Network Coding for TCP Throughput Enhancement over a Multi-Hop Wireless Network

P. Samuel David[†] and Anurag Kumar[‡]

***Abstract*—The poor performance of TCP over multi-hop wireless networks is well known. In this paper we explore to what extent *network coding* can help to improve the throughput performance of TCP controlled bulk transfers over a chain topology multi-hop wireless network. The nodes use a CSMA/CA mechanism, such as IEEE 802.11's DCF, to perform distributed packet scheduling. The reverse flowing TCP ACKs are sought to be X-ORed with forward flowing TCP data packets. We find that, without any modification to the MAC protocol, the gain from network coding is negligible. The inherent coordination problem of carrier sensing based random access in multi-hop wireless networks dominates the performance. We provide a theoretical analysis that yields a throughput bound with network coding. We then propose a distributed modification of the IEEE 802.11 DCF, based on tuning the back-off mechanism using a feedback approach. Simulation studies show that the proposed mechanism when combined with network coding, improves the performance of a TCP session by more than 100%.**

***Index Terms*—multi-hop wireless networks, network coding, TCP performance over wireless networks**

I. Introduction

It has been shown that if router nodes in networks combine packets (e.g., by taking linear combinations), before forwarding them, (a technique called *network coding*) then the packet throughput of the network can be increased [1]–[4]. A simple example, depicted in Figure 1, illustrates how such a mechanism (also called "X-ORs in the Air" [4]) can improve the throughput. It is assumed that Nodes A and C store the packets that they have forwarded for some time duration. After Node B receives a pair of packets from A and C, it XORs the packets and, using wireless broadcast, transmits the XORed packet so that both A and C can attempt to decode it. Nodes A and C, after receiving the X-ORed packet, can use the packets they have previously stored to decode the packets actually meant for them. We see that instead of 4 transmissions overall, the same data transfer can be achieved with 3 transmissions. With reference to Figure 1, considering a TCP transfer from left to right, we can exploit the above potential by combining the rightward flowing TCP data packets with the leftward flowing TCP ACKs. Since, in the original system a separate contention needs to be made for each packet, whether data or ACK, by enabling network coding over a string topology, we expect an appreciable increase in the overall throughput of the TCP controlled transfer.

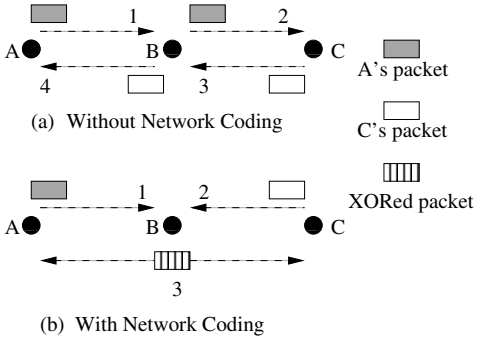

Fig. 1. An example to illustrate the network coding of packets flowing in opposite directions in a multi-hop wireless network.

However, we find that just implementing the above mechanism does not provide any substantial improvement in throughput if the channel access mechanism is itself left unaltered. For multi-hop wireless networks in which not all nodes are within the transmission range of each other, the behavior of the existing CSMA/CA based IEEE 802.11 MAC protocol is known to be far from optimum. Many inefficiencies of the IEEE 802.11 MAC over multi-hop networks have been pointed out in the literature. In [5]–[7], the authors conduct various simulations and conclude that the current version of the IEEE 802.11 is inefficient in its function as a MAC layer protocol for multi-hop wireless networks. Here our concern is with the problem of lack of coordination between nodes when CSMA/CA is used in a multi-hop environment.

Thus, in this paper we consider modifications to the IEEE 802.11 MAC protocol in an effort to derive the expected advantage of network coding for TCP controlled transfers over multi-hop wireless networks. This paper in no wise attempts to design an architecture for performing network coding but uses a very simple model described in the later sections, to perform network coding and in turn brings out the intricacies involved and the required modifications necessary in order to build an efficient architecture.

Before we report simulation results and discuss the proposed modifications, we provide a theoretical analysis to obtain bounds on the performance of TCP over a chain topology wireless network whose nodes use CSMA/CA for channel access. Such an analysis can enhance our understanding of TCP over multi-hop wireless networks, and also provide a bound against which to evaluate the proposed modifications. We then propose a distributed scheme that works by using packet drop events at each node as a feedback to tune the

Work on this project was supported by a research grant from Motorola.

[†]Indian Institute of Technology, Madras; this work was done while this author was visiting the Indian Institute of Science under an Indian National Academy of Engineering Summer Fellowship; email: samuel.david@smail.iitm.ac.in

[‡]Dept. of Electrical Communication Engg., Indian Institute of Science, Bangalore; email: anurag@ece.iisc.ernet.in

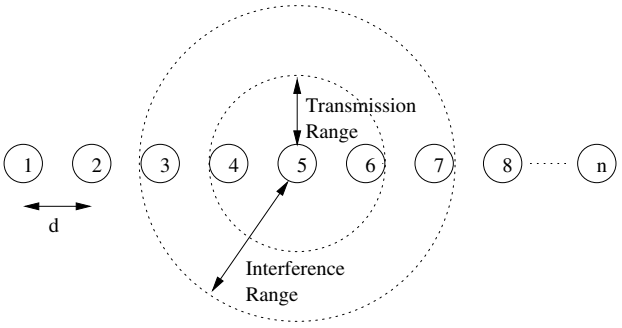

Fig. 2. A string topology of n nodes, the neighbors separated by distance d. The example shows an interference range of two hops; i.e, $k = 2$.

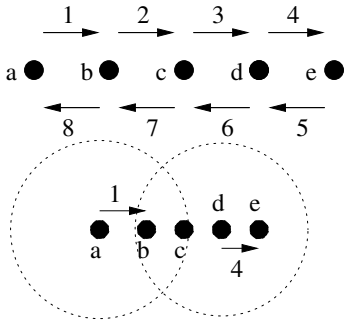

Fig. 3. A multi-hop network string topology with $n = 5$ and $k = 2$. $\{1, 5\}, \{1\}, \{2\}, \{3\}, \ldots \{7\}, \{8\}$ and the empty set are the only feasible link activation sets.

IEEE 802.11 backoff mechanism of that node.

The rest of the paper is organised as follows. In Section II, we introduce the network model, and the interference model, and we describe the way network coding is implemented. In Section III, we obtain the maximum throughput achievable for a window based transport layer protocol. Then, we obtain upper bounds on the throughput of a TCP session over the string topology. We simulate the performance of TCP over IEEE 802.11 with and without enabling network coding; these simulations reflect the undesirable behavior of CSMA/CA over multi-hop networks and are provided in Section IV. Based on the observations made in the above sections, in Section V, we propose a modification to IEEE 802.11 for use in the multi-hop chain topology and obtain its performance. We conclude in Section VI.

