It Is Better to Give Than to Receive – Implications of Cooperation in a Real Environment

Thanasis Korakis[1], Zhifeng Tao[2], Salik Makda[1], Boris Gitelman[1],
and Shivendra Panwar[1]

[1] Department of Electrical and Computer Engineering, Polytechnic University,
Brooklyn, NY 11201
[2] Mitsubishi Electric Research Laboratories, Cambridge, MA 02139
korakis@duke.poly.edu, tao@merl.com, smakda01@utopia.poly.edu,
bgitel01@utopia.poly.edu, panwar@catt.poly.edu

Abstract. Thanks to the immense potential *cooperative* communications displays, extensive investigations have been directed to examine its performance by means of both analysis and simulation. In this paper[1], an implementation approach has for the *first* time been pursued to demonstrate the viability of realizing cooperation at the MAC layer in a real environment. The paper further describes the technical challenges encountered, details the corresponding solution proposed, and shares the experience gained. The experimental measurements in a medium size (i.e., 10 stations) testbed are then reported, which not only helps develop a deeper understanding of the protocol behavior, but also confirms that a cooperative MAC protocol delivers superior performance.

1 Introduction

Cooperative communications, which refer to the collaborative processing and retransmission of overheard information at stations surrounding a source, has recently gained momentum in the research community [1]. The notion of *cooperation* takes full advantage of the broadcast nature of the wireless channel and creates spatial diversity, thereby achieving improvement in system robustness, capacity, delay, coverage range, and interference reduction. The innovation of cooperative communications is not confined only to the physical layer. It is available in various forms at higher protocol layers [2]. A MAC protocol called CoopMAC [3] illustrates how the legacy IEEE 802.11 distributed coordination function (DCF) can be enhanced with minimal modifications to maximize the benefit of cooperative diversity.

As experimentation gradually becomes one of the de facto approaches for benchmarking [4] [5], a preliminary performance evaluation for cooperative MAC protocol was attempted in [6] in a relatively rudimentary experimental setting, for proof of concept purposes. Only 3 stations with dedicated roles as source, destination and helper

---

[1] This work is supported in part by National Science Foundation (NSF) under award 0520054, and the New York State Center for Advanced Technology in Telecommunications (CATT). The work is also supported by Wireless Internet Center for Advanced Technology (WICAT), an NSF Industry/University Research Center at Polytechnic University.

were involved therein, and *throughput* for *only one* TCP session was collected. In this paper, the scope and scale of effort have been significantly expanded, where all major protocol functionalities are implemented in an open source driver for 802.11 devices, and a comprehensive set of experiments are conducted in a testbed consisting of up to 10 stations. As highlighted below, many different aspects of the protocol performance have been scrutinized, and new results obtained thereby reported.

– Delay performance (e.g., average end-to-end delay, jitter)
– Impact of cooperation on the helper station
– Impact of the "Hello Packet" interval
– Impact of buffer overflow on system performance

To familiarize the reader with CoopMAC, the basic idea of the protocol is first summarized in Section 2. The implementation effort is then elaborated in Section 3, and the primary configurations of the experiments specified in Section 4. A rich set of measurement results along with the insights revealed therein are reported in Section 5. Section 6 completes the paper with final conclusions and possible future work. For ease of explanation, the term *relay* and *helper* will be used interchangeably in the following discussion.

2   Cooperation at MAC Layer

In order to deliver an acceptable frame error rate (FER), packets in IEEE 802.11 can be transmitted at different bit rates, which are adaptive to the channel quality. For IEEE 802.11b, in particular, four different rates are supported over the corresponding typical ranges, as depicted in Figure 1.
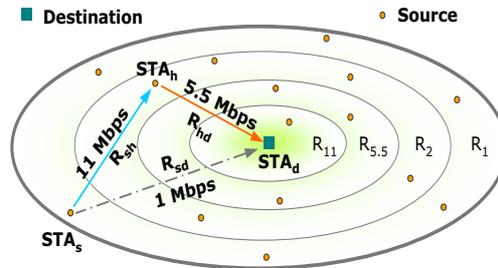

