# End-to-End Coding for TCP

**Yong Cui, Lian Wang, Xin Wang, Yisen Wang, Fengyuan Ren, and Shutao Xia**

**Abstract**

Although widely used, TCP has many limitations in meeting the throughput and latency requirements of applications in wireless networks, high-speed data center networks, and heterogeneous multi-path networks. Instead of relying purely on retransmission upon packet loss, coding has potential to improve the performance of TCP by ensuring better transmission reliability. Coding has been verified to work well at the link layer but has not been fully studied at the transport layer. There are many advantages but also challenges in exploiting coding at the transport layer. In this article, we focus on how to leverage end-to-end coding in TCP. We reveal the problems TCP faces and the opportunities coding can bring to improve TCP performance. We further analyze the challenges faced when applying the coding techniques to TCP and present the current applications of coding in TCP.

TCP is the most important transport protocol today, providing in-order reliable transmission of byte streams. However, current network characteristics and application requirements are far different from those when TCP was initially designed. TCP has limitations in many popular networks such as wireless networks, high-speed networks, and heterogeneous multi-path networks.

TCP suffers from the problems of low throughput and high latency due to its use of loss-based congestion control and retransmission-based loss recovery schemes. Coding is promising in improving the performance of TCP by introducing redundant information to encode packets for forward error correction (FEC). This allows for faster loss recovery for TCP, which helps to alleviate the issues associated with the throughput, latency, and receiving buffer constraints. Although FEC has been widely studied at the link layer as a complement to automatic repeat request (ARQ), it has not been well explored yet in TCP.

There are many challenges to apply coding with TCP. Coding can conflict with TCP design and deployment, and compromise TCP friendliness. The data control and reliability of TCP are tightly coupled with the congestion control in byte-oriented sequence number management. The amount of redundant information added for coding needs to be determined carefully. There is a trade-off between reducing the coding overhead and improving the transmission reliability at the cost of higher transmission redundancy. To enable the practical use of coding in new protocols, it is essential to ensure backward compatibility for incremental deployment.

In this article, we explore the possibilities of designing a practical transport protocol with better performance employing coding technologies. We first present the limitations of the standard TCP, and explain how coding can be used in TCP. We then analyze the opportunities and challenges of applying coding in TCP, and discuss the strengths and weaknesses of existing solutions. Finally, we conclude the work with a brief summary.

*Young Cui, Lian Wang, Yisen Wang, Fengyuan Ren, and Shutao Xia are with Tsinghua University.*

*Xin Wang is with State Universitiy of New York at Stony Brook.*

## End-to-End Coding for TCP: Opportunities

TCP has been observed to be ill suited for many emerging applications. We first analyze the problems from the perspective of TCP design, and then present the opportunities that coding technologies can bring to alleviate the problems.

### Limitations of TCP

The inadequacy of standard TCP in wireless networks, high-speed data center networks (DCNs), and heterogeneous multi-path networks has been extensively documented. Here we list three situations where coding has potential to improve the TCP performance.

First, packet loss may result in TCP throughput much lower than the available path capacity. TCP regards losses as indications of congestion and halves its congestion window in fast retransmission or resets to the initial value upon timeout. However, packet losses may be caused by reasons other than congestion. For example, TCP can mistakenly take the losses due to bad link conditions as congestion signals and reduce its window size unexpectedly although the path is uncongested. This significantly affects the performance as the throughput of TCP is approximately inversely proportional to the square root of the packet loss rate [1].

In addition, TCP does not differentiate between congestion levels, and reacts sharply to transient and mild congestion. For high bandwidth and high delay networks, it will take a long time to reach the available bandwidth after a loss. It is reported that a single packet loss in 10,000 is enough to reduce TCP throughput to a third over a 50 ms gigabit link, and one loss in 1000 leads the throughput to drop by an order of magnitude [2].