II. Network and Interference Model

We consider a multi-hop network of n nodes, in which the nodes form a string topology. The nodes are indexed $1, 2, 3, \cdots n$. The neighbor nodes are equally spaced at a distance d; see Figure 2. In the absence of any other transmission (and therefore, any interference) each node communicates with its immediate one hop neighbor only. Pairs of nodes that can directly communicate are viewed as having "links" between them; thus there is a link between node 1 and node 2, etc. The links are half-duplex. In addition, arbitrary links cannot be used simultaneously owing to interference. Each link's propagation delay is very small compared to the packet transmission time.

A. Definitions:

1) Link transmission rate: All radios use the same transmit power, and transmit at the same bit rate of Γ bits per second.

2) Interference range (k): An interference range of k implies that the transmission from a node can interfere with the reception at any node less than or equal to k hops away from the transmitting node. Define N_0 as the receiver noise power normalised to the transmit power, and β as the minimum SINR (Signal to Interference plus Noise Ratio) required at a receiver for successful reception. Then k is such that $\dfrac{\left(\frac{d}{d_0}\right)^{-\zeta}}{N_0+\left(\frac{kd}{d_0}\right)^{-\zeta}} < \beta$ and $\dfrac{\left(\frac{d}{d_0}\right)^{-\zeta}}{N_0+\left(\frac{(k+1)d}{d_0}\right)^{-\zeta}} \geq \beta$ where d_0 is a reference distance, and ζ is the power law attenuation factor. We only consider distance attenuation and do not account for random fading. The above definition states that the SINR at a receiver due to an interfering node k hops away and a transmitting node 1 hop away will be less than the minimum SINR required. If the interfering node is $(k+1)$ hops away, the received SINR will be greater than the minimum. In Figure 2, the value of k is shown to be 2.

3) Network Coding Efficiency (γ): As we consider a single session, we define network coding efficiency, $\gamma, 0 \leq \gamma \leq 1$, as the ratio between number of transmissions of XORed (encoded) packets at all nodes and the total number of transmissions of TCP data packets (with and without encoding). Evidently, values of γ close to 1 are desirable so that the contention offered by the backward flow of TCP-ACK packets is small.

B. Link Activation Constraints

The above definition of k results in the following interference constraints:

1) When a node is receiving, there can be only one transmitting node in its $k-$hop neighborhood.
2) When a node is transmitting, there can be only one receiving node in its $k-$hop neighborhood.

As an example, let us consider the feasibility of simultaneous activation of Links 1 and 4 in Figure 3. Since $k = 2$, transmission from node d to node e would result in interference at node b. This violates the first constraint, as node b has two transmitting nodes in its $k-$hop neighborhood. Therefore $\{1, 4\}$ is not a feasible activation set. Similar analysis can be done for all possible combinations and the feasible sets are mentioned in Figure 3. Such link activation sets are useful for us later in order to obtain bounds on the maximum throughput achievable.

C. Network Coding Model

For the string topology that we consider, we define the forward flow of TCP-DATA packets as Flow 1 and the backward flow of TCP-ACK packets as Flow 2, and explore network coding opportunities between Flow 1 and Flow 2. Our model for network coding is similar to that used in [4]. We describe our model below:

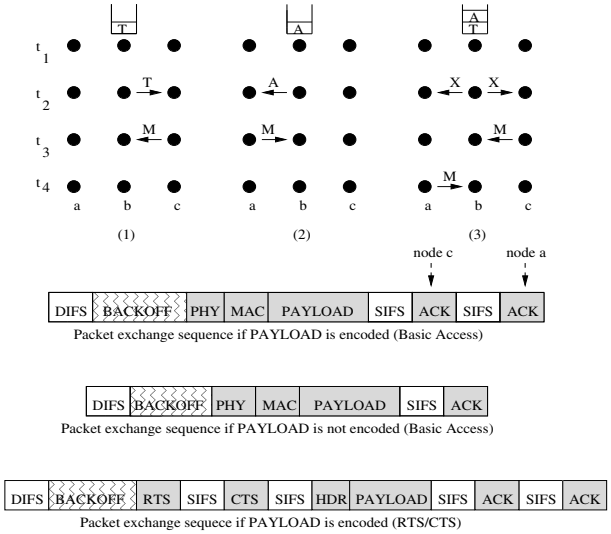

Fig. 4. Three scenarios of packet transmission between consecutive nodes a, b and c; shown as Panels (1), (2) and (3). t_1, t_2, t_3, t_4 are the successive time instants of packet trasmissions where T=TCP-DATA, A=TCP-ACK, X=XORed packet, M=MAC-ACK, HDR=PHY+MAC.

a) Just before the transmission of a packet (TCP-DATA/TCP-ACK) belonging to a particular flow, the transmitting node searches its own transmit queue for a complementary packet (TCP-ACK/TCP-DATA) belonging to the other flow. If a corresponding packet is found, both packets are XORed together and the resulting XORed packet is transmitted.
b) Before the transmission of a packet, if a complementary packet is not found in the transmit queue, the transmitting node will not wait for a complementary packet to arrive. Rather, it would just transmit the available packet as a normal unicast transmit.
c) All the nodes store the transmitted packets for a sufficiently long duration as these could be of potential use in decoding the encoded packets that would be received in the near future.
d) The encoded packet contains identifiers of the packets XORed and is identified by a broadcast address in the MAC header. The transmission time of an XORed packet is the supremum of the transmission times of the packets encoded, i.e., the TCP-DATA packet transmission time $T_{TCP-DATA}$. Whenever a node receives a broadcast packet, it examines the identifiers of the packets XORed and then attempts to decode the encoded packet.
e) As in [4], The encoded packets are transmitted as *pseudo-broadcast*. However, we use synchronous ACKs in our model. If a node transmits an XORed packet, the transmitting node expects MAC-ACKs from both the receiving nodes. The ACK transmissions are scheduled in a synchronous manner as shown in Figure 4. Following the successful reception of an XORed packet, the node ahead transmits a synchronous MAC-ACK after a SIFS. After another SIFS, the node behind transmits its MAC-ACK.
f) The above order is always maintained whenever the packet transmitted is an encoded (XORed) packet. If either of the encoded packets is not ack-ed within their corresponding timeouts, the transmitting node retransmits that particular packet, potentially encoding it with another packet. If the transmitting node transmits a packet that is not encoded, then the packet exchange sequence is the same as for unicast transmissions.
g) The packet exchange sequence is similar for the RTS/CTS mechanism, except that we have only one RTS/CTS exchange with the forward node. We use only the basic access mechanism in our simulations.

A customized patch for the ns simulator was developed to simulate the above model. In the actual simulation, the nodes donot implement the above algorithm literally and the packets are not really XORed but the simulation mimics the above model exactly.

