Implementation of a Cooperative MAC protocol using a Software Defined Radio Platform

Ankit Sharma†, Vikas Gelara†, Shashi Raj Singh◇, Thanasis Korakis◇, Pei Liu◇, Shivendra Panwar◇

◇ Department of Electrical and Computer Engineering, Polytechnic University, Brooklyn, NY 11201
† Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India 208016

e-mail: ankit.iitk1@gmail.com, vikas.gelara@gmail.com, ssingh08@poly.edu, korakis@poly.edu, peiliu@gmail.com, panwar@catt.poly.edu

***Abstract*—Cooperation in wireless networks has shown significant performance gains in comparison to legacy wireless networks. Cooperative wireless protocols achieve such efficiency by enabling cooperation among nodes to exploit spatial diversity. CoopMAC is a Medium Access Control (MAC) protocol that enables cooperation by using an intermediate node as a helper to a direct communication under poor channel conditions. The helper is typically located in a position where it experiences a good channel with both the source and destination. Therefore, it increases the efficiency of the communication by forwarding a packet from the source to the destination using high transmission rates. In an earlier attempt, we demonstrated the benefits of cooperation at the MAC layer by implementing the CoopMAC protocol using an open source wireless driver platform. However, due to some limitations posed by the hardware, the full potential of the protocol could not be explored. In this paper, we proceed with a complete implementation of the cooperative MAC protocol using an OFDM based software defined radio (SDR) platform. We investigate the benefits of the SDR approach, describe the details of the implementation, as well as the experiments we run in order to evaluate the protocol. Experimental results show that CoopMAC can easily be implemented and can lead to a significant improvement in the performance of wireless networks.**

I. Introduction

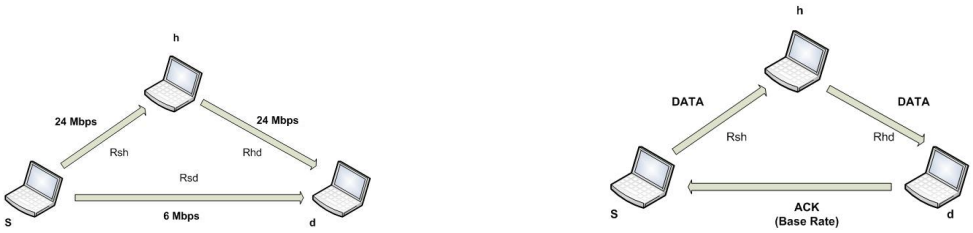
Legacy wireless networks such as IEEE 802.11 use contention based protocols to access the wireless channel. Once a node occupies the channel, other proximate nodes must wait for the end of the current transmission before they contend again for the medium. In networks where nodes have a multi-rate capability, the transmission rate is adjusted based on the channel quality. Therefore, the poorer the channel is, the lower the transmission rate that should be used. Such functionality may lead to prolonged use of the channel by a node that experiences a bad channel, keeping other nodes idle when it captures the medium, and therefore decreasing the overall throughput of the network. A possible solution to this problem is the use of neighboring stations as relays to reduce channel time usage and take advantage of the spatial diversity in the network to boost the rate.

Following this philosophy, a cooperative MAC protocol called CoopMAC [1] was designed to enable cooperation among nodes at the MAC level. The protocol allows a source node that experiences a poor channel with the destination to choose an intermediate node as a *helper* to assist with the transmission. A helper that experiences good channel conditions with both the source and the destination contributes in the transmission by forwarding the packet from the source to the destination. In particular, the source, instead of sending its packet directly to the destination at a low transmission rate, forwards the packet to the helper using a high rate. The helper receives the packet and immediately forwards it to the destination again using a high rate. In this way, a slow one hop transmission is replaced by two fast hop transmissions, boosting the performance of the network. Moreover, under the CoopMAC protocol the receiver receives two copies of the same packet. It may not be able to decode the first hop packet (since it is transmitted at a rate sustainable only at the helper), however, it can store a copy of the signal and by combining it with the copy received in the second hop using cooperative methods, it can further improve the packet error rate (PER) or, alternatively, allow an increase in the transmission rate in the second hop [2].