Fig. 1. Illustration of the Cooperative MAC Protocol

One key observation conveyed by Figure 1 is that a source station STA_s that is far away from the destination STA_d may persistently experience a poor wireless channel, resulting in a rate as low as R_{sd} (e.g., $1Mbps$) for direct transmission over an extended period of time. If there exists some neighbor STA_h who in the meantime can sustain higher transmission rates R_{sh} and R_{hd} (e.g., $11Mbps$ and $5.5Mbps$ in Figure 1) between STA_h and STA_s, and between STA_h and STA_d, respectively, station STA_s can enlist the neighbor STA_h to *cooperate* and forward the traffic on its behalf to the

destination, yielding a much higher effective rate. More specifically, upon the transmission of a packet, station STA_s should access all the rate information in a *cooperation table* (a.k.a. *CoopTable*), and compare an estimation of the equivalent two-hop rate $(R_{sh}R_{hd})/(R_{sh} + R_{hd})$ with the direct rate R_{sd} to determine whether the two-hop communication via the relay yields a better performance than a direct transmission. If cooperative forwarding is invoked, CoopMAC engages the selected relay station STA_h to receive the traffic from the source STA_s at rate R_{sh} and then forward it to the corresponding destination STA_d at rate R_{hd} after a short interframe spacing (SIFS) time. In the end, destination STA_d indicates its successful reception of the packet by issuing an acknowledgment packet (i.e., ACK) directly back to STA_s. As an option, the RTS/CTS signaling defined in IEEE 802.11 can be extended to a 3-way handshake in CoopMAC to further facilitate the ensuing cooperative data exchange.

To identify station that has been selected as a helper, the *Address 4* field in the MAC header of data packet from STA_s to STA_h in CoopMAC should hold the MAC address of the final destination STA_d, while the *Address 1* field contains the MAC address of the selected helper STA_h. When the packet is further forwarded by STA_h to STA_d, the helper will place the address of STA_d in field *Address 1*, and leave the *Address 4* unused.

The key enhancement in the control plane at each station is the establishment and maintenance of the CoopTable, which contains essential information (e.g., R_{hd}, R_{sh}) related to all the potential helpers. For STA_s to acquire the value of R_{hd} and R_{sh}, a passive eavesdropping approach is followed, so that the overhead of additional control message exchange can be kept at minimum level.

It is worthwhile to note that although CoopMAC seemingly bears some resemblance to the ad hoc routing protocols that adopt either minimal hop or other innovative metric for path selection [7], they are in essence fundamentally different. First and foremost, the objective of CoopMAC is to exploit spatial diversity and rate adaptation, not to increase the geographical extent of the network as in ad hoc routing. Secondly, all the associated operations occur in the MAC layer, which enjoys a shorter response time and more convenient access to the physical layer information, as compared to traditional network layer routing. Interested audiences are encouraged to refer to [3] for more detailed protocol specifications and technical discussions.

3   Implementation of Cooperation

Key challenges encountered in the driver implementation and the corresponding solution will be summarized in this section. Nevertheless, due to the constraints of space, certain implementation details cannot be covered. Interested readers can access the official project website [8] for more technical information and free downloading of the CoopMAC driver.

3.1   Inaccessibility to Firmware

When it comes to system design, all the features specified in IEEE 802.11 MAC protocol are logically partitioned into two modules, according to the time-criticality of each task. The lower module, which usually operates on the wireless card as a part

of firmware, fulfills the time-critical functions such as the generation and exchange of RTS/CTS control messages, transmission of acknowledgment (ACK) packets, execution of random backoff, etc. The other module, which normally assumes the form of the system driver, is responsible for more delay-tolerant control plane functions such as the management of MAC layer queue(s), the formation of the MAC layer header, fragmentation, association, etc.

As the cooperative MAC protocol requires changes to both the time-critical and delay-tolerant logic, the inaccessibility to firmware unfortunately causes additional complexity in implementation. Indeed, compromises had to be made and alternative approaches had to be pursued, due to this constraint. For illustrative purposes, three main circumventions that become necessary are outlined below.