Second, delay-sensitive applications (i.e., real-time or interactive applications) suffer from slow recovery of lost packets in TCP. Packet losses are detected by either triple duplicate acknowledgments or retransmission timer expiration. Although the first method targets fast lost packet retransmission, the recovery delay is still high as at least three packets

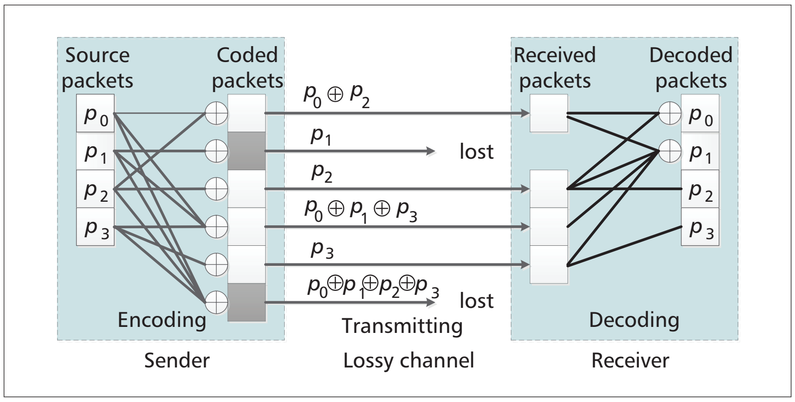

Figure 1. Example of coding for TCP.

after the loss need to be received to confirm loss. In addition, the measurement results show that most losses occur at tails for latency-sensitive flows and are repaired through expensive retransmission timeouts (RTOs) [3]. The minimum value of an RTO is normally set to 200 ms, which is too long for low-latency networks such as DCNs.

The packets behind the lost one(s) cannot be properly delivered to applications until the losses are recovered, thus hurting the quality of user experience of latency-sensitive applications. It is well known that web latency inversely correlates with revenue and profit. For instance, Amazon estimates that every 100 ms increase in latency cuts profits by 1 percent [3]. Thus, a faster loss recovery mechanism is desired.

The third issue is the buffer limitation as a result of out-of-order delivery. When a packet is lost or delayed, the subsequent packets need to be stored in the receiving buffer of TCP. Once the receiving buffer is fully occupied by out-of-order packets, the advertising window decreases to zero, which triggers the flow control mechanism driven by receivers. As a result, data transmissions are blocked even when the link is unoccupied, which not only reduces the throughput but also introduces a large delay.

The receiving buffer size is usually recommended to be set to twice the bandwidth-delay product (BDP), which allows the transmission to continue during fast retransmission [4]. However in heterogeneous multi-path networks, even if the BDP of every single path is small, the buffer size can still be a problem. To make full use of such heterogeneous networks, a multi-path extension of TCP, called Multi-Path TCP (MPTCP) [4], has been proposed to build multiple subflows in one connection. The subflows generally go through different paths and are transmitted concurrently. The BDP of MPTCP is calculated as the product of the aggregate bandwidth and the maximum round-trip time (RTT) of different paths, which is often much larger than that of single-path TCP. As the RTTs and loss rates of subflows can be very different, especially in heterogeneous networks, the buffer size could be set too large in MPTCP.

### Coding for TCP

Coding is promising to solve the problems mentioned above as it can provide better reliability and loss tolerance and faster loss recovery for TCP. The usefulness of coding for TCP was first investigated and verified by G. Karlsson *et al.* [5].

The main function of coding for TCP is to perform FEC. With FEC, a sender encodes its messages in a redundant way by using an error correction code (ECC), and the receiver can recover the lost packets if enough coded packets are received without waiting for retransmission. FEC is not currently used in default TCP. To incorporate FEC into TCP, coding is not applied at the bit level, but at the packet level.

Coding at the packet level is commonly referred to as network coding. The basic idea of network coding is to mix data from different packets across time and across flows into one packet. It has now been used at various layers including the application layer, transport layer, network layer, and even link layer. The coded packets are usually linear combinations of the source packets. For example, as illustrated in Fig. 1, an unlimited number of random linear combinations of the four source packets can be generated and sent over lossy channels. When four arbitrary linear independent coded packets are received, the four source packets can be recovered.

