# Implementing a Cooperative MAC Protocol for Wireless LANs

Thanasis Korakis+, Sathya Narayanan$^{\alpha}$, Abhijit Bagri*, Shivendra Panwar+

+ Department of Electrical and Computer Engineering, Polytechnic University, Brooklyn, New York
$\alpha$ Panasonic Digital Networking Laboratory, Two Research Way, 3rd Floor, Princeton, NJ 08540
* Indian Institute of Technology, Kanpur, India

e-mail: korakis@poly.edu, sathya@research.panasonic.com, abagri@iitk.ac.in, panwar@catt.poly.edu

***Abstract*— In wireless LANs that provide multi-rate support (IEEE 802.11a, 802.11b), stations that experience poor channel quality tend to use low transmission rates to reduce the bit-error-rate (BER) of each transmission. This phenomenon usually leads to a throughput fairness problem between the stations with good channel quality and those without. This fairness problem has been shown to result in throughput degradation for the whole network [8]. The MAC protocol proposed in [5] addresses this issue using an efficient cooperative scheme. Under this scheme, low rate stations are assisted by a high rate station, referred to as helper stations, in its transmissions. With such assistance, the low rate station will be able to transmit data at a higher rate in a two-hop manner using the helper station. We implemented this new protocol in a Linux testbed. This paper describes the assumptions, the implementation process and the challenges we were presented with. We evaluated the protocol using our testbed through experiments. The implementation of the protocol shows that it performs efficiently in supporting TCP applications.**

## I. Introduction

As wireless technology becomes more popular, there is interest in setting up dense infrastructure wireless networks in both enterprises and public hot-spots such as airports. In such an environment, the provision of a fair share of the medium to each user is of major importance. It is preferable that the service a station experiences is independent of its position in the coverage area and its distance to the associated Access Point (AP). But, due to the fundamental limitations of the wireless channel, it is not possible for high data rates to be maintained as the distance between the source and destination increases [10].

Wireless networks that provide multi-rate support give the stations the ability to adapt their transmission rate to the the link quality in order to make their transmissions more reliable. Thus, stations that experience poor channel conditions tend to use lower transmission rates and vice versa [9], [11].

There are two cases where stations will decide to transmit at a low transmission rate:

- When a station experiences a bad channel due to its distance from the Access Point.
- In an indoor environment with strong shadowing and fading effects, a station may experience a low quality channel at some spots in the coverage area.

In both cases, low speed stations occupy the channel for a longer duration, leading to higher delays and decreasing the overall throughput [3] [8].

The authors of [5] propose a protocol that solves this problem. The protocol is based on a simple but efficient cooperative scheme where intermediate stations between the low speed station and the Access Point can act as helpers in the transmission process. Instead of having a slow station transmitting its frame directly to the Access Point, an alternative route through a high speed station is used sending the frame in a two-hop manner. Thus, instead of using one low data rate transmission, two high data rate transmissions are used, decreasing the amount of time the channel is occupied for that particular transmission. This cooperative protocol is compatible with the widely used IEEE 802.11 standard and hence can be deployed incrementally.

We implemented this protocol by modifying the Linux wireless driver HostAP [6]. HostAP is a driver for wireless LAN cards based on Intersil's Prism 2/2.5/3 chipset. It is one of the widely used drivers for experiments as it supports access point functionality in the software, allowing modifications to the MAC layer functionality. As we plan to build upon the implementation reported in this paper to a complete cooperative infrastructure network, HostAP was the natural choice for us.

We set up a testbed and ran experiments using file transfers of various file sizes. In this paper we present the assumptions, the implementation process, the challenges we came across during this process as well as the results of the experiments. We investigate the circumstances under which a helper's participation in the transmission process is advantageous. The implementation results show that the protocol performs efficiently in a real implementation and that the added computation and throughput overhead in its implementation is negligible.

The rest of the paper is organized as follows: In section II we give a brief description of the IEEE 802.11b MAC and Physical layer. In section III, we give the details of the protocol. In section IV, we describe the implementation process and challenges. Finally, in section V we discuss the results of the experiments we ran in the testbed.

