Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2009, Article ID 598140, 19 pages
doi:10.1155/2009/598140

Research Article

Implementation of a Cooperative MAC Protocol: Performance and Challenges in a Real Environment

Thanasis Korakis,[1] Zhifeng Tao,[2] Shashi Raj Singh,[1] Pei Liu,[1] and Shivendra S. Panwar[1]

[1] *Department of Electrical and Computer Engineering, Polytechnic Institute of NYU, 6 Metrotech Center, Brooklyn, NY 11201, USA*
[2] *Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Cambridge, MA 02139, USA*

Correspondence should be addressed to Thanasis Korakis, korakis@poly.edu

Received 31 October 2008; Accepted 31 March 2009

Recommended by Xavier Mestre

Cooperative communication is an active area of research today. It enables nodes to achieve spatial diversity, thereby achieving tremendous improvement in system capacity and delay. Due to its immense potential, extensive investigations have been directed to closely examine its performance by means of both analysis and simulation. However, the study of this new technology in an implementation-based system is very limited. In this paper, we present two implementation approaches to demonstrate the viability of realizing cooperation at the MAC layer in a real environment. The paper describes the technical challenges encountered in each of the approaches, details the corresponding solution proposed, and compares the limitations and benefits of the two approaches. Experimental measurements are reported, which not only help develop a deeper understanding of the protocol behavior but also confirm that cooperative communication is a promising technology for boosting the performance of next-generation wireless networks.

1. Introduction

Today, wireless devices are evolving into multipurpose systems with data extensive applications running on them. Such applications require high-speed connectivity and strong error protection. Those needs, along with the exploding growth of wireless networks and limited spectrum resources, have created a capacity crunch and high interference in today's wireless networks. This situation entails a move toward the development of new wireless techniques that can achieve a more efficient use of the avaliable spectrum. While emerging techniques such as multi-input multi-output systems (MIMO) increase the spectrum efficiency in terms of the number of bits per hertz of bandwidth, its usage is limited because of the size, cost, and power constraints posed by portable wireless devices. An alternative approach called *cooperative communications* [1–3] promises to deliver some of the benefits of MIMO within the given constraints.

Cooperative communication refers to the collaborative processing and retransmission of the overheard information at those stations surrounding the source. The notion of *cooperation* takes full advantage of the broadcast nature of the wireless channel and creates spatial diversity, in particular, transmission diversity, thereby achieving tremendous improvements in system robustness, capacity, delay, interference, and coverage range.

The fundamentals of cooperative communications lie in the physical layer. However, the notion of cooperation is available in various forms at different protocol layers. To facilitate access to the physical layer information and adaptation to mobility, it is natural to introduce the notion of cooperation in the layer directly above the PHY, namely, the medium access control (MAC) layer.

In this paper, we present the implementation of a widely discussed [4–7] cooperative MAC protocol called CoopMAC [8]. The implementation follows two different approaches, one based on an open-source driver for 802.11 devices (which we call the *driver approach*) and the other based on a software defined radio (SDR) platform (which we call the *SDR approach*). By conducting a comprehensive set of experiments in medium-size testbeds, we study the performance of each approach based on the protocol aspects

as follows:

(i) throughput performance under heavy load,

(ii) link performance (e.g., PER),

(iii) delay performance (e.g., average end-to-end delay, jitter, processing and transmission delay),

(iv) impact of cooperation on the *helper station* that helps the source transmit to the destination,

(v) impact of the *Hello Packet* interval (the functionality of this packet will be defined later),

(vi) impact of buffer overflow on system performance,

(vii) impact of cooperative MAC on real-time (video) applications.

These results help in developing a deeper understanding of the protocol behavior and also confirm that the cooperative MAC protocol delivers superior performance as compared to a legacy (802.11) MAC protocol. Equally important, this paper also elaborates the technical challenges encountered in each of the approaches, details the corresponding solution proposed, compares the limitations posed by the approaches and their benefits, and shares the experience gained, thereby exemplifying how implementation of a cooperative protocol can be approached.

Note that given the nascent nature of cooperative communications, its performance evaluation by means of implementation and experimentation has been scarcely discussed or treated so far. Thus, to the best knowledge of the authors, the work presented in this paper represents one of the *first* attempts to develop and further advance the understanding of *cooperative communication protocols* in a real environment.

To familiarize the reader with the necessary background, Section 2 briefly introduces cooperative communications and discusses the recent experimentation efforts in related fields. The protocol that we selected for implementation, namely, CoopMAC, is summarized in Section 3. Then we discuss the driver testbed in Section 4 and the implementation efforts in Section 5. A rich set of measurement results from the driver testbed along with the insights revealed therein are reported in Section 6. Section 7 describes the limitations of the driver approach and introduces the SDR testbed. Section 8 details the implementation on the SDR platform. The results of the SDR implementation approach and the insights we derived are provided in Section 9. Section 10 completes the paper with final conclusions and possible future work.

2. Background and Related Work

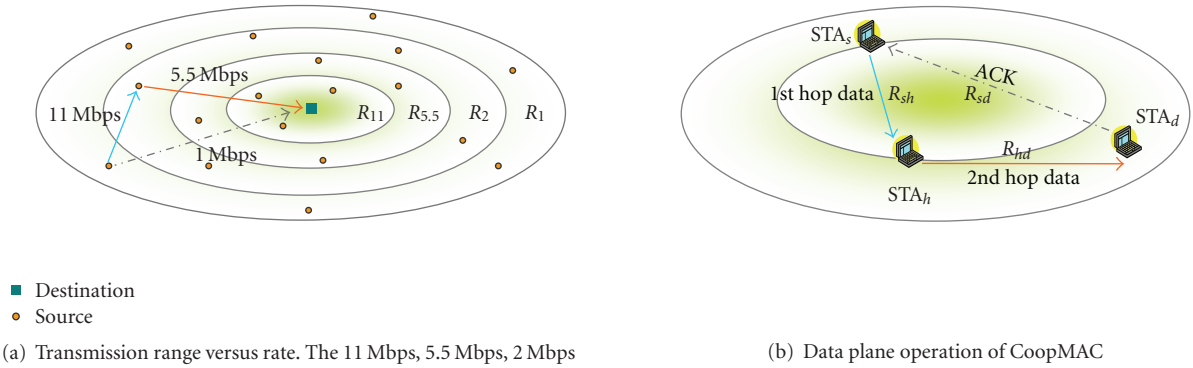
The initial attempts for developing cooperative communications focused on physical (PHY) layer schemes [1–3]. These approaches refer to the collaborative processing and retransmission of the overheard information at those stations surrounding the source and the destination. By combining different versions of the same information transmitted by source and different relay stations, the destination can improve its ability to decode the original packet.

However, albeit highly promising, cooperation at the physical layer encounters several formidable obstacles when system realization is considered. First and foremost, joint decoding at the receiver is plausible only if an accurate synchronization can be maintained among all the stations involved in the communication, which is notoriously difficult to cope with in reality. Secondly, the cooperative coding scheme is significantly different from the conventional ones implemented in commercial wireless products (e.g., IEEE 802.11), so that it demands a total redesign of physical layer hardware, which is yet another daunting undertaking.

Numerous efforts [7–10] have also been reported on designing new MAC layer protocols that take advantage of spatial diversity and support cooperative schemes in the PHY layer. For instance, *CoopMAC* proposed in [8] allows a source station to choose a relay, based on the information collected passively by listening to the transmissions in the neighborhood. The *rDCF* protocol described in [9] follows a more active approach by advertising the ability of each relay to help by using "Hello packets." but for both CoopMAC and rDCF, the source station transmits the packet to the relay and the relay forwards the packet to the destination immediately after the reception. Meanwhile, the relay station in [7] forward the packet only if it does not receive an ACK from the destination that indicates that the destination has failed to decode the packet after the first hop transmission. Persistent RCSMA, described in [10], allows executing a distributed and cooperative automatic retransmission request (ARQ) scheme in wireless networks. These schemes exploit the broadcast nature of the wireless channel in the following manner; once a destination station receives a data packet containing errors, it can request a set of retransmissions from any of the relays which overheard the original transmission.

Thanks to the commoditization of IEEE 802.11 devices (e.g., network interface card (NIC) and access point) and the availability of various open source device drivers [11–13], software-defined radio testbeds [14, 15] and free wireless measurement/testing tools [16], prototyping and experimentation have become a feasible complement to theoretical research in the communication and networking community in recent years [17–22]. However, performance evaluation for all the cooperative MAC protocols [7, 9, 23] at this moment is solely based on simulation and analysis. Since implementation and field experimentation remain the ultimate test of the performance of a new protocol, we are motivated to pursue an implementation approach in this paper.

Among all these MAC protocols, we have chosen *CoopMAC* to implement, because it is one of the first MAC protocols that fully exploits cooperative diversity and has been widely discussed and referenced [4–7]. Moreover, CoopMAC maintains backward compatibility with the legacy IEEE 802.11 distributed coordination function (DCF) [24] and incurs a negligible additional signaling overhead, thereby requiring only minor modifications of the standard, and presenting an opportunity for incremental implementation of cooperation on commercial 802.11 platforms. Nevertheless, note that the experience reported in this study is equally



(a) Transmission range versus rate. The 11 Mbps, 5.5 Mbps, 2 Mbps and 1 Mbps regions are denoted by R_{11}, $R_{5.5}$, R_2, and R_1, respectively

(b) Data plane operation of CoopMAC

Figure 1: Cooperation at MAC layer.

applicable for the development of other cooperative MAC schemes that rely on a single relay for forwarding.

Implementation of CoopMAC can be built upon two possible platforms, namely, *open-source driver* [11–13] and *software defined radio* [14, 15] . Both platforms have their own benefits as well as drawbacks as far as CoopMAC implementation is concerned. In order to be able to conduct extensive studies on the cooperative MAC protocol in a real environment and realize its full potential, we decided to pursue both approaches.

3. Cooperation at MAC Layer

3.1. Multirate Capability and Motivation for Cooperation. Before delving into the protocol details of CoopMAC, the motivation for cooperation and the multi-rate capability of IEEE 802.11b deserve a brief discussion, as they are crucial to comprehending how cooperation at the MAC layer can be capitalized on.

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. In general, the transmission rate is essentially determined by the path loss and instantaneous channel fading conditions. For IEEE 802.11b in particular, four different rates are supported over the corresponding ranges, as depicted in Figure 1(a).

Another key observation conveyed by Figure 1(a) is that a source station that is far away from the destination may persistently experience a poor wireless channel, resulting in a rate as low as 1 Mbps for direct transmission over an extended period of time. If there exists some neighbor who in the meantime can sustain higher transmission rates (e.g., 11 Mbps and 5.5 Mbps in Figure 1(a)) between itself and both the source and the intended destination, the source station can enlist the neighbor to *cooperate* and forward the traffic on its behalf to the destination, yielding a much higher equivalent rate. With the simple participation of a neighboring station in the cooperative forwarding, the aggregate network performance would witness a dramatic improvement, which justifies and motivates the introduction of cooperation into the MAC layer.