Given the simplicity as well as the efficiency of CoopMAC, we decided to implement it in order to study the performance of cooperative schemes in a real environment. We initially attempted to implement the mechanism using a Linux open source driver platform and commercial WiFi cards [3]. The results of this work clearly showed the high efficiency of the Cooperative MAC scheme. However, the full potential of the protocol could not be explored due to some limitations posed by the hardware of the wireless card. Therefore we focused our implementation efforts on a more flexible platform. This new platform is a software defined radio(SDR) called WARP [4]. WARP is a Xilinx Virtex-2 FPGA based platform with embedded PowerPCs. Its flexibility lies in the fact that it allows protocol development from scratch in both the MAC and the PHY layer. This flexibility as well as the sophisticated design that allows realistic PHY rates (up to 36 Mbps) motivated our decision to use WARP for a complete implementation of CoopMAC for the evaluation of its full potential.

In this paper we present the implementation of CoopMAC using the WARP platform. The challenges we faced here were totally different than those in the open source drivers approach. However, using the new platform we are able to implement the functionalities that could not be implemented in our earlier attempt. Therefore, a description of the WARP platform as well as the benefits of this implementation against the earlier one is provided. Finally, a set of experimental setup details and their results are discussed that show enhanced performance of

This work is supported in part by the National Science Foundation *(NSF)* under award 0520054 and 0722868, and the New York State Center for Advanced Technology in Telecommunications *(CATT)*. The work is also supported by the Wireless Internet Center for Advanced Technology *(WICAT)*, an NSF Industry/University cooperative Research Center at Polytechnic University.



(a) Advantage of CoopMAC

(b) CoopMAC operation

Fig. 1: CoopMAC's advantage and operation.

the new, more realistic implementation approach as compared to legacy CSMA/CA, as well as earlier attempt.

The rest of the paper is organized as follows. In Section II we give a brief description of the main concepts of CoopMAC that is necessary for understanding the implementation. In Section III we describe the details of WARP platform along with its implementation benefits. In Section IV we give details of CoopMAC implementation on WARP. A set of experimental setups along with the results obtained are reported in Section V. Section VI completes the paper with our final conclusions and possible future work.

II. The Cooperative MAC Protocol (CoopMAC)

A. Motivation for Cooperation

First, we present the motivation for cooperation given the multirate capability of a wireless network, as they are crucial for the understanding of the advantage of cooperation at the MAC layer. A multirate capability, as defined in the 802.11 standard, gives a station the ability to transmit packets at different rates based on channel conditions in order to support reliable communication. In general, the transmission rate is essentially determined by the path loss and instantaneous channel fading conditions. For IEEE 802.11a, for example, eight different rates, 6, 9, 12, 18, 24, 36, 48 and 54 Mbps are supported. A source station that is far away from the destination may persistently experience poor channel quality, resulting in a rate as low as 6Mbps for direct transmission over an extended period of time. If there exists some neighbor that can sustain higher transmission rates (e.g., 24Mbps and 24Mbps 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 cooperative forwarding, the aggregate network performance can derive a significant improvement, which motivates the introduction of cooperation into the MAC layer.

B. CoopMAC

Cooperation requires availability of a node that can support higher data rates to both source and destination nodes. When the MAC layer of a source node s receives a packet for transmission, it decides whether to use cooperation or direct transmission. For this it maintains a table called *CoopTable* containing information about helpers and the maximum supported rates for a destination. The CoopTable helps node s to decide whether a cooperative transmission would be more efficient than the direct transmission based on two-hop and one-hop transmission times, respectively If s decides to use cooperation, it sends a packet to a selected helper h, which in turn forwards this packet to the destination d. Destination d sends back a direct acknowledgment to the source s confirming reception.

The cooperation process is started by a three-way handshake between the source s, helper h and destination d. The source s transmits an RTS notifying helper h that it has been selected for cooperation. If helper h is free it broadcasts a helper ready-to-send(HTS) message notifying both source s and destination d of its availability. On receiving an HTS, the destination node d transmits a CTS reserving the channel for a two-hop transmission. If the HTS is lost, the channel is reserved only for a one hop transmission by the CTS.

In the case that the packet length is small, the three way handshake can be omitted. The helper is in this case recruited on the fly during the first hop transmission. The source transmits the data packet with the header indicating the MAC address of the selected helper. The helper, by checking the header of the packet, understands that it should assist in the ongoing communication and forward the packet to the receiver. Figure 1 (b) shows the data exchange among the three nodes. R_{sd}, R_{hd} denote the sustainable rates between s and d, between h and d respectively. As denoted in the figure, the ACK is sent at the base-rate directly from the destination to the source, indicating the successful reception of the packet. For a more detailed description of CoopMAC functionality the interested reader can refer to [1].

