RODMAC : A RObust and Distributed MAC Protocol for Efficient Use of White Spaces

Eshan Nanda*, Udit Joshi†, Vinay Ribeiro† and Huzur Saran†
*EMC Software India Pvt. Ltd; Email: eshan.nanda@emc.com
†Department of Computer Science & Engineering, Indian Institute of Technology Delhi, India
Email: {joshudit@gmail.com, vinay@cse.iitd.ernet.in, saran@cse.iitd.ernet.in}

***Abstract*—We describe the design, implementation and evaluation of a RObust and Distributed MAC (RODMAC) protocol for cognitive radio networks which exploit spectrum white spaces for communication. RODMAC incorporates two novel features. These are a robust design resilient to node failure and a scheme for on-demand channel hopping. Other features include a fully distributed architecture and incorporation of Forward Error Correction (FEC) for better performance. RODMAC allows any pair of nodes requiring dedicated bandwidth for communication between them to request exclusive use of a free channel. Besides improving the QoS for communication between this pair of nodes, this mechanism also improves the overall efficiency of bandwidth resource usage. RODMAC can serve as the basis for building MAC protocols for applications which include device-to-device (D2D) communication, tele-medicine or military communication in a jamming environment. Our design has been implemented on the Wireless Open Access Research Platform (WARP) of Rice University. For this purpose, we have established a test bed of 4 WARP nodes. This design has been validated by careful analysis and repeated experiments over a 6 month period. We have analyzed our results for jitter, throughput and loss for different bandwidths. Our results for these parameters show an improvement in loss percentage of about 70% for a 5 Mbps link with video streaming on different channels using our novel on-demand channel hopping scheme as compared to simultaneous video streaming on the same channel.**

I. Introduction

Over a period of time, spectrum has become increasingly scarce due to the advent of a large number of new generation wireless technologies. With this, the spectrum cost has also increased to such an extent that spectrum has become an extremely precious and scarce natural resource. Solutions which make more efficient use of spectrum resources are greatly needed.

Cognitive radio (CR) technology offers a promising solution as it seeks to exploit the unused spectrum white spaces within licensed spectrum bands, thereby obviating high licensing costs. Recent spectrum regulations in several countries around the globe allow unlicensed use of licensed spectrum, at a particular point in time, provided the licensed (or primary) user is not transmitting in that spectrum. These regulations are enablers for bandwidth hungry applications, since often a large fraction of licensed bands remain unused [1].

MAC protocols for such CR networks must overcome several challenges. First, they must be sensitive to the presence or absence of primary users in particular bands because the CR network must vacate any band soon after a primary user begins transmission in it. A large research effort has focussed on developing efficient spectrum sensing methods for detecting the presence of primary users. However, government regulations have simplified matters by mandating databases which list which channels are free for secondary usage. Research work in the domain of unlicensed use of TV white spaces received a significant boost since November 2008 when the United States FCC issued a Report & Order on this subject [2].

A second challenge is that CR networks must decide which channel or set of channels to use among the potentially large pool of spectrum available for opportunistic use. The use of only a single channel by all nodes in the network simplifies communication in the network but at the same time does not exploit the potentially large number of white spaces available. Allowing nodes in the CR network to transmit on various channels can in theory improve throughput of the network, but is besought with the problem that nodes on different channels cannot communicate with each other, especially if each node is equipped with only a single transceiver. This creates the need for a common channel on which control information can be transmitted throughout the network. The MAC protocol must ensure that nodes periodically return to the control channel to sync with the rest of the network.

We develop our protocol keeping the following assumptions and requirements in mind.

- Both primary users (license holders) and secondary users (unlicensed CR users) need to coexist. Spectrum licensed to primary users is accessible to secondary users provided they do not interfere with primary users. The onus rests on the secondary users to ensure that the primary users are not adversely affected.
- The spectrum is divided into non-overlapping orthogonal channels of equal bandwidth. One channel is the fundamental unit of spectrum usage.
- We assume each user transmits from a single-interface half-duplex transceiver, using a single channel at any given time. So each node in our network consists of a single half duplex radio.
- Each radio transceiver is frequency agile, that is, it can quickly switch across channels with low delay. Such a high performance, hardware platform is critical for our network.

978-1-4673-5494-3/13/$31.00 © 2013 IEEE

- Secondary users are aware of the presence of primary users in various channels either by looking up a publicly available database or by performing spectrum sensing.
- Power outages are factored into our setting. Nodes can frequently come in and drop out of the network and the network needs to be inherently robust to handle such situations.
- The network size would be small, around only 10-20 nodes.
- Quality of Service(QoS) guarantees are needed by bandwidth hungry or latency sensitive applications.
- All nodes are within hearing range of each other.

Two key features of our protocol are:

1) Any two users are allowed to on-demand jump to a new channel, transfer data to one another, and then hop back to the original common channel after finishing their data transfer.
2) There is no fixed leader or point of failure.

We believe that such an on demand channel hopping scheme can support several powerful applications some of which are described below.

One application is Device-to-Device (D2D) communication. Here several devices which are close to each other intend to communicate with each other, and exploit each others computational information or peripheral resources. In this scenario, even if one device fails or leaves the network, then the rest of the network should not be affected by it. Initially, assume that all nodes in the D2D network are on a single channel and running RODMAC. In case the channel is reoccupied by a primary user, the whole network will jump to another channel. The new channel and the exact time to jump is controlled by the current "leader" of the network. The channel hopping helps in dynamic reuse of spectrum without having any effect on the primary user. Thus, our protocol can make use of white spaces available.

