On the Performance of TCP over Throughput-Optimal CSMA

Wei Chen*, Yue Wang†, Minghua Chen* and Soung Chang Liew*
*Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, China
Email: {cw008, minghua, soung}@ie.cuhk.edu.hk
†School of Information, Central University of Finance and Economics, Beijing, China
Email: yue.wang@cufe.edu.cn

***Abstract*—An interesting distributed throughput-optimal CSMA MAC protocol, called adaptive CSMA, was proposed recently to schedule any strictly feasible rates inside the capacity region. Of particular interest is the fact that the adaptive CSMA can achieve a system utility arbitrarily close to that is achievable under a central scheduler. However, a specially designed transport-layer rate controller is needed for this result. An outstanding question is whether TCP Reno (one of the most mature versions of TCP) is compatible with adaptive CSMA and can achieve the same result. The answer to this question will determine how close to practical deployment adaptive CSMA is. Our answer is yes and no. First, we observe that running TCP Reno directly over adaptive CSMA results in severe starvation problems. Effectively, its performance is no better than that of TCP Reno over legacy CSMA (IEEE 802.11), and the potentials of adaptive CSMA cannot be realized. We then propose a multi-connection TCP solution with active queue management and prove that it can work with adaptive CSMA to achieve optimal utility. NS-2 simulations demonstrate that our solution can alleviate starvation and achieve fair and efficient rate allocation. We remark that multi-connection TCP can be implemented at either application or transport layer. Application-layer implementation requires no kernel modification, making the solution readily deployable in networks running adaptive CSMA. Our results show that adaptive CSMA can work well with only light-weight TCP modifications, bringing it a step closer to practicality.**

I. Introduction

The Carrier Sense Multiple Access (CSMA) protocol is widely used in local-area networks, including Wi-Fi. As the deployment of Wi-Fi spreads, it is now common to find multiple co-located Wi-Fi networks with partially overlapping coverage. In such networks, each link can only sense a subset of other links, and characterizing the link throughput achievable by CSMA is challenging.

Refs. [1] and [2] show that the saturated link throughput achieved by CSMA can be studied via a time-reversible Markov chain and can be expressed in a product form. Further, authors in [2] showed that the product form results are insensitive to the packet length and back-off time distributions given the ratio of their means.

Based on these results on the CSMA link throughput, Jiang and Walrand [3], and later others [4]–[7], present a distributed adaptive CSMA (A-CSMA) scheme that can achieve desired link throughput according to some system utility objective. It is shown that the distributed A-CSMA scheme can achieve almost the entire capacity region that can be achieved under a central scheduler. Given its attractiveness [8], an outstanding question is how close A-CSMA is to practical deployment. As originally studied in [3], specially designed transport-layer rate controllers were required to interwork with A-CSMA.

In view of the large installed base of TCP Reno, a question is whether TCP Reno over A-CSMA works well. We answer this question in this paper and make the following contributions:

- We observe that running TCP Reno directly over A-CSMA results in severe starvation problems. Effectively, its performance is no better than that of TCP Reno over legacy CSMA (IEEE 802.11) [1] [2] [9] [10]. Our analysis indicates that the positive feedback loop of TCP Reno and A-CSMA makes the system swing towards an unfair rate allocation that starves some links.
- We propose a multi-connection TCP Reno solution with AQM to inter-work with A-CSMA in a compatible manner. Although multi-connection TCP has been explored before in the context of video transmission over cellular and wired networks [11] [12], this paper applies it to solve starvation problems in TCP Reno over A-CSMA. Our solution is provably optimal under the Network Utility Maximization (NUM) framework. Specifically, by adjusting the number of TCP Reno connections for each session, any system utility can be realized. NS-2 simulations demonstrate that this solution can solve starvation problems and achieve fair and efficient rate allocation. The scheme is applicable to single-hop wireless networks and multihop networks with wired and wireless links.

The rest of the paper is organized as follows. We present related work in Section II, and the network model in Section III. We illustrate the starvation of TCP Reno A-CSMA in Section IV. We then propose our solution in the Section V, and evaluate its performance through NS-2 simulations in Section VI. Finally, Section VII concludes our paper.

II. Related Work

Rate control (i.e., congestion control) is important for data transmission over both wired and wireless networks. One of the most mature rate control schemes is TCP Reno. To put our study in context, we classify the possible solutions into four classes, as shown in Table I, where we refer to CSMA/CA scheme used in today's Wi-Fi as legacy CSMA (L-CSMA).

Class 1 solutions use TCP Reno over L-CSMA. There have been extensive work studying the fairness problem of this scheme [1] [2]. They reveal that TCP Reno over L-CSMA results in severe starvation problem. The main reason is that L-CSMA is not an efficient scheduling algorithm and it can only schedule a fraction of the capacity region.

TABLE I
Rate control and Scheduling Solution Classes

	L-CSMA	modified-MAC
TCP Reno	class 1	class 2
modified TCP	class 3	class 4

Class 3 solutions, for instance [9] [10], do not make any changes to the widely used IEEE802.11 MAC and they limit the design space to within the scope of transport layer. In [9], the authors propose an AIMD-based rate control protocol called WCP Wireless Control Protocol (WCP) to address the efficient and fair rate allocation over L-CSMA networks. WCP halves the rates of other flows if they are within interference range of the congested link. Hence, queues of other fast transmitting links are driven to empty, giving the congested link opportunity to transmit. However, WCP requires message passing among the interfered links. Meanwhile, each link must monitor the round-trip time (RTT) of every flow that traverses it. EWCCP [10] is designed to be proportionally fair by periodically broadcasting the average queue information to coordinate among interference links. These periodical broadcasted message consumes the precious wireless bandwidth.

There have been many efforts belonging to class 4. Cross-layer designs based on new TCP and new MAC are adopted [3]–[6], [13], [14]. Cross-layer designs can improve the overall system performance and make interaction between layers more transparent, but raise the hurdle to practical deployment.

Our solution belongs to class 2. First, it is a layered approach rather than a cross-layer one, reducing the implementation difficulty. Second, since TCP is widely used in the Internet, it is preferable to avoid modifying it.

Our solution is based on multiple connection TCP Reno. There has been work on using multiple connections to transfer data. For example, multiple TFRC connections for streaming video [15] and multiple TCP Reno connections for data and multimedia streaming [11] [12] have been discussed. These works focus on wired networks and wireless cellular networks. In contrast, our work focuses on CSMA networks and therefore solves a different set of challenges, including the spatial wireless inference that was not considered in [15] [11] [12].

Our solution utilizes active queue management (AQM) in the MAC. AQM techniques have been proposed to both alleviate congestion control problems [16] as well as to implement the provisioning of quality of service [17]. The classical example of an AQM policy is RED [16]. In our study, we apply a special RED policy: the packet dropping probability is proportional to queue length. In our work, by applying this AQM, our scheme converges to the optimal utility.