1-4244-0355-3/06/$20.00 (c) 2006 IEEE

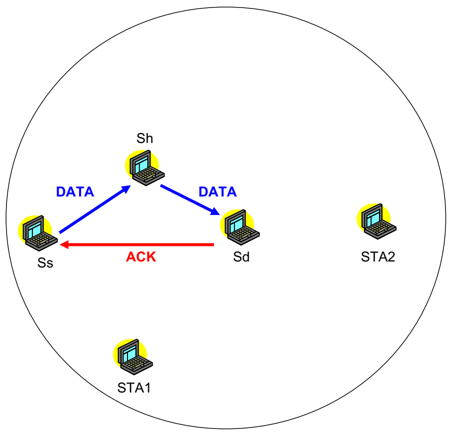

Fig. 1. Exchange of data-ACK frames for Cooperative MAC

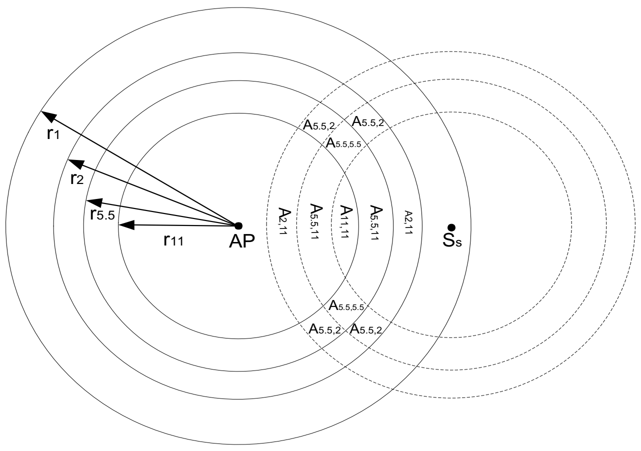

Fig. 2. Cooperative regions for Cooperative MAC

## II. IEEE 802.11B

IEEE 802.11 [1] is a standard that defines the MAC and Physical layer protocol for wireless LANs. The MAC layer is based on a distributed mechanism that is called the Distributed Coordination Function (DCF). It is a CSMA/CA scheme under which every station contends for the medium by sensing if it is idle for a specific period of time called the DIFS interval followed by a random interval to avoid collision. If the station succeeds in getting control of the medium, it transmits its frame and waits for an acknowledgment (ACK) from the receiver. There is also an optional feature of using Request To Send (RTS) and Clear To Send (CTS) frames before the data transmission, in order to ensure that all nodes within hearing range of the sender and receiver are informed about the impending data transmission. The exchange of the four frames is known as the four-way handshake. The time period that the medium remains idle between the frames of the four-way handshake is a constant called the SIFS interval. By defining SIFS to be smaller than DIFS, 802.11 MAC design ensures that the exchange of these frames is not interrupted by a transmission from any other neighboring station.

The band that 802.11b [2] uses is the 2.4GHz band. In the Physical layer, it deploys three different modulation schemes to support 4 different transmission rates, 1, 2, 5.5 and 11 Mbps.

## III. THE COOPERATIVE MAC PROTOCOL

The CoopMAC protocol we implemented is described explicitly in [5]. In this paragraph we provide a high level overview in order for the reader to be able to get a basic understanding and to follow the implementation details. Readers are referred to [5] for a detailed discussion of the CoopMAC protocol.

The basic functionality of the proposed protocol is illustrated in Fig. 1. In this figure, $S_s$ is the source station, $S_d$ is the destination station and $S_h$ a potential helper. The potential helper is an intermediate station between the source and the destination that is able to exchange data with the source and the destination at rates higher than the rate of the direct link between them. As we can see in the figure, the source station, instead of sending its data directly to the destination using a low data rate transmission, transmits the data in a two-hop manner using the station $S_h$ as a helper. The advantage of the two-hop transmission is that the two links that are used are fast and thus the overall time for the transmission from the source to the destination is reduced. When the helper receives the frame from the source, it retransmits it to the destination after a SIFS time, and thus avoids the need to contend for the medium. After the reception of the frame from the helper, the destination station sends a direct ACK to the source, acknowledging the reception.