An important feature provided by the protocol is an on-demand jump to a new channel. Two nodes which want to transfer data between each other can request the leader for a dedicated channel. On being granted an unused channel in some white space, they both jump to this new channel, exchange data, and then jump back to the common channel. The reason for nodes to request a dedicated channel are manifold. For example, the two nodes may have some urgent and critical data to transfer, or may be running an application requiring high QoS in terms of bandwidth and/or latency. By using a dedicated channel, the pair of nodes avoid interference from other nodes of the network. We note that while it is possible to guarantee good QoS over a common channel through the use of TDMA, such solutions typically require strict time synchronization, careful scheduling, and have a single point of failure. Another reason why a pair of nodes may request a new channel is that the nodes have a large file to transfer from one to the other and would rather not clog the common control channel with this file transfer, where therest of the nodes are present. In case a node is running an application which simultaneously transfers data between multiple other nodes, then it can refuse the request from another node to jump to a dedicated channel. The node instead chooses to communicate on the common channel.

RODMAC can also serve as the basis for building MAC protocols for applications in medical and military communications. Patient telemetry has been performed over dedicated wireless channels since the early 1970s. The life-critical nature of the application is a key constraint. In such the scenario the presence of a guaranteed dedicated wireless channel for a desired duration would be the ideal solution keeping in view the life-critical nature of the application. The nodes can perform a channel hop to a free channel and communicate without getting disturbed by the rest of the network. Certain military applications which need to be robust to a single point of failure and yet give users the ability to establish and maintain dedicated control channels, can also benefit from RODMAC.

In the next section, we discuss the design features of RODMAC. Section III describes the platform and test bed used for our experiments. In Section IV, we describe the implementation of RODMAC and present results from our four node testbed in Section V. Section VI describes the related work and finally we conclude along with the future work in Section VII.

II. Protocol Design

When we look at utilizing the available white spaces, it is possible that a large number of channels are immediately available. A channel vector can thus be maintained to keep track of the available non-overlaping channels. This vector can be sorted based on the quality of the channel using the interference level as a metric. Assuming the availability of such a channel vector, it would be ideal if different nodes could reserve bandwidths suitable for their specific applications. We believe that such an on demand channel hopping scheme can support several powerful applications.

In our protocol, at any time, a node is either in *client mode* or *server mode*. The node in server mode, or *server*, is responsible for performing spectrum sensing and thus sending beacons for channel switching. The next channel to switch to is calculated using this channel vector. The client waits for a channel switching packet (CH_HOP packet) and switches to the corresponding channel specified in the beacon. Within a particular channel, nodes use a CSMA protocol to decide which node transfers the data and which others wait.

In this section, we describe the fully distributed architecture, robust design, on-demand channel hoping and spectrum sensing of RODMAC.

A. Fully distributed architecture

There is no fixed centralized entity in our design. On booting each node assumes the role of a client. Each node then waits for some amount of time. The amount of time is determined by the ID of the node. Thus ID of the node also determines the priority among the nodes to become the server. We have implemented a simple scheme using a function of the node ID to determine the server.

The node with the lowest ID gets the highest priority. A timer is set in clients which is a function of the node ID. For our experiments this timer value was set to twice the ID. That is, for the node with ID = 3, its timer expires after 6 seconds. So if within 6 seconds it does not receive any CH_HOP packet it will switch to the server mode. Thus, as soon as the timer expires, the node takes the role of server and starts transmitting CH_HOP packets. These packets have a center frequency and a timer, which indicates to other nodes to switch to the specified channel indicated by the center frequency. Details of the booting procedure are presented in Section IV A. .

B. Robust Design

Due to the RODMAC's distributed protocol design, it does not matter if there is a power outage and a node goes out. Even if the node which fails is a server, over a period of time another node will assume the role of a server when it fails to receive CH_HOP packets from the previous server. In general the node with the lowest ID will act as the server and other nodes would assume the role of clients on receipt of these CH_HOP packets. The server will periodically transmit these channel control packets.

C. Channel Selection

Out of the free non-overlapping channels available, one of the channels is selected to be the next hop channel. Every node builds and maintains a channel vector which is the list of channels available at any point of time. This channel vector is shared by all client nodes with the server, thus the server has an overall picture of the entire network. Out of the group of common available channels, one of the channels is selected on the basis of the least recent usage, i.e., the next channel is that channel which is not used for the longest amount of time. The server maintains a timestamp for every channel which is used for next channel determination. A legal channel vector is generated during spectrum sensing. If a channel is free, then the channel is assigned as 1 else it becomes illegal and is assigned a 0. We have used energy sensing for determining channel availability. More sophisticated techniques involving feature sensing should give better results.

D. On-demand channel hopping

In our protocol we have implemented an on-demand channel hopping scheme. This is a key feature of our design. In this, a node can request for a dedicated channel to communicate with another node. One of the nodes informs the other node that it wishes to initiate a communication with it on a dedicated channel. The other node responds affirmatively provided it has a spare channel available. The channel vectors of both nodes are already available and the node makes a decision based on the status of the channel availability vector. The node responds with the channel ID of the available channel. This channel is now removed from the pool of available channels in the network, by broadcasting this information. Both the nodes now switch to the new channel. Any application can now be run on this dedicated channel. In our current implementation, a CSMA protocol is used for communicating on the dedicated channel.

When the particular application is complete, any of the nodes can now initiate a request to release the channel and also the nodes can then become a part of the overall wireless network, to which both the nodes are affiliated. While releasing the channel, it is ensured that the released channel is added to the pool of available channels, by broadcasting this information.

The key challenge is to communicate this information to all the nodes which could be on potentially different channels. Our protocol ensures that this convergence time, wherein all nodes in the network become aware that a channel which was in use has now become free, is kept as low as possible. This was less than 2 seconds for our experiments.

