SpiderRadio: A Cognitive Radio Network with Commodity Hardware and Open Source Software

S. Sengupta, K. Hong, R. Chandramouli and K. P. Subbalakshmi

***Abstract*—In this paper we present *SpiderRadio* a cognitive radio prototype for dynamic spectrum access networking. SpiderRadio is built using commodity IEEE 802.11a/b/g hardware and the open source MadWiFi driver. This helps us in developing and testing our prototype without having to buy and manage several licensed spectrum bands. We begin with a discussion on the key research issues and challenges in the practical implementation of a dynamic spectrum access network. Then, the lessons learned from the development of dynamic spectrum access protocols, designing management frame structures, software implementation of the dynamic spectrum access network protocol stack and testbed experimental measurement results are presented. Several trade-offs in the prototype implementation complexity vs. network performance are also discussed. We also identify potential security vulnerabilities in cognitive radio networks, specifically as applied to SpiderRadio and point out some defense mechanisms against these vulnerabilities.**

I. Introduction

Dynamic spectrum access (DSA) networking allows unlicensed users/devices ("secondary users") to opportunistically access a licensed spectrum band owned by "primary users" subject to certain spectrum etiquettes and regulations. It is expected that DSA will alleviate some of the radio spectrum scarcity problem. Cognitive radios (CR) enable spectrum sensing, dynamic spectrum access, and dynamic spectrum management. A CR senses primary licensed bands and detects the presence or absence of primary users in these bands. Then secondary users either release or occupy these primary bands depending on whether the primary users are present or not in these bands, respectively. Details on definitions and regulatory aspects of cognitive radios can be found in [1].

Practical implementation of a CR faces several challenges in terms of hardware design, software stack implementation, interfacing the CR device with policy servers, etc. Examples of some these issues are the following:

- **Synchronization:** When two communicating CRs decide to move to a new band or channel they must successfully synchronize with each other to resume communication. Therefore, implementing protocols for accurate synchronization message (e.g., available channel list) exchange, re-synchronizing in the new band as quickly as possible to prevent loss of data from upper layers, data buffering strategies during the synchronization process, planning for the situation when a new band is not available immediately, etc. must be considered.
- **Hardware delays:** Using open source software (as explained in later sections) may result in the radio hardware to be reset and restarted during channel switching. This hardware reset process configures the medium access control (MAC) layer and adapts the radio to the potentially modified transmission and reception parameters in the new band. This hardware reset/restart process causes significant delays during channel switching.

In this paper we describe *SpiderRadio*: the set-up and implementation of a software driven CR using off-the-shelf IEEE 802.11a/b/g hardware supported by Atheros chipset. The software abstraction hides the physical (PHY) layer details from the upper layers in the modified network protocol stack as discussed in Section IV. The software abstraction layer is programmable and allows SpiderRadio to configure the transmission/reception parameters automatically to operate in any unused frequency band in the allowable spectrum bands. The implication of this feature is that SpiderRadio can be on several wireless networks at the same time operating on different frequency bands. It can also be connected to an infrastructure based wireless network and an ad hoc network simultaneously. This is in contrast to current radios which can only be configured to operate statically in any one frequency band connecting to one network.

Some general guiding principles are derived based on our experience in SpiderRadio based DSA testbed experiments.

II. Related Work

Majority of current research in CR enabled DSA focuses on the theoretical aspects (see [2], [3] and references therein) with relatively fewer attempts to build working prototypes. In [4], a software defined cognitive radio prototype is developed which is able to sense spectrum in the UHF band based on waveform analysis, but no dynamic channel switching upon the detection of primary devices is explored. A feature detector design for TV bands, with emphasis on the physical (PHY) layer is presented in [5]. In [6], a cognitive radio network prototype is built based on FPGA and a virtual sensing mechanism is developed. In [7], a cognitive radio prototype is built with off-the-shelf IEEE 802.11 devices for spectrum sensing. Primary incumbent detection based on counting PHY/CRC errors is proposed. A primary device is emulated by Rohde&Schwarz sine-wave signal generator while IEEE 802.11 access cards operating as secondary devices. However, dynamic frequency switching upon the detection of primary devices is not considered here.

S. Sengupta is with the Department of Mathematics and Computer Science, John Jay College of Criminal Justice, CUNY, New York, NY 10019. Email: ssengupta@jjay.cuny.edu. K. Hong, R. Chandramouli and K. P. Subbalakshmi are with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030. Email: {khong,mouli,ksubbala}@stevens.edu.

The research in [8] and [9] investigate into adaptive channel width in wireless networks by focusing on spectrum assignment algorithms to handle spectrum variation and fragmentation. Limitations with most of the above mentioned works are that they do not comprehensively address the major DSA requirements of fast physical switching, data loss issues at the time of physical switching, synchronization failure & overhead issues and hidden incumbents challenges. Synchronization failure between two secondary devices upon switching will prove fatal for secondary network data communication as effective throughput will drop drastically or even worse, loss of communication. Thus in this work, we discuss the SpiderRadio prototype that addresses algorithmic and implementation issues for sensing based dynamic frequency switching and communication.

III. CR Prototype Implementation Challenges

A cognitive radio prototype MAC will have many features similar to any existing standard MAC, e.g., IEEE 802.11 or IEEE 802.16. However, some distinguishing requirements for dynamic spectrum access make the implementation highly challenging.

