Cooperative Network Implementation Using Open-Source Platforms

Thanasis Korakis, Michael Knox, Elza Erkip, and Shivendra Panwar, Polytechnic Institute of NYU

ABSTRACT

Cooperative networking, by leveraging the broadcast nature of the wireless channel, significantly improves system performance and constitutes a promising technology for next-generation wireless networks. Although there is a large body of literature on cooperative communications, most of the work is limited to theoretical or simulation studies. To impact the next generation of wireless technologies and standards, it is essential to demonstrate that cooperative techniques indeed work in practice. This article describes two programmable cooperative communication testbeds built at Polytechnic Institute of NYU to achieve this goal. The testbeds are based on open-source platforms and enable implementation of cooperative networking protocols in both the physical and the medium access control layer. Extensive experiments carried out using the testbeds suggest not only that cooperative communication techniques can be integrated into current wireless technologies, but also that significant benefits of cooperation can be observed in terms of network throughput, delay, and video quality in real applications.

INTRODUCTION

Wireless will be the dominant mode of Internet access for end users in the near future. Technologies such as WiFi and WiMAX attempt to provide broadband wireless access. However, the bandwidth limitations of the wireless channel, interference from multiple users operating in the same band, and channel variations due to fading become bottlenecks for typical multimedia applications that require high bandwidth and an error-resilient communication medium.

Cooperation among users, by enabling wireless terminals to assist each other in transmitting information to their desired destinations, provides a good solution to the problems that current wireless technologies face. At the physical (PHY) layer, terminals overhear one another's signals, processing and retransmitting them to form a virtual antenna array. Through cooperation, it is then possible to obtain the spatial diversity benefits of multi-input multi-output (MIMO) systems without necessarily having a physical antenna array at each terminal. Cooperative communication techniques can adapt easily to a changing environment by opportunistically redistributing network resources such as energy and bandwidth. Incorporating the notion of cooperation at the medium access control (MAC) layer extends the benefits to large networks resulting in high throughput, low delay, reduced interference, low transmitted power, and extended coverage. Using a cross-layer design from the application layer down to the physical layer enables high quality multimedia transmission over cooperative wireless links.

Cooperative communications is a vibrant research area. There is extensive work in the literature on the study of cooperative schemes in the PHY layer [1–3] and to a more limited extent, in the MAC layer [4]. Almost all of this work is based on theory and simulations. The theoretical analysis and the simulation of a specific protocol or technique can give important information about performance in terms of throughput, delay, or power consumption. However, to have analytically tractable models, several simplifications of the real world environment must be made. Although simulations have the ability to incorporate more general models, the evaluation still is limited by the complexity of the simulation software and the simplification of the wireless environment. Some specific limitations of the simulation approach in depicting a real wireless network include inaccurate representation of the wireless medium, simplification of synchronization issues that occur in wireless terminals, and ignoring several aspects such as the computational overhead. Therefore, there is a significant justification for moving one step further than analysis/simulation and implementing cooperative protocols in a real wireless platform for deeper understanding of proposed schemes.

This article outlines the implementation efforts at Polytechnic Institute of NYU in building two programmable cooperative networking testbeds. The goal is to illustrate the feasibility of cooperation and to provide platforms over which different cooperative protocols can be tested. Our testbeds incorporate physical layer cooperation and a cooperative MAC layer and are amenable to cross-layer design to enable applications like cooperative

This work is supported in part by the National Science Foundation (NSF) under awards 0520054, 0722868, and 0708989, Polytechnic Angel funds, the New York State Center for Advanced Technology in Telecommunications (CATT), and the Wireless Internet Center for Advanced Technology (WICAT), an NSF Industry/University Cooperative Research Center at the Polytechnic Institute of NYU.

0163-6804/09/$25.00 © 2009 IEEE

video communications. The first testbed is based on open-source drivers, whereas the second one is built using software-defined radios (SDRs), thus providing different characteristics and abilities for flexible implementations. This article describes the functional characteristics of each platform and highlights the advantages, as well as the limitations of each approach. It also presents the details of the PHY and MAC layer implementations and our experimental results.

Overall, our testbeds represent the first fully functional, cross-layer experimental effort on cooperative networking. The open-source nature of the platforms enables further investigation and experimentation by other research teams. Our results indicate the feasibility of cooperative networking in practice and also suggest that the theoretically predicted gains in error rates, network throughput, delay, and multimedia signal quality apply to practical implementations as well. In the next section, we discuss the details of the two testbeds. Implementation of cooperative MAC protocols on both testbeds is covered in the following section, and cooperative PHY layer is discussed in the final section.

DESCRIPTION OF THE COOPERATIVE NETWORKING TESTBEDS

In setting up the cooperative networking testbeds, our focus was to use commercially available open-source platforms to enable future participation and contribution from other researchers. However, there is currently no system that simultaneously can accommodate the requirements of both PHY and MAC layer cooperation protocols, as well as cross-layer design. To combine the benefits of different platforms, we decided to build two separate, yet complementary testbeds, as shown in Fig. 1. The basic structure of each testbed along with its advantages and disadvantages is summarized in the following subsections.

