Traditional transmission management is performed at the level of single data packet, which is not appropriate to be used here. Some state-of-art network coding solutions' transmission management is based on the group, which is also not flexible and efficient enough. However, in MPTCP-PNC, the receiver can retrieve original packets progressively if coded packets within a group arrive in-order. In this case, the receiver can acknowledge these retrieved original packets to the sender immediately and the congestion window of the corresponding path is increased, achieving higher transmission rate. So we propose a novel hybrid **Transmission Management**, both at group and packet levels, and includes congestion control and retransmission policy.

MPTCP-PNC has four congestion control phases: slow start, congestion avoidance, fast retransmit and fast recovery on each path, which is similar to standard TCP. For each path i (subflow), $cwnd_i$ and $ssthresh_i$ are maintained as congestion window and slow start threshold of this path. Group Time Out (GTO) is defined as the RTO of the first coded packet in the group. *FG* is defined as the first group not completely decoded and coded packets of this group are in front of the send buffer waiting for acknowledgement. *Lost Report* records the number of lost packets in a group.

As the coded packets in *FG* are progressively decoded, the corresponding original data packets can be retrieved immediately. At this moment, a SACK message reports new DSN in *FG* and $cwnd_i$ can be adjusted instantly in the unit of packet, according to [46] in order to increase the sending rate. Additionally, another possible condition is that the receiver collects enough coded packets before acknowledging new or duplicated DSN to the sender, where DSNs in the group are acknowledged to the sender in the unit of group and $cwnd_i$ can also be adjusted in the unit of the whole group. Congestion control still sustains an AIMD feature; however, it is both at group and single packet levels. Also if the *FG*'s *Lost Report* value exceeds three, then the sender should perform fast retransmission. If GTO expires, time out retransmission for the *FG* should be performed.

Congestion control of multipath transmissions in previous works [23], [25] is in a unit of group, which means congestion window of each path can be adjusted only when a group has been decoded. However, in MPTCP-PNC, since the congestion control is both at group and single packet levels, the acknowledgement of any packet in *FG* can trigger the update of $cwnd_i$, clear the *Lost Report* and reset the GTO timer. The update of $cwnd_i$ is improved to grow more steadily and more flexible compared to previous schemes.

When fast retransmission happens, path i with most outstanding data will be selected for congestion window adjustment. The adjustment should have a loss differentiation discriminator to decide the loss reasons. Here we adopt the classifier described in [47] which is used to analyze loss reasons in heterogeneous TCP wireless networks and we adapt it to MPTCP-PNC, as shown in eq. (20):

$$diff_i = \left(\frac{cwnd_i}{\min RTT_i} - \frac{cwnd_i}{RTT_i}\right) * \min RTT_i \tag{20}$$

In eq. (20), $\min RTT_i$ is the minimum RTT on path i since the beginning of the observation and RTT_i is the real time RTT of path i. That is, given two parameters a and b, if $diff_i \geq b$, we assume that path i is congested. While if $diff_i \leq a$, it means that losses should be due to wireless link errors and $cwnd_i$ should not be decreased. At last, if $a < diff_i < b$, it means the network state is the same to the previous state. Here we take $a = 1, b = 3$ to get the estimation with highest accuracy [47]. When path i is congested according to the rules mentioned above, $ssthresh_i$ and $cwnd_i$ should be adjusted according to eq. (21):

$$\begin{aligned} ssthresh_i &= \max\left(cwnd_i/2, \frac{BW_i * \min RTT_i}{MSS_i}\right) \\ cwnd_i &= \begin{cases} \min(cwnd_i,\ ssthresh_i), & diff_i \geq b \\ cwnd_i, & otherwise \end{cases} \end{aligned} \tag{21}$$

In eq. (21), BW_i is the real time bandwidth of path i and $ssthresh_i$ has an additional factor than $cwnd_i/2$ only as in regular TCP.

When timeout retransmission happens, implying serious congestion or wireless link error, we perform congestion window adjustment moderately, similar to standard TCP for all involved paths, as described in eq. (22).

$$\begin{aligned} ssthresh_i &= \frac{cwnd_i}{2} \\ cwnd_i &= MSS_i \end{aligned} \tag{22}$$

All coded packets in both fast retransmission and timeout retransmission should set the DSN equal to the last packet (S_N) in the group and select coding vector from **RPCM** as mentioned in Section IV. Then the sender retransmits these packets on paths with best quality.

