Evaluation and Enhancement of TCP with Network Coding in Wireless Multihop Networks

Yanli Xu[1,2], Xiaolin Bai[1], Ping Wu[1], and Lianghui Ding[1]

[1] Signals and Systems, Dept. of Engineering Sciences, Uppsala University, Uppsala, Sweden
yanli.xu@angstrom.uu.se
[2] National Mobile Communications Research Laboratory, Southeast University, Nanjing, China

Abstract. In this paper, we, based on NS-2 simulator, evaluate the performances of different TCP protocols with network coding in wireless multihop networks, and then propose two schemes to enhance the performances of TCPs with network coding. In particular the network coding scheme considered and used here is COPE, which is one of the well-known practical network coding schemes, the TCP protocols evaluated are TCP-NewReno, TCP-FeW and TCP-AP, and the TCP protocols with COPE are implemented in NS-2. The simulation results show that COPE performs very differently in improving the performances of the TCP's in different wireless network topologies. In some topologies COPE performs well, resulting in significant performance improvement; while in other ones it performs worse than the same cases without network coding. To overcome this problem, we propose two schemes to improve the performance of TCP with network coding. One is called Encode Once, which ensures the packet being encoded at most one time. The other is called Network Coding Aware TCP, in which the transmitting rate of TCP is made adaptive to the status of the node's output queue. The evaluation results indicate that the proposed two schemes can significantly improve the goodputs of TCP's with network coding, and the latter scheme performs better.

Keywords: TCP, COPE, NS-2, network coding, wireless multihop networks.

1 Introduction

In multihop wireless networks, it is challenging to provide high goodput service due to the scarce bandwidth resources and harsh radio propagation environments. Network coding is a promising techniques that can improve wireless goodghput essentially. Network coding was originally proposed for wired networks by Ahlswede *et.al.* [1], and then it was shown to be able to offer benefits for wireless networks [2,3,4]. The basic idea of network coding is to ask an intermediate node to mix the messages it received and forward the mixture to several

P. Ren et al. (Eds.): WICON 2011, LNICST 98, pp. 490–500, 2012.

destinations simultaneously. Compared to time sharing based schemes where destinations are served in turn, the use of network coding can increase the overall goodput dramatically. COPE is one of the practical network coding schemes for wireless networks. It was proposed by S. Katti *et.al.* in [5]. In COPE, each node opportunistically overhears those packets transmitted by its neighbors, which are not addressed to itself, and notices what packets the neighbors currently possess (by adding piggyback reception reports on the data packets the node transmits). Each node can intelligently XOR multiple packets destined to nodes in next hop such that multiple packets can be forwarded in a single transmission, resulting in a significant improvement in node transmission efficiency. Results obtained from the first test bed deployment of wireless network coding showed that COPE can substantially improve the goodput of multihop wireless networks.

However, extending coding technologies to the network setting in a practical way has been a challenging task. In the Internet, flow control and congestion control are predominantly based on transmission control protocol (TCP), i.e., controlling transmission rate according to sliding transmission window of packets, whose size is controlled based on feedback from destination nodes [6,7]. Most wireless applications rely on legacy TCP to communicate with TCP-dominant wired hosts, and it is likely that TCP will remain as the major transport protocol for the clients of 802.11 networks [8]. But it was shown in [9] that the performance gain of TCP with COPE in a 802.11 network can be neglected. A lot of research has been done on jointing implementation of TCP and network coding such as [10,11,12]. However, as we know, no attention has been paid to the performance degradation of TCP caused by the rate control mechanism of TCP itself and the routing protocol [8,13,14] has not been noted.

To find the reasons that cause the performance degradation of TCP in wireless multihop networks, especially with focus on the rate control mechanism of TCP, we will evaluate the performances of three TCP protocols: TCP-NewReno [15] for wired communication, TCP-FeW (Fractional Window Increment) [8] and TCP-AP (TCP with Adaptive Pacing) [14] for wireless communication using network simulator NS-2, and then compare the performances of the protocols with COPE to those without COPE in order to observe the improvement of performance due to use of COPE. Finally, we provide two new schemes which jont TCP and network coding better in wireless multihop networks.

The rest of the paper is organized as follows: we first introduce the three TCP protocols and the model for implementation of tcp with cope in NS-2 in Section 2. Then, we present simulation results in Section 3. Afterwards, we provide two novel schemes in Section 4 and conclude the whole paper in Section 5.