Applying coding in TCP has several advantages. First, coding in TCP is complementary to approaches such as multipath opportunistic routing, which exploits the broadcast nature of the wireless medium. Second, it is unnecessary to make changes in network devices such as routers, switches, and network interfaces. Third, to implement coding, the option field of TCP can be leveraged to negotiate whether coding can be used. Moreover, to help better schedule the coding process, network condition parameters such as RTT, congestion level, loss rate, and bandwidth can be estimated in TCP. Finally, the socket buffers of TCP can be used directly to store data for encoding and decoding.

## End-to-end Coding for TCP: Challenges

The design and implementation of coding techniques for TCP need to address several issues pertaining to usability and effectiveness. We summarize the five main design issues in Table 1, including TCP friendliness, congestion control, redundancy tuning, coding overhead, and compatibility. They can be used as the evaluation criteria for the design of coded TCP variants.

### TCP Friendliness

It is important to ensure fair sharing of network resources among different flows. With the widespread deployment of TCP, the discussion of fairness for transport protocols moves toward *TCP-friendly*, which means that new transport protocols should behave like an ordinary TCP flow under congestion conditions and share no more bandwidth than a TCP flow when competing for the same link [6]. To ensure TCP-friendly transmissions, a flow should not send more packets than an ordinary TCP flow under comparable conditions. On the other hand, coding schemes introduce redundancy to recover the lost packets. There is a trade-off when enforcing coded TCP variants to be friendly with standard TCP. When the loss rate is high, coded TCP will generally have higher goodput while standard TCP is kept busy with retransmissions; however, the goodput of coded TCP may be lower when the loss rate is low. Coded TCP can enjoy a higher benefit and can be more flexible to apply in a new network environment not dominated by conventional TCP, such as in the popular DCNs.

### Congestion Control

Congestion control algorithms play an important role in TCP. The Internet Engineering Task Force (IETF) standardized TCP congestion control algorithm is based on Reno, which takes loss as the signal of congestion. Other types of delay-based congestion control algorithms such as Vegas and Westwood monitor congestion based on estimation of RTT. Introducing coding into TCP may impact congestion detection (loss detection or RTT estimation) to different extents depending on the design of codes and acknowledgment methods. As coding can mask packet losses, it may

| Challenges | Classifications | Factors |
|---|---|---|
| TCP friendliness [6] | Steady state<br>Transient state | Fairness<br>Aggressiveness and responsiveness |
| Congestion control [7, 8] | Loss-based (Reno)<br>Delay-based (Vegas/Westwood)<br>ECN-based | Differentiate wireless loss from congestion loss<br>Fairness<br>Network compatibility |
| Redundancy tuning [3, 9] | Proactive<br>Reactive | Packet loss detection and loss model prediction<br>Feedback-based scheduling |
| Coding overhead [7, 9] | Time cost<br>Bandwidth cost<br>Computation cost | Code complexity and data arriving rate<br>Redundancy setting and coding header for synchronization<br>Code complexity and code implementation |
| Compatibility [4, 10] | Network compatibility<br>Application compatibility | Middleboxes<br>Socket extensions |

Table 1. Challenges of applying coding in TCP.

seriously affect loss detection. Relatively speaking, RTT estimation is easier to be maintained and has fewer restrictions on the use of coding.

### Redundancy Tuning

The use of coding helps to recover lost data more quickly at the cost of higher bandwidth consumption. Decoding will fail if the number of redundant packets is smaller than that of the lost ones, in which case coding-based recovery falls back to acknowledgment-based recovery and provides little benefit. To make full use of bandwidth as well as recover data immediately, coding rate should be tuned properly. Precise estimation and prediction of loss rate can help determine the appropriate coding rate proactively; however, the prediction is difficult for a network with varying conditions. On the other hand, timely and adequate feedback is useful for adjusting the coding rate reactively. Moreover, due to the buffer or delay limit, block codes (also referred to as batch-based codes) are often used. To reduce the latency, it may help to increase the redundancy and carefully select an appropriate block size.