The key issue in the previous scheme is the right selection of the helper. In order to achieve this, every station maintains a table, called a CoopTable, of all the possible helpers around it. Each row of this table corresponds to a potential helper. Every time a station receives a packet (RTS, CTS, data or ACK) from any other station, it checks if the transmitting station is already in the table. If not, a new row is added for the particular station on the table. Then it updates the corresponding row with information it takes from the receiving packet. For the control packets (that are always transmitted in the basic rate) it calculates the path loss taking under consideration the transmission and the reception power and then it estimates the appropriate transmission rate between itself and the sender of the packet. For the data packets it detects directly the transmission rate, because this information is included in the physical layer header of the packet.

The CoopMAC protocol also defines a five-way handshake mechanism involving RTS-CTS with a new message called HTS (Helper ready To Send). Details of this five-way handshake mechanism and the maintenance of the required information in a CoopTable can be found in [5].

When a station $S_s$ has $L$ octets of data to send to a destination station $S_d$, and its data rate is either 1 Mbps or 2 Mbps, it checks the CoopTable and calculates the time needed to transmit via each potential helper. Since the transmission will be in two steps, first from the source to the potential helper and then from the potential helper to the destination, the transmission time is $8L/R_{sh} + 8L/R_{hd}$, ignoring contention time and overhead, where $R_{sh}$ is the data rate between the source and the helper, and $R_{hd}$ is the data rate between the helper and the destination. After checking all the potential helpers, it selects the one with the minimum transmission time. If more than one potential helper has the same minimum transmission time, one of the helpers could be chosen arbitrarily or chosen based on the last known transmission. This chosen helper is denoted by $S_h$. If the direct transmission rate is $R$ between $S_s$ and $S_d$, $8L/R$ is the time needed for direct transmission. If $8L/R_{sh} + 8L/R_{hd} < 8L/R$, two hop transmission via the helper will be more efficient. If $8L/R_{sh} + 8L/R_{hd} \geq 8L/R$, the frame will be transmitted directly, the same as in the current 802.11 MAC standard.

A typical scheme of regions of potential helpers is shown in Fig.2. If a helper exists in region $A_{x,y}$, station $S_s$ can transmit its frames in two hops using a rate of $x$ Mbps from itself to the helper and a rate of $y$ Mbps from the helper to the destination.

For a given node $S_s$, the likelihood of finding a helper with a particular two-hop data rate combination is related to the area of overlap shown in Fig.2. The authors of [5] derive an expression for this probability, in a network with uniform distribution of nodes. The experimental results presented in this paper assume the existence of a helper for each test scenario.

## IV. Implementation Issues

### A. Assumptions

Before we go in to the details of our implementation and experiments, it is important to list the assumptions made for this first version of our implementation. A discussion of how we intend to loosen these initial restrictive assumptions will follow.

The assumptions are as follows,

- The network will consist of a single source, destination and a potential helper node.
- The CoopTable information, with the single entry about the helper node, will be configured manually into the source node.
- The experiments will be conducted using the ad-hoc mode of the 802.11 MAC.
- There will be no acknowledgment sent from the destination to the source. (The issue of suppressing intermediate ACKs with the helper node is discussed later in greater detail.)
- The forwarding of data at the helper node will be done as an independent transmission as it is not possible to ensure forwarding within a SIFS interval without modifying the firmware in the 802.11 card. This assumption will have a negative impact on the performance improvement we measure in our experiments because of the increased contention time introduced by the helper transmissions.

The first three assumptions listed above were made to simplify the implementation process and are relatively easy to remove. We plan to implement dynamic learning about helper nodes in an infrastructure mode network with a collection of nodes and collect performance metrics to study the effectiveness of CoopMAC. The last two assumptions are due to the fundamental restriction we have from the card's firmware, and therefore in not being able to achieve transmissions on the medium within the pre-defined SIFS interval. We are investigating the possibility of acquiring an 802.11 setup that allows us to modify the firmware. In the absence of such a setup, we consider the possibility of implementing a CoopACK messages on top of the broadcast transmission scheme described below. This will require us to implement packet retries and ACK wait timers in the driver.

