Balancing Throughput and Fairness for TCP Flows in Multihop Ad-Hoc Networks

Yuedong Xu, Yue Wang, John C.S. Lui
Department of Computer Science & Engineering
The Chinese University of Hong Kong
Email: {ydxu, ywang, cslui}@cse.cuhk.edu.hk

Dah-Ming Chiu
Department of Information Engineering
The Chinese University of Hong Kong
Email: dmchiu@ie.cuhk.edu.hk

***Abstract*— Analyzing transport layer operation and enhancing its performance over multihop ad hoc networks have attracted a lot of attentions. Although the fundamental reasons for the performance degradation of TCP have been studied for many years, the tight coupling between transport layer and the wireless MAC layer is still not well understood. In this paper, we focus on the interactions between the hidden nodes and network congestion[1]. By modeling the frame loss ratios of competing flows, a novel index is presented to measure congestion and fairness of these interfered links simultaneously. We formulate a practical optimization framework for TCP flows, and propose a distributed algorithm to improve the end-to-end throughput, and at the same time, provide per-flow fairness by exploiting cross-layer information. In the link layer, each node uses a proportional controller to determine the ECN marking probability for the purpose of notifying incipient congestion. Then the *rate based TCP* sender adjusts its sending rate according to the feedbacks from the link layer. Compared with standard TCP/802.11 as well as recent wireless TCP enhancements, our method substantially improves both long-term fairness and short-term fairness without sacrificing the aggregate end-to-end throughput. For some topologies with long hops, the throughputs of our proposed algorithm even outperform basic TCP/802.11 by over 100%.**

I. Introduction

Transmission Control Protocol (TCP) is designed to provide reliable and efficient end-to-end data delivery, and it is widely adopted in the current Internet infrastructure. With the rapid deployment of Mobile Ad Hoc Networks (MANETs), esp. IEEE 802.11 (WiFi) Ad Hoc networks, TCP protocol is also extended to these networks so as to perform the data transfer in such dynamic and autonomous wireless environments. But TCP was not originally developed for wireless networks in which physical links are relatively un-reliable as opposed to wired networks and the transmissions are interference-limited. As a matter of fact, experimental research has shown that the performance of TCP degrades considerably in 802.11 multihop ad hoc networks [1], [2].

Under wireless environments, TCP not only exhibits serious throughput degradation, but also experiences severe unfairness among competing flows [3]. Much effort has been invested to improve throughput or fairness of TCP traffics over multi-hop ad hoc networks. These previous works can be roughly grouped into two classes: *layered design* [4], [9], [10] and *cross-layer design* [6], [8], [12], [13]. In the layered design, the approach is to find efficient solutions in the transport layer or MAC layer independently. While in the cross-layer design, the interactions among TCP, routing and MAC protocols convey some important information that can be utilized to diagnose the reasons of performance degradation over multihop links. Thus the TCP sender will learn how to adjust its congestion window, retransmission timeout or methods to reply acknowledgement. However, approaches of layered-design such as CWL [4], Few [9], Adaptive ACK [10], do not consider the fairness issue. The cross-layer design such as Neighborhood RED (NRED) [12] sacrifices the aggregate throughput in favor of the per-node fairness. The Link RED [8] uses the average number of MAC retries to generate ECN probability, but it cannot reflect the contributions of competing flows to the network congestion.

In this paper, we quantitatively investigate the impact of hidden nodes on the collision probability and end-to-end performance over 802.11 multihop networks, and show that previous approaches of using the average number of MAC retries or collision probability in an individual node are *inadequate* to guarantee good throughput and fairness. A novel index is proposed to simultaneously measure congestion & fairness among competing TCP flows. We also formulate a practical unconstrained optimization model that captures both the rate allocation and fairness issues. A distributed cross layer algorithm is presented to address the fair rate allocations among TCP flows. The idea is that each wireless node measures its congestion status and uses a proportional controller to calculate the explicit congestion notification (ECN) marking probability of outgoing packets. We illustrate the drawbacks of *window based TCP* (e.g. AIMD mechanism) in 802.11 multihop ad-hoc networks and propose to add a rate adjustment mechanism to the original TCP. If an ECN echoed acknowledgement packet is received by a TCP sender, it will slow down the sending rate by choosing a *larger transmission interval*. Otherwise, the TCP sender increases its sending rate. Our algorithm significantly reduces the number of TCP timeout and drives TCP senders to operate around the optimal average sending rates. Simulation results show that our proposed algorithm can dramatically enhance the TCP throughput and improve the fairness.

The organization of this paper is as follows. In Section II

[1]Congestion generally refers to the channel contentions in IEEE 802.11 multihop ad hoc networks, which is quite different from the buffer overflow in wired Internet.

we first review the background and the important algorithms that our work is based on. In Section III, we analyze the interactions of hidden nodes. In section IV, we formulate an optimization model for multihop TCP flows and propose a distributed control algorithm to address throughput and fairness problems simultaneously. Comprehensive experiments have been carried out to evaluate the proposed algorithm in section V. Finally, conclusion is given in Section VI.

II. **Related Works & Their Challenges**

A. *Overview*

The deployment of TCP in 802.11 multihop ad hoc networks is greatly challenged by throughput degradation and unfairness problems. Most of the previous transport layer or MAC layer designs are focused on either throughput improvement or fairness enhancement respectively. The enhanced TCP versions in [1] [6] are based on the notion that routing dynamic affects the end-to-end performance significantly, authors proposed alternatives to distinguish the packet losses caused by network congestion from those induced by channel errors and routing failures, etc. Chen et al. [4] proposed a congestion window limit (CWL) to identify the bandwidth delay product of a path in MANET. Fu et al. [8] presented a *Link RED* (LRED) algorithm to decide the wireless link's ECN mark probability based on the perceived frame losses. TCP-FEW [9] adopted a fractional window increment to prevent the aggressiveness of AIMD mechanism from degrading TCP performance. Another prominent problem with TCP over wireless network is the contention between TCP data flow and the corresponding acknowledgement (ACK) flow. Oliveira and Braun [10] proposed dynamic acknowledgement strategies to minimize the number of ACK packets in transit and mitigating spurious retransmissions.

The above algorithms improve the end-to-end TCP throughput, however, they cannot guarantee fairness among competing flows with serious interference. For example, fairness cannot be achieved even in a simple symmetric cross-chain or a dumbell topology. To resolve the fairness problem, Xu et al. [12] used a RED-like distributed queue management scheme to balance the throughput among competing TCP flows in multihop ad hoc networks. TCP-AP [13], which is originated from TCP pacing for wired networks, can provide good fairness for TCP flows that span 4-hops. The sender of TCP-AP can adapt its transmission rate using an estimate of the current 4-hop propagation delay and the coefficient of variation of recently measured round-trip times. But TCP-AP relies on the accurate estimation of 4-hop propagation delay and requires the knowledge about the number of hops.

