Research Article

Distributed Testbed for Coded Cooperation with Software-Defined Radios

Changcai Han and Si Li

School of Electronic Information Engineering, Tianjin University, Tianjin 300072, China

Correspondence should be addressed to Changcai Han; cchan@tju.edu.cn

Received 13 July 2013; Revised 19 October 2013; Accepted 24 October 2013

Copyright © 2013 C. Han and S. Li. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Evaluating the performance of the cooperative communications using the hardware testbed in the real environment is critical for system design and promoting the potential practical applications in the future, considering the disadvantages of the theoretical analysis and simulation methods. In this paper, a distributed hardware testbed using software-defined radios is designed and implemented for cooperative communications, and the performance of two coded cooperation schemes with Turbo codes is evaluated in the physical layer. For the testbed, the fundamental point-to-point link uses the differential binary phase shift keying (DBPSK) constellations, and the physical (PHY)/media access (MAC) layer frames for the performance evaluation are devised. Furthermore, a distributed node synchronization scheme is implemented and the source node and relay node work in the time division protocol without any centralized controlling. Finally, two popular coded cooperation schemes with Turbo codes for the three-node model and the four-node model are tested in the real indoor environment. The experiment results verify the effectiveness of the testbed for system performance evaluation in the PHY layer.

1. Introduction

Cooperative communications [1, 2] involving resource sharing among different nodes can save the resources for wireless ad hoc and sensor networks and have promising practical applications. In cooperative wireless networks, various relay protocols including amplify-and-forward (AF), decode-and-forward (DF), and coded cooperation have been investigated in the literature [3–5]. For coded cooperation, channel coding is flexibly combined with cooperative transmission protocols to improve the system performance and a variety of schemes have been designed with different channel codes such as Turbo codes, low-density parity-check (LDPC) codes, and generalized low-density (GLD) codes [4–12]. For the cooperative communications with Turbo codes, one strategy is based on rate compatible punctured codes (RCPC) and the other, called the distributed Turbo code, employs distributed encoding of Turbo codes at the different nodes [7, 8].

For most of the cooperative schemes above, the design and the performance evaluation mainly rely on theoretical analysis or computer simulations. However, the theoretical analysis and the computer simulations are usually based on some simplified physical factors, which may fail to form the accurate models of the real signal propagation environments. For the potential applications in the future systems, the performance evaluation of different cooperative transmission schemes has attracted great interest. Therefore, the hardware testbed should be developed to evaluate the performance of the cooperative communication systems in real environment [13–15].

Recently, several types of existing platforms with various complexity and flexibility are developed to evaluate the cooperative communications in the community. These testbeds are usually constructed by employing the simple commodity hardware, commodity wireless cards, digital signal processors (DSPs), field programmable gate arrays (FPGAs), or software-defined radios (SDRs), and each type has its own advantages and disadvantages [16–24]. Considering the design flexibility and the development cost, SDR is an attractive way for the development and configuration attributed to its general purpose processor [24].

For the testbeds with SDR, Universal Software Radio Peripheral (USRP) and GNU Radio are widely used, and
several cooperative communication testbeds have been developed with such platforms. The typical examples are summarized as follows. In [15], an evaluation platform is developed based on USRPs and GNU radios to implement selective cooperation by comparing the quality of signals from the sender and the relay. In [24], the authors introduced a reconfigurable testbed to evaluate the performance of cooperative communications. More specifically, they focus on the single-relay cooperation with signal combination and the synchronized multirelay cooperation. In the testbed, in order to achieve strict time synchronization which is the prerequisite for the synchronized multirelay cooperation, a single external Global Positioning System (GPS) signal to the USRP2 board at each relay node is employed in [24].

However, the more complex and effective relay protocols such as the coded cooperation are not completely investigated in the work mentioned above, for the limitation of the existing testbeds. In this paper, a cooperative communication evaluation platform based on software-defined radios is developed to support the two popular coded cooperation schemes, the distributed Turbo codes and the rate compatible Turbo codes. The experiment results verify the effectiveness of the platform and the performance of the two schemes are also compared carefully in a real indoor environment.