III. The Software Defined Radio Platform(WARP)

A. WARP Architecture

Wireless Open-Access Research Platform (WARP) [4] is a software defined radio platform developed by Rice University for research and educational purposes.

WARP consists of a Xilinx II Virtex Pro FPGA board with embedded Power PC processors. This provides a complete embedded programming environment for the design of physical (PHY) and medium access control (MAC) layers. In addition, it has four daughter card slots in which radio cards or customized cards can be inserted to connect to 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 2.4GHz/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 a advanced MAC protocols. WARPMAC and WARPPHY are set of functions that provide MAC type functionality and functions to access the PHY layer. Rice University provides many software resources at the WARP web site [5] including an Aloha-like MAC and a CSMA-like MAC. For our implementation we based our development on the CSMA-like MAC.

B. Why pursue the SDR approach?

As mentioned earlier we initially implemented CoopMAC on a Linux open source wireless driver platform using commercial WiFi cards. To be more specific, we implemented a version of CoopMAC using the HostAP driver [6] and Intersil chipset based WiFi cards. These implementation efforts show that CoopMAC protocol boosts the total throughput of the network by a factor of roughly two, with the most poorly served users seeing improvements by a factor of five compared with the existing WiFi technology [3]. However, 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 functionalities of the wireless card. Unfortunately, these functions are not implemented in the driver but rather in the firmware of the card. As a result, the final implementation of the scheme in HostAP was limited to an emulation of the original protocol. More specifically:

In the original protocol, the helper directly forwards the packet to the destination within a SIFS interval after the reception of the packet. However, in the open source drivers platform this is not possible. The implementation of CoopMAC is part of the driver, and therefore the packet has to reach there in order to be retransmitted. Since the packet comes to the driver, it is treated like an incoming packet from the wireless card, and is retransmitted following the legacy 802.11 MAC procedure. Therefore, the helper transmits the packet using the standard CSMA/CA contention method. This changes the original design of CoopMAC that required immediate forwarding of the packet without contention, and therefore adds significant delay. It also affects the fairness in the network, as now the packet that is being transmitted by the helper is considered as a separate packet in the network and therefore it 'uses' one of the helper's chances for transmission. In this sense the helper shares its bandwidth between its own transmissions, and the second hop transmissions of the helped source. This problem does not exist in the case of a direct transmission by the helper after a SIFS since this two hop transmission is considered by the neighbors as a seamless communication similar to the four way handshake of 802.11.

In the original protocol the ACK control packet is transmitted directly from the destination to the source. Unfortunately, this functionality could not be implemented in the open source driver platform since control packets are generated by the firmware of the card, and thus there is no way to change their functionality. Therefore, in the driver implementation we had two ACK packets. One ACK was generated from the helper to the source, and one ACK from the destination to the helper. Such a modification again changes the profile of the protocol since now 1) there is a higher overhead since we have two ACKs, and 2) the source has no control of the retransmission process in case the packet fails in the second hop.

The RTS-CTS functionality of IEEE 802.11 could not be changed (since it is controlled by the firmware) and therefore it was impossible to implement the HTS packet.

Based on the above points, we conclude that the open source driver implementation of CoopMAC missed several critical functional characteristics of the original protocol. The cooperative protocol implemented on the open source driver, is very similar to a layer 3 forwarding mechanism that takes into consideration channel quality of the next hop. In particular, it introduces more overhead and suffers from longer delays. Nevertheless, the experimental results showed significant benefits of using cooperation at 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 can be changed to any extent. WARP seemed a promising solution since it consists of a Xilinx Virtex II Pro FPGA board with embedded Power PC processors. It allows for realistic MAC and PHY layer implementations that could give PHY layer rates similar to those provided in IEEE 802.11a,g.

Using the WARP radio platform we were able to overcome all of the three limitations listed above. 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.

IV. Implementation

In our implementation of CoopMAC, 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 a transmission. The *Direct mode* is the legacy direct mode under the CSMA protocol (no cooperation) and the *Cooperative mode* which 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 the 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 CoopMAC.