B. *RED-like Link Layer Methods*

Link RED plus Adaptive Pacing: The effects of multihop wireless link on TCP throughput and loss behavior have been studied in [8]. Authors revealed three interesting results. First, given a specific network topology and flow patterns, there exists a TCP congestion window size W^*, at which its throughput is highest via improving the spatial channel reuse. Second, The standard TCP does not operate around the optimal congestion window size W^* and typically grows much larger than the optimal value. Third, the buffer occupancy of wireless nodes is fairly *low* and the packet losses are dominated by channel contention instead of buffer overflow [8]. In the paper, authors proposed an interesting solution based on the observation that TCP can potentially benefit from AQM-like drop/mark mechanism in the MAC layer. The main idea is to further adjust the wireless link's drop probability according to the perceived link frame losses. The end-to-end throughput of link RED algorithm can outperform the standard TCP/802.11 by 4% to 21%. However, the number of retries in a wireless node cannot reflect its contribution to the channel congestion accurately. Link RED might aggravate the unfairness of competing flows in many wireless topologies. We will illustrate this problem quantitatively in the next section.

Neighborhood RED: Because the original RED active queue management can enhance the per-flow fairness in wired networks, Xu et al. [12] extended this algorithm to the distributed neighborhood queues of wireless ad hoc networks. In NRED algorithm, each active node estimates the "total size" of the distributed queue length and applies RED method to calculate the aggregate drop rate of the wireless channel. Through broadcasting the aggregate drop probability to its neighboring nodes, each node can acquire the congestion information beyond its transmission range. The aggregate drop rate is shared by the distributed queues in the neighboring nodes according to their proportions of channel busy time. Neighborhood RED can substantially improve TCP fairness in some topologies and traffic patterns, at the cost of decreased throughput.

Neighborhood RED still has several adverse aspects. First, it sacrifices TCP throughput for fairness under some topologies and traffic patterns. Second, the overhead of neighborhood coordination might be significant. Consider an 802.11 wireless network comprised of four nodes and two links that can sense each other. We tested the impact of periodic broadcast and found that 3% to 5% throughput were consumed by asynchronously broadcasting the notification packets every 0.25s. Periodic broadcast can also result in potential hidden node collisions in multihop scenarios. Third, NRED usually over-reacts to the channel contentions by dropping both the incoming and outgoing data packets, and it is also difficult to configure the controlling parameters. Fourth, NRED is unable to control contentions which are within the carrier sensing range but outside of the transmission range through neighborhood coordination.

In this paper, we propose a novel distributed congestion control algorithm to enhance TCP throughput and fairness simultaneously. A proportional controller is used to decide the mark probability of incoming packets. Based on binary feedback, a *rate-based* TCP congestion control algorithm can greatly outperform the standard 802.11/TCP, as well as some of the recent wireless TCP proposals.

III. **On the Interactions Among Hidden Nodes**

Some previous works on wireless TCP enhancement are based on measurements of MAC retries or collision probability

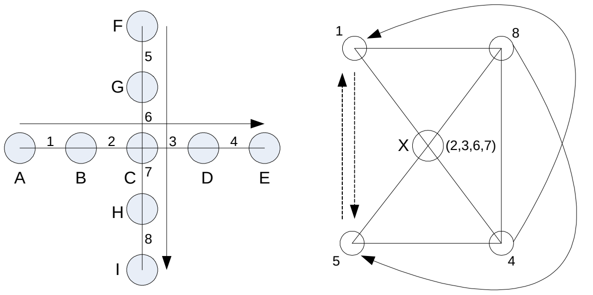

Fig. 1. (a) Short Cross-Chain Topology, (b) Contention Graph

[8], [27]. The goal of this section is to illustrate that solely using the collision probability or the number of MAC retries is not *adequate* to reveal the congestion status. To make this claim, we present the interactions between hidden nodes and transport layer flow rates to show the inadequacy of using collision probability to predict congestion.

A. Contention Graph

Given a set of wireless nodes and flows, a network can be mapped into a contention graph. In general, contention graph is used to represent the interference within the carrier sensing range of a transmitting node. In a general multihop network topology, there exists two types of interferences, namely (a) hidden node contention, and (b) carrier sensing contention.

Based on the general analytical framework in [16], one can construct the flow contention graph and study the influence of hidden links quantitatively. Consider a cross-chain topology as depicted in Fig.1(a) wherein we have two transport layer flows: $A \rightharpoonup E$ and $F \rightharpoonup I$. The link layer flows are represented by numbers in Fig.1. Each node has a transmission range of 250m and a carrier sensing range of 550m. All the neighboring nodes are evenly spaced apart with a distance of 200m. We construct a flow contention graph for this cross-chain topology according to following three steps:

Step 1: From the network topology, we generate an undirected graph that depicts the interference within carrier sensing range of a given node. Node *D*, *F* are beyond the carrier sensing range of node *A*, and node *A*, *H* are outside of the carrier sensing range of node *F*. In brief, link 4 is the *intra-flow hidden link* of link 1, and link 5 is the *inter-flow hidden link* of link 1, while link 1 (link 8) is the *inter-flow hidden link* (*intra-flow hidden link*) of link 5. When the sending rate is low, the transmission of node *D* (or node *H*) can be completed before node *A* (or node *F*) initiates its next frame transmission. Therefore, the intra-flow collisions can be neglected. When the sending rate is high, all the nodes can be saturated and the intra-flow collisions cannot be ignored.

Step 2: From the network topology and the set of active links, one can construct a flow contention graph $G(V, Eg)$ (depicted in Fig.1(b)) that captures the carrier sensing contention, where V represents the set of active links and Eg is the set of undirected edges. For example, link 1 interferes with link 2,3,6 and 7. Then link (1,2), (1,3), (1,6), (1,7) are edges in the set Eg. Because each node in the set {*B, C, G*} can sense all other nodes, for the ease of presentation, we replace the maximal clique $f_{\overrightarrow{BC}}, f_{\overrightarrow{CD}}, f_{\overrightarrow{GC}}, f_{\overrightarrow{CH}}$ by the character *X* in the contention graph.

Step 3: We add the hidden node interference to the flow contention graph using *directed dash lines*. We represent the final contention graph by $G(V, Eg, Eg^{'})$, where $Eg^{'}$ is the set of hidden link interferences. Assume that node D first transmits its buffered packet to node E after a random backoff period by initiating the DATA-ACK or RTS-CTS-DATA-ACK handshakes. After hearing from node D, node B will defer until the transmission is completed. Node A cannot sense the on-going transmission beyond its carrier sensing range and transmits its buffered packet to node B. But node B cannot reply to node A within this period, resulting a frame loss. Similarly, we can find other three hidden nodes' interference depicted in the final contention graph $G(V, Eg, Eg^{'})$ in Fig.1(b).

B. Collision Probability

From a link's perspective (i.e., link i), it can determine the following information: (a) x_i, where $0 \leq x_i \leq 1$ is the normalized airtime in transmitting its frame, (b) γ_i, its collision probability. Similar to those in [14], [15], we assume that the packet collisions are mainly induced by hidden nodes, while the collision event with links inside its carrier sensing range is negligible.

Let γ_{ij} denote the collision probability induced by the j^{th} hidden link of link i and γ_i denote the overall collision probability of link i. Let θ_U and θ_T be the fraction of time used for transmitting a UDP data packet and a TCP data packet respectively. Let TX_d to be the transmitting time of a data packet and T_{ACK} to be the transmitting time of a MAC ACK. TX_a is the transmitting time of an acknowledgement packet if the source uses TCP protocol, we have

$$\theta_U = \frac{TX_d}{DIFS + TX_d + SIFS + T_{ACK}}$$
$$\theta_T = \frac{TX_d}{2*DIFS + TX_d + TX_a + 2*SIFS + 2*T_{ACK}}$$

Since a TCP flow frequently timeouts and is bidirectional in nature (data & ack sub-flows), we focus on the unidirectional transport layer flows in this section in order to simplify our analysis. Let $\theta_U x_i$ and $\theta_U x_j$ represent the normalized transmitting time of data packets for link i and j respectively.