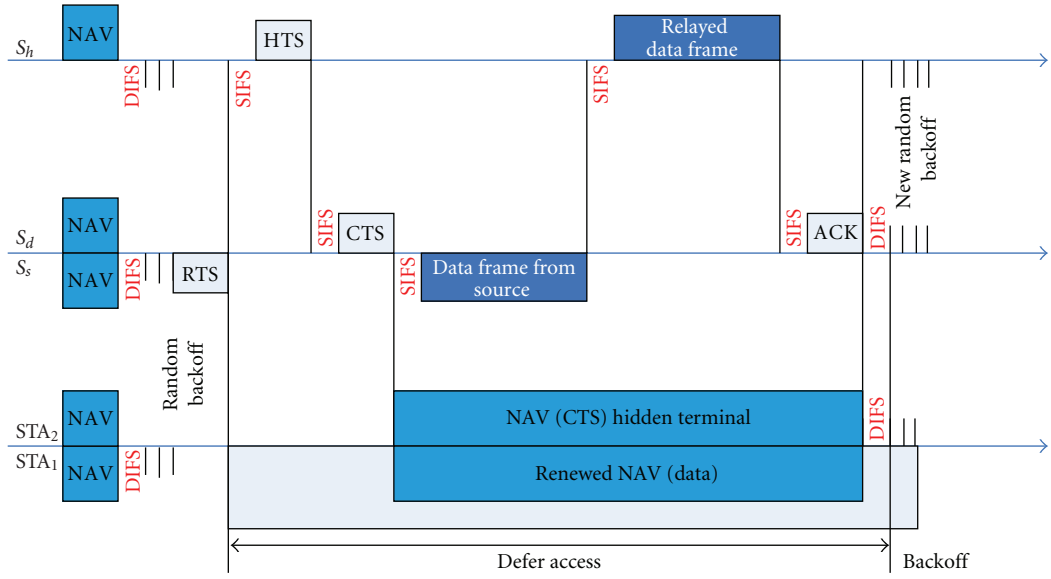
Table 1: Addressing scheme for different scenarios.

Scenario	Address 1	Address 2	Address 4
Source to destination	Destination	Source	Not used
Source to helper	Helper	Source	Destination
Helper to destination	Destination	Source	Not used

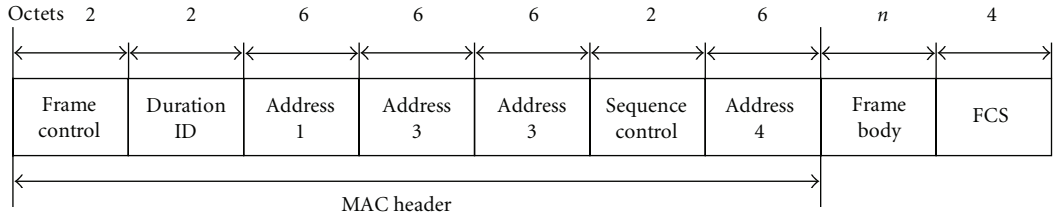
3.2. A Cooperative MAC Protocol. The set of new features of cooperative MAC spans both the data plane and control plane of the protocol stack. For ease of explanation, the terms *relay* and *helper* will be used interchangeably in the following discussion. As shown in Figure 1(b), STA_s, STA_h, and STA_d represent the source, helper, and destination station, respectively. Moreover, R_{sd}, R_{sh}, and R_{hd} denote the sustainable rates between STA_s and STA_d, between STA_s and STA_h, and between STA_h and STA_d, respectively.

3.2.1. Data Plane. Before the transmission of a packet, station STA_s should access all the rate information in a *cooperation table*, and compare a rough 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 aggregate 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 forwards it to the corresponding destination STA_d at rate R_{hd} after an SIFS interval. In the end, destination STA_d indicates its successful reception of the packet by issuing an acknowledgment packet (ACK) directly back to STA_s. We would like to mention here that we also considered PHY and MAC overheads in our original paper [8] for a more accurate estimation to decide whether a direct or two-hop is used. Based on the results we concluded that for an average length packet, those parameters do not have a significant effect.

As an option, the RTS/CTS signaling defined in IEEE 802.11 can be extended to a three-way handshake in CoopMAC to further facilitate the ensuing cooperative data exchange. Figure 2(a) depicts the transmission of the packet preceded by a three-way hand shake.



(a) Basic functionality of CoopMAC



(b) MAC header

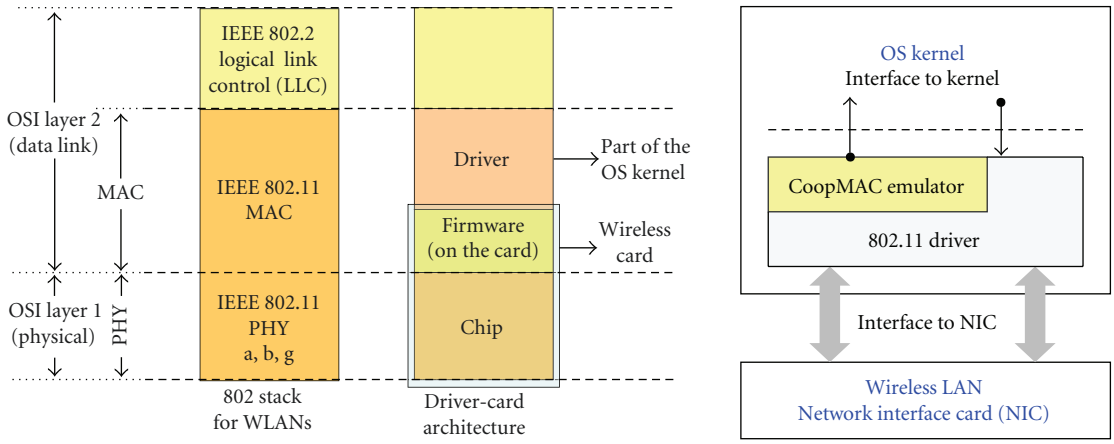
Figure 2: NAV settings and MAC header.

ID (48 bits)	Time (8 bits)	R_{hd} (8 bits)	R_{sh} (8 bits)	Number of failures
MAC address of helper 1	Time the last packet heard from helper 1	Transmission rate between helper 1 and the destination	Transmission rate between the source and helper 1	Count of sequential transmission failures
⋮	⋮	⋮	⋮	⋮
MAC address of helper N	Time the last packet heard from helper N	Transmission rate between helper N and the destination	Transmission rate between the source and helper N	Count of sequential transmission failures

Figure 3: CoopTable.

In order to distribute the identity of the station that has been selected as a helper, a minor modification has to be introduced to the addressing schemes defined in the legacy 802.11. More precisely, *Address 4* field in the legacy 802.11 MAC header as shown in Figure 2(b) is left unused if the data packets are not sent between access points. For the data packet from STA_s to STA_h in CoopMAC, however, this field should hold the MAC address of the final destination STA_d, while *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.

3.2.2. Control Plane. The key enhancement in the control plane at each station is the establishment and maintenance of a special data structure called the cooperation table (a.k.a. *CoopTable*) as shown in Figure 3, which contains essential information related to all the potential helpers.



(a) Driver arcitecture (b) Driver implementation

Figure 4: Driver architecture and CoopMAC implementation.

Each entry in the CoopTable, which corresponds to one candidate helper STA_h, is indexed by its MAC address. The values of R_{hd} and R_{sh} associated with STA_h are stored in the third and fourth field of the CoopTable, respectively. The main indication of the freshness of the learned information, namely, the time at which the most recent packet is overheard from STA_h, is held in the second field called *Timestamp*. The last field, *Number of Failures*, reflects the reliability of each helper, by recording the number of consecutive unsuccessful transmissions that use STA_h as a helper.

Whenever a packet is overheard from an STA_h, if that neighbor has no corresponding entry in the CoopTable, a new entry is created and inserted into the table; otherwise, all the fields associated with STA_h would undergo any necessary updates. It is worthwhile to note that 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. More specifically, the physical layer header (PLCP header) of any 802.11 data packet has rate information in its PLCP signaling field. Since PLCP header is always transmitted at the base rate, it can be decoded and understood by all other stations in the network, which includes STA_s. However, STA_s may not be able to correctly retrieve the MAC address of the transmitter and receiver directly from the corresponding data packet, since such information is contained in the MAC header and is in many instances transmitted at a rate higher than what STA_s can support. Fortunately, since each data packet sometimes are preceded by a successful handshake of RTS/CTS or succeeded by an acknowledgment, and all these control messages are exchanged at the base rate, STA_s eventually can find out the identity of STA_h and STA_d, with which the rate R_{hd} is associated. If there are direct transmissions between STA_s and STA_h, the rate estimation should proceed as prescribed by the rate adaptation algorithm that is used in the particular WLAN [25]. Although the described mechanism takes advantage of the rate adaptation capability of the network, it is independent of the particular rate adaptation algorithm. When no communication between these two stations occurs during an extended period of time, STA_s is still able to derive the highest rate R_{sh} that it can sustain by estimating the quality of the link between STA_s and STA_h based upon the signal strength of the packet that STA_s overhears from STA_h.

Since protocol design is not the primary focus of this paper, it will not be covered at length hereafter. It is worthwhile to note that although CoopMAC seemingly bears some resemblance to the conventional ad hoc routing protocols, they are in essence fundamentally different. First and foremost, forwarding in CoopMAC per se is just one practical means to accomplish the goal of leveraging cooperative diversity, instead of the goal itself. 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 readers are encouraged to refer to [8] for more detailed protocol specifications and technical discussions.

4. Driver Implementation Approach

When implementation was first attempted, only the two most widely used open-source Linux drivers for IEEE 802.11 wireless device were available, namely, *HostAP* [11] and *MADWiFi* [12]. Upon a thorough examination of the architecture of the respective driver and chipset, and the degree of freedom for protocol change allowed therein, it was determined that *HostAP*, which is based on *IntersilPrism* 2, 2.5, or 3 chipset, was more suitable to be adopted as the platform at that time. The basic wireless stack architecture of the driver-chipset platform is depicted in Figure 4(a). The wireless driver typically controls the functionality of the MAC layer that does not involve any time-sensitive issues (e.g., sending of an ACK after an SIFS period). We modified the wireless driver to implement CoopMAC as shown in Figure 4(b).

5. Implementation of Cooperation Using Open Source Drivers

Figure 4(b) depicts the way CoopMAC has been implemented in the driver-chipset platform. Due to the constraints of space, certain implementation details cannot be covered. Nevertheless, the key challenges encountered in the driver implementation are summarized. Interested readers can access the official project website [26] for more technical information. The driver for cooperative MAC is also available at the website for free downloading.