- *Suspension of 3-way Handshake*
  As mentioned in Section 2, a 3-way handshake option has been defined in the cooperative MAC protocol, which requires the selected helper to transmit a new control message called "Helper ready To Send" (HTS) between the RTS and CTS messages. Since the strict sequence of RTS and CTS packet has been hardwired in the firmware, an insertion of HTS becomes impossible at the driver level. It was therefore not possible to implement this option.
- *Unnecessary Channel Contention for Relayed Packet*
  Once the channel access has been allocated to the source station, the helper should relay the packet a $SIFS$ time after its reception, without any additional channel contention. Since the $SIFS$ time is set as $10\mu s$ in IEEE 802.11b, any function demanding such a short delay must be implemented in the firmware. As a result, a compromise has been made in the implementation, where channel contention for the relayed packet on the second hop has to be attempted.
- *Duplicate ACK*
  Each successful data exchange in the original cooperative MAC protocol involves only one acknowledgment message, which is sent from the destination to the source directly. Since the acknowledgment mechanism is an integral function of firmware, it is impossible to suppress the unnecessary ACK message generated by the relay station for the packet it will forward on behalf of the source. Therefore, an unwanted ACK from the relay had to be tolerated.

As an implication of the circumventions described above, a faithful implementation of cooperative MAC is anticipated to outperform the one demonstrated in this paper.

3.2   Maintenance of the CoopTable

As described in Section 2, the *CoopTable* is critical to enabling the cooperative operation. The passive approach for rate learning defined in the original CoopMAC protocol [3], however, has not been realized in our implementation due to the following reasons:

- *Unwanted Packet Filtering*
  All the packets with a destination address different from the local MAC address are filtered out by the firmware. Hence, the driver is unable to retrieve any rate information from them.

– *Controllability of the Experiment Environment*
  Even if the driver had access to such packets (e.g., by periodically switching the wireless card to the promiscuous mode), the additional random delay incurred by frequent mode switches, and traffic load and pattern at each station may complicate the collection of data in the experiments.

Therefore, for sake of controllability of the experimental environment, an active information distribution approach has been followed instead. More specifically, a *Hello* packet is broadcast by each station in a periodic manner, to notify the neighbors about its existence as well as the sustainable transmission rate on the respective link. Upon the reception of the *Hello* packet, a station either inserts a new entry or updates an existing one in its CoopTable.

4 Experiment Configuration and Measurement Methodology

4.1 Testbed Configuration

The testbed used in the experiment consists of 10 IBM T23 laptops, each of which contains an Intel Pentium III processor of 800 MHz and 384 MB memory. Redhat Linux 9.0 with kernel version 2.4.20 is installed as the operating system. In the ensuing experimental study, three different network topologies will be used. In each topology, one station is a dedicated destination, which mimics the functionality of an access point. The rest of the stations are either traffic sources, or helpers or both. To calibrate the testbed, the positions of stations have been adjusted until the throughputs achieved by all stations are roughly equal.

4.2 Measurement Methodology

The majority of the statistics generated in the experiment, including throughput, packet loss and jitter, are measured by using *Iperf* [9], which is a powerful tool for traffic generation and measurement. A typical experimental setup could be to run an *Iperf* client at a handful of stations to generate UDP or TCP traffic streams, while an *Iperf* server residing on the dedicated destination receives the traffic and collects the statistics. To remove any random effect and short-term fluctuation, we run each experiment 5 times and each run lasts 10 minutes. Then, we get the average results.

The measurement of average delay was non-trivial, since no mean end-to-end delay statistics are provided by *Iperf* or other off-the-shelf traffic measurement tools. As further explained in [5], tight synchronization between the transmitter and receiver is mandated, if the delay is to be measured directly.

4.3 Baseline Scenario

To circumvent the synchronization requirement, which is notoriously difficult to meet, the end-to-end delay is therefore derived based upon a round trip delay that can be measured more easily. More specifically, a new testing function has been implemented in the driver, which lets the transmitter periodically broadcast a packet. Once the receiver

successfully decodes the packet, it immediately sends another broadcast packet back to the transmitter. Since the delay incurred in each direction can be considered to be identical, the one-way end-to-end delay experienced by a data packet is approximately equal to half of the round-trip delay observed at the transmitter. The delay statistics derived thereof essentially is the time from the moment that the wireless MAC driver pushes the packet into the MAC transmission queue, until the time the packet is passed from the physical layer to the MAC buffer at the receiver. A closer examination of this delay value reveals that it consists of several major components, namely the delay incurred at the transmitter (e.g., kernel interrupt delay in the driver, random backoff time, DIFS), transmission time, and delay experienced at the receiver (e.g., delay associated with kernel interrupt that signals to the MAC layer the arrival of a new packet, etc.). Note that no time will be spent on transmitting an ACK packet, because a broadcast transmission does not require any acknowledgment.

