TCP adaptation with network coding and opportunistic data forwarding in multi-hop wireless networks

Chen Zhang Corresp., [1], **Yuanzhu Chen** [1], **Cheng Li** [2]

[1] Department of Computer Science, Memorial University of Newfoundland, St. John's, Canada
[2] Department of Electrical and Computer Engineering, Memorial University of Newfoundland, St. John's, Canada

Corresponding Author: Chen Zhang
Email address: cz5670@mun.ca

Opportunistic data forwarding significantly increases the throughput in multi-hop wireless mesh networks by utilizing the broadcast nature of wireless transmissions and the fluctuation of link qualities. Network coding strengthens the robustness of data transmissions over unreliable wireless links. However, opportunistic data forwarding and network coding are rarely incorporated with TCP because the frequent occurrences of out-of-order packets in opportunistic data forwarding and long decoding delay in network coding overthrow TCP's congestion control. In this paper, we propose a solution dubbed TCPFender, which supports opportunistic data forwarding and network coding in TCP. Our solution adds an adaptation layer to mask the packet loss caused by wireless link errors and provides early positive feedbacks to trigger a larger congestion window for TCP. This adaptation layer functions over the network layer and reduces the delay of ACKs for each coded packet. The simulation results show that TCPFender significantly outperforms TCP/IP in terms of the network throughput in different topologies of wireless networks.

TCP Adaptation with Network Coding and Opportunistic Data Forwarding in Multi-Hop Wireless Networks

Chen Zhang[1]**, Yuanzhu Chen**[1]**, and Cheng Li**[2]

[1]**Department of Computer Science, Memorial University of Newfoundland**
[2]**Department of Electrical and Computer Engineering, Memorial University of Newfoundland**

ABSTRACT

Opportunistic data forwarding significantly increases the throughput in multi-hop wireless mesh networks by utilizing the broadcast nature of wireless transmissions and the fluctuation of link qualities. Network coding strengthens the robustness of data transmissions over unreliable wireless links. However, opportunistic data forwarding and network coding are rarely incorporated with TCP because the frequent occurrences of out-of-order packets in opportunistic data forwarding and long decoding delay in network coding overthrow TCP's congestion control. In this paper, we propose a solution dubbed TCPFender, which supports opportunistic data forwarding and network coding in TCP. Our solution adds an adaptation layer to mask the packet loss caused by wireless link errors and provides early positive feedbacks to trigger a larger congestion window for TCP. This adaptation layer functions over the network layer and reduces the delay of ACKs for each coded packet. The simulation results show that TCPFender significantly outperforms TCP/IP in terms of the network throughput in different topologies of wireless networks.

Keywords: TCP, Network Coding, Opportunistic Data Forwarding, Multi-Hop Wireless Networks

1 INTRODUCTION

Wireless mesh networks have emerged as the most common technology for the last mile of Internet access. The Internet provides a platform for rapid and timely information exchanges among clients and servers. Transmission Control Protocol (TCP) has become the most prominent transport protocol on the Internet. Since TCP was originally designed primarily for wired networks that have low bit error rates, moderate packet loss, and packet collisions, the performance of TCP degrades to a greater extent in multi-hop wireless networks, where several unreliable wireless links may be involved in data transmissions (Aguayo et al., 2004; Jain and Das, 2005). However, multi-hop wireless networks have several advantages, including rapid deployment with less infrastructure and less transmission power over multiple short links. Moreover, a high data rate can be achieved by novel cooperation or high link utilization (Larsson, 2001). Some important issues are being addressed by researchers to utilize these capabilities and increase TCP performance in multi-hop wireless networks, such as efficiently searching the ideal path from a source to a destination, maintaining reliable wireless links, protecting nodes from network attacks, reducing energy consumption, and supporting different applications.

In multi-hop wireless networks, data packet collision and link quality variation can cause packet losses. TCP often incorrectly assumes that there is congestion, and therefore reduces the sending rate. However, TCP is actually required to transmit continuously to overcome these packets losses. As a result, such a problem causes poor performance in multi-hop wireless networks. There are extensive studies working on these harmful effects. Some studies were proposed to reduce the collision between TCP data packets and TCP acknowledgements or dynamically adjust the congestion window. Other relief may come from network coding. The pioneering paper proposed by Ahlswede et al. (2000) presents the fundamental theory of network coding. Instead of forwarding a single packet at each time, network coding allows nodes to recombine input packets into one or several output packets. Furthermore, network coding is also very well suited for environments where only partial or uncertain data is available for making a decision (Mehta and Narmawala, 2011).

The link quality variation in multi-hop wireless networks is widely studied in the opportunistic data forwarding under User Datagram Protocol (UDP). It was traditionally treated as an adversarial factor in wireless networks, where its effect must be masked from upper-layer protocols by automatic retransmissions or strong forwarding error corrections. However, recent innovative studies utilize the characteristic explicitly to achieve opportunistic data forwarding (Biswas and Morris, 2005; Chen et al., 2009; Wang et al., 2012). Unlike traditional routing protocols, the forwarder in opportunistic routing protocols broadcasts the data packets before the selection of next-hop forwarder. Opportunistic routing protocols allow multiple downstream nodes as candidates to forward data packets instead of using a dedicated next-hop forwarder.

Since the broadcasting nature of wireless links naturally supports both network coding and opportunistic data forwarding, many studies work on improving UDP performance in multi-hop wireless networks by opportunistic data forwarding and network coding. However, opportunistic data forwarding and network coding are inherently unsuitable for TCP. The frequent dropping of packets or out-of-order arrivals overthrow TCP's congestion control. Specifically, opportunistic data forwarding does not attempt to forward packets in the same order as they are injected in the network, so the arrival of packets will be in a different order. Network coding also introduces long coding delays by both the encoding and the decoding processes; besides, it is possible along with some scenarios of not being able to decode packets. These phenomena introduce duplicated ACK segments and frequent timeouts in TCP transmissions, which reduce the TCP throughput significantly.

Our proposed protocol, called *TCPFender*, uses opportunistic data forwarding and network coding to improve TCP throughputs. TCPFender adds an adaptation layer above the network layer to cooperate with TCP's control feedback loop; it makes the TCP's congestion control work well with opportunistic data forwarding and network coding. TCPFender proposes a novel feedback-based scheme to detect the network congestion and distinguish duplicated ACKs caused by out-of-order arrivals in opportunistic data forwarding from those caused by network congestion. We compared the throughput of TCPFender and TCP/IP in different topologies of wireless mesh networks, and analyzed the influence of batch sizes on the TCP throughput and the end-to-end delay. Since our work adapts the TCPFender to functioning over the network layer without any modification to TCP itself, it is easy to deploy in wireless mesh networks.