In order to calculate the collision probability of a given link, we introduce three random transmission events. Define $Event_A^{i,j}$ as the event that the normalized data transmission time of link i ($\theta_U x_i$) and link j ($\theta_U x_j$) overlaps, $Event_B^{i,j}$ as the event that the transmission of link j starts before link i, while $Event_C^{i,j}$ is the event that common neighboring links of link i and link j do not transmit. Taking the flow contention graph in Fig.1(b) as an example, one can find that links $2, 3, 6, 7$ are the common neighboring links for link $1, 4, 5, 8$. Base on the analytic method in [16], for link i, the probability of $Event_A^{i,j}, Event_B^{i,j}, Event_C^{i,j}$ are:
Prob$\{Event_A^{i,j}\} = \theta_U x_i + \theta_U x_j$,
Prob$\{Event_B^{i,j}\} = \theta_U x_j$, Prob$\{Event_A^{i,j} Event_B^{i,j}\} = \theta_U x_j$,

Prob$\{Event_C^{1,5}\}$ = Prob$\{Event_C^{1,4}\}$ = Prob$\{Event_C^{5,1}\}$ = Prob$\{Event_C^{5,8}\}$=$1-\bigcup x_c$=$1-x_2-x_3-x_6-x_7$.
The collision probabilities of link 1 and 5 can be represented by:

$$\gamma_{1,5} = Prob\{Event_A^{1,5} Event_B^{1,5} | Event_C\} = \frac{\theta_U x_5}{1-\bigcup x_c}, \quad (1)$$

$$\gamma_{1,4} = Prob\{Event_A^{1,4} Event_B^{1,4} | Event_C\} = \frac{\phi\theta_U x_4}{1-\bigcup x_c}, \quad (2)$$

$$\gamma_1 = \gamma_{1,5} + \gamma_{1,4} = \frac{\phi\theta_U x_4 + \theta_U x_5}{1-\bigcup x_c}, \quad (3)$$

where ϕ is the weight of the intra-flow contentions and $0 \leq \phi \leq 1$. We assume that the competing flows have the same ϕ. Similarly, for link 5, we also have

$$\gamma_5 = \gamma_{5,1} + \gamma_{5,8} = \frac{\phi\theta_U x_8 + \theta_U x_1}{1-\bigcup x_c}. \quad (4)$$

When the offered load is very light, the influence of intra-flow hidden node can be negligible compared with that of inter-flow hidden node. If the network is saturated, the adverse impacts of intra-flow hidden node are more pronounced. Let us consider these two extreme cases:

1) *If the offered load is light, the intra-flow collisions are negligible. The collision probabilities of link 1 and 5 are:*

$$\gamma_1 = \frac{\theta_U x_5}{1-\bigcup x_c}; \quad \gamma_5 = \frac{\theta_U x_1}{1-\bigcup x_c} \quad (5)$$

2) *If the offered load is high and all links are saturated, the influence of intra-flow contentions is prominent, i.e.*

$$\gamma_1 = \frac{\theta_U x_4 + \theta_U x_5}{1-\bigcup x_c}; \quad \gamma_5 = \frac{\theta_U x_8 + \theta_U x_1}{1-\bigcup x_c} \quad (6)$$

According to above analysis, one can find that the collision probability of link 1 is *smaller* than that of link 5 in both cases when the rate of flow $A \rightharpoonup E$ is larger than that of flow $F \rightharpoonup I$. Because the link set {1, 2, 3, 4} belongs to transport layer flow $A \rightharpoonup E$ and the set {5, 6, 7, 8} belongs to flow $F \rightharpoonup I$, the airtime of each link must satisfy the flow constraints: $x_1(1-\gamma_1) = x_2 = x_3 = x_4$ and $x_5(1-\gamma_5) = x_6 = x_7 = x_8$. We have the following claim:

Theorem *1: If the airtime of link 1 is larger than that of link 5, the collision probability of link 5 is not less than that of link 1, and vise versa. Formally, we have:* $(x_1 - x_5)(\gamma_1 - \gamma_5) \leq 0$.

Proof: Please refer to the Technical Report [28]. ∎

C. Congestion Evaluation

The importance of offered load control has been well studied in [14]–[16]. When the offered load increases from a low level, the end-to-end throughput also increases linearly. When the offered load is larger than certain threshold, the throughput decreases. For TCP flows over 802.11 multihop ad hoc networks, the packet drops are mainly caused by MAC contention due to hidden terminals. Fu et.al [8] observed that the TCP packets were dropped even at a low buffer occupancy in a 9-node chain topology.

Link RED [8] measures the average number of MAC retries of each packet and generates the ECN marks with calculated probability. But there is an obvious drawback in the collision based algorithms. The aggressive flow has a smaller collision probability, while the meek flow has a higher collision probability that is illustrated in Theorem 1. Therefore, Link RED exerts *more penalty* on the meek flow than that on the aggressive one.

IV. Optimal Rate Allocation: A Cross-Layer Approach

In this section, we formulate the congestion control of TCP flows as an optimization problem. We propose a novel method to evaluate the channel congestion and adopt explicit congestion notification (ECN) bit as the binary feedback signal. A distributed rate allocation scheme is also presented.

A. Motivations

The resource allocation in wired Internet is originally formulated as an optimization problem in [17]. TCP congestion control algorithms are interpreted as distributed primal-dual problems to maximize the aggregate utility, and a user's utility function is defined by the its TCP version [18]. The fundamental idea is widely extended to the cellular networks and IEEE 802.11 ad hoc networks such as [19]–[22].

Consider a continuous, concave, twice-differentiable and strict increasing *utility function* $U(r_i)$ for a TCP source with rate r_i in an 802.11 multihop ad hoc network. In [19] the congestion control and MAC contention control are designed jointly. The objective is to maximize the aggregate network utility subjected to both flow constraints and link scheduling constraints. Let $\boldsymbol{r}$ be the rate vector of source flows, $\boldsymbol{c}$ be the link rate vector, $\boldsymbol{\varepsilon}$ be the capacity vector of maximal cliques in the contention graph, the optimization problem is:

$$\sum_{\{r_s \in S\}} \max_s U_s(r_s) \quad (7)$$
$$\text{s.t.} \quad R\boldsymbol{r} \leq \boldsymbol{c} \quad \& \quad F\boldsymbol{c} \leq \boldsymbol{\varepsilon},$$

where R is the routing matrix and F is the contention matrix that depicts the mutual interference of wireless links.

This optimization framework captures the major characteristics of rate allocation problem in shared medium and interference-limited networks. However, it has four major problems when one applies to real CSMA ad hoc networks.