In DSA, when a cognitive radio node is switched on, it may use an etiquette such as *listen before talk* by scanning all the channels to find out whether any incumbent in the interfering zone is using any particular channel and builds a spectrum usage report of vacant and used channels. Unlike the existing single frequency radio devices (which operate using only one static frequency), cognitive radio nodes need to discover its communication peers through extensive channel scanning and beacon broadcasting [10]. Once a cognitive radio node locates broadcasts from communicating peers, it then tunes to that frequency and transmits back in the uplink direction with the radio node identifier. Authentication and connection registrations are then done gradually. Due to such extensive connection establishment procedure at the beginning, when the number of candidate frequency channels is large, the initial neighbor discovery process is likely to become highly time consuming. The available channel list may also change randomly due to the random arrival/departure of primary users in these bands. Unless the MAC layers of the communication CR nodes synchronize in a different band proactively within a certain delay threshold, network connectivity may be lost. If network connectivity is lost then the nodes must go through the highly time consuming neighbor discovering process repeatedly. An efficient and robust synchronization mechanism is thus crucial.

Another challenge for the nodes is the method and implementation to exchange the list of currently available channels and the channel to which they will resume communication upon detection of a primary in the current operating channel.

Upon a successful channel switch and synchronization the wireless card must reconfigure itself to the new frequency channel and thus it needs to stop the data flow from the upper layers. This operation will adversely affect the performance at the higher layers degrading the data throughput performance unless some remedial actions are taken to enhance the DSA MAC.

Note that, despite these challenges, dynamic channel switching must still be simple with a goal towards *fast switching*, *reduced synchronization failure*, *reduced synchronization overhead* and *increased effective throughput*.

IV. SpiderRadio System Design

As discussed before, SpiderRadio prototype is based on IEEE 802.11a/b/g wireless access cards built with Atheros chipsets. The building block for the software stack (see, Fig. 1) is the Madwifi driver (http://madwifi-project.org/). Madwifi

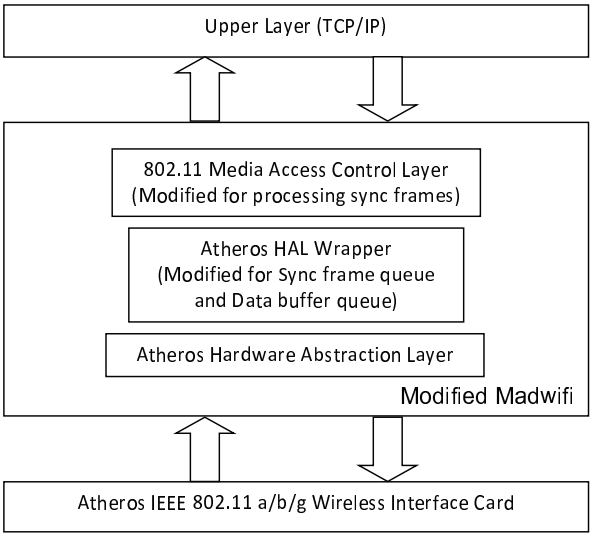


Fig. 1. Proposed protocol stack for SpiderRadio

contains three sublayers: IEEE 802.11 Media Access Control Layer, the Wrapper (an interface to lower layer) of Atheros Hardware Abstraction Layer (HAL), and Atheros Hardware Abstraction Layer (the only closed source component). For the SpiderRadio , the IEEE 802.11 Media Access Control Layer is modified to speed up and increase the reliability of channel switching, while Wrapper of Atheros HAL is modified to build special hardware queues for the prototype.

A. Modifications to Atheros HAL Wrapper

We propose and implement two special hardware queues that become active whenever any dynamic channel switching action needs to be triggered. The first hardware queue is *synchronization queue* (sync queue), which is used for transmitting synchronization management frames only. The synchronization management frames are special purpose frames and are used for synchronization between the communicating nodes at the time of switching. Whenever any of the two communicating CR nodes sense the necessity for channel switching (initiator node), it then enables the sync queue and triggers the channel switching request management frame from the sync queue with the ongoing data communication. Inside the synchronization management frame, we pack the destination channel information (called the candidate frequency

channel(s), i.e., to which the CR nodes desire to switch to upon vacating the current channel).

The second hardware queue, *data buffer queue* is enabled when the communicating CR nodes are physically switching channel and the MAC for both the nodes are being configured with the transmission and reception parameters in the new frequency band. With data buffer queue enabled, we allocate a local memory for buffering the data temporarily from upper layer so that no data from upper layer will be lost and the switching scheme will not create any adverse effect. These modifications are implemented within the Madwifi driver in such a way that dynamic channel switching in the PHY/MAC layer is hidden from the upper layers, not affecting the upper layer functionalities at all thus creating a smooth, seamless switching.

B. Extended Management Frame Structure

We use an extended management frame for dynamic channel switching and synchronization purposes between two SpiderRadio nodes. To explain the new management frame we begin with a discussion on the standard IEEE 802.11 MAC Frame Structure and MAC header [11].