When it comes to system design, all the features specified in the IEEE 802.11 MAC protocol are logically partitioned into two modules, according to the time-criticality of each task. The lower module, implemented as firmware on the wireless card, fulfills the time-critical mission such as the generation and exchange of RTS/CTS control messages, transmission of acknowledgment (ACK) packets, execution of random backoff, and so on. The other module, which normally assumes the form of system driver, is responsible for more delay-tolerant control plane functions such as the management of MAC layer queue(s), the formation of MAC layer header, fragmentation, association, and so on.

As the cooperative MAC protocol requires changes to both time-critical and delay-tolerant logics, the inaccessibility to firmware unfortunately causes additional complexity in implementation. Indeed, compromises have to be made and alternative approaches have to be pursued, due to this constraint. For illustrative purpose, three main circumventions that have been made are outlined as follows.

(i) Suspension of Three-Way Handshake. As mentioned in Section 3.2.1, a three-way handshake 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 timing sequence of RTS and CTS packets has been hardwired in the firmware, an insertion of an HTS becomes impossible at the driver level. Consequently, three-way handshake of the protocol was suspended.

(ii) Unnecessary Channel Contention for Relayed Packet. Once the channel access has been allocated to the source station, the helper should relay the packet an *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 firmware. As a result, a compromise has been made in the implementation, in that the second hop transmission takes place after channel contention.

(iii) 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, the unwanted ACK from the relay has to be tolerated, instead of being eliminated.

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

5.1. Maintenance of the CoopTable. As described in Section 3.2.2, the *CoopTable* plays a key role in facilitating the cooperative operation. The passive approach defined therein for rate learning, however, has not been realized due to the following reasons.

(i) Unwanted Packet Filtering. All the packets with a destination address different from the local MAC address are filtered out by the firmware, instead of being passed up to the driver. Hence, the driver is not aware of such packets, and therefore unable to retrieve any information from them.

(ii) Controllability of the Experimental Environment. Even if the driver has access to such packets (e.g., by periodically switching the wireless card to the promiscuous mode), the traffic load and pattern at each station may cause inconvenience during experiments.

Therefore, for the sake of controllability of the experimental environment, an active information distribution approach is followed instead. More specifically, a *Hello* packet is broadcast by each station periodically which notifies its neighbors about its existence as well as the sustainable transmission rate on the respective link. The frequency of the *Hello* packet broadcasts in all the scenarios, except for the one described in Section 6.2, is one packet per second. Upon the reception of the *Hello* packet, a station either inserts a new entry or updates an existing one in its CoopTable.

To further increase flexibility, the frequency at which the *Hello* packet is transmitted, as well as the rate information to be carried has been implemented as parameters in the driver, which can be configured on the fly by the *iwpriv* command.

5.2. New Shim Header. No flexible mechanism is available on the *HostAP* platform to pass three MAC addresses down to firmware to generate a proper MAC header, which implies that the addressing scheme described in Section 3.2.1 cannot be faithfully followed. As a tentative solution, a new shim header called *CoopHeader*, which contains the MAC addresses of source, helper, and destination, has instead been inserted between the MAC header and the MAC payload.

5.3. Summary of Implemented Functionality. Depending on the specific role a station assumes, different new functions will be invoked, which is summarized in Table 2. In a real environment, every station can be assumed as a candidate helper for any neighbor station. Thus, irrespective of the actual role a station plays in the communication, it always transmits the *Hello* message periodically. On the other hand, once a station receives a *Hello* message, it updates its *CoopTable* based upon the received information. Under this

Table 2: Summary of implemented functionality.

Role	New functionality
Source	(1) Helper selection, based upon *CoopTable* (2) Creation and insertion of a *CoopHeader* in the packet to be transmitted
Helper	(1) Creation and insertion of a new *CoopHeader* in the packet to be relayed (2) Cooperative packet relay
Destination	(1) Packet reception and payload extraction

Table 3: Basic configuration of mobile stations.

Model	IBM T23
CPU power	Intel Pentium III processor 1 GHz
Memory	384 MB
Operating system	Redhat Linux 9
Kernel version	2.4.32
802.11 NIC	EnGenius 2511 CD PLUS, PCMCIA
802.11 Chipset	Intersil Prism 2.5

scheme, a station is always aware of candidate helpers in the area.

5.4. Experimental Setup. The setup used in the experiment consists of 10 laptops, whose basic configurations are outlined in Table 3.

In the ensuing experimental study, three different network topologies will be used, which are depicted in Figure 5. In each possible topology, one station is a dedicated destination, which mimics the functionality of an access point. The rest of the stations are either traffic sources, helpers, or both. To calibrate the testbed, the positions of stations have been adjusted until the throughputs achieved by all stations become roughly equal.

5.5. Measurement Methodology. The majority of the statistics generated in the experiment, including throughput, packet loss, and jitter, are measured by using *Iperf* [16], which is a powerful tool for traffic generation and results measurement. A typical experiment 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 is nontrivial, since no mean end-to-end delay statistics are provided by *Iperf* or other off-the-shelf traffic measurement tools. As further explained in [19], tight synchronization between the transmitter and receiver is needed, if the delay is to be measured directly.

To circumvent the synchronization requirement, which is 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 delays incurred in each direction can be considered 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 is the time from the instant 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.

6. Performance Evaluation for the Driver Approach

Based upon the testbed described in Section 5.5, numerous experiments have been conducted, and the results obtained are reported and analyzed in this section.

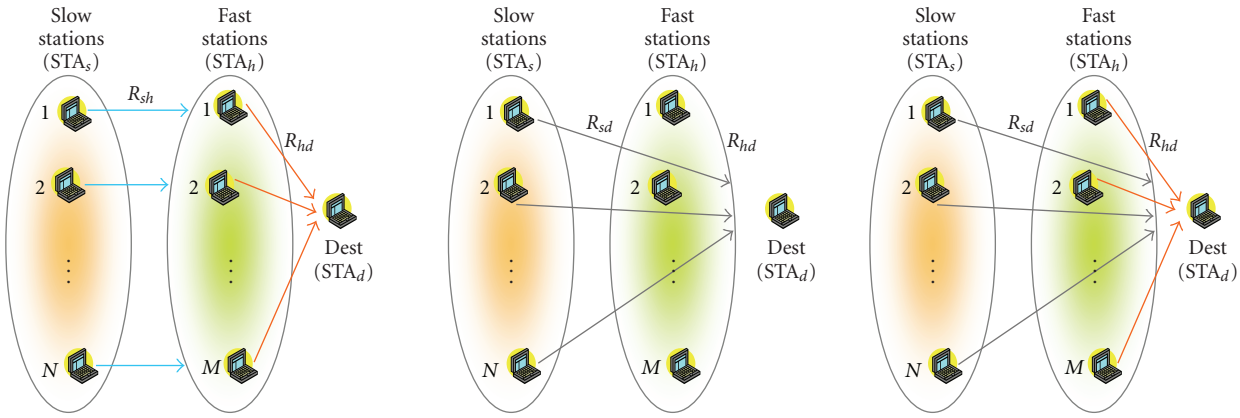
6.1. Baseline Scenario. A baseline scenario, which only consists of 1 transmitter, 1 helper, and 1 receiver, is first used to develop a basic understanding of the implication of cooperation, and establish a benchmark for performance study of more sophisticated settings. Thanks to its simplicity, this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

6.1.1. Throughput Improvement at Source. In the first 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. All rate combinations used in the experiment have been lised in Table 4.

Note that the helper STA_h in this case does not pump its own traffic into network. Separate experiments have been run for UDP and TCP traffic, respectively, and the results are depicted in Figure 6.

For both types of traffic, CoopMAC enables STA_s to deliver substantially higher throughput, as readily demonstrated in Figure 6. In addition, Figures 6(a) and 6(b) also disclose that the throughput gain achieved by cooperation becomes more pronounced, as transmission rates R_{sh} and R_{hd} are increased.

6.1.2. Throughput Improvement at Helper. The impact of cooperation on helpers, however, is not that straightforward, and requires further exploration. In the second experiment, the *Iperf* client at STA_h is switched on, so that it not only



(a) Cooperative MAC: helper may or may not generate traffic

(b) 802.11b—fast stations do not generate traffic

(c) 802.11b—fast stations also generate traffic

Figure 5: Experimentation topology.

Table 4: Settings for baseline scenario.

Case	R_{sd}	R_{sh}	R_{hd}	Traffic
1	1 Mbps	11 Mbps	11 Mbps	MSDU size = 1000 bytes
2	1 Mbps	11 Mbps	5.5 Mbps	
3	1 Mbps	5.5 Mbps	5.5 Mbps	
4	1 Mbps	11 Mbps	2 Mbps	Saturation load
5	1 Mbps	5.5 Mbps	2 Mbps	

relays traffic on behalf of STA$_s$, but also transmits its own packets to STA$_d$.

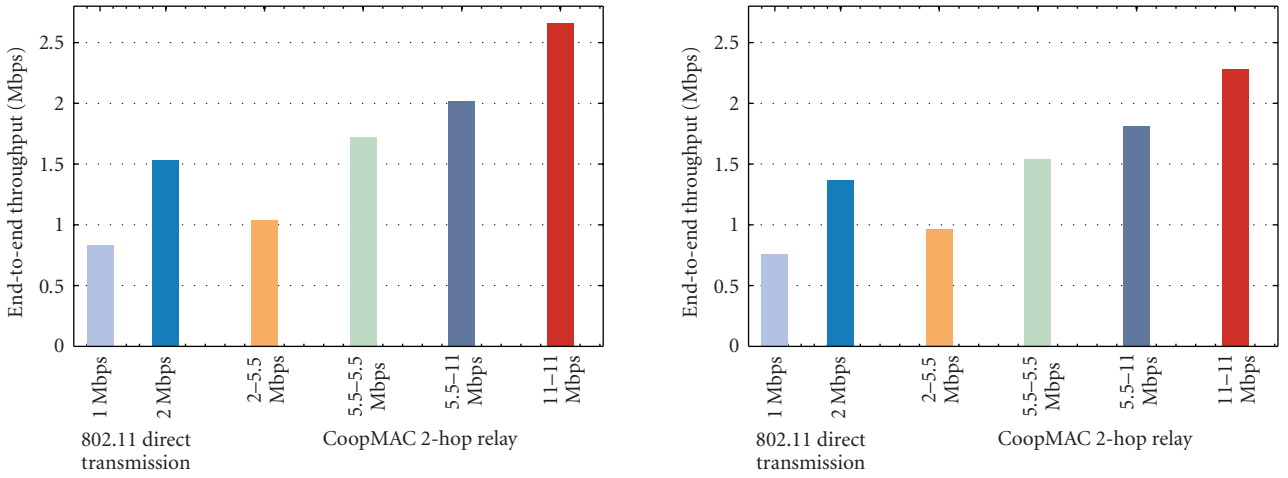
As suggested in Figure 7, CoopMAC protocol creates a *win-win* situation, instead of a zero-sum game. That is, STA$_h$ can derive some benefit by helping forward the packets for the slow source station. At first glance counterintuitive, this observation can be explained by the fact that if STA$_h$ participates in forwarding, STA$_s$ can finish its packet transmission much earlier, thereby enabling both STA$_s$ and STA$_h$ to transmit more bits in a unit time. From these results we conclude that CoopMAC also solves the performance anomaly problem of 802.11 [26] by boosting the slow stations' performance that results in the improvement of fast stations' performance.