### Coding Overhead

Coding would naturally introduce extra overhead. First, it takes time to perform encoding and decoding, which introduces extra latency. For latency-sensitive applications, coding latency cannot be ignored. Second, the redundancy introduced by coding costs extra bandwidth. Enough redundant packets are needed to compensate for lost ones. In addition, for protocols to deliver the coding information, additional bandwidth will incur cost to transmit a coding header to indicate the information of which the coded packet consists and how many coded symbols are received. Third, the coding process uses computation resources and may induce energy problems for power-constrained mobile terminals. Coding can be useful only when its gain is larger than the overheads.

### Compatibility

For TCP extensions, compatibility is also an issue. The compatibilities mainly include application compatibility, TCP compatibility, and network compatibility [4]. Application compatibility can be realized through socket extensions. TCP compatibility is included in TCP friendliness and has been discussed separately. For network compatibility, coded TCP is expected to work transparently in the presence of various middleboxes in the Internet, which may be difficult to handle. The compatibility requirements make the design of coded TCP variants more complex [10].

## Applications

There have been some research effort on exploiting end-to-end coding to improve TCP performance. In this section, we review the work on three typical categories of applications: improving the throughput of wireless networks, reducing the latency caused by loss, and solving the out-of-order problem of MPTCP in heterogeneous multi-path networks. The main characteristics of the presented work are summarized and compared in Table 2.

### Improve Throughput of Wireless Networks

In wireless networks, the dynamic and uncertain transmission medium makes loss a common problem. Due to the loss-based congestion control used in TCP, wireless losses can cause sending rate reduction and consequent significant throughput degradation even when the network is not congested. Schemes to solve this problem can be classified into three broad categories: end-to-end protocols to recover the loss at senders; link-layer protocols to provide local transmission reliability; and split-connection protocols that break the end-to-end connection into two parts in intermediate nodes. While end-to-end schemes are not as effective as local recovery techniques in handling wireless losses, they are promising since significant gains can be achieved without requiring extensive support at the network and link layers at routers and base stations [5].

The basic idea of end-to-end coding is to add redundancy to facilitate loss recovery. Given a certain loss probability, an amount of redundancy is injected into the transmission flow such that there are no more lost packets than redundant ones. Actually, the loss rate of a path may be very dynamic and cannot be precisely estimated. L. Baldantoni *et al.* [5] developed an adaptive algorithm to choose the optimal number of redundancy packets considering the connection environment. The RTT is estimated in a normal way, while the bandwidth and loss rate are estimated in a similar way through explicit feedback from the receiver. Experiments show that TCP with coding outperforms the existing TCP (New Reno and SACK) over a large range of loss probability and propagation delay. Note that end-to-end coding also recovers losses caused by congestion; hence, Reno-based congestion control becomes invalid.

A complete protocol design of a coded TCP variant called TCP/NC is presented from theory to implementation in [7]. TCP/NC introduced a new acknowledgment mechanism based on "degree of freedom" (i.e., a linear combination that reveals one unit of new information) to incorporate coding into the control algorithm. The receiver acknowledges every degree of freedom even if this is not done for a new packet immediately. For easier deployment, TCP/NC imports a new network

| | Year | Goal | Networks | TCP-friendly | Congestion control | Path estimation | Rateless | Retransmit | Middle-boxes |
|---|---|---|---|---|---|---|---|---|---|
| TCP/NC [7] | 2011 | Throughput | Wireless | Aggressive | Vegas | Yes | Yes | No | No |
| A-FEC [5] | 2004 | Throughput | Wireless | Aggressive | Reno-based | Yes | Yes | Yes | No |
| LT-TCP [8] | 2005 | Throughput | Wireless | Aggressive | Reno-based | Yes | Yes | Yes | No |
| Corrective [3] | 2013 | Latency | DCN | Slightly aggressive | Reno-based | No | No | Yes | Yes |
| Redundancy [11] | 2012 | Latency | — | — | — | — | — | — | — |
| Repflow [12] | 2014 | Latency | — | — | — | — | — | — | — |
| FMTCP [9] | 2014 | Both | Multi-path | Friendly | Reno-based | Yes | Yes | No | Yes |
| NC-MPTCP [13] | 2012 | Both | Multi-path | Friendly | Reno-based | Yes | Yes | No | No |
| SC-MPTCP [14] | 2014 | Both | Multi-path | Friendly | Reno-based | Yes | Yes | No | No |

Table 2. Applications of coding for TCP.

coding (i.e., NC) layer between the transport layer and the network layer to minimize changes to protocols. The network coding layer conducts buffering and encoding, decoding, and acknowledgment at the sender and receiver, respectively.

