Dynamic Spectrum Cognitive MAC (DySCO-MAC) for Wireless Mesh & Ad hoc Networks

Shashi Raj Singh, Buddhika De Silva, Tie Luo and Mehul Motani
Department of Electrical and Computer Engineering
National University of Singapore, Singapore
E-mail: singh_shashiraj@yahoo.com, gbbuddhika.ds@gmail.com, elelt@nus.edu.sg, motani@nus.edu.sg

***Abstract*—Mesh networks have emerged as a strong alternative for providing last mile broadband network access. To further improve their performance, in light of increasing consumer bandwidth demands, much effort is being put into better using the available bandwidth resources. Cognitive dynamic spectrum access is one such technique that allows opportunistic use of underused spectrum, thereby increasing spectrum utilization. In this paper, we present the design and implementation of a new medium access control (MAC) protocol called *Dynamic Spectrum Cognitive* MAC (DySCO-MAC). It enables cognitive dynamic spectrum access for a single radio, asynchronous wireless mesh/ad hoc network. This protocol is based on a unique approach of node cooperation that takes advantage of the broadcast nature of the wireless channel. We have implemented DySCO-MAC using an open source Linux Wi-Fi device driver and evaluated its performance on a real-world testbed. To the best of our knowledge, this is the first work that implements dynamic spectrum use at the MAC level using COTS Wi-Fi cards. The experimental measurements confirm that cognitive dynamic spectrum access is a promising realistic technology for boosting the performance of next generation wireless networks.**

***Index Terms*—Cognitive dynamic spectrum access, cooperation, DISH, MAC, IEEE 802.11, ath5k.**

I. Introduction

Wireless mesh networks have become a realistic technology in the broadband access market over the past years. Their application ranges from providing broadband Internet backhauls in cities to military and maritime use. They promise wider coverage than traditional wireless LANs and lower deployment and operation costs. This has led network operators and service providers to consider mesh networks as a serious alternative for wireless backhauls to provide last mile access. As consumer bandwidth demands keep increasing, mesh backhauls would carry a large amount of broadband traffic. Hence it is imperative to improve mesh network performance so as to keep ahead of the demand. To this end, cognitive dynamic spectrum access offers a promising solution to efficiently using the bandwidth for mesh networks. Cognitive dynamic spectrum access arises from the observation that a major part of the licensed and unlicensed spectrum is mostly under-utilized or inefficiently used. For instance, the FCC reported that 70% of the allocated spectrum in the US is not utilized. Cognitive dynamic spectrum access allows the *opportunistic* use of free spectrum without interfering with existing or incumbent users, thereby increasing the overall bandwidth usage. Due to such benefits, the enabling technologies of cognitive dynamic spectrum access have been extensively researched and protocols have been proposed at the various OSI layers.

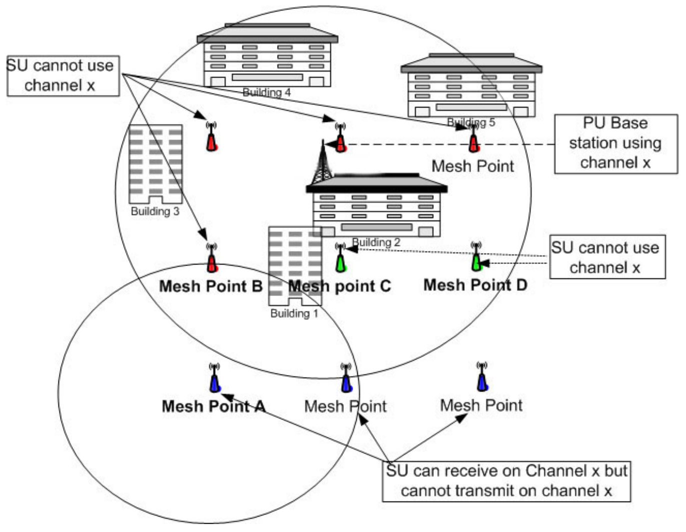
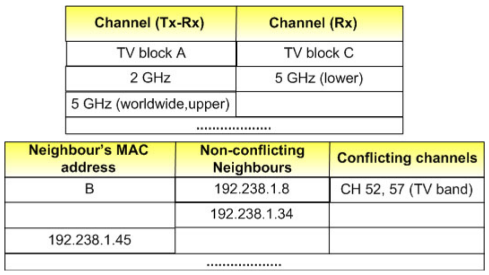


Fig. 1: A secondary user mesh network and a PU base station.