1) *The model does not incorporate the hidden node problems in 802.11 multihop scenarios.*
2) *The implementation of the distributed algorithm needs at least the knowledge of local contention graph, while the construction of local contention graph will introduce significant overheads.*
3) *The model is insufficient to describe the dynamic network conditions, esp. $\boldsymbol{c}$, $\boldsymbol{\varepsilon}$, R and F are all time-varying when one considers mobility.*
4) *The clique based optimization framework may produce incorrect solutions in some topologies topologies, (e.g. in a pentagon graph). The maximal normalized sum rate*

is 5/2 according to the optimization model, but the actual sum rate is 2 for CSMA networks [19].

Due to above difficulties, an alternative method is introduced to reformulate the congestion control problem. In this paper, we formulate the TCP congestion control as an unconstrained optimization problem. Except for the utility functions, the objective function contains the penalty function which approximately models the contention and is calculated directly from the wireless nodes' status.

B. Problem Formulation

Given a set of TCP sources $\mathcal{S} = \{1, 2, 3, ..., S\}$ and a set of 802.11 active links $\mathcal{L} = \{1, 2, 3, ..., L\}$, we denote the i^{th} flow's utility function as $U(r_i)$, where r_i is the sending rate of flow i. To send back the congestion prices to the sources, we propose to use the explicit congestion notification (ECN) which is a binary feedback mechanism. A general system-wide objective function $J(\boldsymbol{r})$ can be represented by:

$$\text{maximize} \;\; J(\boldsymbol{r}) = \alpha \sum_{i=1}^{S} U(r_i) - \frac{\beta}{\nu} \sum_{i=1}^{S} p^i_{mark} r_i^{\nu}, \qquad (8)$$

where α and β are constant controlling parameters, p^i_{mark} is the mark probability of the i^{th} source. The first term represents the aggregate utility while the second term represents the penalty (which is expressed as marking probability). A constant integer ν is introduced in order to reinforce the penalty of the aggressive flows when the network gets congested. The objective function can be tuned to achieve the desired trade-off between maximizing the aggregate utility and penalizing frame losses. We choose $\nu = 1$ in this paper for simplicity. Our mathematic model is similar to [22], yet their work focused on the contention control in 802.11 MAC layer and mainly considered the scenarios of multiple single-hop flows within a single cell.

Defining the mark probability of flow i in link l as p_{li}, one can obtain the expression of aggregate marking probability of flow i as:

$$p^i_{mark} = 1 - \prod_{l=1}^{L} (1 - p_{li}). \qquad (9)$$

The mark probability of each link is decided by the measured link variables. Consider a simple case where each link l has only one flow. The marking probability of the i^{th} flow on the l^{th} link, p_{li}, is

$$p_{li} = f(T_l, \gamma_l) = k_p \times \gamma_l T_l^2. \qquad (10)$$

Here, T_l, γ_l are the normalized transmission time and the TCP data packet collision probability of link l respectively, and k_p is the proportional coefficient of the active MAC controller. For multiple TCP flows sharing the same link, the implicit mark probability of i^{th} flow is $p_{li} = \frac{T_{li}}{T_l} f(T_l, \gamma_l)$, where T_{li} is the normalized airtime of i^{th} flow in link l. We denote $\digamma = \gamma_l T_l^2$ as our congestion index and use this index to generate the ECN marking probability at each node. The intuitive idea of the congestion index $\digamma$ is that the aggressive flow should be penalized more even if its collision probability is smaller than its competing flows. The network needs to tolerate the meek flows with larger collision probability. A possible enhancement is to use the congestion index $\digamma$ so one can correctly penalize the aggressive flows and protect the meek ones.

C. Verifying the Congestion Index

In Section III, we analyze the interactions of hidden links among competing flows. The average number of MAC retries, which is a strictly increasing function of collision probability, is insufficient to estimate the congestion and fairness of wireless channels. If we rely on the average number of retries to decide the mark probability, the aggressive flows might occupy most of the channel capacity eventually.

Let us first illustrate the advantages of our proposed index. Consider two TCP flows $A \rightharpoonup E$ and $F \rightharpoonup I$ in Fig.1(a) with *fixed average sending rates*. The link capacity is 2Mbps and the TCP packet size is 1000 Bytes. We will demonstrate how our rate allocation scheme can capture the congestion and unfairness issues simultaneously.

Case 1 - Light Offered load: The sending rate of the TCP flow $A \rightharpoonup E$ is about 100Kbps and another TCP flow $F \rightharpoonup I$ rate is about 50Kbps. The sending time of a TCP packet is randomly chosen in the very small vicinity of a fixed time scale. The average collision probability is measured every 0.5s and the exponential weighted moving average (EWMA) parameter ω is set to 0.125. The collision probabilities of link flow $f_{\overrightarrow{AB}}$ and flow $f_{\overrightarrow{FG}}$ are shown in Fig.2. One can see that the data packet collision probability of the aggressive link flow $f_{\overrightarrow{AB}}$ is smaller than that of the meek link flow $f_{\overrightarrow{FG}}$, which indicates that naviely using the collision probabilty/MAC retries [8], [27] may give an incorrect conclusion of which flow generates more congestion.

Case 2 - Heavy Offered load: In this experiment, We set the rates of $A \rightharpoonup E$ and $F \rightharpoonup I$ to be around 200Kbps and 100Kbps respectively. As is shown in Fig.3, the data packet collision probabilities of link flow $f_{\overrightarrow{AB}}$ and $f_{\overrightarrow{FG}}$ are comparable. This indicates that at heavy load, one may *not* be able to distinguish which flow causes the congestion. Although we investigate the interactions of hidden nodes with unidirectional transport layer flows in Section III, the above experiments of fix-rate TCP flows also coincide with our analytic results to a great extent.

Effectiveness of Congestion Index $\digamma$: We demonstrate the congestion index $\digamma$ of links $f_{\overrightarrow{AB}}$ and flow $f_{\overrightarrow{FG}}$ in the cross-chain topology under both light and heavy offerload patterns in Fig. 4 and Fig. 5. Using $\digamma$ has two advantages. First, it can detect congestion effectively. The congestion indices $\digamma$ of the heavy offered load case are remarkably larger than those of the light offereed load for both link flow $f_{\overrightarrow{AB}}$ and $f_{\overrightarrow{FG}}$, so one can use the value of $\digamma$ to infer congestion. Second, $\digamma$ can identify which flow contributes more to the congestion. The congestion index $\digamma$ of link flow $f_{\overrightarrow{AB}}$ is larger than that of link flow $f_{\overrightarrow{FG}}$ in both the light and heavy offered load case. Therefore, the link layer can penalize the aggressive flow to alleviate the channel congestion.