E. Spectrum Sensing

While we have mainly focussed on the MAC layer we have also incorporated a rudimentary spectrum sensing scheme based on energy sensing. Energy based spectrum sensing is the most simple scheme for spectrum sensing. Such a scheme has minimal computational or implementation complexity. Therefore such schemes are also extremely popular and have been implemented in several instances [3], [4], [5], [6], [7], [8]. We utilise the CSMA carier sense period to carry out this spectrum sensing function. However, in order to be fully effective, we would have to incorporate feature based sensing as in [9], [10], [11].

In our present scheme we detect the presence of an incumbent based on a specified threshold, just above the ambient noise level, which we have determined experimentally for the Wi-Fi band where we are working at the moment. Potentially our protocol is designed for the TV white spaces and a simple change in the frequency used is all that is needed to switch to any band of our choice, keeping the application in mind. As our WARP hardware does not support transmission in the sub-GHz TV white spaces, we are at present confined to the Wi-Fi band for our experiments.

F. Forward Error Correction (FEC)

We have implemented our application on the latest OFDM reference design version 16.1 [12]. This version has inbuilt support for FEC. We have been able to get better results after incorporating the latest OFDM reference design, compared to earlier ones.

III. Prototype

We have performed our experiments on a test bed of 4 WARP boards. We describe the WARP platform and other details of our implementation in this section.

A. Platform

To build our prototype, we needed a platform which could operate at the speed of a hardware solution, provide very fast frequency hopping yet also retain the exibility of a software solution. We eventually chose Rice Universitys WARP [12], a scalable and extensively programmable wireless SDR platform. Our prototype makes use of the OFDM reference design v16.1 available in the WARP repository.
The WARP motherboard is equipped with a Xilinx Virtex-4 FPGA (older versions use the Virtex-2) and can be interfaced to 4 daughter boards. Each RF daughterboard operates in the 2.4GHz and 5GHz ISM bands. For our experiments, we used the 2.4GHZ frequency band.
The WARP reference design provides the ability to write a MAC application using the libraries provided by it.

B. Test bed

The test bed for our experiments consists of 4 nodes in a topology illustrated in Figure 1. Each node consisted of a WARP board with a client PC attached to it via Ethernet. Every node communicates with every other node. Of all the nodes in range, only one of them will act as a server and others will be clients who will hop according to the information sent by the server. This decision for server selection is done on the basis of node ID. A node with the lowest ID will be selected as a server.

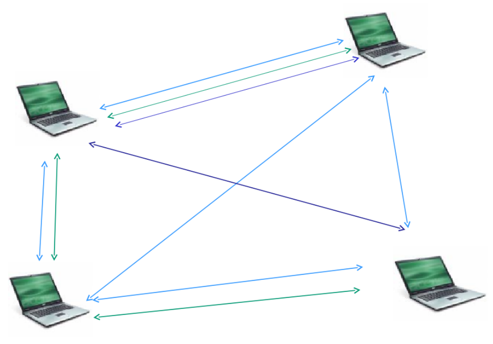
Fig. 1. Network Topology

IV. Implementation Details

The following subsections describe the specific outline for complete protocol

A. Node Booting

Initially, when a node boots up, it scans all the available channels, thus ensuring generation of the channel vector. This scan involves energy sensing as well as packet reception. Thus the list of available and busy channels is available with all nodes. A node, post this scanning,knows whether a server already exists or if it should take up the role of a server. If a server already exists, the node will join that particular channel and become a part of the network.

Any node after joining the network, waits for a specific amount of time before assuming the role of the server. This will ensure that the node does not miss out on the presence of the server. If a server already exists, the node will remain in client mode. We have carried out several experiments to decide on a suitable metric for this purpose. This wait time has been set as two times the node's ID in seconds. In general the wait time can be a suitable function of the node ID. Since node id's can be manually set on WARP boards, we can safely assume that the node ID's are small. Among the various configurations, this setting was found to work well, which was confirmed through a number of experiments with different parameters.

This setting also ensures a deterministic priority among the nodes. A node with the lowest ID, will first become the server, provided all nodes boot up at approximately the same time. Other nodes, on receiving CH_HOP packets from a node of lower ID will continue to act as clients. This CH_HOP packet contains the following information

- Next Channel to hop
- Time Left for hopping
- Server ID (ie. senders own node ID)
- On-demand session information ie. which nodes are currently on which on-demand channel

This CH_HOP packet is sent 10 times within the hopping interval. The hopping interval has been set as 5 seconds and can be modified through serial input. Ultimately, we will be hopping only when the control channel gets occupied by a primary user. We are hopping after every 5 seconds only in order to test our protocol. Effectively we send a CH_HOP packet after every 500ms. This ensures adequate redundancy. Even if some CH_HOP packets are lost in transit, the nodes are unlikely to lose synchronization. The booting process does not take more than two seconds and all the nodes are synchronized quickly. Even in case of an outage, when the node acting as the server goes out, parity is regained within 5 seconds.

Channel vector information can also be generated if we have some sort of TV white space database. If that is the situation, we know about the available channels and set the starting default channel as the channel with the lowest center frequency. But since this database is not available, we have used a scanning method to determine the channel vector.

B. Maintaining Priority

Once the network gets set up, the node with the lowest ID assumes the role of the server. This node keeps on sending CH_HOP packets to ensure synchronization amongst all the nodes in the network. The other nodes which are receiving these CH_HOP packets, keep acting as clients. If a node is acting as a server, receives a CH_HOP packet from a node with higher priority, i.e., lower ID, this node switches to client mode. In this way, the priority among the nodes is maintained.