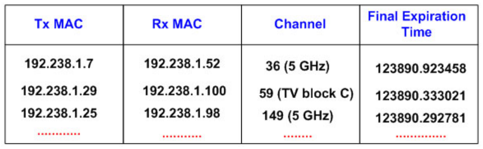
Our study focuses on the MAC layer and we present a new protocol called DySCO-MAC in this paper. This MAC protocol enables cognitive dynamic spectrum access in single-radio and asynchronous wireless mesh/ad hoc networks. DySCO-MAC is a cognitive extension of CAM-MAC [1] which is our previous effort in addressing multi-channel networking issues. In this paper, we modify CAM-MAC significantly and introduce a cognitive framework to allow spectrum awareness in addition to an improved multi-channel use. We have implemented a prototype of DySCO-MAC using commercial Wi-Fi cards and built a medium size mesh testbed to evaluate the performance gain from applying cognitive dynamic spectrum access to 802.11 mesh networks. In spite of the availability of a large number of orthogonal channels, 802.11 mesh networks use fixed channel assignment policy, which often leads to inefficient spectrum usage and degraded performance. On the other hand, a mesh network based on DySCO-MAC and 802.11 PHY is spectrum aware and capable of efficiently using the ISM/UNII and, if possible, other bands. We elaborate our implementation approach, compare its limitations and benefits, and share the experience gained, in order to demonstrate how the implementation of cognitive dynamic spectrum access techniques can possibly be achieved.



Channel (Tx-Rx)	Channel (Rx)
TV block A	TV block C
2 GHz	5 GHz (lower)
5 GHz (worldwide,upper)	
....	

Neighbour's MAC address	Non-conflicting Neighbours	Conflicting channels
B	192.238.1.8	CH 52, 57 (TV band)
	192.238.1.34	
192.238.1.45		
	....	

(a) Available spectrum resource & Neighbor table.



Tx MAC	Rx MAC	Channel	Final Expiration Time
192.238.1.7	192.238.1.52	36 (5 GHz)	123890.923458
192.238.1.29	192.238.1.100	59 (TV block C)	123890.333021
192.238.1.25	192.238.1.98	149 (5 GHz)	123890.292781
....	....	....	....

(b) Channel usage table.

Fig. 2: Spectrum management tables.

We discuss the design challenges in Section II and the DySCO-MAC design in Section III. In Section IV we discuss our implementation approach including the implementation platform, design, and lastly the benefits and limitations. Section V presents system evaluation including a testbed and experimental results. Section VI concludes the paper and discusses our future work.

II. Design Challenges

One of the major challenges in cognitive dynamic spectrum access is how to manage the available spectrum resource, which can be used by cognitive nodes in communication. This spectrum management challenge can be divided into two categories: (1) *available spectrum identification* and (2) *available spectrum access*. Available spectrum identification or finding out the free spectrum to use in communication is essential for cognitive nodes to avoid interfering with licensed users or primary users (PU). The information about spectrum availability can be gathered by a node based on its own sensing efforts which may involve physical spectrum sensing (PSS) or virtual spectrum sensing (VSS) [2]. PSS is scanning each channel, where the typical technique used is energy detection. VSS exploits the control information to predict the spectrum usage by overhearing the control information exchange. This entails the knowledge of control frame format used by the targeted primary network. Based on the sensing results from either PSS or VSS, a node can identify available portions of the spectrum for its communication. However, the spectrum availability information for a given node may not apply to other nodes in its radio neighborhood due to various reasons. The main reason is the hidden terminal problem that arises when a secondary user (SU) transmitter is out of the range of a PU but the intended SU receiver is within the range. For example see Fig. 1 which shows a mesh network as a SU network consisting of a number of mesh points (MP). MP A is out of range of the PU base station and MP B is within its range. So according to A's spectrum availability information, channel x can be used, but this information is