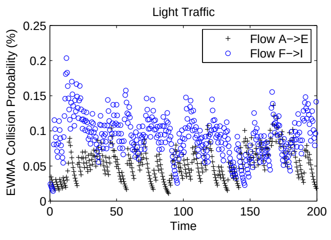

Fig. 2. Col. Prob. of Light Traffic

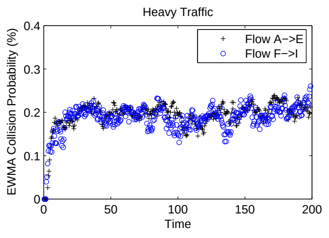

Fig. 3. Col. Prob. of Heavy Traffic

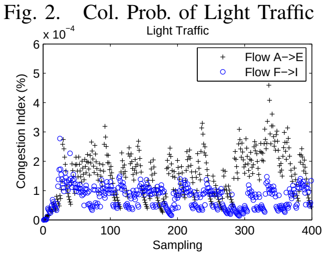

Fig. 4. Index $\digamma$ of Light Traffic

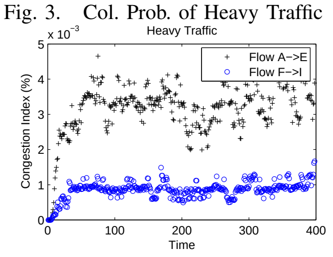

Fig. 5. Index $\digamma$ of Heavy Traffic

D. Distributed Fair Rate Allocation

After we formulate the optimization framework for TCP flows over multihop ad hoc networks, the remaining question is to derive a general methodology for TCP rate adaptation. Note that the system objective function $J(\boldsymbol{r})$ is maximized when

$$\frac{dJ(r_i)}{dr_i} = 0 \iff \alpha U^{'}(r_i^*) - \beta p_{mark}^i = 0 \quad (11)$$

$$\iff \alpha - \frac{\beta p_{mark}^i}{U^{'}(r_i^*)} = 0, \quad (12)$$

where the optimal TCP rate of flow *i* is denoted by r_i^*. Because the utility function $U(r_i)$ is an increasing, strict concave and differentiable function for any $r_i \geq 0$, one can employ the distributed congestion control algorithm using the gradient based method. Let $\dot{r}_i$ be the gradient of rate r_i, we have

$$\dot{r}_i = \alpha - \frac{\beta p_{mark}^i}{U^{'}(r_i^*)}. \quad (13)$$

Therefore, one can easily see that the sending rate r_i arrives equilibrium point r_i^* when $\frac{dJ(r_i)}{dr_i} = 0$. This equilibrium point is actually the solution of system-wide objective function $J(\boldsymbol{r})$. Let us state the following theorem.

Theorem 2: *Suppose all flows in the network have concave, differentiable, strict increasing utility function $U(r)$ for $r \geq 0$, the gradient based rate allocation scheme in Eq.(13) converges to the unique equilibrium point r_i^* for some positive constants α and β.*

Proof: Please refer to the Technical Report [28]. ■

Consider an utility function in the log form, $U(r_i) = \log r_i$, which corresponds to the proportional fairness rate allocation. The congestion control law of TCP flows can be represented by

$$\dot{r}_i = \alpha - \beta p_{mark}^i r_i \ . \quad (14)$$

In the equilibrium state, the following equation holds if we take the utility function of log-form for TCP flows,

$$\alpha = \beta p_{mark}^{i*} r_i^* \ . \quad (15)$$

Proposition 1: *A vector of mutually interfered TCP rates $\boldsymbol{r} = (r_i, i \in R)$ is proportionally fair if it is feasible, that is $\boldsymbol{r} \geq 0$ and $\boldsymbol{r}$ subjects to flow constrains, and if $\boldsymbol{r}$ satisfies the rate allocation scheme Eq.(14).*

Remark: A class of distributed congestion control algorithms can be derived for different goals based on our framework in Eq.(8). If ν is set to be 2, the gradient of the flow control problem is:

$$Grad(r_i) = \alpha - \beta p_{mark}^i r_i^2, \quad (16)$$

and the control law can be chosen as:

$$\dot{r}_i = \frac{\alpha}{r_i} - \beta p_{mark}^i r_i. \quad (17)$$

This controller can accelerate the response speed if the current sending rate deviates far from the optimal rate. The proof for convergence is very close to Theorem 2, which is omitted here.

E. TCP: Window Based VS Rate Based

We propose to add a rate based controller to the existing TCP congestion control mechanism for two reasons. First, TCP acknowledgement sub-flow usually shares the same channel with the TCP data sub-flow in 802.11 wireless networks. The backward queueing delays of acknowledgement packets are mainly composed of the transmission time of TCP data packets. Therefore, if the congestion window increases, the RTTs also increase significantly, and vice versa. We have experimentally observed that the average sending rate of a TCP flow with the constant congestion window size of 3 is not very far from that with the constant window size of 6 in a 8-hop chain topology. Second, the bandwidth delay product of a general multihop ad-hoc network is very small, thus the optimal congestion window size is small too (i.e. 2 to 5). Therefore, it is not efficient for TCP's AIMD to adjust on a very small congestion window. In this paper, we use ECN bits to notify the sources to decrease the sending rate where it is the reciprocal of the sending interval T_{intv}. If a TCP sender receives an ECN echoed acknowledgement, it will slow down the sending rate by choosing a larger transmission interval T_{intv}. If an acknowledgement packet is not ECN echoed, the source will decrease the transmission interval slightly. In summary, the operations are:

1. If an ECN echo is reveiced, $T_{intv} = \beta \cdot T_{intv}$; $(\beta > 1)$
2. If ECN is not received, $T_{intv} = \frac{T_{intv}}{1 + \alpha T_{intv}}$, $(\alpha > 0)$

Our design utilizes the potential benefits of cross-layer information. The proposed distributed congestion control algorithm, which we call the DCC algorithm, adds a sub-layer between transport layer and MAC layer to gather channel information and generate feedback signals for rate adaptation.

F. Practical Considerations

As a practical protocol, our DCC algorithm needs to avoid the some kinds of wrong manipulations, that is to establish the upper and lower bounds for the transmitting interval T_{intv}. We provide an upper bound T_{ub} of two average RTTs for a sending interval, which corresponds to "$CWND = 0.5$" in other TCP variants.

Lower bound T_{lb} is important for it decides maximal sending rate of a TCP source. Authors in [5] investigated bandwidth delay product (BDP) in an 802.11-based MANET. A delay based upper bound of BDP of a chain topology with more than 4 hops is derived. The upper bound of bandwidth delay product of a chain topology cannot exceed $\frac{\sum_{i=0}^{n} d_i + \sum_{i=0}^{n} d_i'}{4d_{max}}$, where d_i and d_i' are the per-hop packet transmission delays along the forward and return paths, and d_{max} is the maximum per-hop delay. This is to say, the minimum sending interval is $4 \times d_{max}$. If we choose $4 \times d_{max}$ as the lower bound, the knowledge of hop count is indispensable.

But in our distributed congestion control algorithm, the aim of lower bound is to avoid the extreme worse case. For simplicity, we just set the lower bound to be the aggregate one-hop transmission time of a TCP data packet and an ack packet, that is

$$\begin{aligned} T_{lb} &= TX_d + DIFS + SIFS + T_{ACK} + Backoff \\ &+ TX_a + DIFS + SIFS + T_{ACK} + Backoff \end{aligned}$$

The upper and lower bounds are rarely met during the transmission unless there is very few contention loss or most of the in-flight packets collide with the packets from the hidden links. We recommend to impose a slightly larger lower bound in the TCP slow start stage to suppress the aggressiveness. The detailed description of the algorithm is shown in Fig.6.