In the IEEE 802.11 MAC header, a 2 bit type field indicates whether the frame is control, management or data frame, while a 4 bit subtype field indicates different subtypes of frames under one particular type of a frame. For example, with type field set for management frame, there can be 16 different subtypes of management frames. 10 different subtypes of management frames are already defined in IEEE 802.11: *beacon, probe request, probe response, association request, association response, re-association request, re-association response, disassociation, authentication and de-authentication frame*. Six more subtypes for 802.11 MAC management frames could be defined out of which we use one subtype of management frame for channel switching and synchronization. Under this subtype, 4 more extended subtypes (signifying switching request, switching response, confirmation request and confirmation response frames) are defined. Identification for these extended subtypes is built in the first two bytes of the frame body. The necessity and detailed usage of these four different extended management frames are explained in Section V.

In Fig. 2, the structure of switching request/response frame is shown. 2 byte SubType Identification field indicates this as a channel switch request/response frame. In United States, there are 3 non-overlapping channels in IEEE 802.11g, and 13 non-overlapping channels in IEEE 802.11a. 2 Byte destination channel bitmap is enough to create a bitmap for all these 16 channels. For bitmapping more number of channels, the 2 byte destination channel bitmap can be extended. The 8 byte timestamp indicates the time when this request frame is prepared for transmission.

Similar to the switching request and response frame, confirm management frames are auxiliary synchronization management frames. A node receiving a confirmation response packet will compare the current channel information from the confirmation request packet with the channel it is operating on currently. If they are the same, this node will copy the current channel and confirmation count field to confirmation response frame bit by bit and send it back to the transmitter.

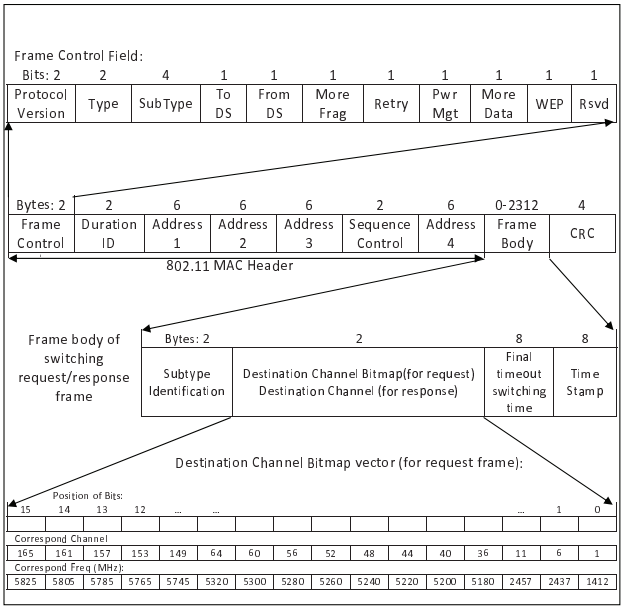


Fig. 2. Detail structure of switching request/response frame

C. Bitmap Channel Vector

In order to address the hidden incumbent problem [12], we embed the candidate channels information inside the channel switching request management frame, instead of initiator CR node attempting to convey channel switching request using only one frequency channel information. The number of candidate channels is updated dynamically by the initiator node depending on the feedback received from the receiving CR node. The reason behind transmitting synchronization message with multiple candidate frequencies is that even if the receiving CR node encounters a licensed incumbent transmission (hidden to the initiator CR node), it still has ways to choose other candidate channel(s) and report this incumbent transmission to the initiator using a similar management frame called channel switch response management frame. With this mechanism, even with the presence of hidden node incumbent, risk of synchronization failure is reduced significantly. Fig. 2 shows the proposed channel switching request management frame structure in detail. Recall that, there are 3 non-overlap channels in 802.11g, and 13 non-overlap channels in 802.11a which are emulated as primary bands in our testbed experiments. Clearly, with primary devices dynamically accessing the bands, the availability of the spectrum bands for SpiderRadio nodes changes dynamically. Since we use multiple candidate frequency channels sent by initiator nodes, embedding absolute information (spectrum band frequency) of candidate frequency channels would again invoke the challenge of variable length management frame. This may consume more time to decode the header information for variable length frames.

In order to solve this issue, we use a *bitmap channel vector*

for sending candidate channel(s) information. Since we have 16 non-overlapping channels, we implement the length of this bitmap channel vector as two bytes in the MAC payload as shown in Fig. 2, thus mapping the availability of each channel to a single bit. When a channel is available (candidate), the corresponding bit will be set to 1; otherwise, it will be set to 0.

Note that, the advantage of using bitmap channel vector for transmitting candidate channel information is that fixed length management frame can be used even though the channel availability information is variable length. The fixed length bitmap channel vector is sufficiently easy and quick to decode. Moreover, with the usage of bitmap channel vector, the management frame becomes easily scalable. If there are more than 16 non-overlapping channels in any system, we only need to expand the programmable bitmap vector field for that system. Following the destination channel bitmap vector field, the next field signify the *final timeout switching time* from initiator's perspective. This field is designed to indicate when the initiator node will timeout from the current synchronization mechanism (if no synchronization could be established; i.e., even after multiple switching requests, no response frame is received from the other communicating CR node), will vacate the current channel and start the re-synchronization attempt through quick probing following the destination (candidate) channel bitmap vector.

