Network Coding for Improving the Fairness of Long-Hop TCP Flows in a Multi-Hop Wireless Network

Yong Oh Lee, Manish Kumar Singh
Dept. of Electrical and Computer Engineering, Texas A & M University
Email:[hisfy205, mksingh]@ece.tamu.edu

***Abstract*—Long-hop TCP flows undergo throughput degradation when contending with short-hop TCP flows in a multi-hop wireless network due to collision related losses. Long hop flows compared to the short hop flows have to contend for the medium more number of times and hence suffer more number of collisions. In this paper, we propose a network coding scheme based on the timed network coding to improve the fairness of long-hop TCP flows in a multi-hop wireless network. The network coding reduces the number of transmission in wireless which leads in less contention and less collisions. As per our knowledge, this is the first attempt to apply network coding for alleviating TCP unfairness problem. We corroborate our proposed scheme with simulations in NS-2. Simulation results show that the proposed network coding scheme provides improvement in fairness of long-hop TCP flows.**

***Index Terms*—TCP unfairness, network coding**

I. Introduction

TCP performance within a multi-hop wireless network has been reported in many papers. There are three different types of challenges for TCP design by such networks [1]. First, as the topology changes due to mobility, TCP goes into repeated, exponentially increasing time-outs with severe performance impact. Efficient retransmission strategies have been proposed to overcome such problems [2, 3, 4]. The second problem has to do with the fact that TCP performance in multi-hop wireless network depends critically on the congestion window in use. If the window grows too large, there are too many packets (including ACKs) on the path all competing for the same medium. Increased congestion degrades the throughput [5]. In [18], the authors find the optimal maximum window size corresponding to the number of hops. The third problem is significant TCP unfairness which has been revealed and reported through both simulations and test-bed measurements [6, 7, 8].

Our research focuses on the TCP unfairness problem: unfairness of long-hop TCP flows in a multi-hop wireless network. This problem arises due to several undesirable features of TCP in wireless networks. First, TCP in wireless network is operated under the congested network environment due to ACK bunching and retransmission by collision. Second, TCP confuses wireless channel errors, a frequent occurrence in wireless networks, with congestion related losses and triggers its congestion control mechanism unnecessarily. Third, TCP's fairness is RTT-dependent: Flows traversing a smaller number of hops, or more generally flows with smaller RTT, get a larger share of channel bandwidth than those with larger RTT. Finally, long-hop TCP flow undergoes unfairness because it is exposed to more opportunities of collision-related losses. If the loss occurs, TCP tries to retransmit packets and it makes network more congested. If the network is very congested, the performance of long-hop TCP flows is seriously degraded compared to the short-hop TCP flows.

Fortunately, recent advances in network coding provide hope for a solution to this problem. Network coding was first introduced in [9]. The initial research in network coding was primarily associated with multicast in wired network. But today, we see a number of applications in wireless networks. Network coding in wireless network relies on the the broadcast characteristic of the transmission. Network coding helps to improve the throughput and energy consumption of wireless network. [10] and [11] show that opportunistic network coding increases throughput of networks and [12] and [13] show that increasing opportunity of network coding reduces the number of transmissions, resulting in reduction of the transmission energy.

In this paper, we focus on improving TCP fairness using network coding rather than improving throughput and energy consumption. One of the reasons of TCP unfairness is overall more congestion in long flows compared to that in short flows. Using network coding we intend to decrease the number of transmissions in the wireless network, thus, decreasing the overall congestion in long flows. Network coding is more effective when the network is congested. Decrease in congestion reduces the round trip time of TCP flows and leads to improvement in throughput of network [10]. In addition, RET (retransmission retry time) drop happens when the MAC layer max retransmissions have been exceeded [18]. If RET drop occurs, TCP congestion window size goes down to 1. RTT of long-hop and short-hop TCP flows are different and rate of increase of congestion window is also different. In other words, rate of increase of congestion window of a short-hop TCP flow is higher than that of a long-hop TCP flow. Network coding can help to reduce the RTT and is expected to decrease the large difference in rate of increase of congestion window size between long-hop and short-hop flows [18].

Results from paper [10] show that if all nodes in ad hoc wireless networks are within the sensing ranges (no hidden node problem), we can gain 38% increased goodput of TCP.