2 Implementation TCP with COPE on NS-2

2.1 COPE-Coding Opportunistically

COPE is the first practical network coding-based packet forwarding architecture that substantially improves the goodput of wireless networks. It inserts a coding shim between the IP and MAC layers, which identifies coding opportunities

and benefits from them by forwarding multiple packets in a single transmission. COPE incorporates three main techniques: Opportunistic Listening: COPE sets the nodes in promiscuous mode, makes them snoop on all communications over the wireless medium and store the overheard packets for a limited period; Opportunistic Coding: the node aims to maximize the number of native packets delivered in a single transmission, while ensuring that each intended next hop has enough information to decode its native packet; Learning Neighbor State: each node announces to its neighbors the packets it stores in reception reports. When reception reports get lost in collisions, a node can not rely solely on reception reports to make code decisions, then it will leverage the routing computation to guess whether a neighbor has a particular packet. Occasionally, a node may make an incorrect guess, which causes the coded packet to be undecodable at some next hop. In this case, the relevant native packet is retransmitted, potentially encoded with a new set of native packets.

2.2 TCP Protocols

TCP is one of the core protocols of the Internet Protocol Suite, which is a connection oriented protocol, provides end-to-end, reliable and ordered delivery of a stream of bytes between computers. In this work, we consider the following three TCP protocols: TCP-NewReno, TCP-FeW and TCP-AP.

TCP-NewReno. TCP-NewReno is a slight modification over TCP Reno. When receiving three duplicate ACKs, TCP Reno assumes there is packet loss happening on the link, and then starts up the 'Fast Retransmit' and 'Fast Recovery' processes, which will immediately retransmit the lost packet. TCP Reno doesn't perform well under the condition of high packet loss. To overcome this problem, TCP-NewReno improves retransmission during the 'Fast Recovery' stage of TCP Reno. For every lost packet, TCP-NewReno retransmits it and waits for the acknowledgment before exiting from 'Fast Recovery' process. TCP-NewReno successfully prevents TCP from going back to 'Slow Start' when there are multiple packet drops. TCP-NewReno keeps the same performance at low packet loss ratio, and substantially outperforms TCP Reno at high packet loss ratio.

TCP-FeW. In the TCP-FeW scheme, it prevents the over-reaction of the on-demand routing protocol by limiting TCP's aggressiveness. TCP-FeW allows the TCP congestion window to grow by a fractional rate $\alpha \leq 1$ (packets) in each RTT (Round-Trip-Time). That means adding one packet into the congestion window at every $1/\alpha$ RTT. Assumming the size of the current congestion window is $W^{current}$, the source node transmits $W^{current}$ packets to destination node and receives the same number of ACKs from the destination node during each RTT. When receiving an ACK, the source node updates the current congestion window size through the following formula 1:

$$W^{new} = W^{current} + \frac{\alpha}{W^{current}} \tag{1}$$

where $0 < \alpha \leq 1$. When $\alpha = 1$, the pattern of increase of the formula above is the same as the traditional style, i.e., increasing the window size by 1 when receiving an ACK. So TCP-FeW keeps the original mechanism of TCP congestion window growth. In addition to that, TCP-FeW can monitor the network traffic without any other additional information and provide quick reactions.

TCP-AP. TCP-AP is a rate-based scheduling of transmissions within the TCP congestion window. The source node adapts its transmission rate according to the estimation of the current 4-hop propagation delay and coefficient of variation of recently measured RTTs. Unlike the previous solutions, TCP-AP keeps the end-to-end semantics of TCP, does not need to modify the routing or link layer, and does not rely on the cross-layer information from intermediate nodes along the path. Moreover, TCP-AP achieves excellent fairness and quick reaction to control network traffic conditions.

2.3 NS-2 and Simulation of TCP with COPE

Network Simulator NS-2. The network simulator, is a discrete event simulator targeted at networking research. NS-2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks [16]. NS-2 includes almost all the main parts of network, and it is a free and open-source platform for network simulation. That's why NS-2 is used widely in academic fields, and chosen here in this work for the simulation of TCP with network coding and the evaluation of its performance.