III. Problem Settings

In this section, we provide problem settings and necessary background on A-CSMA and TCP Reno. We model a CSMA wireless network by a graph $G = (V, E)$, where V is the set of nodes (i.e., wireless transmitters and receivers) and E is the set of links. We consider the scenario where multiple users communicate over the wireless network. Each user is associated with a flow over a pre-defined route between a source and a destination. The key notations used in this paper are listed in Table II. We use bold symbols to denote vectors and matrices of these quantities, e.g., $\boldsymbol{x} = [x_s, s \in S]$.

TABLE II
Key Notation

Notation	Definition
V	the set of nodes
E	the set of links
$\mathcal{S}$	the set of source-destination pairs
$\mathcal{I}$	the set of independent sets
x_s	rate of flow s
T_s	round trip time of flow s
n_s	number of connection on flow s
y_l	average rate of link l
τ_i	time portion of independent set i
r_l	transmission aggressiveness (TA) of link l
p_l	price of link l
$l \in i$	link l belongs to independent set i
$l \in s$	flow s passes through link l

In CSMA wireless networks, a transmitter applies a carrier sensing mechanism to prevent collisions. In particular, a transmitter listens for an RF carrier before sending its data. If a carrier is sensed, the transmitter defers its transmission attempt, enters a countdown process in which it waits for a random amount of time before making another transmission attempt. As such, two links are allowed to transmit simultaneously only if the simultaneous transmissions are under the CSMA carrier sensing operations.

To bring out the main essence of the problem, we make two assumptions on the problem formulation. We discuss them in the following.

- Perfect carrier sensing. If the carrier sensing range is not properly set, simultaneous transmissions allowed by the carrier sensing operations may still interfere with each other, resulting in the hidden-node problem [18]. Hidden-node-free CSMA network designs are attractive not only because they avoid throughput degradation due to hidden-node collision but also because the throughput analysis is tractable [1] [2]. In particular, the stationary distribution of the system state is a product form. This observation is fundamental to the design of adaptive CSMA [3]. As studied in [19], any CSMA wireless network can always be made to be hidden-node-free by setting the carrier sensing range properly.
- Homogeneous links, i.e., the data rate of each wireless link is identical. This assumption can be relaxed and we can incorporate the data rate factor into our model and algorithms, by multiplying τ_i by the data rate of link l in Eq. (1).

A. Capacity Region of CSMA Wireless Networks

For CSMA wireless networks, we use the conflict graph approach in [20] to model the relationship of wireless link interference. We denote the conflict graph of G by $G_c = (V_c, E_c)$.

The vertices of G_c correspond to links E in the communication graph G, i.e., $V_c = E$. An edge between two vertices means that the two corresponding links can carrier sensing each other. An independent set corresponds to a feasible schedule which is a set of links in the communication graph that can be active simultaneously.

Let $\mathcal{I}$ denote the set of all independent sets, τ_i denote the fraction of time the system is in schedule $i \in \mathcal{I}$. Assuming that all links have unit capacity, the capacity region of link rate is given by

$$\Pi = \{\boldsymbol{y} \geq 0 | \forall l,\ y_l \leq \sum_{i:l\in i} \tau_i, \sum_{i\in\mathcal{I}} \tau_i = 1, \boldsymbol{\tau} \geq 0\}. \tag{1}$$

Note that finding all the independent sets is NP-hard [21]. Consequently, characterizing the capacity region in Eq. (1) is thus very challenging, let alone attaining it in practice.

B. Throughput-optimality of A-CSMA

Refs. [22], [1] and [2] show that the link scheduling in CSMA networks can be studied via a Markov chain, in which the states are all independent sets. In this model, the transmitter of link l counts down an exponentially distributed random period of time with mean $\exp(-\beta r_l)$ before transmission, where both β and r_l are positive constant parameters that can be obtained by reverse engineering according to CSMA protocol. Larger β, smaller the mean count down time. When the link l starts a transmission, it keeps the channel for an exponentially distributed period of time with mean one[1]. This CSMA Markov chain converges to its product form stationary distribution, which is given by

$$p(i, \boldsymbol{r}) = \frac{\exp\left(\sum_{l\in i} \beta r_l\right)}{\sum_{i'\in\mathcal{I}} \exp\left(\sum_{l\in i'} \beta r_l\right)},\ \forall i \in \mathcal{I}, \tag{2}$$

where β is a positive constant. The average portion of time link l is active, denoted by y_l, achieved in steady state is given by

$$y_l(\boldsymbol{r}) = \sum_{i:l\in i} p(i, \boldsymbol{r}) = \frac{\sum_{i:l\in i} \exp\left(\sum_{l\in i} \beta r_l\right)}{\sum_{i'\in\mathcal{I}} \exp\left(\sum_{l\in i'} \beta r_l\right)},\ \ \forall l \in E. \tag{3}$$

As a result, with fixed β and $\boldsymbol{r}$, existing CSMA algorithm can only obtain a single point in the rate region Π in Eq. (1) [2], [22].

In the contrast, by adjusting r_l, the authors in [3] introduce a fully distributed adaptive CSMA (A-CSMA) algorithm that can achieve almost the entire capacity region Π as follows:

Algorithm A-CSMA.

$$\dot{r}_l = \alpha[a'_l - d'_l]^+_{r_l}, \tag{4}$$

where a'_l and d'_l are the empirical average arrival rate and service rate of link l, α is constant step size and notationally,

$$[g(z)]^+_z = \begin{cases} \max(g(z), 0), & \textit{if } z \leq 0 \\ g(z), & \textit{otherwise} \end{cases} \tag{5}$$

Theorem 1 ([3]). *Assume arrival rate vector $\boldsymbol{y}'$ is in the interior of capacity region Π. Then with Algorithm A-CSMA, $\boldsymbol{r}$ converges to $\boldsymbol{r}^*$, with probability one, where $\boldsymbol{r}^*$ satisfies that $y'_l \leq y_l(\boldsymbol{r}^*)$.*

Remark 1: it is not difficult to verify that r_l is proportional to its local queue length of link l. Therefore, A-CSMA scheduling algorithm can be implemented in a fully distributed way. In A-CSMA, before transmitting, any link l sets up a random exponentially distributed counter with mean equal to $\exp(-\beta r_l)$. When link l obtains the channel, it transmits a packet with length which is exponentially distributed with mean equal to one. Note A-CSMA differs from L-CSMA in that the link's countdown timer is a function of its queue length, as a result, A-CSMA gives priority to links with heavy loads. We remark that A-CSMA is simple to implement by slight modification to today's CSMA, and there have been efforts in doing so [8].