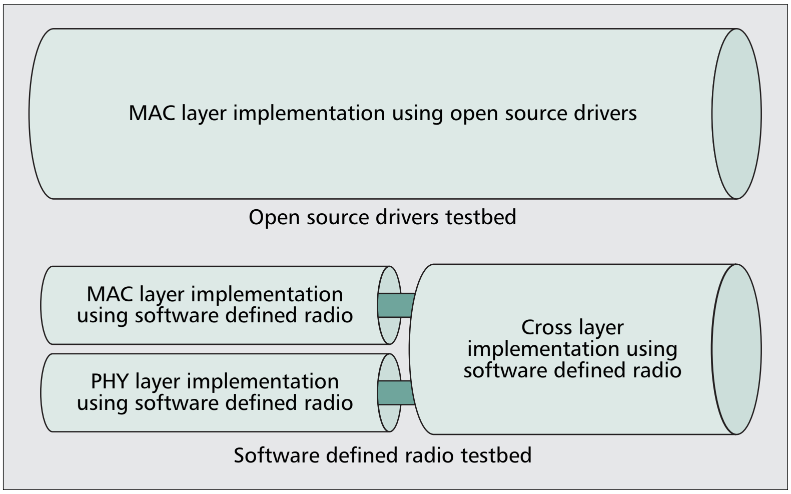

MAC layer implementation using open source drivers
Open source drivers testbed
MAC layer implementation using software defined radio
PHY layer implementation using software defined radio
Cross layer implementation using software defined radio
Software defined radio testbed

Figure 1. *Illustration of open source drivers and software defined radio cooperative testbeds.*

OPEN-SOURCE DRIVERS COOPERATIVE TESTBED

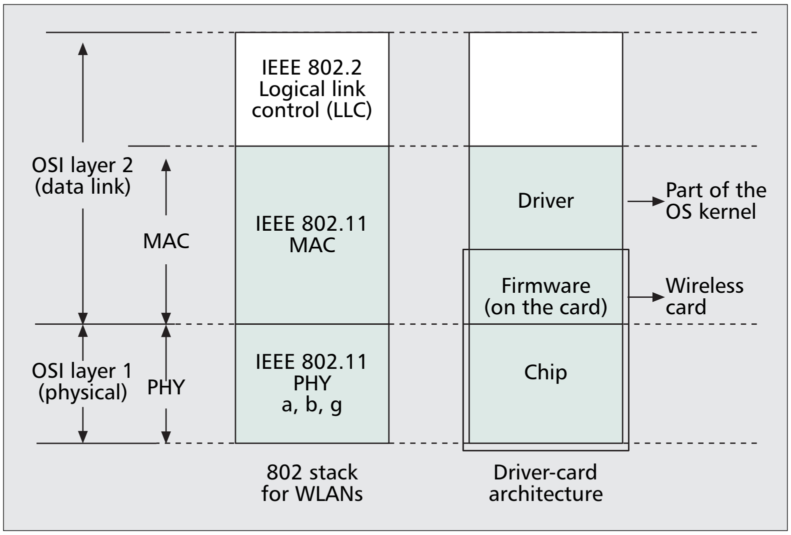
The goal of our first testbed is to implement cooperative wireless protocols focusing on the functionalities of the MAC and network layer (routing). The nodes are based on commercially available WiFi cards that have a fixed PHY layer. The MAC and network layer functionality is implemented in software based on open-source wireless drivers based on 802.11 protocol.

- *Advantages*:
  - The implementation is backward compatible with current WiFi products. This enables us to develop protocols based on a realistic detailed implementation of IEEE 802.11 and opens up the possibility of impacting WiFi standards in the near future.
  - The performance of the implemented cooperative protocols can be compared directly with commercial 802.11 solutions.
- *Disadvantages*:
  - We have access only to a portion of the MAC layer functionality. We cannot change time-sensitive functions.
  - There is no access to the PHY layer, and thus we cannot build PHY/MAC cross-layer algorithms and fully exploit the notion of user cooperation.

In the open-source drivers testbed, even though PHY layer cooperation is not exploited, cooperation at higher layers still results in many benefits for individual users and the network as shown in the next section.

A wireless driver that does not involve any time-sensitive issues (e.g., sending of an acknowledgment [ACK] after a short interframe space [SIFS] period) typically controls the functionality of the MAC layer. Thus, by changing the driver, one can change a significant part of the MAC layer and to some extent, one can build new protocols. There are three open-source Linux drivers available today: the MadWiFi driver that is based on Atheros chipsets; the HostAP driver, based on the Intersil Prism 2, 2.5, or 3 chipset; and the Intel PRO/Wireless 2100/2200/3945 drivers, based on the Intel chipset.

In a typical driver-card architecture, all the features specified in IEEE 802.11 MAC protocol are logically partitioned into two modules, according to the time criticality of each task. The lower module, which usually operates on the wireless card as a part of firmware, fulfills the time-critical functions, such as the generation and exchange of request to send/clear to send (RTS/CTS) control messages, transmission of ACK packets, execution of random back off, and so on. The other module, which normally assumes the form of the system driver, is responsible for more delay-tolerant control-plane functions, such as the management of MAC layer queue(s), the formation of the MAC layer header, fragmentation, authentication, association, and so on. For the MAC implementation, ideally we would like to have access to the firmware of the card as well and thus, have the ability to change the time-critical functionalities of the protocol. Unfortunately, the firmware is not accessible because it is proprietary. Thus, the only option is to change the part of the MAC functionality that is controlled by the driver. The basic wireless stack architecture of a typical



■ **Figure 2.** *The driver-chipset architecture for a typical IEEE 802.11 card.*

chipset is depicted in Fig. 2. Intersil and Intel chipsets follow this approach, whereas Atheros follows a slightly different architecture.