6.1.3. Interaction with Transport Protocol. In Figure 7, 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 7(a) is that the bandwidth in the IEEE 802.11 network is equally shared by the two UDP sources at STA$_s$ and STA$_h$, respectively, in spite of the fact that physical layer bit rate supported by STA$_h$ is 11 times higher than that at STA$_s$. Indeed, this notion of fairness that 802.11 strives to maintain has been known as the major culprit for a serious network-wide throughput degradation [27]. The CoopMAC protocol obviously preserves this fairness, as no significant disparity in the throughput of STA$_h$ and STA$_s$ can be seen in Figure 7(a), while significantly increasing network throughput.

For TCP traffic in the 802.11 network, however, Figure 7(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. A closer examination discloses that long-term fairness can no longer be honored, primarily because of the widely known problematic cross-layer interaction between the random access MAC protocol and the TCP congestion control mechanism [28]. More precisely, the transmission of the slow station STA$_s$, which inevitably occupies more channel time, may cause the fast station STA$_h$ to experience an excessively long channel access delay. Even worse, due to the short-term unfairness issue described in [29], the channel can be captured by the slow STA$_s$ for an extended period of time. As illustrated in Figure 8, this channel capture effect further exacerbates the delay perceived by the fast station, which may lead to a TCP timeout, resulting in a reduction in the TCP congestion window, and ultimately slows down the TCP traffic at STA$_h$.

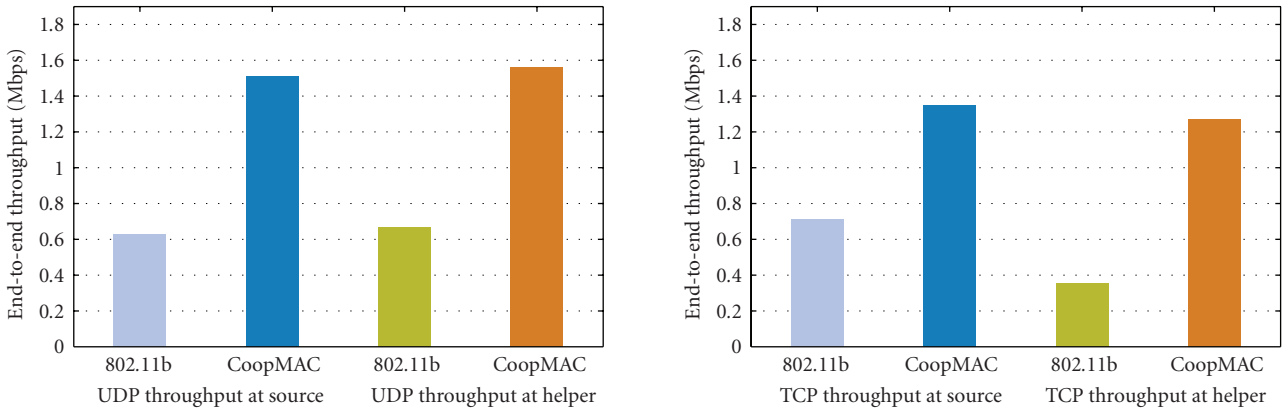
With cooperation, this mismatch between the MAC and TCP protocols can be ameliorated, and the long-term fairness be restored, as readily demonstrated in Figure 7(b). Thanks to the assistance of the cooperative relay, packets from the slow source station will release the wireless channel much earlier. Consequently, the delay experienced at the fast relay falls to a value too low to incur any higher layer timeout under most circumstances.

6.2. 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 set up 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 5. The second



(a) UDP (b) TCP

Figure 6: Throughput comparison: no active traffic from helper.



(a) UDP (b) TCP

Figure 7: Throughput comparison: active traffic from helper.

Table 5: 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

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 9. A small value of this interval lets the source STA_s be constantly updated of the current state of relay STA_{h1}, but unavoidably causes more overhead. On the other hand, 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 *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.

6.3. 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 a network's capability of supporting delay QoS-sensitive applications.

The scenario configured to measure the average end-to-end delay has been summarized in Table 6. The delay measurement methodology described in Section 5.4 has been

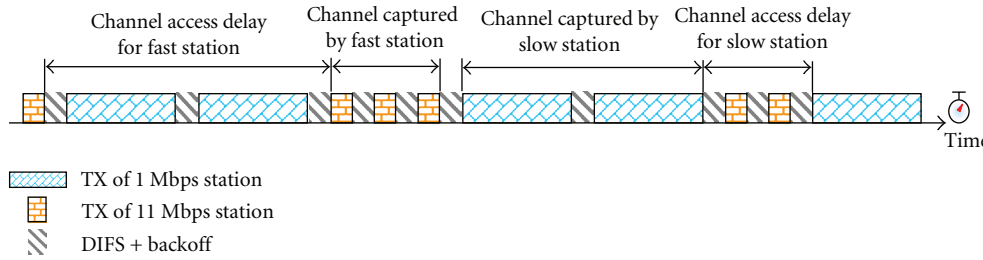


Figure 8: Short-term unfairness.

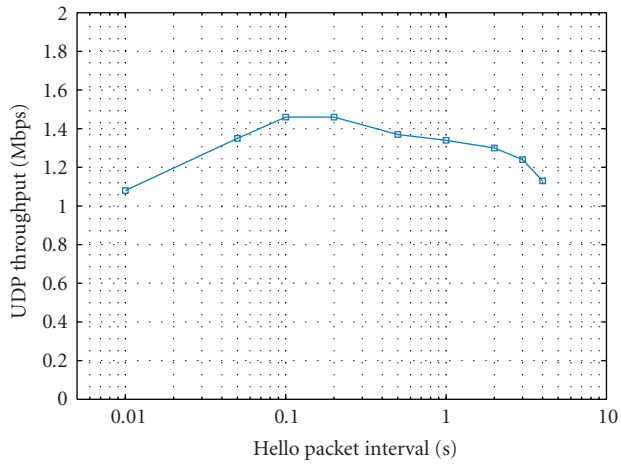


Figure 9: Impact of hello packet interval.

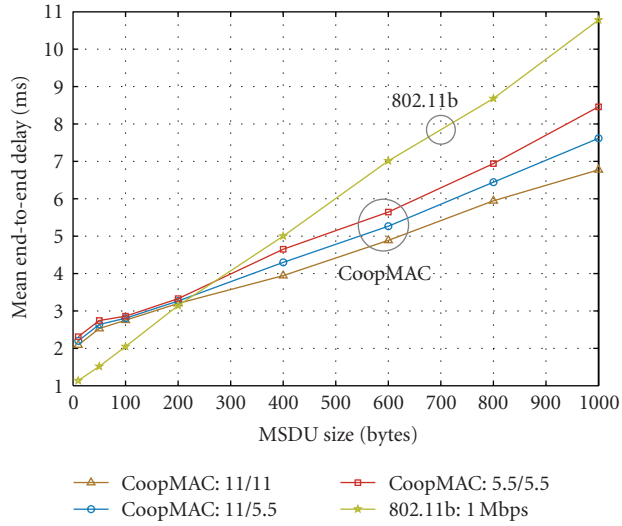


Figure 10: Mean end-to-end delay.

applied, and the average delay is obtained based upon the experimental results for over 10^6 broadcast packets.

As portrayed in Figure 10, it is evident that the cooperative forwarding significantly lowers the average delay for all the cases studied, when the MSDU size is reasonably large. Once the MSDU size drops below 200 bytes, IEEE 802.11b seems to perform better, since it avoids the overhead

Table 6: Settings for study of 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

Table 7: 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

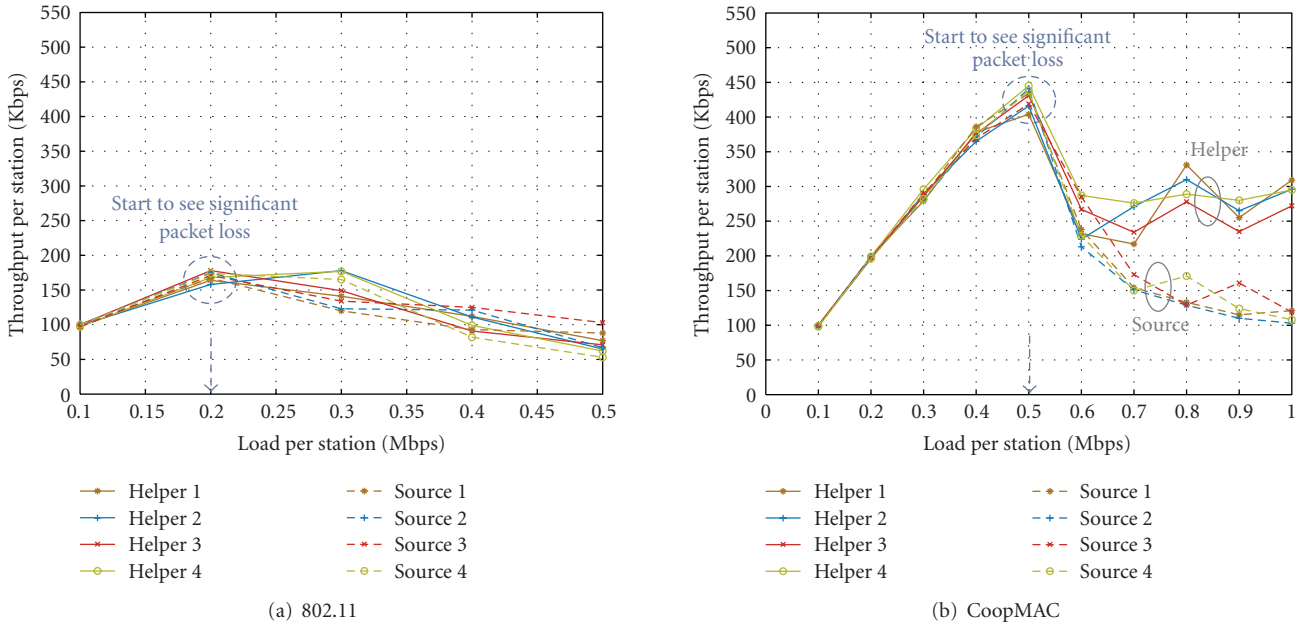
associated with CoopMAC. Nonetheless, note that this small adverse operation region may never be entered, if CoopMAC adopts a dynamic relay selection algorithm, in which the source STA$_s$ would simply fall back to legacy 802.11 for small frames.