III. Performance Bounds

The performance measure that is of main concern to us is the TCP throughput. Formally, we define TCP throughput as follows. Assuming that each TCP DATA packet causes a TCP ACK packet to be transmitted, let $\Omega(t) =$ the number of returning TCP-ACKs in the time interval $[0, t]$, then throughput τ is

$$\tau = \lim_{t\to\infty} \frac{\Omega(t)}{t}$$

assuming that the limit exists. We consider node 1 as the source node and node n as the destination node for the string topology.

A. Bounds for Perfect Link Scheduling

Let us consider the following straight forward results which were pointed out in [8]:

Lemma 3.1: The maximum number of simultaneous transmissions possible, for a string topology of n nodes is,

$$T_{max} = \left\lceil \frac{n-1}{k+2} \right\rceil$$

Under the assumption that we have a source node that is infinitely backlogged, it is simple to prove that the maximum throughput for a string topology of n nodes is given by

$$\tau^* = \begin{cases} \frac{1}{n-1}\Gamma, & \text{for } 2 \le n \le k+2 \\ \frac{1}{k+2}\Gamma, & \text{for } n \ge k+3 \end{cases}$$

and this is achieved under perfect scheduling with all links having the same scheduling ratio given by

$$f_l = \begin{cases} \frac{1}{n-1}, & \text{for } 2 \le n \le k+2 \\ \frac{1}{k+2}, & \text{for } n \ge k+3 \end{cases}$$

Since TCP is a window based protocol, it is useful to obtain the throughput bound as a function of the window size W. To do so, we make the following assumptions:

a) The window size is held constant at W for the entire session. This would be true provided there are no packet loss events during the session, in which case the TCP window size reaches a maximum value W_{max} and remains constant thereafter.

b) The TCP-ACK packets do not flow through the network. This can never be true. However, if network coding efficiency $\gamma = 1$, we can the model the network traffic as only TCP-DATA packets flowing forward, and the TCP-ACK packets travelling backward "for free," without contending.

Having made the above assumptions, using Lemma 3.1, we arrive at the following result [8].

Theorem 3.1: The throughput for a window based transport layer protocol that always keeps W packets in the network has an upper bound given by

$$\tau^* = \begin{cases} \frac{1}{n-1}\Gamma, & \text{for } 2 \leq n \leq k+2 \\ \frac{W}{n-1}\Gamma, & \text{for } n \geq k+3 \text{ and } W < \left\lceil \frac{n-1}{k+2} \right\rceil \\ \frac{1}{k+2}\Gamma, & \text{for } n \geq k+3 \text{ and } W \geq \left\lceil \frac{n-1}{k+2} \right\rceil \end{cases}$$

B. Bounds for Random Access Scheduling

Let us note that the above bounds were obtained assuming perfect scheduling of channel access. Such bounds can be expected to be loose when compared to the results from a simulation in which IEEE 802.11 DCF is used. We can obtain tighter bounds by using the analysis for CSMA multi-hop networks developed in [9], [10], and recently extended in [11]. We review a slightly modified version of the model in [9], focusing on link activity rather than node activity.

Let $\mathcal{L}$ denote the set of links in the multi-hop wireless network. At any time certain sets of links can be simultaneously activated. These are called *independent sets* [12]. Let $\mathcal{A}(\subset 2^{\mathcal{L}})$ be the collection of all independent sets; we denote the empty set of links by $\Phi(\in \mathcal{A})$. When an independent set $A \in \mathcal{A}$ is active, then certain links cannot be activated as these cannot *coexist* with the ongoing transmissions in A, in the sense that they will interfere with or will be interfered by transmissions in A. On the other hand, a link i that is independent of A (i.e., $A \cup \{i\}$ is also an independent set) can be activated. In order to obtain an upper bound on performance, let us assume that the CSMA/CA mechanism is *ideal* in that a link that is not independent of an active set of links (itself an independent set) will not attempt to transmit (i.e., there are no "hidden nodes"). Thus, for each link there are time periods when it is *blocked* which alternate with time periods during which it can attempt and transmit.

It is assumed in [9] that all links are permanently backlogged. When Link i is *unblocked*, it attempts after an exponentially distributed time with mean $\frac{1}{\beta_i}$. This models the backoff time before an attempt is made on the link. Then a packet is transmitted whose transmission time is exponentially distributed[1] with mean $\frac{1}{\mu_i}$. The propagation delay is assumed to be zero; hence, when a Link i attempts, all other links that can potentially interfere with Link i *instantly* get blocked. The probability that two links make an attempt at the same instant

[1]It is shown in [9] that this exponential assumption can be relaxed to include any Coxian distribution.

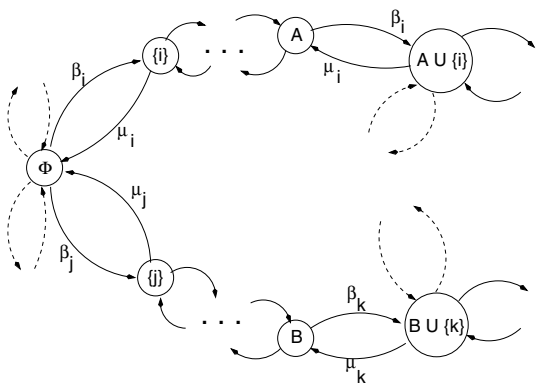

Fig. 5. A fragment of the transition rate diagram of the activation set CTMC, $S(t)$. Note that Link i is independent of the independent set A, hence Link i attempts at the rate β_i when the set of links in A are transmitting. Similarly, Link k is independent of the independent set B.

is zero. Thus, collisions are not modeled. It follows that, if the blocked periods of Link i were removed, then during the remaining time, Link i will be seen to execute an alternating renewal process, whose life-times are exponentially distributed with parameters β_i and μ_i. We note that β_i is akin to the effective attempt rate during back-off periods defined in [13]. However, since collisions are not modeled, the $\beta_i, i \in \mathcal{L}$, are free variables, over which we will optimise the throughput expression that we obtain.

With the above model, let $S(t) \in \mathcal{A}, t \geq 0$, denote the independent set that is transmitting at time t. It is easy to see that $S(t)$ is a Continuous Time Markov Chain (CTMC) with the transition rates shown in Figure 5. It can also be checked that this transition structure satisfies the Kolmogorov Criterion for reversibility (see [14]). It follows that the stationary probability distribution $\pi(A), A \in \mathcal{A}$, satisfies the following detailed balance equations (where $A \cup \{i\} \in \mathcal{A}$)

$$\pi(A)\beta_i = \pi(A \cup \{i\})\mu_i$$

It then follows that the stationary probability distribution has the form

$$\pi(A) = \left(\prod_{i \in A} \frac{\beta_i}{\mu_i} \right) \pi(\Phi) \tag{1}$$

where $\pi(\Phi)$ is determined from the normalisation equation

$$\sum_{A \in \mathcal{A}} \pi(A) = 1 \tag{2}$$

Then the rate at which packets are transmitted over Link i is given by

$$\theta_i = \left(\sum_{\{A \in \mathcal{A}: i \in A\}} \pi(A) \right) \mu_i \tag{3}$$

where the term in large brackets on the right hand side is the fraction of time that Link i is carrying packets.