Implementation of TCP with COPE on NS-2. The core of the wireless module in NS-2 is mobile node, which was originally ported as CMU's (Carnegie Mellon University) Monarch group's mobility extension to NS [16]. It is a basic node object equipped with wireless functionalities, and the mobile node can move within the given topology, receive and send radio signals through wireless channel. The characteristics of mobility like node movement, periodic location update, topology maintenance, etc., are implemented by C++, while the internal network components (like classifier, Link Layer (LL), Media Access Control (MAC), Channel, etc.) are assembled using OTcl, which is an object oriented extension of the scripting language Tcl (Tool Command Language).

We implement three key parts of COPE as well as complementary parts for IEEE 802.11 network, pseudo-broadcast, asynchronous acknowledgment and retransmission. We modify IEEE 802.11 MAC and the Interface Queue (IFq) protocol stacks on NS-2 to adapt to the present application. Furthermore, the original structure of the mobile node in NS-2 are modified for COPE (refer to our work [17] for detail). For opportunistic listening, the up-target of MAC is no longer LL, because the COPE is implemented on IFq layer and all the incoming packets should pass through COPE. If the destination or next hop of incoming packet is this node, then the packet is passed to LL, which now is the up-target of IFq. Otherwise, COPE saves it in the packet pool or drops it depending on whether the packet can be decoded or not. The new structure of the mobile node with COPE looks like that shown in fig. 1.

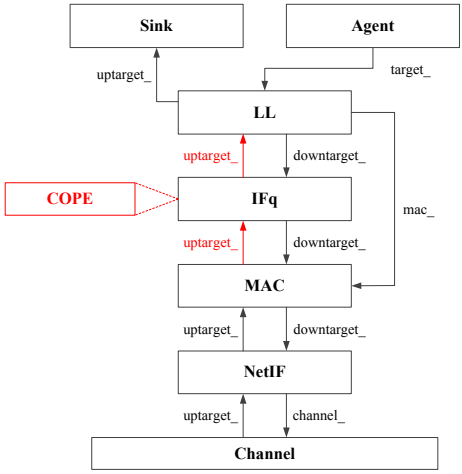

Fig. 1. Structure of mobile node with COPE in NS-2

3 Simulation Results and Performance Analysis

In this section, the performances of three different TCP protocols: TCP-Newreno, TCP-FeW and TCP-AP, are evaluated using NS-2 simulator. Three topologies, namely, chain topology, classic X topology (named as X-I topology) and an extended scenario of X-I topology (called X-II topology), are considered. For the MAC layer, IEEE 802.11 with RTS/CTS (Request to Send/Clear to Send) are adopted. For TCP traffic, FTP (File Transfer Protocol) traffic is used. In all the topologies, TCP flows start randomly between first $2sec$ and $3sec$. The evaluated metrics are listed as follows:

– *Network goodput:* the measured total end-to-end goodput of all flows. The average goodput T of each flow in the network is computed according to

$$T = \frac{received\_packets\_size \times 8}{simulation\_time}(kbps) \tag{2}$$

– *Goodput Gain:*the ratio between the measured network goodput with COPE to the traditional TCP[5], i.e.,

$$Goodput\_Gain = \frac{G_{cope}}{G_{nocope}} \tag{3}$$

where G_{cope} and G_{nocope} are the average goodputs of TCP with and without COPE, respectively.

The chain topology shown in fig. 2 is the first one considered in the present work. In this topology, the distance of the contiguous nodes is 200 meters and the transmission range of each node is 250 meters. The simulation results are shown

in fig. 3, in which the network goodputs of TCP with COPE for TCP-NewReno and TCP-FeW are significantly higher than those without COPE, which proves the benefits of network coding. The results also show that TCP-NewReno is the worst choice for network coding, and TCP-AP with COPE performs better than TCP-FeW with COPE when the number of hops is less than and afterwards the former becomes worse than the latter.