### B. Hardware and Software

Every station in our testbed is a Linux laptop with an 802.11b wireless card. The wireless card is based on the Intersil Prism 2.3 chip-set. All the stations share channel 3 (2.422GHz) of the 2.4 GHz band.

For the implementation of the protocol we use the HostAP driver [6]. HostAP is a Linux driver for wireless LAN cards based on Intersil's Prism 2/2.5/3 chipset. HostAP is a widely used driver for experiments and allows the support of Access Point (AP) functionality in the software. Since we plan to continue the development of this testbed to include a complete implementation of CoopMAC in infrastructure mode, HostAP is the current choice for the driver.

### C. Implementation Description

In order to have the flexibility to add enough information about the helper in the data frames, we defined a header that we call the CoopHeader to be inserted between the 802.11 header and the payload. The CoopHeader consists of three addresses: Destination Address, Source Address and Helper Address. We modified the behavior of the receive mode of the driver such that, in order to recognize the destination of the frame, it looks at the Destination Address of the CoopHeader instead of the destination address of the 802.11 header. In this way, during the two-hop transmission, a frame may change Source Address and Destination Address in its 802.11 header (depending on the transmission link) but keep the original information about the source, the destination and the helper stations in its CoopHeader.

One of the major difficulties in modifying 802.11 functionality for conducting experiments is that time critical tasks, like ACK transmission and the ability to transmit within SIFS time, are only possible by modifying the firmware of 802.11 cards. The first problem we faced was that we could not suppress the ACK frame sent by the helper node. As we mentioned, the cooperative protocol transmits the data in a two-hop manner and there is only one direct ACK that is generated from the destination to the source. Thus, we would like to suppress the ACKs that acknowledge the receptions from the source to the helper and from the helper to the destination. Unfortunately, this is not possible because the firmware automatically sends an ACK back to the sender after a SIFS interval.

In order to address this problem, we decided to use broadcast transmissions for the data frames since the firmware recognizes the broadcast nature of the transmission and does not generate an ACK. Thus we used two different kinds of data transmissions, one with 802.11 broadcast frames and another with 802.11 unicast frames (which are followed by ACKs). The problem we had in this case was that because the firmware controls the ACK transmissions, there is no way to generate a direct ACK from the destination to the source.

Thus we had to work with two modifications of the protocol:

- **Unicast transmissions for the two-hop data transmission:** In this case we have 2 ACKs, one for data reception at the helper and one the data reception at the final destination.
- **Broadcast transmissions for the two-hop data transmission:** In this way we suspend the ACKs for the two transmissions but we have no way to generate a direct ACK from the destination to the source. Thus, there is no acknowledgment in the whole procedure.

In both transmission scenarios, the CoopHeader is removed at the driver before the packet is sent to higher layers in the protocol stack. More accurately, in the transmission mode, we right shift the payload and put the CoopHeader just between the 802.11 header and the payload. As the HostAP cannot recognize the CoopHeader, it considers it as a part of the payload. In the reception mode, after the successful reception of a packet in the MAC, we pop the CoopHeader from the MAC packet before the sending of the packet to the higher layer, shift the payload left, and thus bringing the packet back to the initial form, ready to be processed by the higher layers.

Following is a brief description of the steps taken in the driver to implement CoopMAC:

**In the transmission path**: When the MAC receives a frame from the higher layer it puts it in a buffer ready to be sent. Then, we need to decide which helper, if any, will participate in the transmission. We implemented the CoopTable (table with potential helpers) in every station, but in this first version we manually configure the helper address. Once the helper is identified, we build a CoopHeader and insert it between the 802.11 MAC header and the payload. In the broadcast transmission case, we set the Destination address in the 802.11 MAC header to “ff:ff:ff:ff:ff:ff”. Finally, this frame is scheduled for transmission.