See also [15], [16], where Tobagi and Brazio studied the underlying Markov process and derived conditions for general access protocols to have similar *product form* state probabilities. We use the theory developed above to obtain tighter bounds for the throughput of TCP when network coding is

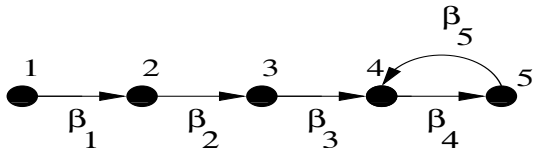

Fig. 6. A five node string topology showing the attempt rates for each node.

enabled. We illustrate our approach by an example.

Example: Let us consider the example topology in Figure 6. Let $\mathcal{L} = \{1, 2, 3, 4, 5\}$, where 5 denotes the reverse link from Node 5 to Node 4. Recalling our earlier terminology, we take the interference range k to be 2 here. This yields $\mathcal{A} = \{\Phi, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 5\}\}$. The attempt rates for TCP-DATA packets at nodes $1, 2, 3$ and 4 are represented by β_1, β_2, β_3 and β_4 respectively. The service rates are denoted by $\mu_1 = \mu_2 = \mu_3 = \mu_4 = \frac{1}{T_{TCP-DATA}} =: \mu_{data}$. If $\gamma = 1$, the TCP-ACK packets reach from node 4 to node 1 by being freely encoded with the forward travelling TCP-DATA packets. But TCP-ACK packets will have to be unicast from Node 5 to Node 4 therefore we represent this by the attempt rate β_5, and service rate $\mu_5 = \frac{1}{T_{TCP-ACK}} =: \mu_{ack}$. The assumption that $\gamma = 1$ is valid because we are interested in calculating a bound for the maximum throughput for the above network when network coding is enabled.

Further, let us denote, for $i \in \{1, 2, 3, 4, 5\}$,

$$\rho_i = \frac{\theta_i}{\mu_i}$$

and

$$x_i = \frac{\beta_i}{\mu_i}$$

Let us write $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$. Then using (1), (2), and (3), and defining

$$\Delta(\mathbf{x}) = 1 + x_1 + x_2 + x_3 + x_4 + x_5 + x_1 x_5$$

it is can be seen that

$$\begin{aligned} \rho_1(\mathbf{x}) &= \frac{x_1(1+x_5)}{\Delta(\mathbf{x})} \\ \rho_2(\mathbf{x}) &= \frac{x_2}{\Delta(\mathbf{x})} \\ \rho_3(\mathbf{x}) &= \frac{x_3}{\Delta(\mathbf{x})} \\ \rho_4(\mathbf{x}) &= \frac{x_4}{\Delta(\mathbf{x})} \\ \rho_5(\mathbf{x}) &= \frac{x_5(1+x_1)}{\Delta(\mathbf{x})} \end{aligned} \tag{4}$$

where we have shown that the normalised link throughputs depend on the link attempt rates $\mathbf{x}$.

Further, since each TCP DATA packet causes a TCP ACK packet to be sent backwards, we have

$$\theta_4 = \theta_5$$

Also we can write (see Section IV-A for the calculation of η)

$$\frac{\mu_4}{\mu_5} = \frac{T_{TCP-ACK}}{T_{TCP-DATA}} = 0.3027 =: \eta$$

Hence

$$\rho_5 = \eta \rho_4 \tag{5}$$

Let us now denote the end-to-end data packet throughput normalised to μ_{data} by ρ. The solution to the following problem will then serve as an upper bound to the end-to-end throughput.

$$\max \rho$$

subject to

$$\begin{aligned} \rho_1(\mathbf{x}) := \frac{x_1(1+x_5)}{\Delta(\mathbf{x})} &\geq \rho \\ \rho_2(\mathbf{x}) := \frac{x_2}{\Delta(\mathbf{x})} &\geq \rho \\ \rho_3(\mathbf{x}) := \frac{x_3}{\Delta(\mathbf{x})} &\geq \rho \\ \rho_4(\mathbf{x}) := \frac{x_4}{\Delta(\mathbf{x})} &\geq \rho \\ \rho &\geq 0 \\ x_i &\geq 0 \quad 1 \leq i \leq 5 \end{aligned}$$

From the form of the constraints for Links 1, 2, 3, and 4, we can see that at the optimum these constraints will be met with equality. For if, at the optimum, some constraint had a slack, say for Link j, then x_j could be reduced slightly without violating its own constraint, but while increasing the throughputs of the other three links, resulting in an overall increase in ρ, thus leading to a contradiction.

Thus, in order to determine the end-to-end throughput, we wish to solve

$$\rho = \rho_1 = \rho_2 = \rho_3 = \rho_4 = \rho_5 \frac{1}{\eta} \tag{6}$$

This yields the following equations

$$x_1(1+x_5) = x_2 = x_3 = x_4 = x_5(1+x_1)\frac{1}{\eta}$$

Equating the first and the last expressions yields

$$x_5 = \frac{\eta x_1}{1 + (1-\eta)x_1}$$

Using this to express $x_1(1+x_5)$ in terms of x_1, we obtain

$$x_2 = x_3 = x_4 = x_5(1+x_1)\frac{1}{\eta} = x_1(1+x_5) = \frac{x_1(1+x_1)}{1+(1-\eta)x_1}$$

Under the condition (6), we can now express $\Delta(\mathbf{x})$ in terms of x_1 as

$$\begin{aligned} \Delta(\mathbf{x}) &= 1 + x_1 + (3+\eta)\frac{x_1(1+x_1)}{1+(1-\eta)x_1} \\ &= \frac{(1+x_1)(1+4x_1)}{1+(1-\eta)x_1} \end{aligned}$$

Finally, the end-to-end normalised throughput is given by

$$\begin{aligned} \rho &= \frac{\frac{x_1(1+x_1)}{1+(1-\eta)x_1}}{\frac{(1+x_1)(1+4x_1)}{1+(1-\eta)x_1}} \\ &= \frac{x_1}{1+4x_1} \end{aligned} \tag{7}$$

We observe that if we want to maximise ρ (the normalised

end-to-end throughput), then $x_1 \to \infty$ and the limiting value of ρ is $\frac{1}{4}$, the same value that was obtained with a much simpler analysis done earlier. However, in practice the attempt rate at a node is upper bounded. Since $x_i = \frac{\beta_i}{\mu_{data}}, 1 \leq i \leq 4$, and we have $\frac{1}{\beta_i} \geq$ DIFS $+ \frac{\text{CW}_{\min}}{2}$ (see Figure 4), and $\frac{1}{\mu_i} =$ PHY + MAC + PAYLOAD + 2×SIFS + 2×MAC ACK, we obtain a bound on $x_i, 1 \leq i \leq 4$, given by

$$x_i \leq 4.108$$

where we take $\frac{\text{CW}_{\min}}{2} = \frac{31}{2}$ slots. Imposing the above condition, we find that ρ is maximised when $x_1 = 3.166; x_2 = x_3 = x_4 = 4.108$; yielding $\rho^* = \frac{x_1^*}{1+4x_1^*} = 0.2317$.