The authors first use TCP-Vegas as it is more compatible with their modifications. Furthermore, a real-world implementation with TCP-Reno is presented. Both obtain significant gains in throughput over standard TCP in the presence of lossy network links. The fairness of TCP is maintained in the absence of losses (i.e., the loss rate is set to 0, and no redundancy packet is added), while TCP friendliness is still harmed in lossy networks. TCP/NC adds a coding header on top of the TCP header to communicate packet combination, which damages the TCP/IP header and may conflict with middleboxes that read TCP headers. Moreover, the coding header introduces a large overhead (i.e., when the coding window has $n$ packets, every coded packet needs to carry $4n + 7$ extra bytes at the header). Besides, TCP/NC adopts random linear coding and decodes by performing Gaussian elimination, which has high computation complexity, and the latency introduced by coding has not been considered yet. Finally, the estimation of the various parameters of TCP/NC (e.g., the loss rate) according to the connection environment are not mentioned.

The above two methods solved the wireless loss problem but cannot react to congestion effectively. To differentiate between wireless loss and congestion, LT-TCP [8] exploits ECN to assist coding in TCP. Timeout effects due to packet erasures are combated using a dynamic and adaptive FEC scheme that includes adaptation of TCP's maximum segment size. Proactive and reactive FEC overhead enhance TCP SACK to protect original segments and retransmissions, respectively. LT-TCP improves TCP performance even for packet loss rates up to 30 percent.

## Reducing Latency Caused by Loss

As mentioned above, timely transmission is critical for some web flows and data center flows. Web flows play increasingly important roles with the popularity of mobile applications, while web service providers are still troubled by the latency problem. For data center networks, the partition-aggregation workflow pattern requires that every flow have low latency in order to provide real-time performance to users.

T. Flach *et al*. [3] performed a measurement study of billions of latency-sensitive TCP connections and found that flows with packet loss take on average five times longer to complete than those without any loss due to the loss detection and recovery mechanism of TCP. Measurements show that most losses occur at the tail of a transmission burst, which does not have enough acknowledgments to trigger fast retransmission; thus, losses are recovered through expensive retransmission timeouts (RTOs).

Unnecessary round-trips or RTOs during loss recovery have hampered further latency improvement as the RTT required between clients and servers largely determines the overall latency of most latency-sensitive flows. Simply reducing the interval of RTO does not address the latency problem due to frequent spurious timeouts and window reductions. Thus, they see the demand for and potentiality of reducing latency through the use of coding with TCP.

FEC coding is promising in reducing the delivery latency of TCP flows by providing loss recovery without introducing round-trip delay. As latency-sensitive flows are generally short, the bandwidth overhead introduced by coding can be neglected. T. Flach *et al.* [3] proposed a low-overhead mechanism referred to as *corrective* to cooperate FEC with coding. *Corrective* adds a coded packet for each window of packets, which has low bandwidth overhead. The coding is done by XORing the payload of packets in the window, which has low computation overhead.

*Corrective* is designed as a TCP extension and uses TCP options to deliver the number of bytes that the payload encodes. Packet losses that are recovered with coded packets will be indicated explicitly using a corrective option along with acknowledgments. *Corrective* can recover a single packet loss in one window. If more than one packet is lost, the receiver will notify the sender with a corrective option to trigger fast recovery. Experiments show that *Corrective* significantly reduces the average latency and the tail of latency.

*Corrective* is mainly designed for congestion loss in the data center environment, where loss rate is relatively low. In wireless transmission scenarios, higher redundancy is needed to recover more than one loss in one window. With the goal of lower latency, the codes must have low complexity.