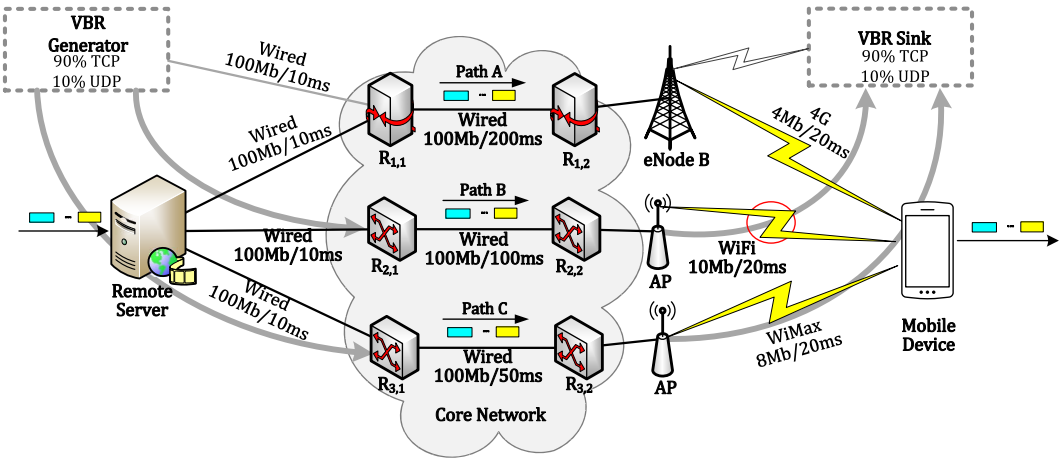

Fig. 5. Heterogeneous Wireless Network Topology in the Simulation.

TABLE II
PATH PARAMETER SETTINGS IN THE SIMULATION

Parameters	Path A	Path B	Path C
Wireless technology	4G	IEEE 802.11b	IEEE 802.16
Access bandwidth	4Mbps	10Mbps	8Mbps
Access link delay	20ms	20ms	20ms
Core network delay	200	100	50
Uniform loss rate	0.01-0.02	0.01-0.1	0.1-0.2
Markov loss rate	0.01	0.01	0.01

VI. PERFORMANCE EVALUATION

This section evaluates the performance of MPTCP-PNC for video data delivery. In our tests, we have tested MPTCP-PNC against three important previously proposed schemes: MPTCP/NC, NC-MPTCP and CMT-NC. MPTCP/NC is a reliable multipath protocol with network coding and utilizes all available networks in parallel [25]. NC-MPTCP introduces batch-based network coding to some but not all MPTCP subflows, and its essential principle is the mixed use of regular and NC subflows to achieve higher throughput over heterogeneous path conditions compared to original MPTCP [24]. CMT-NC firstly applied network coding into CMT-SCTP to address the reordering problem [23]. The comparison-based evaluation is performed in terms of throughput, delay, retransmission times and packet header overhead introduced by coding coefficients.

A. Simulation Setup

All the schemes are modeled and simulated in Network Simulator version 2.35 (NS-2). A heterogeneous wireless network topology as shown in Fig. 5 has been designed for this test. The mobile device is accessing video data from a remote sever via three independent paths, whose dissimilar configurations are listed in Table II.

In real deployment, the data rate is largely constrained by the narrow bandwidth of the wireless access links. So the wired links' bandwidth is set to 100Mbps in order not to limit the transmission in the core network. To simulate wireless errors, a Uniform loss model is attached to every wireless link, representing distributed loss to random contention, wireless interference or link handoff. A Two-State Markov loss model with a fixed rate of 0.01, representing infrequent continuous loss due to signal fading, transient failure or stream burst, is also attached to these paths. Moreover, cross traffic is injected through the network to simulate the Internet background as [14] did. The cross traffic is generated by Variable Bit Rate (VBR) with Pareto distribution, which then flows into the three paths to reach a VBR receiver. Based on the results presented in [48], the packet sizes in cross traffic are chosen as follows: 49% are 44 bytes, 1.2% are 576 bytes, 2.1% are 628 bytes, 1.7% are 1300 bytes and 46% are 1500 bytes. 90% of these packets are carried by TCP and the rest 10% are over UDP. The aggregated cross traffic on each path varies randomly between 0∼50 percent of the access link bandwidth. All the wired and wireless links are set to 50 packets queue limit with a Droptail queuing model, such that any saturated link would drop new incoming packets to simulate congestion loss.

In the simulation, the receiver buffer size is changed to observe the performance of MPTCP-PNC and other three schemes, while the sender buffer size is set large enough not to cause any issue. Other parameters are set to MPTCP default values. Each of the tested scheme starts in turn at 5.0s, whereas the background traffic always starts at 0.0s. All results presented are the average of 100 trials with a confidential level of 95%, which ensures the results are not influenced by any stochastic factors.

B. Simulation Results

Fig. 6 illustrates the change of throughput for the duration of the simulation. The plots are obtained for MPTCP-PNC, CMT-NC, NC-MPTCP, and MPTCP/NC, respectively, and indicate the traffic over paths A, B, C and the overall throughput of the solution when a 128KB receiver buffer is used. For all cases, it can be seen that the throughput increases abruptly at the beginning as the solutions try to make use of the available network capabilities and the slow-start algorithm doubles the window size repeatedly. At around 100.0s, due to packet loss and recovery, the throughput of all schemes fluctuates, while the congestion window is adjusted in the