5   Performance Evaluation

Based upon the testbed described in Section 4, numerous experiments have been conducted, and the results obtained are reported and analyzed in this section.

A baseline scenario, which only consists of 1 transmitter, 1 helper and 1 receiver, is first used to develop a basic understanding of the implication of cooperation, and establish a benchmark for performance study of more sophisticated settings. Thanks to its simplicity, this scenario isolates interfering factors such as collisions, and creates an ideal environment that gives rise to several crucial insights related to the behavior of CoopMAC.

Throughput Improvement. In this experiment, source station STA_s generates traffic using an *Iperf* client, while the corresponding *Iperf* server running at the destination STA_d collects the end-to-end throughput statistics. The *Iperf* client at STA_h has been switched on, so that it not only relays traffic on behalf of STA_s, but also transmit its own packets to STA_d.

As readily demonstrated in Figure 2, CoopMAC enables STA_s to deliver substantially higher throughput. More importantly, Figure 2 confirms that CoopMAC protocol creates a *win-win* situation, instead of a zero-sum game. That is, STA_h benefits by helping forward the packets for the slow source station. At first glance counterintuitive, this observation can be explained by the fact that if STA_h participates in forwarding, STA_s can finish its packet transmission much earlier, thereby enabling both STA_s *and* STA_h to transmit more bits in unit time.

Interaction with Transport Protocol. In Figure 2, we can see the throughput comparison in a scenario of a source, an active helper and a destination. Direct transmission between source station STA_s and destination STA_d always occurs at 1 Mbps, and helper station STA_h can sustain 11 Mbps for communication with both STA_s and STA_d. An important trend displayed in Figure 2(a) is that bandwidth in the IEEE 802.11 network is equally shared by the two UDP sources STA_s and STA_h, respectively, in spite of the fact that physical layer bit rate supported by STA_h is over 10 times higher than that at STA_s. Indeed, this notion of fairness that 802.11 strives to maintain has been known

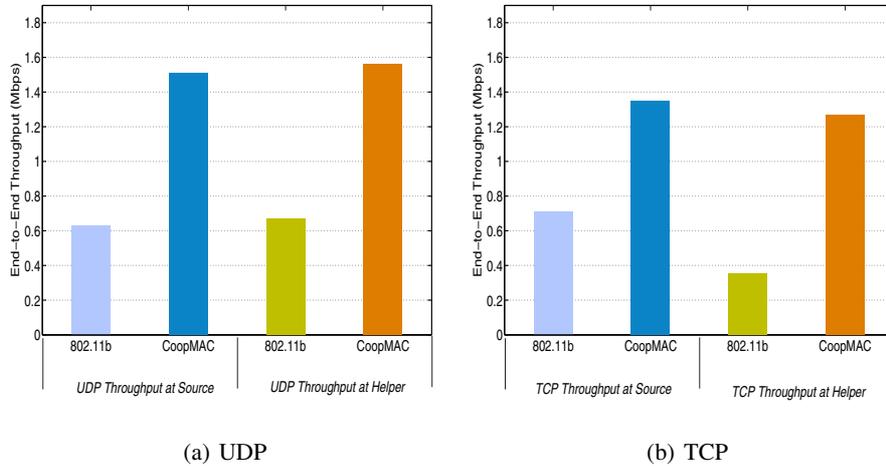(a) UDP   (b) TCP

Fig. 2. Throughput Comparison: Active Traffic from Helper

to be at the cost of serious network-wide throughput degradation [10]. The CoopMAC protocol preserves this fairness; no significant disparity in the throughput of STA_h and STA_s can be seen in Figure 2(a).

For TCP traffic in the 802.11 network, however, Figure 2(b) indicates that the slow source station STA_s surprisingly grabs even more bandwidth than the fast helper station STA_h, which seems to defy conventional wisdom.