Remark 2: Theorem 1 shows that the product-form distribution $\boldsymbol{y}(\boldsymbol{r})$ achieved by A-CSMA is larger than any interior feasible rate vector. This means A-CSMA stabilizes the system for all feasible arrival rate vector, i.e., A-CSMA scheduling algorithm is throughput-optimal. Therefore, A-CSMA has superior performance as compared to L-CSMA.

For more details of A-CSMA, interested readers can refer to the work in [3]–[7]. Because its simplicity and throughput optimality, we think A-CSMA will be very popular soon. And it is important to study whether it works well with TCP.

C. TCP Reno Congestion Control Modeling

TCP Reno adjusts the flow rate by controlling the congestion window size W. During the congestion avoidance phase, TCP Reno increments its congestion window W by one every round trip time if no loss is observed. It halves its window whenever packet loss is detected.

In [23], the authors model this dynamic of the congestion avoidance phase of TCP Reno as the following dynamic equation:

$$\dot{x_s} = \frac{x_s^2}{2}\left(\frac{2}{T_s^2 x_s^2} - \sum_{l\in s} p_l\right) \tag{6}$$

where T_s is round-trip-time(RTT) of flow s, and p_l denotes the price (specifically, probability a packet will be dropped) of link l. When link l applies drop-tail queue,

$$p_l = \frac{\left(\sum_{s:l\in s} x_s - y_l\right)^+}{\sum_{s:l\in s} x_s}, \tag{7}$$

where y_l denotes average throughput of link l, and $[\cdot]^+$ denotes $\max(0, \cdot)$.

IV. TCP Reno over L-CSMA and A-CSMA

Fairness is one of the key problems that must be considered in designing any MAC, which allows contending links to share the wireless channel fairly. The random backoff algorithm of L-CSMA (IEEE 802.11 DCF) gives each host equal average count-down time to access the channel. This algorithm works fine in a symmetric environment where all hosts connected to a single access point. However, in asymmetric settings, it fails to achieve the fair channel accessing, even though a fair rate control protocol such as TCP can not solve this MAC layer fairness problem.

As stated in the previous section, A-CSMA scheduling algorithm is throughput optimal and can be easily implemented by slightly modifying the existing L-CSMA. Observing the large installed base of TCP Reno, a natural question to ask before actual deployment of A-CSMA network is how well TCP Reno can perform over A-CSMA networks.

[1]Without loss of generality, we normalize the mean of transmission time to one.

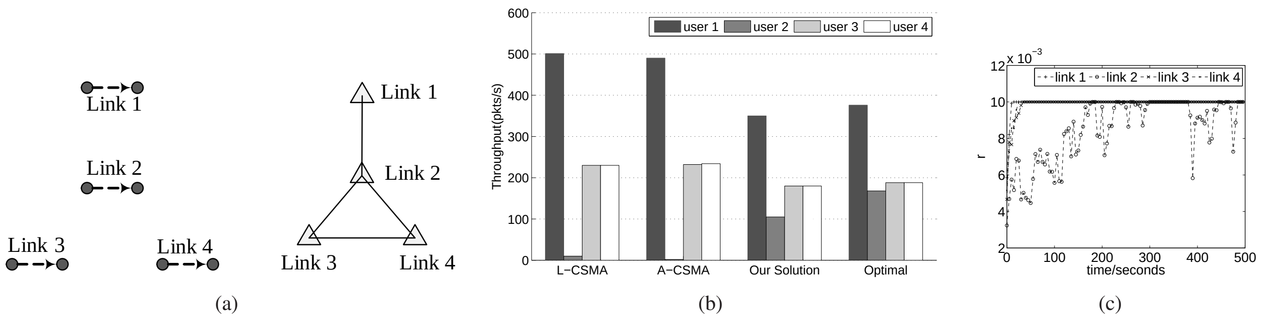


Fig. 1. (a) a wireless network of four links. Its conflict graph is shown on the right hand side. (b) throughput under 1) L-CSMA: NS-2 simulation of TCP Reno over L-CSMA, 2) A-CSMA: NS-2 simulation of TCP Reno over A-CSMA, 3) our solution: NS-2 simulation of multi-connection TCP Reno over A-CSMA with AQM, 4) optimal: optimal throughput computed from problem **MP**. (c) transmission aggressiveness under A-CSMA in NS-2 simulation.

In this section, we examine the performance of TCP Reno over L-CSMA and A-CSMA, respectively. We will see that they both suffer from starvation, but we remark that *the causes are fundamentally different*.

We carry out NS-2 simulation to demonstrate the starvation problem. The network topology is shown in Fig. 1(a). Each link carries a single TCP Reno connection. The simulation settings of L-CSMA are just as that of 802.11 DCF. The simulation settings of A-CSMA are as follows: 1) update step size and update interval for r are 0.05 and 2.0 respectively; 2) β: weight of entropy term is 800 (large β makes sure that the mean count down time $\exp(\beta r_l)$ is small; 3) data rate of wireless networks is 11Mbps; 4) TCP payload is 1000 bytes. To prevent the countdown from being too aggressive[2], we set the upper bound for $\boldsymbol{r}$ to be $r_{\max} = 0.01$. We remark that our simulation results hold as long as $\beta r_{\max}$ is bounded. To get a clear understanding, we only consider the forward link (the link that transmits TCP DATA) in the analysis by letting the backward link send back the TCP ACK with higher priority than forward link TCP DATA. To achieve this goal, we set the backward link with a fixed maximum transmission aggressiveness r_{max}. Consequently, most of the packets are buffered in the queue of the forward links.

Fig. 1(b) shows the TCP throughput achieved over A-CSMA and that over L-CSMA, respectively. In both cases, user 2 has much less throughput than other users and therefore starves. We explain *their fundamentally different causes* as follows.

A. TCP Reno Starves over L-CSMA

The starvation phenomenon of TCP Reno over L-CSMA is caused by the fundamental inefficiency of L-CSMA and can be intuitively explained as follows. When link 1 is transmitting, link 2 freezes its count down process because it is within the carrier sensing range of link 1. While link 2 is frozen, either link 3 or link 4 can transmit. This will continuously freeze link 2 after link 1 finishes its transmission, leaving link 2 very small chance to transmit.