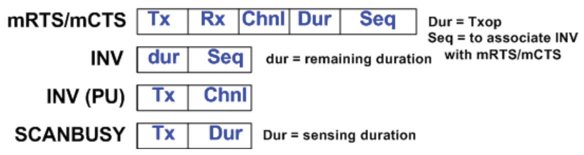


Fig. 3: Packet formats (common fields are not shown like CRC).

not true for B. The same problem may also arise when the nodes are shadowed from the presence of the PU. In Fig. 1 MPs C and D are shadowed from the PU's transmission by high rise buildings, so they will fail to detect the PU and can cause interfering transmissions. To avoid such scenarios, it is essential for a node to also have the spectrum availability information of one's neighbors. The other challenge is how to coordinate the available spectrum access among the cognitive nodes. This is a multichannel (different parts of available spectrum) coordination (MCC) problem [1] which has two variants: (1) channel conflict problem, where nodes select channels that are already in use, and (2) deaf terminal problem, where a node tries to communicate with a node that is on a different channel (performing data communication or spectrum sensing). Single-radio based architectures are especially prone to MCC problems. Since the single radio is used for both data communication and keeping track of channel usage information, a node will miss channel usage information once it switches to any data channel. To meet this MCC challenge, various approaches have been proposed in the past. One main approach is using *multiple radios* and dedicating one of them to monitor channel usage. The other main approach is utilizing *time synchronization* to regulate the irregular node behaviors using well-known time slots or channel hopping sequences. In prior work [1], we have tried a simple *single radio* and *asynchronous* solution which effectively addresses MCC problems and yields high performance. In this paper, we adopt the same idea but make significant changes to the original design for further improvement and particularly adapting to the cognitive context. In addition, we also aim to solve another problem called the exposed terminal problem. This problem arises due to insufficient understanding of the radio neighborhood. In such cases, a transmitter, say U, can actually use a channel occupied by another transmitter say V, if U's receiver is out of the range of the node V. However, node U is not aware of this and decides not to use the channel, leading to inefficiency.

III. DySCO-MAC Design

To solve the available spectrum identification challenge, nodes maintain an available spectrum resource table and a neighbor table as shown in Fig. 2(a). The available spectrum resource table has two columns, one includes all the channels that can be used for both transmission and reception and the other includes all the channels only for reception. An entry in a node A's neighbor table consists of a neighboring node B, nodes that B can not hear but A can hear (non-conflicting for B) and channels that B cannot use but A can use. Based on this table, A knows the channels that B can use in communication. The exposed terminal problem is also solved using the non-conflicting neighbor information. Nodes periodically broadcast the list of their neighbors which other nodes use to maintain their neighbor tables. To solve the MCC problem, we employ a unique approach of node cooperation called distributed information sharing (DISH), which allows neighbors to share spectrum usage information with nodes who need it. DISH can be used to solve the two problems: (1) channel conflict problem and (2) deaf terminal problem. To allow DISH, nodes maintain a spectrum usage table as shown in Fig. 2(b). The table stores the channels that are currently in use in the network. Each entry consists of a MAC addresses (Tx & Rx nodes), channel (center freq. + width) and corresponding expiration time. The protocol uses a control channel shared by all the nodes to allow sharing of spectrum usage information. We assume that the control channel can be used from any of the following: a) SUs have a licensed spectrum. b) an unlicensed band such as the ISM/UNII bands.

A. Available Spectrum Identification

This consists of three components: spectrum capability, regulatory policy and scan control. The spectrum capability block finds out the range of frequencies that a node can use based on its RF frontend. The regulatory policy block contains geographical spectrum regulatory information, which spectrum is assigned to which PU system, minimum interference threshold for a PU system, etc. Scan control is a master block that takes input from the other two blocks in making sensing decisions. Based on the inputs like the PU system present, it chooses parameters such as frequency and duration of sensing. We note that the Scan control block depends on the physical layer capabilities of the node to facilitate physical sensing techniques such as energy detection.