The open-source cooperative testbed is housed at the Wireless Implementation Testbed Laboratory (WITest Lab) [5] at the Polytechnic Institute of NYU. It currently consists of 20 nodes, and the network is managed by one server. Each node has a motherboard with a 1 GHz Pentium processor, 512 MB RAM, 40 GB of local disc, and the appropriate input-output interfaces. It has two mini peripheral-component interconnect (PCI) slots, where mini PCI wireless cards, based on one of the chipsets mentioned above, can be inserted.

Software Defined Radio Cooperative Testbed

The goal of the second testbed is to build an open-source architecture for rapid prototyping of PHY and MAC layers by leveraging existing open-source radio platforms. The testbed is developed by first modifying existing reference designs at the PHY and MAC layers separately. We then use the developed algorithms as a starting point for designing a joint cooperative PHY/MAC layer.

- *Advantages*:
  - –We have the flexibility to implement a cooperative PHY layer and build PHY/MAC cross-layer protocols, thus obtaining a more complete cooperative implementation.
  - –The benefits of cooperation at different settings, independent of a particular environment or standard, can be established.
- *Disadvantages*:
  - –We can build only simplified versions of MAC protocols used in the standards because no software exists with detailed implementation of any standard.
  - –The hardware places specific limitations on the system performance. For example, some of the SDR platforms that we tested limit the minimum time between two sequential transmissions to a period longer than the one in commercial 802.11 cards. Therefore, it is not possible to compare our protocols directly with commercial solutions. To provide comparisons with standard MAC protocols such as 802.11, we must build emulations of the standards, based on the same hardware architecture.

To accelerate deployment, we leverage two existing SDR platforms, namely the wireless open-access radio platform (WARP) from Rice University [6] and the GNU/universal software radio peripheral (USRP) [7]. The SDR testbed is housed within the Wireless Information Systems Laboratory (WISL) at the Polytechnic Institute of NYU and consists of six nodes of each technology.

The WARP system uses a field programmable gate array (FPGA) to process all symbol-rate and bit-rate data. This approach requires coding all of the modulation, demodulation, and communication algorithms directly inside the FPGA. The platform can process signals in excess of 20 MHz of radio bandwidth. The WARP system includes a baseband processing board, radio frequency (RF) radio board, and the open-source code. WARP nodes are based on a Xilinx Virtex-4 FPGA that has two embedded PowerPC processor cores. The Virtex-4 provides dedicated digital signal processing (DSP) slices, hardware blocks designed specifically for high-speed multiply-accumulate and other DSP operations. The WARP FPGA board provides four daughtercard slots, each wired to a large number of dedicated FPGA input/output (I/O) pins. These slots can house peripheral wireless cards. The four slots are functionally identical, enabling users to mount the combination of peripheral cards that best suits their application. The PHY layer radio platform, based on WARP nodes, provides a wideband radio front-end covering the unlicensed frequency bands at 2.4 GHz and 5.6 GHz. An advantage of the WARP system is that high symbol rates can be achieved while performing all signal processing inside the FPGA. However, the disadvantage of WARP is a steeper learning curve requiring knowledge of MATLAB Simulink and Xilinx System Generator. WARP also provides a framework called WARPLAB that enables signal processing and waveform generation in MATLAB and uses WARP boards to transmit and receive only these waveforms. Although WARPLAB is useful for rapid prototype design and over-the-air testing, the simplicity of implementing the PHY layer in MATLAB comes at the expense of low transmission rates.

GNU Radio [7] is an open-source software toolkit for building software radios, generally independent of the hardware. The GNU software is easily configured with the USRP, available from Ettus Research [8]. We chose to use GNU radio due to its popularity and ease of programming. However, GNU radio does not support 802.11 due to slow data transfer over the USB port to a personal computer. Hence, in this article, we focus on WARP, which can handle high data rates, rapid automatic gain control (AGC) for the received packets, and provides a good framework to integrate the PHY and the MAC layers.