Extensive work, such as [1], [2], [22], studied the fairness issue of L-CSMA via a Makov chain, with all links count down with equal mean time. For simplicity, we define $\rho = \exp(\beta r_l)$, where r_l are equal for each link. The throughput of link l achieved by L-CSMA is then given by

$$y_l = \frac{\sum_{i:l\in i}\left(\prod_{l\in i}\rho\right)}{\sum_{i'\in\mathcal{I}}\left(\prod_{l\in i'}\rho\right)},\ \forall l\in E. \tag{8}$$

In our example, the independent sets are $\emptyset$, $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{1,3\}$ and $\{1,4\}$, and the throughput of link 2 is as follows:

$$y_2 = \frac{\rho}{2\rho^2+4\rho+1}. \tag{9}$$

Plugging $\rho \approx 2.24$ [1] into Eq. (9), we get $y_2 \approx 0.11$, which is much smaller compared to the throughput of the other links. As such, the TCP Reno session running over link 2 starves. We remark that such starvation is caused by the fundamental inefficiency of L-CSMA and is not limited to TCP. In fact, any flows including those running over UDP also suffer from starvation.

B. TCP Reno Starves over A-CSMA

To avoid starvation, link 2 must count down with larger transmission aggressiveness than the other links, so as to compete more aggressively for the fair channel share against the other three links. This is the way that A-CSMA works.

However, the potential of A-CSMA cannot be realized by running TCP Reno directly over it and we also observe the starvation phenomenon. We explain this as follows. TCP Reno increases its rate more slowly when it experiences larger RTT, and vice versa. Initially, link 2 suffers from freezing its backoff countdown for a long time because of the transmissions of its neighbors. Consequently, TCP Reno over link 2 experiences larger RTT and increases its congestion window more slowly than those over the other links. Thus link 2's queue increases more slowly than those of the other links. Since A-CSMA sets the transmission aggressiveness to be proportional to the queue length, link 2 ends up having the smallest transmission aggressiveness. This explanation is verified by the simulation results on transmission aggressiveness in Fig. 1(c).

So the starvation phenomenon of TCP Reno over A-CSMA is caused by the positive feedback loop between TCP Reno and A-CSMA, making the system swing towards an unfair rate allocation that starves some links: Larger transmission aggressiveness leads to shorter RTT. Shorter RTT makes TCP Reno increase its rate faster, thus the larger congestion window. Larger congestion window leads to longer queues, thus larger transmission aggressiveness. This loop continues until the transmission aggressiveness of all the links reach the

[2]If transmission aggressiveness are too large, then $\exp(-\beta r_l)$ will be very small, which will result in zero countdown in digital system.

same upper limit $r_{\max}$ (when all queues reach the max length). With $r_l = r_{\max}$ for $l = 1, 2, 3, 4$, the performance of TCP Reno over A-CSMA falls back to that of TCP Reno over L-CSMA, and link 2 suffers starvation.

V. Proposed Solution

In this section, we propose to use a layered approach of *multi-connection TCP Reno over A-CSMA with AQM* as a practical solution to achieve fairness. This solution is provably optimal under NUM framework. Interestingly, it can be extended to wired network and multihop networks with wired and wireless links without any change. In this solution, we stick to today's TCP Reno instead of designing a new TCP for the following reasons. First, many Internet services (e.g. HTTP, FTP) rely on TCP Reno. Second, TCP Reno has been well tested in the Internet scale. Last, compatibility concern seems challenging and not well-understood yet [24]. Hence, a new TCP must be carefully designed to be compatible with today's TCP Reno.

A. Mathematical Formulation of Our Solution

We observe that TCP Reno interacts with A-CSMA through both RTT and packet loss rate in the previous section. First, any loss based TCP will interact with the queue-based A-CSMA through packet loss rate. For the interaction to behave properly, we always need to employ AQM to calibrate the mapping between the link queue length and the link loss rate. Second, it has been observed and discussed in the previous section that the reason of the flow starvation is that it suffers long RTT, which is caused by the long MAC access delay. To remove TCP bias on RTT, one way is to open multiple number of connections proportional to the RTT that TCP experiences. In this way, TCP Reno with large RTT that will open more connections to compensate for the small congestion window of each connection. As a result, there are more outstanding packets filling in the queue of the link. This makes A-CSMA allocate more airtime to the link, reducing its MAC accessing delay of the link and thus the RTT of the flow.

Inspired by the above observations, we propose to use multi-connection TCP Reno to remove the RTT interaction between TCP Reno and A-CSMA, and use AQM to calibrate their packet loss interaction.

In our solution, every user s monitors the round trip time T_s, and opens $n_s = kT_s$ (where k is a constant) TCP Reno connections to remove the RTT bias. We remark that now the user rate x_s is the aggregate rate of n_s TCP Reno connections. We present the overall scheme as the following dynamic system:

$$\begin{cases} n_s = kT_s, \ \forall s \in \mathcal{S}; & (10) \\ \dot{x}_s = \dfrac{x_s^2}{2n_s}\left(\dfrac{2n_s^2}{T_s^2 x_s^2} - \displaystyle\sum_{l\in s} r_l\right)^+_{x_s}, \ \forall s \in \mathcal{S}; & (11) \\ \dot{r}_l = \alpha\Big[\displaystyle\sum_{s:l\in s} x_s - \sum_{i:l\in i} \tau_i(\boldsymbol{r})\Big]^+_{r_l}, \ \forall l \in L, & (12) \end{cases}$$

where

$$\tau_i(\boldsymbol{r}) = \frac{\exp\left(\sum_{l\in i}\beta r_l\right)}{\sum_{i'\in\mathcal{I}}\exp\left(\sum_{l\in i'}\beta r_l\right)}, \ \forall i \in \mathcal{I}. \tag{13}$$

We remark that how T_s is updated is irrelevant to our solution as we always compensate for its impact by opening n_s connections. The multi-connection TCP Reno dynamic equation can be expressed in the form of Eq. (11) [3] [11] when the end-to-end price (e.g., packet loss rate) is $\sum_{l\in s} r_l$. This link price can be fed back by packet loss based technique like Active Queue Management (AQM). In this paper, we adopt an AQM policy in which each link drops or ECN-marks packet with probability equal to r_l. We remark that the independent sets distribution satisfies Eq. (13) as long as each link applies CSMA mechanism and backoff with mean equal to $\exp(-\beta r_l)$. The dynamic equation Eq. (12) are exact to A-CSMA algorithm introduced in Section III-B and r_l is proportional to queue length.

The above dynamic system achieves certain system utility. With the time-scale separation assumption[4], it is stable and converges to the unique equilibrium. We summarize the result as the following theorem.

Theorem 2. *Equilibrium of dynamic system in (10)-(12) solves the following Network Utility Maximization problem[5]:*

$$\textbf{MP:} \ \max_{\boldsymbol{x}\geq 0, \boldsymbol{\tau}\geq 0} \ \sum_s -\frac{1}{x_s} - \frac{1}{\beta}\sum_{i\in\mathcal{I}} \tau_i \log \tau_i$$

$$s.t. \quad \sum_{s:l\in s} x_s \leq \sum_{i:l\in i} \tau_i, \ \forall l \in E, \tag{17}$$

$$\sum_{i\in\mathcal{I}} \tau_i = 1, \tag{18}$$

where the system utility function is $U_s(x_s) = \sum_{s\in\mathcal{S}} -\frac{1}{x_s}$.