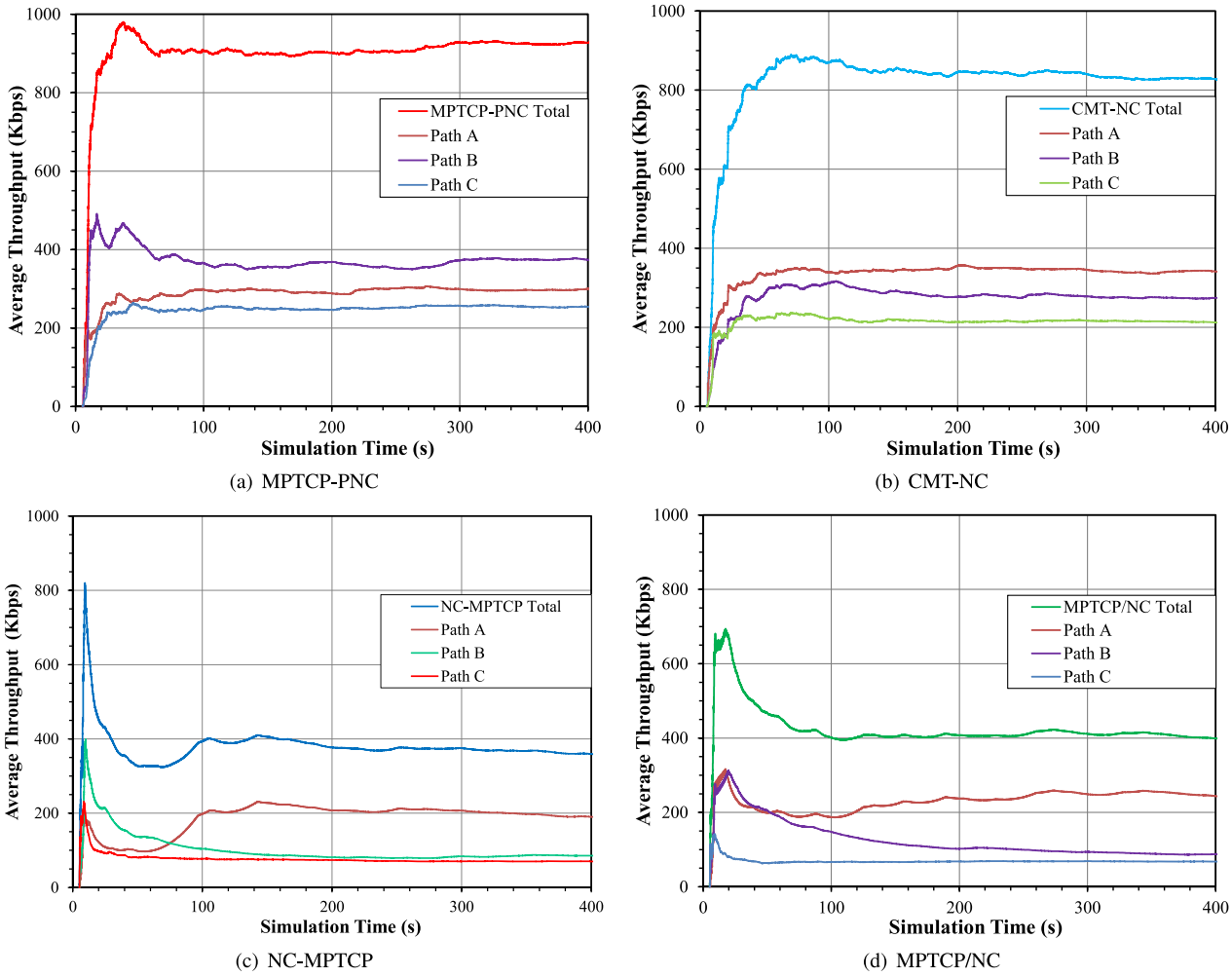

Fig. 6. Throughput Performance of MPTCP-PNC, CMT-NC, NC-MPTCP and MPTCP/NC with 128KB receiver buffer.

congestion avoidance phase. By making use of the pipeline network coding scheme in MPTCP-PNC, data reordering can be largely concealed, and in meanwhile, coded packets and proper number of redundant packets can be distributed to paths with better quality trying to achieve in-order delivery (thanks to the proposed Quality-based Data Distribution). Additionally, the supporting congestion control and retransmission policy make best efforts to sustain a high sending rate and avoid unnecessary congestion window reduction. Thus MPTCP-PNC achieves the highest level of throughput. NC-MPTCP does not perform well in this highly challenging heterogeneous wireless network environment and results in low throughput. The reason behind this phenomenon is obvious: on one hand, group size is not changed based on the network conditions, implying reordering still happens frequently; on the other hand, it could not identify the real cause of packet loss wireless error or path congestion, and therefore the window sizes are updated improperly and spurious retransmissions happen sometimes.

Finally, the throughput of all solutions reach a relative stable state at 360.0s, which is a cumulative effect of factors mentioned above. At time 400.0s, MPTCP-PNC achieves the best performance in terms of overall throughput (935.1Kbps), compared with CMT-NC (833.9Kbps), NC-MPTCP (362.7Kbps), and MPTCP/NC (377.3Kbps), respectively. This demonstrates the excellent performance of MPTCP-PNC in dynamic wireless environment.

Further, we vary the receiver buffer size to assess their comparative performance in terms of average throughput, average delay and number of retransmissions. As we can see from Fig. 7, all schemes have higher throughput with the increase of receiver buffer size, for larger receiver buffer size offering better tolerance to path dissimilarities. MPTCP/NC has the worst throughput performance for every buffer size. On the contrary, MPTCP-PNC is significantly better than the other three schemes. Although CMT-NC performs much better than NC-MPTCP and MPTCP/NC, it still has lower throughput than the innovative MPTCP-PNC for almost all buffer sizes. The reason behind this is that CMT-NC distributes data in a fast manner, instead of handing packets out to the path with better quality which is the case of MPTCP-PNC.