6.4. 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 7 lists all the rate information related to the experiment.

For both the 802.11 and CoopMAC network, Figure 11 illustrates how the throughput achieved by each station changes with respect to the load applied. A simple comparison of Figures 11(a) and 11(b) shows that the per station throughput for both 802.11 and CoopMAC would increase, until the load saturates the system. In addition, both the fast helper stations and slow source stations still can accomplish a fair share of the bandwidth, which is anticipated.

However, the difference between the behavior of two protocols is more pronounced than the similarity, and the superiority of cooperative MAC is clear in this setting.

(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 thereby achieved by CoopMAC is approximately 2.5 times higher than that for 802.11.



(a) 802.11 (b) CoopMAC

Figure 11: Throughput comparison.

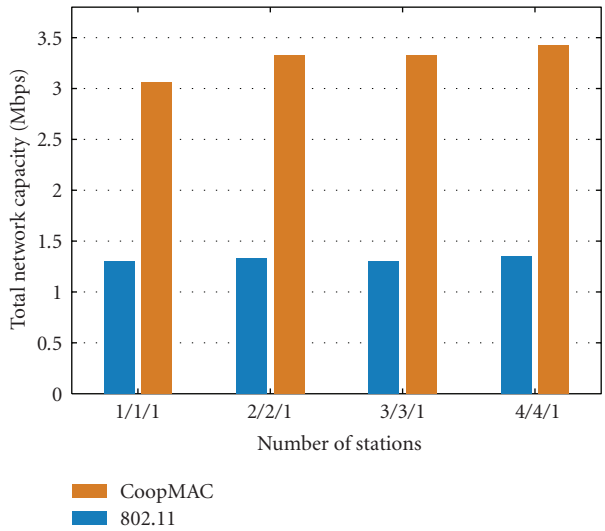


Figure 12: Network capacity versus number of stations.

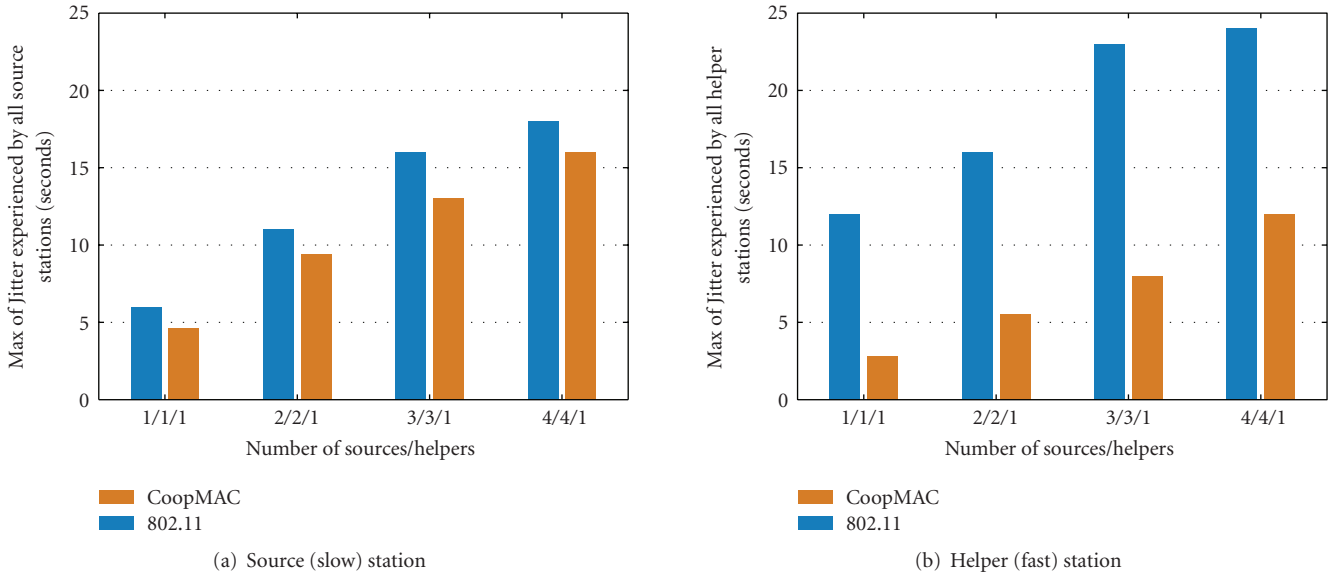
(2) Postsaturation Regime. Once entering the respective saturation regions, all stations in 802.11 invariably start to witness significant packet drop and throughput deterioration. For helper stations in cooperative MAC, however, the decrease is stalled after an initial dip, and then is stabilized at a plateau of about 0.28 Mbps/station. On the other hand, in spite of the fact that throughput of source stations in CoopMAC more or less follows the same trend of monotonic decline observed in 802.11, its absolute value is still notably higher.

A closer scrutiny further suggests that this performance disparity between the helper stations and source stations in a CoopMAC network is an artifact of our present implementation approach, and is expected to disappear once the access to firmware becomes available. More specifically, as explained in Section 5, 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.

Table 8: Settings for study of network capacity and jitter.

Acronym notation	Num. of source stations	Num. of helper stations	Num. of destination
1/1/1	1	1	1
2/2/1	2	2	1
3/3/1	3	3	1
4/4/1	4	4	1

6.5. Network Capacity and Jitter. To gain a high level view of the protocol performance, the aggregate network capacity and jitter statistics for UDP traffic are collected and depicted in Figures 12 and 13, respectively. The corresponding experiment settings are summarized in Table 8. The *number of stations* referred in the horizontal axis of Figure 12 includes both the source and helper stations, but not the destination. 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.



(a) Source (slow) station

(b) Helper (fast) station

Figure 13: Jitter comparison.

As demonstrated in Figure 12, the CoopMAC protocol easily delivers a network capacity that is up to 2.5 times higher than the achievable by 802.11. In addition, this improvement is sustainable across a variety of network sizes.

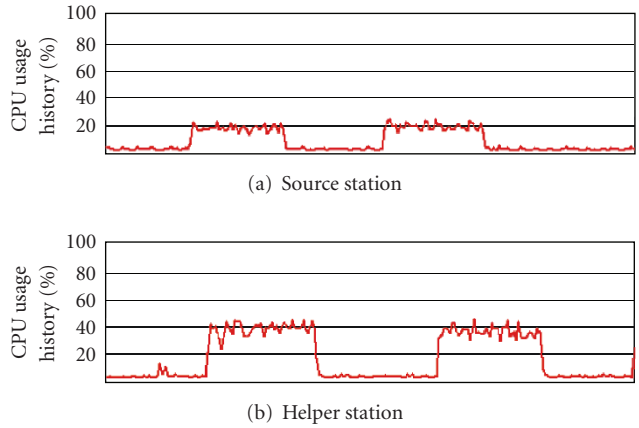
Concerning the jitter, *Iperf* can provide a measurement for each traffic stream. Use $J(\text{STA}_{s_i})$ and $J(\text{STA}_{h_j})$ to denote the jitter observed at each source and helper station. To compare the worst case scenario, $\max[J(\text{STA}_{s_i})]$ and $\max[J(\text{STA}_{h_j})]$ have been extracted from the statistics and depicted in Figure 13 for source and helper station, respectively.

As compared to network capacity, both Figures 13(a) and 13(b) indicate that jitter is more sensitive to the network size. Moreover, although helper stations support a higher transmission rate than source stations, they experience a higher variance in end-to-end delay (jitter) in an 802.11 network. A similar trend has been previously identified and an explanation was offered in Section 6.1.3, where the interaction with 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.

6.6. Computational Overhead. A substantial proportion of mobile devices deployed in the field have limited computing power. To assess the feasibility of leveraging cooperative diversity from devices with such a constraint, the computational overhead incurred by cooperation should be evaluated.

In this experiment, settings similar to that outlined in Section 6.1 has been used. Two UDP packet trains with 4 Mbps load are generated at the source station, and CPU usage at both the source and helper station during the time interval of packet transmission is recorded by using the *GNOME System Monitor* tool. The captured traces are displayed in Figure 14, the horizontal and vertical axes of which represent time evolution and the percentage of CPU resources used at that time, respectively. The comparison of Figures 14(a) and 14(b) suggests that more CPU cycles have to be consumed by the helper to relay a packet than by the source station to transmit a packet, which is primarily due to the fact that the reception of the packets to be forwarded at the helper causes additional computation expense. Note that once the protocol is implemented in the firmware, this additional CPU expense would not occur, because all the processing associated with relay then would be handled by the wireless LAN card and be transparent to the host CPU.



(a) Source station

(b) Helper station

Figure 14: CPU usage comparison.

Despite the increase of computational overhead at the helpers, neither the source nor the helper have been overwhelmed by the processing associated with cooperation. Moreover, since the laptops used in the testbed are not top

(a) Legacy 802.11 (b) CoopMAC

Figure 15: Video quality comparison: a snapshot.

of the line, the impact of additional computational overhead would be even less noticeable on state-of-the-art mobile devices.

6.7. A Demo of a Video Application. Although highly encouraging, all the aforementioned results are obtained in experiments that rely on artificial traffic patterns. The final judgment regarding the efficacy of cooperation cannot be made until the improvement is delivered to the application layer and becomes appreciable through user perception.

To this end, the transmission of a video clip is considered in the testbed described in Section 6.1. A Video LAN Client (*VLC*) [30] server is placed at the source station and constantly streams a commercial video clip, while the destination station runs a *VLC* media player to play the video.

As anticipated, the user perception is poor for video transmission in the 802.11 network, as noticeable freezes and distortions occur frequently. The video is smooth and artifact-free when it is received over a cooperative MAC network. Figures 15(a) and 15(b) provide a snapshot of the video seen at the destination for 802.11 and cooperative MAC, respectively. The comparison of these two figures is typical and reveals the substantial difference between the video quality that these two different protocols can deliver.

7. Software-Defined Radio Approach

As mentioned earlier, we were not able to implement CoopMAC in its full sense due to several limitations posed by the architecture of the wireless card hardware. In particular, CoopMAC defines some features that require modifications in the time-critical functionality of the wireless card. As a result, the final implementation of the scheme in HostAP was limited to an emulation of the original protocol. The open-source drivers implementation of CoopMAC missed several critical functional characteristics of the original protocol. The cooperative protocol, the way it was implemented on the open source drivers, is very similar to a layer 3 forwarding mechanism that takes into consideration the channel quality for the next hop forwarding. In particular, it introduces more overhead and it suffers from longer delays. Nevertheless, the experimental results showed significant benefits of using cooperation in the MAC layer. Therefore we decided to move forward and continue with the implementation of the protocol using a more flexible platform in order to achieve more accurate results.