Besides exploiting coding to reduce the latency by adding redundant transmissions, replicated flows can also be transmitted. In data center scenarios, the demand for consistent low latency outweighs the need to save the bandwidth. Replication of flows is the main method to apply to improve transmission reliability [11, 12]. Also in [3], T. Flach *et al*. proposed *proactive*, which takes the aggressive stance of proactive transmissions of copies of each TCP segment.

Unlike replication, coding improves reliability via data redundancy, which is more flexible in a redundancy setting and can use bandwidth more efficiently. Moreover, with packet-level equal cost multi-path (ECMP) routing, coding can leverage path redundancy after eliminating the out-of-order problem.

### Heterogeneous Multi-Path Networks

After much discussion, the idea of concurrent multi-path transferring has finally been standardized as a multi-path extension of TCP (MPTCP) [4]. MPTCP aims to make full use of the capacity of heterogeneous networks concurrently. But we need to solve the problem of out-of-order delivery to meet its potential.

MPTCP connections have one or more TCP subflows, each of which behaves like one TCP flow. The subflows of one MPTCP connection usually follow different paths in heterogeneous multi-path networks, and experience different RTT and loss rates. If one packet is lost on the slowest path, the receiver window cannot advance until the lost packet is recovered. The duration from sending packets to receiving the acknowledgment will be $2RTT$ if it is recovered through fast recovery and $RTT + RTO$ if it is recovered through timeout retransmission. During loss recovery, if the receiver buffer is full, the subflows cannot send new packets even if their links are unoccupied. Thus, fast and reliable flows would be blocked by bad flows.

Large enough receiver buffers can avoid the buffer blocking problem mentioned above. Regardless of the timeout, the minimum receiver buffer size to avoid blocking should be at least twice the product of the sum of bandwidths of all subflows and the RTT of the slowest subflow (i.e., $2(\Sigma_i BW_i)RTT_{max}$), which is recommended by MPTCP [4]. When high bandwidth paths coexist with high latency paths, this buffer requirement can be very large. Moreover, even with a sufficiently large receiver buffer, significant additional delay will be introduced due to reordering out-of-order segments from different paths [15]. Another solution is to limit the usage of slow paths, which may go against the bandwidth aggregation feature of MPTCP and discourage full utilization of network resources.

Y. Cui *et al.* [9] proposed a rateless-code-based MPTCP, called FMTCP, to address this problem. The main idea is to leverage the sequence-agnostic properties of rateless coding, where coded packets in the same block are sequence-agnostic and relatively independent of each other. Thus, with the use of coding, what matters is the percentage of packets lost rather than which packets are lost. FMTCP incorporated low-complexity fountain codes into MPTCP and included a packet scheduling algorithm based on the estimation of the delivery time of each subflow.

M. Li *et al.* proposed to introduce network coding to parts of the subflows (NC-MPTCP) in [13] and then further explored the usage of systematic coding (SC-MPTCP) in [14]. They also adopted block coding and proposed sending a certain number of coded redundant packets at the beginning of a block. With systematic coding, the original packets are sent directly without modification. To deal with underestimation of proactive redundancy, they proposed a pre-blocking warning mechanism to retrieve reactive redundancy from the sender to further avoid transmission blocking. When a subflow gets packets in order in the subflow-receiving queue but finds no space in the shared connection-level buffer, it signals the sender to transmit the first unacknowledged packet on this path.

NS-based simulations of these works show that coding is effective in improving the performance of MPTCP. Computation overhead and coding delay may reduce the benefits of coding in real-world implementations; however, they are not considered in the simulations due to the limitation of the simulation methodology. Implementations and experimental results are needed to further explore the advantages and disadvantages of coding in MPTCP.

## Conclusion and Future Prospects

In this article, we present several limitations of TCP including low throughput in wireless networks, high latency due to loss, and the out-of-order serving problem of multi-path TCP in heterogeneous multi-path networks. In these situations, coding serves as a complementary or alternative loss recovery method of TCP. We analyze what benefits coding can bring, discuss the opportunities and challenges of incorporating end-to-end FEC into TCP, and summarize the state of the art in this field of research.