It has long been known that the cross-layer interaction between the random access wireless MAC protocol and TCP congestion control mechanism is problematic [11]. We will conduct further investigation regarding the cause of this counter-intuitive phenomenon.
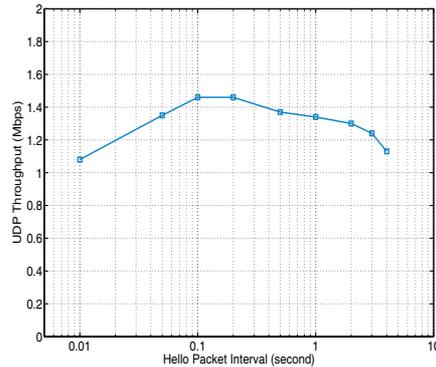
5.1   Hello Packet Interval

It is known that the frequency at which the *Hello* packet is broadcast exerts crucial influence on the system performance. A new experimental scenario that contains 1 source, 2 helpers and 1 destination has been setup to investigate this impact. Packets are only generated at source station STA_s in this experiment, and the rates supported on all related links are listed in Table 1. The second relay STA_{h2} remains available all the time, while the first one STA_{h1} alternates between *awake* and *dormant* state every *15* seconds to mimic user mobility and dynamic channel conditions. Note that since relay STA_{h1} maintains fast links to both the source and destination, it will be chosen as the helper as long as the source thinks that STA_{h1} is still located in close physical proximity. Of course, if the *Hello* packets from STA_{h1} disappear after it becomes dormant, STA_s eventually would realize that STA_{h1} is unavailable, and therefore turns to STA_{h2} for help.

The *Hello* packet interval is varied in the experiment, and the resultant UDP throughput is collected and plotted in Figure 3. A small value of this interval lets the source

Table 1. Settings for Study of Hello Packet Interval

R_{sd}	R_{s_h1}	R_{h1_d}	R_{s_h2}	R_{h2_d}
1 Mbps	11 Mbps	11 Mbps	11 Mbps	5.5 Mbps

Fig. 3. Impact of Hello Packet Interval

STA_s be constantly updated on the current state of relay STA_{h1}, but causes more overhead. On the other, overhead can be reduced, but the information about the status of STA_{h1} may become stale at the source, as the interval grows excessively large. When the interval falls between the range of 0.1 to 0.2 seconds, a balance can be struck and the maximum throughput can be achieved, given that STA_{h1} goes off every 15 seconds. However, a general optimal operating region of the *Hello* interval value is far more complicated to predict, as the availability and suitability of a relay in reality depend on such highly random factors as channel fading, mobility and usage pattern.

5.2   End-to-End Delay

Another key dimension of performance for any MAC protocol is the delay, which in fact plays a more critical role than throughput in determining the network's capability of supporting QoS-sensitive applications.

The scenario configured to measure the average end-to-end delay has been summarized in Table 2. The delay measurement methodology described in Section 4.2 has been applied, and the average delay is obtained based upon the experimental results for over 10^6 broadcast packets.

As portrayed in Figure 4, it is evident that cooperative forwarding significantly lowers the average delay for all the cases studied, provided the MSDU size is larger than about 200 bytes. But once the MSDU size drops below 200 bytes, IEEE 802.11b seems to perform better, since it avoids the overhead associated with CoopMAC. Nonetheless, note that this drawback can be avoided, if CoopMAC adopts a dynamic relay selection algorithm, in which the source STA_s would simply fall back to legacy 802.11 for small frames.

Table 2. Settings for the Study on End-to-End Delay

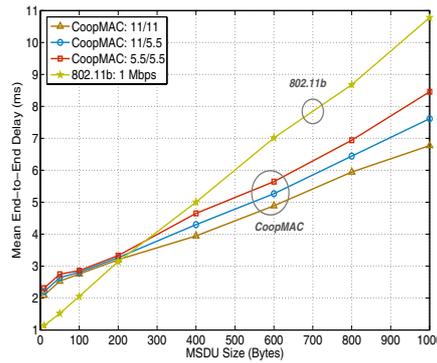| Case	R_{sd}	R_{sh}	R_{hd}
1	1 Mbps	11 Mbps	11 Mbps
2	1 Mbps	11 Mbps	5.5 Mbps
3	1 Mbps	5.5 Mbps	5.5 Mbps

Fig. 4. Mean End-to-End Delay

5.3   Protocol Dynamics

To study the dynamic behavior of the protocol, a medium size testbed has been constructed, where 4 sources, 4 helpers and 1 dedicated destination are involved in the experiment. The UDP traffic is originated from both the source and the helper station, which implies that the channel access opportunities seized by each helper somehow have to be shared by both the locally generated traffic and the forwarded traffic. Table 3 lists the rate information related to the experiment.

For both 802.11 and CoopMAC network, Figure 5 illustrates how the throughput achieved by each station changes with respect to the load applied.

1. *Saturation Point*
   The 802.11 network passes the critical tipping point as early as $0.2 Mbps/station$, while CoopMAC does not experience saturation until a load of $0.5 Mbps/station$. Thus, the maximum throughput achieved by CoopMAC is approximately 2.5 times higher than that for 802.11.

2. *Post-Saturation Regime*
   Once entering their respective saturation regions, all stations in 802.11 invariably start to witness significant packet drop and throughput deterioration. For helper stations in cooperative MAC, however, the decrease ends after an initial dip, and then stabilizes at a plateau of about $0.28 Mbps/station$. The throughput of source stations in CoopMAC more or less follows the same trend of monotonic decline as observed in 802.11, but its absolute value is still notably higher.

Table 3. Settings for Study of Network Dynamics

$R_{s_i d}, \forall i \in [1,4]$	$R_{s_i h_j}, \forall i,j \in [1,4]$	$R_{h_i d}, \forall j \in [1,4]$
1 Mbps	11 Mbps	11 Mbps

(a) 802.11   (b) CoopMAC
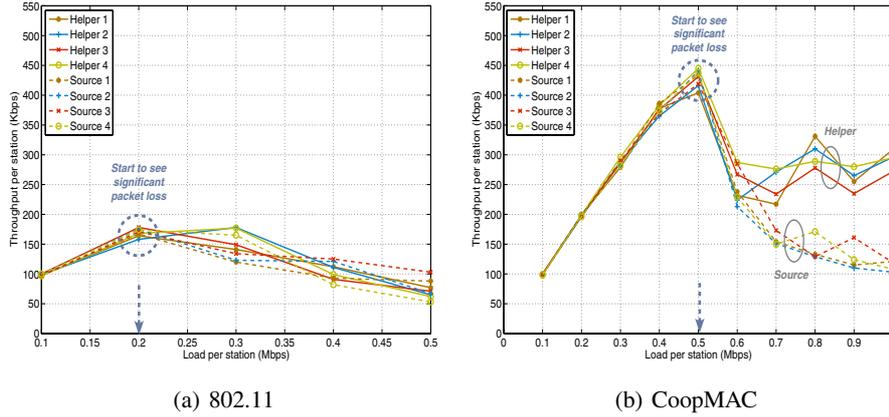
Fig. 5. Throughput Comparison

A closer scrutiny further suggests that this performance disparity between the helper stations and source stations in a CoopMAC network is an artifact resulting from our present implementation approach, and is expected to disappear once the access to firmware becomes available. More specifically, as explained in Section 3, the cooperative MAC protocol is currently realized at the driver level, which forces the helper stations to pass the received foreign packets into the *driver space* and queue them together with the native traffic in the *same* buffer. When the local load at the helpers grows high enough, the arrival rate of the indigenous packets at the buffer far surpasses that of the packets received from the source stations. Therefore, the rate at which the packets can be received at the helpers places a bottleneck on the end-to-end throughput of the forwarded traffic, which essentially gives local helper traffic preferential treatment.

5.4   Jitter

To gain a more profound view of the delay performance, the jitter statistics for UDP traffic are collected and depicted in Figure 6. Direct transmission between source stations STA_s and destination STA_d always occur at 1 Mbps, and helper stations STA_h can sustain 11 Mbps for communication with both STA_s and STA_d.

Figure 6 depicts the maximum jitter for source and helper station, which is collected by *Iperf* for each traffic stream. Both Figures 6(a) and 6(b) indicate that jitter is sensitive to network size. Moreover, although helper stations support higher transmission rates than source stations, they suffer higher variance in end-to-end delay (jitter) in an 802.11
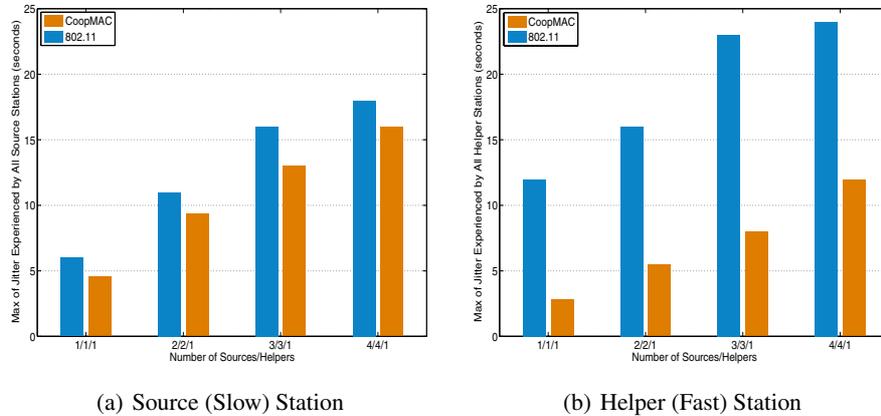
(a) Source (Slow) Station

(b) Helper (Fast) Station

Fig. 6. Jitter Comparison

network. A similar phenomenon was previously identified and an explanation offered in Section 5, where the interaction with the TCP layer was first investigated.

Once cooperative MAC is adopted, the jitter performance for both source and helper stations can be improved. In addition, the fast helper stations now perceive lower jitter than the slow source stations, implying that the issue of unfairly high jitter for fast stations has been successfully resolved by CoopMAC.

6   Conclusions and Future Work

This paper represents one of the first attempts that relies on an experimental approach to develop an understanding of cooperation at the MAC layer. The measurement results obtained confirm that cooperative MAC can substantially improve the performance (e.g., throughput, mean end-to-end delay, jitter) for not only the stations being helped, but also the ones who offer the cooperation. Furthermore, the paper sheds light on several critical issues particular to cooperation, such as the impact of MAC cooperation on the TCP protocol, the dynamics of protocol behavior, etc., which to the best of our knowledge have been presented for the first time.

As for possible future work, user mobility would be incorporated into the experiment to examine its impact on the protocol performance. It is also worthwhile to develop further understanding about the implication of cooperation on power consumption, as energy efficiency is always a major design constraint for mobile devices. Moreover, investigation of the possible interference reduction effect of cooperative MAC would be attempted in a larger testbed that can emulate a multicell environment. In addition, access to the firmware codebase will be actively pursued, so that all the artifacts and constraints imposed by the current driver can be mitigated or completely eliminated, and the entirety of the cooperative protocol can finally be implemented in a faithful manner.

References

1. J. N. Laneman, D. Tse, and G. W. Wornell, "Cooperative Diversity in Wireless Networks: Efficient Protocols and outage Behavior," *IEEE Transactions on Information Theory*, vol. 50, pp. 3062–3080, December 2004.

2. G. Jakllari, S. V. Krishnamurthy, M. Faloutsos, P. V. Krishnamurthy, and O. Ercetin, "A Framework for Distributed Spatio-Temporal Communications in Mobile Ad hoc Networks," in *Proceedings of IEEE INFOCOM'06*, (Barcelona, Spain), April 2006.

3. P. Liu, Z. Tao, and S. Panwar, "A Cooperative MAC Protocol for Wireless Local Area Networks," in *Proceedings of IEEE ICC'05*, (Seoul, Korea), June 2005.

4. A. Raniwala and T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network," in *Proceedings of IEEE INFOCOM'05*, (Miami, FL), March 2005.

5. I. Dangerfield, D. Malone, and D. Leith, "Experimental Evaluation of 802.11e EDCA for Enhanced Voice over WLAN Performance," in *Proceedings of Second International Workshop On Wireless Network Measurement (WiNMee 2006)*, (Boston, USA), April 2006.

6. T. Korakis, S. Narayanan, A. Bagri, and S. Panwar, "Implementing a Cooperative MAC Protocol for Wireless LANs," in *Proceedings of IEEE ICC'06*, (Istanbul, Turkey), June 2006.

7. S. Biswas and R. Morris, "Opportunistic Routing in Multi-Hop Wireless Networks," in *Proceedings of ACM SIGCOMM'05*, (Philadelphia, PA), August 2005.

8. "Official Website of Cooperative MAC Implementation," `http://eeweb.poly.edu/coopmac/`.

9. "Iperf: The TCP/UDP Bandwidth Measurement Tool," `http://dast.nlanr.net/Projects/Iperf/`.

10. M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance Anomaly of 802.11b," in *Proceedings of IEEE INFOCOM'03*, (San Francisco, CA), April 2003.

11. S. Xu and T. Saadawi, "Revealing the Problems with 802.11 Medium Access Control Protocol in Multi-hop Wireless Ad Hoc Networks," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 38, pp. 531 – 548, March 2002.