This completes the analysis of the 5 node example shown in Figure 6.

There is an observation that we make from the above analysis. The values of x_i at the optimum exhibit a bell shaped behavior as i varies from 1,2,...n. As a consequence, when we bound the values of x_i by $x_i \leq 4.108$, we always observe that $x_{\lceil \frac{n}{2} \rceil} = 4.108$. It also means that for the same desired packet rate θ_i, the attempt rate β_i is higher for nodes at the center compared with others, implying that there is a higher contention for the channel among the nodes located near the center of the string topology than those located towards the ends.

As the value of n increases beyond 5, the simultaneous equations relating ρ_i and x_i viz. (4) become highly coupled, making it difficult to solve analytically. The bounds for such values of n could be found by using iterative methods. One such method which was pointed out in [10] is as follows: After noting that at the optimum, $\rho_i = \rho = \rho_n \frac{1}{\eta}$ for $i \in \{1, 2, \ldots n-1\}$ and also $x_{\lceil \frac{n}{2} \rceil} = 4.108$, We can rewrite (4) in the form

$$x_i = \rho \, F_i(\mathbf{x}) \tag{8}$$

where $F_i(\mathbf{x})$ are continuous, monotonic functions in $x_i, i \in \{1, 2, \ldots n\}$. Beginning with some estimates of ρ and x_i, we can iteratively compute new estimates of ρ and x_i from the current estimates using the set of equations (8). The authors propose in [10] that such iterations will converge if a solution exists for the given set of equations (8). When we proceed as mentioned above, the iterations indeed converge and the throughput bounds that are obtained for different values of n are tabulated in Table III.

IV. Performance of TCP over IEEE 802.11 String Topology.

We obtain the performance of TCP over IEEE 802.11 using $ns2$ simulations.

A. Simulation Setup

Please refer to Figure 2 for the description of the topology. We set $d = 200m$, Transmission range to $250m$ and Interference range to $550m$. Thus we have $k = 2$. In Tables I and II, we tabulate the parameter specifications that we have used in our simulations.

TABLE I
$ns2$ SIMULATION PARAMETERS.

Parameter	Value
SIFS	$10\mu s$
DIFS	$50\mu s$
EIFS	$364\mu s$
Slot time (δ)	$20\mu s$
Basic Rate (Γ_b)	2 Mbps
Data Rate (Γ_d)	11 Mbps
(CW$_{min}$, CW$_{max}$)	(31,1023)
Retry Limit (Short, Long)	(7,4)

TABLE II
$ns2$ SIMULATION PARAMETERS.

Parameter	Size(Bytes)	Rate	Time(μs)
TCP-DATA	1460	Γ_d	1062
TCP-ACK	40	Γ_d	29.1
TCP-IP Header	40	Γ_d	29.1
RTS	44	Γ_b	176
CTS	38	Γ_b	152
MAC ACK	18	Γ_b	72
MAC Header	47	Γ_d	34.2
PHY	24	1 Mbps	192

Using the parameter values in Table II, we obtain

$$\begin{aligned} T_{TCP-DATA} &= \text{PHY} + \text{MAC} + \text{TCP-IP} + \text{TCP-DATA} \\ &\quad + 2 \times \text{SIFS} + 2 \times \text{MAC-ACK} \\ &= 1481.09 \; \mu s \end{aligned}$$

Proceeding similarly and using TCP-ACK instead of TCP-DATA in the above equation, we have

$$T_{TCP-ACK} = 448.36 \; \mu s$$

and hence

$$\eta = \frac{T_{TCP-ACK}}{T_{TCP-DATA}} = 0.3027$$

The TCP throughput at the network layer is given by

$$\tau^* = \theta^* \times (\text{TCP-DATA} + \text{TCP-IP})$$

where

$$\theta^* = \rho^* \times \mu_{data}$$

Using above equations, the throughput bounds are obtained as

$$\tau^* = \rho^* \times 8.102 \text{ Mbps}$$

and are tabulated in Table III, and plotted in Figure 15.

B. TCP over IEEE 802.11

We perform $ns2$ simulation of TCP over a 10 node string topology with IEEE 802.11 as the MAC and obtain the performance. The simulations are run for 100s and the TCP

TABLE III
THROUGHPUT BOUNDS FOR THE STRING TOPOLOGY.

Nodes n	ρ_{max}	τ^*[kbps]
1	1.000	8102
2	0.647	5242
3	0.393	3184
4	0.282	2285
5	0.232	1880
6	0.212	1718
7	0.197	1596
8	0.188	1523
9	0.188	1523
10	0.188	1523
∞	0.188	1523

version that we use is TCP/Newreno. We study the variation of TCP throughput as a function of TCP maximum congestion window size W_{max}.

The throughput obtained is plotted in Figure 7. In Section III, we have obtained the bounds assuming that all the nodes are permanently backlogged. In order to obtain the throughput bound as a function of the window size W, we will have to relax such an assumption which renders the problem intractable. For $W = 1$, the throughput bound can be easily calculated and assuming that saturation is attained at $W = 3$ (As was in Theorem 3.1 for $n = 10$ and $k = 2$), we obtain a throughput bound shown in Figure 7. The throughput bound for $W = 2$ is suitably interpolated. We see that the throughput obtained is much lower than its corresponding bound especially as W_{max} becomes large. Motivated by the potential gains that network coding can offer, we enable network coding for the above topology. The model that we used for network coding was described in Section II-C. The coding efficiency γ was around 50% and the throughput obtained is again plotted in Figure 7.