2 RELATED WORK

2.1 Opportunistic data forwarding

ExOR (Extreme Opportunistic Routing) is a seminal effort in opportunistic routing protocols (Biswas and Morris, 2005). It is an integrated routing and MAC protocol that exploits the broadcast nature of wireless media. In a wireless mesh network, when a source transmits a data packet to a destination by several intermediate nodes which are decided by the routing module, other downstream nodes not in the routing path, can overhear the transmission. If the dedicated intermediate node, which is in the routing path, fails to receive this packet, other nearby downstream nodes can be scheduled to forward this packet instead of the sender retransmitting. In this case, the total transmission energy consumption and the transmission delay can be reduced, and the network throughput will be increased. Unfortunately, traditional IP forwarding dictates that all nodes without a matching receiver address should drop the packet, and only the node that the routing module selects to be the next hop can keep it for forwarding subsequently, so traditional IP forwarding is easily affected by link quality variation. However, ExOR allows multiple downstream nodes to coordinate and forward packets. The intermediate nodes, which are 'closer' to the destination, have a higher priority in forwarding packets towards the destination. ExOR can utilize the transient high quality of links and obtains an opportunistic forwarding gain by taking advantage of transmissions that reach unexpectedly far or fall unexpectedly short. In ExOR, a forwarding schedule is proposed to reduce duplicate transmissions. This schedule guarantees that only the highest priority receiver will forward packets to downstream nodes. However, this 'strict' schedule also reduces the possibilities for spatial reuse. The study in (Chachulski et al., 2007) shows that ExOR can have better spatial reuse of wireless media. Furthermore, this schedule may be violated due to frequent packet loss and packet collision.

PeerJ Comput. Sci. reviewing PDF | (CS-2016:05:10857:2:0:NEW 6 Sep 2016)

2.2 Opportunistic data forwarding with network coding

Studies show that network coding can reduce the data packet collision and approach the maximum theoretical capacity of networks (Ahlswede et al., 2000; Li et al., 2003; Koetter and Médard, 2003; Laneman et al., 2004; Jaggi et al., 2005; Ho et al., 2006). Many researchers incorporate network coding in opportunistic data forwarding to improve the throughput performance (Chachulski et al., 2007; Lin et al., 2008, 2010; Zhu et al., 2015). MORE (MAC-independent Opportunistic Routing and Encoding) is practical opportunistic routing protocol based on random linear network coding (Chachulski et al., 2007). In MORE, the source node divides data packets from the upper layer into batches and generates coded packets of each batch. Similar to ExOR, packets in MORE are also forwarded based on a batch. The destination node can decode these coded packets to original packets after receiving enough independently coded packets in the same batch. The destination receives enough packets when the decoding matrix reaches the full rank, then these original packets will be pushed to the upper layer. MORE coordinates the forwarding of each node using a transmission credit system, which is calculated based on how effective it would be in forwarding coded data packets to downstream nodes. This transmission credit system reduces the possibility that intermediate nodes forward the same packets in duplication. However, MORE uses a 'stop-and-wait' design with a single batch in transmission, which is not efficient utilizing the bandwidth of networks. COPE focuses on inter-session network coding; it is a framework to combine and encode data flows through joint nodes to achieve a high throughput (Basagni et al., 2008). CAOR (Coding Aware Opportunistic Routing) proposes a localized coding-aware opportunistic routing mechanism to increase the throughput of wireless mesh networks. In this protocol, the packet carries out with the awareness of coding opportunities and no synchronization is required among nodes (Yan et al., 2008). NC-MAC improves the efficiency of coding decisions by verifying the decodability of packets before they are transmitted (Argyriou, 2009). The scheme focuses on ensuring correct coding decisions at each network node, and it requires no cross-layer interactions.

CodeOR (Coding in Opportunistic Routing) improves MORE in a few important ways (Lin et al., 2008). In MORE, the source simply keeps transmitting coded packets belonging to the same batch until the acknowledgment of this batch from the destination has been received. CodeOR allows the source to transmit multiple batches of packets in a pipeline fashion. They also proposed a mathematical analysis in tractable network models to show the way of 'stop-and-wait' affects the network throughput, especially in large or long topology. The timely ACKs are transmitted from downstream nodes to reduce the penalty of inaccurate timing in transmitting the next batch. CodeOR applies the ideas of TCP flow control to estimate the correct sending window and the flow control algorithm is similar to TCP Vegas, which uses increased queueing delay as congestion signals. SlideOR works with online network coding (Lin et al., 2010), in which data packets are not required to be divided into multiple batches or to be encoded separately in each batch. In SlideOR, the source node encodes packets in overlapping sliding windows such that coded packets from one window position may be useful towards decoding the packets inside another window position. Once a coded packet is 'seen' by the destination node, the source node only encodes packets after this seen packet. Since it does not need to encode any packet that is already seen at the destination, SlideOR can transmit useful coded packets and achieve a high throughput.

CCACK (Cumulative Coded ACKnowledgment) allows nodes to acknowledge coded packets to upstream nodes with negligible overhead (Koutsonikolas et al., 2011). It utilizes a null space-based (NSB) coded feedback vector to represent the entire decoding matrix. CodePipe is a reliable multicast protocol, which improves the multicast throughput by exploiting both intra and inter network coding (Li et al., 2012). CORE (Coding-aware Opportunistic Routing mEchanism) combines inter-session and intra-session network coding (Krigslund et al., 2013). It allows nodes in the network to setup inter-session coding regions where packets from different flows can be XORed. Packets from the same flow uses random linear network coding for intra-session coding. CORE provides a solution to cope with the unreliable overhearing and improves the throughput performance in multi-hop wireless networks. NCOR focuses on how to select the best candidate forwarder set and allocate traffic among candidate forwarders to approach optimal routing (Cai et al., 2014). It contracts a relationship tree to describe the child-parent relations along the path from the source to the destination. The cost of the path is the sum of the costs of each constituent hyperlink for delivering one unit of information to the destination. The nodes, which create the path with the minimum cost, can be chosen as candidate forwarders. Hsu et al. (2015) proposed a stochastic dynamic framework to minimize a long-run average cost. They also analyzed the problem of whether to delay packet transmission in hopes that a coding pair will be available in the future or transmit a packet without coding. Garrido et al. (2015) proposed a cross-layer technique to balance the load between relaying nodes based on bandwidth of wireless links, and they used an intra-flow network coding solution modelled by means of Hidden Markov Processes. However, the schemes above were designed to utilize opportunistic data forwarding and network coding, but none of these was designed to support TCP.

2.3 Network coding in TCP

A number of recent papers have utilized network coding to improve TCP throughput. In particular, Huang et al. introduce network coding to TCP traffic, where data segments in one direction and ACK segments in the opposite direction can be coded at intermediate nodes (Huang et al., 2008). The simulation showed that making a small delay at each intermediate node can increase the coding opportunity and increase the TCP throughput. TCP/NC enables a TCP-compatible sliding-window approach to utilize network coding (Sundararajan et al., 2011). Such a variant of TCP is based on ACK-based sliding-window network coding approach and improves the TCP throughput in lossy links. It uses the degree of freedom in the decoding matrix instead of the number of received original packets as the sequence number in ACK. If a received packet increases the degree of freedom in the decoding matrix, this packet is called an innovative packet and this packet is 'seen' by the destination. The destination node will generate an acknowledgment whenever a coded packet is seen instead of producing an original packet. However, TCP/NC cannot efficiently control the waiting time for the decoding matrix to become full rank, and the packet loss can make TCP/NC's decoding matrix very large, which causes a long packet delay (Sun et al., 2015). TCP-VON introduces online network coding (ONC) to TCP/NC, which can smoothly increase the receiving data rate and packets can be decoded quickly by the destination node. However, these protocols are variants of RTT-based congestion control TCP protocols (e.g., Vegas), which limits their applications in practice since most TCP protocols are loss-based congestion control (Bao et al., 2012). TCP-FNC proposes two algorithms to increase the TCP throughput (Sun et al., 2015). One is a feedback based scheme to reduce the waiting delay. The other is an optimized progressive decoding algorithm to reduce computation delay. It can be applied to loss-based congestion control, but it does not take advantage of opportunistic data forwarding. Since TCP-FNC is based on traditional IP forwarding, it is easily affected by link quality variation. ComboCoding (Chen et al., 2011) uses both inter- and intra-flow networking to support TCP with deterministic routing. The inter-flow coding is done between the data flows of the two directions of the same TCP session. The intra-flow coding is based on random linear coding serving as a forward-error correction mechanism. It has an adaptive redundancy to overcome variable packet loss rates over wireless links. However, ComboCoding was not designed for opportunistic data forwarding.