MPTCP-PNC and CMT-NC need 256KB buffer size to reach a stable throughput, while the buffer size for NC-MPTCP and MPTCP/NC to reach that relatively steady state is double. Proper group size and appropriate number of redundant packet based on real-time network conditions are the reasons of this phenomenon. In this sense, buffer capacity has

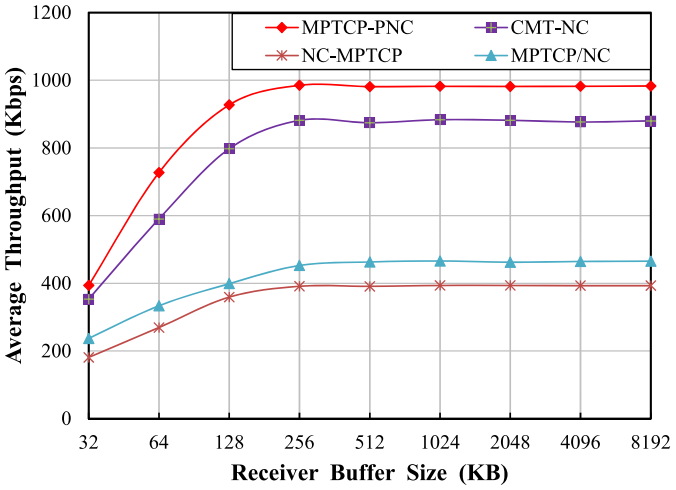

Fig. 7. Throughput variation with different buffer size.

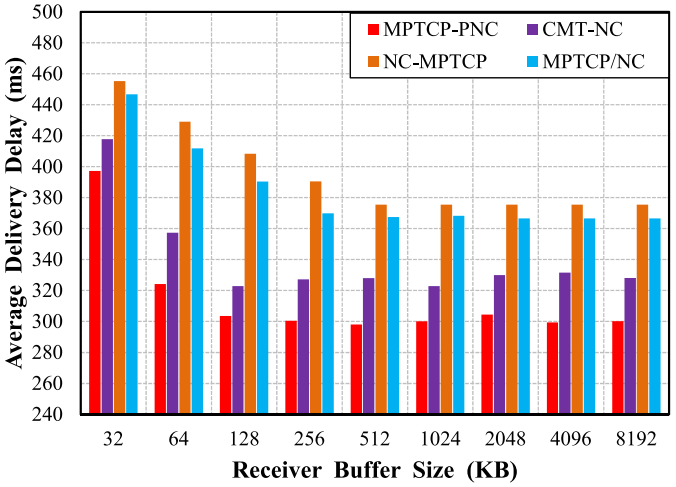

Fig. 8. Delivery delay variation with different buffer size.

been exploited adequately by MPTCP-PNC through applying pipeline network coding. As a result, with buffer size larger than 256KB, the throughput benefit of MPTCP-PNC is still high compared with that of the other three schemes. Even in the case of a 1MB buffer, MPTCP-PNC (983.2Kbps) still has 11.7% higher throughput than that of CMT-NC (880.1 Kbps), which performs best among the three compared schemes.

Fig. 8 illustrates the average packet delivery delay with the variation of buffer size. Notably, MPTCP-PNC achieves the least delay in comparison with the other three schemes. CMT-NC declares that delivery delay mainly includes transmission delay and buffer-holding delay in the receiver. However, it ignores the encoding delay while applying network coding to multipath transmissions. In fact, delivery delay should contain encoding delay, transmission delay and decoding delay. The transmission delay is small and almost identical in the same topology, so encoding and decoding delay is of vital importance and has a significant influence on the performance. Regarding the encoding process, the pipeline encoding in MPTCP-PNC supports a progressive encoding process, instead of waiting for the arrival of all the original packets within a group before beginning the encoding process. Moreover, the proposed economic coding coefficient rules avoid frequent coefficient generation, and thus the encoding delay is further reduced, compared to that of the other three schemes. In terms of the decoding process in MPTCP-PNC, appropriate number of redundant packets following a group based on the real time network conditions can compensate the packet loss, speeding up the decoding and reducing the delay. More importantly, any of the coded packets can be decoded and delivered to upper layer progressively if the packets before the current packet in the same group have arrived or it is the first packet of the group, unlike the other three schemes. MPTCP-PNC can proceed with the decoding operation without waiting for the arrival of all coded packets of this group, further reducing the decoding delay. Additionally, NC-MPTCP does not provide an effective distribution strategy which results in waiting for enough arrival of a group. MPTCP/NC has no module to determine group size and redundancy. That is why they behave poorly in the delay comparison.