C. Maintaining Robustness

If a node which is currently acting as a client does not receive the CH_HOP packet for a specified time emperically set as two times its ID, then it switches to server mode and starts broadcasting CH_HOP packets. This ensures that if a node with ID as 1 goes down, then node with ID 2 assumes the role of the server and so on. Thus the rest of the network is again synchronized within the switching time from client to server of the node with lowest ID. This mechanism ensures adequate robustness and resilience to node failures and outages. No matter which node fails, the rest of the network quickly assumes a new stable state.

If a node is currently a server, and still receives CH_HOP packets, it checks for the source ID of those packets. If the source ID is less than its own ID, it switches to client mode else it continues as a server. This particular check ensures that priority is maintained amongst all the nodes. Figure 2 represents the flow chart of how robustness is ensured.

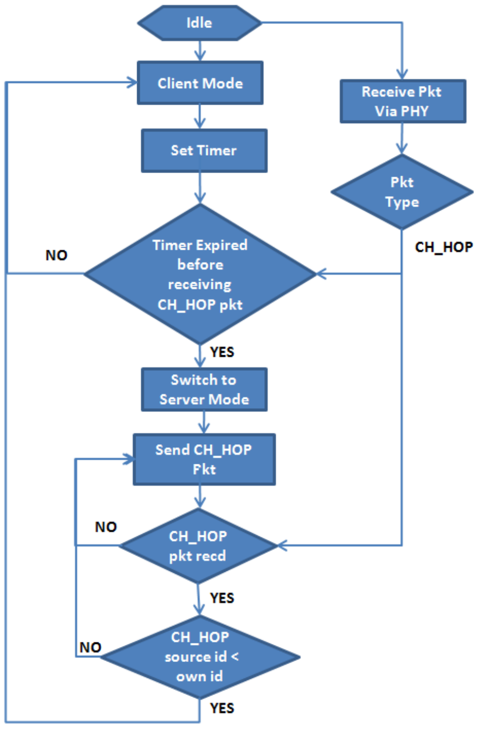

Fig. 2. Maintaining Robustness

D. Requesting On-Demand Bandwidth

Fig 3 explains the steps involved in requesting on-demand bandwidth. A node, if required can request for a dedicated channel for communication. The requesting node sends a COG_REQ packet to the destination node. The COG_REQ packet contains the ID of the requesting node and the channel proposed for subsequent communication.

The destination node, sends a COG_RES packet on accepting the request to jump to the new specified channel. This packet is broadcast so that other nodes mark that particular channel as illegal. This ensures subsequent undisturbed transmission among the nodes on that particular channel.

The destination node can also deny the request by sending a COG_DEN packet. On receiving such a packet, the initiator node knows that its request has been denied.

Another reason for broadcasting the COG_RES packet is to ensure that every nodes will get the information about on-demand sessions currently going on. By broadcasting the COG_RES, all other nodes know that this particular channel is currently busy and they cant request for that particular channel at this time. Thus the nodes know about the busy channels, and when they request for their own dedicated bandwidth, they do not select channels which have been marked as busy. We also ensure that if the channel on which nodes are communicating gets occupied by a primary user, the nodes get re-synchronized and then again request for a new free channel instead of directly jumping to a new channel.

E. Ending On-Demand Bandwidth Allocation

When any of the nodes involved in the On Demand Bandwidth session, want to terminate the session, they broadcast a COG_END packet on the selected channel. On receiving such a packet, the node will again start channel hopping in the usual manner as explained earlier. Then these nodes re-synchronize with the rest of the network, i.e., as soon as these nodes receive a CH_HOP packet from a node which was not a part of session, they broadcast a COG_BACK packet. On receiving a COG_BACK packet, all other nodes in the rest of the network, make the particular channel which was selected for on demand bandwidth, as legal. The COG_BACK packet is sent twice to ensure redundancy. Moreover, the information about which nodes are on which channel is broadcast by the server along with the CH_HOP packet . This ensures that, every node has the updated information about the lists of busy and free channels. Thus every node will now know that the nodes which were involved in the session are back, and the spectrum which was being used is again available for re-use. Figure 4 explains the termination of on-demand bandwidth allocation.

F. Re-Synchronization

After ending the session, the participating nodes need to re-synchronize with the rest of the network. These nodes do not know the channel on which the network is on. So they scan all the channels, starting from the channel from which they had jumped, and re-synchronize. We believe that the criticality is only during the on demand stage and it is acceptable even if the nodes take some time to come back and re-synchronize as the on demand period would be over by then. As soon as the nodes gets re-synchronized, they broadcast a COG_BACK packet. This packet informs the rest of the network to mark the previous On-Demand channel as free. This packet is sent as soon as the nodes receive a CH_HOP packet from a node which is a part of the rest of the network.

V. Performance Evaluation

We perform experiments to determine the throughput, jitter and loss percentage for different scenarios using the test bed shown in Figure 1. We have used QPSK as our modulation scheme. We have used VLC Player to stream a video in order to test our cognitive MAC protocol. One server has been