2.4 Contribution of TCPFender

Opportunistic data forwarding and network coding do not inherently support TCP, so many previous research on opportunistic data forwarding and network coding were not designed for TCP. Other studies modified TCP protocols by cooperating network coding into TCP protocols; these work created different variants of TCP protocols to improve the throughput. However, TCP protocols (especially, TCP Reno) are widely deployed in current communication systems, it is not easy work to modify all TCP protocols of the communication systems. Therefore, we propose an adaptation layer (TCPFender) functioning below TCP Reno. With the help of TCPFender, TCP Reno do not make any change to itself and it can take advantage of both network coding and opportunistic data forwarding.

3 DESIGN OF TCPFENDER

3.1 Overview of TCPFender

We introduce TCPFender as an adaptation layer above the network layer, which hides network coding and opportunistic forwarding from the transport layer. The process of TCPFender is shown in Fig. 1. It confines the modification of the system only under the network layer. The goal of TCPFender is to improve TCP throughput in wireless mesh networks by opportunistic data forwarding and network coding. However, opportunistic data forwarding in wireless networks causes many dropped packets and out-of-order arrivals, and it is difficult for TCP sender to maintain a large congestion window. Especially the underlying link layer is the stock IEEE 802.11, which only provides standard unreliable broadcast or reliable unicast (best effort with a limited number of retransmissions). TCP has its own interpretation of the arrival (or absence) of the ACK segments and their timing. It opens up its congestion window based on continuous ACKs coming in from the destination. The dilemma is that when packets arrive out of order or are dropped, the TCP receiver cannot signal the sender to proceed with the expected ACK segment. Unfortunately, opportunistic data forwarding can introduce many out-of-order arrivals, which can significantly reduce the congestion window size of regular TCP since it increases the possibility of duplicated ACKs. Furthermore, the long decoding delay for batch-based network coding does not fare well with TCP, because it triggers excessive time-out events.

The TCPFender adaptation layer at the receiving side functions over the network layer and provides positive feedback early on when innovative coded packets are received, i.e. suggesting that more information has come through the network despite not being decoded for the time being. This process helps the sender to open its congestion window and trigger fast recovery when the receiving side acknowledges the arrival of packets belonging to a later batch, in which case the sending side will resend dropped packets of the unfinished batch. On the sender side, the ACK signalling module is able to differentiate duplicated ACKs and filter useless ACKs (shown in Fig. 1).

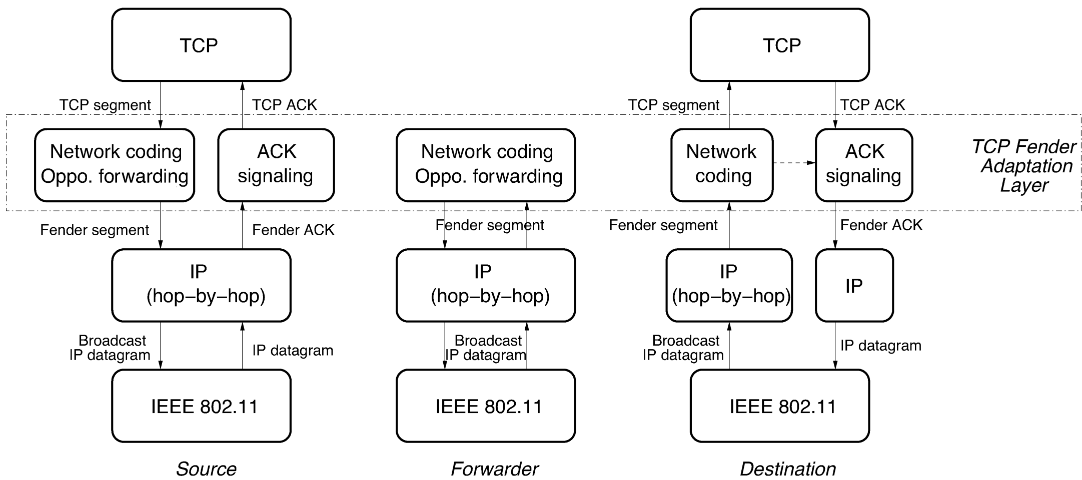

Figure 1. TCPFender design scheme.

3.2 TCPFender Algorithm

To better support TCP with opportunistic data forwarding and network coding, TCPFender inserts the TCP adaptation layer above the network work layer at the source, the forwarder, and the destination. The main work of the TCP adaptation layer is to interpret observations of the network layer phenomena in a way that is understandable by TCP. The network coding module in the adaptation layer is based on a batch-oriented network coding operation. The original TCP packets are grouped into batches, where all packets in the same batch carry encoding vectors on the same basis. At the intermediate nodes, packets will be recoded and forwarded following the schedule of opportunistic data forwarding proposed by MORE, which proposes a transmission credit system to describe the duplication of packets. This transmission credit system can compensate the packet loss, increase the reliability of the transmission, and represent the schedule of opportunistic data forwarding. The network coding module in the destination node will try to decode received coded packets to original packets when it receives any coded packet. The ACK signalling modules at the source and the destination are responsible for translation between TCP ACKs and TCPFender ACKs.

3.2.1 Network Coding in TCPFender

We implement batch-oriented network coding operations at the sender and receiver to support TCP transmissions. All data pushed down by the transport layer in sender are grouped into batches, and each batch has a fixed number β ($\beta = 10$ in our implementation) of packets of equal length (with possible padding). When the source has accumulated packets in a batch, these packets are coded with random linear network coding, tagged with the encoding vectors, and transmitted to downstream nodes. The downstream nodes are any nodes in the network closer to the destination. Any downstream node can recode and forward packets when it receives a sufficient number of them. We use transmission credit mechanism, as proposed in MORE, to balance the number of packets to be forwarded in intermediate nodes.