V. Dynamic Frequency Switching Implementation

Two SpiderRadio secondary devices communicating with each other on a frequency channel must vacate the channel upon detecting the arrival of a primary device (or for co-existence) on that particular channel and must switch to a new channel to resume communication. To enable efficient spectrum switching, each node maintains the spectrum usage report in a local *spectrum usage report database* (SURD), which keeps track of the bands occupied by the primary user or available open spectrum bands. When a channel switching event is triggered the secondary devices have three requirements:
(i) Switch as fast as possible to reduce wastage of time and resume data communication quickly.
(ii) Switch successfully to reduce synchronization failure so that the nodes do not end up in different channels and lose communication.
(iii) Keep the overhead for synchronization as small as possible to maximize effective data throughout.

With the above goals in mind, we discuss next the implementation of three gradually improving versions of channel switching protocols for SpiderRadio in increasing order of complexity. Note that, each version is more robust, complex and requires more overhead than its predecessor.

A. Dynamic Frequency Switching: Version 1

In version 1, two SpiderRadio nodes communicating on a frequency channel upon detecting a primary user activity on this particular channel trigger frequency switching procedure and move to a new channel. The dynamic frequency switching procedure is initiated by one of the SpiderRadio nodes which transmits a *channel switching request management frame* to the other node for synchronization. It then moves to the new channel. (Channel switching request management frame is explained in detail in section IV.) The node initiating channel switching request management frame is called the *Initiator SpiderRadio*, while the other node is called *Receiver SpiderRadio*. Receiver SpiderRadio upon receiving the channel switching request management frame switches to the new channel indicated in the payload of the management frame and re-synchronizes with the Initiator. In Fig. 3(a), we present the synchronization procedure for channel switching request management frame.

The advantage of this method is its simplicity and reduced overhead. Moreover, dynamic synchronization is possible with initiation of channel switch request management frame carrying the new channel information thereby making channel switching quite fast.

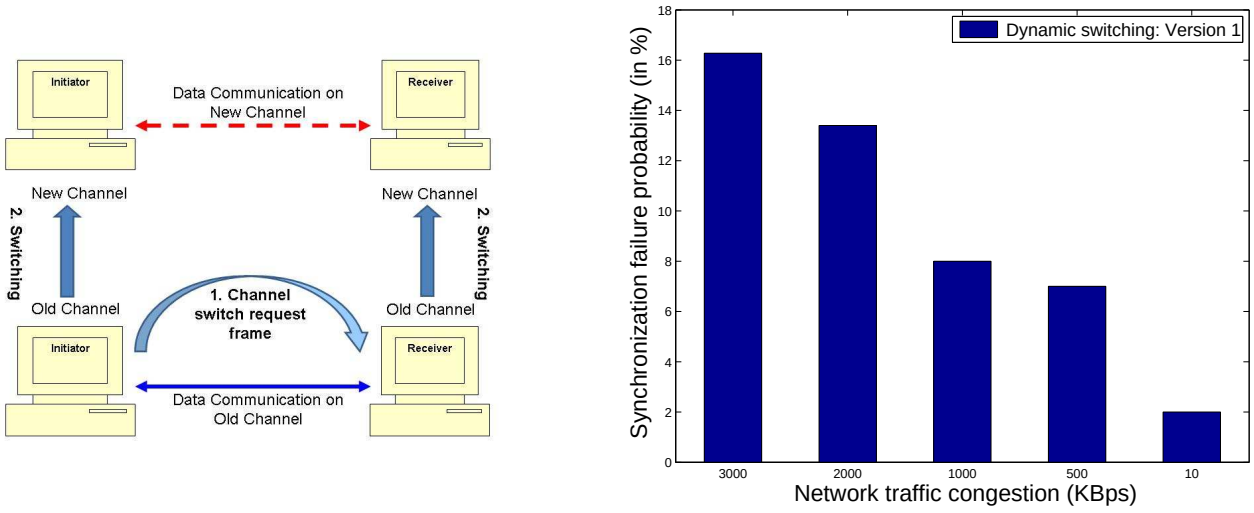
However, this method also has its drawbacks in terms of robustness, as follows:

(i) As synchronization is heavily dependent on the channel switch request management frame, if this frame is lost, there is a high probability of synchronization failure as the initiator would end up being in the new channel whereas the receiver will still be in the old channel, resulting in the loss of communication. (ii) As initiator initiates the channel switch request management frame, the new channel information (to which the nodes would move to) is inserted in this frame by the initiator from its local SURD. The problem with such a protocol is that the receiver may have a primary device operating in the new frequency channel in its vicinity, however, the initiator does not have any information about this in its local SURD. As a result, the initiator would again end up being in the new channel whereas the receiver will still remain to be in the old channel thereby resulting in the loss of communication. (iii) Another key issue of this method is the determination of the initiator node. If both communicating SpiderRadio nodes detect the arrival of a primary device and simultaneously initiate channel switch request management frame with information about the new channel from their local SURDs, there is a probability that synchronization failure will happen. As both the SpiderRadio nodes now act as initiators without knowing the status of other node, both the nodes will move to their specified new channels. Unless new channels selected by both the initiators are same, synchronization failure is bound to happen.

In Fig. 3(b), we present the experimental synchronization failure probability of version 1. We find that with the network traffic decreasing, the synchronization failure probability decreases as well. However, as observed from Fig. 3(b), synchronization failure probability in version 1 is very high thus making this version a poor choice for SpiderRadio.

B. Dynamic Frequency Switching: Version 2

In this method, synchronization is no longer dependent on channel switching request management frame only. We



(a) (b)

Fig. 3. a) Dynamic channel switching with switching request management frame; b) Synchronization failure probability results for version 1

introduce another management frame called channel switch response management frame (as discussed in detail in section IV).