Moreover, the hardware testbed developed in this paper has its own superiority. For example, the distributed time synchronization is implemented for different nodes in cooperative wireless networks. For the testbed in [13], the centralized control scheme is utilized, where a single computer controls all the three different nodes in the network and performs all the baseband processing. This scheme brings some convenience for the performance evaluation and is easy for implementation. However, it could not support the network with nodes distributed in a relatively large range. Also, it brings the limitation of the sending data rate due to the computer’s resource bottleneck. Indeed, the centralized control is a popular strategy in the design of cooperative platforms based on other devices [18]. Meanwhile, in [24], a time synchronization method based on the hardware timestamp is employed to guarantee the synchronization of all the relay nodes, where a GPS clock serves as a centralized clock to control all the relay nodes. In contrast, in our platform, the synchronization of the different nodes does not work in a centralized way but works in a completely distributed way. It is implemented based on the Flooding Time Synchronization Protocol (FTSP) usually used in the wireless sensor networks [25–27]. In this scheme, each node independently calculates the skew and offset parameters of its own clock using linear regression. In the distributed scheme with FTSP, the nodes participating in the cooperative scheme can locate in a relatively large area in the real environment. Moreover, the simplified FTSP has the superiority of low communication payload, high timing precision, and good scalability to different number of nodes, compared with the traditional node synchronization protocols, such as Reference Broadcast Synchronization (RBS) protocol [28] and Timing-sync Protocol for Sensor Network (TPSN) [29]. Therefore, the FTSP is very suitable for developing an extendable evaluation platform for cooperative communications.

The rest of the paper is organized as follows. Section 2 presents the fundamentals of coded cooperation and the overall architecture of the evaluation platform. In Section 3, the design details such as the design consideration for the point-to-point link, the physical (PHY)/media access (MAC) layer frames, and the node synchronization schemes are illustrated in detail. In Section 4, the implementation parameters and simulation/experiment results are given. Conclusions are drawn in Section 5.

2. The Testbed Architecture for Coded Cooperation

2.1. Coded Cooperation with Turbo Codes. In this paper, we first address the three-node model; that is, the system includes a source $S$, a relay $R$, and a destination $D$, as shown in Figure 1. Here, the source and the relay can represent the neighbor sensor nodes, while the destination may be a sink node in the wireless sensor networks, which is relatively far from the source and relay nodes. Coded cooperation was proposed in [4, 5] and the rate compatible convolutional codes are first used for coded cooperations. Then, the Turbo codes have been integrated in this cooperation scheme. The two promising methods are the coded cooperation using rate compatible Turbo codes and the distributed Turbo codes [7, 8]. The distributed Turbo codes can also be viewed as a special case of the rate compatible codes. In this paper, we will investigate the two methods using practical hardware platform. In this part, the fundamentals will be briefly introduced.

For the coded cooperation scheme using rate compatible Turbo codes, the relay node is assumed to only act as a cooperative agent for the source node, which is a little different from that in [4]. The work procedure is depicted as shown in Figure 2. Specifically, the source node encodes the source data using a Turbo code with a relatively low code-rate (e.g., $r = 1/3$) mother code, and then the mother code is punctured to form a higher code-rate Turbo code (e.g., $r = 1/2$). Then, the punctured codewords are transmitted by the source node. In this way, the destination node and the relay node receive the punctured codewords. If the destination node is fortunate to

![Figure 1: The three-node cooperation model.](image-url)
decode the data successfully, the relay keeps silent. Otherwise, when the relay node successfully receives the source data, the relay node will help the destination node in the following way. It reencodes the decoded source data with the same encoder as the source, discards the data sent by the source, and only transmits the data punctured by the source in the first phase.

For the scheme using distributed Turbo codes, the work process is illustrated in Figure 3. The source node encodes and sends the source data using a Recursive Systematic Convolutional (RSC) code, which is the component code of a certain Turbo code. Then, the destination node and the relay node receive the transmitted RSC codewords. If the destination node successfully decodes the source data but the relay node fails to decode it, the relay node helps the source node by reencoding the decoded source data with another RSC encoder, and only transmits the newly encoded parity-check bits. Indeed, this scheme can be viewed as a special puncturing case of rate compatible Turbo codes. However, it lowers the encoding and decoding complexity of the relay node.