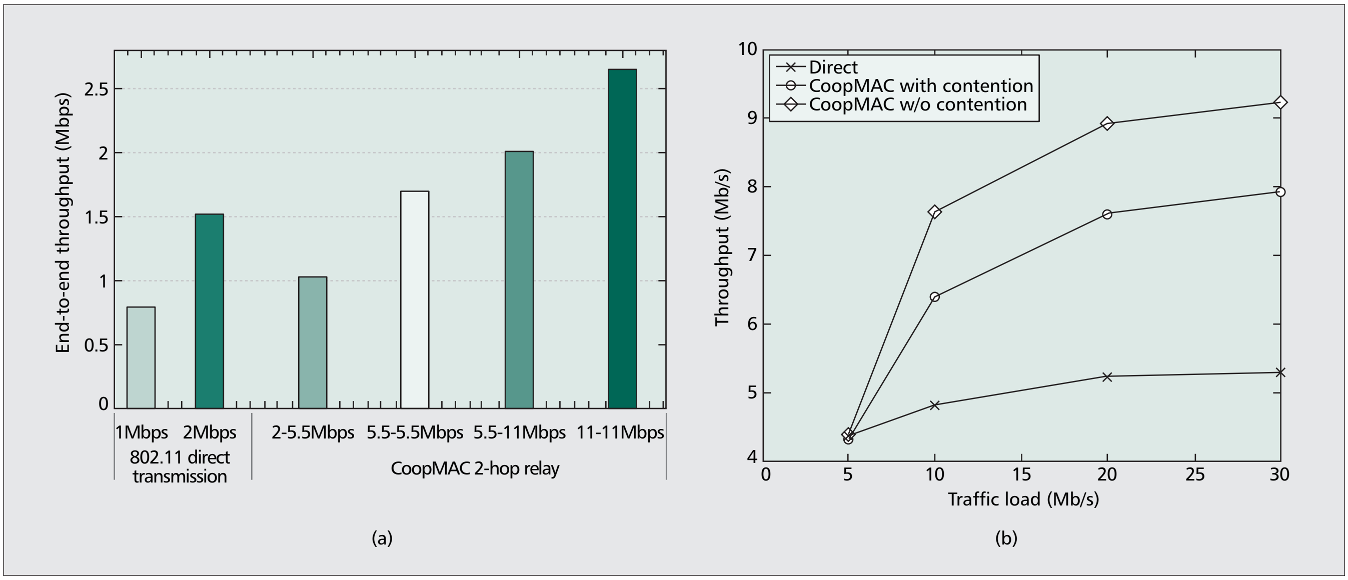


Figure 3. *Throughput comparison for UDP traffic: a) open source drivers platform; b) SDR platform.*

IMPLEMENTATION OF MAC LAYER COOPERATIVE PROTOCOLS

Our MAC layer implementations include several cooperative MAC schemes both in the open-source drivers testbed, as well as in the SDR testbed. In the open-source drivers platform, we implement cooperative MAC protocols for unicast and multicast applications based on both HostAP and MadWiFi. Although the implementations are realistic and fully functional, some of the details of the protocols must be omitted due to platform limitations. Therefore, we also proceed with the SDR platform, in particular the WARP nodes. The details and the codes for all of our MAC layer implementations and our experimental results can be found at the WiTest Lab Web site [5]. In this article we focus on the implementation of a unicast cooperative MAC protocol called *CoopMAC* [9] in both testbeds.

THE COOPERATIVE MAC PROTOCOL

A cooperative MAC protocol called *CoopMAC* developed in [9] enables participation of a third node (called the relay or the helper) to facilitate communication between a source and a destination. In conventional wireless networks, when a source experiences a bad channel toward its destination, it lowers its modulation scheme and coding rate to maintain a desired level of reliability. In CoopMAC, the source can use an intermediate relay that experiences a relatively good channel with both the source and the intended destination. Instead of sending its packets directly to the destination at a low transmission rate, the source transmits at a high rate to the relay, and then the relay forwards the packet to the destination again at a high rate. By using a two-hop alternative path via the relay, which collectively is faster than the original direct link, the protocol can take advantage of the spatial diversity between the three nodes.

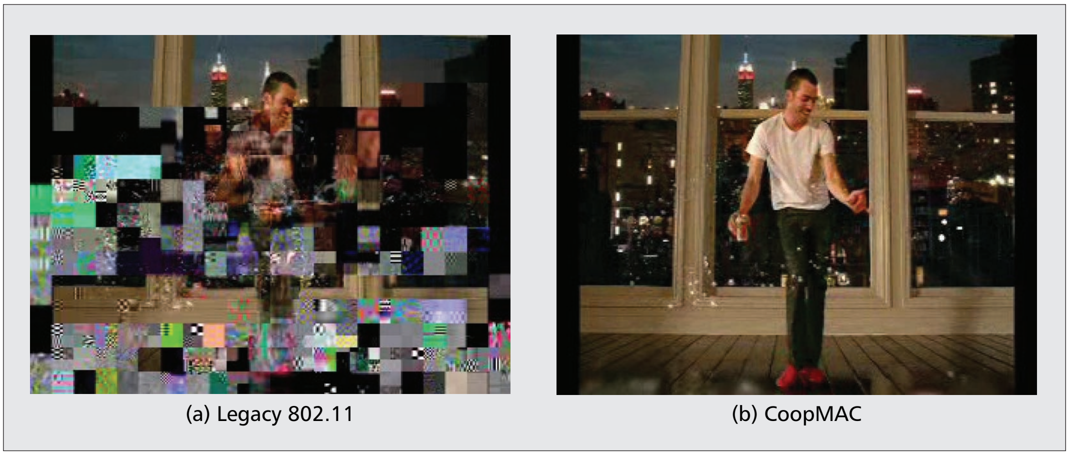
The basic functionality of CoopMAC is described as follows. The source chooses a suitable relay, based on the two-hop sustainable rate, and this decision is broadcast through the RTS in the control plane. The relay indicates its availability to participate by transmitting a new control packet called helper ready-to-help (HTS). The destination completes the three-way handshake by sending a CTS packet. In the data plane, the source transmits the packet in the first hop, and the relay retransmits the packet over the second hop. Each node maintains a table called a *CoopTable*, containing information about available relays and the maximum supported rates for the two-hop transmission toward a destination. For further details, see [9].