With larger buffer size, the delay of all the four schemes decreases. It is notable that MPTCP-PNC achieves the least delay than those of the other schemes. Thus we can draw a conclusion that pipeline network coding embedded in MPTCP is of great advantage in reducing delivery delay in contrast with those of other three schemes which are batch coding-based. Additionally, the economic coding coefficient rules also play an important role in reducing the encoding and decoding delays further, compared with that of the other three schemes, where coding coefficients are frequently and randomly generated from the finite fields for each coded packet, causing too much encoding delay. As the receiver buffer size increases, delay becomes stable. Finally, MPTCP-PNC's stable value (300ms) is much lower than the values of the other three schemes (CMT-NC 328ms, MPTCP/NC 367ms, NC-MPTCP 376ms). These figures prove that there is a great advantage of MPTCP-PNC in terms of delivery delay.

Fig. 9 presents the retransmission times when transporting one 150MB video file over the same network topology using the four schemes. The figure distinguishes two kinds of retransmission, e.g., fast and timeout retransmission. A larger buffer size would not only increase the throughput due to better tolerance of data reordering, but also would involve more retransmissions because more gaps may exists in the receiver buffer. MPTCP-PNC has less retransmissions than the other three schemes for each buffer size, and it can be seen that NC-MPTCP and MPTCP/NC have a large number of retransmissions. Furthermore, MPTCP-PNC and CMT-NC almost have no timeout retransmissions, and MPTCP-PNC has the least fast retransmissions compared with CMT-NC.

MPTCP/NC and NC-MPTCP have more retransmissions because of improper number of redundant packets compensating packet loss and inefficiency in packets distribution. CMT-NC outperforms the above two schemes because it provides relatively proper group size and redundancy, however, it is inferior to MPTCP-PNC because of its distribution scheme which is not based on path quality. Reasons behind this observably performance are as follows. First, the number of redundant packets which is estimated based on network

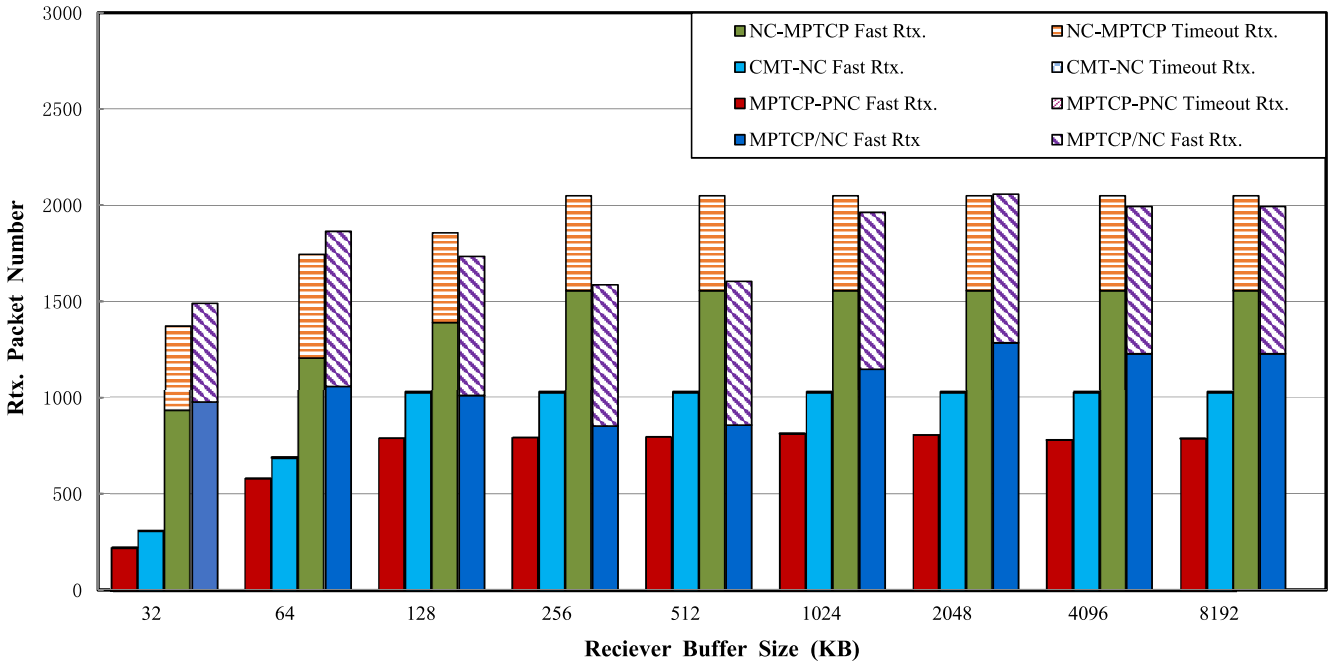

Fig. 9. Retransmission times variation with different buffer size.

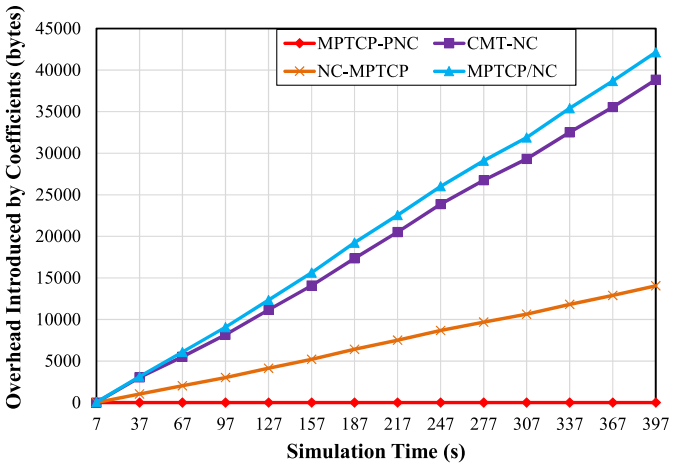

Fig. 10. Packet header overhead introduced by coding coefficients.

condition can make up for packet loss. Each coded packet can be complemented by other redundant packets within the same group. Second, our quality-based distribution scheme makes sure that packets are transmitted on better paths to guarantee the best-effort in-order delivery. Most importantly, the design of transmission control is important, as it is a mix based on group and single packet levels. It guards against the group timer from expiration with a high probability, thus timeout retransmissions are effectively controlled.