In real environment where hidden node problem can't be ignored, performance gain is not so high. Nonetheless, based on the understading of dynamics of networking coding, we can expect that TCP flows with network coding will achieve better fairness than that without network coding.

Our motivation is thus to apply Network Coding in multi-hop wireless networks and utilize its effectiveness to improve the fairness of long-hop TCP flows.

The rest of the paper is organized as follows. Section II reviews the related works. Section III explains why network coding improves TCP fairness. Section IV provides network coding scheme, modified from timed network coding. [15] In section V, we do NS-2 simulation to verify our claim. Section V discusses some issues in our network coding scheme and possible improvements for TCP fairness, and finally concludes the paper.

II. Related Works for TCP fairness

We briefly introduce how the previous research work have overcome TCP unfairness.

Some efforts have addressed the TCP fairness issue in ad hoc networks. In [6], Tang and Gerla et al investigated scenarios, TCP fairness over different MAC protocols. IEEE 802.11 always came on top in terms of both throughput and fairness, but it could not achieve acceptable fairness in the ring and grid topologies. A simple MAC layer technique, an additional yield time was used to restrain the node that used the channel last, is proposed. In [8], Xu et al investigated TCP fairness over 802.11 MAC in ad hoc networks. Their work provides good understanding of the underlying reasons that trigger TCP unfairness, but no remedy is proposed in that work. In [7], Gerla et al investigated TCP unfairness on a path across a wired and multi-hop wireless network. They identified hidden and exposed terminals and the interaction of IEEE MAC and TCP congestion control as the key factors that prevent TCP form stabilizing at fair-share. Instead of focusing on channel and MAC protocol features in an attempt to identify the factors triggering TCP unfairness, [1] and [14] modify RED scheme to solve the TCP unfairness problem. [1] proposed Neighbor RED which extends the RED concept to distributed neighborhood queue. Zhenghua Fu et al [14] proposed link RED and adaptive pacing which made link drop probability sufficient to stabilize the large TCP window size.

III. Network Coding for TCP fairness in wireless multi-hop networks

We propose network coding scheme to overcome unfairness of long-hop TCP flows in multi-hop wireless networks.

In network coding, we allow an intermediate node to combine a number of packets it has received or created into one or several outgoing packets. Assume that each packet consists of L bits. When the packets to be coded do not have the same size, the shorter ones are padded with trailing 0s. However, coded packet with large size of packet and small size packet does not show significant improvement of throughput of network [10]. Our network coding scheme only combines TCP data packets, not ACK packets. With linear network coding, outgoing packets are linear combinations of the original packets, where addition and multiplication are performed over the field. The reason for choosing a linear framework is that the algorithms for coding and decoding are well understood [9].COPE [10] shows improvement of throughput with opportunistic network coding in wireless networks by using the broadcast nature of the wireless medium. Fig. 1 is a simple example of network coding in a wireless network. Node A and node B want to exchange packets via an intermediate node S. In the traditional method (without network coding), four transmissions are required to exchange packet a from node A to node B and packet b from node A to node B. However, we can reduce the number of transmissions with network coding. In the network coding when the intermediate node S receives both packets a and b, it broadcasts the XORed packet (a xor b) to both node A and node B. Both A and B can recover the packet of interest using XOR operation, while the number of transmissions is reduced to three from four.

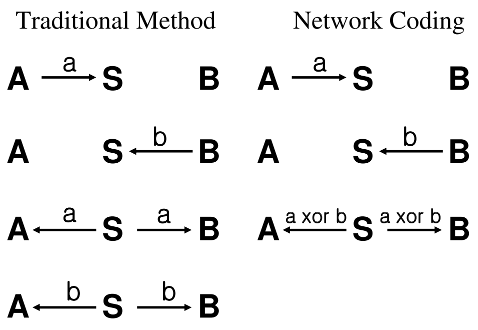

Fig. 1. Traditional and network coding in wireless network

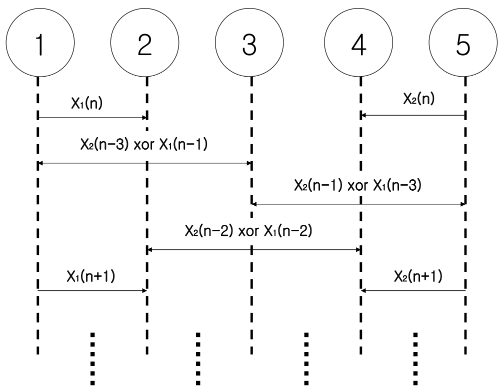

Fig. 2. Packet transmission using network coding in wireless network

Fig. 2 is the network coding gain of linear chain topology in wireless network. The limitation of the network coding gain is that there must be two or more flows whose direction is opposite. In addition, the network should be very congested which gives more opportunity to code several packets in the receiving queue.