The obvious choice was a software-defined radio, since in such an approach, both the PHY and the MAC layers are designed in software and therefore they can be changed. Moreover, an all-software radio platform gives us the ability to go lower in the PHY layer and design MAC and PHY cross-layer schemes that enable PHY layer cooperation at the receiver. Two strong candidate platforms were GNU radio [14] and WARP [15]. GNU radio is a popular platform that has the MAC and PHY layer implementations in software that can run on a PC. The PC communicates though a USB cable with a simple transceiver that takes care of the transmission/reception of the signal. Due to the USB connectivity between the PC and the wireless board, GNU radio has a very limited capability of implementing sophisticated PHY and MAC protocols since this communication experiences long delay. This delay introduces synchronization problems in the MAC layer and performance limitations in the PHY layer. Therefore, although GNU radio allows us to build a version of our protocol, the above limitations do not allow for a realistic implementation.

WARP seemed a more promising solution since it consists of a Xilinx Virtex II Pro FPGA board with embedded Power PC processors. The processing and the transmission/reception of the frames are done on the board. Such a hardware platform allows for realistic MAC and PHY layer implementations with PHY layer rates similar to those in IEEE 802.11a,g.

Using the WARP radio platform, we were able to overcome all of the three limitations listed earlier. However, in this paper we focus on the description and the implications of the first two limitations. The RTS-CTS model is an optional supportive functionality that copes with the hidden terminal problem. Consequently, the RTS-HTS-CTS model is an extension of the RTS-CTS scheme that copes with the same problem. Since the focus of this work is the study of the benefits of cooperation between stations in the MAC layer, the study of the hidden terminal problem is out of the scope of this paper.

7.1. Software-Defined Radio Testbed Architecture. WARP consists of a Xilinx Virtex II Pro FPGA board with embedded

Power PC processors and allows for realistic MAC and PHY layer implementations that could give PHY layer rates similar to those provided in IEEE 802.11a,g. It provides a complete embedded programing environment for the design of PHY and MAC layers. In addition, it has four daughter card slots in which radio cards or customized cards can be inserted to connect to the FPGA. The current physical layer design uses an Orthogonal Frequency Division Multiplexing (OFDM) implementation that is loosely based on the PHY layer of the 802.11a standard. The radio board uses 2.4 GHz/5 ISM/UNII bands for transmission.

In the MAC layer WARP provides a framework called WARPMAC and WARPHY which is used for the development of advanced MAC protocols. WARPMAC and WARPPHY are a set of functions that provide MAC-type functionalities and functionalities to access the PHY layer, respectively, and they work as the interface between the PHY and the user application layer. This MAC framework is implemented in the PowerPC and the code is written in the C language using Xilinx Platform Studio.

Rice University provides many software resources at the WARP web site [15] including an Aloha-like MAC and a CSMA-like MAC. For our implementation we based our development on the CSMA-like MAC.

8. Implementation of CoopMAC Using the All Software Radio Platform

In our implementation of CoopMAC on an all software radio platform, the OFDM reference design version 8 of the WARP platform has been used. The OFDM reference design version 8 implements the CSMA protocol for medium-access control, so it is a perfect candidate for legacy wireless protocols. We implemented CoopMAC using the CSMA model of the WARPMAC framework. Whenever we refer to a node in the following discussion, we refer to a WARP node.

In our implementation we define two operational modes for the transmission. *Direct mode* is the legacy direct mode under the CSMA protocol (no cooperation), and *Cooperative mode* is the mode that enables CoopMAC. In this mode the packet is forwarded to the destination through the helper using two fast hops. The decision about whether the transmission is in *Direct mode* or in *Cooperative mode* is taken by the source station after considering the information maintained in the *CoopTable* about candidate helpers in neighborhood and the rates they can sustain with both the source and the destination. In the rest of this section we describe the changes we introduced in several parts of the CSMA functionality of WARPMAC in order to implement the cooperative MAC protocol.

8.1. Addressing and Packet Structure. The *addressing scheme* that we used for CoopMAC is based on the one defined in WARPMAC. Each node has a unique nodeID which is determined by 4 dip switches. Therefore, a total of 16 unique nodeID's can be generated. Based on the nodeID, the MAC code generates a MAC address string and assigns it to the node. The nodes maintain a table that maps nodeID's to the corresponding MAC addresses.

The following is the description of the *Packet structure* that is defined in WARPMAC as well as the necessary changes we made to support CoopMAC. We call the enhanced packet structure *CoopFrame*. CoopFrame consists of two parts, the *MAC Header* and Data Payload. The MAC header consists of two fields, *Phyheader* and *isNew*. *isNew* is a flag that indicates whether a packet is under transmission process (transmitted but not acknowledged) or reception process (received but not yet processed) and its functionality is not a part of the CoopMAC implementation. The *Phyheader* consists of following sub-fields.

(1) *Source address*: the MAC address of the source station (in both direct and cooperative mode).

(2) *Destination address*: the MAC address of the destination station. In direct mode this is the address of the final destination. In the cooperative mode this is the address of the immediate destination in that particular hop (i.e., the helper in the first hop, the final destination in the second hop).

(3) *CoopDestinationID*: a new subfield we introduce in order to handle the forwarding process. It is used in the cooperative mode and it indicates the *final destination* for the packet. In the first hop, this field indicates the final destination while the *Destination Address* field (mentioned above) indicates the address of the helper. *CoopDestinationID* is used by the helper when it generates the header of the packet for the second hop in order to define the final destination of the packet.

(4) *PktType*: used to indicate the nature of the packet.

   (a) *DATAPACKET*: a packet that is used in direct mode.

   (b) *COOPPACKET*: a packet that is used in CoopMAC for the first hop transmission (source to helper).

   (c) *COOPFINAL*: a packet that is used in CoopMAC for the second hop transmission.

   (d) *ACK*: a control packet that acknowledges successful reception and sent by the destination to the source.

(5) *Full Rate*: used to indicate the rate at which the payload of a packet is transmitted.

(6) *Current Resend*: used to indicate the number of retransmissions.

(7) *Length*: used to indicate the length of the payload.

(8) *Checksum*: a checksum value that is calculated and handled by the PHY layer.

8.2. Transmission. When the MAC layer of a node receives a packet for transmission from the application layer, it refers to the *CoopTable* to decide whether to use a helper (cooperative mode) or transmit directly (direct mode). Based on the chosen mode, the MAC header is created. In Direct mode the *Packet Type* is *DATAPACKET*, *Source Address* is the node's MAC address, and the *Destination Address* is the destination MAC address. *CoopDestinationID* field is not used in this case. In case of Cooperative mode, *Destination Address* is the address of the helper and the *CoopDestinationID* is the ID of the final destination. This allows the helper to generate the second hop packet header as was described in the last subsection. Once the packet is appended with the appropriate MAC header, the node initiates a transmission using the CSMA protocol. The packet, in the case of the direct mode, is transmitted directly to the destination, while in the case of the cooperative mode, it is forwarded to the helper. The transmission rate in each case is adjusted through a rate adaptation scheme that is based on the channel condition between the source and the intended destination. We must mention here that the source will use a cooperative mode only if the rates in the two hops are higher than the direct rate (and therefore will lead to gains using this scheme). Figure 16(a) provides a simplified flow graph of the transmission process in CoopMAC.

8.3. Reception. On reception of a packet the node checks whether it is the receiver by checking the *Destination Address* field in the packet header. If the node is the receiver, four cases can arise, based on the value of the *Packet Type* field.

(1) *DATAPACKET*: if the received packet is DATAPACKET, then an ACK is transmitted back to the source node.

(2) *COOPPACKET*: the packet type used between the source and the helper. On receiving a COOPPACKET, the receiver realizes that it should react as a helper. Therefore, it replaces the *Destination Address* field with that of the final destination address based on the *CoopDestinationID* field, and forwards the packet immediately, without contending for the channel.

(3) *COOPFINAL*: the packet type used between the helper and the final destination. On receiving COOPFINAL packet, the destination sends back an ACK, directly to the source node.

(4) ACK: on receiving an ACK, the source node stops the timeout process and proceeds with the next transmission.

A simplified flow graph of the reception process is shown in Figure 16(b). In this particular figure we do not show the ACK reception. In order to enable the ACK transmission directly from the destination to the source, the *Source Address* field of the packet header remains the same throughout the two hop transmission. In this way the final receiver is aware of the actual source.

8.4. Implementation of the CoopTable. The CoopTable is an important feature of CoopMAC since it allows nodes to decide whether they should use cooperation or not. The WARP implementation of CoopTable is shown in Table 9.

The sustainable transmission rate is represented with a metric of the channel which is a measure of the achievable transmission rate. In our implementation, as a metric value, we use the numeric mask that defines a particular data rate in the PHY layer. The metric to data rate mapping for WARP is shown in Table 10.

In this table, a higher metric value implies a higher data rate. The CoopTable is updated passively after the reception of any packet that is transmitted by a node in the neighborhood. By checking the *Full Rate* subfield in the MAC header of the packet, the node is aware of the bit rate of the packet payload and therefore the channel condition between its source and the destination. In this way, a node gets information about the channel conditions between neighboring helpers and itself, as well as with potential destinations. We should mention that the MAC header of the packet is transmitted at the base rate (BPSK), and therefore any node in the proximity of the transmitter can receive it, decode it, and use the information contained within it to update its CoopTable. In addition to this passive approach, we implemented an active approach where periodic *Hello* packets are transmitted by each node in the network. A *Hello* packet contains information about the sustainable rates between the particular node and its neighbors (*Rate Table*). A node that receives a *Hello* packet updates its CoopTable based on this information.