We make two important changes to improve the network coding process of MORE for TCP transmissions. For a given batch, the source does not need to wait until the last packet of a batch from the TCP before transmitting coded packets. We call this accumulative coding. That is, if k packets ($k < \beta$) have been sent down by TCP at a point of time, a random linear combination of these k packets is created and transmitted. Initially, the coded packets only include information for the first few TCP data segments of the batch, but will include more towards the end of the batch. The reason for this "early release" behaviour is for the TCP receiving side to be able to provide early feedback for the sender to open up the congestion window. On the other hand, we use a deeper pipelining than MORE where we allow multiple batches to flow in the network at the same time. To do that, the sending side does not need to wait for the batch acknowledgement before proceeding with the next batch. In this case, packets of a batch are labeled with a batch index for differentiation, in order for TCP to have a stable, large congestion window size rather than having to reset it to 1 for each new batch. The cost of such pipelining is that all nodes need to maintain packets for multiple batches.

3.2.2 Source adaptation layer

The source adaptation layer buffers all original packets of a batch that have not been acknowledged. The purpose is that when TCP pushes down a new data packet or previously sent data packet due to a loss event, the source adaptation layer can still mix it with other data packets of the same batch. The ACK signalling module can discern duplicated ACKs which are not in fact caused by the network congestion. Opportunistic data forwarding may cause many extra coded packets, specifically when some network links are of the high quality at a certain point. This causes the destination node to send multiple ACKs with same sequence number. In this case, such duplicated ACKs are not a signal for the network congestion, and should be treated differently by the ACK signalling module in the source. These two cases of duplicated ACKs can actually be differentiated by tagging the ACKs with the associated sequence numbers of the TCP data segment. These ACKs are used by the TCPFender adaptation layer at the source and the destination and should be converted to original TCP ACKs before being delivered to the upper layer.

The flow of data or ACKs transmissions is shown in the left of Fig. 1. Original TCP data segments are generated and delivered to the module of "network coding and opportunistic forwarding". Here, TCP data segments may be distributed to several batches based on their TCP segment sequences, so the retransmitted packets will be always in the same batch as their initial distribution. After the current TCP data segment mixes with packets in a batch, TCPFender data segments will be generated and injected to network via hop-by-hop IP forwarding, which is essentially broadcasting of IP datagrams. On the ACK signalling module, when it receives TCPFender ACKs, if the ACK's sequence number is greater than the maximum received ACK sequence number, this ACK will be translated into a TCP ACK and delivered to the TCP sender. Otherwise, the ACK signalling module will check whether this duplicated ACK is caused by opportunistic data forwarding or not. Then it will decide whether to forward a TCP ACK to the TCP or not. The reason for differentiating duplicated ACKs at the source instead of at the destination is to reduce the impact of ACK loss on TCP congestion control.

3.2.3 Destination adaptation layer

The main function of the destination adaptation layer is to generate ACKs and detect congestion in the network. It expects packets in the order of increasing batch index. For example, when it is expecting the bth batch, it implies that it has successfully received packets of the previous $b-1$ batches and delivered them up to the TCP layer. In this case, it is only interested in and buffers packets of the bth batch or later. However, the destination node may receive packets of any batch. Suppose that the destination node is expecting the bth batch, and that the rank of the decoding matrix of this batch is r. In this case, the destination node has "almost" received $\beta \times (b-1) + r$ packets of the TCP flow, where $\beta \times (b-1)$ packets have been decoded and pushed up the TCP receiver, and r packets are still in the decoding matrix. When it receives a coded packet of the b'th batch, if $b' < b$, the packet is discarded. Otherwise, this packet is inserted into the corresponding decoding matrix. Such an insertion can increase r by 1 if $b' = b$ and this received packet is an innovative packet. The received packet is defined as an innovative packet only if the received packet is linearly independent with all the buffered coded packets within the same batch. In either case, it generates an ACK of sequence number $\beta \times (b-1) + r$, which is sent over IP back to the source node. One exception is that if $r = \beta$ (i.e. decoding matrix become full rank), the ACK sequence number is $\beta \times (\hat{b} - 1) + \hat{r}$, where $\hat{b}$ is the next batch that is not full and $\hat{r}$ is its rank. At this point, the receiver moves on to the $\hat{b}$th batch. This mechanism ensures that the receiver can send multiple duplicate ACKs for the sender to detect congestion and start fast recovery. It also supports multiple-batch transmissions in the network and guarantees the reliable transmission at the end of the transmission of each batch.

The design of the destination adaptation layer is shown on the right of Fig. 1. The network coding module has two functions. First, it will check whether the received TCPFender data segment is innovative or not. In either case, it will notify the ACK signalling to generate a TCPFender ACK. Second, it will deliver original TCP data segments to TCP layer if one or more original TCP data segment are decoded after receiving an innovative coded data packet. This mechanism can significantly reduce the decoding delay of the batch-based network coding. On the other hand, TCPFender has its own congestion control mechanism, so TCP ACK that is generated by the TCP layer will be dropped by the ACK signalling module at the destination.

3.2.4 Forwarder adaptation layer

The flow of data at forwarders is shown in the middle of Fig. 1. The ACK is unicast from the destination to the source by IP forwarding, which is standard forwarding mechanism and is not shown in the diagram. The intermediate node receives TCPFender data segment from below and this segment will be distributed into corresponding batches and regenerates a new coded TCPFender data segment. This new TCPFender data segment will be sent to downstream forwarders via hop-by-hop IP broadcasting based on the credit transmission system proposed by MORE.

4 PERFORMANCE EVALUATION

In this section, we investigate the performance of TCPFender through computer simulations using NS-2. The topologies of the simulations are made up of three exemplar network topologies and one specific mesh. These topologies are depicted in Fig. 2 “diamond topology”, Fig. 3 “string topology”, Fig. 4 “grid topology”, and Fig. 5 “mesh topology”. The packet delivery rates at the physical layer for the mesh topology are marked in Fig. 5, and the packet delivery rates for other topologies are described in Table. 1. The source node and the destination node are at the opposite ends of the network. One FTP application sends long files from the source to the destination. The source node emits packets continuously until the end of the simulation, and each simulation lasts for 100 seconds. All the wireless links have a bandwidth of 1Mbps and the buffer size on the interfaces is set to 100 packets. To compensate for the link loss, we used the hop-to-hop redundancy factor for TCPFender on a lossy link. Recall that the redundancy factor is calculated based on the packet loss rate, which was proposed in MORE (Chachulski et al., 2007). This packet loss rate should incorporate the loss effect at both the Physical and Link layers, which is higher than the marked physical layer loss rates. The redundancy factors of the links are thus set according to these revised rates. We compared our protocol against TCP and TCP+NC in four network topologies. In our simulations, TCP ran on top of IP, and TCP+NC has batch-based network coding enabled but still over IP. The version of TCP is TCP Reno for TCPFender and both baselines. The ACK packet for the three protocols are routed to the source by shortest-path routing.

In this paper, we examined whether TCPFender can effectively utilize opportunistic forwarding and network coding. TCPFender can provide reliable transmissions in these four topologies and the analysis metrics we took are the network throughput and the end-to-end packet delay at the application layer. We repeated each scenario 10 times with different random seeds for TCPFender, TCP+NC, and TCP/IP, respectively. In TCPFender, every intermediate node has the opportunity to forward coded packets and all nodes operate in the 802.11 broadcast mode. By contrast, for TCP/IP and TCP+NC, we use the unicast model of 802.11 with ARQ and the routing module is the shortest-path routing of ETX Couto et al. (2003).