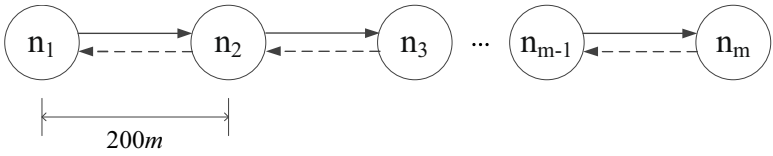

Fig. 2. Chain Topology

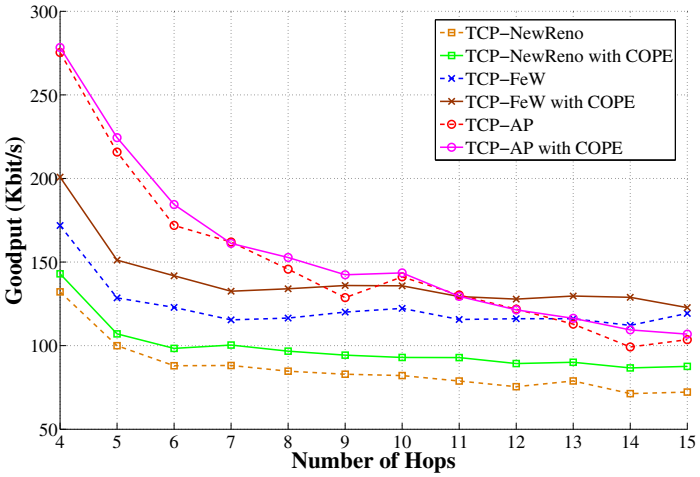

Fig. 3. Goodput in Chain Topology

More realistic topologies are X-shaped. The first X topology considered is shown in fig. 4 , and called X-I topology to distinguish from an extended X topology to be dealt with below. In this topology there are two flows: one from $n0$ to $n2$ via $n1$ and the other from $n3$ to $n4$ via $n1$. The simulation results for this topology are illustrated in fig. 5. From the figure we can observe that COPE brings marginal improvement of performances for both TCP-NewReno and TCP-FeW (with about and respectively); and the worst case is TCP-AP with COPE, where network coding has no contribution. One reason is that TCP-AP is based on the estimation of the current 4-hop propagation delay and coefficient of variation of recently measured RTTs. However, there are only 2 hops, which restricts the performance of TCP-AP. The other reason is that TCP-AP is too

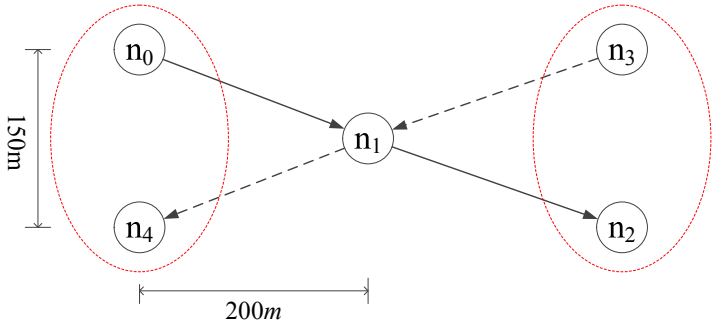

Fig. 4. X-I Topology

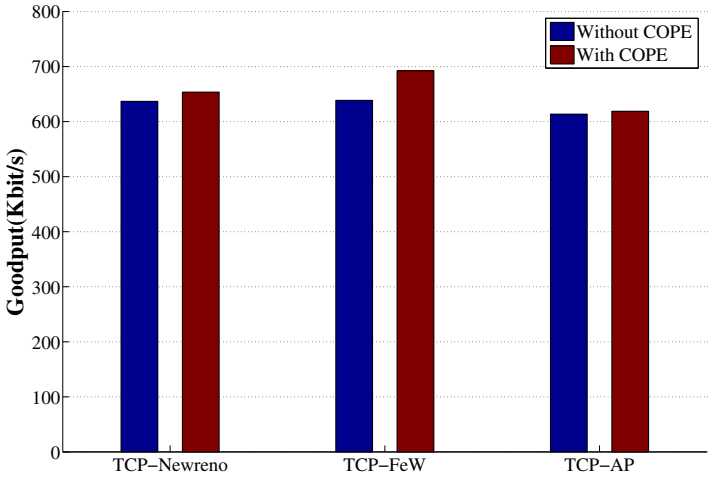

Fig. 5. Goodput in X-I Topology

proactive for congestion control, and as a result, fewer packets in the output queue are used for network coding.