Although there are a number of studies on applying coding in TCP, several problems remain open or have not been well addressed:

- First, the necessity of ensuring coded TCP variants to be TCP-friendly is disputable. TCP friendliness is important for TCP variants to work together with standard TCP, but may not be essential for networks where TCP can be entirely replaced. In this case, fair bandwidth sharing is enough.
- The impact of coding on congestion control should be carefully considered as coding may impact the congestion detection. A better protocol design is needed to decouple the coding and the congestion control, or new congestion control algorithms should work efficiently for coded TCP variants.
- It is difficult to trade off between the efficiency in bandwidth usage and the decoding probability. To choose an appropriate coding rate, it helps to accurately estimate the link quality and adapt the redundancy level in the coding. There is also a lack of codes that provide good loss tolerance while having low redundancy, especially when the flows are small.
- Comprehensive analysis and modeling are needed to indicate when coding is useful for TCP. This article has only discussed part of the applications where coding can facilitate TCP transmissions, such as in heterogeneous multi-path networks. New scenarios on the use of coding with TCP can be further explored.

### Acknowledgment

This work is supported by National Natural Science Foundation of China (NO. 61422206, 61120106008, 61371078), NSF grants CNS 1247924, and ECCS 1408247.

### References

[1] J. Padhye *et al.*, "Modeling TCP throughput: A Simple Model and Is Empirical Validation," *Proc. ACM SIGCOMM Comp. Commun. Rev.*, vol. 28, no. 4. 1998, pp. 303–14.
[2] M. Balakrishnan *et al.*, "Maelstrom: Transparent Error Correction for Communication Between Data Centers," *IEEE/ACM Trans. Networking*, vol. 19, no. 3, 2011, pp. 617–29.
[3] T. Flach *et al.*, "Reducing Web Latency: The Virtue of Gentle Aggression," *Proc. ACM SIGCOMM 2013*, 2013, pp. 159–70.
[4] A. Ford *et al.*, "Architectural Guidelines For Multipath TCP Development," IETF Informational RFC, vol. 6182, 2011, pp. 2070–2721.
[5] L. Baldantoni, H. Lundqvist, and G. Karlsson, "Adaptive End-to-End FEC For Improving TCP Performance Over Wireless Links," *Proc. 2004 IEEE ICC*, vol. 7, 2004, pp. 4023–27.
[6] S.-C. Tsao, Y.-C. Lai, and Y.-D. Lin, "Taxonomy and Evaluation of TCP-Friendly Congestion-Control Schemes on Fairness, Aggressiveness, and Responsiveness," *IEEE Network*, vol. 21, no. 6, 2007, pp. 6–15.
[7] J. K. Sundararajan et al., "Network Coding Meets TCP: Theory and Implementation," *Proc. IEEE*, vol. 99, no. 3, 2011, pp. 490–512.

[8] O. Tickoo *et al.*, "LT-TCP: End-to-End Framework to Improve TCP Performance over Networks with Lossy Channels," *Proc. IWQoS 2005*, Springer, 2005, pp. 81–93.
[9] Y. Cui *et al.*, "FMTCP: A Fountain Code-Based Multipath Transmission Control Protocol," *IEEE/ACM Trans. Networking*, vol. 23, no. 2, 2015, pp. 465–78.
[10] M. Honda *et al.*, "Is it Still Possible To Extend TCP?" *Proc. 2011 ACM SIGCOMM Internet Measurement Conf.*, ACM, 2011, pp. 181–94.
[11] A. Vulimiri *et al.*, "More is Less: Reducing Latency via Redundancy," *Proc. 11th ACM Workshop on Hot Topics in Networks*, 2012, pp. 13–18.
[12] H. Xu and B Li, "Repflow: Minimizing Flow Completion Times with Replicated Flows in Data Centers," *IEEE INFOCOM 2014*, 2014.
[13] M. Li, A. Lukyanenko, and Y. Cui, "Network Coding Based Multipath TCP," *Proc. IEEE INFOCOM Wksps. 2012*, 2012, pp. 25–30.
[14] M. Li *et al.*, "Tolerating Path Heterogeneity in Multipath TCP with Bounded Receive Buffers," *Computer Networks*, vol. 64, 2014, pp. 1–14.
[15] Y.-C. Chen *et al.*, "A Measurement-Based Study of Multipath TCP Performance over Wireless Networks," *Proc. 2013 Internet Measurement Conf.*, ACM, 2013, pp. 455–68.