Remark: at the equilibrium, the utility function $-\frac{1}{x_s}$ guarantee an α-fairness among users with $\alpha = 2$ [25]. One of its implications is that *no user will starve at the optimal solution at the equilibrium*. This provides theoretical justification that our proposed solution will effectively address the TCP Reno over A-CSMA starvation problem. Note that we can set β large to eliminate the effect of the entropy term, which also results in small $\boldsymbol{r}$. The proof of Theorem 2) is given in [26].

B. Discussions on Implementation

Inspired by the observations from the dynamic systems in Eq. (10)-(12). We can solve the problem by running multi-

[3]Recall that the single connection TCP Reno congestion window dynamic equation is,

$$\dot{W}_s = \frac{1}{T_s} - \frac{W_s^2}{2T_s}\Big(\sum_{l\in s} p_l\Big), \tag{14}$$

where T_s is the round trip time and $\sum_{l\in s} p_l$ is the end-to-end packet loss rate experienced by the TCP Reno. If flow s opens n_s number of TCP connection, the flow rate x_s can be expressed as

$$x_s = \frac{n_s W_s}{T_s}. \tag{15}$$

From Eq.(14) and Eq. (15), we now have

$$\dot{x}_s = \frac{x_s^2}{2n_s}\Big(\frac{2n_s^2}{T_s^2 x_s^2} - \sum_{l\in s} p_l\Big). \tag{16}$$

[4]Without this assumption, the system turns into a stochastic dynamic system, where the measured rate of link l does not satisfy Eq. (8). Under small step size and update interval, the resulted system converges to a bounded neighborhood of the optimal solution with probability 1 [7].

[5]This formulation is similar to rate control over TDMA network except that there is an additional entropy term in the objective function. As shown in [3] [7], this leads to distributed implementation by using A-CSMA.

connection TCP Reno over A-CSMA with AQM. We stress multi-connection is important in our solution in that it removes the RTT interaction with A-CSMA. Without removing such RTT bias on RTT, TCP Reno over A-CSMA with AQM also results in starvation problem. Interested readers can refer to our technique report [26] for detailed results and analysis.

The multi-connection layer can be implemented in two ways, without modification to the transport layer. One method is to insert an intermediate layer between the application layer and transport layer, called Multi-connection API. It provides the universal API to the application layer, and functions to maintain kT_s TCP Reno connections from monitored RTT. This solution requires no modification to TCP/IP stack and is compatible with today's applications (like FTP, HTTP, etc.).

The other implementation method is to rewrite the applications. The application keeps monitoring RTT and opens multiple sockets. To obtain the RTT, we can simply set up a UDP connection to measure the RTT [11], [15].

We prefer the first implementation in which we insert an multi-connection API in between. This implementation does not require any modification to today's applications and hence simplifies programming. From the structure of the dynamic system in Eq. (10)-(12), it can be implemented in a layered manner as follows:

- In the MAC layer, we run A-CSMA to schedule the link transmissions. This is directly obtained from Eq. (12). A-CSMA maintains transmission aggressiveness vector $\boldsymbol{r}$ to be proportional to the lengths of link queues.
- We run AQM at each link that drops or marks packets with a probability proportional to $\boldsymbol{r}$, and thus proportional to the queue lengths. In this way, the prices of using individual links can be fed back to end users via packet losses or ECN marks. We remark that AQM is required as long as we want to run loss based TCP (e.g. Reno) over A-CSMA.
- In the transport layer, we perform rate control based on the sum of the prices of individual links, which is fed back in a form of packet losses or ECN marks. This result is directly obtained from Eq. (11). When loss ratio of link l is r_l, the total loss ratio that TCP Reno sees is $1 - \prod_{l\in s}(1 - r_l) \approx \sum_{l\in s} r_l$, when r_l is small.
- In the multi-connection layer, we maintain $n_s = kT_S$ connections to remove the RTT bias.

In practice, user s can only open an integer number of connections. By setting $n_s = \text{int}(kT_s)$, user s can react to the change in T_s in fine granularity if k is large, and in coarse granularity for small k. While large k is preferred in theory, in practice user s may need to maintain a large number of connections if k is too large. Large k also make the n_s adjustment more responsive to the changes in T_s, so it induces more overhead due to frequently opening and closing TCP Reno connections.

Packet loss ratio should be far less than one, otherwise it will waste considerable bandwidth for dropping a large number of packets. Besides, the total loss ratio $1 - \prod_{l\in s}(1 - r_l) \approx \sum_{l\in s} r_l$ does not hold when r_l is large. By using a small k, we can scale r_l so as to make it sufficiently small.

C. Discussions on Advanced Features

We discuss the following two questions in this subsection: 1) By using multi-connection TCP Reno, can we achieve arbitrary utility? 2) Is our solution applicable to wired network as well as multihop networks with wired and wireless links?

1) Achieve Arbitrary Utility: For the first question, our answer is yes. Let $U_s(x_s)$ be an arbitrary utility function of problem **MP**. Consider our solution in Eq. (10)-(12) but replacing the updating equation of n_s as follows: $n_s = k\sqrt{\frac{U_s'(x_s)T_s^2x_s^2}{2}}$. It is not difficult to verify that the equilibrium of the modified solution solves problem **MP** with the specified utility function $U_s(x_s)$.

In particular, our proposed solution in Section V-A can be understood as a special case of the above approach with the specified utility function being $U_s(x_s) = -\frac{1}{x_s}$.

2) Extension to Networks with Both Wired and Wireless links: Our answer to the second question is also yes. We consider a multihop network composed of wired and wireless links. Let l denote the wireless link and w denote the wired link. Our dynamic system to solve the extension problem is as follows:

$$\begin{cases} n_s = kT_s,\ \forall s \in \mathcal{S}; & (19)\\ \dot{x}_s = \dfrac{x_s^2}{2n_s}\left(\dfrac{2n_s^2}{T_s^2x_s^2} - \sum_{l\in s} r_l - \sum_{w\in s} p_w\right)_{x_s}^{+},\ \forall s \in \mathcal{S}; & (20)\\ \dot{r}_l = \alpha\Big[\sum_{s:l\in s} x_s - \sum_{i:l\in i} \tau_i(\boldsymbol{r})\Big]_{r_l}^{+},\ \forall l \in E', & (21)\end{cases}$$

where

$$\tau_i(\boldsymbol{r}) = \frac{\exp\left(\sum_{l\in i}\beta r_l\right)}{\sum_{i'\in\mathcal{I}}\exp\left(\sum_{l\in i'}\beta r_l\right)},\ \forall i \in \mathcal{I}, \tag{22}$$

$$p_w = \frac{\left(\sum_{s':w\in s'} x_{s'} - C_w\right)^+}{\sum_{s':w\in s'} x_{s'}},\ \forall w \in E_w, \tag{23}$$

and E_w denotes the set of wired links, E_l denotes the set of wireless links, C_w stands for the capacity of wired link w, and p_w is the packet loss ratio of wired link (when it applies drop-tail queue).