IMPLEMENTATION OF COOPMAC USING OPEN-SOURCE DRIVERS

We discuss here only our implementation using HostAP; MadWiFi efforts can be found at the WiTest Lab Web site [5]. Because the implementation of CoopMAC requires changes to both time-critical and delay-tolerant functions, unfortunately, the inaccessibility to firmware forces some compromises and alternative approaches in implementation. For illustrative purposes, three main circumventions are outlined below. An implication of these circumventions is that a faithful implementation of CoopMAC potentially outperforms the one demonstrated in this section.

- *Suspension of three-way handshake*: The strict sequence of RTS and CTS packets was hardwired in the firmware of 802.11 cards; therefore, an insertion of HTS as required by CoopMAC becomes impossible at the driver level. As an alternative, we suppress the use of control packets before data transmission.
- *Unnecessary channel contention for relayed packet*: After the channel access has been allocated to the source, the CoopMAC protocol suggests that the relay should forward the packet a *SIFS* time after its reception, without any additional channel contention.

As anticipated, the user perception is poor for video transmission in the 802.11 network because noticeable freezes and distortions occur frequently. Meanwhile, the video is smooth and artifact-free when CoopMAC is enabled.


(a) Legacy 802.11

(b) CoopMAC

■ **Figure 4.** *Open source drivers demo snapshot: a) legacy 802.11; b) CoopMAC.*

However, the ability to do this is controlled by the firmware and cannot be changed. As a result, we compromised on this approach by inserting channel contention for the relayed packet on the second hop.
- *Duplicate ACK*: Each successful data exchange in the original CoopMAC protocol involves only one ACK message, which is sent from the destination to the source directly. Because the acknowledgment mechanism is an integral function of firmware, it is impossible to suppress the unnecessary ACK message generated by the relay for each packet it forwards on behalf of the source. Therefore, the unwanted ACK from the relay must be tolerated instead of being eliminated.

Based on the CoopMAC implementation described above, we ran extensive experiments under different network topologies and different traffic loads. Experimental results in different topologies, with different number of nodes (up to nine) clearly show significant improvement in terms of throughput, delay, and jitter over legacy 802.11. As an example, in Fig. 3a, we show a throughput comparison between CoopMAC and IEEE 802.11 for a topology that consists of three nodes: a source, a destination, and a potential relay. We generate traffic from the source to the destination using *iperf*. Different first- and second-hop relay rates are shown on the horizontal axis. The figure suggests that CoopMAC performs efficiently in a real implementation and can give up to three-fold throughput improvements compared to IEEE 802.11. Further details of the implementation and the experimental results can be found in [10].

The above results are obtained in experiments that rely on large file-transfer traffic patterns. To obtain more insights into the performance of CoopMAC, we also consider video applications. To this end, we transmit a video clip in a testbed that consists of a source, a destination, and a relay. A video server is placed at the source and constantly streams a commercial video clip, while the destination plays the video. We assume that the source has a bad channel with the destination, whereas the relay has a good channel with both nodes. Therefore, when 802.11 is used, the direct transmission rate is set to 1 Mb/s, whereas the transmission rates between the source and the relay and the relay and the destination are both 11 Mb/s.

As anticipated, the user perception is poor for video transmission in the 802.11 network because noticeable freezes and distortions occur frequently. Meanwhile, the video is smooth and artifact-free when CoopMAC is enabled. Figure 4a and Fig. 4b provide snapshots of the video seen at the destination for 802.11 and CoopMAC, respectively. These two snapshots are typical and reveal the substantial difference between the video quality that these two different protocols can deliver.

COOPMAC USING SOFTWARE-DEFINED RADIO APPROACH

Whereas the open-source driver testbed enables backward compatibility with the IEEE 802.11 standard, the SDR testbed offers an environment where we can overcome the limitations discussed earlier. Furthermore, the SDR testbed enables us to design MAC and PHY cross-layer schemes jointly.

To study the performance of CoopMAC in the WARP implementation, we conducted several experiments measuring the total number of successful packets (throughput), as well as the average delay per packet. Details, as well as extensive experimental results, can be found in [11]. Here we describe the basic scenario that gives a clear indication of the performance gains. We compare the implemented CoopMAC protocol with two protocols, the first being a carrier-sense multiple-access (CSMA) approach that emulates the IEEE 802.11 MAC. We call this scheme *direct transmission*. The second protocol is an emulation of our implementation in 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 accurate implementation of the CoopMAC mechanism is called *CoopMAC without contention*.

In Fig. 3b, we give the throughput comparisons of the above three schemes for a simple network of a source, a destination, and a relay, as well as for User Datagram Protocol (UDP) traffic. We assume that the source-destination channel is poor, and the relay is located in between. Direct transmission occurs at a data rate of 6 Mb/s (binary-phased shift keying [BPSK]), and the transmission through the relay for both hops is fixed at 24 Mb/s (16-QAM). The figure validates the open-source driver results by showing large throughput improvement of CoopMAC over IEEE 802.11. Moreover, it shows that the WARP implementation further improves the performance over open-source drivers significantly, by eliminating the overhead generated due to contention in the second hop and the double ACK per transmission.