IV. Introducing the Network Coding Scheme

In this paper, we use pseudo-broadcast for MAC layer communication and timed network-coding as network coding policy. We introduce the following network coding policy similar to timed network coding [15]. As defined in [10], an innovative packet is defined as a packet contributes to make a combined packet. A packet which cannot contributes to make a combined packet is a non-innovative packet. Before detailed scheme, we introduce the '*Packet Pool*'. When a packet is received at a node, it enters into packet pool before entering the queue. The packet will stay in the packet pool waiting for an other packet to be combined together for transmission. If packets are already in the pool, an innovative packet may immediately combined together with a waiting packet. All the packets entering packet pool are soted for decoding in the future.

A. Pseudo broadcast with RTS/CTS hand-shaking

Pseudo broadcast with RTS/CTS hand-shaking is based on the IEEE 802.11 DCF. Broadcasting is not suitable for network coding due to collision. IEEE 801.11b broadcasting is not suitable for network coding due to collisions which results in throughput degradation. Pseudo broad cast is more reliable transmission than IEEE 802.11b, but it also occur collisions such as hidden/exposed collisions. Without RTS/CTS handshaking, the network throughput is reduced due to so many collisions. [10] To prevent such collisions, pseudo broadcast with RTS/CTS handshaking use RTS/CTS mechanism. RTS/CTS mechanism not only prevents collisions, but also is more reliable because of the retransmission even when collision occurs. The combined packet by network coding has two or more next nodes to forward. Pseudo broad cast choose one of them and put its address into the header field. (The chosen next node is defined as $NextNode_{PB}$) A node with longer RTT of that requires better end-to-end delay is chosen as $NextNode_{PB}$. For example, in the TCP case, pseudo broadcast with RTS/CTS handshaking choose the node in longer hop as $NextNode_{PB}$. [10] It improves TCP fairness because long-hop flow undergoes unfairness against the short-hop. The sending node exchanges RTS/CTS message with $NextNode_{PB}$. After RTS/CTS message, the neighbor nodes of sending node and $NextNode_{PB}$ enter the NAV state. However, for receiving the combined packet in $NextNode_{PB}$, the node in NAV state should wake up during the DATA duration. After receiving DATA, only $NextNode_{PB}$ sends ACK to sending node. Pseudo broadcast with RTS/CTS induces RTS+CTS+3SIFS+ACK delay compared with pseudo broadcast.

B. Timed network coding

Probabilistic, Semi-deterministic and Timed network coding are proposed in [2]. Probabilistic and semi-deterministic network coding has forwarding factor, ρ. In probabilistic network coding, a new incoming packet is combined with previous delivered packet with probability ρ. In semi-deterministic network coding, each node receives exact given number of packets and then create combained packet. These two schemes suffer long waiting time for combining packets due to collision or loss. However, timed network coding makes combined packet with the received packet during τ interval. τ interval value is uniform random variable in [0, τ_{max}]. Timed network coding also has an additional delay, but, even in the worst case, the waiting time for network coding is bounded τ_{max}. In this paper, timed network coding is modified for reducing delay. First, t interval is fixed rather than randomly chosen. Second, three conditions are added.

Policy 1: The new incoming packet triggers a timer and waits for creating a combined packet only if the IFQ (queue between Link layer and Mac layer) is occupied.

To reduce the delay, new incoming packet can wait when the queue is occupied. This policy is based on the idea that if the queue (IFQ) is empty and a a combined packet can't be combined, the network throughput decreases. Furthermore, if the queue is occupied by the other packet and new packet is queued, the new packet does not do anything while the previous queued packet is being served, (This is the case of no network coding) even though there is an opportunity to create a combined packet by network coding. So, if the queue is occupied, the new incoming packet waits for another new packet in the *Packet Pool* to make a combined packet instead of getting queued. With the waiting, timer starts.