A. 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*. A CoopFrame consists of two parts, the *MAC Header* and data payload. In the following list, due to limitations in space we only describe subfields of *MAC Header* that are related to our implementation:

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 the particular hop (i.e, the helper in the first hop, the final destination in the second hop).
3) *CoopDestinationID*: This is a new subfield we introduce in order to handle the forwarding process. It is used in the cooperative mode to indicate 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*: It is used to indicate the nature of the packet:
   - *DATAPACKET*: A packet that is used in direct mode.
   - *COOPPACKET*: A packet that is used in CoopMAC for the first hop transmission (source to helper).
   - *COOPFINAL*: A packet that is used in CoopMAC for the second hop transmission.
   - *ACK*: A control packet that acknowledges successful reception directly to the source.
5) *Full Rate*: This field is used to indicate the rate at which the payload of a packet is transmitted.

B. 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, *Destination Address* is the destination MAC address. The *CoopDestinationID* field is not used in this case. In case of Cooperative mode, the *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 subsection above. 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 effective rate in the two hops is higher than the direct rate.

C. 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 *PktType* field:

1) *DATAPACKET*: If the received packet is a DATAPACKET, then an ACK is transmitted back to the source node.
2) *COOPPACKET*: This is 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*: This is 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.

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.

TABLE I: CoopTable

$Destination\ MAC$	$DirectR\ (Mbps)$	$Helper\ MAC$	$R_{sh}\ (Mbps)$	$R_{hd}\ (Mbps)$
16.24. ...e2.c3	6	16.24. ...e2.c4	24	24
16.24. ...e2.c7	6	16.24. ...e2.c8	24	12
...	...	....	....	....

D. Implementation of the CoopTable

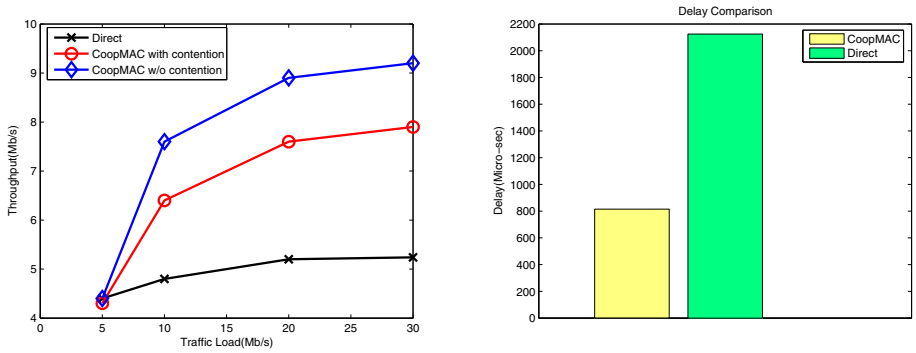
The CoopTable is an important feature of CoopMAC since it allows nodes to decide whether they should use cooperation or not. An example of the CoopTable is depicted in Table I.

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 the 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 II.

TABLE II: Supported Data Rates in WARP

$ModulationScheme$	$Metric$	$PHY\ Rate$
BPSK	1	6
QPSK	2	12
QAM16	4	24
QAM64	6	36

In this table, a higher metric value implies a higher data rate. The CoopTable is updated passively after the reception of



(a) Throughput Performance (b) Delay Performance

Fig. 2: Throughput and delay performance in scenario 1

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 itself and the destination. In this way, a node gets information about 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, 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.

E. Transmission Rates

WARP nodes support dynamic modulation per packet. 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 uses BPSK in 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 the loss of ACKs. A WARP node with the current configuration supports four PHY rates as shown in Table II.

V. Performance Evaluation

In order to study the performance of 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. An interested reader can refer to the *Wireless Implementation Testbed Laboratory* website [7] for more information.

We compared the implemented CoopMAC protocol with two protocols, the first being a CSMA approach that emulates the IEEE 802.11 MAC. We call this scheme *direct transmission*. The second protocol is an emulation of our implementation using the open source drivers, in which we implement contention for the second hop transmission as well as two ACK packets, one for each hop. We call this scheme *CoopMAC with contention*. The faithful implementation of the CoopMAC mechanism is called *CoopMAC without contention*.