In the following, we also extend the three-node model to general models, including multiple relay model. In the experiments, we present the results in the three-node and four-node scenarios. For the three-node scenario, we compare the two schemes above carefully. In the four-node scenario, we explain the performance improvement using more relays. All the tests are performed on the developed testbed.

2.2. The Testbed Architecture. In this paper, we design hardware evaluation platform for the two coded cooperation schemes described above. Usually for such coded cooperation, the whole transmission procedure is divided into two phases. The sender sends the packet in the first phase and then the relay forwards it in the second phase. After receiving the two sequences from the different paths, the receiver combines them together to decode the packet. One of the main challenges in this configuration is how to combine the two copies of the signals transmitted from different paths at the destination. The schemes using rate compatible Turbo codes and distributed Turbo codes are among the most promising schemes. Therefore, in this paper, we investigate these two strategies. In the evaluation using the hardware-based testbed, the main problems are how to choose the hardware platform, how to simulate the real transmission link under fading, and how to acquire the global synchronization for all the nodes which work in the time division mode. The three aspects will be addressed in the following.

In our hardware platform, all the three types of nodes are implemented using a USRP plus a personal computer (PC). The USRP hardware, including the motherboard and the radio frequency (RF) daughter board, acts as an analog front end. A motherboard has analog-to-digital converters and digital-to-analog converters. It is implemented using the AD9862 chips. The chip includes two analog-to-digital converters (ADCs) with the precision of 12 bits and 64 Mega samples per second and two digital-to-analog converters (DACs)
with the precision of 14 bits and 128 Mega samples per second. In this paper, the RFX2400 daughter board is used, which acts as a transceiver with a peak output power of 100 mW and operates from 2.3 GHz to 2.9 GHz. The other two problems, how to simulate the real transmission link and how to acquire the global synchronization, will be investigated in the following section.

3. Implementation Aspects

The testbed implementation is decomposed into two parts. First, the point-to-point link and the PHY/MAC layer frame structure are described. Then, the distributed synchronization scheme of multiple nodes is addressed, which is different from the synchronization methods for other testbeds mentioned above. In the following, we only address the time synchronization of the source node and relay node to destination node. Indeed, this scheme can be extended to the configuration with multiple source/relay nodes in a straightforward way.

For point-to-point link, all the parameters are designed by the user. In this paper, we choose the differential binary phase shift keying (DBPSK) constellation. Although the joint transmission of the source and relay promises large theoretical gains in the form of coded cooperation, the distributed synchronization of the nodes is a large implementation hurdle. For the links between nodes, time division scheme is adopted and the source/relay nodes are synchronized to the destination node with a time synchronization scheme first designed for wireless sensor networks in [26, 27], which will be implemented with USRP hardware in our contribution.

3.1. Point-to-Point Link Implementation. First, the basic link implementation for point-to-point transmission is presented. In this paper, the DBPSK constellations are used and the detailed procedure of the DBPSK transmission is presented in Figure 4.

As shown in Figure 4(a), at the transmit node, the packet from the upper layer is processed by the DBPSK modulator into the modulated baseband complex signals, which are then processed by USRP board before sending to the antenna. Specifically, at the transmitter, the binary data after channel coding are first differentially encoded and then mapped using BPSK constellations. Here, the bit “0” is mapped to “+1” and the bit “1” to “−1.” Finally, the modulated symbols are fed into the shaped filter and then are sent to the USRP equipment.

In the receiving side, the RF signal is transformed to digital baseband signal in USRP hardware and then processed by DBPSK demodulator implemented with software until they are fed into the decoder as the input. At the receiver, the receive procedure is described as shown in Figure 4(b).

For point-to-point link, all the parameters are designed by the user. In this paper, we choose the differential binary phase shift keying (DBPSK) constellation. Although the joint transmission of the source and relay promises large theoretical gains in the form of coded cooperation, the distributed synchronization of the nodes is a large implementation hurdle. For the links between nodes, time division scheme is adopted and the source/relay nodes are synchronized to the destination node with a time synchronization scheme first designed for wireless sensor networks in [26, 27], which will be implemented with USRP hardware in our contribution.

In summary, the system parameters are listed in Table 1, which are similar as the configuration in [13]. However, the data rate is much higher.

Then, in order to realize the point-to-point data transmission, the physical layer protocol data unit is formed as shown in Figure 5.