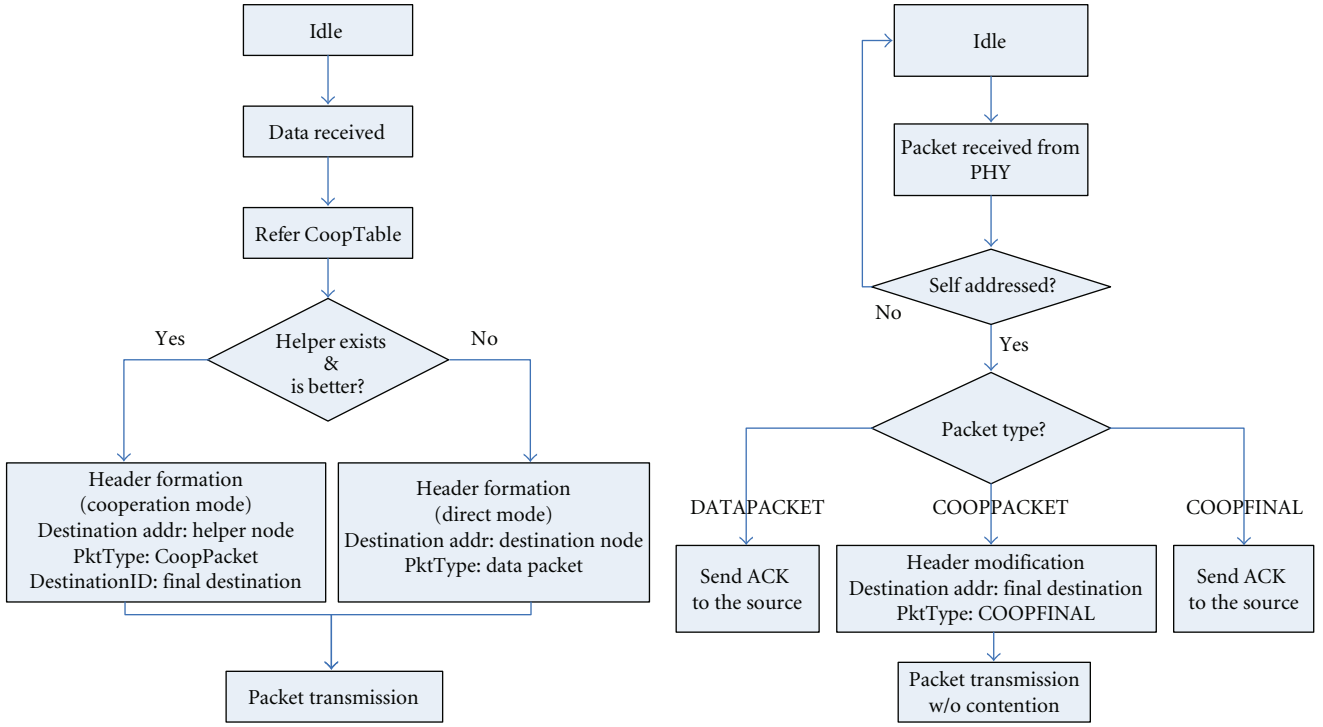
8.5. Transmission Rates. WARP nodes supports dynamic modulation on a per packet basis. This information is included in the *Full Rate* subfield of the MAC header of the packet and is used for the demodulation of the packet at the receiver. The MAC header is transmitted at the base rate, which is BPSK for our implementation. This is done to increase the robustness of the decoding process of the header at the receiver. Similarly, an ACK is transmitted at the lowest rate using BPSK in order to minimize ACK loss.

9. Performance Evaluation on the Software Radio Approach

In order to study the performance of the CoopMAC we conducted several experiments. In this paper, due to space limitations, we describe two basic scenarios that give a clear picture of the performance of the new implementation. In the performance evaluation, we compared the implemented CoopMAC protocol with two other schemes.

(i) The CSMA approach that emulates the IEEE 802.11 MAC protocol. We will call this scheme *direct transmission*.

(ii) Approach 1 as described prevously based on the Driver platform. In order to differentiate between the two cooperative approaches we call this scheme *CoopMAC with contention*, while we call the accurate implementation (SDR approach) of the mechanism *CoopMAC* without contention.



(a) Transmission flow (b) Reception flow

Figure 16: Simplified flow graphs of transmission and reception processes in CoopMAC.

Table 9: CoopTable.

Destination MAC address	DirectRate (Mbps)	Helper MAC address	R_{sh} (Mbps)	R_{hd} (Mbps)
16.24.63.53.e2.c3	6	16.24.63.53.e2.c4	24	24
16.24.63.53.e2.c7	6	16.24.63.53.e2.c12	24	12
—	—	—	—	—

Table 10: Supported data rates in WARP.

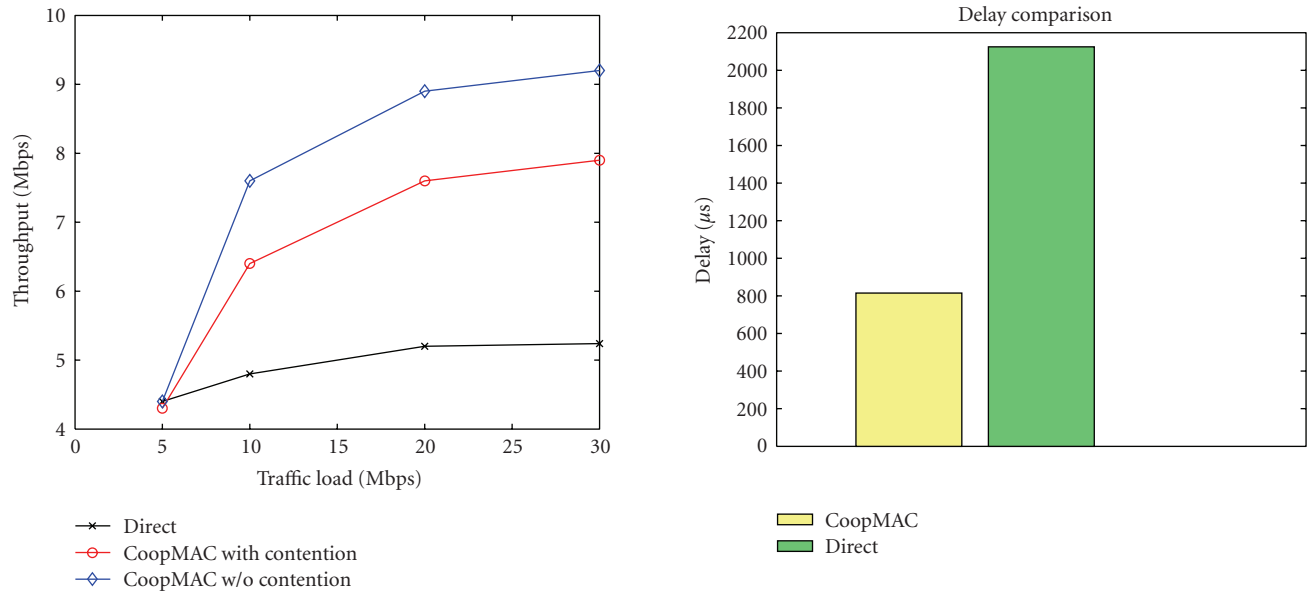
ModulationScheme	Metric	PHY rate (Mbps)
BPSK	1	6
QPSK	2	12
QAM16	4	24
QAM64	6	36

As evaluation metrics we use the total number of successful packets (throughput) as well as the average delay per packet. We should mention that the QAM 64 modulation scheme of the current PHY layer implementation in the WARP platform is not very stable, and therefore we avoided using this rate in our experiments. We only used BPSK, QPSK, and QAM-16.

We currently have only three WARP nodes for conducting experiments. However, this small-scale testbed was enough for our purposes which was to show the fundamental benefits gained when using cooperative MAC schemes in a real environment.

All measurements were done indoors. For generating UDP traffic, we used Iperf [16]. In all cases, the UDP packet length was 1470 bytes. Each scenario was run 10 times, for 50 seconds each time, and the results were averaged. For the experiments, three nodes were used: a source, a destination, and a helper. Therefore, the information in CoopTable was statically entered with metrics depending on the particular scenario. The metric selection is described in detail for each experiment.

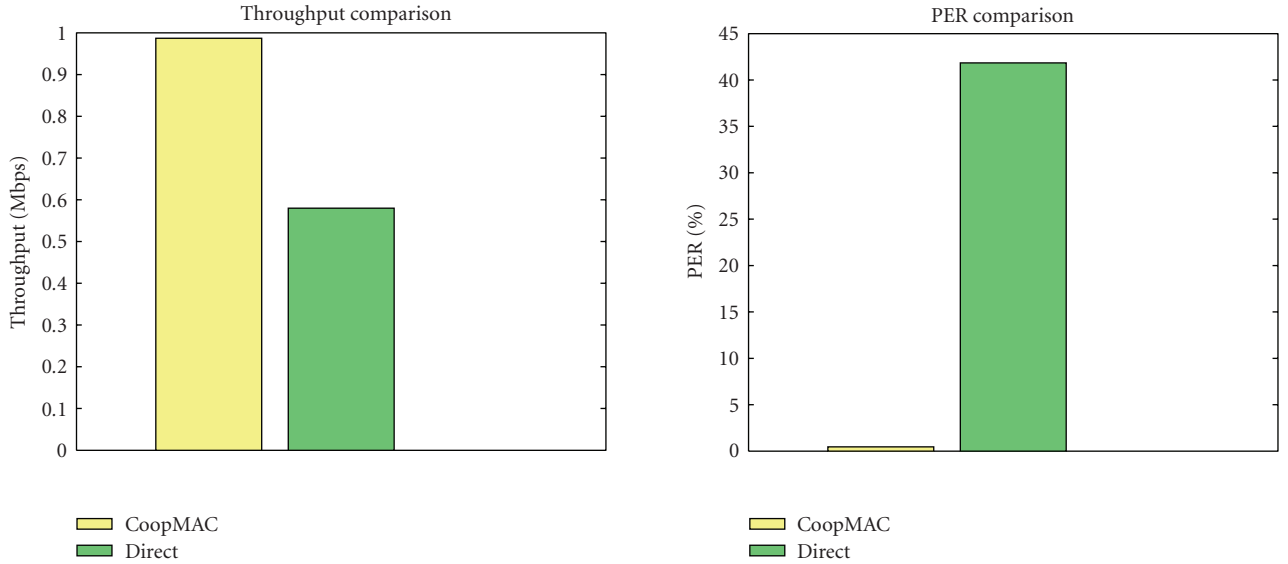
9.1. Scenario 1. In the first experiment we study the performance of the cooperative MAC protocol in a typical scenario. We consider the case when the channel between the source and the destination is poor and that the helper is located in between the two nodes. Therefore, it has a good channel quality with both of them. We compare the CoopMAC implementation with *CoopMAC with contention* as well as to *direct transmission*. We emulated the bad channel in direct mode by forcing the data rate in the direct transmission to be 6 Mbps (BPSK). The transmission via the helper for both hops was fixed at 24 Mbps (QAM-16). Using Iperf we generated UDP traffic that was passed to the WARP nodes connected to PCs through an Ethernet cable.



(a) Throughput performance

(b) Delay performance at heavy traffic load (30 Mbps)

Figure 17: Throughput and delay performance in scenario 1.



(a) Throughput comparison

(b) Packet error rate comparison

Figure 18: Throughput and packet error rate comparison in scenario 2.