The synchronization protocol in version 2 is summarized as follows.

Step 1: Initiator sends channel switching request management frame carrying information of new channel(s)
Step 2: Upon successful reception of the request frame, responder transmits channel switching response management frame carrying information about the agreed new channel back to initiator as acknowledgment; then, responder switches to new channel
Step 3: Initiator after receiving response frame switches to the new channel.

As the initiator switches only after receiving the response management frame, the chance of synchronization failure reduces significantly (see, Fig. 4(a)). Using version 2, we can also solve the problem of both nodes being initiators. If both the SpiderRadio nodes initiate channel switch request management frame, the one with earlier timestamp will win. The eight byte timestamp field from the switching request management frame will let both the nodes decide on the winner and other node will automatically follow the role of responder.

Even with the enhancement, version 2 faces a drawback in terms of higher channel switching request frame loss probability as shown in Fig. 4(b). This is because only one channel switching request frame is transmitted for channel switching initiation thus making the switching request frame highly loss-prone.

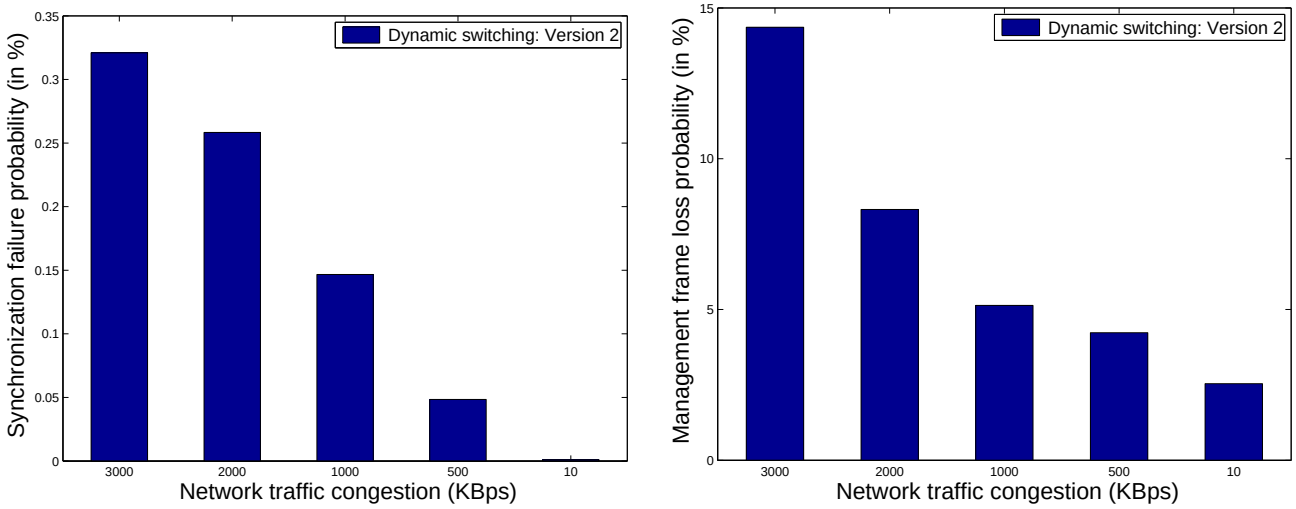
C. Dynamic Frequency Switching: Version 3

To avoid the synchronization failure due to the loss of the channel switching response management frame, we introduce two more types of management frames, *confirm request management frame* and *confirm response management frame*. In Fig. 5(a), we present the version 3 implementation of the dynamic channel switching protocol.

The synchronization protocol in version 3 is summarized as follows.

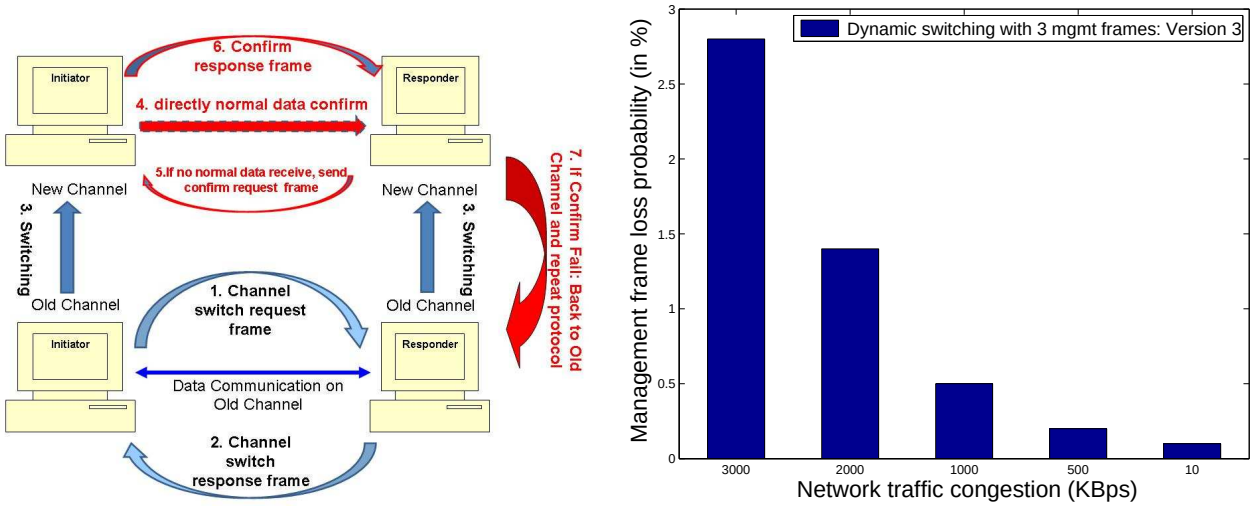
Step 1: Initiator sends channel switching request management frame(s) carrying information of new channel(s)
Step 2: Upon successful reception of the request frame, responder initiates channel switching response management frame carrying information about the new channel back to initiator as acknowledgment; then, responder switches to the new channel
Step 3: Initiator, after receiving the response frame will switch to the new channel
Step 4: Responder will monitor the data communication on the new channel
Step 5: If no data communication is received from the initiator, responder will send a confirm request management frame to the initiator
Step 6: If the initiator is on the new channel, it will send confirm response frame and communication on new channel will resume
Step 7: If responder could not receive confirm response, it will consider that the initiator is in the old channel, will go back to old channel and try to repeat the protocol if it is within the time threshold permitted by primary device standard.