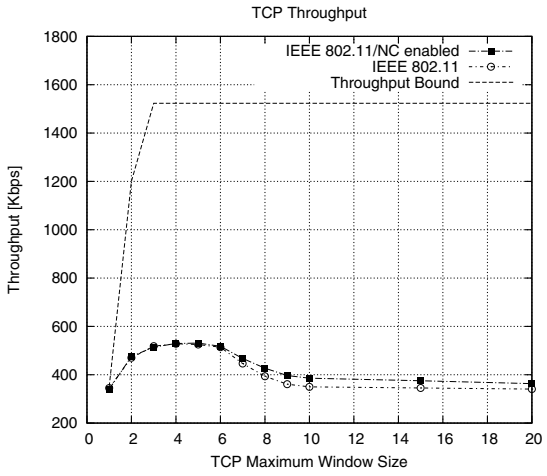

Fig. 7. TCP throughput over IEEE 802.11 for 10 nodes, plotted vs. the maximum window size, with and without network coding.

It comes as a surprise that enabling network coding did not provide any appreciable improvement in the throughput. To understand this unexpected behavior, let us monitor the TCP congestion window.

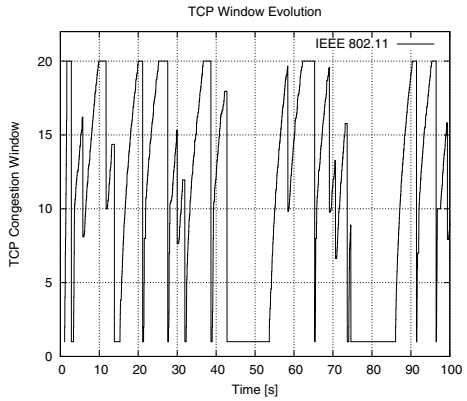

Fig. 8. TCP Window evolution for $n = 10$ and $W = 20$ over IEEE 802.11, without network coding.

In Figure 8, we plot the TCP Window as it evolves over time. We see that there are many instances at which TCP time-out occurs which is potentially because of packet drop events which occur as the number of retries for the transmission of a packet reach the retry limits. We might expect that it is the contention between the forward flowing TCP-DATA packets and the backward flowing TCP-ACK packets that is causing such packet-drops. If the above explanation is true, we would expect the window behavior to improve if network coding is enabled, as it reduces the number of TCP-ACK packets that contend.

We plot the TCP Window evolution when network coding is enabled in Figure 9. We find similar window behavior as with network coding disabled! Again there are frequent TCP timeouts.

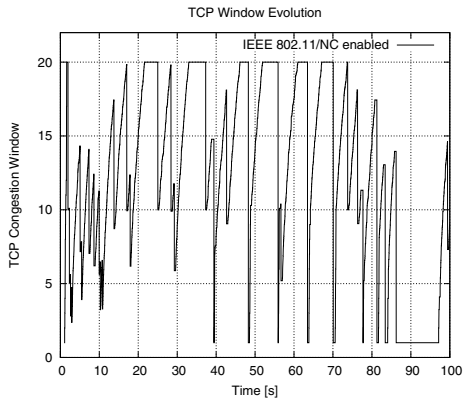

Fig. 9. TCP Window evolution with network coding. $n = 10$ and $W = 20$.

To further deepen our understanding, we simulate a hypothetical scenario. If network coding was perfectly operational, we expect no TCP-ACKs to contend for the channel. They can be modeled as traveling back "freely" as TCP-DATA packets move forward. To model the situation $\gamma = 1$, we simulated

a paradigm wherein the TCP-ACK packets reach the source directly from the destination without traveling through the network. Once TCP-DATA packets reach the destination, the corresponding TCP-ACK packets are copied directly into the source's receive buffer. We have performed such a simulation with a patch that we developed for $ns2$. We have also obtained the TCP Window evolution for such a scenario and plotted it in Figure 10.

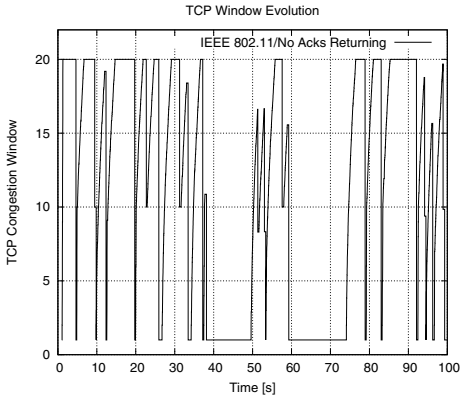

Fig. 10. TCP Window evolution with no TCP-ACKs returning. $n = 10$ and $W = 20$.

Once again, we find similar TCP congestion window behaviour. From the above simulations, we understand the basis for such behavior of throughput that we obtained in Figure 7. Initially when the maximum TCP Congestion window W_{max} is small, there is very little contention in the network and as W_{max} increases, the advantage due to spatial reuse increases and hence the throughput increases. But as W_{max} increases further, it results in increased contention which leads to more collisions and packet drop events, decreasing the throughput because of TCP timeouts. But why does the throughput remain the same when network coding is enabled? As indicated by Figure 10, even if we achieve coding efficiency $\gamma = 1$ i.e, even if we have a single flow of TCP-DATA packets, we find no improvement in the throughput. It points to the basic problem of coordination among the nodes. Even if there is a single flow of TCP-DATA packets, the forward traveling packets contend with each other for the channel which results in the packet drops that we see in Figure 9 or 10. Since this contention is among packets belonging to the same flow we can call this "self-contention". Unless we solve the problem of self-contention, any other performance improvement technique would remain inefficient as was shown above in the context of network coding.

V. A Distributed Scheme

A. An Adaptive Backoff

Having observed the problem, we now seek to find a solution. Our approach is similar to the one in [8], but we propose a distributed and a more general modification. In principle, we would like to achieve the flow scenario shown in Figure 11 which achieves the maximum throughput.

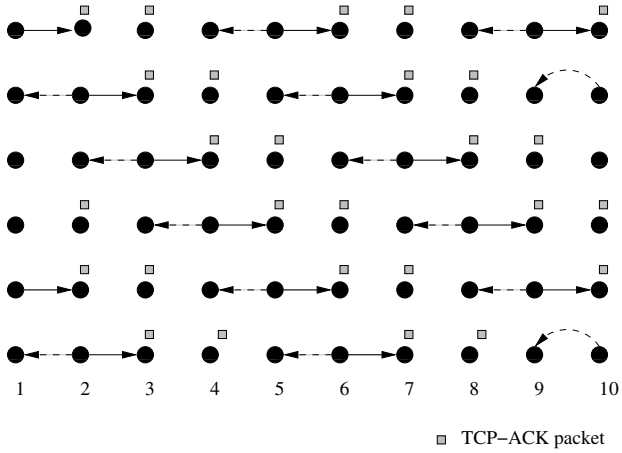

Fig. 11. An ideal flow scenario with perfect scheduling. $n = 10$ and $k = 2$.