Distributed Congestion Control Algorithm

```
Part 1 : Calculate_MarkProb()
1: For each transmission
2:     ntrans ++
3: For each failed transmission
4:     failedcount ++
5: If (DccMacTimerExpire)
6:     Measure Normalized Transmitting Time $T_l$
7:     retry_prob = failedcount/ntrans
8:     OverLoad = retry_prob$\times T_l^2$
9:     $p_l = k \times Overload$
10:end if

Part 2: TCP_Rate_Adaptation()
1: Disable ECN Induced CWND Adaptation (optional)
2: If (ECN-echoed ACK)
3:     $T_{intv} = \beta \times T_{intv}$
4: else if (Non-ECN-echoed ACK)
5:     $T_{intv} = \frac{T_{intv}}{1 + \alpha T_{intv}}$
6: end if
7: Check upper & lower bounds of $T_{intv}$
8: If (DccTCPTimerExpire)
9:     If (seqno $<$ cwnd + highest_ack)
10:        send(pkt)
11:    end if
12:end if
13: Adjust CWND size according to AIMD,
    or Choose Constant CWND size
```

Fig. 6. Algorithm Specification

V. Simulations

A. Simulation Setup

We evaluate the end-to-end throughput and fairness of our proposed algorithm and other related schemes such as TCP-FEW [9], CWL [4] and Link RED plus adaptive pacing [8] over a variety of network scenarios. The transmission range, carrier sensing range, hop distance, TCP packet size and link capacity are the same as those in the preceding experiments. The original TCP/ECN mechanism is disabled. Because RTS/CTS does not help much to reduce collisions caused by hidden links when the carrier sensing range is more than twice that of the transmission range, we also disable the RTS/CTS handshake. In our distributed congestion control algorithm, the sampling interval in the 802.11 MAC layer is set to 0.125 second and the EWMA weighting factor is set to 0.125. The decrement factor β is 1.05 because it is more effective to tune the sending rate than to decrease the congestion window for the purposes of avoiding congestion. The increment factor α is set to 0.1. The above parameters are fixed and serve as standard configurations in all the simulations. Therefore, only the proportional parameter k_p is configurable which can range from 0 to 0.1. The default value of k_p is set to 13.2030 in our experiments.

B. Chain Topology

Our DCC proposal is tested over a chain topology in Fig.7 with different number of hops and with different parameter configuration. We demonstrate both the dynamic behaviors and end-to-end throughput.

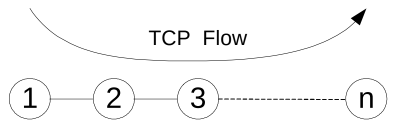

Fig. 7. Chain Topology with a single TCP flow

1) Dynamic Behaviors: Consider a single TCP flow over a 9-node chain topology, the end-to-end throughput of the TCP flow is sampled every 2 seconds. Fig.7 illustrates that the throughput of a standard TCP flow tend to strongly oscillate over time. Because the first two wireless nodes sense smaller interference than their downstream nodes, the source will send more traffic into the network than it can successfully transmit. When a downstream node fails to transmit a frame after a number of retries, the frame will be discarded and a link breakage is reported. During this period, no packet can be transmitted until the routing protocol fixes the route or discover a new route. Authors in [9] elaborated the details of the cross-layer interactions leading to the severe network instability. The DCC algorithm well solves the routing instability problem. The end-to-end throughput is stabilized by controlling the TCP sending rate according to the link feedbacks.

2) Throughputs VS Hops: We compare the average throughput of DCC algorithm and other TCP enhancements by varying the number of hops in the chain topology. The increment parameter of TCP-Few is set to 0.01, and the

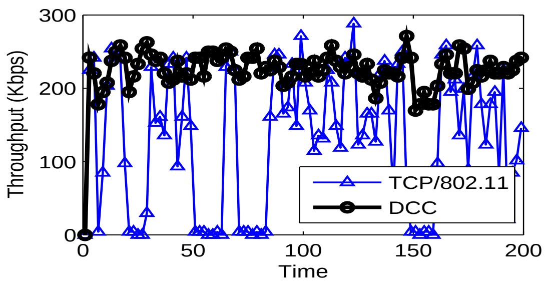

Fig. 8. End-to-end Throughput

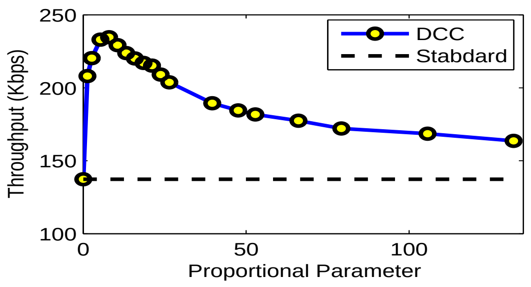

Fig. 10. Robustness of the proportional parameter k

congestion window limits of CWL are the same as those in [4]. The minimum, maximum thresholds and the maximum mark probability of Link RED plus adaptive pacing are 0.2, 2.0 and 0.2 respectively. We set the Delayed ACK (DACK) interval to be 100ms. As is shown in Fig.9, the throughput of DCC algorithm is significantly better than standard TCP/802.11. When the number of hops is larger than 11, our DCC algorithm outperforms all the rest schemes in terms of throughput. Later we will show that the proposed algorithm can achieve very good fairness in the cross-chain and grid topologies, while the other schemes cannot.

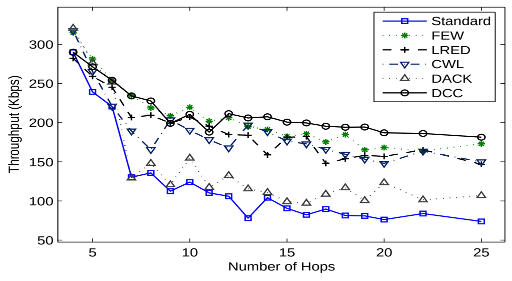

Fig. 9. Throughput VS Hops

3) Robustness of the Controller: The selection of controlling parameters is very important for all the enhanced algorithms. A good algorithm should tolerate the imperfect parameter configurations and maintain the robustness of performance. In this set of experiments, the robustness of the proportional parameter k_p is tested in an 8-hop chain topology. When we vary k_p from 0 to 132.030, the maximum throughput 236.1Kbps is achieved at $k_p^* = 7.922$. Fig. 10 shows that at the point k_p being 13.203, the end-to-end throughput is 164.67Kbps. Note that even when k_p is arbitrarily chosen to be nearly 20 times of k_p^*, the throughput is still significantly larger than the standard TCP/802.11 mechanism.

C. Cross-Chain Topologies

In this set of simulations, we concentrate on the fairness issues of competing TCP flows.

1) New Insights of TCP fairness: TCP unfairness is observed in a lot of previous research. But they merely concern the long-term unfairness, while neglecting the short-term unfairness. Consider the two long-lived TCP flows from node A to E and node F to I in Fig.1(a), serious unfairness is caused by the interactions of congestion control, routing and 802.11 MAC protocols. If two flows starts at the same time, one flow will occupy the whole wireless channel for a long time until it timeouts due to intra-flow contentions. Then another TCP flow has an opportunity to take over the channel. If the average end-to-end throughputs are measured in a very long time interval, the unfairness seems not so severe. However, when the throughputs over time are examined in a fine-grained mode, the two competing flows achieve quite different short-term throughput. There is a tradeoff between throughput and fairness. Some enhanced TCP versions can improve the aggregate throughput, yet aggravate the per-flow fairness. The reason lies in that the maximum aggregate throughput is achieved only when one of the flows occupies the whole channel capacity and operates around the optimal sending rate. In our distributed congestion control algorithm, the proportional parameter k is a knob to balance the aggregate throughput and per-flow fairness.