In order to make version 3 more robust, we also configure SpiderRadio so that the initiator sends multiple channel



(a) (b)

Fig. 4. a) Synchronization failure probability with version 2; b) Channel switching request frame loss probability with version 2



(a) (b)

Fig. 5. a) Dynamic Frequency Switching: Version 3; b) channel switching request frame loss probability with version 3

switching request management frames to reduce the loss probability of switching request frames. In our experiment, we program the initiator to transmit 3 switching request management frames. The result is presented in Fig. 5(b), which shows very low loss probability for switching request frames thus making version 3 highly robust. From experiments, we also compare the synchronization failure probabilities of all the three versions. The synchronization failure probability (in percentage) for version 3 even under very high network traffic congestion (approx. 3000 KBps) turns out to be almost negligible (0.0050%) as compared to version 1 and 2.

VI. Testbed Setup and Experimental Results

For conducting extensive experiments with SpiderRadio enabled nodes, we built two groups of SpiderRadio prototypes: one for *indoor testing* and the other for *outdoor testing*.

Each node of the indoor group is a standard desktop PC running Linux 2.6 operating system. They were all equipped with Orinoco 802.11 a/b/g PCMCIA wireless card. There is no PCMCIA slot for the desktop PC; so we make use of an ENE-CB1410 PCMCIA to PCI adapter card for converting the PCMCIA devices to operate on the desktop PC.

In the outdoor group, SpiderRadio is deployed on two laptops running Linux 2.6 operating system: Compaq NC4010 and Dell Inspiron 700m. Both of them were equipped with

Orinoco 802.11 a/b/g PCMCIA wireless card. The TX powers of these wireless devices were set to 100mW. Another laptop running Windows VISTA and equipped with Wi-Spy 2.4x, acted as a monitor in the test bed. These Orinoco devices are equipped with Atheros 5212 (802.11 a/b/g) chipsets.

For our testbed setup, the primary user bands were emulated using the 900MHz, 2.4GHz and 5.1GHz Wi-Fi spectrum bands. The primary user communication was emulated in two different ways: two cordless phones communicating with each other using the intercom feature and Agilent signal generator (e4437b) operating in the Wi-fi bands. The SpiderRadio node was configured to be the secondary user device for the experiments. For the purpose of sensing and detecting the arrival of primary user, we implemented the spectrum sensing methodology based on observed PHY errors, received signal strengths and n-moving window strategy as proposed in our earlier work [13]. We placed SpiderRadio nodes at a distance of $5 - 20$ meters from each other, communicating with TCP data streams. We carried out experiments under five different network traffic congestion scenarios: 3MB/s, 2MB/s, 1MB/s, 0.5MB/s, and 10KB/s during day and night times. Note that since the testbed is located in Hoboken (in close proximity to Manhattan, New York City) the radio interference is significantly different during day and night. Interference due to students using the Stevens campus wireless network also varies significantly between night and day.

In Fig. 6(a), we present the average time to synchronize under all five network traffic congestion scenarios. For showing the effectiveness of version 3 with 3 switch request management frames, we compare this with a simpler version 3 where only 1 switch request management frame is transmitted. The comparison result is shown in Fig. 6(a). The first observation from this plot is that when the network traffic congestion decreases the average time to synchronize also decreases for both the mechanisms. However, the more interesting observation is that, with higher network traffic, version 3 with 3 management frames performs much better than the older version 3 (i.e., with 1 management frame). The difference in the performance decreases gradually with a decrease in the network traffic congestion. At the lowest network traffic congestion (i.e., 10 KBps), version 3 with 1 management frame results in a better performance as that with 3 management frames. This is because with very low network traffic, the loss probability of management frame is very low, the need for redundant management frames to reduce loss probability is no longer needed. Thus it can be concluded that at night time (or when network traffic congestion is very low or almost zero), version 3 with 1 management frame might be a better choice compared to 3 management frames to reduce overhead for the same performance.