B. Available Spectrum Access

On the control channel, nodes negotiate for a data channel by following a handshake as shown in Fig. 4. The handshake is divided into a control session and a data session. In the control session, a sender and a receiver negotiate a data channel (center freq. + width) using an mRTS/mCTS exchange, which is followed by a cooperation phase. Then in the data session, both nodes switch to their chosen data channel and the sender senses the channel for duration Tcs, and if free, sends packets to the receiver based on a transmission opportunity (Txop) limit. The receiver replies with a combined ACK (cACK) at the end of Txop (here both width and Txop can vary as per SU network traffic). To start the control session, a transmitter first sends an mRTS on the control channel which carries the data channel selected by the sender. If the receiver also deems the data channel to be free, it will reply with an mCTS which duplicates the channel. On the other hand, if the receiver finds that the data channel is in use by other cognitive nodes or by a PU, it sends an INV or INV(PU) packet respectively. In the case of INV, the transmitter will try again with another channel and for INV(PU), the transmitter updates the third column of its neighbor table accordingly. Now suppose a deaf

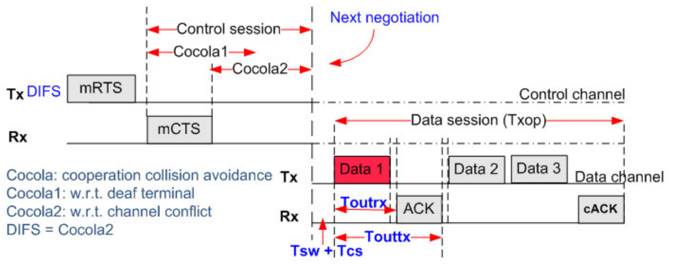


Fig. 4: Protocol design: control & data session.

terminal problem is created, in this case a common neighbor of the sender and the receiver may identify this problem and send an INV packet to inform the sender to back off. Such an INV packet is sent during a $cocola_1$ (cooperation collision avoidance) period, where a $cocola$ period is used to mitigate INV collision caused by multiple neighbors who identify the same MCC problem sending INV simultaneously. In this mechanism, a neighbor with INV to send will sense the control channel for a period of $U[0, cocola]$ where $U[\cdot]$ denotes the uniform distribution. If the channel remains idle, it sends the INV at the end of the random period, and otherwise, it cancels the INV. Therefore, any neighbor who sends INV will suppress its neighbors via CSMA. In the rest of cases where collision is not avoided (since not all the neighbors may hear each other), the alarm message that the handshake should *not* proceed still gets conveyed because INV represents a negative message what is lost is the duration information carried by INV which helps sender determine a backoff period. This, however, does not present a serious problem, because the sender will just have to *estimate* a backoff period with less accuracy. Now suppose a channel conflict problem is created, i.e., the sender and the receiver agree on a channel that is already in use by some other nodes. In this case, either sender's or receiver's neighbors may send an INV packet during $cocola_2$ period using the same CSMA mechanism as in $cocola_1$. If no INV is received, both nodes will switch to the data channel and transmitter will send the first data packet as a probe and wait for ACK. Probe is used to avoid the case where only receiver receives an INV from its neighbor. After an ACK timeout period of Touttx, if no ACK is received, transmitter will switch back and try again. A similar data packet time out (Toutrx) period is defined for the receiver. To avoid the deaf terminal problem due to asynchronous sensing we use a control packet called SCANBUSY as shown in Fig. 3. When-ever a node starts spectrum sensing, it broadcasts this packet on the control channel, so that its neighbors know that it is not available for data communication. If any neighboring node misses this broadcast due to some reason (e.g.,it was on data channel) and tries to start communication with this node, other neighbors can alert it by sending an INV packet. After sensing the spectrum if a node finds any changes in the spectrum availability it alerts the neighbors by sending INV(PU) stating the channels that are now in use by a PU. Its neighbors, on receiving INV(PU), will flag the channels mentioned in it as NEEDSCAN, and these channels will not

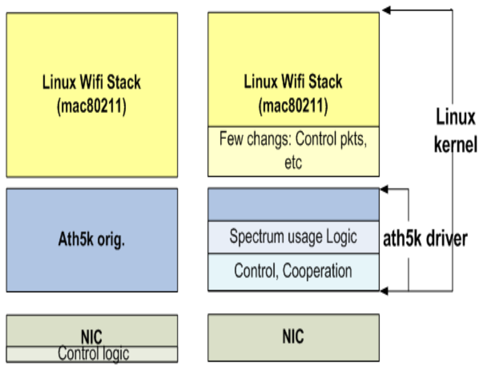


Fig. 5: DySCO-MAC Framework and Implementation design.