Theorem 3. *The equilibrium of the dynamic system in Eq. (19)-(21) solves the following Network Utility Maximization problem:*

$$\textbf{EP:}\ \max\ \sum_s -\frac{1}{x_s} - \frac{1}{\beta}\sum_{i\in\mathcal{I}}\tau_i\log\tau_i - \sum_{w\in E_w}\int_0^{\sum_{s':w\in s'}x_{s'}}\frac{(y-C_w)^+}{y}dy \tag{24}$$

$$s.t.\ \sum_{s:l\in s} x_s \le \sum_{i:l\in i}\tau_i,\ \ \forall l \in E', \tag{25}$$

$$\sum_{i\in\mathcal{I}}\tau_i = 1, \tag{26}$$

$$\boldsymbol{x} \ge 0, \boldsymbol{\tau} \ge 0, \tag{27}$$

where the system utility function is $U_s(x_s) = \sum_{s\in\mathcal{S}} -\frac{1}{x_s}$ and $\int_0^{\sum_{s':w\in s'}x_{s'}}\frac{(y-C_w)^+}{y}dy$ is the penalty function which penalizes the arrival rate for exceeding the wired link capacity c_w.

This theorem can be proved using similar method of the proof of Theorem 2.

From Theorem 3, extension dynamic system solves the rate control problem of multihop networks with wired and wireless links and we have the following observations from the extension dynamic system: 1) in MAC layer, wireless links perform A-CSMA with AQM; 2) link dropping ratio at wireless link is exact equal to packet dropping probability at today's routers (with drop-tail queue). Therefore, our solution discussed in Section V-B can be directly extended to networks with wired and wireless links without modification to any wired network components.

D. Discussions on Other Practical Issues

- Out-of-order application packet delivery. Applications packets are transmitted through multiple TCP connections and potentially arrive at the receiver out-of-order. However, reordering application packets from multiple TCP connections using a receive buffer is a rather mature technology, and has been widely used in peer-to-peer applications, such as BitTorrent file sharing [27]. The same technique can be applied to resolve the problem in our scenario.
- Security problem. People may question what if a hacken application deliberately opens multiple TCP connections to gain advantage over standard users. Note that our proposed solution does not introduce a new security problem. Nowadays, quite a number of application-layer softwares, such as CuteFTP and BitTorrent, already open multiple TCP connections to transfer a file.

VI. Simulations

We evaluate the performance of multi-connection TCP Reno over A-CSMA stated in Section V-A with NS-2 simulation. A-CSMA with AQM is implemented by modifying the IEEE 802.11 module. All simulations are conducted with zero channel losses, so as to give us a clear observation. We set the parameters as follows: 1) k: the coefficient for n_s, is 10; 2) update step size α and update interval for r are 0.05 and 2.0 second, respectively; 3) the weight of entropy term $\beta = 2000$; 4) data rate is 11Mbps; 5) carrier sensing range and transmission range are 800m and 250m respectively to avoid hidden-node problem. The reasons of the parameters setting are: k is set small to give a small r_l (which is interpreted as packet loss ratio), α is set small to give a small step size for the accuracy of convergence, update step size is set large so that the link rate approximates the steady state, and β is set very large in order to mitigate the effect of the entropy term. We do not wish to vary the number of TCP Reno connection too dynamically. In the simulation we update the connection number every 5 seconds. In the following subsections, we will investigate our solution under three different scenarios: single-hop networks, multihop pure wireless networks and multihop networks with wireless links and wired links.

A. Single-hop Networks Scenario

In this part, we consider single-hop wireless link data transmission. We run TCP Reno over L-CSMA, TCP Reno over A-CSMA, and Multi-connection TCP Reno over A-CSMA with AQM on the given topologies. We measure the throughput achieved by each user in different schemes, and compare these throughput with the optimal value. The optimal throughput is computed by solving the utility maximization problem **MP**.

To understand the performance of multi-connection TCP Reno over A-CSMA with AQM, we examine three different topologies with conflict graphs shown in Fig. 2. All topologies consist of four links, with each link carrying a single user data flow. Each of these three topologies has the starvation problem when TCP Reno is running over L-CSMA networks, which has been studied in [2]. In topology a, which has been studied in previous sections, TCP flow running over link 2 starves. In topology b, link 1 starves and links 2, 3 and 4 obtain the whole portion of channel access. In topology c, links 2 and 3 starve, link 1 and link 4 get the half of the channel bandwidth. The simulation results in Fig. 3 shows the link throughputs.

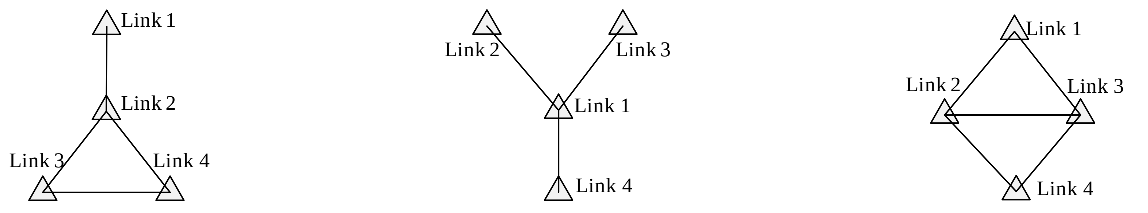
We plot the throughput obtained by different schemes in Fig. 3. Our multi-connection TCP Reno scheme guarantees the contending users in wireless network fairly share the channel. We define utility gap $\Delta U = \sum_s (U(x_s) - U(x_s^*))$ as the difference between the achieved utility and the optimal utility. The closer ΔU to zero, the closer the system to the optimal. We give the utility gap under different scheme in all topologes in Table III. This proves that our solution is capable of realizing the benefit of A-CSMA networks and achieving fair rate allocation in wireless network without any modification to TCP/IP stack in a fully distributed way. We observe that there is an optimality gap between our solution and optimal. Count down wastes some bandwidth, therefore, the performance is surely lower than optimal. Due to wireless random access and the dynamic behavior of TCP, RTT oscillates sharply. While n_s updates slowly (every 5 seconds), so n_s cannot compensate the effect of RTT in Eq. (11). This will result in optimality gap.