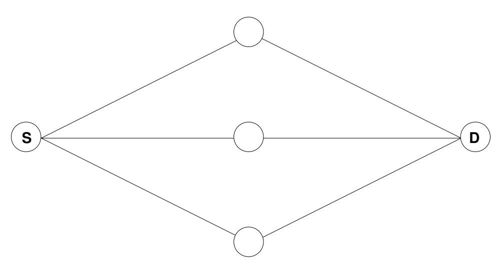

Figure 2. Diamond topology

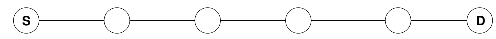

Figure 3. String topology

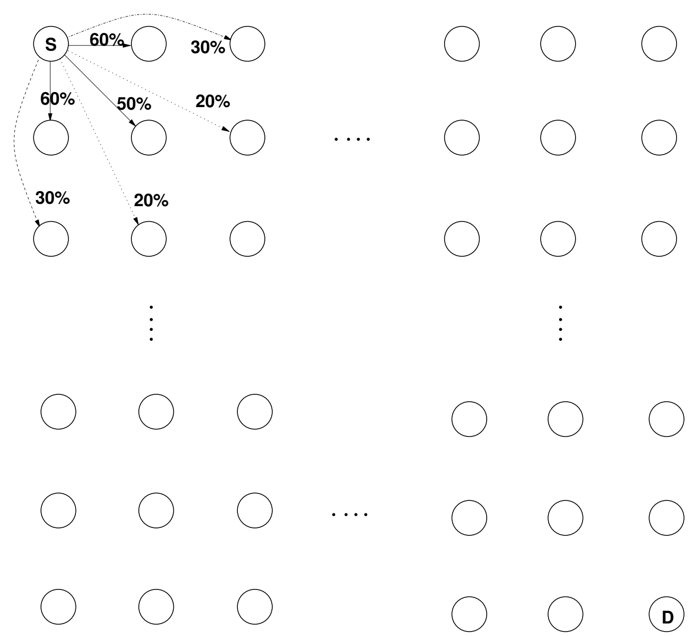

Figure 4. Grid topology

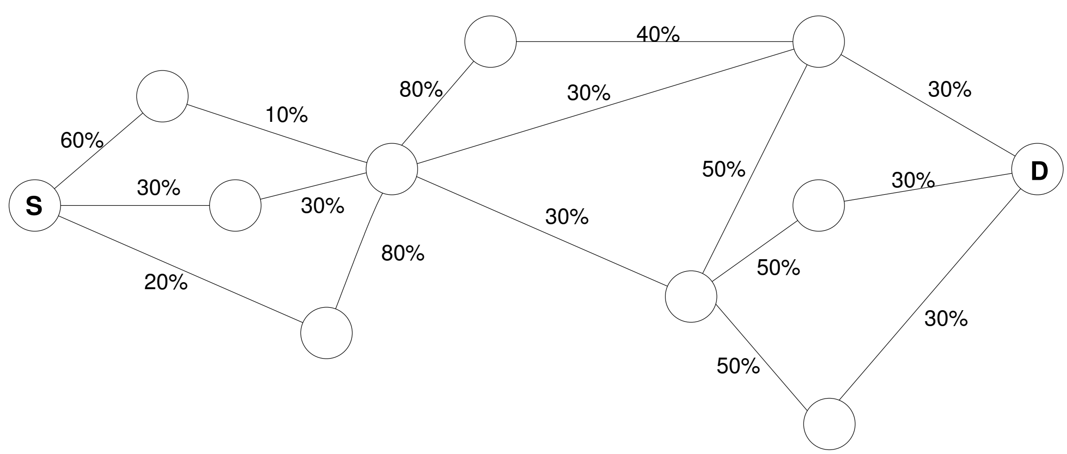

Figure 5. Mesh topology

In the diamond topology (Fig. 2), the source node has three different paths to the destination. TCP and TCP+NC only use one path to the destination, but TCPFender could utilize more intermediate forwarders thanks to the opportunistic routing. The packet delivery rates for each link are varied between 20%, 40%, 60% and 80%. We plotted the throughput of these three protocols in Fig. 6. In all cases, the TCPFender has the highest throughput, and the performance gain is more visible for poor link qualities.

Next, we tested these protocols in the string topology (Fig. 3) with 6 nodes. The distance between the two nodes is 100 meters, and the transmission range is the default 250 meters. Different combinations

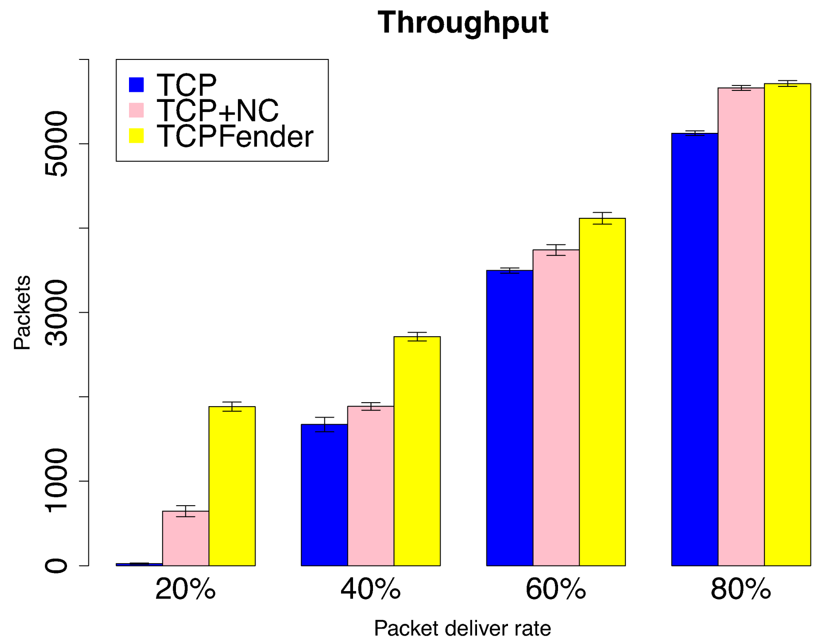

Figure 6. Throughput for diamond topology

Table 1. Packet delivery rate

100m	200 m			
100%	80%	60%	40%	20%
80%		60%	40%	20%
60%			40%	20%
40%				20%

of packet delivery rates for 100-meter and 200-meter distances are described in Table 1. As a result, the shortest path routing used by TCP and TCP+NC can decide to use the 100m or 200m links depending on their relative reliability. The throughputs of the three protocols are plotted in Fig. 7, where we observed how they perform under different link qualities. Except for the one case where both the 100m and 200m links are very stable (i.e 100% and 80%, respectively), the gains of having network coding and opportunistic forwarding are fairly significant in maintaining TCP's capacity to the application layer. When the links are very stable, the cost of the opportunistic forwarding schedule and the network coding delay will slightly reduce the network throughput.