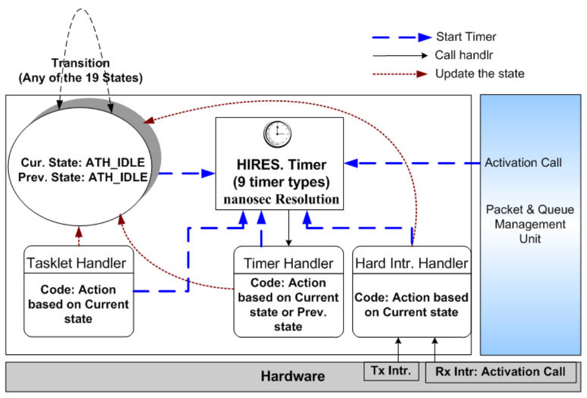

Transition (Any of the 19 States)
Start Timer
Call handlr
Update the state
Cur. State: ATH_IDLE
Prev. State: ATH_IDLE
HIRES. Timer (9 timer types) nanosec Resolution
Activation Call
Packet & Queue Management Unit
Tasklet Handler
Code: Action based on Current state
Timer Handler
Code: Action based on Current state or Prev. state
Hard Intr. Handler
Code: Action based on Current state
Hardware
Tx Intr.
Rx Intr: Activation Call

Fig. 6: DySCO-MAC control unit.

be used until after the next round of spectrum sensing. If neighbors find a NEEDSCAN channel busy, it will be removed from their available spectrum resource table. If they find it free, the channel will be moved to only for reception entry in the available spectrum resource table (Fig. 2(a)) and not be used for transmission as it might affect PU systems in the neighborhood.

IV. Implementation of DySCO-MAC

A. Implementation Platform

The implementation of DySCO-MAC is done on commercial Wi-Fi cards. We have used the Linux 802.11 stack (mac80211) and open source linux Wi-Fi driver (ath5k) [3]. DySCO-MAC requires disabling the Wi-Fi distributed coordination function (DCF) which is implemented on the cards. To achieve this, access to hardware registers was essential. Further, we decided to implement the protocol over a mesh capable driver. An initial candidate for the driver was Madwifi. It can be divided into two parts 1) *Software layer* and 2) *Hardware Abstraction Layer (HAL)*. Only the software layer is accessible to developers as the HAL is in binary format. All the functions to access the hardware is in the HAL thus severely limiting design options. In particular, hardware access (card registers) and configuration are not possible. Also, the Madwifi driver only supports net80211 stack that does not have mesh support. On the other hand, ath5k is fully accessible and supports mac80211 stack which has mesh support. Therefore we chose ath5k as our development driver.

B. Implementation Design

The basic design of the Linux Wi-Fi stack (mac80211) with ath5k driver is shown in Fig. 5. We modified the mac80211 stack, ath5k driver and the hardware register values. The implementation design of DySCO-MAC consists of two units 1) Spectrum Availability Controller and 2) Channel Access Controller.