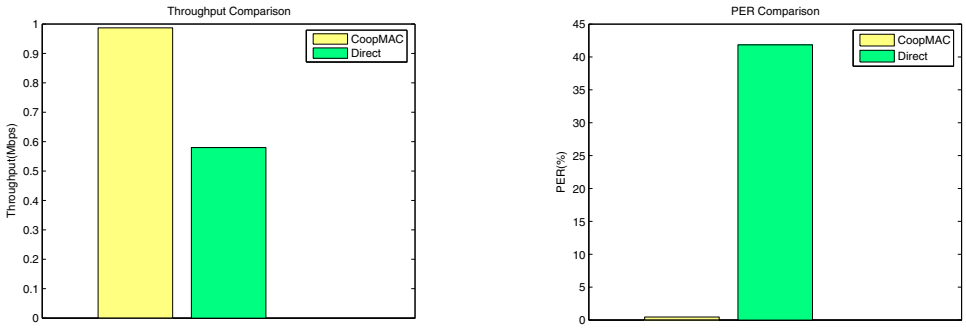
As evaluation metrics we use the total number of successful packets (throughput) as well as the average delay per packet. For our experiments we used BPSK, QPSK and QAM-16. We currently have three WARP nodes for conducting experiments. However, this small-scale testbed was enough for our purposes which 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 [8]. In all cases, the UDP packet length was 1470 bytes. Each scenario was run 10 times, each for 50 seconds, 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.

A. 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 $6Mbps$ (BPSK). The transmission via the helper for both hops was fixed at $24Mbps$ (QAM-16). Using Iperf we generated UDP traffic that was passed to the WARP nodes connected to PCs through an Ethernet cable.

In Figure 2(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 CoopMAC 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



(a) Throughput Comparison

(b) Packet Error Rate Comparison

Fig. 3: Throughput and Packet Error Rate comparison in Scenario 2.

successful forwarding. In the more faithful 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. It is worth mentioning that the performance increase when we use *CoopMAC without contention* is almost double that of *direct transmission*. This improvement is very close to the theoretical limit. Since in the case of CoopMAC we use $24Mbps$ in both the hops, we generate a channel equivalent to $12Mbps$, which has double the rate of the original direct transmission.

In Figure 2(b), we depict the delay, which is processing and transmission time for a packet on a WARP node. It is clear that the cooperative protocols decrease the transmission time of the slow node thus reducing in this way the delay.

B. 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 setup the topology of the experiment in a way 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 such that it is in the line of sight with both nodes. The direct transmission rate as well the rate for the two hops of the cooperative communication is kept at $6Mbps$ (BPSK). We run $Iperf$ and applied different traffic loads. We compared the performance of implemented scheme to that of the *direct transmission*. In Figure 3 we show the throughput and the PER for traffic load of $1Mbps$. As we can see in Figure 3(a) the throughput of CoopMAC is almost double than 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 which therefore doubles the transmission time. The explanation of this result is apparent in figure 3(b) which 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 we can increase the effective throughput of the network.

VI. Conclusions

In this paper we explored the full potential of a Cooperative MAC protocol by implementing it on a software defined radio platform. In a previous attempt to implement CoopMAC using an open source driver platform, we faced several limitations that did not allow us to implement significant features of the protocol. By using a software defined radio we were able to implement a realistic version of the cooperative scheme and to evaluate its full functionality. By conducting several experiments we compared the performance of the new implementation with the previous one as well as with a CSMA-like MAC protocol that emulated IEEE 802.11. Experimental results show that the new implementation outperforms both the above mentioned schemes due to its cooperative nature and its more efficient use of the wireless channel. As a next step, 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 the wireless network performance by enabling cooperation at the PHY layer as well.

References

[1] P. Liu, Z. Tao, S. Narayanan, T. Korakis and S. Panwar, "A Cooperative MAC protocol for Wireless LANs", IEEE Journal on Selected Areas in Communications, Special Issue on Cooperative Communications and Networking, Volume 25, No. 2, February 2007.

[2] F. Liu, T. Korakis, Z. Tao, S. Panwar, "A MAC-PHY Cross-Layer Protocol for Wireless Ad-Hoc Networks", Proceedings of IEEE WCNC 2008, Las Vegas, NV, March 2008".

[3] T. Korakis, Z. Tao, S. Makda, B. Gitelman, S. Panwar, "To Serve is to Receive: Implications of Cooperation in a Real Environment", Proceedings of Networking 2007, Atlanta, Georgia, USA, May 2007.

[4] K. Amiri, Y. Sun, P. Murphy, C. Hunter, J. R. Cavallaro, A. Sabharwal, "WARP, a Modular Testbed for Configurable Wireless Network Research at Rice"' Proceedings of IEEE SWRIF Huston, TX, May 2007.

[5] "WARP: http://warp.rice.edu/trac/."

[6] "Host AP: http://hostap.epitest.fi/."

[7] "WITEST Lab: http://witestlab.poly.edu/."

[8] "Iperf:http://dast.nlanr.net/Projects/Iperf/."