We also plotted these three protocols' throughputs in a grid topology (Fig. 4) and a mesh topology (Fig. 5). Each node has more neighbours in these two topologies, compared to string topology (Fig. 3), which increases the chance of opportunistic data forwarding. The packet delivery rates are indicated in these two Figures (Fig. 4 and Fig. 5). In general, the packet delivery rates drop when the distance between a sender and a receiver increases. In our experiment, the source and destination nodes deploy at the opposite ends of the network. The throughput of TCPFender is depicted in Fig. 8 and it is much higher than TCP/IP because opportunistic data forwarding and network coding increase the utilization of network capacity. The gain is about 100% in our experiment. The end-to-end delays of the grid topology and the mesh topology are plotted in Fig. 8. In general, TCP+NC has long end-to-end delays because packets need be decoded before delivered to the application layer, this is an inherent feature of batch-based network coding. TCPFender can benefit from backup paths and receive packets early, so it reduces the time-consumption of waiting for decoding and its end-to-end delay is shorter than TCP+NC.

Next, we are interested in the impact of batch sizes on the throughput and the end-to-end delay. Fig. 9 shows the throughput of TCPFender in the mesh topology for batch sizes of 10, 20, 30, ..., 100 packets. In general, batch sizes will have an impact on then TCP throughput (as exemplified in Fig. 11). When the

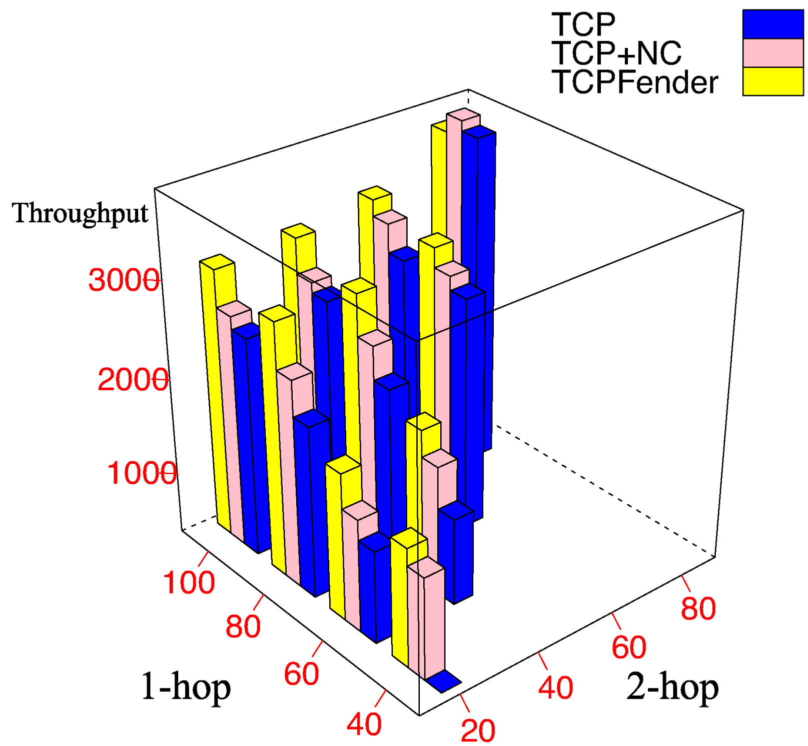

Figure 7. Throughput for string topology

batch size is small (≤ 40), the increment of the batch size can increase the throughput, since it expands the congestion window. However, if the batch size is too large (> 40), the increment of the batch size will decrease the throughput because the increase of batch size will amplify the fluctuation of the congestion window and also increase packet overhead by long encoding vectors. The Fig. 11 also describes how many packets are transmitted in the network. Each intermediate node will keep all unfinished batches. From the Fig. 11, since the number of packets transmitted in the network is smaller than two batch sizes, intermediate nodes only need to keep two batches of packets and the memories required to store the packets are acceptable. The nature of batch-based network coding will also introduce decoding delays, so the batch size has a direct impact on the end-to-end delay, as summaries in Fig. 9. In Fig. 10, we plotted the end-to-end delays of all packets over time in two sample simulations. Note that these tests were done for files that need many batches to carry. On the other hand, when the file size is comparable to the batch size, the file-wise delay will be comparable to the decoding delay of an entire batch, which may seem large relatively. However, because the file size is small, this delay is not overly significant as the delay is at the order of its transmission time. Nevertheless, network coding does add considerable amount of delay in comparison to pure TCP/IP.

5 CONCLUDING REMARKS

In this paper, we proposed TCPFender, which is a novel mechanism to support TCP with network coding and opportunistic data forwarding. TCPFender completes the control feedback loop of TCP by creating a bridge between the adaptation modules of the sender and the receiver. The sender adaptation layer in TCPFender differentiates duplicate ACKs caused by network congestion from these caused

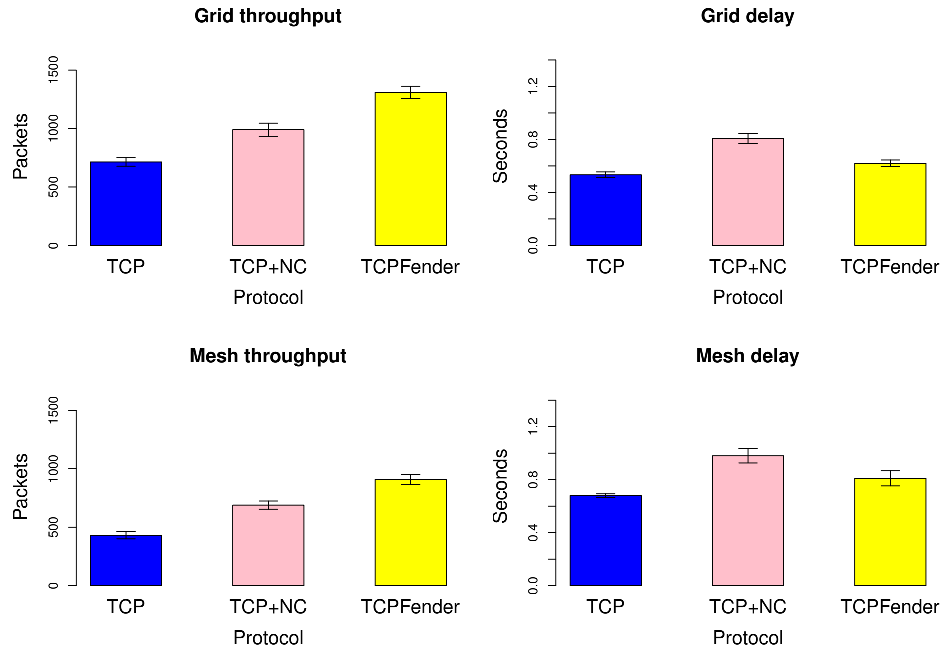

Figure 8. Throughput and delay for grid topology and mesh topology.

by opportunistic data forwarding, and the receiver side releases ACK segments whenever receiving an innovative packet. In current work, we implemented our algorithm to support TCP Reno. In fact, TCPFender can also support other TCP protocols with loss-based congestion control (e.g., TCP-NewReno, TCP-Tahoe). The adaptive modules are designed generally enough to not only support network coding and opportunistic data forwarding, but also any packet forwarding techniques that can cause many dropping packets or out-of-order arrivals. One example will be multi-path routing, where IP packets of the same data flow can follow different paths from the source to the destination. By simulating how TCP receiver will signal the TCP sender, we are able to adapt TCPFender to functioning over such the multi-path routing without having to modify TCP itself.