1) Spectrum Availability Controller (SAC): The SAC carries out the available spectrum identification in supported bands (here 2.4 and 5 GHz bands). The spectrum capability block resides in the driver and produces a list of supported channels by carrying out a hardware enquiry, here we should mention that the hardware should support this functionality. We built a regulatory block by reusing a part of cfg80211 which is a kernel configuration utility used by mac80211. It has information about allowed 802.11 channels based on the geographical area, maximum allowed transmit power and presence of PU systems (in this case radar pulses as described by 802.11h standard [4]). The scan control block resides in the driver and scans all possible supported channels obtained from the feedback of the other two block's. It generates a list of channels based on user defined logic. Currently, the logic is to scan for the orthogonal channels in 2.4 and 5 GHz bands with a channel width of 20MHz. Nodes scan the spectrum periodically every 100*n sec, where n is calculated using the mechanism similar to the random backoff in 802.11, if idle to update the available spectrum information. We use virtual sensing and energy detection. Virtual sensing involves looking for different types of 802.11 frame, like management and control frames. Physical sensing involves observing the channel for a particular period of time to measure channel activity. We read the channel status register in the card to calculate the channel activity. Based on these observations we classify a channel as unused, low traffic, high traffic.

2) Channel Access Controller: This is the code introduced into the driver to implement DySCO-MAC control and cooperation logic. Following is the description of its 3 subunits. Firstly the "Packet Queue Management Unit" (pqm) carries out the queue and packet handling. We have implemented the transmit opportunity (Txop) limit in the form of a packet train. When the transmission time of the train is approximately equal to the Txop limit, we start a negotiation for the data exchange. Packet train based transmission complicates the working of network management protocols like ICMP. It can also complicate the mesh management process of mac80211 stack. To avoid this we use both per packet and train based transmission. Only UDP packets are allowed for packet train transmission. We introduced a new transmit control flag to mark UDP packets for this. All unmarked packets are transmitted using per packet transmission. To handle the two transmission types, we have introduced one new queue in software. On arrival of a packet, pqm unit prepares the packet for hardware transmission

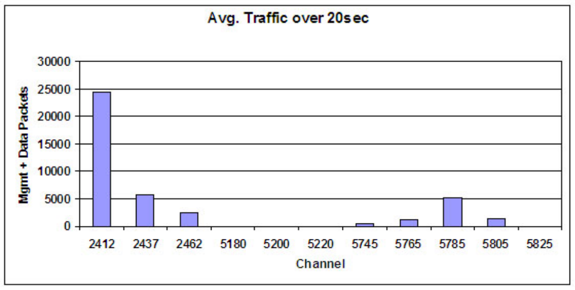


Fig. 7: Virtual sensing result.

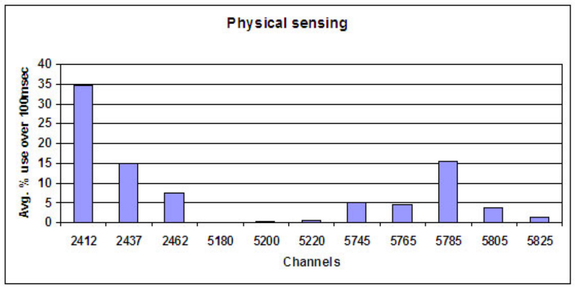


Fig. 8: Physical sensing result.

as is normally done in ath5k, then places the packet in the correct queue. If the control unit is not already started pqm unit will start it if the Txop limit is reached in the case of marked packets or immediately in the other case. The "Control Unit" handles all the DYSCO-MAC protocol logic. We have disabled the original control unit in the hardware and introduced our control logic into the driver. The control unit consists of 19 internal control logic states, high resolution timers, tasklets and interrupt handlers as shown in Fig. 6. In every handler we take actions based on the current or the previous states, update the states and may start a new timer or tasklet. The following paragraph describes some of the important changes to the original ath5k driver. We could not disable carrier sensing so reduced the DIFS period as much as possible by putting the slot time to zero and reducing SIFS time to the extent that it does not affect the performance. As required by the DySCO-MAC protocol we have introduced new control packets MRTS, MCTS and INV in the mac80211 subsystem. We use permanent buffers in the hardware for these control packets so we do not have to create them in the kernel every time we want them. We use a separate data queue in the hardware for these control packets. High resolution timers provide nanosecond resolution which is critical for our design. The timers are used to ensure control packets are sent at the correct time. They are also used to make sure that transmitter and receiver are roughly synchronized so they switch to the data channel and back to the control channel at the same time. Channel switching is employed using a new function that does fast channel change and takes 500 microseconds. Initially we had a switching delay of 20 milliseconds. The current control unit does not implement a few features of the DYSCO-MAC design. INV for deaf terminal, cACK on the data channel(data is sent in burst mode) and the collision avoidance (random transmission of INV between 0 and cocola) in the cocola periods are not currently included.