As shown in Figure 4(a), at the transmit node, the packet from the upper layer is processed by the DBPSK modulator into the modulated baseband complex signals, which are then processed by USRP board before sending to the antenna. Specifically, at the transmitter, the binary data after channel coding are first differentially encoded and then mapped using BPSK constellations. Here, the bit “0” is mapped to “+1” and the bit “1” to “−1.” Finally, the modulated symbols are fed into the shaped filter and then are sent to the USRP equipment.

In the receiving side, the RF signal is transformed to digital baseband signal in USRP hardware and then processed by DBPSK demodulator implemented with software until they are fed into the decoder as the input. At the receiver, the receive procedure is described as shown in Figure 4(b). The received sequences from the USRP are first compensated using the Automatic Gain Control (AGC) module and filtered using the matched filter. Then, a carrier and timing recovery procedure is designed. After the differential demodulation, the frame synchronization is acquired according to the synchronization sequence. Finally, differential modulation can be detected according to the difference between the two successive symbols.

In summary, the system parameters are listed in Table 1, which are similar as the configuration in [13]. However, the data rate is much higher.

In the receiving side, the RF signal is transformed to digital baseband signal in USRP hardware and then processed by DBPSK demodulator implemented with software until they are fed into the decoder as the input. At the receiver, the receive procedure is described as shown in Figure 4(b). The received sequences from the USRP are first compensated using the Automatic Gain Control (AGC) module and filtered using the matched filter. Then, a carrier and timing recovery procedure is designed. After the differential demodulation, the frame synchronization is acquired according to the synchronization sequence. Finally, differential modulation can be detected according to the difference between the two successive symbols.

In summary, the system parameters are listed in Table 1, which are similar as the configuration in [13]. However, the data rate is much higher.

In the receiving side, the RF signal is transformed to digital baseband signal in USRP hardware and then processed by DBPSK demodulator implemented with software until they are fed into the decoder as the input. At the receiver, the receive procedure is described as shown in Figure 4(b). The received sequences from the USRP are first compensated using the Automatic Gain Control (AGC) module and filtered using the matched filter. Then, a carrier and timing recovery procedure is designed. After the differential demodulation, the frame synchronization is acquired according to the synchronization sequence. Finally, differential modulation can be detected according to the difference between the two successive symbols.

In summary, the system parameters are listed in Table 1, which are similar as the configuration in [13]. However, the data rate is much higher.

In the receiving side, the RF signal is transformed to digital baseband signal in USRP hardware and then processed by DBPSK demodulator implemented with software until they are fed into the decoder as the input. At the receiver, the receive procedure is described as shown in Figure 4(b). The received sequences from the USRP are first compensated using the Automatic Gain Control (AGC) module and filtered using the matched filter. Then, a carrier and timing recovery procedure is designed. After the differential demodulation, the frame synchronization is acquired according to the synchronization sequence. Finally, differential modulation can be detected according to the difference between the two successive symbols.

In summary, the system parameters are listed in Table 1, which are similar as the configuration in [13]. However, the data rate is much higher.

In the receiving side, the RF signal is transformed to digital baseband signal in USRP hardware and then processed by DBPSK demodulator implemented with software until they are fed into the decoder as the input. At the receiver, the receive procedure is described as shown in Figure 4(b). The received sequences from the USRP are first compensated using the Automatic Gain Control (AGC) module and filtered using the matched filter. Then, a carrier and timing recovery procedure is designed. After the differential demodulation, the frame synchronization is acquired according to the synchronization sequence. Finally, differential modulation can be detected according to the difference between the two successive symbols.
3.2. Time Synchronization Implementation for the Distributed Nodes. In this paper, we implement a completely distributed time synchronization scheme for the testbed. First, the superiority of the distributed schemes falls into supporting the emulations when the nodes locate in a relatively wide range. The distributed schemes avoid the bondage of the wire connection usually with cables between the slave node and the central node. The wire connection may limit the range of the node locations when they are verified using the testbed in the real wireless environment. In contrast, the distributed schemes can support the nodes which are located in a quite wide area. Then, among all the distributed node synchronization protocols, we choose the simplified FTSP [26, 27] to implement the synchronization of all the nodes based on the USRP hardware. In the cooperative communication verification, the node number is limited and we can easily choose one to serve as the central node. Further more, the simplified FTSP is quite suitable for this case due to its low communication payload and relatively high timing precision. In addition, this synchronization scheme can adapt to the different number of slave nodes, which is quite suitable for our evaluation testbed. Therefore, we choose the simplified FTSP rather than the RBS, TPSN, and so forth.