In the simulation results, we compared TCPFender and TCP/IP in four different network topologies. The result shows that TCPFender has a sizeable throughput gain over TCP/IP, and the gain will be very distinct from each other when the link quality is not that good. We also discussed the influence of batch size on the network throughput and end-to-end packet delay. In general, the bath size has a small impact on the network throughput, but it has direct impact on end-to-end packet delay.

In future, we will consider TCP protocols with RTT-based congestion control and also analyze how multiple TCP flows interact with each other in a network coded, opportunistic forwarding network layer, or a more generally error-prone network layer. We will refine the redundancy factor and the bandwidth estimation to optimize the congestion control feedback of TCP. Finally, we will propose a theoretical model of TCP with opportunistic forwarding and network coding, which will enable us to study the TCPFender as a function in various communication systems.

REFERENCES

Aguayo, D., Bicket, J., Biswas, S., Judd, G., and Morris, R. (2004). Link-level measurements from an 802.11b mesh network. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, volume 34, pages 121–132, New York. ACM.

Ahlswede, R., Cai, N., Shuo-Yen, Li, R., and Yeung, R. W. (2000). Network information flow. *Information Theory*, 46(4):1204–1216.

Argyriou, A. (2009). Wireless network coding with improved opportunistic listening. *IEEE Transactions on Wireless Communications*, 8(4):2014–2023.

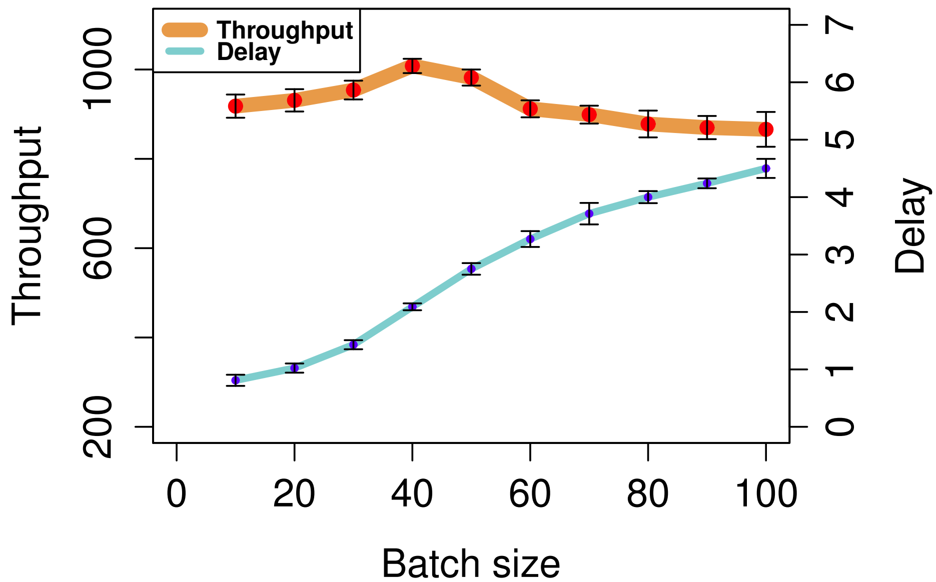

Figure 9. Throughput and delay for different batch sizes

Bao, W., Shah-Mansouri, V., Wong, V. W., and Leung, V. C. (2012). TCP VON: Joint congestion control and online network coding for wireless networks. In *Global Communications Conference (GLOBECOM)*, pages 125 – 130, Anaheim. IEEE.

Basagni, S., Conti, M., Giordano, S., and Stojmenovic, I. (2008). XORs in the air: Practical wireless network coding. *IEEE/ACM Transactions on Networking*, 16(3):497–510.

Biswas, S. and Morris, R. (2005). ExOR:opportunistic multi-hop routing for wireless networks. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 133–144. ACM.

Cai, S., Zhang, S., Wu, G., Dong, Y., and Znati, T. (2014). Minimum cost opportunistic routing with intra-session network coding. In *IEEE International Conference on Communications (ICC)*, pages 502 – 507, Sydney, NSW.

Chachulski, S., Jennings, M., Katt, S., and Katabi, D. (2007). Trading structure for randomness in wireless opportunistic routing. *ACM SIGCOMM Computer Communication Review*, 37(12):169–180.

Chen, C.-C., Chen, C., Oh, S. Y., Park, J.-S., Gerla, M., and Sanadidi, M. (2011). ComboCoding: Combined intra-/inter-flow network coding for TCP over disruptive MANETs. *Journal of Advanced Research*, 2:241–252.

Chen, Y., Zhang, J., and Marsic, I. (2009). Link-layer-and-above diversity in multi-hop wireless networks. *IEEE Communications Magazine*, 47(2):118–124.

Couto, D. S. J. D., Aguayo, D., Bicket, J., and Morris, R. (2003). A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 134–146, New York. ACM.

Garrido, P., Gómez, D., Agüero, R., and Serrat, J. (2015). Combination of random linear coding and cross-layer opportunistic routing: Performance over bursty wireless channels. *IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1692 – 1696.

Ho, T., Médard, M., Koetter, R., Karger, D. R., Effros, M., Shi, J., and Leong, B. (2006). A random linear network coding approach to multicast. *IEEE Transmission on Information Theory*, 52(10):4413–4430.

Hsu, Y.-P., Abedini, N., Gautam, N., Sprintson, A., and Shakkottai, S. (2015). Opportunities for network