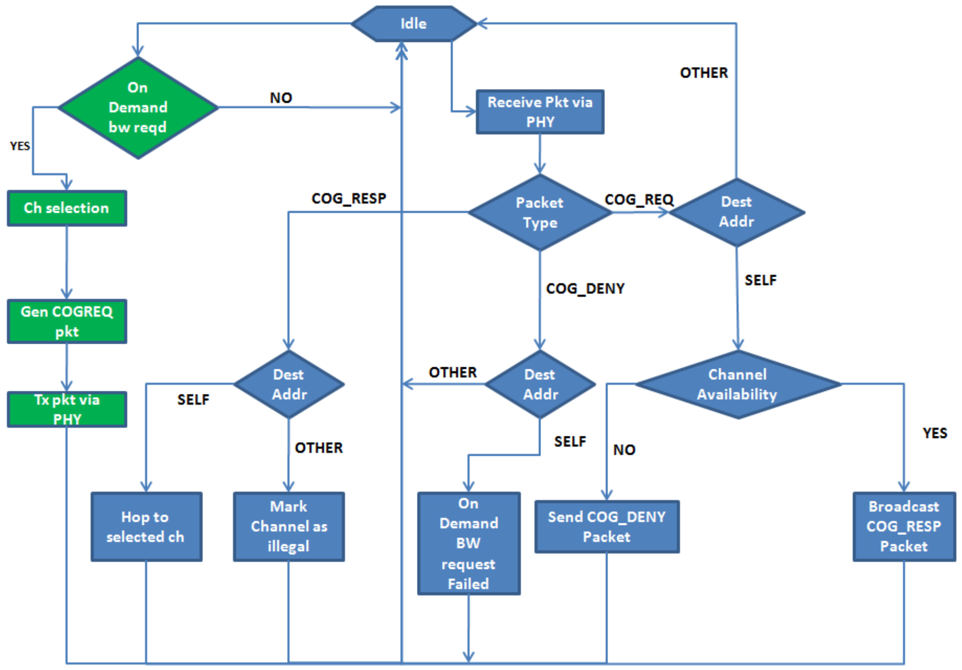

Fig. 3. Requesting On-Demand BW

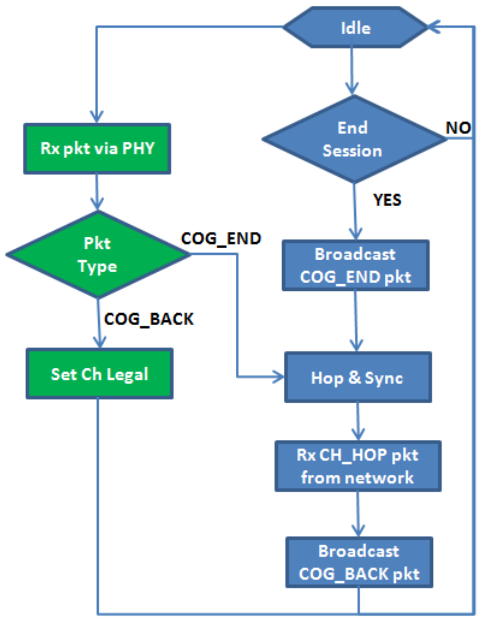

Fig. 4. Termination of On-Demand Bandwidth Session

used to stream the video to two clients. We have observed that the video stream continues uninterrupted across frequency changes. Resetting of any node causes the video link to break. The link gets revived in some time when the nodes get synchronized again.

The performance in terms of throughput, loss and jitter were measured by performing UDP Iperf test. The experiments performed have been discussed below

A. *Iperf Test between two nodes*

In the following test, Iperf measurements have been carried out for different offered loads in order to observe the net throughput and loss figures, with changes in the offered load. In these experiments, only two nodes were involved in transmission while other two nodes were silent. Iperf tests were performed on the nodes involved in transmission. The result is summarized in table 1.

TABLE I
UDP TEST RESULTS

Offered Load (Mbps)	Throughput (Mbps)	Jitter (ms)	Loss(%) (%)
1.0	0.98	0.01	0
2.0	1.97	0	0.35
3.0	2.89	0.1	0.68
4.0	3.9	4.84	1.5
5.0	4.9	5.34	1.2

The throughput in this case was found to be almost equal to the offered load as other nodes in the network were silent.

B. *Iperf Test with Streaming Video*

In the following test, Iperf measurements has been carried out with a video being streamed on VLC, all nodes being on the same channel at any instance of time. Thus a large decrease in throughput is observed in Table II as expected. The decrease is caused by the presence of video being streamed which takes a large portion of the available bandwidth.

C. *Iperf Test with Streaming Video On Different Channels*

In the following test, Iperf measurements has been carried out with a video being streamed on VLC, after invoking on

TABLE II
IPERF TEST WITH STREAMING VIDEO

Offered Load (Mbps)	Throughput (Mbps)	Jitter (ms)	Loss(%) (%)
1.0	1	0.07	0
2.0	1.90	0.09	0.39
3.0	2.71	3.4	1.4
4.0	3.7	3.9	3
5.0	4.74	5.2	5.1

demand bandwidth between a pair of nodes. Each pair of nodes is on a different channel at the same instance of time, leading to an improvement in the net throughput. Iperf test results are shown in table III.

TABLE III
IPERF TEST WITH STREAMING VIDEO ON DIFFERENT CHANNELS

Offered Load (Mbps)	Throughput (Mbps)	Jitter (ms)	Loss(%) (%)
1.0	0.99	0.07	0
2.0	1.98	0.09	0.32
3.0	2.87	1.4	0.58
4.0	3.89	3.6	1.2
5.0	4.87	4.5	1.4

It was observed that there is a marked improvement in throughput as compared to the scenario when video is being streamed on the same channel. The throughput value was found to be comparable to the throughput value of the scenario when all other nodes were on silent mode (Table I) .

D. Offered Load vs. Loss % For Different Scenarios

Fig 5 shows the graph that has been plotted between the offered load and the loss for our three different settings as described in the previous points. As expected, the best performance is when on demand bandwidth is invoked, as a dedicated channel is now available between a pair of nodes. This is the best scenario as now there is no overhead of CH_HOP packets, and a dedicated link is available between a pair of nodes or for a group of nodes, as may be the case.