To evaluate the performance of COPE and find out the reason that leads to no gain for TCP-AP, we extend the X-I topology to another X topology as shown in fig. 6, which is called as X-II topology. X-II topology is similar to X-I topology except it has four more nodes, so each of the two TCP flows traversing from $n0$ to $n4$ and from $n5$ to $n8$, respectively, has 4 hops, which satisfies the prerequisite of TCP-AP.

As described in fig. 7, both TCP-NewReno and TCP-FeW with COPE get worse results compared to TCP-NewReno and TCP-FeW without COPE. That's because of TCP-NewReno's bursty behavior and due to the fact that TCP is agnostic to the underlying network coding. While TCP-FeW alleviates this bursty situation, but still not so suitable for network coding. Similar to the X-I topology, TCP-AP with COPE still has no contribution to the TCP Performance. We believe that this is caused by the rate control mechanism of TCP-AP.

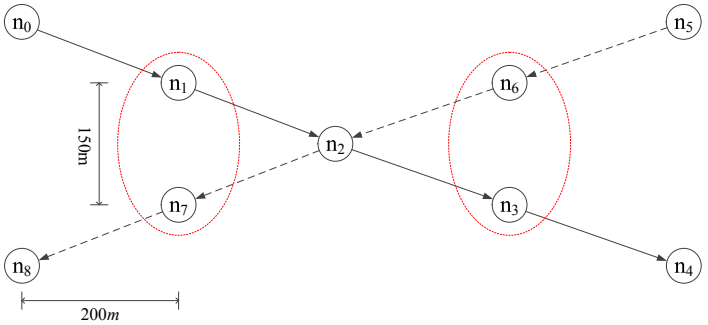

Fig. 6. X-II Topology

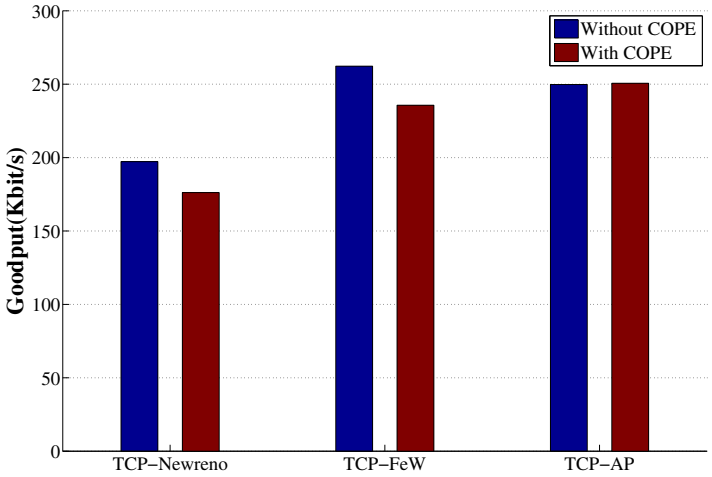

Fig. 7. Goodput in X-II Topology

4 Proposed Schemes

As shown above, COPE does not make performance improvement for X-II topology. In this section, we propose two schemes to improve the goodput of TCP with COPE. Improvement of the performance is made from two aspects: COPE mechanism and TCP protocols, respectively.

4.1 Encode Once

From simulation, we find that retransmission can not significantly improve the probability to decode a packet if the packet can not be decoded at the first time. For example, in the X-I topology, node n_1 encodes packet p_1 of node $n0$ and p_2 of node $n3$, then $n1$ broadcasts the encoded packet $p_1 \bigoplus p_2$. Node n_0 decodes the encoded packet correctly but node n_3 dose not. Therefore n_3 informs node n_1 that it has not yet obtained packet p_2. Then, node n_1 starts retransmission process,

and encodes packets p_2 and p_3 which is a native packet of $n0$ and broadcasts the encoded packet $p_2 \bigoplus p_3$. However, node n_3 can not decode the encoded packet packet to obtain packet p_2. To solve this problem, we modify the retransmission scheme of COPE. The main idea is that each packet only can be encoded once, which, therefore, we call Encode Once. As a special case, a node directly retransmits a lost native packet instead of encode it again and then send.

Fig. 8 shows the comparison of the goodputs of three TCP's (TCP-NewReno, TCP-FeW and TCP-AP) without COPE, with COPE and with COPE and Encode Once scheme, respectively, in the case of X-I topology. The comparison demonstrates that the goodputs of the three TCP's with COPE and Encode Once scheme are improved by 5%, 4% and 2%, respectively, compared to the TCP's with COPE.