In Figure 17(a) we see the throughput of the three schemes as the traffic load increases. It is clear that CoopMAC (with or without contention) performs better than the direct transmission. This is due to the fact that the cooperative protocol significantly reduces the transmission time taken by the slow node. Therefore, cooperation enables efficient use of the wireless channel to achieve extra capacity. Additionally, we can see in the figure that the new implementation (*CoopMAC without contention*) performs much better than our earlier implementation (*CoopMAC with contention*). This is because in *CoopMAC with contention*, the source and helper compete with each other for the medium for the first and second hop transmissions. Additionally, in this scheme two ACKs are generated for each successful forwarding. In the more accurate implementation of CoopMAC (*CoopMAC without contention*) there is no contention for the second hop transmission. Additionally, there is a single direct ACK for each two hop transmission. Therefore, the boost due to cooperation is even higher.

In Figure 17(b), we depict the delay for each scheme under heavy load. It is clear that the cooperative protocols decrease the transmission time of the slow node thus

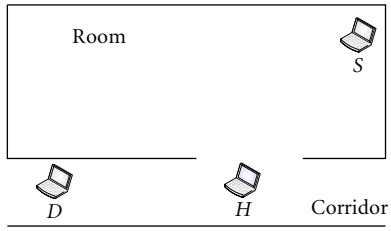


Figure 19: Scenario 2 setup.

reducing the delay. For *CoopMAC without contention* the delay is even smaller since it avoids the extra delay that is introduced in the second hop due to contention before the transmission of the packet.

9.2. Scenario 2. In a typical cooperative system the gains of cooperation can be translated into different metrics. By using cooperative protocols we can boost the transmission rates while keeping the packet error rate (PER) constant, or we can decrease the PER for the same transmission rate, or we can decrease the transmission power for the same transmission rate and PER. In the previous scenario we showed the benefits of cooperation by boosting the transmission rate, while keeping the PER constant. In this experiment we show the gains obtained by decreasing the PER while fixing the transmission rate. We set up the topology of the experiment as shown in the Figure 19, where the source S and the destination D were not in the line of sight of each other, and they were located in positions where their communication was poor even at the basic rate (BPSK). The helper H is put in a position between the source and the destination. The transmission rate for the direct as well as for the two hops of the cooperative communication is 6 Mbps (BPSK). We ran Iperf and applied different traffic loads. We compared the performance of the newly implemented scheme to that of the *direct transmission*. In Figure 18 we show the throughput and the PER for a traffic load of 1 Mbps. As we can see in Figure 18(a), the throughput of CoopMAC is almost double that of *direct transmission*. This initially seems counter-intuitive due to the fact that now the cooperative MAC protocol breaks the transmission into two hops, each at the basic rate and therefore doubles the transmission time. The explanation of this result is apparent in Figure 18(b) that depicts the PER for the same scenario. As the figure shows, the PER for the direct transmission is very high (higher than 40%). However, by using the cooperative scheme and forwarding the packets through a helper that sustains a good channel with both the source and the destination, we can keep the PER of the communication at a very low level (less than 2%), and therefore increase the efficiency of the network.

10. Conclusions and Future Work

The impact of a performance study in a real environment can never be overemphasized as it is able to identify the limitations of the predictions yielded by theoretical analysis and simulation, and valuable practical insights into protocol design and potential improvements are gained.

This paper represents one of the few attempts that rely on an experimental approach to develop an understanding of cooperation at MAC layer. The measurement results obtained confirm that the cooperative MAC protocol can substantially improve the performance (e.g., throughput, mean end-to-end delay, jitter, etc.) 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, and the dynamics of protocol behavior, which to the best knowledge of the authors have been presented for the first time. Note that early awareness, precise comprehension, and proper caution when addressing these issues can help in future implementation and experimentation.

The paper describes two different implementation approaches. An implementation that is based on open source drivers and an implementation that is based on an software-defined radio platform are presented. A detailed description of the motivations for the implementation of the protocol on each platform is given, as well as the benefits and limitations of the two approaches. The SDR approach seems to be highly promising, as it allows modification of the physical layer functions, and therefore makes it possible to realize MAC-PHY cross-layer mechanisms. On the other hand, an open-source wireless driver platform limits the capability of modifying physical layer functions. However, it enables the resultant prototype to be directly compared with 802.11 commercial products, something that is not feasible in the case of the SDR.

As for possible future work, we are planning to continue with the implementation of cooperative schemes in the PHY layer, and combine them with the existing cooperative MAC protocol. In this way, we will implement realistic cooperative cross layer mechanisms that will further improve wireless network performance by enabling cooperation at the PHY layer as well.

Acknowledgment

Some results contained in this paper have been previously presented at IFIP/TC6 Networking 2007 [31] and in LANMAN 2008 [32].

References

[1] J. N. Laneman, D. N. C. Tse, and G. W. Wornell, "Cooperative diversity in wireless networks: efficient protocols and outage behavior," *IEEE Transactions on Information Theory*, vol. 50, no. 12, pp. 3062–3080, 2004.

[2] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity—part I: system description," *IEEE Transactions on Communications*, vol. 51, no. 11, pp. 1927–1938, 2003.

[3] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity—part II: implementation aspects and performance analysis," *IEEE Transactions on Communications*, vol. 51, no. 11, pp. 1939–1948, 2003.

[4] K. Tan, Z. Wan, H. Zhu, and J. Andrian, "CODE: cooperative medium access for multirate wireless ad hoc network," in *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '07)*, pp. 1–10, San Francisco, Calif, USA, June 2007.

[5] S. Sayed and Y. Yang, "A new cooperative MAC protocol for wireless LANs," in *Proceedings of the London Communications Symposium (LCS '07)*, London, UK, September 2007.

[6] L. M. Feeney, B. Cetin, D. Hollos, M. Kubisch, S. Mengesha, and H. Karl, "Multi-rate relaying for performance improvement in IEEE 802.11 WLANs," in *Proceedings of the 5th International Conference on Wired/Wireless Internet Communications (WWIC '07)*, vol. 4517 of *Lecture Notes in Computer Science*, pp. 201–212, Coimbra, Portugal, May 2007.

[7] N. Agarwal, D. Channegowda, L. N. Kannan, M. Tacca, and A. Fumagalli, "IEEE 802.11b cooperative protocols: a performance study," in *Proceedings of the 6th International IFIP-TC6 Networking Conference on Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet (NETWORKING '07)*, vol. 4479 of *Lecture Notes in Computer Science*, pp. 415–426, Atlanta, Ga, USA, May 2007.

[8] P. Liu, Z. Tao, S. Narayanan, T. Korakis, and S. S. Panwar, "A cooperative MAC protocol for wireless LANs," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 2, pp. 1–40, 2007.

[9] H. Zhu and G. Cao, "rDCF: a relay-enabled medium access control protocol for wireless ad hoc networks," in *Proceedings of the 24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 1, pp. 12–22, Miami, Fla, USA, March 2005.

[10] J. Alonso-Zárate, E. Kartsakli, C. Verikoukis, and L. Alonso, "Persistent RCSMA: a MAC protocol for a distributed cooperative ARQ scheme in wireless networks," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, Article ID 817401, 13 pages, 2008.

[11] "Host AP driver for Intersil Prism2/2.5/3," http://hostap.epitest.fi/.

[12] "MADWiFi: Multiband Atheros Driver for WiFi," http://madwifi.org/.

[13] "Intel Wireless WiFi Link drivers for Linux*," http://www.intellinuxwireless.org/.

[14] "GNU Radio—The GNU Software Radio," http://gnuradio.org/trac.

[15] "WARP: Wireless Open-Access Research Platform," http://warp.rice.edu/trac.

[16] "Iperf: The TCP/UDP Bandwidth Measurement Tool," http://en.wikipedia.org/wiki/Iperf.

[17] A. Miu, G. Tan, H. Balakrishnan, and J. Apostolopoulos, "Divert: fine-grained path selection for wireless LANs," in *Proceedings of the 2nd International Conference on Mobile Systems, Applications and Services (MobiSys '04)*, pp. 203–216, Boston, Mass, USA, June 2004.

[18] A. Raniwala and T.-C. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *Proceedings of the 24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 3, pp. 2223–2234, Miami, Fla, USA, March 2005.

[19] I. Dangerfield, D. Malone, and D. Leith, "Experimental Evaluation of 802.11e EDCA for Enhanced Voice over WLAN Performance," in *Proceedings of the 2nd International Workshop on Wireless Network Measurement (WiNMee '06)*, pp. 1–7, Boston, Mass, USA, April 2006.

[20] H. Zhang, A. Arora, and P. Sinha, "Learn on the fly: data-driven link estimation and routing in sensor network backbones," in *Proceedings of the 25th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '06)*, pp. 1–12, Barcelona, Spain, April 2006.

[21] "ORBIT Project: Open-Access Research Testbed for Next-Generation Wireless Networks," http://www.orbit-lab.org/.

[22] "TAP Project: Transit Access Points Project," http://taps.rice.edu/index.html.

[23] P. Liu, Z. Tao, Z. Lin, E. Erkip, and S. S. Panwar, "Cooperative wireless communications: a cross-layer approach," *IEEE Wireless Communications*, vol. 13, no. 4, pp. 84–92, 2006.

[24] ANSI/IEEE Std 802.11, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 1999 Edition, 1999.

[25] A. Kamerman and L. Monteban, "WaveLAN-II: a high-performance wireless LAN for the unlicensed band," *Bell Labs Technical Journal*, vol. 2, no. 3, pp. 118–133, 1997.

[26] "Official Website of Cooperative MAC Implementation," http://eeweb.poly.edu/coopmac/.

[27] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *Proceedings of the 22nd IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 2, pp. 836–843, San Franciso, Calif, USA, March-April 2003.

[28] S. Xu and T. Saadawi, "Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks," *Computer Networks*, vol. 38, no. 4, pp. 531–548, 2002.

[29] G. Berger-Sabbate, A. Duda, O. Gaudoin, M. Heusse, and F. Rousseau, "Fairness and its impact on delay in 802.11 networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 5, pp. 2967–2973, Dallas, Tex, USA, November-December 2004.

[30] "VLC Media Player," http://www.videolan.org/vlc/.

[31] T. Korakis, Z. Tao, S. Makda, B. Gitelman, and S. S. Panwar, "It is better to give than to receive—implications of cooperation in a real environment," in *Proceedings of the 6th International IFIP-TC6 Networking Conference on Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet (NETWORKING '07)*, vol. 4479 of *Lecture Notes in Computer Science*, pp. 427–438, Atlanta, Ga, USA, May 2007.

[32] A. Sharmat, V. Gelaras, S. R. Singh, T. Korakis, P. Liu, and S. S. Panwar, "Implementation of a cooperative MAC protocol using a software defined radio platform," in *Proceedings of the 16th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN '08)*, pp. 96–101, Chij-Napoca, Transylvania, September 2008.