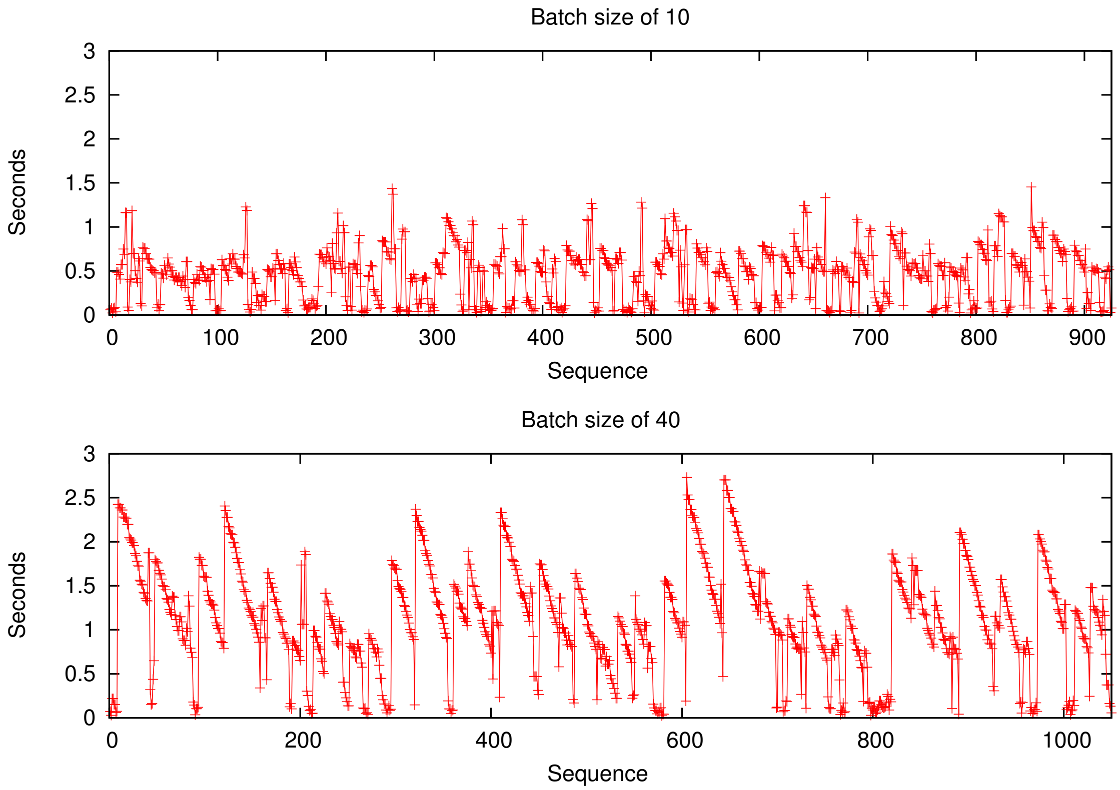

Figure 10. Delay for two specific cases with batch sizes of 10 and 40

coding: To wait or not to wait. *IEEE/ACM Transactions on Networking*, 23(6):1876–1890.

Huang, Y., Ghaderi, M., Towsley, D., and Gong, W. (2008). TCP performance in coded wireless mesh networks. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 179–187, San Francisco. IEEE.

Jaggi, S., Sanders, P., Chou, P., and Effros, M. (2005). Polynomial time algorithms for multicast network code construction. *IEEE Transmission on Information Theory*, 51:1973–1982.

Jain, S. and Das, S. (2005). Exploiting path diversity in the link layer in wireless ad hoc networks. In *World of Wireless Mobile and Multimedia Networks (WoWMoM)*, pages 22–30. IEEE.

Koetter, R. and Médard, M. (2003). An algebraic approach to network coding. *IEEE Transmission on Networking*, 11(5):782–795.

Koutsonikolas, D., Wang, C.-C., and Hu, Y. C. (2011). Efficient network-coding-based opportunistic routing through cumulative coded acknowledgments. *IEEE/ACM Transactions on Networking (TON)*, 19(5):1368–1381.

Krigslund, J., Hansen, J., Hundeboll, M., Lucani, D. E., and Fitzek, F. H. P. (2013). CORE: COPE with MORE in wireless meshed networks. In *Vehicular Technology Conference (VTC Spring)*, pages 1 – 6, Dresden.

Laneman, J. N., Tse, D. N. C., and Wornell, G. W. (2004). Cooperative diversity in wireless networks: Efficient protocols and outage behavior. *IEEE Transmission on Information Theory*, 50(12):3062–3080.

Larsson, P. (2001). Selection diversity forwarding in a multihop packet radio network with fading channel and capture. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):47–54.

Li, P., Guo, S., Yu, S., and Vasilakos, A. V. (2012). Codepipe: An opportunistic feeding and routing protocol for reliable multicast with pipelined network coding. In *INFOCOM*, pages 100–109, Brazil.

Li, S.-Y. R., Yeung, R. W., and Cai, N. (2003). Linear network coding. *IEEE Transmission on Information Theory*, 49(2):371–381.

Lin, Y., Li, B., and Liang, B. (2008). CodeOR: opportunistic routing in wireless mesh networks with segmented network coding. In *IEEE International Conference on Network Protocols (ICNP)*, pages 13

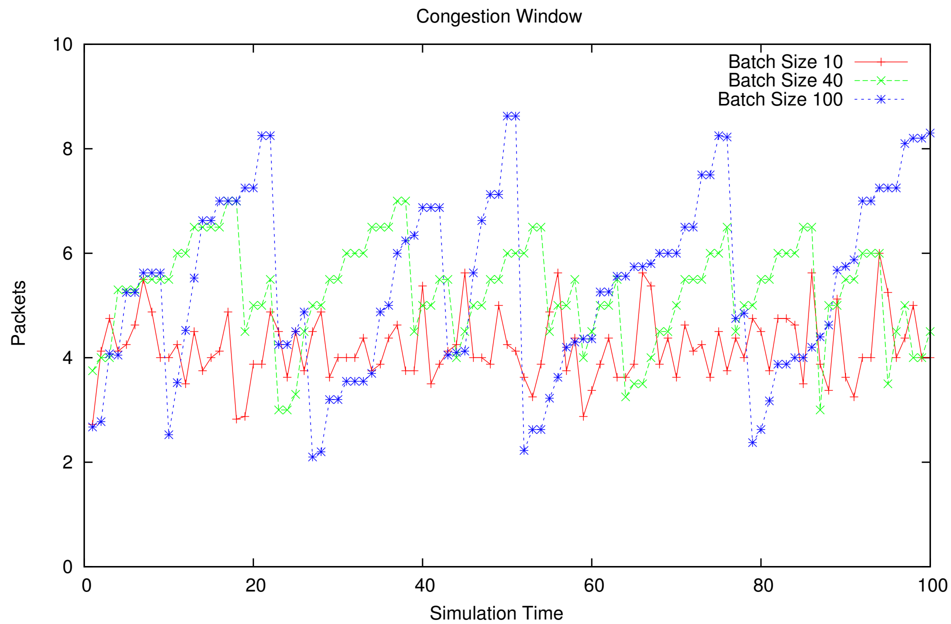

Figure 11. Evolution of congestion window for three different batch sizes simulated in the mesh topology.

– 22, Orlando, FL. IEEE.
Lin, Y., Liang, B., and Li, B. (2010). SlideOR: Online opportunistic network coding in wireless mesh networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 – 5, San Diego, CA. IEEE.
Mehta, T. and Narmawala, Z. (2011). Survey on multimedia transmission using network coding over wireless networks. In *Nirma University International Conference on Engineering*, pages 1–6, Ahmedabad.
Sun, J., Zhang, Y., Tang, D., Zhang, S., Zhao, Z., and Ci, S. (2015). TCP-FNC: A novel tcp with network coding for wireless networks. In *International Conference on Communications (ICC)*, London. IEEE.
Sundararajan, J. K., Shah, D., Medard, M., Jakubczak, S., Mitzenmacher, M., and Barros, J. (2011). Network coding meets TCP: Theory and implementation. *Proceedings of the IEEE*, 99(3):490–512.
Wang, Z., Chen, Y., and Li, C. (2012). CORMAN: A novel cooperative opportunistic routing scheme in mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 30(2):289–296.
Yan, Y., Zhang, B., Mouftah, H. T., and Ma, J. (2008). Practical coding-aware mechanism for opportunistic routing in wireless mesh networks. In *IEEE International Conference on Communications*, pages 2871–2876, Bei Jing.
Zhu, D., Yang, X., Yu, W., Lu, C., and Fu, X. (2015). INCOR: inter-flow network coding based opportunistic routing in wireless mesh networks. In *IEEE International Conference on Communications (ICC)*, pages 3666 – 3671, London.