The fundamental process of the simplified FTSP is summarized into the three steps as described in the following three parts.

3.2.1. Broadcasting the Time Synchronization Message. In our testbed, the destination node serves as the master node and broadcasts a time synchronization message periodically. Assume that the master is sending the i-th time synchronization message. The master node broadcasts the synchronization message frame and the synchronization information field structure is shown in Figure 6. The time synchronization message includes its sequence number i, the global timestamp \( GT(i - 1) \), and the Cyclic Redundancy Check (CRC) field [26]. The sequence number is used to detect the loss of the time synchronization messages. Considering the important influence of time synchronization message to the cooperative communication systems, the sequence number and the global timestamps are both encoded using the repetition codes with the code rate of 1/8 to guarantee the transmission reliability. This is different from the design in [26] due to the relatively large data rate adopted in this paper. The time synchronization message frame will be transferred to the PHY layer after adding the control field.

Specifically, in the testbed, a timer using the 10 KHz clock as the driver is designed and 32 bits are used to represent the current global time. The timer is implemented using software in the PC, and thus the current time can be easily included in the synchronization frame. However, this also brings the low accuracy, for the global time is based on the timer implemented in the software. In this way, the timestamp is generated according to the software timer. Considering the various processing delay of software, the timing accuracy may be degraded, and in this paper we only guarantee the timing accuracy in millisecond level. We use a little large guard interval to guarantee the verification of different cooperation schemes sacrificing the utilization rate of the time slot.

3.2.2. Time Synchronization Table. All the slave nodes need to maintain a time synchronization table in the implemented synchronization scheme as described in [26]. In this paper, the slave nodes include the source node and the relay node. The table includes n entries and each entry in the table includes a local timestamp \( LT(i) \), a global timestamp, and a valid flag. It should be stressed that the time synchronization table needs to be filled with valid (local time, global time) entries before the offset and skew parameters can be accurately calculated. Specifically, in this paper, in order to reduce the processing time jitter of the software and hardware, the synchronization table has the length of 16 and the linear regression uses the 16 groups of data to keep the time synchronization accurate. In this platform, the table is implemented using the first-in, first-out queue of length 16 to store the global timestamp and the local timestamp.

Further more, the overall time synchronization process is divided into two processes, the fast synchronization process and the synchronization maintaining process. In the fast synchronization process, the master node sends 16 consecutive time synchronization message frames and thus rapidly fills the time synchronization table, improving the synchronization convergence speed. This process is only used when the platform starts to work. In the synchronization maintaining process, the master node broadcasts the time synchronization message once in a super cycle. The slave node receives the message and adjusts the local time to keep the node synchronization with the master node.

3.2.3. Time Synchronization Method. Finally, the program calculates the offset and the skew as shown in (1) to (3) using...
a classical linear regression [26]. Then, based on the two parameters, the slave node can compute a global timestamp \( GT_x \) using the local timestamp \( LT_x \) with (4) [26]. Therefore, a node judges whether its time slot begins or not according to the calculated time with (4).

\[
\text{offset} = \frac{1}{n} \sum_{i} \text{Diff}(i) = \frac{1}{n} \sum_{i} [GT(i) - LT(i)],
\]

\[
\text{LT} = \frac{1}{n} \sum_{i} LT(i),
\]

\[
\text{skew} = \frac{\sum_{i} \left[ LT(i) - \text{LT} \right] \left[ \text{Diff}(i) - \text{offset} \right]}{\sum_{i} \left( LT(i) - \text{LT} \right)^2},
\]

\[
GT_x = LT_x + \text{offset} + \text{skew} \cdot (LT_x - \text{LT}).
\]

Further more, based on the implemented synchronization method, we design the slot allocation method shown in Figure 7. The whole time is divided into three slots, the synchronization slot, the slot 1 for the source transmission, and the slot 2 for the relay transmission. The three slots form a super cycle with the period of total 50 ms. The synchronization slot occupies 10 ms and the source slot occupies 10 ms. The relay slot occupies 30 ms, and during this period, the destination node simultaneously receives and combines the data. With our designed data rate and the slot period, the active transmit period only occupies a small part of the slot period in order to avoid the collisions caused by the low timing accuracy due to the time-variant software processing delay. In the future, we will use the hardware timestamp to improve the timing accuracy and thus can improve the utilization rate of the time slot.