E. Varying Throughput With On Demand Bandwidth (Test at 3Mbps)

In this experiment, we tried to determine the effect on throughput our protocol will have. We start with an Iperf test between two nodes. Meanwhile, a VLC video transmission is started by some other node on the same channel. After the start of video, the two nodes on which Iperf test was performed were moved to a new on-demand channel.

Fig 6 shows the effect on throughput when an on-demand channel request is invoked. As expected the throughput spikes up prominently when the nodes have negotiated a dedicated channel and have made the transition. The transition period is for the negotiation phase when the link is being established. So the throughput falls heavily during this short phase. When the on demand bandwidth session is terminated, the nodes revert back to the initial state after a brief period.

F. Varying Loss (%) With On Demand Bandwidth (Test at 3Mbps)

Figure 7 shows the effect on loss percentage when a request for on-demand bandwidth is invoked. As expected the loss percentage reduces drastically when the nodes have negotiated a dedicated channel and have made the transition. The transition period is for the negotiation phase when the link is being established. So the loss percentage spikes up heavily during this short phase. When the on-demand bandwidth session is terminated, the nodes revert back to the initial state after a brief period.

G. Robustness

The Server sends the CH_HOP packets to the clients so that the clients can initiate the hopping process. If a server goes down, the clients can come out of synchronized hopping. Within the threshold time , the client next in line for promotion to server will switch to server mode and start sending the CH_HOP packets. This results in establishing a synchronized state ensuring that rest of the network is not affected.

VI. RELATED WORK

We have developed a fully distributed and robust MAC protocol for Cognitive Radio. Furthermore, in keeping with current trends and the advances in digital communication, we have implemented a single transceiver protocol wherein every node has a single high performance frequency agile half-duplex transceiver.
Already the IEEE 802.22 [http://www.ieee802.org/22/] working group has standardized a MAC layer based on CR for reuse of spectrum that is allocated to TV broadcast service. The architecture of the 802.22 MAC layer is centralized and relies on a base station, while our protocol is designed for fully decentralized operation.
Several projects have also been implemented to exploit the vacant white spaces in the TV band. The WhiteFi project [13] is one such promising project. Like RODMAC, the need for a dedicated control channel has been obviated here. WhiteFi implements a technique called SIFT (Signal Interpretation before Fourier Transform) which enables nodes to rapidly discover base stations operating at different centre frequencies while also having different channel widths. These signals are analysed in the time domain. A new metric, called MCham has also been proposed which allows base stations to choose the best slice of the spectrum to operate on. This spectrum slice can obviously span multiple channels. In our network, however, all nodes are equal and are within hearing distance of each other so the need for having dedicated base stations is obviated. Also each node operates exclusively on a single channel at any point of time.

A protocol called Cooperative MAC or CoopMAC [14] has been implemented on the WARP platform. Here bridge nodes are used to connect nodes which do not have direct connectivity on a common channel, or are operating under poor and dynamically changing channel conditions.