Fig. 10 shows the advantage brought by our economic coding coefficient rules in reducing the packet header overhead. In MPTCP-PNC, coefficients are not contained in packet header, which means no overhead will be introduced by coefficients. However, in other three schemes, coding coefficients are contained in the header, which means coefficients are transmitted in the network and huge amount of bandwidth are wasted. Due to the lack of determination of group size N, we take the default value of 4 and 6 for NC-MPTCP and MPTCP/NC, respectively. For CMT-NC, group size is updated according to [23, eq. (4)]. Group size N means every coded packet in this group has N coefficients inside the packet header. As shown, as the simulation time increases, overhead introduced by all other schemes increases as well. Due to smaller group size, NC-MPTCP achieves the least overhead compared with MPTCP/NC and CMT-NC during the simulation period. However, thanks to our novel economic coding coefficient rules, MPTCP-PNC introduces no overhead for packet header, thus it overlaps with x-axis in Fig. 10.

VII. Conclusion

This paper proposes a pipeline network coding-based MPTCP solution (MPTCP-PNC) for multipath data transfer in heterogeneous wireless network environments. MPTCP-PNC creatively makes use of pipeline network coding to address the data reordering problem in the receiver buffer, achieving encoding and decoding delay decreases. Furthermore, MPTCP-PNC pioneers a novel approach which embeds the economic coding coefficient rules in the pipeline network coding scheme, efficiently reducing encoding delay and effectively saving bandwidth. A quality-based distribution scheme for data packets over the dissimilar paths is also described to maximize the advantages brought by the pipeline network coding. Based on the pipeline coding scheme, we also propose a supporting congestion control scheme and retransmission policy which increase throughput and improve the robustness and efficiency of multipath data transmission. Simulation results involving the delivery of video content over a heterogeneous network environment show that MPTCP-PNC increases throughput and reduces the delivery delay and

number of retransmissions, compared with three current state-of-the-art solutions CNT-NC, NC-MPTCP, and MPTCP/NC. Additionally, MPTCP-PNC does not introduce any overhead which is especially very beneficial for video transmissions in bandwidth limited network environments.

REFERENCES

[1] C. Xu, F. Zhao, J. Guan, H. Zhang, and G.-M. Muntean, "QoE-driven user-centric VoD services in urban multihomed P2P-based vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 5, pp. 2273–2289, Jun. 2013.

[2] C. Xu, S. Jia, L. Zhong, and G.-M. Muntean, "Socially aware mobile peer-to-peer communications for community multimedia streaming services," *IEEE Commun. Mag.*, vol. 53, no. 10, pp. 150–156, Oct. 2015.

[3] D. Wu, J. Huang, J. He, M. Chen, and G. Zhang, "Toward cost-effective mobile video streaming via smart cache with adaptive thresholding," *IEEE Trans. Broadcast.*, vol. 61, no. 4, pp. 639–650, Dec. 2015.

[4] C. Xu, S. Jia, L. Zhong, H. Zhang, and G.-M. Muntean, "Ant-inspired mini-community-based solution for video-on-demand services in wireless mobile networks," *IEEE Trans. Broadcast.*, vol. 60, no. 2, pp. 322–335, Jun. 2014.

[5] A. Hava, Y. Ghamri-Doudane, G.-M. Muntean, and J. Murphy, "Performance-aware mobile community-based VoD streaming over vehicular ad hoc networks," *IEEE Trans. Broadcast.*, vol. 61, no. 2, pp. 238–250, Jun. 2015.

[6] C. Xu *et al.*, "Performance-aware mobile community-based VoD streaming over vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 3, pp. 1201–1217, Mar. 2015.

[7] S. Barré, C. Paasch, and O. Bonaventure, "Multipath TCP: From theory to practice," in *Proc. 10th Int. IFIP TC 6 Conf. Netw.*, Valencia, Spain, 2011, pp. 444–457.

[8] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," IETF, Fremont, CA, USA, RFC 6182, Mar. 2011.

[9] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," IETF, Fremont, CA, USA, RFC 6824, Jan. 2013.

[10] J. B. Ernst, S. C. Kremer, and J. J. P. C. Rodrigues, "A survey of QoS/QoE mechanisms in heterogeneous wireless networks," *Phys. Commun.*, vol. 13, pp. 61–72, Dec. 2014.

[11] J. B. Ernst, S. C. Kremer, and J. J. P. C. Rodrigues, "Performance evaluation of heterogeneous wireless networks considering competing objectives and viewpoints," in *Proc. ACM Symp. Appl. Comput. (ACM SAC)*, Salamanca, Spain, 2015, pp. 680–687.