**In the reception path**: When a station receives a frame, if the destination address in the CoopHeader matches its own address and the source address in the 802.11 MAC header is the same as that of the helper address, then the frame is forwarded to the higher layer software within the station. If the destination address in the CoopHeader is different than its own address but the helper address in CoopHeader matches its own address, the data frame is forwarded to the final destination as a new transmission from the helper to the destination. In order to forward the frame to the final destination, the destination address in the 802.11 MAC header will be set to either the unicast address of the destination or the broadcast address (depending on the particular transmission mode). If neither the destination nor the helper address match the address of the station, the frame will be dropped.

We considered this complex design to support broadcast transmissions in order to suppress the ACK messages to approximate the specification of the CoopMAC protocol as closely as possible. As we will discuss later, the approach using broadcast transmissions resulted in an adverse effect on TCP performance and hence the final results from the experiments.

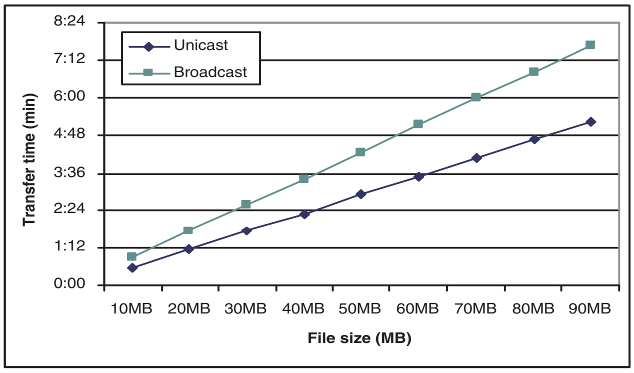

Fig. 3. Unicast vs broadcast transmissions scheme

## V. Experiments

In our experiments, we use the basic setup of three stations, with a source, a destination, and a helper. We ran different experiments changing the position of the helper between the different regions in Fig. 2 by appropriately forcing the data rates between the source, helper and destination nodes. For every position of the helper, large files were transferred from the source to the destination with and without cooperation from the helper node. For the file transmissions, two TCP based Unix file transfer applications, FTP and SCP were used. In this paper, we present the average transmission time from repeating each experiments 10 times.

The first experiment is the comparison between the two protocol schemes we described in the previous section: the unicast transmissions scheme and the broadcast transmissions scheme. The results for a helper with a transmission rate of 11MB in both link (source to helper, helper to destination) are illustrated in Fig. 3.

We expected to have higher throughput in the case of broadcast transmissions as in this case we did not have the added overhead from the ACK transmission. Nevertheless, as we can see in Fig. 3, the scheme with the unicast transmissions performs better. Studying the TCP window size in each experiment, we concluded that this is due to the fact that in the broadcast transmissions scheme, there is no acknowledgment for the MAC transmissions. Thus, the communication at the MAC layer is not reliable, resulting in lost frames and subsequent reduction of TCP window size. On the other hand, with unicast transmissions every transmission is acknowledged by an ACK, providing a reliable MAC layer communication. This leads to a better performance at the TCP layer, and thus faster transfer.

We must mention here that a full implementation of our protocol will perform even better than the unicast transmissions situation since there will be only one ACK directly from the destination to the source, and the forwarding at the helper will take place after a SIFS time without channel contention.

Our next experiment was to study the overhead that is added by the modifications we made to the HostAP driver. Even though it is reasonable to assume that the processing time introduced by the addition of the CoopHeader in the source and by the examination of this header at the destination