Policy 2: If a non-innovative packet is received, the awaiting packet for making a combined packet enters into the queue and the new incoming packet (the recieved non-innovative packet) starts waiting for an innovative packet. The previous timer is stopped and a new timer starts.

If it receives non-innovative packets, which cannot be used for network coding, the previous packet waiting in the Packet Pool is queued and scheduled to transmit without network coding, and the timer is stopped. Then the non-innovative packet starts waiting and trigger the new timer. This policy can reduce the additional delay as well as prevent reordering in the destination node.

Policy 3: If the new incoming packet is innovative packet and the timer is not expired, a new combined packet is created with XOR operation and it enters the queue(IFQ).

If an innovative packet is coming before the timer is expired, a new combined packet is made with XOR operation. In the proposed timed network coding, the probability of making a combined packet is 1. After making a combined packet, the combined packet enters the queue and is scheduled to be transmitted.

The packets that the node received are stored in the Packet Pool. When the node receives a new packet, it checks its header whether the packet is a combined packet or a native (not combined) packet. If the packet is a combined packet, the node checks whether it is decodable with information in the Packet Pool. If the combined packet is decodable, it is decoded

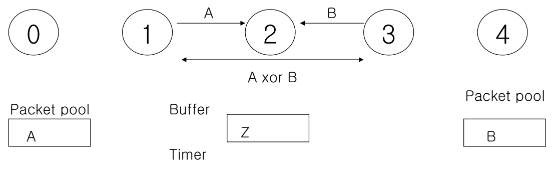

Fig. 3. Example of proposed timed network coding

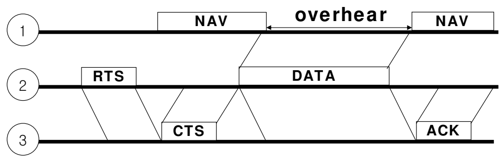

Fig. 4. Example of pseudo broadcasting with RTS/CTS handshaking

and the decoded packet is treated as a new coming packet. If the combined packet is not decodable, it is discarded.

Here is an example. We consider the linear chain topology case. When a new incoming packet is coming form the opposite direction to the direction where the previous waiting packet came. In ideal case, which each node always has a packet to send and scheduling is perfect, the throughput gain is 2. But, in the practical scenario, it is less than 2. Node 1 sends packet A to Node 2. If the queue of Node 2 is empty, packet A enters the queue directly without waiting in Packet Pool. If packet Z is in the queue of Node 2, packet A waits in the Packet Pool and triggers the timer A. Then, if Node 1 sends packet B, packet A enters the queue of Node 2 and stops timer A. And then, packet B waits in the Packet Pool and triggers the timer B. In the other hand, if Node 3 sends packet B and the timer A is not expired, new combine packet A xor B is made and enters the queue. Node 3 is selected as $NextNode_PB$, Node 2 sends RTS to Node 3 and receives CTS from Node 3. Node 1 and Node 4 enters NAV state after receiving RTS and CTS. Node 1 and Node 4 awake up during DATA interval. Node 1 and Node 3 receive the combined packet A xor B. Node 1 has packet A in the Packet Pool and Node 3 has packet B in the Packet Pool. So Node 1 and Node 2 can decode B and A.

V. Simulation

To verify that network coding improves the fairness of long-hop TCP flows in a multi-hop wireless network, we carried out simulations in NS-2.

We use linear chain topology (fig. 5) and the simulation parameters are below in Table 1.

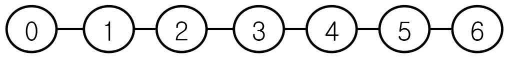

Fig. 5. Linear chain topology

TABLE I
ENERGY-EFFICIENT DATA RATE SELECTION ALGORITHM

MAC protocol: IEEE 802.11
Routing protocol: NOAH [19]
Transport protocol: TCP
Traffic: FTP Signal Propagation
model: Two-ray model
Network topology: Linear chain topology with 7 nodes
Packet size: 1Kbyte
Simulation time: 100 sec

Nodes are placed in linear chain topology at a distance of 240 meters from its neighbor. The choice of this distance is based on the following simulation criteria. We intend to do away with hidden node problem in our simulation. Even though 802.11 employs RTS/CTS mechanism to handle hidden problem, the problem still persists in wireless environment where normally interference range is larger than transmission range [16]. We use the design criteria ($d_{node} = 0.56 * R_{tx}$) suggested in [16] to calculate the suitable inter-node distance for our simulation. Here d_{node} is the inter-node distance and R_{tx} is the transmission range of the nodes (same for all).