[12] S. Barré, O. Bonaventure, C. Raiciu, and M. Handley, "Experimenting with multipath TCP," in *Proc. ACM SIGCOMM*, New Delhi, India, 2010, pp. 443–444.

[13] G. Sarwar, R. Boreli, E. Lochin, and G. Smith, "Mitigating receiver's buffer blocking by delay aware packet scheduling in multipath data transfer," in *Proc. IEEE WAINA*, Barcelona, Spain, 2013, pp. 1119–1124.

[14] C. Xu, T. Liu, J. Guan, H. Zhang, and G.-M. Muntean, "CMT-QA: Quality-aware adaptive concurrent multipath data transfer in heterogeneous wireless networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 11, pp. 2193–2205, Nov. 2013.

[15] B.-H. Oh and J. Lee, "Constraint-based proactive scheduling for MPTCP in wireless networks," *Comput. Netw.*, vol. 91, pp. 548–563, Nov. 2015.

[16] D. Ni, K. Xue, P. Hong, and S. Shen, "Fine-grained forward prediction based dynamic packet scheduling mechanism for multipath TCP in lossy networks," in *Proc. IEEE ICCCN*, Shanghai, China, 2014, pp. 1–7.

[17] D. Zhou, W. Song, and M. Shi, "Goodput improvement for multipath TCP by congestion window adaptation in multi-radio devices," in *Proc. IEEE CCNC*, Las Vegas, NV, USA, 2013, pp. 508–514.

[18] S. Ferlin-Oliveira, T. Dreibholz, and O. Alay, "Tackling the challenge of bufferbloat in multi-path transport over heterogeneous wireless networks," in *Proc. IEEE IWQoS*, Hong Kong, 2014, pp. 123–128.

[19] S. Shailendra, R. Bhattacharjee, and S. K. Bose, "A multipath variant of SCTP with optimized flow division extension," *Comput. Commun.*, vol. 67, pp. 56–65, Aug. 2015.

[20] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[21] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[22] S. Katti *et al.*, "XORs in the air: Practical wireless network coding," in *Proc. ACM SIGCOMM*, Pisa, Italy, Sep. 2006, pp. 243–254.

[23] C. Xu, Z. Li, L. Zhong, H. Zhang, and G.-M. Muntean, "CMT-NC: Improving the concurrent multipath transfer performance using network coding in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1735–1751, Mar. 2016.

[24] M. Li, A. Lukyanenko, and Y. Cui, "Network coding based multipath TCP," in *Proc. IEEE INFOCOM Workshops*, Orlando, FL, USA, 2012, pp. 25–30.

[25] J. Cloud *et al.*, "Multi-path TCP with network coding for mobile devices in heterogeneous networks," in *Proc. IEEE VTC Fall*, Las Vegas, NV, USA, 2013, pp. 1–5.

[26] J.-S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Medard, "Codecast: A network-coding-based ad hoc multicast protocol," *IEEE Wireless Commun.*, vol. 13, no. 5, pp. 76–81, Oct. 2006.

[27] Q. Dong, J. Wu, W. Hu, and J. Crowcroft, "Practical network coding in wireless networks," in *Proc. ACM MobiCom*, Montreal, QC, Canada, 2007, pp. 306–309.

[28] C.-C. Chen, S.-Y. Oh, P. Tao, M. Geria, and M. Y. Sanadidi, "Pipeline network coding for multicast streams," in *Proc. Int. Conf. Mobile Comput. Ubiquitous Netw. (ICMU)*, Seattle, WA, USA, 2010, pp. 1–7.

[29] P. Li, S. Guo, S. Yu, and A. V. Vasilakos, "Reliable multicast with pipelined network coding using opportunistic feeding and routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3264–3273, Dec. 2014.

[30] P. Li, S. Guo, S. Yu, and A. V. Vasilakos, "CodePipe: An opportunistic feeding and routing protocol for reliable multicast with pipelined network coding," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, 2012, pp. 100–108.

[31] R. Stewart, "Stream control transmission protocol," IETF, Fremont, CA, USA, RFC 4960, Sep. 2007.

[32] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 951–964, Oct. 2006.

[33] C. Xu, Z. Li, J. Li, H. Zhang, and G.-M. Muntean, "Cross-layer fairness-driven concurrent multipath video delivery over heterogeneous wireless networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 7, pp. 1175–1189, Jul. 2015.

[34] C. Xu, E. Fallon, Y. Qiao, and G.-M. Muntean, "Performance evaluation of multimedia content distribution over multi-homed wireless networks," *IEEE Trans. Broadcast.*, vol. 57, no. 2, pp. 204–215, Jun. 2011.

[35] C. Xu, J. Zhao, and G.-M. Muntean, "Congestion control design for multipath transport protocols: A survey," *IEEE Commun. Surveys Tuts.*, vol. PP, no. 99, Apr. 2016, doi: 10.1109/COMST.2016.2558818.

[36] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[37] T. Ho *et al.*, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[38] M. Wang and B. Li, "How practical is network coding?" in *Proc. IEEE IWQoS*, New Haven, CT, USA, 2006, pp. 274–278.