The effective throughput is shown in Fig. 6(b). The results are shown for different switching intervals (1, 2, 3, 5 and 10 seconds). For benchmarking purposes, we calculate the ideal maximum throughout achievable under the same operating environment and conditions without any frequency switching. The dotted line in the figure depicts the maximum possible throughput (3.353 MB/second – benchmark). As evident from the figure, the proposed CR system demonstrates high throughput even with very high frequent switching and as obvious, with less frequent switching (switching every 5 seconds or 10 seconds), the throughput achieved is almost same as the benchmark throughput; proving the effectiveness of the proposed CR prototype.

VII. Future Directions: Securing the SpiderRadio

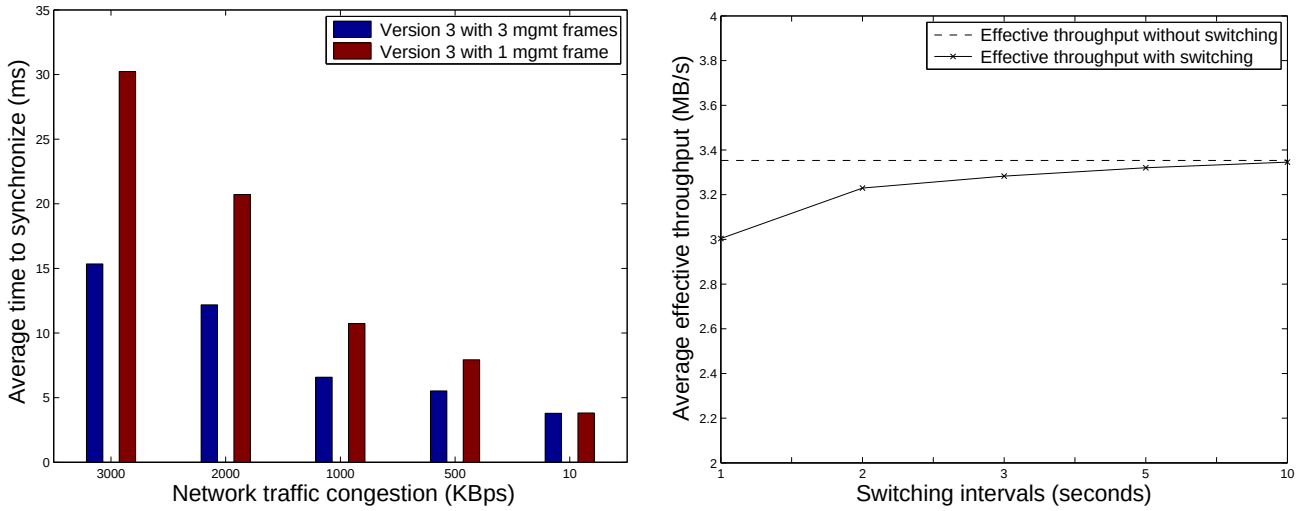
Recent research in the area of cognitive radio security [14] has underlined the need to consider security in the design stages of the CRN. Several flavors of denial-of-service (DoS) attacks can be launched on the CRN if the architecture and protocols are not designed specifically to avoid these problems. In keeping with that spirit, we discuss some potential security issues that the Spider Radio will need to address and some potential solutions to these problems.

Since the primary differentiating factor between wireless networks and CRN is the need to sense and switch between spectrum bands, most of the unique security threats to CRN come from these two functionalities. We will focus on the vulnerabilities existing in switching/synchronization functionality of CRNs., rather than those that occur in sensing [15].

In Section V, several protocols for resynchronization have been proposed for SpiderRadio. All the protocols assume that the channel to which the Initiator and Receiver SpiderRadios must switch (rendezvous), has already been determined. A malicious node with an intent to jam or desynchronize communication between these nodes can easily do so with very minimal resource expenditure on its part, by tracking the two communicating nodes and successively jamming these channels. In order to prevent this type of DoS, the security of the rendezvous sequence must be guaranteed at least to some extent. Several solutions will have to be co-opted to achieve this goal. These include - a secure pseudo-random rendezvous sequence with meaningful convergence guarantees (to ensure faster channel synchronization), efficient cryptographic authentication of the switching request frame as well as response management frame (in Version 2 of the protocol) and confirmation frames (Version 3). All solutions will have to be optimized for time-to-convergence.

VIII. Conclusions

SpiderRadio's software abstraction-based implementation platform at the MAC layer hides the PHY layer details from the higher layers in a network protocol stack efficiently. The special purpose queues built into the stack help alleviate higher layer packet losses during dynamic channel switching. The three versions of the proposed dynamic channel switching protocols seem to gracefully trade-off complexity for achievable throughput. These protocols also achieve fast channel switching with negligible synchronization failure rate between the transmitter and the receiver. The empirical throughput observed in testbed experiments for different channel switching intervals is close to the ideal throughput without fixed channel access. This implies that the channel switching protocol and implementation in SpiderRadio is fast enough for practical dynamic spectrum access networking applications.



(a) (b)

Fig. 6. a) Average time to synchronize; b) Average effective throughput with various frequency intervals

ACKNOWLEDGEMENT

This work is supported by grants from the National Institute of Justice #2009-92667-NJ-IJ, the National Science Foundation #0916180 and PSC-CUNY Award #60079-40 41.

REFERENCES

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey," *Comput. Netw.*, vol. 50, no. 13, pp. 2127–2159, 2006.