4. Simulation and Experiment Results

In this section, the testbed is first verified using the two coded cooperation schemes with Turbo codes in the real indoor environment. Then, experiments with three nodes and four nodes are also compared to show the scalability and flexibility of the testbed for different cooperation schemes including various number of nodes.

4.1. Experiments with Three Nodes. In this experiment, the parameters of the Turbo code for the two coded cooperation schemes are described as follows. The whole code length without puncturing is 1500 bits and the code rate is 1/3. In the rate compatible case, the code rate after puncturing is 1/2 and the relay forwards the other 500 parity-check bits punctured by the source node. In the distributed Turbo code, the source and the relay generate 500 parity-check bits, respectively. The source transmits the information bits and the 500 parity-check bits, while the relay forwards the other 500 parity-check bits. The generating polynomial for the RSC code is

\[
G(D) = \left( 1, \frac{1 + D + D^3}{1 + D^2 + D^3} \right).
\]

The interleaver can be represented as the permutation with

\[
i = \Pi(j) = (P_0 j + i_0) \mod P,
\]

where \( P = 500 \), the interleaver depth \( P_0 = 31 \), the initial offset \( i_0 = 17 \), and \( i \) and \( j \) are the addresses after and before the interleaving, \( i, j \in [0, P - 1] \).

First, the bit error rate (BER) and frame error rate (FER) of the two coded cooperation schemes over flat rayleigh fading are given by computer simulations. The simulation is performed under the assumption that the relay always successfully decodes the source data and all transmission blocks suffer flat fading with independent identical distributions. The simulation results are presented in Figure 8. From this figure, it can be observed that the two cooperative schemes both can achieve performance gains compared with the direct transmission where the source independently transmits its data encoded by Turbo code with the code rate of 1/3. Further more, for the two coded cooperation schemes, the performance of the distributed Turbo code is better than the rate compatible case.
Then, with our testbed, the rate compatible Turbo codes and the distributed Turbo codes are investigated and compared in a real indoor environment. The schemes are evaluated in a real scenario as shown in Figure 9, which is an office for students located in Tianjin University. The room for real experiments is filled with computers, computer tables, metal bookcase, whiteboard, and so forth, and in Figure 9, we only sketch the main objects. We perform all the experiments in the night when there are no people moving in the room. The distance between the source node and the destination node \(d_{SD}\) is about 5.0 meters and that between the relay and the destination \(d_{RD}\) is also about 5.0 meters. The distance between the source and the relay \(d_{SR}\) is about 1.5 meters. In order to depict the significant performance of the distributed codes, the relay is located near the source. In both cases, if the relay fails to decode the data, it will keep silent.

The experiment results are shown in Figure 10, in which the FER at the destination and the FER at the relay are given, respectively. In the experiments, we do not calibrate the transmit power, and the transmit power in the figure is only used for relative comparison. First, it can be observed that if the transmission power of the source node or the relay node is less than \(-21\) dBm, the rate compatible Turbo code exhibits better performance than that of the distributed Turbo code. When the transmission power is greater than \(-21\) dBm, the FER curve of the distributed Turbo code falls more steeply than the rate compatible counterpart. It shows that the scheme using the distributed Turbo code achieves higher diversity gain.

Then, Figure 10 gives further observation on the FER at the relay node. The relay node achieves lower FER performance in the rate compatible Turbo code than the distributed Turbo code. It is because that Turbo code with the code rate of \(1/2\) is decoded at the relay node in the rate compatible Turbo code scheme, while RSC code is decoded for the distributed Turbo code scheme. It can also be observed that the slopes of the curves for relay decoding are steeper than that of the direct transmission from the source to the destination and even parallel to the cooperation schemes. It may be because the channel statistics of the source-relay channel and source-destination channel are different. For the source-relay channel, the distance is quite limited and there is a strong light-of-sight transmission.