For a transmission range of 446 meters obtained for the physical-layer parameters chosen in our simulation, we get a d_{node} of less than 250 m to avoid hidden node problem. We choose 240m as the inter-node distance.

First, we simulate the impact of timer value on the network performance because timer value is critical value for network coding performance. There are two flow: one is from node 0 to node 4, and the other is from node 4 to node 0.

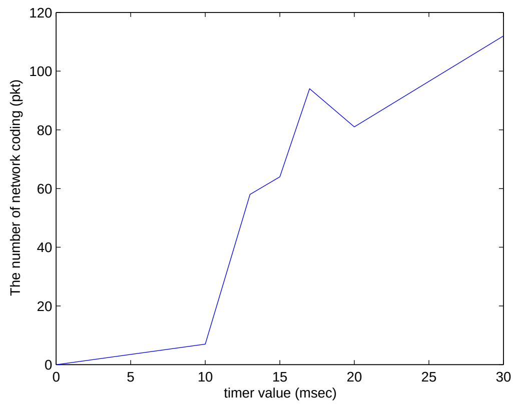

Fig. 6. The number of network coding in TCP

As the timer value increases, the number of combining opportunities increases. (fig. 6) However, the problem is the network throughput. (fig. 7) Compare with the no network coding (timer value is 0), there is no throughput improvement in TCP. Except 15msec and 17msec of the timer value, the network throughput is worse than no network coding case. The reason is inefficiency of retransmission of TCP flow after a the collision. In addition, the number of network coding is so low compared with the total number of transmission. This is because TCP cannot generate enough traffic to make network coding work. The network coding is not much in TCP, but timed network coding increases the delay, so the network

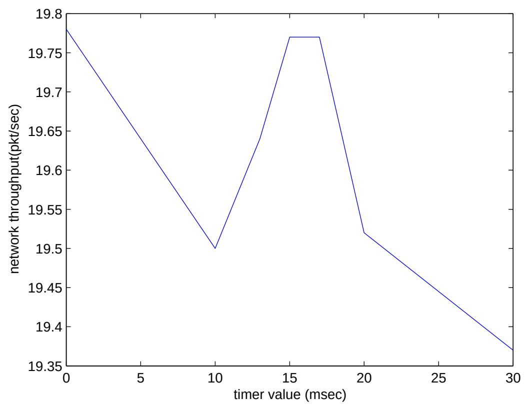

Fig. 7. The network throughput in TCP

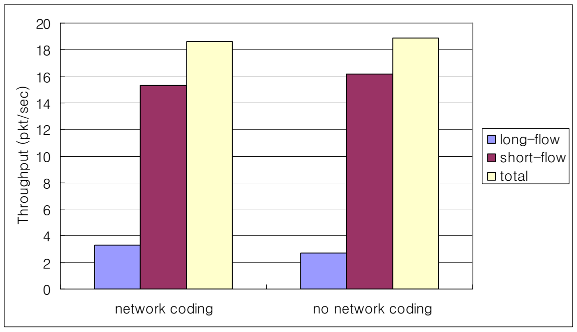

Fig. 8. The network throughput comparison between network coding and no network coding

throughput is not improved. In the given scenario and previous work [15], the timer value between 15 msec and 17 msec is the optimal timer value for network throughput in the case of linear chain topology.

So we choose 15 msec of timer value. For simulation of TCP unfairness, we use 7 nodes in linear chain topology (Fig 5) and generate four TCP flows: two of them are long-hop TCP flows (flow 1 and flow2) and the other two are short-hop TCP flows (flow 3 and flow 4) The long-hop flows are in the opposite direction to the short-hop flows for network coding to work. The long-hop flows are from node 0 to node 6. The short-hop flows are from node 5 to node 2. Measurement metric is throughput of each flow.