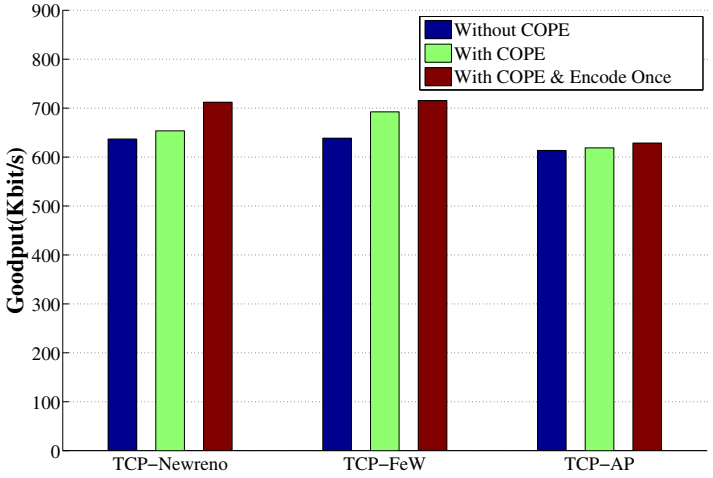

Fig. 8. Goodput of X-I topology with Encode Once

4.2 Network Coding Aware TCP

COPE carries out network coding on the packets within the output queue. So the performance of this scheme depends on the number of packets in the output queue. If the output queue is empty or holds only one packet there is no chance to do network coding, leading to no improvement made by network coding, e.g. TCP-AP with COPE in the X-I topology. If there are too many packets in the output queue, then severe congestion happens, giving rise to bad goodput, e.g., TCP-NewReno and TCP-FeW with COPE in the X-II topology.

To solve this problem, we let the TCP rate be adaptive to the length of the output queue to increase the opportunity of network coding. For example when the output queue does not contain enough packets for encoding for TCP-AP with X-I topology, we increase the number of the packets in the output queue by adapting the rate interval, which is the duration between successive packets to generate an appropriate number of packets. To let the number of packets change

in a reasonable range, i.e., be large enough to implement network coding but not too large to cause the overflowing in the output queue, we introduce a factor λ to adjust the rate interval so that the new rate interval becomes

$$r_{new} = \lambda r_{old} \tag{4}$$

where r_{old} is the original rate interval of TCP-AP and r_{new} is the new one that is adaptive to improve the performance of TCP-AP with COPE. Through various simulations and analyses, we have found an empirical value for λ, namely $\lambda = 0.2$ at which TCP-AP achieves the highest goodput.

Using the TCP protocol with COPE proposed above, we calculate the goodput. The result is illustrated in fig. 9 together with TCP's in other three cases. The compassion of the goodputs in four cases in the figure shows that the proposed new TCP scheme with COPE yields about 12% increase of network goodput compared to TCP-AP with COPE, and performs even better than TCP with Encode Once.

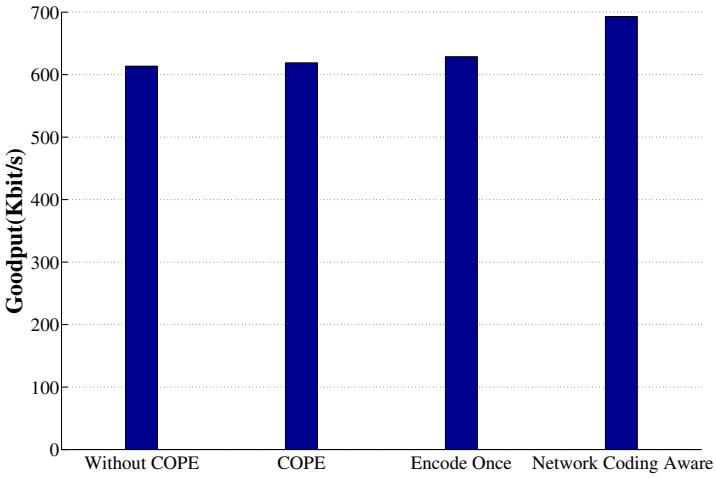

Fig. 9. Goodputs of TCP-AP without COPE, with COPE, with Encode Once and with Network Coding Aware, respectively