Moreover, the relay and source node cooperation probability \(P_c\) is presented in Figure 11. The cooperative probability is related to the frame error rate at the relay; that is, \(P_c = 1 - \text{FER}\). Under the relatively high transmission power, \(P_c\) at the relay node of the two schemes is both high. For example, when the transmission power of the source node is \(-21\) dBm, \(P_c\) is as high as 0.968 when using the distributed Turbo coding scheme, while \(P_c\) is almost 1 when the rate compatible codes are used.

In summary, the two cooperative schemes have their own advantages and disadvantages. If the transmission quality of the source-relay link is poorer, the rate compatible Turbo code shows more proper applications, and we can further adjust the puncturing rate according to actual relay channel conditions. If the source-relay link transmission quality is better, the distributed Turbo code scheme exhibits higher diversity gain. The experiment results are in accordance with the simulation results with perfect source-relay channel in Figure 8. Moreover, the relay only needs to decode the RSC code and has lower complexity when the distributed Turbo codes are used.

### 4.2. Experiments with Four Nodes

In this experiment, we compare two schemes using rate compatible Turbo codes with different number of relays. This experiment is only used to illustrate the supporting capability for different number of nodes. The same mother Turbo code is used as described above. In the first rate compatible case, three nodes are involved in the cooperation, which is identical to the scheme above. In the second rate compatible case with four nodes, the code rate after puncturing is \(2/3\) and two relays forward the other 750 parity-check bits punctured by the source node.
5. Conclusion

A hardware testbed using USRPs is developed for coded cooperation schemes based on Turbo codes in the PHY layer of cooperative communications. First, the fundamental point-to-point link based on DBPSK constellations is implemented and the PHY/MAC layer frames are designed. Then, a distributed node synchronization scheme is provided to synchronize all the nodes including the source node and relay node without any centralized controlling, which is different from the previous work in the literature. With this testbed, two popular coded cooperation schemes with Turbo codes are investigated in the real indoor environment and the experiment results prove the feasibility of the testbed.

However, in this paper, we do not implement the explicit MAC layer protocols, and thus other metrics for end-to-end Quality-of-Service (QoS), such as the throughput and the delay, could not be reasonably compared. In this paper, we only design the functions for code cooperation emulation in the MAC layer and only compare the performance of the PHY layer, that is, the FER curve. In the future, we will devise the full and explicit functions for the MAC layer and compare the throughput/delay of different schemes.

Acknowledgments

The authors would like to thank the editor and two anonymous reviewers for their constructive comments, which have greatly helped to improve this paper. This work was supported in part by National Natural Science Foundation of China (61002010), Program for Maritime Economies Promotion with Science and Technology in Tianjin (KJXH2011-02), and Seed Foundation of Tianjin University.
References

[1] 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.

[2] 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.

[3] 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.

[4] A. Nosratinia, T. E. Hunter, and A. Hedayat, "Cooperative communication in wireless networks," IEEE Communications Magazine, vol. 42, no. 10, pp. 74–80, 2004.

[5] T. E. Hunter and A. Nosratinia, "Diversity through coded cooperation," IEEE Transactions on Wireless Communications, vol. 5, no. 2, pp. 283–289, 2006.

[6] Y. Li, "Distributed coding for cooperative wireless networks: an overview and recent advances," IEEE Communications Magazine, vol. 47, no. 8, pp. 71–77, 2009.

[7] B. Zhao and M. C. Valenti, "Distributed turbo coded diversity for relay channel," Electronics Letters, vol. 39, no. 10, pp. 786–787, 2003.

[8] M. C. Valenti and B. Zhao, "Distributed turbo codes: towards the capacity of the relay channel," in Proceedings of the 58th Vehicular Technology Conference (VTC ’03), pp. 322–326, Orlando, Fla, USA, October 2003.

[9] A. Chakrabarti, A. de Baynast, A. Sabharwal, and B. Aazhang, "Low density parity check codes for the relay channel," IEEE Journal on Selected Areas in Communications, vol. 25, no. 2, pp. 281–291, 2007.

[10] C. Han and W. Wu, "Distributed generalized low-density codes for wireless relay networks," IEEE Communications Letters, vol. 12, no. 10, pp. 724–726, 2008.