IMPLEMENTATION OF PHY-LAYER COOPERATIVE PROTOCOLS

The spatial diversity provided by user cooperation can be exploited at the PHY layer by enabling the destination to combine signals coming from the source and the relay. This results in higher data rates and a more robust communication system [1–3]. It has been shown, in theory and simulations, that system performance can be further improved when the attributes of PHY- and MAC-layer protocols are combined in a cooperative configuration [4].

This section summarizes some of our PHY-layer implementation results carried out using the WARP SDR testbed. These results provide the basis for our next phase of research involving a combined PHY-MAC implementation. We emphasize that our SDR testbed constitutes one of the few cooperative PHY-layer implementation efforts that goes beyond techniques that require the simple selection of a relay node [12] and provides a platform where source and relay-signal combination at the destination is possible.

The two most popular PHY-layer cooperative protocols are amplify-and-forward and decode-and-forward [3]. These two techniques form the building blocks for most of the known cooperative schemes. Whereas the amplify-and-forward approach was investigated from an implementation perspective in [13], we focused our implementation efforts on a type of decode-and-forward technique, also known as *cooperative coding*, [14, 15], which fits more naturally into a cross-layer perspective.

COOPERATIVE CODING

A rate *k*/*n* channel encoder generates n coded bits for every *k* information bits. In a standard communication system, all coded bits are transmitted directly by the source. In cooperative coding [15], transmission is divided into two slots. In the first slot, the source punctures the code and transmits only a portion of the coded bits. These bits are received both by the relay and the destination. The destination temporarily stores the received data from the source. The relay attempts to decode the source information (successful decoding is possible, as long as the rate of the punctured code is at least one) and then re-encodes to obtain the coded bits. In the second time slot, the relay transmits only the coded bits that the source left off. The destination multiplexes the two received data streams into a single stream and passes through the channel decoder.

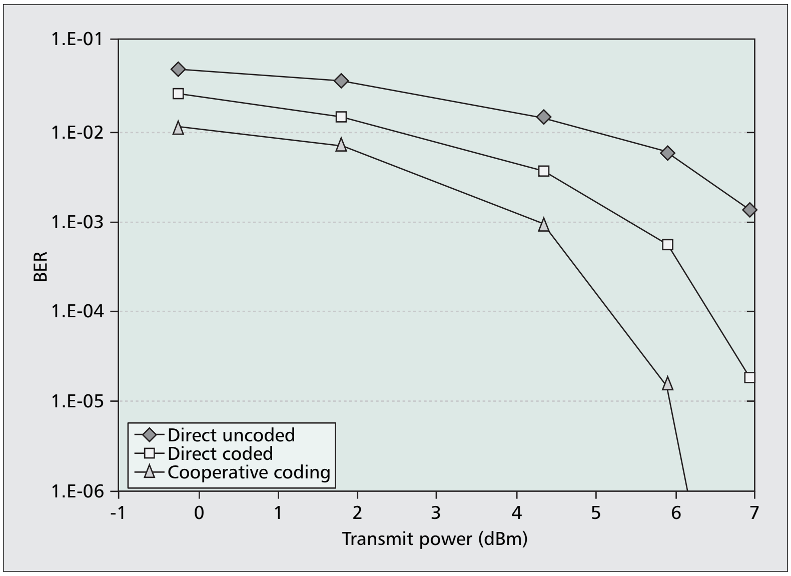
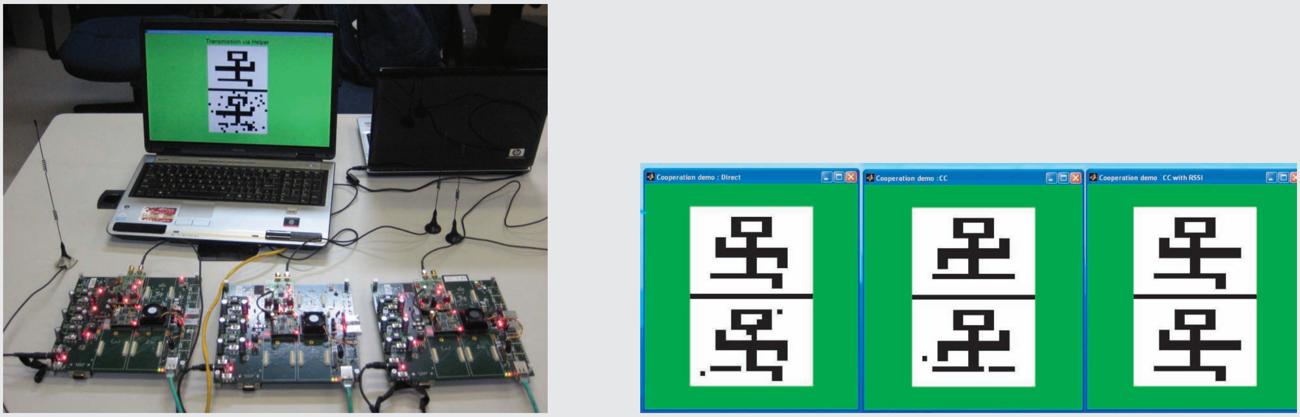


Figure 5. *Measured BER for a three-node cooperative coded system.*

Compared with direct communication from the source to the destination, in the cooperative system the total time and frequency resources remain unchanged. When cooperative coding is employed, the destination still receives the same number of coded bits; however, now, part comes from the source; the remaining part comes from the relay, thereby resulting in spatial diversity. The resulting diversity and code design criteria are discussed in [15].