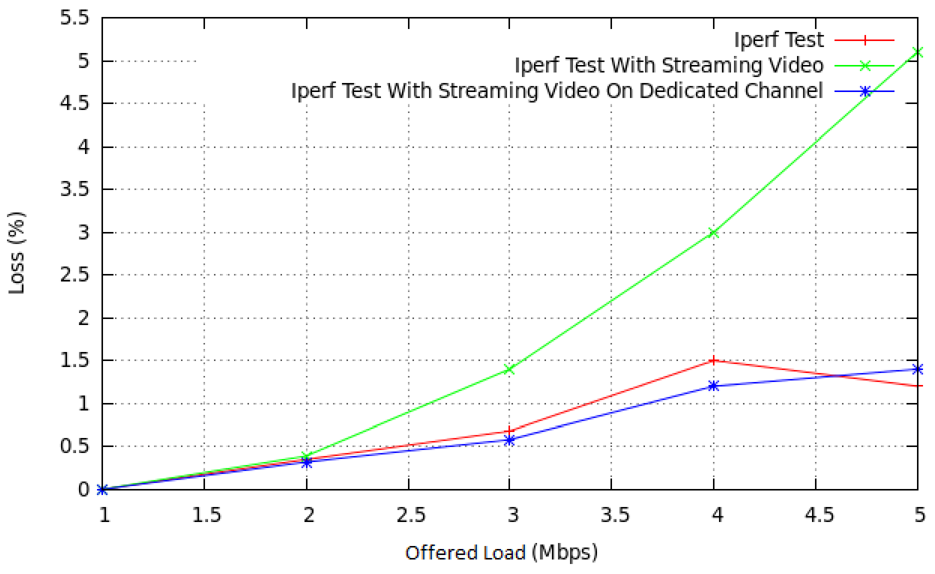

Fig. 5. Offered Load vs. Loss % For Different Scenarios

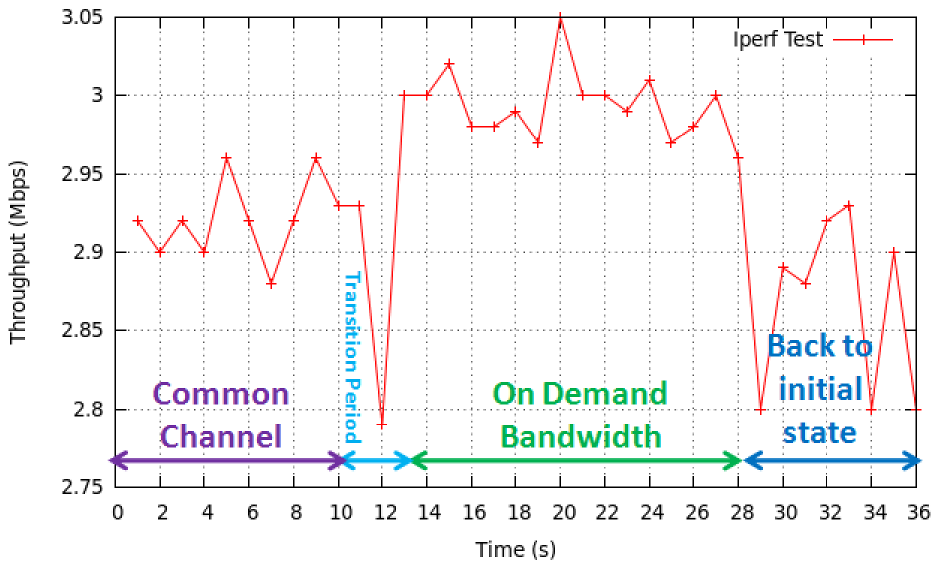

Fig. 6. Varying Throughput With On Demand Bandwidth (Test at 3Mbps)

In [15], the authors present a fully distributed MAC protocol implemented on the WARP platform. This protocol focusses on maintaining a pool of available channels and assigns weights to these channels based on a channel selection algorithm. Over a period of time this data is collated and the best available channel is provided for use. At any point of time all the nodes in a network are on the best available channel which has the minimum interference. Our protocol is an improvement over this protocol as we also have a facility for on demand channel hopping which can be used in our applications. So different groups of nodes can be on different channels simultaneously thereby improving the network throughput.

In [16], a node's MAC address determines the channel with which the node will be associated. This particular channel is termed as the home channel and is used by the node to wait for incoming packets. A source node wanting to communicate with another node has to switch to the receiver nodes home channel before transmission, and immediately return to its home channel after completion. The home channel is a single channel which is provided to the node for listening to incoming transmissions. This approach considerably increases the switching and the synchronization overhead vis-a-vis RODMAC where nodes are operating either on a single common channel or have jumped to dedicated channels.

Channel Hopping Multiple Access (CHMA) [17] and the

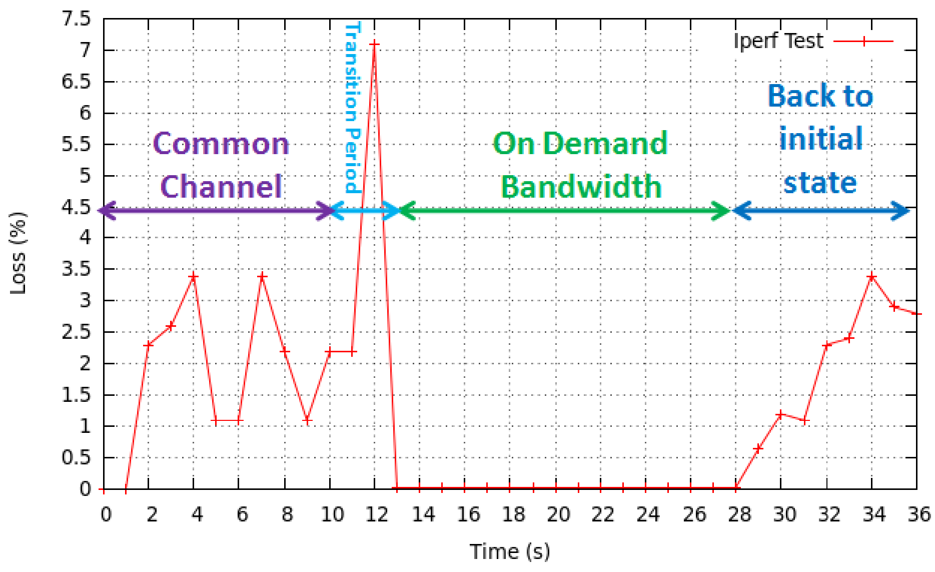

Fig. 7. Varying Loss (%) With On Demand Bandwidth (Test at 3Mbps)

Slotted Seeded Channel Hopping (SSCH) algorithm [18] use a channel hopping approach. If a node wants to communicate with another node, it follows the other nodes schedule. Only if two nodes are able to successfully exchange control information, do they go ahead and initiate data transfer.

The Hop Reservation Multiple Access (HRMA) protocol [19] is a multi-channel MAC scheme for slow frequency hopping spread spectrum (FHSS) wireless adhoc networks. Here all nodes hop according to a pre-defined hopping pattern. Whenever a node has a data packet to send, it exchanges RTS/CTS control packets with the receiver. Both nodes remain in the same hop for the entire data transmission duration. Other nodes not involved in the communication are not affected and follow the pre-defined hopping sequence. This approach is quite different from RODMAC where the server dynamically informs all nodes to switch to a channel with every CH_HOP packet. Our protocol potentially has better spectral efficiency then HRMA or any pre-defined hopping pattern based protocol. Moreover, a fixed approach will fail in the presence of incumbents as the behaviour of an incumbent is inherently random, while the channel hopping always follows a fixed pattern in HRMA.

VII. Conclusion & Future Work

We have developed and implemented a CR MAC protocol called RODMAC which can be used in several application contexts. RODMAC has several features, such as robust design, on-demand channel hopping, fully distributed architecture which make it apt for applications having specific bandwidth requirements. We have established a basic experimental test bed of 4 WARP nodes for our platform for implementing RODMAC. The feature of on-demand bandwidth enhances the throughput of the network, while also reducing the loss percentage by avoiding collisions.

Our prototype implementation on WARP boards validates the technique RODMAC which we proposed in this paper. The code for our implementation will in due course be added to the WARP software repository to benefit the research community.