As we can see in Figure 11, every link has a scheduling ratio given by $f_l^* = \frac{1}{k+2} = \frac{1}{4}$. More importantly, after a node transmits a packet, it doesn't transmit for the next 3 instants i.e, it defers its transmission for $3 \times T_{TCP-DATA} = (k+1) \times T_{TCP-DATA}$ and after $3 \times T_{TCP-DATA}$ from the earlier transmission, it resumes its transmission. This is the basic idea that we seek to implement - "After transmitting a packet, a node has to defer its further transmissions until the packet that it transmitted has crossed its and its receiver's $k-$hop interference range." By doing so, the node will not interfere with the transmission of the packet that it sent just a moment ago. We make the node defer its transmission by giving it a lower priority during the contention for the channel i.e, by making its contention window CW large. Therefore, as soon as a node transmits a packet, it sets its CW large (512) for a duration of $(k+1) \times T_{TCP-DATA}$. Let $T_p = (k+1) \times T_{TCP-DATA}$. After T_p s, the node again shrinks back its contention window to gain the channel for another transmission. Practically, the above proposition can be realised by using a timer at each node. After transmitting a TCP-DATA packet, the transmitting node sets a timer to expire after T_p s. Every node adjusts its own contention window CW based on its own timer's state. If the timer is on, use $CW = 512$ otherwise, use $CW = 32$.

The above implementation seems fine. However, for the above implementation to work, each node has to know its own T_p. Note that in an ad-hoc network, k is different for each node, and hence, each node has different T_p. Determining the values of k for each node using distributed algorithms is a tough problem by itself. We address this problem of determining the interference range by estimating the values using feedback from packet-drop events as follows. Let us assume, for the above topology, that node 1 has its estimate of k as 1 instead of 2. If nodes implement the above modification, node 1 sets CW to 512 after the transmission of the first packet but shrinks back its CW to 32 after $2 \times T_p$ instead of $3 \times T_p$ and hence potentially gains the channel and collides with the packet that it sent earlier at node 4 as shown in Figure 12. However, if node 1 still had its CW large after $2 \times T_p$, the probability of collision would be far less.

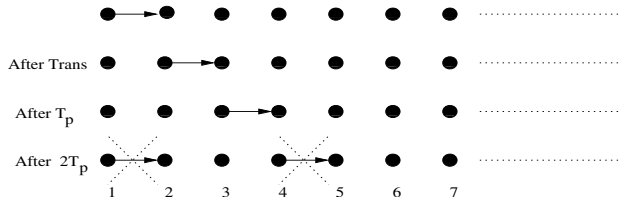

Fig. 12. Forward moving packets contend with each other if the amount of time for which an upstream node defers is too short. Here $n = 10$ and $k = 2$.

When such collisions happen frequently for node 1's transmission (because of its incorrect estimate of k), there would be packet drop events because of retry limits. We use such instances of packet drops as a feedback to improve node 1's estimate of k. When node 1's estimate approaches the actual value of k, the incidence of packet drop events at that node $\rightarrow 0$. Since the minimum value of k is 1, each node begins its estimate of k as 1 and if a node drops a packet, it will increase its estimate of k by 1 (since k is an integer, the next best value is $k + 1$). When implemented, this feedback works well and we find that soon, nodes gain a very good estimate of their values of k. Our proposed modification is summarised in the algorithm which every node executes as stated below.

Algorithm 1 Proposed Modification to IEEE 802.11

if TCP-DATA is sent **then**
  Set a timer which resets after T_p s
end if
if To send TCP-DATA/TCP-ACK **then**
  if The timer is set **then**
    $CW = 512$
  else
    $CW = 32$
  end if
end if
if A packet is dropped **then**
  $k = (k + 1)$
end if

In the above implementation, if a node overestimates its k by $k+1$ or $k+2$, it does not affect the performance significantly because if the neighboring nodes' estimate of k is correct, it still sees no collision. Since we are already trying to schedule the links in some sense, we disable the RTS/CTS mechanism as its use only has the impact of reducing the throughput by added overheads.

B. Simulation Results

1) Straight-line string topology: In this section, we provide simulation results from an implement of the proposed algorithm in $ns2$. We consider the case $k = 2$. At first, we plot the TCP congestion window evolution as shown in Figure 13. As we can see, we succeeded in avoiding TCP-timeouts by reducing packet-drop events as a result of effective scheduling.

We now proceed to plot TCP throughput for the string topology with and without enabling network coding versus the maximum congestion window size W_{max}, for $n = 10$ in

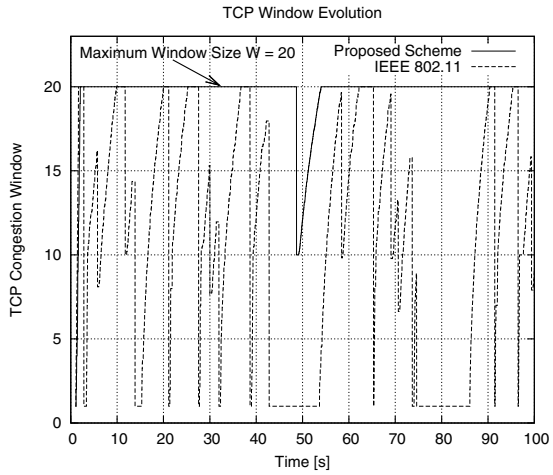

Fig. 13. For the proposed scheme, TCP Congestion Window remains almost stable at $W = W_{max} = 20$.

Figure 14. We find that for larger values of maximum window size W_{max}, the TCP throughput does not drop but remains steady with an improvement of 140% for $n = 10$. Moreover, the gain only due to network coding can also be noticed. We observe that saturation in throughputs is attained for window sizes of 10 or more. And hence, the assumption of infinite backlogging is not absolutely required and TCP window sizes of 10 or more will suffice for the analysis.

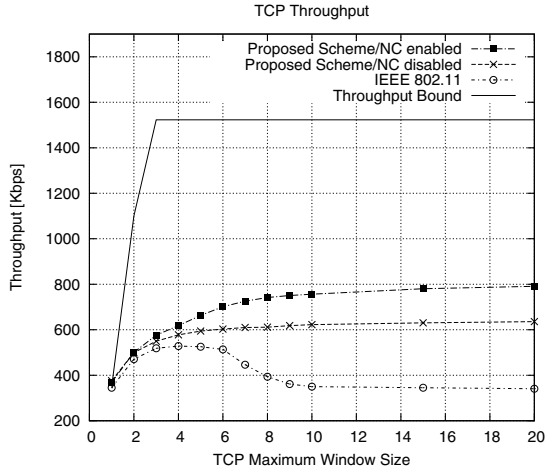

Fig. 14. TCP throughput vs. the maximum window size with the standard MAC, the new MAC without network coding, and the new MAC with network coding. $n = 10$.