TABLE III
Utility gap ΔU under different schemes

	Topo. *a*	Topo. *b*	Topo. *c*	Topo. *d*	Topo. *e*
L-CSMA	-268.5	-541.4	-190.8	-538.3	-1.8
Our Solution	-3.9	-2.8	-3.8	-3.5	-0.9

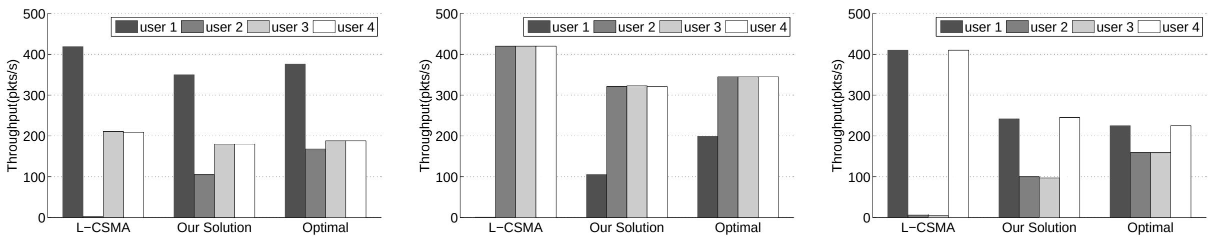
Fig. 4 shows the evolution of transmission aggressiveness $\boldsymbol{r}$ and number of connection n_s of topology a. We observe that by applying our solution, $\boldsymbol{r}$ and the number of connections n_s are stable. Note that dropping probability r of each link, around $0.001 \sim 0.003$, is very small, so it does not waste too much bandwidth by dropping large number of packets. The number of connection n_s link 2 opens is around 15 in the simulation. The number of connections can be reduced by using a small k. Larger k ensures finer granularity in adjusting n_s, thus more responsive to the change in T_s, and therefore better system performance, but at the cost of consuming more device resources. In reality, both link backoff procedure and transmission delay affects RTT, resulting in very dynamic RTT. This is why we see sharp oscillation of n_s in Fig. 4, because n_s is tuned to remove RTT bias. We can smooth the n_s by take the average of the current value and last value, but at the cost of not responsive to the change in T_s.

The results indicate that our multi-connection TCP Reno over A-CSMA with AQM scheme achieves the fair and efficient rate allocation among different users in the single hop



(a) Topology *a* (b) Topology *b* (c) Topology *c*

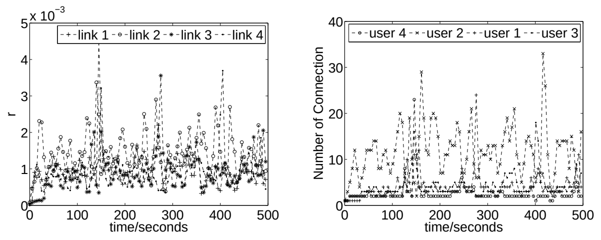
Fig. 2. Link Conflict Graphs of single-hop network topologies.



(a) Topology *a* (b) Topology *b* (c) Topology *c*

Fig. 3. Throughput under 1) L-CSMA: NS-2 simulation of TCP Reno over L-CSMA, 2) our solution: NS-2 simulation of multi-connection TCP Reno over A-CSMA with AQM and 3) optimal throughput computed from problem **MP**.

scenario.



(a) Transmission Aggressiveness (b) Number of connections

Fig. 4. Simulation results for topology *a* under multi-connection TCP Reno over A-CSMA with AQM: curves of transmission aggressiveness and number of connections.

B. Multihop Networks Scenario

In this part, we consider multiple hop networks scenario. The simulated topology and its associated conflict graph under study is shown in Fig. 5(a). It is a nine-node network with six links represented by dashed lines. There are three user flows whose paths are represented by solid lines.

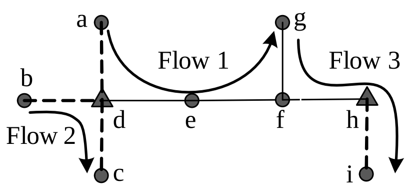
Fig. 5(b) plots the optimal throughput and throughput achieved by multi-connection TCP Reno scheme and TCP Reno over L-CSMA. The middle TCP flow starves when running TCP Reno over L-CSMA. In contrast, multi-connection TCP Reno over A-CSMA with AQM scheme assigns 83 packets to middle TCP flow, which is within 25% of the optimal achievable rate for this topology.

Fig. 5(c) and 5(d) give the evolution of transmission aggressiveness r and number of connection n_s. They are both stable in this scenario. The r is mostly around 0.003, therefore, the dropping probability is not large.

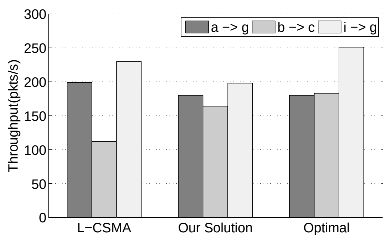
The results indicate that our multi-connection TCP over A-CSMA with AQM scheme is also applicable for multiple hop networks scenario.

C. Multihop Networks with Wireless and Wired Links Scenario

In this subsection, we validate our multi-connection TCP Reno solution for multihop networks with wired and wireless links scenario. The simulated topology under study is shown in Fig. 6(a). It is a nine-node wireless network with two wireless APs denoted by node d and node h. Wireless client nodes a, b and c are associated with AP d while wireless client node i is associated with AP h. Nodes e, f and g are wired nodes. In the graph, wireless links are represented by dashed lines and wired links are represented by solid lines. There are three user flows whose paths are from node a to node g, node b to node c and node i to g respectively. The wired link f-to-g has the link capacity of 2Mbps and other wired links have the link capacity of 20 Mbps. So the wired bottleneck is on the link f-to-g. The wireless link bandwidth is 11 Mbps. Other simulation parameters are the same as the previous simulations.



(a)



(b)

Fig. 6. Multihop networks with wired and wireless links scenario: (a) topology *e* (b) throughput

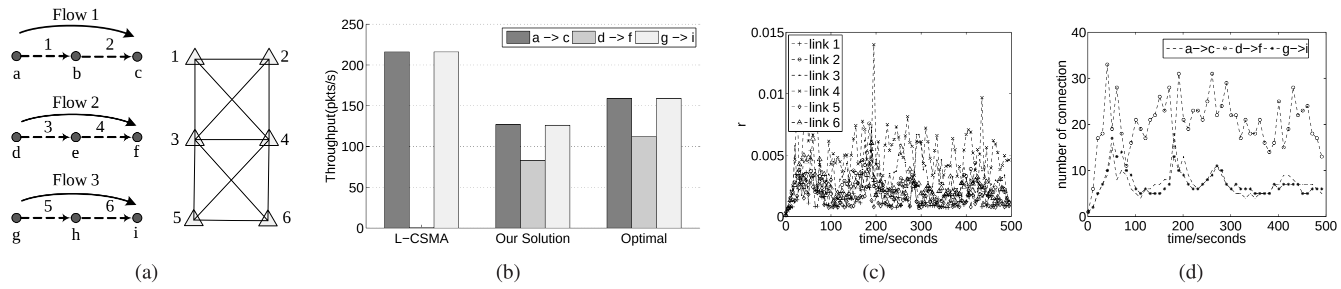


Fig. 5. Multihop networks scenarios: (a) topology *d* and its associated conflict graph. (b) throughput (c) curves of transmission aggressiveness. (d) curves of number of connections.