We have only implemented a rudimentary energy based spectrum sensing scheme. This scheme needs improvement. Incorporating a spectrum sensing scheme based on feature sensing should give better results. The technique proposed in this paper requires the communicating nodes to be in range of each other. As future work, we will want to extend the technique to multi-hop scenario. Moreover, while currently we are using CSMA for intra-channel communication, we can explore the usage of a TDMA based contention scheme for QoS guarantees.

The on-demand channel hoping helps in utilizing the white space efficiently without causing interference with the already established data transfer. In future, we will like to explore the capability of having more than 2 nodes communicating on the on-demand channel.

Acknowledgment

This work was supported by the projects titled “Design and development of a rapidly deployable Wimax based mesh network” (RP02177) and “SPARC: Spectrum Aware Rural Connectivity” (RP02565) funded by the Department of Information Technology, Government of India.

References

[1] A. Iyer, K. K. Chintalapudi, V. Navda, R. Ramjee, V. Padmanabhan, and C. Murthy, *SpecNet: Spectrum Sensing Sans Frontires*, in 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI), Mar 2011.
[2] *Second Report and Order and Memorandum Opinion and Order In the Matter of Unlicensed Operation in the TV Broadcast Bands, Additional Spectrum for Unlicensed Devices Below 900 MHz and in the 3 Ghz Band*, Federal Communication Commision, Document 08-260, Nov. 14, 2008.

[3] F. Digham, M. Alouini, and M. Simon, *In the energy detection of unknown signals over fading channels*, in Proc. IEEE Int. Conf. Commun., vol. 5, Seattle, Washington, USA, May 2003.

[4] S. Shankar, C. Cordeiro, and K. Challapali, *Spectrum agile radios: utilization and sensing architectures*, in Proc. IEEE Int. Symposium on New Frontiers in Dynamic Spectrum Access Networks, Baltimore, Maryland, USA, Nov. 2005.

[5] S. Jones and N. Wang, *An experiment for sensing-based opportunistic spectrum access in CSMA/CA networks*, in Proc. IEEE Int. Symposium Symposium on New Frontiers in Dynamic Spectrum Access Networks, Baltimore, Maryland, USA, Nov. 2005.

[6] J. Lehtomaki, J. Vartiainen, M. Juntti, and H. Saarnisaari, *Spectrum sensing with forward methods*, in Proc. IEEE Military Commun. Conf., Washington, D.C., USA, Oct. 2006.

[7] T. Yucek and H. Arslan, *Spectrum characterization for opportunistic cognitive radio systems*, in Proc. IEEE Military Commun. Conf., Washington, D.C., USA, Oct. 2006.

[8] A. Ghasemi and E. Sousa, *Optimization of spectrum sensing for opportunistic spectrum access in cognitive radio networks*, in Proc. IEEE Consumer Commun. and Networking Conf., Las Vegas, Nevada, USA, Jan. 2007.

[9] Z. Tian and G. B. Giannakis, *A wavelet approach to wideband spectrum sensing for cognitive radios*, in Proc. IEEE Int. Conf. Cognitive Radio Oriented Wireless Networks and Commun. (Crowncom), Mykonos Island, Greece, June 2006.

[10] K. Maeda, A. Benjebbour, T. Asai, T. Furuno, and T. Ohya, *Recognition among OFDM-based systems utilizing cyclostationarity-inducing transmission*, in Proc. IEEE Int. Symposium on New Frontiers in Dynamic Spectrum Access Networks, Dublin, Ireland, Apr. 2007.

[11] P. D. Sutton, K. E. Nolan, and L. E. Doyle, *Cyclostationary signatures for rendezvous in OFDM-based dynamic spectrum access networks*, in Proc. IEEE Int. Symposium on New Frontiers in Dynamic Spectrum Access Networks, Dublin, Ireland, Apr. 2007.

[12] ”Rice University WARP Project”, *http://warp.rice.edu*.

[13] P. Bahl, R. Chandra, T. Moscibroda, R. Murty,M. Welsh *White Space Networking with Wi-Fi Connectivity*, in SIGCOMM 2009, Barcelona, Spain.

[14] J. Sharma, V. Gelara, S. Singh, T. Korakis, P. Liu, S. Panwar, *Implementation of a cooperative MAC protocol using a software defined radio platform, in Local and Metropolitan Area Networks*, in LANMAN 2008.

[15] Junaid Ansari, Xi Zhang and Petri Mhnen, *A Decentralized MAC for Opportunistic Spectrum Access in Cognitive Wireless Networks* , in CoRoNet'10, September 20, 2010, Chicago, Illinois, USA.

[16] S. Chaudhuri, R. Kumar, and A. Saha, *A MAC Protocol for Multi Frequency Physical Layer*, Technical Report, Rice University ,Texas, Jan. 2003.

[17] A. Tzamaloukas and J. Garcia-Luna-Aceves, *Channel-hopping multiple access*, in IEEE ICC ,June 2000.

[18] P. Bahl, R. Chandra, and J. Dunagan, *SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks*, in ACM MobiCom, Sept. 2004.

[19] Z. Tang and J. J. Garcia-Luna-Aceves, *Hop-Reservation Multiple Access (HRMA) for Ad Hoc Networks*, in IEEE Infocom, 1999.

[20] J. So and N. Vaidya, *Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using a Single Transceiver*, in ACM Mobihoc, May 2004.

[21] Carlos Cordeiro, Kiran Challapali, *C-MAC: A Cognitive MAC Protocol for Multi-Channel Wireless Networks*, in Proceedings of IEEE DySPAN, April 2007.

[22] A. Chawla, V. Yadav, V. D. Sharma, J. Bajaj, E. Nanda, V. Ribeiro, H. Saran *RODEO: Robust amd Rapidly Deployable TDM Mesh with QoS Differentiation*, in WISARD 2012.