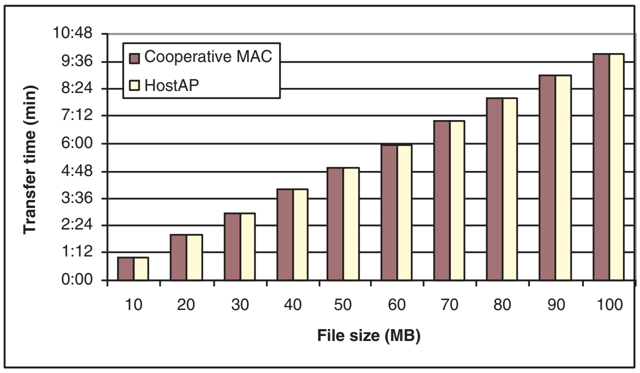

Fig. 4. HostAP driver vs Cooperative MAC driver

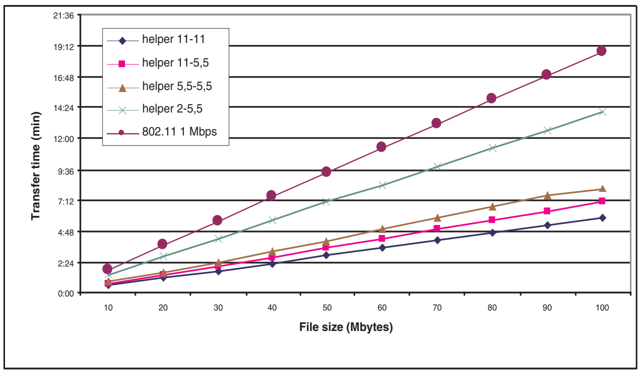

Fig. 6. Cooperative MAC vs 802.11 with an 1 Mbps direct link (for several helpers)

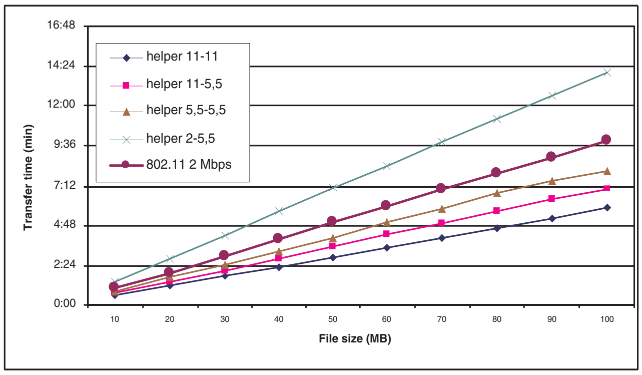

Fig. 5. Cooperative MAC vs 802.11 with a 2 Mbps direct link (for several helpers)

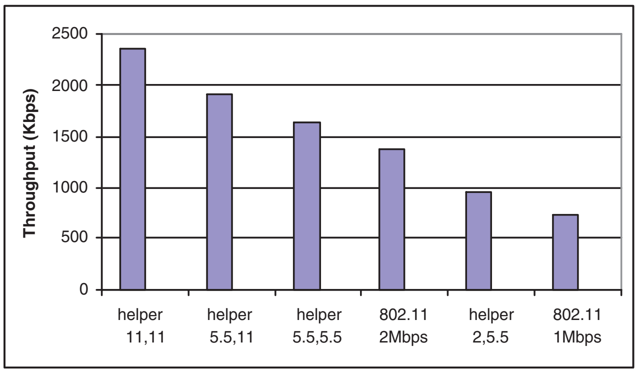

Fig. 7. End to end throughput for CoopMAC, for several helpers

will have a negligible effect on overall transmission times, we decided to verify that assumption by comparing the total file transfer time between two stations with and without our modifications. This experiment was conducted with direct transmissions from the source to destination.

The only difference between our driver and the HostAP driver in this case is the fact that our protocol changes the frame format, adding and removing the CoopHeader in the transmitter and the receiver respectively. The results of this experiment for transmission of files of different sizes with a 2 Mbps transmission rate in the direct link are shown is Fig. 4. As can be seen from the figure, the transfer time is identical, meaning that there is no overhead in the transmission procedure due to the addition of the cooperative MAC functionality in the hostAP driver. This result is due to the fact that frame sizes used for file transfer are usually large (1600 bytes) so that the addition of 18 bytes for CoopHeader has a negligible effect on the final performance. In the future, we plan to analyze the impact of these additional bytes on VoIP traffic, where a significantly smaller payload is transmitted [4], [7].