When we perform simulations, coding efficiency γ was found to be around 50%. Plotted in Figure 15 is the steady-state throughput (when TCP window size is unrestricted i.e, for large values of W_{max}) as a function of the number of nodes n in the string topology. As we see, the proposed algorithm achieves remarkably higher throughputs with an overall improvement of over 100%.

2) Asymmetric string topology: We also perform simulations on an asymmetric string topology in which different nodes have different values of k as shown in Figure 16.

We plot TCP throughput as a function of W_{max} with and

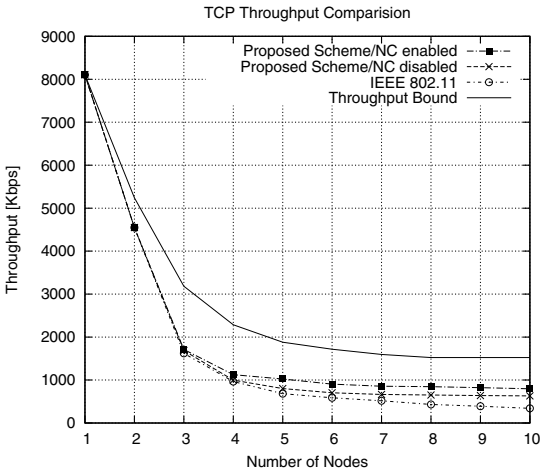

Fig. 15. We can notice that after modification of the MAC, TCP throughput improves. The source of the throughput bound here is Table III.

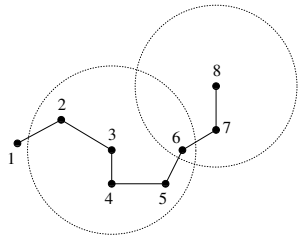

Fig. 16. A string topology in which different nodes have different values of k. Here node 3 has $k = 4$ where as node 8 has $k = 2$.

without enabling network coding for the asymmetric topology in Figure 16. The results are shown in Figure 17. Once again we find the gain only due to network coding which we could not observe without the modification of the MAC.

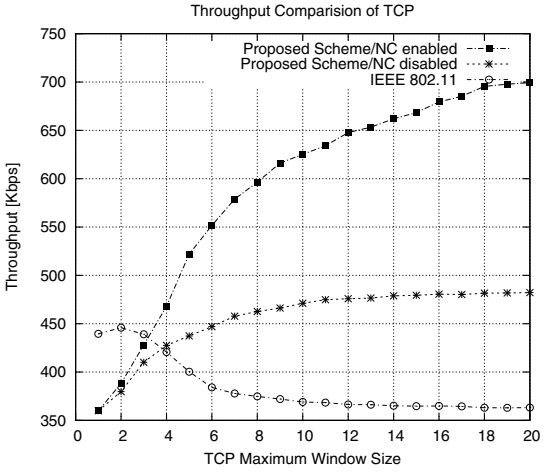

Fig. 17. The proposed modification gives a throughput improvement of around 94% even for an asymmetric string topology.

The plots in Figure 17 also show that IEEE 802.11 performs better compared with the proposed scheme for lower values of TCP maximum window size which implies that there is a possibility of fine tuning the proposed modification for much better performance than what is observed, especially for smaller window sizes.

VI. Summary

In this paper, we dealt with the inefficiency of using IEEE 802.11 as the MAC layer protocol for wireless multi-hop networks. Through various simulations, we have understood that unless we modify the channel access mechanism, we cannot achieve an appreciable gain in throughput due to network coding. We also use a simple model of random access in multi-hop networks in order to obtain throughput bounds. Drawing inferences from the above, we proposed an effective modification to IEEE 802.11 which solves the problem of self-contention in a distributed fashion without any added overheads. Thus, we could realise the potential of network coding, only after the modification of the MAC.

ACKNOWLEDGMENT

We thank D. Manjunath for his guidance on ns simulations.

References

[1] R.Alhswede, N.Cai, S.R.Li, and R.W.Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, 2000.
[2] S.R.Li, R.W Yeung, and N.Cai, "Liner Network Coding," *IEEE Transactions on Information Theory*, 2003.
[3] R.Koetter and M.Medard, "An algebraic approach to Network Coding," *IEEE/ACM Transactions on Network Coding*, 2003.
[4] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Medard, and Jon Crowcroft, "XORs in the Air: Practical Wireless Network Coding," in *ACM SIGCOMM*, 2006.
[5] Mario Gerla, Rajive Bagrodia, Lixia Zhang, Ken Tang, and Lan Wang, "TCP over wireless multi-hop protocols: Simulation and experiments.," in *IEEE ICC*, 1999.
[6] Mario Gerla, Ken Tang, and Rajive Bagrodia, "TCP performance in wireless multi-hop networks," in *IEEE WMCSA*, 1999.
[7] Shugong Xu and Tarek Saadawi, "Revealing the problems with 802.11 medium access control protocol in mult-hop wireless ad hoc networks," in *Elsevier Computer Networks*, 2002.
[8] Mathivanan Prabakaran, Arun Mahasenan, and Anurag kumar, "Analysis and Enhancement of TCP Performance over an IEEE 802.11 Multihop Wireless Network: single session case," in *IEEE ICPWC*, 2005.
[9] Robert R. Boorstyn, Aaron Kershenbaum, Basil Maglaris, and Veli Sahin, "Throughput Analysis in Multihop CSMA Packet Radio Networks," *IEEE Transactions on Communications*, vol. COM-35, no. 3, March 1987.
[10] Aaron Kershenbaum, Robert R. Boorstyn, and Mon song chen, "An Algorithm for Evaluation of Throughput in Multihop Packet Radio Networks with Complex Topologies," *IEEE Journal on Selected Areas in Communications*, vol. SAC-5, no. 6, July 1987.
[11] Michele Garetto, Theodoros Salonidis, and Edward W.knightly, "Modeling Per-flow Throughput and Capturing Starvation in CSMA Multi-hop Wireless Networks," *IEEE/ACM Transactions on Networking*, August 2008.
[12] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.
[13] Anurag Kumar, Eitan Altman, Daniele Miorandi, and Munish Goyal, "New insights from a fixed point analysis of single cell IEEE 802.11 wireless LANs," in *Proceedings IEEE Infocom*, 2005, Also Technical Report No. RR-5218, INRIA, Sophia-Antipolis, France, June 2004, and to appear in IEEE Transactions on Networking 2007.
[14] F.P. Kelly, *Reversibility and Stochastic Networks*, John Wiley, 1979.
[15] J.M.Brazio and F.A.Tobagi, "Theoretical aspects in throughput analysis of multihop packet radio networks," in *ICC*, June 1984.
[16] F.A.Tobagi and J.M.Brazio, "Throughput analysis of multihop packet radio networks under various channel access schemes," in *INFOCOM'83*, April 1983.