5 Conclusions

In this paper, we have presented three TCP protocols (i.e., TCP-NewReno, TCP-FeW and TCP-AP) with COPE and the implementation of them in network simulator NS-2. Then we have evaluated their performances in chain, classic X-I and X-II topologies, respectively. The simulation results demonstrate that in the chain topology both TCP-FeW and TCP-AP with COPE could greatly improve the performances of the TCP protocols without network coding; in the X-I topology COPE results in significant performance improvement for TCP- NewReno and TCP-FeW but not for TCP-AP; and in the X-II topology COPE only improves TCP-AP's performance. After evaluating TCP's with

COPE, we have proposed two schemes to improve the performances (in terms of goodputs) of the TCP protocols with network coding. The first scheme is called Encode Once that ensures the packet being encoded at most one time and the second is called Network Coding Aware TCP in which the transmitting rate of TCP is made adaptive to the status of the node's output queue. The NS-2 simulation results for TCP-AP demonstrate that the proposed two schemes provide significant improvement on the goodputs of TCP's with network coding, and the network coding aware TCP scheme yields even better improvement.

References

1. Ahlswede, R., Cai, N., Li, S.-Y.R., Yeung, R.W.: Network information flow, vol. IEEE Trans. on Information Theory 46(4), 1204–1216 (2000)
2. Wu, Y., Chou, P.A., Kung, S.Y.: Minimum-energy multicast in mobile ad hoc networks using network coding. IEEE Trans. on Comm. 53(11), 1906–1918 (2005)
3. Lun, D.S., Médard, M., Koetter, R.: Network coding for efficient wireless unicast. In: Proc. IZS, Zurich, Switzerland (2006)
4. Katti, S., Gollakota, S., Katabi, D.: Embracing wireless interference: Analog network coding. In: Proc. of SIGCOMM, Kyoto, Japan, pp. 397–408 (2007)
5. Katti, S., Rahul, H., Hu, W., Katabi, D., Médard, M., Crowcroft, J.: XORs in the air: Practical wireless network coding. In: Proc. SIGCOMM, Pisa, Italy (2006)
6. Stevens, W.R.: TCP/IP Illustrated: Volume 1: The Protocols. Addison-Wesley (1994)
7. Wright, G.R., Stevens, W.R.: TCP/IP Illustrated, Volume 2: The Implementation. Addison-Wesley (1994)
8. Nahm, K., Helmy, A., Jay Kuo, C.C.: Tcp over multihop 802.11 networks: Issues and performance enhancement. In: Proc. ACM MobiHoc, Urbana-Champaign, IL, USA (2005)
9. Katti, S., Rahul, H., Hu, W., Katabi, D., Medard, M., Crowcroft, J.: XORs in the air: Practical wireless network coding. IEEE/ACM Trans. on Networking 16(3), 497–510 (2008)
10. Scalia, L., Soldo, F., Gerla, M.: Piggycode: a mac layer network coding scheme to improve tcp performance over wireless networks. In: Proc. GlobeCom, Washington, D.C., USA (2007)
11. Samuel David, P., Kumar, A.: Network coding for tcp throughput enhancement over a multi-hop wireless network. In: Proc. COMSWARE, Bangalore (2008)
12. Kumar Sundararajan, J., Shah, D., Médard, M., et al.: Network coding meets TCP. In: Proc. INFOCOM, Janeiro, Brazil (2009)
13. Ding, L., Wang, X., Xu, Y., Zhang, W., Liu, Y.: Vegas-W: An enhanced TCP-Vegas for wireless ad hoc networks. In: Proc. ICC, Beijing, China (2008)
14. EIRakabawy, S.M., Klemm, A., Lindemann, C.: TCP with adaptive pacing for multihop wireless networks. In: Proc. IEEE MobiHoc, Urbana-Champaign, IL, USA (2005)
15. Floyd, S., Henderson, T.: RFC 2582: The NewReno Modification to TCP's Fast Recovery Algorithm (1999), `http://www.ietf.org/rfc/rfc2582.txt`
16. Ns-2, `http://nsnam.isi.edu/nsnam/index.php/MainPage`
17. Bai, X.L., Wu, P., Ding, L.H.: Evaluation and enhancement of tcp with network coding in wireless multihop networks, Master's thesis, Uppsala University (2011)