In the next experiment, the file transfer time for a source that transmits directly to the destination at a data rate of 2 Mbps was compared with the file transfer time if the same source received assistance from a helper node at various higher transmission rates. The results from this experiment is shown in Fig. 5. In this figure, *helper x-y* stands for a data rate *x* Mbps between the source and the helper and *y* Mbps between the helper and the destination. Potential positions of the *helper x-y* are in the area $A_{x,y}$ of Fig. 2.

As can be seen in Fig. 5, CoopMAC performs better, resulting in shorter transmission times for the files. The potential helpers in this case lie in the regions $A_{11,11}$, $A_{5.5,11}$, $A_{5.5,5.5}$ of Fig. 2.

We repeat the same experiment, with a data rate of 1 Mbps for direct transmission between the source and the destination. Results are shown in Fig. 6. The cooperative MAC results in shorter transfer times, even in the case of a helper that lies in the region $A_{2,5.5}$ of Fig. 2.

Finally, we calculate the throughput that is achieved at the application layer. This throughput was calculated by dividing the amount of data transferred each time by the transfer time. The results shown in Fig. 7 are the average values from the experiment. We must bear in mind that the calculated throughput is the application layer throughput after the TCP overhead and not the raw MAC layer throughput.

## VI. Conclusions and Future Work

In this paper we discuss implementation issues and present experimental results for a cooperative MAC protocol that was presented in our earlier work. The implemented protocol is based on a scheme under which a source station that experiences a poor quality channel with a specific destination chooses to use a helper station as a relay, sending its data in a two-hop manner instead of transmitting its data directly to the destination at a low rate. The cooperative scheme takes advantage of the fact that the helper is selected in a way such that the two links (source to helper and helper to destination) have high transmission rates. We describe the implementation procedure as well as the limitations placed on a designer due to the lack of control over time sensitive tasks performed by the wireless card's firmware. We set up a three node testbed and conducted experiments to measure the TCP performance using a file transfer program. The experimental results show that the protocol performs better than the standard single hop 802.11 MAC protocol. In our future work we plan to set up a larger testbed with many stations and to study the performance of the protocol in a more dynamic environment. This will include a full implementation of the CoopTable as described in [5].

## REFERENCES

[1] IEEE 802.11, Wireless LAN Medium Access Control (MAC) and Physical layer (PHY) Specifications, Standard, IEEE, Aug 1999.

[2] IEEE 802.11b, Part 11: Wireless LAN Medium Access Control (MAC) and Physical layer (PHY) specifications: High-Speed Physical layer Extension in the 2.4GHz Band, supplement to IEEE 802.11 Standard, Sept. 1999.

[3] Sathya Narayanan, Pei Liu, Shivendra Panwar, "On the advantages of multi-hop extensions to IEEE 802.11 infrastructure mode", *Proc. of IEEE WCNC'05*, March 2005.

[4] Sathya Narayanan, Shivendra Panwar, "When two-hop meets VoFi",*Proc. of IEEE CCNC'06*, January 2006.

[5] P. Liu, Z.Tao, S. Panwar, "A cooperative MAC protocol for wireless local area networks", *Proc. of IEEE ICC'05*, June 2005.

[6] HostAP: Wireless 802.11 driver, http://hostap.epitest.fi/

[7] T. M. Malathi Veeraraghavan, Nabeel Cocker. "Support of voice services in IEEE 802.11 wireless LANs", *Proc. of IEEE INFOCOM'01*, April 2001.

[8] Martin Heusse, Franck Rousseau, Gilles Berger-Sabbatel, Andrzej Duda, "Performance anomaly of 802.11b", *Proc. of IEEE INFOCOM'03*.

[9] Daji Qiao, Sunghyun Choi, Kang G. Shin, "Goodput analysis and link adaptation for IEEE 802.11a wireless LANs", *IEEE Transactions on Mobile Computing*, VOL.1, NO.4, October-December 2002.

[10] Giuseppe Bianchi, "Performance analysis of the IEEE 802.11 Distributed Coordination Function", *IEEE Journal on Selected Areas in Communications*, VOL.18, NO.3, March 2000.

[11] G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multi-hop wireless networks", Proc. ACM MobiCom'01, pp. 235-251, July 2001.