2) Cross-Chain Topology: We run extensive simulations to investigate the unfairness issue with two competing TCP flows on a cross-chain topology in Fig.1(a). The parameter configurations are the same as those in the previous experiments. We measure the average aggregate throughput in 200 seconds and calculate the fairness indexes according to [24]:

$$Fairness\ Index\ :\quad F(\boldsymbol{r}) = \frac{(\sum_{i=1}^{N} r_i)^2}{N\sum_{i=1}^{N} r_i^2}.$$

In our experiment, the aggregate throughput of DCC algorithm is only larger than that of Link RED and the Fairness Index of TCP-FEW algorithm is the largest over 200s simulation. If we judge fairness depending on the fairness index over at least 200s, we will draw a wrong conclusion that all the above algorithms can achieve satisfactory fair rate allocation. In order to highlight the short-term unfairness, we measure the average throughput of above algorithms every 2 seconds. Fig.11 to 17 show that our DCC algorithm can guarantee both the long-term and short-term fairness for the two competing TCP flows: $A \rightharpoonup E$ and $F \rightharpoonup I$ in the cross-chain topology while other schemes cannot. For standard TCP and other enhanced algorithms, the throughput of one TCP flow is extremely low, even equal to zero, when its competing flow begins to transmit. In addition, dynamic behaviors of the sending intervals of the DCC algorithm are shown in Fig.17. Because the sending intervals of two competing TCP flows are very close to each other, the short-term fairness is guaranteed. Our DCC algorithm can also improve throughput and short-term fairness in long cross-chain and dumbell topologies, which is omitted due to page limit.

D. Grid Topology

On a 5×5 grid topology shown in Fig.18, we run six long-lived FTP flows. One can see in Fig.19 that the middle flows FTP2 and FTP4 under the CWL, DACK and standard TCP/802.11 schemes suffer from serious unfairness while the DCC can provide good fairness for all the flows. The short-term fairness is also evaluated and Fig.20 shows that the instantaneous fairness index of DCC algorithm is about 0.9 and the fairness indexes of other algorithms oscillate around 0.5. A fairness index of 0.5 represents very serious unfairness among competing flows. Although the aggregate end-to-end throughput of our DCC algorithm is slightly less than that of standard TCP/802.11 and other enhanced methods, it achieves much better fairness even in small time scales.

VI. Conclusion

Improving the performance of TCP over IEEE 802.11 multihop ad hoc networks has attracted a lot of attentions. In this paper, we study the interactions between hidden nodes and transport layer flows, and show some new insights and fundamental properties of TCP unfairness with competing flows. The close coupling among the MAC protocols, routing schemes and TCP congestion control can easily lead to performance degradation of wireless networks. A potential way to enhance TCP performance is to control the TCP traffics by exploiting the cross-layer information. The contributions of this paper are:

1. We reveal the relationships between traffic rates and collision probabilities due to hidden nodes in a wireless multihop environment with competing flows.

2. We present a novel congestion index for a general multihop ad hoc network. The congestion index can effectively evaluate the link's congestion status and per-flow fairness at the same time.

3. We propose an unconstrained optimization framework for multihop TCP flows that does not need the global information of the networks or any local coordination.

4. We propose a fully distributed congestion control algorithm to balance throughput and fairness for TCP flows in multihop ad hoc networks. The advantages of our DCC algorithm are validated through NS-2 simulations in a variety of network environments.

References

[1] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks", *Wireless Networks*, Vol.8, Pages: 275-288, 2002.

[2] S. Xu and T. Saadawi, "Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks", *Elsevier Computer Networks*, Vol.38, Pages: 531-548, 2002.

[3] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?", *IEEE Communication Magazine*, Vol.39, Pages: 130-137, 2001.

[4] K. Chen, Y. Xue and K. Nahrstedt, "On Setting TCP's Congestion Window Limit in Mobile Ad Hoc Networks", *Proceedings of ICC'03*, Vol.2, Pages: 1080-1084, Alaska, May 2003.

[5] K. Chen, Y. Xue, S. Shah and K. Nahrstedt, "Understanding Bandwidth-Delay Product in Mobile Ad Hoc Networks", *Elsevier Computer Communications*, Vol.27, Pages: 923-934, 2004.

[6] X. Yu, "Improving TCP Performance over Mobile Ad Hoc Networks by Exploiting Cross-Layer Information Awareness", *Proceedings of ACM Mobicom 2004*, Pages: 231-244, 2004.

[7] Z. Fu, X. Meng, and S. Lu, "How bad TCP can perform in mobile ad-hoc networks", *Proceedings of IEEE ICNP 2002*, Paris, France, 2002.

[8] Z. Fu, P. Zerfos, H. Luo, S. Lu, et al, "The impact of multihop wireless channel on TCP performance", *IEEE Trans. Mobile Computing*, Vol.4, Pages: 209-221, 2005.

[9] K. Nahm, A. Helmy, C. Kuo, "TCP over Multihop 802.11 Networks: Issures and Performance Enhancement", *Proceedings of ACM/IEEE Mobihoc 2005*, Pages: 277-287, 2005.

[10] R. Oliveria, T. Braun, "A Dynamic Adaptive Acknowledgement Strategy for TCP over Multihop Wireless Networks", *Proceedings of IEEE Infocom 2005*, Vol.3, Pages: 1863-1874, Miami, USA, 2005.

[11] K. Sundaresan, V. Anantharaman, H. Hsieh and R. Sivakumar, "ATP: a reliable transport protocol for ad-hoc networks", *Proceedings of ACM MobiHoc 2003*, Annapolis, ML, 2003.

[12] K. Xu, M. Gerla, L. Qi and Y. Shu, "Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED", *Proceedings of ACM Mobicom 2003*, Pages: 16-28, San Diego, USA, 2003.

[13] S. Elrakabawy, A. Klemm, C. Lindemann, "TCP with Adaptive Pacing for Multihop Wireless Networks", *Proceedings of ACM/IEEE Mobihoc 2005*, Pages: 288-299, 2005.

[14] C. Ng, S.C. Liew, "Offered Load control in IEEE 802.11 Multi-hop Ad-hoc Networks", *Proceedings of IEEE Int. Conf. on Mobile Ad-hoc and Sensor System*, Pages: 80-89, Oct 2004.

[15] Y. Gao, D.M. Chiu, C.S. Lui, "The fundamental role of hop distance in IEEE 802.11 multi-hop ad hoc networks", *Proceedings of IEEE Int. Conf. on Network Protocols*, Pages: 1-10, 2005.

[16] Y. Gao, D.M. Chiu, C.S. Lui, "Determining the end-to-end throughput capacity in multi-hop networks: methodology and applications", *ACM SIGMetrics/Performance'06*, Pages: 39-50, 2006.

[17] F.P. Kelly, A.K. Maulloo, D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability", *Journal of the Operational Research Society*, Vol.49, Pages: 237-252, 1998.

[18] S.H. Low, D.E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence", *IEEE/ACM Transactions on Networking*, Vol.7(6), Pages: 861-875, 1999.

[19] L. Chen, S.H. Low, J.C. Doyle, "Joint Congestion Control and Media Access Control Design for Ad Hoc Wireless Networks", *Proceedings of IEEE Infocom 2005*, Vol3, Pages: 2212-2222, 2005.