[39] P. Wang, G. Mao, Z. Lin, X. Ge, and B. D. O. Anderson, "Network coding based wireless broadcast with performance guarantee," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 532–544, Jan. 2015.

[40] D. Vukobratović, C. Khirallah, V. Stanković, and J. S. Thompson, "Random network coding for multimedia delivery services in LTE/LTE-advanced," *IEEE Trans. Multimedia*, vol. 16, no. 1, pp. 277–282, Jan. 2014.

[41] A. Mohapatra, N. Gautam, S. Shakkottai, and A. Sprintson, "Network coding decisions for wireless transmissions with delay consideration," *IEEE Trans. Commun.*, vol. 62, no. 8, pp. 2956–2976, Aug. 2014.

[42] Y. Cui, L. Wang, X. Wang, H. Wang, and Y. Wang, "FMTCP: A fountain code-based multipath transmission control protocol," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 465–478, Apr. 2015.

[43] S.-K. Lee, Y.-C. Liu, H.-L. Chiu, and Y.-C. Tsai, "Fountain codes with PAPR constraint for multicast communications," *IEEE Trans. Broadcast.*, vol. 57, no. 2, pp. 319–325, Jun. 2011.

[44] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP westwood: End-to-end congestion control for wired/wireless networks," *Wireless Netw.*, vol. 8, no. 5, pp. 467–479, Sep. 2002.

[45] F. Yang, Q. Wang, and P. D. Amer, "Out-of-order transmission for in-order arrival scheduling for multipath TCP," in *Proc. IEEE WAINA*, Victoria, BC, Canada, pp. 749–752, May 2014.

[46] C. Raiciu, M. Handly, and D. Wischik, "Coupled congestion control for multipath transport protocols," IETF, Fremont, CA, USA, RFC 6356, Oct. 2011.

[47] S. Bregni, D. Caratti, and F. Martignon, "Enhanced loss differentiation algorithms for use in TCP sources over heterogeneous wireless networks," in *Proc. IEEE GLOBECOM*, vol. 2. San Francisco, CA, USA, 2003, pp. 666–670.
[48] W. John, "Characterization and classification of Internet backbone traffic," Ph.D. dissertation, Dept. Comput. Sci. Eng., Chalmers Univ. Technol., Göteborg, Sweden, Feb. 2010.

Changqiao Xu received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences (ISCAS), in 2009. He was an Assistant Research Fellow with ISCAS from 2002 to 2007, where he was a Research and Development Project Manager in the area of communication networks. From 2007 to 2009, he was a Researcher with the Software Research Institute, Athlone Institute of Technology, Athlone, Ireland. He joined the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2009. He is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology, and the Director of the Next Generation Internet Technology Research Center, BUPT. He has published over 100 technical papers in prestigious international journals and conferences. His research interests include innovative transport protocols, wireless networking, multimedia communications, and future Internet technology. He serves as the Co-Chair and the Technical Program Committee Member for a number of international conferences and workshops. He currently serves as the Co-Chair for the IEEE MMTC Interest Group "Green Multimedia Communications" and the Board Member of the IEEE MMTC Services and Publicity.

Peng Wang received the B.S. degree from the School of Information Engineering, China University of Geosciences, Beijing, in 2014. He is currently pursuing the M.S. degree with the Institute of Network Technology, Beijing University of Posts and Telecommunications. His research interests include multipath transmission protocol, network coding, and multimedia transmission over heterogeneous wireless networks.

Chunshan Xiong received the Ph.D. degree from the Control Science and Control Engineering Department, Huazhong University of Science and Technology, China, in 2000. He is a Senior Standard and Research Engineer with Huawei Technologies Ltd. He was one of the key contributors for the 3GPP 4G EPS core network architecture standard. His main interests include 3GPP next generation network architecture, mobile edge computing, mobility management, mobile CDN, mobile video, QoS, security and standards in 3GPP, ETSI, and IETF.

Xinpeng Wei received the master's degree from the Mechanical Engineering and Automation Department, Harbin Institute of Technology, China, in 2011. He is a Senior Wireless Network Research Engineer with Huawei Technologies. He holds over 30 patents on different network research areas. His research work mainly focuses on transport protocol, mobility management, Internet of Things, and security areas.

Gabriel-Miro Muntean received the Ph.D. degree from Dublin City University (DCU), Ireland, in 2003, researching on quality-oriented adaptive multimedia streaming. He is a Senior Lecturer with the School of Electronic Engineering, DCU, the Co-Director of the DCU Performance Engineering Laboratory, and a Consultant Professor with the Beijing University of Posts and Telecommunications, China. He has published over 250 papers in prestigious international journals and conferences, has authored three books and 16 book chapters, and has edited six other books. His research interests include quality-oriented and performance-related issues of adaptive multimedia delivery, performance of wired and wireless communications, energy-aware networking, and personalized technology-enhanced learning. He is an Associate Editor of the IEEE TRANSACTIONS ON BROADCASTING, an Editor of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and a Reviewer for other important international journals, conferences, and funding agencies. He is a Coordinator of the Horizon 2020 EU Project NEWTON. He is a member of the IEEE Broadcast Technology Society.