IMPLEMENTATION OF COOPERATIVE CODING USING SOFTWARE-DEFINED RADIO

The operation of the three-node cooperative-coded system was verified using the WARP hardware and associated WARPLAB software running under Mathworks MATLAB R2006b. A channel code of rate 1/2 with a constraint length of 5 and a generator polynomial matrix of (37 33) was used. The feedback connection polynomial of the encoder is 37 [16]. The destination uses a hard-decision decoding algorithm that enables the use of the basic WARPLAB reference design without substantial modification. The bit error rate (BER) performance is measured on the WARP platform for three different test cases; a single uncoded link between source-to-destination, a single-coded link between the source-to-destination, and a cooperative-coded system using the relay node as described above. The same channel encoder/decoder is used for the two-coded configurations. Figure 5 shows the measured BER for the three system configurations. For the cooperative case, the transmit power level of the relay is set equal to the transmit power of the source. The transmit power levels are adjusted over the range of 0 dBm to +7 dBm, and the BER is measured for each configuration. In all cases, the receive gain was fixed at each node. The relay node is physically positioned between the source and destination nodes, resulting in very high-quality links between the source-to-relay and the relay-to-destination paths. For example, the measured BER



(a)

(b)

■ **Figure 6.** *PHY layer demo on WARP platform: a) demo setup; b) demo snapshots from phases 1, 2, and 3, respectively.*

from the source-to-relay was less than 10^{-7}, and the relay-to-destination was less than 10^{-4}. As shown in the figure, the measured BER for the cooperative-coded system outperforms the system that has a single-coded link from the source-to-destination. As expected, the uncoded system operating directly from source-to-destination shows the lowest performance of all three test cases. The measured BER performance trends exhibit a remarkable similarity to the ones found by analysis and simulations [15]. Note that the measured performance for this cooperative system includes the effects of forwarding errors in the relay link.

To further demonstrate the advantage of cooperation in the PHY layer, we built a set up for transmission of a MATLAB video clip over the cooperative-coded system described above [17]. The demo set up is shown in Fig. 6a. The source continuously transmits frames of a video clip over the air. The destination receives the frames and displays the clip in MATLAB. The demo consists of three sequential phases. After all three phases are completed, the cycle starts from the beginning. To automate the phase transition, a MATLAB script is written that switches sequentially from mode to mode every 30 seconds. The phases can be summarized as follows, and the results are illustrated in Fig. 6b:

- **Phase 1 (direct mode)**: Non-cooperative network — source directly communicates with the destination. The quality of the received video is bad; noticeable distortions occur frequently.
- **Phase 2 (multihop)**: Relay is used for forwarding, but only the relay signal is used for decoding at the destination. The quality of the video is good.
- **Phase 3 (cooperative mode)**: The destination combines the signals from the source and the relay; the received signal strength indicator (RSSI) weighting is used in decoding. As expected, this results in the best quality of video.

The reported BER measurements and the above demo were based on hard-decision decoding; our next step is to implement optimum soft-decision decoding algorithms. We also plan to migrate WARPLAB implementations to the WARP FPGA to enable operation at high data rates.

CONCLUSIONS

In this article, we describe the implementation of cooperative wireless networking using two testbeds, as well as the results of the experiments we performed on the testbeds. We discuss the challenges that arose in implementation and the solutions we devised to address them. In addition to the MAC layer implementation, we present one of the first efforts on the implementation of cooperative-coding schemes in the PHY layer. Through the development of these schemes in a real environment, we show clearly the significant benefits of cooperation in wireless networks. Our ongoing work includes combining the MAC- and PHY-layer implementations into a unified cross-layer scheme for multi-node cooperative networks.

REFERENCES

[1] A. Sendonaris, E. Erkip, and B. Aazhang, "User Cooperation Diversity — Part I: System Description," *IEEE Trans. Commun.*, vol. 51, Nov. 2003, pp. 1927–38.
[2] A. Sendonaris, E. Erkip, and B. Aazhang, "User Cooperation Diversity — Part II: Implementation Aspects and Performance Analysis," *IEEE Trans. Commun.*, vol. 51, Nov. 2003, pp. 1939–48.
[3] J. N. Laneman, D. Tse, and G. W. Wornell, "Cooperative Diversity in Wireless Networks: Efficient Protocols and Outage Behavior," *IEEE Trans. Info. Theory*, vol. 50, Dec. 2004, pp. 3062–80.
[4] P. Liu *et al.*, "Cooperative Wireless Communications: A Cross-Layer Approach," *IEEE Commun. Mag.*, special issue on MIMO systems, vol. 13, Aug. 2006, pp. 84–92.
[5] "Witest Lab: Wireless Implementation Testbed Laboratory at Polytechnic University," http://witestlab.poly.edu/wiki
[6] WARP Project: Wireless Open-Access Research Platform, Rice University, http://warp.rice.edu/
[7] GNU Radio — The GNU Software Radio, http://www.gnu.org/software/gnuradio/
[8] Ettus Research, http://www.ettus.com/
[9] P. Liu *et al.*, "CoopMAC: A Cooperative MAC for Wireless LANs," *IEEE JSAC*, special issue on Cooperative Communications and Networking, vol. 25, Feb. 2007, pp. 340–54.
[10] T. Korakis *et al.*, "To Serve is to Receive Implications of Cooperation in a Real Environment," *Proc. Networking '07*, Atlanta, GA, May 2007.