[11] C. Han and J. Huang, "Coded cooperation with permutation and block encoding for wireless networks," in Proceedings of the 8th International Conference on Wireless Communications Networking and Mobile Computing, pp. 1–4, Shanghai, China, 2012.

[12] C. Han and W. Wu, "Distributed generalized low-density codes for multiple relay cooperative communications," EURASIP Journal on Wireless Communications and Networking, vol. 2008, Article ID 852397, 9 pages, 2008.

[13] G. Bradford, A framework for implementation and evaluation of cooperative diversity in software-defined radio [Ph.D. thesis], University of Notre Dame, 2008.

[14] G. J. Bradford and J. N. Laneman, "A survey of implementation efforts and experimental design for cooperative communications," in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP ’10), pp. 5602–5605, March 2010.

[15] G. J. Bradford and J. N. Laneman, "An experimental framework for the evaluation of cooperative diversity," in Proceedings of the 43rd Annual Conference on Information Sciences and Systems (CISS ’09), pp. 641–645, Baltimore, Md, USA, March 2009.

[16] A. Bletsas and A. Lippman, "Implementing cooperative diversity antenna arrays with commodity hardware," IEEE Communications Magazine, vol. 44, no. 12, pp. 33–40, 2006.

[17] J. Alonso-Zarate, J. Sanchez Recacha, N. Zorba et al., "Cooperative communications: from theory to experimental implementation," in Proceedings of the Global Communications Conference (GLOBECOM ’12), pp. 1722–1727, Anaheim, Calif, USA, March 2012.

[18] T. Korakis, M. Knox, E. Erkip, and S. S. Panwar, "Cooperative network implementation using open-source platforms," IEEE Communications Magazine, vol. 47, no. 2, pp. 134–141, 2009.

[19] X. Liu, G. Qiu, and F. Yu, "Experimental study of cooperative communication utilizing GNU Radio and USRP2," in Proceedings of the 2nd International Conference on Consumer Electronics, Communications and Networks, pp. 2577–2580, Yichang, China, April 2012.

[20] L. Lu, T. Wang, S. C. Liew et al., "Implementation of physical-layer network coding," Physical Communication, vol. 6, pp. 74–87, 2013.

[21] Z. Per, M. Christos, S. Aris et al., "Experimental investigation of cooperative schemes on a real-time DSP-based testbed," EURASIP Journal on Wireless Communications and Networking, vol. 2009, Article ID 368752, 2009.

[22] C. S. Yu, T. N. Shen, K. G. Shin et al., "Multihop transmission opportunity in wireless multihop networks," in Proceedings of the IEEE Conference on Computer Communications (INFOCOM), pp. 1–9, San Diego, Calif, USA, March 2010.

[23] X. Zhang and K. G. Shin, "DAC: distributed asynchronous cooperation for wireless relay networks," in Proceedings of the IEEE Conference on Computer Communications (INFOCOM ’09), pp. 1–9, San Diego, Calif, USA, March 2010.

[24] J. Zhang, J. Jia, Q. Zhang, and E. M. K. Lo, "Implementation and evaluation of cooperative communication schemes in software-defined radio testbed," in Proceedings of the IEEE Conference on Computer Communications (INFOCOM ’10), pp. 1–9, San Diego, Calif, USA, March 2010.

[25] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, “The flooding time synchronization protocol,” in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys ’04), pp. 39–49, Baltimore, Md, USA, November 2004.

[26] N. Tackett, E. Jovanov, and A. Milenkovic, “An implementation of time synchronization in low-power wireless sensor networks,” in Proceedings of the 43rd IEEE Southeastern Symposium on System Theory (SSST ’11), pp. 61–66, Auburn, Ala, USA, March 2011.

[27] D. Cox, E. Jovanov, and A. Milenkovic, “Time synchronization for ZigBee networks,” in Proceedings of the 37th Southeastern Symposium on System Theory (SSST ’05), pp. 133–138, Tuskegee, Ala, USA, March 2005.

[28] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," ACM SIGOPS Operating Systems Review, vol. 36, no. 5, pp. 147–163, 2002.

[29] S. Ganeriwal, R. Kumar, and M. B. Srivastava, “Timing-sync protocol for sensor networks,” in Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys ’03), pp. 138–149, Los Angeles, Calif, USA, November 2003.