Fig. 6(b) plots the simulation throughput and optimal throughput. We observe TCP Reno does not suffer server starvation over L-CSMA in this topology. However, the utility gap of our solution is -0.9 better than that of TCP Reno over L-CSMA, whose utility gap is -1.8. This result validates our solution can work properly in multihop networks with wired and wireless links.

VII. Conclusion

In this paper, we study whether TCP Reno (one of the most mature TCP versions) is compatible with adaptive CSMA. We find that running TCP Reno directly over adaptive CSMA suffers from the same severe starvation problems as TCP Reno over legacy CSMA (IEEE 802.11). The root cause is that the system swings towards an unfair rate allocation that starves some links because of the feedback loop of TCP Reno and A-CSMA. We then propose a multi-connection TCP Reno solution to interwork with A-CSMA in a compatible manner. By adjusting the number of TCP connections for each session, our solution can achieve arbitrary system utility, thus realizing the full potentials of A-CSMA. Our solution can be implemented at the application layer or transport layer. Application layer implementation requires no kernel modification, and the solution can be readily deployed in networks running adaptive CSMA. Finally, we show that our solution is applicable to single-hop wireless networks as well as multihop networks with wired and wireless links.

References

[1] X. Wang and K. Kar, “Throughput modelling and fairness issues in CSMA/CA based ad-hoc networks,” in *Proceedins of IEEE INFOCOM*, 2005.

[2] S. Liew, C. Kai, J. Leung, and B. Wong, “Back-of-the-envelope computation of throughput distributions in CSMA wireless networks,” *IEEE Trans Mobile Computing*, 2010. [Online]. Available: http://arxiv.org/pdf/0712.1854

[3] L. Jiang and J. Walrand, “A distributed CSMA algorithm for throughput and utility maximization in wireless networks,” in *Proceedings of Communication, Control, and Computing*, 2008, pp. 1511–1519.

[4] J. Liu, Y. Yi, A. Proutiere, M. Chiang, and H. Poor, “Towards utility-optimal random access without message passing,” *Wireless Communications and Mobile Computing*, pp. 115–128, 2010.

[5] J. Ni, B. Tan, and R. Srikant, “Q-CSMA: Queue-Length based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks,” in *IEEE INFOCOM Mini-Conference*, 2010.

[6] S. Rajagopalan and D. Shah, “Distributed algorithm and reversible network,” in *Proceedings of CISS*, 2008, pp. 498–502.

[7] M. Chen, S. Liew, Z. Shao, and C. Kai, “Markov approximation for combinatorial network optimization,” in *Proceedings of IEEE INFOCOM*, 2010.

[8] J. Lee, J. Lee, Y. Yi, S. Chong, A. Proutiere, and M. Chiang, “Implementing Utility-Optimal CSMA,” in *Proceedings of Allerton Conference*. Citeseer, 2009.

[9] S. Rangwala, A. Jindal, K. Jang, K. Psounis, and R. Govindan, “Neighborhood-centric congestion control for multi-hop wireless mesh networks,” *IEEE/ACM Transactions on Networking*, 2009.

[10] K. Tan, F. Jiang, Q. Zhang, and X. Shen, “Congestion control in multihop wireless networks,” *IEEE Transactions on Vehicular Technology*, pp. 863–873, 2007.

[11] M. Chen and A. Zakhor, “Flow control over wireless network and application layer implementation,” in *Proceedings of IEEE INFOCOM*, 2006.

[12] S. Tullimas, T. Nguyen, R. Edgecomb, and S. Cheung, “Multimedia streaming using multiple TCP connections,” *ACM TOMCCAP*, pp. 1–20, 2008.

[13] X. Lin and N. Shroff, “Joint rate control and scheduling in multihop wireless networks,” in *IEEE CDC*, 2004.

[14] L. Chen, S. Low, and J. Doyle, “Joint congestion control and media access control design for ad hoc wireless networks,” in *Proceedings of IEEE INFOCOM*, 2005.

[15] M. Chen and A. Zakhor, “Multiple TFRC connections based rate control for wireless networks,” *IEEE Transactions on Multimedia*, pp. 1045–1062, 2006.

[16] S. Floyd and V. Jacobson, “Random early detection gateways for congestion control,” *IEEE/ACM Transactions on Networking*, pp. 397–412, 1993.

[17] D. Clark and W. Fang, “Explicit allocation of best-effort packet delivery service,” *IEEE/ACM Transactions on Networking*, pp. 362–373, 1998.

[18] L. Jiang and S. Liew, “Improving throughput and fairness by reducing exposed and hidden nodes in 802.11 networks,” *IEEE Transactions on Mobile Computing*, pp. 34–49, 2008.

[19] C. Chau, M. Chen, and S. Liew, “Capacity of large-scale CSMA wireless networks,” in *Proceedings of MobiCom*, 2009, pp. 97–108.

[20] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, “Impact of interference on multi-hop wireless network performance,” *Wireless Networks*, pp. 471–487, 2005.

[21] B. Baker, “Approximation algorithms for np-complete problems on planar graphs,” *Journal of the ACM*, pp. 153–180, 1994.

[22] R. Boorstyn, A. Kershenbaum, B. Maglaris, and V. Sahin, “Throughput analysis in multihop CSMA packet radio networks,” *IEEE Transactions on Communications*, pp. 267–274, 1987.

[23] S. Shakkottai and R. Srikant, “Network optimization and control,” in *Foundations and Trends in Networking*, 2007, pp. 271–379.

[24] A. Tang, J. Wang, S. Low, and M. Chiang, “Equilibrium of heterogeneous congestion control: Existence and uniqueness,” *IEEE/ACM Transactions on Networking*, p. 837, 2007.

[25] J. Mo and J. Walrand, “Fair end-to-end window-based congestion control,” *IEEE/ACM Transactions on Networking*, pp. 556–567, 2000.

[26] W. Chen, Y. Wang, M. Chen, and S. Liew, “TCP over adaptive CSMA,” *Technique Report*, 2010. [Online]. Available: http://arxiv.org/abs/1007.5239

[27] “The BitTorrent Protocol Specification,” http://www.bittorrent.org/beps/bep_0003.html.