As per the protocol requirement, we introduced a "Cooperation unit" in the ath5k driver. The core of the unit is a Channel usage table as shown in Fig. 2(b). The unit also consists of neighbor table as shown in Fig. 2(a). Currently, the neighbor table is created manually. When a control packet is received by the control unit of a node, it is passed to the cooperation unit. The unit checks for the nodes relation to the Tx and Rx nodes in the packet i.e. whether the node is a common neighbor of the Tx-Rx pair or just a neighbor of one of the nodes in the pair. Then it checks for any channel conflict in the channel usage table. Based on the check results, cooperation unit sends feedback (whether to send INV) to the control unit. The channel usage table is updated whenever a node receives an mRTS, mCTS not to itself or receives an INV, depending on the newness of the information carried.

C. Limitations and Benefits

Usually, time critical functions that involve the 802.11 control logic (DCF) are in hardware and software drivers only prepare packets and hand it to the card. In our implementation, these time critical functions are handled by the driver (kernel space), limiting tight time synchronization. The presence of other kernel threads and user processes also add to delay. The benefit of such an implementation is that it can be benchmarked with commercial hardware (e.g., Wi-Fi cards), making the idea more promotable.

V. System Evaluation

We have set up a testbed consisting of custom PCs (CPC) for wireless protocol testing and evaluation. Each CPC is equipped with two wireless interfaces (only one was used in the current evaluation) and an Ethernet interface. The Ethernet interface is used for remote configuration, statistics collection and testbed monitoring. The CPC nodes in the testbed form a mesh network and run the DySCO-MAC protocol. To evaluate the protocol performance we conducted two types of experiments. One was to evaluate the cognitive feature and the other to evaluate the advantage of the dynamic spectrum access. Only the channels allowed in Singapore have been used for the experiments which do not include the spectrum used by radar.

A. Available spectrum identification

This set of experiments are to demonstrate how our spectrum access controller unit works. We employed VSS in Fig. 7 and PSS (energy detection) in Fig. 8. In VSS, we count number of management and data packets for 20s on a channel. In PSS, channel activity is checked every 10 μs over a 100ms period (an interval between two AP beacons). Both the sensings are done for 12 hours. The VSS result (Fig. 7) shows that channel 2412 MHz have comparatively high traffic and other channels either are unused or have very low traffic. The PSS result shown in Fig. 8 corroborates the VSS results, and those

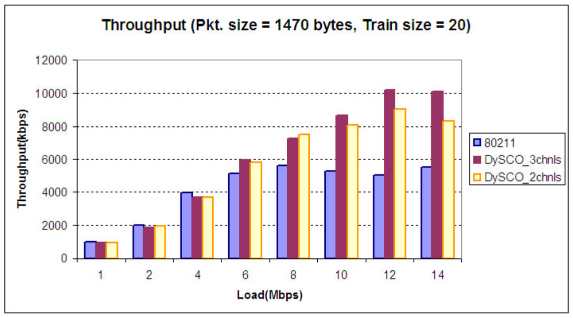


Fig. 9: DySCO-MAC vs. 802.11 MAC Performance.

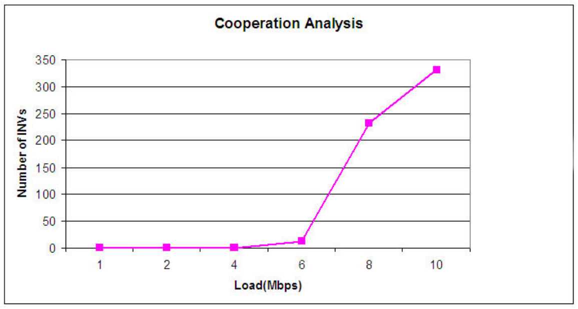


Fig. 10: Cooperation analysis: number of INVs sent.

nominal values for unused channels are likely due to noise. These results indicate that VSS can effectively detect PU and avoid false alarm (treating noise as PU). Based on the results nodes can set priorities for accessing channels. For example, unused channels have highest priority of use, followed by low traffic channels. To apply to practice, a node on startup uses VSS for 20s on all channels. Subsequently based on the sensing results, it uses PSS for 100ms on the channels that have traffic. This was used in the experiment described next.