[20] M. Chiang, "Balancing Transport and Physical Layers in Wireless Multihop Networks: Jointly Optimal Congestion Control and Power Control", *IEEE Journal on Selected Area in Communications*, Vol.23, No.1, Pages: 104-116, 2005.

[21] J. Mo, J. Walrand, "Fair end-to-end window-based congestion control", *IEEE/ACM Trans. on Networking*, Vol.8, No.5, Pages: 556-567, 2000.

[22] T. Nandagopal, T.E. Kim, X. Gao and V. Bharghhavan, "Achieving MAC layer fairness in wireless packet networks", *Proceedings of ACM Mobicom 2000*, Pages: 87-98, 2000.

[23] J. Li, C. Blake, D.S.J. Couto and H.I. Lee, "Capacity of Ad Hoc Wireless Network", *Proceedings of ACM Mobicom 2001*, Rome, Italy, July 2001.

[24] D. Chiu, R. Jain, "Analysis of the increase and decrease algorithm hms for congestion avoidance in computer networks", *Computer Networks and ISDN Systems*, Vol.17, Pages: 1-14, 1989.

[25] C.E. Kosal, H. Kassab and H. Balakrishnan, "An analysis of short-term fairness in wireless media access protocols", *Proceedings of ACM Sigmetrics 2000*, Pages: 118-119, California, 2000.

[26] J. Mo, J. Kawk and J. Walrand, "Revisiting the Joint Transport and MAC Optimization for Wireless Ad Hoc Networks", *Proceedings of Wiopt 2006*, Boston, 2006.

[27] K. Kim, P. Lorenz, M. LEE, "A New Tuning Maximum Congestion Window for Improving TCP Performance in MANET", *Proceedings of Systems Communications 2005.*

[28] Y. Xu, Y. Wang, C.S. Lui, D.M. Chiu, "Balancing Throughput and Fairness for TCP Flows in Multihop Ad Hoc Networks", *CUHK Technical Report, Oct. 2006.*

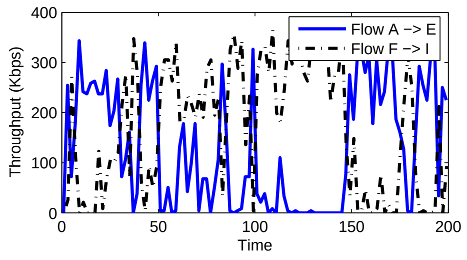

Fig. 11. Standard TCP/802.11

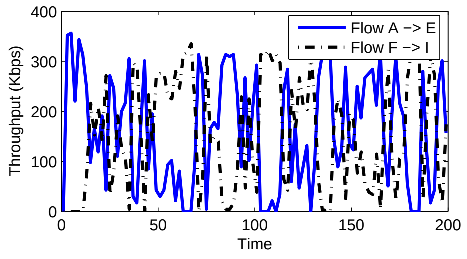

Fig. 12. TCP-FEW

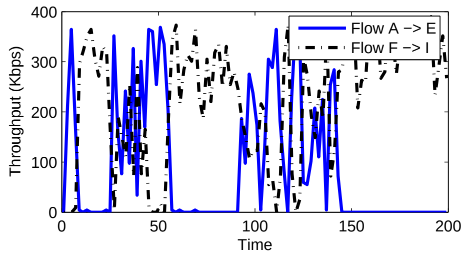

Fig. 13. Delayed ACK

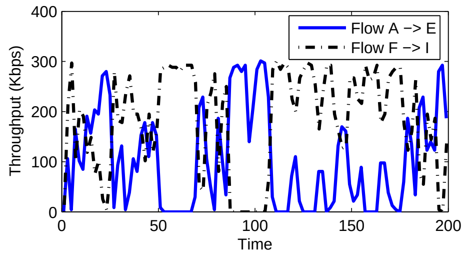

Fig. 14. Link RED

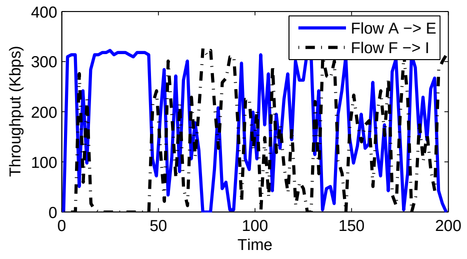

Fig. 15. Congestion Window Limit (CWL)

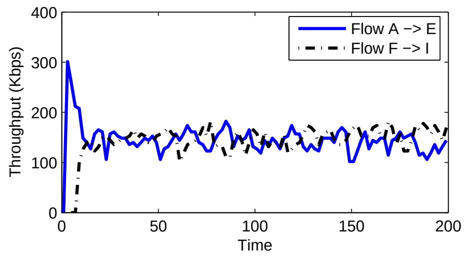

Fig. 16. Distributed Congestion Controller

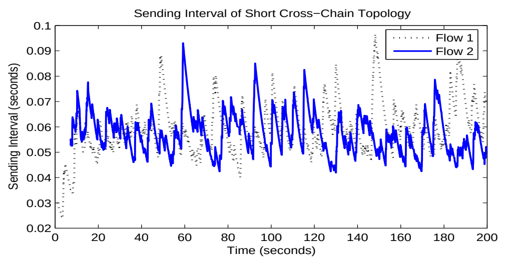

Fig. 17. Sending Intervals of the Competing Flows over Time

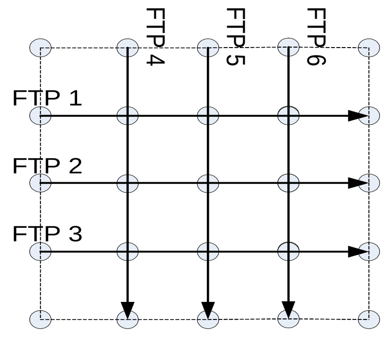

Fig. 18. A 5×5 Grid Topology

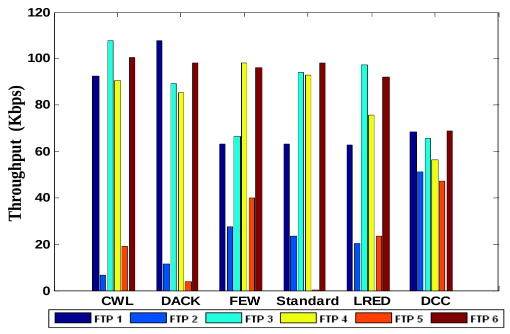

Fig. 19. Average Throughput of Grid Topology

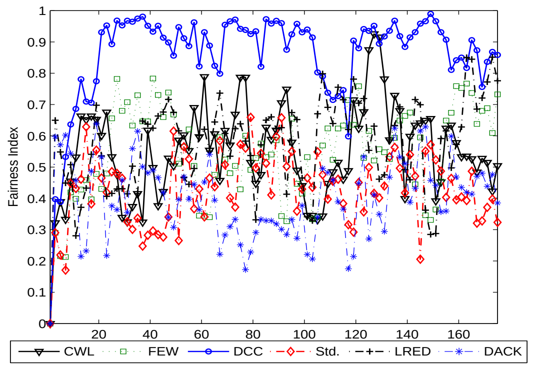

Fig. 20. Fairness Index Evolution over Time