Fig. 8 is throughput of short- and long-hop flow without network coding and with network coding. The total throughput shows small improvement. The throughput of short-hop is decreased and the throughput of long-hop is increased with network coding compared to no network coding. Ratio of the throughput of the two flows, give a raw idea of the fairness between the two flows. Specifically, for our purpose, we define fairness index as ratio of the throughput of long flow to the throughput of short flow. In this simulation, the fairness index of no network coding is 0.15 and the fairness of network coding is 0.18.

Lastly, we see the throughput of long-hop and fairness with changing the ratio of hop number of long-hop to hop number of short-hop. The hop number of long-hop is fixed as 6, and the hop number of short-hop varies from 6 to 3. We can see TCP unfairness in fig. 9. When the ratio of hop number of long-hop to hop number of short-hop is 1 (the hop number of both flows is 6), the fairness index is almost 1. As the ratio of hop number of long-hop to hop number of short-hop increases, the fairness index decreases as shown in fig. 9. However, network coding improves the TCP fairness compared to no network coding. When the ratio of hop number of long-hop to hop number of short-hop is 1.5, the fairness index improves from 0.16 without network coding to 0.21 with network coding. When the ratio of hop number of long-hop to hop number of short-hop is 2, the fairness index improves from 0.15 without network coding to 0.18 with network coding. As the network coding is used, the number of transmission is reduced. As a result, throughput of

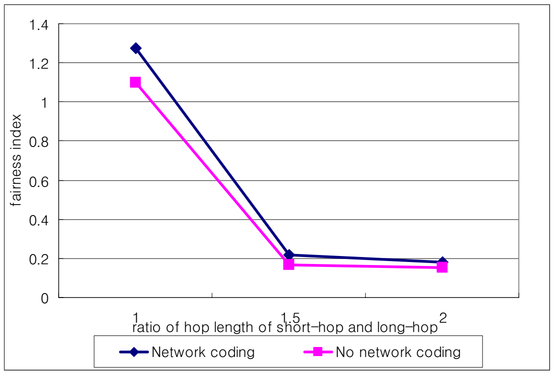

Fig. 9. Fairness index

long-hop increases because of lower contention and collision related loss. In [18], authors proposed a maximum window size for improving TCP fairness. However, our simulation results dont show appreciable improvement in the TCP fairness with bounded maximum window size for both the long and short TCP flows. In the simulation, network coding helps reduce RTT and recovery congestion window after the size of congestion window becomes 1 due to 3 consecutive time-out for long-flows.

VI. Conclusion and Future Works

We considered unfairness of long-hop TCP flow in wireless network and proposed a scheme based on network coding to alleviate the problem. As we can see from the simulation results, long-hop TCP undergoes the unfairness in throughput. The main cause of unfairness is the high congestion and collision which results in retransmission. Network coding reduces the number of transmission in wireless networks and makes more combined packets when network is congested. We propose network coding scheme for TCP fairness. It is based on the timed network coding. To overcome the additional delay of timed network coding, we proposed monitoring the queue occupancy and found the timer value which can make more combined packets. Our simulation results show improvement of TCP fairness.

To obtain more TCP fairness improvement, the opportunity of network coding should be increased and the number of combined packet should be increased. We expect to obtain more TCP fairness by implementing network coding for random topology, beyond the linear chain topology. In the random topology, we can make network more congested and it leads to more opportunities of network coding. Also, we make coded packet with three or more packets. We will provide simulation results of random topology before the submission of camera-ready version of this paper.

ACKNOWLEDGEMENT

We thank Dr. Reddy for his insightful guidance, support, and feedback at this work. We also would like to thank Dr. Sprintson for discussion on this topic.

REFERENCES

[1] Kaixin Xu, Mario Gerla, Lantao Qi, Yantai Shu, ”Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED”, Proceedings of the 9th annual international conference on Mobile computing and networking, 2003

[2] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A feedback-based scheme for improving TCP performance in ad hoc wireless neworks. IEEE Personal Communications Magazine, 8(1), Feb. 2001.

[3] T. D. Dyer and R. V. Boppana. A comparison of TCP performance over three routing protocols for mobile ad hoc networks. Proceedings of ACM MobiHoc’01, Oct. 2001

[4] G. Holland and N. H. Vaidya. Analysis of TCP performance over mobile ad hoc networks. Proceedings of ACM MobiCom’99, Aug. 1999.