## Biographies

YONG CUI [M] (cuiyong@tsinghua.edu.cn) received his B.E. and Ph.D. degrees in computer science and engineering from Tsinghua University, China, in 1999 and 2004, respectively. He is currently a full professor at Tsinghua University, and Co-Chair of the IETF IPv6 Transition WG Softwire. His major research interests include mobile wireless Internet and computer network architecture. Having published more than 100 papers in refereed journals and conferences, he received the National Award for Technological Invention in 2013, and the Influential Invention Award of China Information Industry in both 2012 and 2004. Holding more than 40 patents, he has authored 3 Internet standard documents, including RFC 7040 and RFC 5565, for his proposal on IPv6 transition technologies. He serves on the Editorial Boards of *IEEE TPDS* and *IEEE TCC*.

LIAN WANG (wanglian.cst@gmail.com) received her B.E. degree in computer science and engineering from Tsinghua University in 2008. She is now a Master's student in the Department of Computer Science and Technology, Tsinghua University. Her supervisor is Prof. Yong Cui. Her research interests include rateless coding and mobile wireless Internet.

XIN WANG [M] (xwang@ece.sunysb.edu) received her B.S. and M.S. degrees in telecommunications engineering and wireless communications engineering, respectively, from Beijing University of Posts and Telecommunications, China, and her Ph.D. degree in electrical and computer engineering from Columbia University, New York. She is currently an associate professor in the Department of Electrical and Computer Engineering, State University of New York at Stony Brook. Her research interests include algorithm and protocol design in wireless networks and communications, mobile and distributed computing, as well as networked sensing and detection. She has served on the Executive Committees and Technical Committees of numerous conferences and funding review panels, and is a referee for many technical journals. She achieved the NSF career award in 2005, and the ONR challenge award in 2010.

YISEN WANG (eewangyisen@gmail.com) received his B.E. degree in electrical engineering from South China University of Technology in 2014. He is now a Ph.D. candidate in the Department of Computer Science and Technology, Tsinghua University. His supervisor is Prof. Shutao Xia. His research interests include coding theory and networking.

FENGYUAN REN [M] (renfy@tsinghua.edu.cn) received his B.A and M.Sc. degrees in automatic control from Northwestern Polytechnic University, China, in 1993 and 1996, respectively. In December 1999, he obtained his Ph.D. degree in computer science from Northwestern Polytechnic University. He is currently a professor in the Department of Computer Science and Technology at Tsinghua University. From 2000 to 2001, he worked in the Electronic Engineering Department of Tsinghua University as a postdoctoral researcher. In January 2002, he moved to the Computer Science and Technology Department of Tsinghua University. His research interests include network traffic management and control, control in/over computer networks, wireless networks, and wireless sensor networks. He has (co)-authored more than 80 international journal and conference papers. He has served as a Technical Program Committee member and Local Arrangement Chair for various IEEE and ACM international conferences.

SHUTAO XIA (xiast@sz.tsinghua.edu.cn) received his B.S. degree in mathematics and Ph.D. degree in applied mathematics from Nankai University, Tianjin, China, in 1992 and 1997, respectively. Since January 2004, he has been with the Graduate School of Shenzhen of Tsinghua University, where he is currently a professor. From March 1997 to April 1999, he was with the Research Group of Information Theory, Department of Mathematics, Nankai University. From September 1997 to March 1998 and from August to September 1998, he visited the Department of Information Engineering, Chinese University of Hong Kong. His current research interests include coding theory, information theory, and networking. He has published more than 40 papers in *IEEE Transactions on Information Theory* and other core domestic journals.