B. *DySCO-MAC vs. 802.11 MAC Performance*

The experimental topology consisted of three Tx-Rx pairs which were placed at appropriate distances to support the 24 Mbps data rate. Three experiments were conducted. One was based on the original 802.11 MAC (RTS-CTS not used) which uses a single channel, and the other two were based on DySCO-MAC. For DySCO-MAC, one unused channel was used as the control channel and three or two unused channels (obtained as described in the end of SectionV-A) were used as data channels. We measure the aggregated throughput and show the results in Fig. 9. In the experiment of DySCO-MAC with 3 data channels, called DySCO_3chnls, which was free of channel conflict by a proper channel selection strategy (choose the previously used channel first). In the experiment on DySCO-MAC with 2 data channels, called DySCO_2chnls, channel conflict may occur within the network due to lack of cooperative nodes. We placed a dedicated cooperative node to overcome channel conflict. We used packet size of 1470 bytes and train size of 20 (11 msec Txop), and used `Iperf` to generate UDP traffic. As shown in Fig. 9, the performance of DySCO-MAC (DySCO_2chnls and DySCO_3chnls) slightly trail behind that of 802.11 when traffic load is lower than 4Mbps. This is because a single channel is sufficient for mild channel contention, and hence the control session acts as an overhead and degrades the performance. We can also see in Fig. 10 that there is no INV sent when the load is less than 4 Mbps. After that, as the traffic load increases, channel contention becomes more prominent and the single channel is no longer sufficient. As such, the throughput of 802.11 saturates below 6Mbps and if we use RTS-CTS, the performance will further degrade. For DySCO-MAC, performance is far better than 802.11 due to its multi-channel use enabled by the dynamic spectrum access MAC as compared to the fixed channel MAC use. Although the contention for the control channel also increases, it is much lower than data channel contention because of the small control-session duration. We also would like to mention again that due to the limitation discussed in IV-C, we cannot realize the full potential of DySCO-MAC. Therefore, we firmly believe that DySCO-MAC cards can achieve far better performance than the commercially available 802.11 cards.

VI. Conclusion & Future work

Cognitive dynamic spectrum access techniques can be utilized in wireless mesh backhauls to fulfill the rising broadband capacity demands. In this paper, we present the design and implementation of a MAC protocol that enables cognitive dynamic spectrum access in mesh/ad hoc networks. Our work exemplifies how an implementation based on COTS Wi-Fi cards can be achieved. In addition, the experimental results obtained from a mesh testbed based on DySCO-MAC illustrate the advantages of the technique in realistic settings. Although our implementation was constrained by the platform, it still outperformed commercial Wi-Fi cards, which ascertains far better gains in the form of manufacturers' (unconstrained) hardware implementation. To continue our research efforts, we would like to explore opportunistic use of the white spaces present in TV spectrum. This will require hardware additions in our implementation to support the 700MHz band.

VII. Acknowledgment

This work was supported by the National Research Foundation under Grants NRF2008NRF-POC001-078 (NUS WBS R-263-000-537-281) and NRF2007IDM-IDM002-069 (NUS WBS R-705-000-018-279).

References

[1] T. Luo, M. Motani, and V. Srinivasan, "Cooperative asynchronous multi-channel MAC: Design, analysis, and implementation," *IEEE Transactions on Mobile Computing*, vol. 8, pp. 338–52, March 2009.
[2] T. Luo and M. Motani, "Cognitive DISH: Virtual spectrum sensing meets cooperation," in *IEEE SECON*, (Rome, Italy), June 2009.
[3] "Ath5k project." http://madwifi-project.org/wiki/About/ath5k.
[4] http://en.wikipedia.org/wiki/IEEE_802.11h-2003.