[5] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on TCP throughput and loss. IEEE INFOCOM’03, Mar. 2003.

[6] K. Tang and M. Gerla. Fair sharing of MAC under TCP in wireless ad hoc networks. Proceedings of IEEE MMT’99, Oct. 1999

[7] K. Xu, S. Bae, S. Lee, and M. Gerla. TCP behavior across multihop wireless networks and the wired internet. Proceedings of ACM WoWMoM’02, Sep.2002

[8] S. Xu and T. Saadawi. ’Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?’, IEEE Communications Magazine, 39(6), Jun. 2001.

[9] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, ”Network information flow,” IEEE Trans. on Information Theory, vol. 46, pp. 1204-1216, 2000

[10] S. Katti,H. Rahul, Wenjun Hu, D. Katabi,M. Medard, J. Crowcroft,”XORs in the Air: Practical Wireless Network Coding,” SIGCOMM, 2006.

[11] S. Sengupta, S. Rayanchu, S. Banerjee,”An Analysis of Wireless Network Coding for Unicast Sessions: The Case for Coding-Aware Routing”,INFOCOM 2007, May 2007

[12] Danail Traskov, Niranjan Ratnakar, Desmond S. Lun, Ralf Koetter, and Muriel Medard, ”Network Coding for Multiple Unicasts: An Approach based on Linear Optimization”,ISIT 2006, July 2006

[13] Sukwon Kim, Michelle Effros, and Tracey Ho, ”On Low-Power Multiple Unicast Network Coding Over a Wireless Triangular Grid”, Forty-Fifth Annual Allerton Conference, September 2007

[14] Zhenghua Fu, Petros Zerfos, Haiyun Luo, Songwu Lu, Lixia Zhang, Mario Gerla, ”The Impact of Multihop Wireless Channel on TCP Throughput and Loss”, in IEEE INFOCOM’03, San Francisco, March 2003

[15] E. Fasolo, M. Rossi, J. Widmer, J and M. Zorzi, ’On MAC Scheduling and Packet Combination Strategies for Practical Random Network Coding’, ICC ’07

[16] Kaixin Xu, Mario Gerla, Sang Bae ”How Effective is the IEEE 802.11 RTS/CTS Handshake in Ad Hoc Networks?”, GLOBECOM 2002.

[17] A.Raniwala, P.De, S.Sharma, R. Krishnan T.Chiueh, ”End-to-End Flow Fairness over IEEE 802.11-based Wireless Mesh Networks”, INFOCOM 2007

[18] Wei Ren and Hai Jin,”A Combination Scheme to Improve TCP Throughput over Multihops Wireless Mobile Ad Hoc Networks”, LNCS, Volume 3942, pp 805-809, 2006 March.

[19] http://icapeople.epfl.ch/widmer/uwb/ns-2/noah/

APPENDIX

A. *Implementation of network coding in NS-2*

We have implemented Network Coding in NS-2 as described in the following discussion. Network coding is performed between link layer (LL) and interface queue (IFQ). Before enqueueing packets in IFQ (maintained as FIFO queue in NS-2), there is a memory area, called as ’Packet Pool’, which buffers the packets coming to the LL. In our network, coding policy is as follows; If a packet is received at the LL and the queue(IFQ in NS2) is empty, the received packet is scheduled and put in the IFQ without worrying about network coding.If a packet is received at LL and the queue (IFQ) is occupied with other packets,, the received packet waits for an network coding opportunity until the timer expires (based on the determined timer value). (We can make an coded packet with packets from opposite directions in the linear chain topology). If the next packet comes from the opposite direction, coded packet is generated and enqueued to IFQ (packets are scheduled). If the next packet comes from the same direction, the previous packet is enqueued without network coding, but the new packet waits in the packet pool for an network coding opportunity for the duration of determined timer value. The packets in packet pool are also used for decoding coded packets. If coded packets are able to be decoded with the packets in packet pool, decoding is performed and native packet (original packet before encoding) is recovered. However, if coded packets are impossible to be decoded because of lack of packets in packet pool, the coded packet is discarded.

B. *Algorithm details*

We have implemented Network Cording module as part of the link layer and hence perform the coding when the packets are received at LL and are due to moved into IFQ. The downward flow of packets in NS-2 traverses the route as IP (Routing agent)$- >$LL$- >$IFQ$- >$MAC$- >$PHY(NetIF) whereas the upward flow follows PHY$- >$MAC$- >$LL$- >$IP. IFQ as we can observe isn’t involved in the upward flow and hence was discarded by us as a probable candidate layer for implementing the Network Coding. LL is the preferred choice.

To implement network coding, a few modifications were made in the LL Class (in NS-2) and a few fields were added to the common packet header (hdrcmn) defined in packet.h. A LL Timer was also defined. Mac 802.11 module was also modified to support pseudo-broadcast. Below follows description of the changes made.

Changes to LL

Link Layer is defined in ll.cc and ll.h in <ns-version> \mac directory. Class LL is defined in ll.h. A few functions are defined in ll.cc. As part of the changes to LL class, three protected members lltimer, code buffer [BUFFERSIZE] and buffercount were added. The class LLtimer was added as a friend class. Also the functions void sendUp(Packet$*$ p) and void sendDown(Packet$*$ p) were modified.

codebuffer[BUFFERSIZE] serves as the ”Packet Pool”. It stores the packets (number constrained by the BUFFERSIZE) which are moved to the queue (IFQ) from the LL. These packets could either have been involved in network coding or just plainly have been forwarded to the lower layer. These packets are later used for decoding purpose. This packet pool is maintained like FIFO queue and hence when size of the pool approaches BUFFERSIZE, the first packet stored is removed.

buffercount keeps the current index of last packet in the buffer and is $\geq$ BUFFERSIZE -1.

void sendUp(Packet$*$ p) function is modified to decode a encoded incoming packet. A packet is decodable if the network coding components (packets) exist in the ”Packet Pool”. To identify the encoding components, the coded packet is appended with pkt ids of the packets involved in the coding. Hence, a check is made for the existence of encoding components and if they are present, decoding is done otherwise packet is dropped or discarded.

void sendDown(Packet$*$ p) function is modified to encode two encodable packets together. We use XOR operation to encode the packets. Packets qualify for network coding if they are destined in opposite directions. If an incoming packet is not encodable with the existing un-scheduled (scheduling is done by LL layer) packet in the ”Packet Pool”, the buffered packet in the pool is scheduled to be queued in IFQ and the newly arrived packet is buffered in the pool. It then waits for an encoding opportunity with an incoming packet until the LLTimer expires. Hence at any point of time, we have only one packet outstanding to be scheduled in the Packet Pool. As such the Packet Pool stores all the incoming packets but only the packets due to be scheduled. If the incoming packet can be encoded with an existing buffered packet, both are XORed and the coded packet is scheduled to be pseudo-broadcasted. This basically means that the coded packet is unicasted in the direction of long-hop flow and is also received by the neighboring node in the directon of short-hop flow. In fact, the common packet header and MAC src-dest address of the encoded packet is appropriately modified before being scheduled.

Defining LL Timer

We implemented a Link Layer Timer (class LLTimer) to realize timed Network Coding.

Two new codes were written as part of LL Timer implementation.

(1) ll-timers.cc

(2) ll-timers.h

These two codes , present in <ns-version> \mac directory ,define LLTimer which is used in the LL class to time the wait for an encoding opportunity by a packet in ”Packet Pool”.

Three functions namely void start(double time) , void stop(void) and void handle(Event$*$) constitute the LL Timer.

Changes in Mac-802.11 Module

The code mac-802.11c (in <ns-version> \mac directory) is modified to implement pseudo-broadcast of encoded packets. The procedure of Address filtering is extended to allow the packets with encoded bit set to be received by the MAC layer. Normally only the MACBROADCAST packets and the packets destined to the present MAC layer are allowed to be received by the MAC layer.The other packets are discarded. The functions recvtimer() and recvDATA were modified to handle the encoded packet that is overheard by the MAC layer.

Changes to Packet Header

Packet structure is defined in packet.h present in <ns-version> \common directory.

This header file is modified to include two new fields in the common packet header (struct hdrcmn).

int encoded field is used to keep track of either a packet is an encoded packet int pktids - this array is used to store the packet ids of the encoded components. This is used during decoding process to identify whether an incoming encoded packet is decodable based on the packets present in the Packet Pool.