[2] H. Salameh, M. Krunz, and O. Younis, "Mac protocol for opportunistic cognitive radio networks with soft guarantees," *IEEE Transactions on Mobile Computing*, vol. 8, no. 10, pp. 1339–1352, Oct. 2009.

[3] S.-Y. Tu, K.-C. Chen, and R. Prasad, "Spectrum sensing of ofdma systems for cognitive radio networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 7, pp. 3410–3425, Sept. 2009.

[4] H. Harada, "A software defined cognitive radio prototype," *IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–5, 2007.

[5] R. DeGroot, D. Gurney, K. Hutchinson, M. Johnson, S. Kuffner, A. Schooler, S. Silk, and E. Visotsky, "A cognitive-enabled experimental system," *IEEE DySPAN*, pp. 556–561, 2005.

[6] Y. Yuan, P. Bahl, R. Chandra, P. Chou, J. Ferrell, T. Moscibroda, S. Narlanka, and W. Yunnan, "KNOWS: Cognitive radio networks over white spaces," *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, pp. 416–427, April 2007.

[7] K. Shin, H. Kim, C. Cordeiro, and K. Challapali, "An experimental approach to spectrum sensing in cognitive radio networks with off-the-shelf ieee 802.11 devices," *4th IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 1154–1158, Jan. 2007.

[8] P. Bahl, R. Chandra, T. Moscibroda, R. Murty, and M. Welsh, "White space networking with wi-fi like connectivity," *in Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pp. 27–38, 2009.

[9] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl, "A case for adapting channel width in wireless networks," *in Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pp. 135–146, 2008.

[10] C. Cordeiro, K. Challapali, D. Birru, and S. Shankar, "IEEE 802.22: the first worldwide wireless standard based on cognitive radios," *IEEE DySPAN*, pp. 328–337, 2005.

[11] "IEEE standard for information technology - Telecommunications and Information exchange between systems - Local and Metropolitan area Networks-Specific requirements - Part 11: Wireless Lan medium access control (MAC) and physical layer (PHY) specifications," March 2007.

[12] S. Sengupta, S. Brahma, M. Chatterjee, and S. Shankar, "Enhancements to cognitive radio based ieee 802.22 air-interface," *IEEE International Conference on Communications (ICC)*, pp. 5155–5160, 2007.

[13] K. Hong, S. Sengupta, and R. Chandramouli, "Spiderradio: An incumbent sensing implementation for cognitive radio networking using IEEE 802.11 devices," *IEEE International Conference on Communications (ICC)*, 2010.

[14] F. Granelli, P. Pawelczak, R. V. Prasad, K. Subbalakshmi, R. Chandramouli, J. A. Hoffmeyer, , and H. S. Berger, "Standardization and research in cognitive and dynamic spectrum access networks: IEEE SCC41 efforts and open issues," *IEEE Communications Magazine*, Jan. 2010.

[15] G. Jakimoski and K. Subbalakshmi, "Towards secure spectrum decision," *IEEE International Conference on Communications, Symposium on Selected Areas of Communication*, June 2009.

BIOGRAPHIES

Shamik Sengupta (ssengupta@jjay.cuny.edu) is an Assistant Professor in the Department of Mathematics and Computer Science, John Jay College of Criminal Justice of the City University of New York. Shamik Sengupta received his B.E. degree (First class Hons.) in Computer Science from Jadavpur University, India in 2002 and the Ph.D. degree from the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando in 2007. His research interests include cognitive radio, dynamic spectrum access, game theory, security in wireless networking, and wireless sensor networking. Shamik Sengupta serves on the organizing and technical program committee of several IEEE conferences. He is the recipient of an IEEE Globecom 2008 best paper award.

Kai Hong (khong@stevens.edu) received his B.S. degree in Automatic Control from Beijing Institute of Technology in 2004 and his M.S. degree in Automatic Control from Beijing Institute of Technology in 2007. He is currently a PhD candidate in Computer Engineering at Stevens Institute of Technology where he is a member of the Media Security,

Networking, and Communications (MSyNC) Lab. His research focus is in the area of Cognitive Radio, dynamic spectrum access networks and wireless security.

R. Chandramouli (mouli@stevens.edu) is a Professor in the Electrical and Computer Engineering (ECE) Department at Stevens Institute of Technology. His research in wireless networking, cognitive radio networks, wireless security, steganography/ steganalysis, and applied probability is funded by the NSF, U.S. AFRL, U.S. Army, ONR, and industry. He served as an Associated Editor for IEEE Transactions on Circuits and Systems for Video Technology (2000-2005). Currently, he is the Founding Chair of the IEEE COMSOC Technical Sub-Committee on Cognitive Networks, Technical Program Vice Chair of the IEEE Consumer Communications and Networking Conference (2007), and Chair of the Mobile Multimedia Networking Special Interest Group of the IEEE Multimedia Communications Technical Committee.

K.P. (Suba) Subbalakshmi (ksubbala@stevens.edu) is an Associate Professor in the Dept. of E.C.E at Stevens Institute of Technology. Her research interests lie in cognitive radio networks, wireless security and information forensics and security. Her research is supported by grants from US National Science Foundation, National Institute of Justice and other DoD agencies. Suba serves as the Chair of the Security Special Interest group of IEEE ComSoc's Multimedia Technical Committee. She has given tutorials on cognitive radio security at several IEEE conferences and has served as Guest Editior for several IEEE special issues in her area of interest.