[11] A. Sharma *et al.*, "Implementation of a Cooperative MAC Protocol Using a Software Defined Radio Platform," *Proc. IEEE LANMAN*, Cluj-Napoca, Romania, Sept. 2008.

[12] A. Bletsas and A. Lippman, "Implementing Cooperative Diversity Antenna Arrays with Commodity Hardware," *IEEE Commun. Mag.*, vol. 44, Dec. 2006, pp. 33–40.

[13] P. Murphy, A. Sabharwal, and B. Aazhang, "Building a Cooperative Communications System," under review, http://warp.rice.edu/ trac/ wiki/JSAC_CooperativeComm

[14] M. Janani *et al.*, "Coded Cooperation in Wireless Communications: Space-Time Transmission and Iterative Decoding," *IEEE Trans. Signal Proc.*, vol. 52, Feb. 2004, pp. 362–71.

[15] A. Stefanov and E. Erkip, "Cooperative Coding for Wireless Networks," *IEEE Trans. Commun.*, vol. 52, no. 9, 2004, pp. 1470–76.

[16] "Polynomial Description of a Convolutional Encoder," Mathworks MATLAB Help Reference, Communications Toolbox, v. R2006a.

[17] S. Singh *et al.*, "A Demonstration of Cooperative Coding Schemes Using the WARP Platform," *Mobicom — WinTECH*, San Francisco, CA, Sept. 2008.

BIOGRAPHIES

THANASIS KORAKIS (korakis@poly.edu) received his B.S. and M.S. degrees in informatics and telecommunications from the University of Athens, Greece, in 1994 and 1997, respectively, and his Ph.D. degree in computer and communication engineering from the University of Thessaly, Greece, in 2005. Currently, he is a research assistant professor of electrical and computer engineering at the Polytechnic Institute of New York University (NYU). He is also affiliated with the New York State Center for Advanced Technologies in Telecommunications (CATT) and the Wireless Internet Center for Advanced Technology (WICAT), Polytechnic University, where he was a research scientist from 2005 to 2006. His research focuses on access-layer protocols, cooperative networks, directional antennas, quality-of-service provisioning, and network management. In the summer of 2004, he was a visiting researcher in the Computer Science and Engineering Department, University of California, Riverside.

MICHAEL KNOX (mikeknox@poly.edu) received his B.S., M.S., and Ph.D. degrees in electrical engineering from Polytechnic University, Brooklyn, New York, in 1983, 1988, and 2008, respectively. Currently, he is an associate industry professor at the Polytechnic Institute of NYU, Brooklyn, New York. His research interests are in wireless communications, software-defined radio, analog circuit design, and RF/microwave antenna design.

ELZA ERKIP (elza@poly.edu) received her M.S and Ph.D. degrees in electrical engineering from Stanford University and her B.S. degree in electrical and electronic engineering from the Middle East Technical University, Turkey. Currently, she is an associate professor of electrical and computer engineering at the Polytechnic Institute of NYU. She received the NSF CAREER Award in 2001, the Communications Society Rice Paper Prize in 2004, and the ICC Communication Theory Symposium Best Paper Award in 2007. She co-authored a paper that received the ISIT Student Paper Award in 2007. Currently, she is an Associate Editor of *IEEE Transactions on Communications*, a Publications Editor of *IEEE Transactions on Information Theory*, and Co-Chair of the GLOBECOM 2009 Communication Theory Symposium.

SHIVENDRA S. PANWAR (panwar@catt.poly.edu) received a B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1981, and M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, in 1983 and 1986, respectively. He joined the Department of Electrical Engineering at the Polytechnic Institute of New York, Brooklyn (now the Polytechnic Institute of NYU), where he is a professor in the Electrical and Computer Engineering Department. Currently, he is the director of the New York State Center for Advanced Technology in Telecommunications (CATT). His research interests include the performance analysis and design of networks. Current work includes video systems over peer-to-peer networks, switch performance, and wireless networks. He spent the summer of 1987 as a visiting scientist at the IBM T. J. Watson Research Center, Yorktown Heights, New York, and was a consultant to AT&T Bell Laboratories, Holmdel, New Jersey. He has served as the secretary of the Technical Affairs Council of the IEEE Communications Society (1992–1993) and is a member of the Technical Committee on Computer Communications. He is a co-editor of two books: *Network Management and Control*, vol. 2, and *Multimedia Communications and Video Coding* (Plenum, 1994 and 1996, respectively) and co-author of *TCP/IP Essentials: A Lab-Based Approach* (Cambridge University Press, 2004). He is a co-recipient of the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems.