

Mesh Testbed for Multi-channel MAC Development: Design and Experimentation

Shashi Raj Singh
Electrical and Computer Engineering
National University of Singapore
Singapore
singh_shashiraj@yahoo.com

Mehul Motani
Electrical and Computer Engineering
National University of Singapore
Singapore
motani@nus.edu.sg

ABSTRACT

Multi-channel access presents a huge potential to boost the performance of 802.11 mesh/adhoc networks. It allows multiple simultaneous transmissions in a given radio neighborhood that improves network throughput and scalability. Still, its research is limited to simulations only and its experimental study is rare. To bridge this gap, we have setup a testbed called *OpenWireless* to develop and test multi-channel MAC protocols in a mesh/adhoc scenario. The testbed consists of 20 nodes that are based on commodity hardware and use the Linux 802.11 networking architecture for wireless applications. In this paper, we present the design of the testbed and development experience gained. We will discuss experimental studies that identify some of the issues involved in using commodity hardware for the multi-channel MAC development. In the end, the paper also discusses the implementation of our multi-channel MAC protocol and its experimental evaluation that validates the potential of the multi-channel access in mesh/adhoc networks as compared to the traditional single channel access.

Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]: [Wireless communication]

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Mesh testbed, IEEE 802.11, multi-channel access, orthogonal channels, close interface effect, software MAC, Linux, ath5k.

1. INTRODUCTION

In the next few years, we will see a number of 802.11 mesh/adhoc networks around us due to introduction of IEEE 802.11s and WiFi Direct technologies. Current 802.11 networks work only in the infrastructure mode where end user devices do not interact. The new technologies will allow networks based on mesh/adhoc mode

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WiNTECH'10, September 20, 2010, Chicago, Illinois, USA.
Copyright 2010 ACM 978-1-4503-0140-4/10/09 ...\$10.00.



Figure 1: A customized PC.

where end user devices will interact. This will lead to plethora of new devices and applications that will further boost the usage of the WiFi technology. To take advantage of this emerging WiFi landscape, research is already going on to create high performance 802.11 mesh/adhoc networks.

One approach to boost the network performance is the use of multi-channel access in the network. In a mesh/adhoc scenario where multiple devices try to communicate, traditional approach of the single channel use can act as a bottleneck to the network throughput and scalability. Also, the 802.11 standard theoretically has 3 and 20 orthogonal channels in 2 and 5GHz bands respectively. But the 802.11 network uses only single channel even when other channels may be free, which is very inefficient. By employing multi-channel access in 802.11 mesh/adhoc networks, better performance can be achieved by using all the possible channels in the mentioned bands.

Multi-channel access enables a node to access multiple channels and thus allows multiple simultaneous transmissions in a given radio neighborhood. This boosts the network throughput and allows more nodes in the network. To realize its potential, a number of multi-channel MAC design approaches [1, 3, 7–10, 12, 13, 15] have been proposed in the past. These studies are based on simulations only and experimental evaluation of multi-channel access is rare. To bridge this gap and understand the potential of our multi-channel MAC [9] in the real world, we have setup a wireless mesh testbed that is called *OpenWireless* testbed. A node in the testbed consists of commodity hardware and uses the Linux 802.11 networking architecture for wireless protocol development and application. In this paper, we first present the testbed details including the node design issues involved. Then we discuss two experimental studies, one on the number of orthogonal channels in the 5 GHz band and the other on *close interface effect* present in a commodity hardware. The former study explores the actual num-

ber of orthogonal channels in the 5 GHz band that can be used as data channels for a 802.11 multi-channel MAC. The later discusses about the interference due to the close PCI interfaces present in a commodity hardware and its effect on the number of orthogonal channels. In the end, we detail our multi-channel MAC protocol implementation and its experimental evaluation. Initial experiment results clearly demonstrate the potential of the multi-channel access over the traditional single channel access in 802.11 mesh/adhoc networks. Overall, the paper demonstrates how a commodity hardware can be used to setup such a testbed and issues likely to be faced, and how one can approach a multi-channel MAC implementation using the Linux 802.11 networking architecture.

We present some past work on mesh testbeds in Section 2 and compare them with our testbed. In Section 3 and Section 4, we discuss the design of our testbed and the development issues faced respectively. Section 5 presents the two experimental studies. Then in Section 6, we discuss the multi-channel MAC design, the implementation approach and its limitations and benefits. In Section 7, we present system evaluation of the new MAC including performance comparison with the 802.11 MAC. Section 8 concludes the paper and discusses our future work.

2. RELATED WORK

In this section, we discuss some of the prior works on the mesh testbed and would try to differentiate our work with them. MIT Roofnet [4] is an experimental multi-hop 802.11 b/g mesh network prototype which provides broadband Internet access to users in Cambridge, MA. It consists of about 50 nodes among which a few act as gateways to the wired Internet. Each node has a single antenna and a single Ethernet port. A routing protocol is designed such that the longest route is only four hops long. Extensive measurements have been conducted on Roofnet. For instance, [2] analyzes the patterns and causes of packet loss.

An urban mesh network is deployed in Houston which provides high-speed Internet access to low-income communities. Camp et al. [5] present measurements from the network, focusing on PHY layer issues. The study aimed at characterizing the propagation environment and correlating received signal strength with application layer throughput.

In [6], the authors deployed an initial testbed consisting of 7 mesh nodes to solve the communication needs of a traffic control system called SCATS in Sydney, Australia. They placed nodes at intersections with traffic lights, and carried out initial trial experiments. The feature of the study is its focus on real-time communication.

The WAND project [14] built a multi-hop wireless ad hoc testbed in the center of Dublin, where 11 nodes were mounted on traffic lights along a 2km route in urban area. Their topology is a chain topology which is not usually the case in a mesh context.

MeshCluster [11] is a self-configuring and secure infrastructure mesh network architecture, in which each nodes is equipped with multiple radios. A subset of radio interfaces are used for providing network access to end-devices and other radio interfaces are used for forwarding packets to the nearest Internet gateway.

Our work differs from prior work of mesh testbeds in the following aspects: (1) prior testbeds are based on 802.11 MAC, but we develop a new multi-channel MAC layer on top of the 802.11 PHY, (2) prior work focuses on deployment and measurement issues and the development work was in the routing layer, our work is in the MAC layer and focuses on throughput enhancement by multi-channel use, (3) regardless of using single or multiple radios, prior testbeds employ single-channel radios, whereas our testbed is built on multi-channel radios and we measure the effect from us-

ing multiple radios in a node, and (4) prior work uses the 2.4 GHz ISM band, but our work uses the 5 GHz band that provides more available channels for a multi-channel MAC.

3. OPEN WIRELESS: THE TESTBED

The testbed consists of 20 nodes called customized PCs (CPC), deployed in 4 adjacent labs. A node has two radios, which allows it to act as a mesh access point (MAP) or as a mesh point (MP). As mentioned before, a node is based on commodity hardware and the Linux 802.11 networking architecture that comprises of the *Linux 802.11 subsystem*, *open source drivers* and *commercial WiFi cards*. Initially, we had two options for the node platform: 1) software defined radio and 2) Linux 802.11 networking architecture. Software defined radio platform (WARP, GNU radio) provides a great degree of implementation freedom and flexibility as both the physical and the MAC layers are reconfigurable. But mostly, it is not interoperable with the IEEE 802.11 standard/WiFi commercial products. If the goal is to evaluate and compare the performance against commercial product, it is not a suitable choice. The Linux 802.11 networking architecture provides implementation flexibility that varies with the OSI layers. The PHY layer that resides in the card cannot be changed. MAC layer and above are flexible and can be modified to adopt new designs. The advantage it has over the SDR platform is that implementations based on it can be compared to the commercial products and so more acceptable. Due to this, it is frequently used for development and testing of MAC and routing protocols based on the IEEE 802.11 standard. Our goal was to compare multi-channel MAC performance with commercial 802.11 products, so we chose the Linux 802.11 networking architecture. In the next subsections, we will discuss our MAC protocol development approach and, the testbed hardware and software.

3.1 MAC development: Software MAC

We use the Linux 802.11 architecture for our MAC protocol development. The architecture consists of Linux 802.11 subsystem, open source driver and WiFi cards. The latest 802.11 subsystem is mac80211, which we use in our testbed along with another subsystem, net80211. We use Madwifi and ath5k open source drivers for Atheros chipset based WiFi cards.

In the Linux 802.11 architecture shown in Fig. 2(a), the IEEE 802.11 MAC protocol is logically divided into two modules. One module is in the card and implements control features that are time-critical functions like distributed coordination function(DCF)). The other module is in the form of the kernel subsystem, which is responsible for more delay-tolerant management sublayer functions like authentication, association, etc., and 802.11 frame management. The driver acts as an interface between the kernel world and the wireless card and configures the PHY layer parameters, hardware queues and the MAC control unit in the wireless card.

For MAC development, we disable the 802.11 control unit on the card and implement the new MAC design completely in the driver as a *software MAC* as shown in the Fig.2(a). This demands high processing power and sometimes microsecond time resolution (based on the MAC design) for running the software MAC state machine.

3.2 Testbed: Node Hardware

The CPC is shown in Fig.1, its hardware specification was chosen in an attempt to reduce its size without limiting it in terms of processing power. We use Biostar G3I-M7 TE socket microATX mainboard and core duo E5400 Intel processor (2.7 GHz) to make up the base system. We chose a 4GB CF card as opposed to a hard disk, which was sufficient to house an Ubuntu server edition oper-

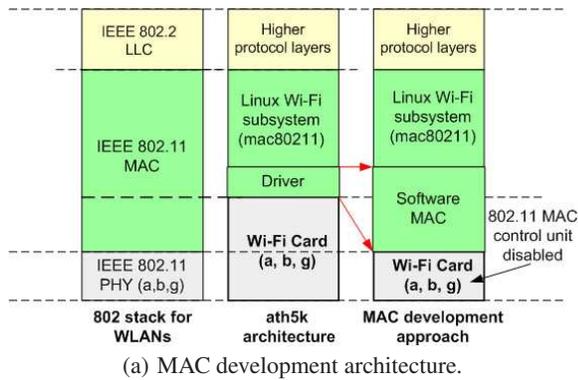


Figure 2: Development architecture & Close interface.

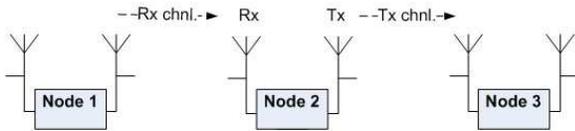


Figure 3: Close interface effect experiment.

ating system and also support protocol development activities. The board provides two PCI interfaces that we use as wireless interfaces by using Atheros based PCI WiFi cards. We use a combination of 2.4 GHz b/g and 5 GHz a/b/g WiFi cards. In addition to the attached hardware, the board comes with a number of useful on-board features. It allows on site configuration by providing a VGA port and USB ports. This combination of interfaces can allow a monitor and a keyboard to be attached for an operator access. An on-board Ethernet interface is used for remote configuration of the nodes in the testbed.

3.3 TestBed Software

A minimalistic Ubuntu server edition operating system is installed on the CPCs. We use ath5k driver and mac80211 for all the protocol development and for creating mesh point interface. For the AP interface, we use the hostapd daemon and the Madwifi driver. The testbed control software is a Java based GUI on top of the openssh server-client application, which is used to monitor node health, remote setting of a number of node's configurations/parameters and, experimentation and statistics collection. The server resides on each of the CPCs, whereas the client with the GUI resides on the control workstation. All communication between the control workstation's client program and servers on the CPCs is carried out on an Ethernet network. The Java GUI application uses the *iw configuration utility* commands to configure wireless interfaces of a node. We also add new command options in the *iw utility* based on our requirements.

4. CPC DESIGN ISSUES

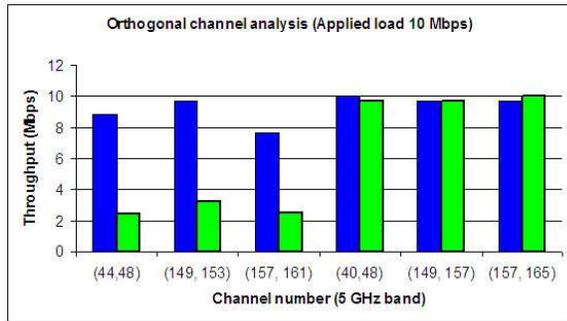
In the section, we discuss the main issues faced during the testbed node design process and the corresponding solutions adopted.

1. *Processing power & hrestimer support*: There are many COTS boards that support the Linux 802.11 architecture for wireless applications. Soekris net4521 was an early candidate for our testbed node. During the MAC protocol development process, we found that the board was not capable of meeting *software MAC* (Fig.2(a)) development requirements. The board comes with a 133MHz 486 class processor, 64MB

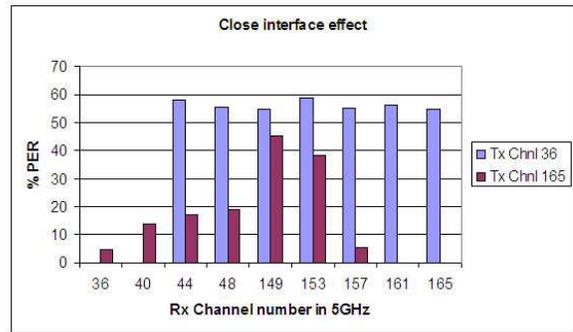
RAM, an Ethernet port and two cardbus interfaces for wireless applications. Although the hardware components met our specifications, the processing power fell short. We found that a great deal of packet dropping would take place for a *software MAC* implementation due to delayed event firing. Also, we were not able to configure and use the *high resolution timer* (hrestimer), which is critical for the implementation of a *software MAC* with strict timing requirements (micro second resolution). Apart from this, the RAM was a very tight fit for an operating system. In our initial work, we ran the Gentoo OS on the board and found that the low processing power forced us to carry out all CPU intensive activities such as installation and compiling on a separate workstation.

After this, we decided to build our own node based on commodity hardware. From the experience of the Soekris board, we realized that high processing power is required for the implementations where the MAC state machine runs in the driver. Some *software MAC* implementation will require microsecond resolution to run, which can only be provided by hrestimers. Standard Linux timers that are HZ based cannot be used in such implementations as their resolution is limited to millisecond. We decided to go for a much higher processing power board with hrestimer support. We did some research on the hrestimer support and came to the conclusion that the requirements can be divided into hardware requirement and software requirement. In the former case, hrestimer requires *time stamp counter*(TSC) to function. TSC is present in all x86 processors since the Pentium. In the later case, HRT API can only be used if CONFIG_HIGH_RES_TIMERS is enabled in the kernel, which was merged in kernel 2.6.21. So based on the requirements, we looked for a suitable board that is described in the section 3.2.

2. *Reset on hang*: In a testbed deployed over a large area, capability of a node to reset on hang is a must. During protocol development stages, it is frequent that nodes hang and so remote reset is not possible. Manually resetting the node everytime it hangs, can be hectic. For the solution, we use the *watchdog timer* present on the board that resets the node if it does not receive pulses from a program for a given duration of time.
3. *Close Interfaces*: The PCI interfaces that we use as wireless interfaces are placed close on commodity boards, as shown in Fig. 2(b). This arrangement creates interference issue under certain operational conditions. If we only use a single interface or use both the interfaces in different bands like one



(a) Orthogonal channels results.



(b) Close interface results.

Figure 4: Experiment results.

in the 2.4 GHz and the other in the 5 GHz band, there is no interference issue. But if both the interfaces work in the same band and traffic load is high, we find interference and throughput degradation, which is irrespective of the channel spacing between the channels in use. To resolve the issue, we use cable to put the antennas at a distance. We also tried using PCI extender to put the interfaces apart, but they cause system hang. We will discuss the effects of close interface in detail in the next section.

4. *Channel switching delay*: Multi-channel access requires minimum possible channel switching delay, preferably less than 200 microseconds. Open source drivers do not have any function specifically for fast channel switching. Most of them reset the card to change channel that incurs millisecond order delay, which is a huge overhead for a multi-channel MAC. After some research, we found that some Atheros chipsets support fast channel switching. The method called synth only change is much faster than resetting the whole chip. We use the card with the required chipsets and introduced a new function in the driver to implement synth only change. Our implementation takes around 600 microseconds for channel switching. Since the COTS WiFi cards are not optimized for fast channel switching requirements, we cannot achieve values close to 200 microseconds or less without having packet loss.

5. EXPERIMENTAL STUDIES

Orthogonal data channels are necessary requirement for multi-channel MAC designs. 802.11 networks use 2.4 and 5 GHz bands for their operation, so a multi-channel MAC will use the orthogonal channels present in these bands as data channels. In this section, we analyze the orthogonal channels present in the 5GHz band. We will also look into the effects of close positioning of PCI interfaces in a commodity hardware on the number of orthogonal channels. In the experiments we use the ath5k driver in mesh mode and use the Iperf to generate UDP traffic with a packet size of 1470 bytes. We mainly investigated 5 GHz band in our studies.

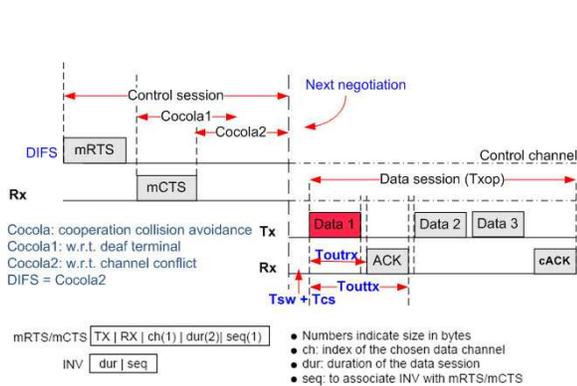
5.1 Orthogonal channels

In theory, the 5GHz band has 20 orthogonal channels assuming standard 802.11 channel bandwidth, but based on the regulatory domain the number will vary. For our domain, theoretically we have the orthogonal channels shown on the x-axis in Fig. 4(b). To establish their orthogonal behavior, we did a basic experiment of UDP traffic transmission using two Tx-Rx pairs. Through experiments we found that transmissions on two consecutive orthogonal chan-

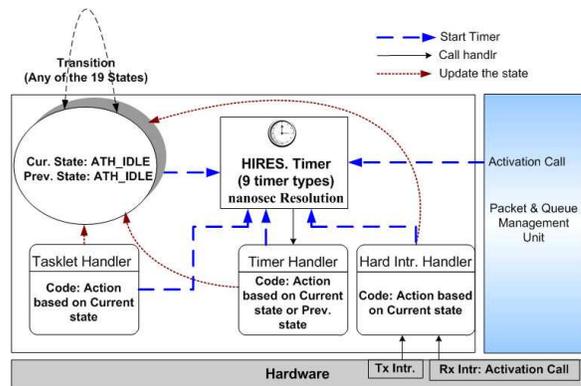
nels like (36, 40), (44, 48) etc., were interfering. Fig. 4(a) shows the throughput results for the two Tx-Rx pairs where the x axis is the channel pair they use respectively. Theoretically, the transmissions on the channels should not interfere, but in practice that is not the case and throughput loss is observed. When the Tx-Rx pairs used channels with a channel spacing in between, like (40, 48) in Fig. 4(a), they achieved full throughput without any loss. So in practice, consecutive orthogonal channels like (36,40) are not orthogonal, but channels with a channel spacing in between like (36,44) are orthogonal. The reason behind the interference could be attributed to the lack of a guard band in between two consecutive channels. To mitigate the interference, one solution could be to use low transmit power. By standardizing the transmission power for the 802.11 standard by taking into account the multi-channel use, we may achieve the number of orthogonal channels that are theoretically possible.

5.2 Close interface effect

A popular approach in multi-channel MAC design is to use multiple radios on a node. To understand the effects of multiple operational radios in a CPC, we conducted experiments with three CPCs as depicted in Fig.3. In the experiments, Node 1 transmitted UDP packets to Node 2 on Rx channel and Node 2 transmitted UDP packets to Node 3 on Tx channel simultaneously. We fixed Tx channel and varied Rx channel, and measured packet error rate (PER) of the Rx channel link. Experiment results are shown in Fig.4(b) for Tx channels 36 and 165 with the applied Iperf load of 10 Mbps. We found a huge PER at the receiver interface of Node 2 when we applied high load at the Tx interface of Node 2. This loss can be attributed to processing load or interference from the transmitter interface. In the former case, the transmit softirq in Linux has a higher priority over the receive softirq, so a high load at the transmitting interface might saturate a receiving interface. In the later case, probably the close placement of the radios or the interfaces was causing interference even for orthogonal channels. To identify the real cause of high Rx PER, we switched the Tx interface to 2.4 GHz band and repeated the experiment. This time, we found the Rx PER around 1% that shows that the cause of the high PER was channel interference. To further confirm this, we moved one radio away by using cable and repeated the same experiment in 5GHz band. This time we found 0 PER, so the clear culprit of the high Rx PER was interference caused due to the closeness of the interfaces, which we call the *close interface effect*. Also by comparing the PER for the two Tx channels in Fig.4(b), it is clear that when we use a high frequency Tx channel and low frequency Rx channel, interference is comparatively low than if we switch the roles.



(a) Protocol design: control & data session and packet formats.



(b) Software control unit design.

Figure 5: The protocol & software control unit.

One more observation from Fig.4(b) is that for Tx channel 36, we see a low PER (0.01%) for Rx channel 40 that implies no interference. As discussed in previous subsection, channel 36 and 40 are not completely orthogonal. So this triggers carrier sensing (CS) on one channel if transmission is happening on the other. Same is true for Tx channel 165 and Rx channel 161. In the previous experiment, no CS was triggered as the nodes were comparatively further apart.

For low Tx traffic loads (less than 4 Mbps), the PER was around zero. This could be because as the traffic load goes down, the chances of both the radios being operational at the same time goes down and this results in less interference. So the close interface effect comes into play for high Tx traffic loads and clearly reduces the number of orthogonal channels, which can severely affect the advantage of multi-channel MACs. So when designing a multi-channel MAC testbed based on commodity hardware, one can take care of the effect by using cables to put the antennas apart and can also use attenuators.

6. MAC PROTOCOL DEVELOPMENT

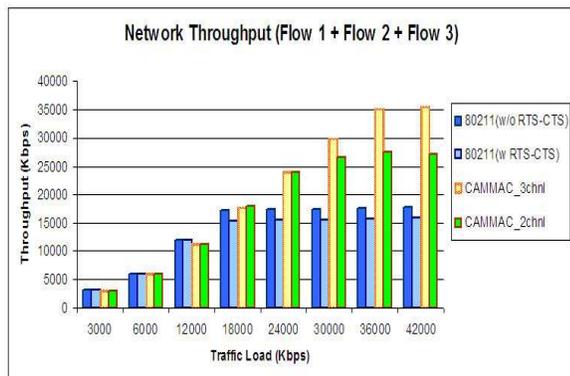
We have implemented a modified version of the *Cooperative Asynchronous Multichannel MAC* (CAMMAC) [9]. CAMMAC exploits neighboring nodes as another resource of control information to solve the Multi-Channel Coordination problem (MCC) [9], providing a single-radio and asynchronous solution. In CAMMAC neighboring nodes act as judges when a sender and a receiver is setting up communication. If an MCC problem is created for the pair, these judges will send messages to alert the pair to avoid the problem. For this, nodes maintain a spectrum usage table to store channels that are currently in use in the network. Each entry of the table consists of Tx & Rx MAC addresses, data channel and corresponding expiration time. The protocol uses a control channel shared by all the nodes to allow them to 1) negotiate data channels 2) alert a pair of nodes of MCC problem. Control channel can be used from any of the following: a) network has a licensed spectrum, b) an unlicensed band such as the ISM/UNII bands. On the control channel, nodes negotiate for a data channel by following a handshake called control session as shown in Fig. 5(a). In the control session, a sender and a receiver negotiate a data channel using an mRTS/mCTS exchange, which is followed by a cooperation phase where neighbors can alert the pair of any MCC problem. 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 the TxOp limit. The

receiver replies with a combined ACK (cACK) at the end of the TxOp.

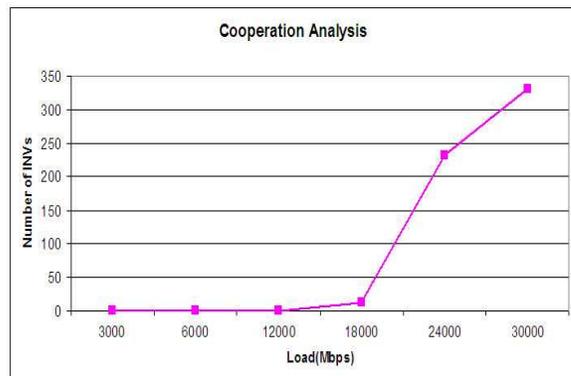
The modified design has optimized control channel handshake that increases the protocol robustness. Following is the implemented protocol design: To start the control session, a transmitter first sends an mRTS carrying an index of a data channel, on the control channel. If the receiver also deems the data channel to be free, it will reply with a mCTS which duplicates the channel. On the other hand, if the receiver finds that the data channel is in use by other nodes, it sends an INV packet. On receiving the INV, the transmitter will try again with another channel. As shown in Fig.5(a), Cocola1 and Cocola2 show the cooperation periods to avoid deaf terminal problem and channel conflict problem respectively. During the periods, 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. A *cocola* period is used to mitigate INV collision caused by multiple neighbors who identify the same MCC problem sending INV simultaneously. It is similar to CSMA-CA mechanism, where neighbors with INV to send, randomly choose sensing duration from $U[0, cocola]$ where $U[\cdot]$ denotes the uniform distribution. 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. If no INV is received, both nodes will switch to the data channel and transmitter senses the channel for duration Tcs. if the channel is free, it sends the first data packet as a probe and waits for ACK. Probe is used to avoid the case where only receiver receives an INV from its neighbor. Within an ACK timeout period (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. On receiving the ACK, the sender sends data packets to the receiver based on a particular duration called transmission opportunity (TxOp). The receiver replies with a combined ACK (cACK) at the end of the TxOp.

6.1 Implementation

To introduce our new protocol design as a *software MAC*, we modified the mac80211 stack, ath5k driver (2.6.30) and the card register values. Our main challenge was how to disable the 802.11 control unit on the card. We were able to disable control functions like random backoff, ACK packets etc., but could not disable car-



(a) 802.11 MAC vs. CAMMAC throughput performance.



(b) Number of INVs sent in two data channel experiment.

Figure 6: Performance and cooperation results.

rier sensing (CS). So we minimized the DIFS period as much as possible to minimize the CS time. We cannot make DIFS zero as it affects card operation and packet loss occurs. Another challenge was to realize the new control unit in the driver. The concept of the new MAC control unit is shown in the Fig. 5(b). The control unit consists of 19 control logic states called internal control states, 9 timers and associated handlers. The unit uses current and previous state flags, *high resolution timers*, tasklets and hard interrupts for running the control state machine in the driver.

We use packet train transmission to implement TxOps. Packet train based transmission required new queue management design in the driver. When using packet trains we need to differentiate between control packets from top layers like ICMP and data packets like UDP packets. If packet train design is used for all the packets, it will create problem in the proper functioning of the control packets. To avoid any complication to higher layer control protocols, we use both per packet and train based transmission. Only UDP packets are allowed for packet train transmission.

The current control unit does not implement few features of the design. INV for deaf terminal, cACK on the data channel (data is sent in burst mode) and the random transmission of INV (we assume only one neighbor will send INV, so no need to randomize) between 0 and cocola in the cocola periods are not currently included. Cocola duration (also DIFS) is of 120 microseconds, random backoff slot in the driver is of 30 microseconds and the mRTS-mCTS exchange takes about 200 microseconds.

6.2 Limitations and Benefits of the Approach

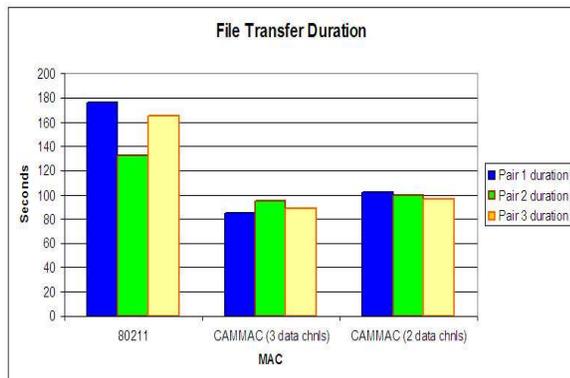
Usually, a MAC protocol can be divided into two sets of functions, one that are time critical and the other that are delay tolerant. The time critical functions are implemented in the form of hardware and the delay tolerant functions are usually present in software. For example, one time critical function of the IEEE 802.11 MAC is the DCF, which is present in the WiFi cards. Our *software MAC* approach involves implementing the whole MAC including the time critical part of the MAC in the software (kernel space). This limits tight time synchronization required for the time critical functions. Since Linux is not real-time OS, the presence of other kernel threads and user processes will add to the delay and the result is improper functioning and degraded performance. So the approach could be a draw back for MAC designs requiring strict time synchronization. The benefit of such an implementation is that it can be benchmarked with commercial hardware (e.g., Wi-Fi cards), making the idea more acceptable.

7. MULTI-CHANNEL MAC EVALUATION

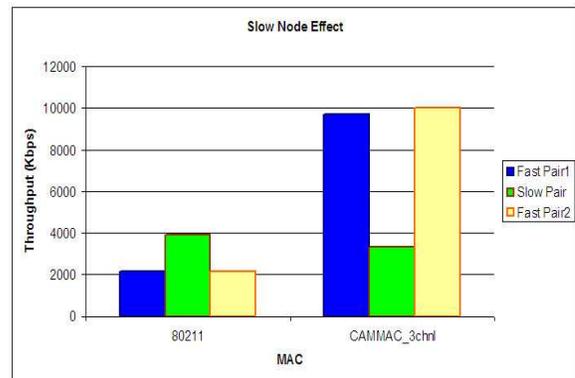
We evaluated the performance of the CAMMAC against the single channel 802.11 MAC to demonstrate the advantages of a multi-channel MAC. We conducted 3 types of experiment to evaluate throughput performance, delay performance and Slow node effect. The experimental topology as shown in Fig. 8 consisted of three Tx-Rx pairs that were chosen in the testbed to support 24 Mbps data rate and an idle node to alert in case of a channel conflict. We used Iperf to generate UDP traffic with a packet size of 1470 bytes. All the channels that we used in the experiments were in 5GHz band, unused and with a channel spacing in between as per the results discussed in the section 5. Packet train size of 20 (TxOp = 11 msec) was used for CAMMAC in all the experiments.

7.1 Throughput performance

We conducted four experiments of UDP traffic transfer to measure the throughput performance of both the MACs. Two experiments were based on the original 802.11 MAC (w RTS-CTS and w/o RTS-CTS) and the other two were based on the CAMMAC. In the CAMMAC experiment, one channel was used as the control channel and, three (CAMMAC_3chnl) and two (CAMMAC_2chnl) channels were used as data channels. The CAMMAC_3chnl experiment was free of channel conflict by a proper channel selection strategy of choosing the previously used channel first. In the CAMMAC_2chnl experiment, three pairs and only two data channels would create channel conflict, so we placed a dedicated cooperative node (idle node) to prevent it. We measured aggregated throughput (Flow 1+2+3) for various loads and show the result in Fig. 6(a). As shown, the performance of the CAMMAC slightly trails behind that of 802.11 when total traffic load (sum of applied iperf load at the 3 Tx) is low. This is because a single channel is sufficient for mild channel contention, so the control session in CAMMAC acts as an overhead and degrades the performance. As shown in Fig. 6(b), we found that there was no INV sent when the load was less than 12 Mbps that also shows less contention. After that, as the traffic load increases, channel contention becomes more prominent and a single channel is no longer sufficient for data communication. As such, the throughput of 802.11 (w/o RTS-CTS) saturates below 18 Mbps and 80211(RTS-CTS) around 15.5 Mbps. For the CAMMAC, performance is far better than 802.11 due to its multi-channel use enabled by the multi-channel access MAC. Although, the contention for the control channel in CAMMAC also increases, it is much lower than data channel contention because of the small control-session duration. We also streamed high definition (avg. rate 6Mbps) video over the 3 links. In case of 802.11 (w/o RTS-



(a) File transfer duration for the 3 Flows.



(b) Slow node effect 802.11 vs CAMMAC.

Figure 7: Delay performance and slow node effect.

CTS), two links can play the video smoothly, but in the case of 3 links streaming hangs. CAMMAC was able to stream video on the 3 links smoothly. Due to the limitation discussed in the section 6.2, we cannot realize the full potential of the CAMMAC. Therefore, we firmly believe that a prototype with CAMMAC control unit in the hardware can achieve far better performance.

7.2 File Transfer Duration & Slow node effect

In file transfer experiment, we use the same topology of the last experiment. We transfer a file using Iperf transfer rate of 10 Mbps and physical rate of 24 Mbps. The file transfer delays are shown in Fig. 7(a). Again CAMMAC outperforms 802.11 under heavy load due to its higher throughput performance. It can transfer double the traffic than 802.11 in the same time.

Slow node experiment compares the effect of the presence of a slow node in a mesh/ad-hoc network and how CAMMAC mitigates this effect by multi-channel use. A slow node is a node that can support very low data rates. We use the same topology as in the last experiments. One Tx-RX pair (slow node) is set to 6Mbps PHY rate and the other two at 24Mbps. Results are shown in Fig. 7(b), which clearly shows the advantage of a multi-channel MAC. A single slow node can adversely affect the performance of a 802.11 mesh/ad-hoc network, but it will require more slow nodes in case of CAMMAC to have the same effect due to the multi-channel use.

We would like to mention that in all the experiments 802.11 MAC had the same free channels available in 5Ghz band that CAMMAC used. But it could not use them because of its single channel architecture and so its throughput suffered significantly under heavy load and due to presence of a slow node. CAMMAC on the other hand allowed nodes to use the channels and achieved better performance under heavy load conditions. This advocates use of multi-channel access in mesh/adhoc networks to solve their throughput and scalability issues.

8. CONCLUSION

The paper presents the design of a mesh testbed for multi-channel MAC protocol development and shares the experience gained in the development of the testbed. It demonstrates how commodity hardware can be used to setup such a testbed. We present experimental studies on the number of orthogonal channels in the 5GHz band, the close interface effect in commodity hardware and implementation of our 802.11 multi-channel MAC. The first two studies identify the interference issues involved in the multi-channel MAC development using commodity hardware and propose solutions to mitigate the effects. The third experimental study bridges the gap between a

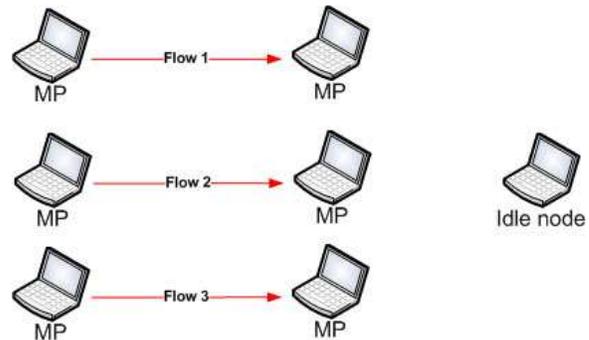


Figure 8: Experiment Topology.

multi-channel MAC design simulations and its performance in the real world. We carry out experiments on the testbed to evaluate the performance of our multi-channel MAC. By comparing to the 802.11 single channel MAC, we find that multi-channel MAC can indeed reap the benefit from using multiple channels and significantly outperforms 802.11 MAC. This is even despite the fact that in our experiments, 802.11 runs in hardware and the multi-channel MAC runs in software which is about a magnitude slower than hardware. Our future work includes large scale multi-hop experiments and comparing our MAC with other multi-channel MACs.

9. ACKNOWLEDGEMENT

We would like to thank Mr. Buddhika De Silva and Dr. Tie Luo for their suggestions on the testbed development. This work was partially supported by project grant NRF2008NRF-POC001-078 from the National Research Foundation, Singapore, and project grant NRF2007 IDM-IDM002-069 from the Interactive and Digital Media Project Office, Media Development Authority, Singapore.

10. REFERENCES

- [1] A. Adya, P. Bahl, J. Padhye, and A. Wolman. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *IEEE Broadnets*, 2004.
- [2] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM*, 2004.
- [3] 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*, 2004.

- [4] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *ACM MobiCom*, pages 31–42, New York, NY, USA, 2005. ACM.
- [5] J. Camp, J. Robinson, C. Steger, and E. Knightly. Measurement driven deployment of a two-tier urban mesh access network. In *ACM MobiSys*, Uppsala, Sweden, June 2006.
- [6] K. chan Lan, Z. Wang, R. Berriman, and et al. Implementation of a wireless mesh network testbed for traffic control. In *WiMAN*, 2007.
- [7] J. Chen, S. Sheu, and C. Yang. A new multichannel access protocol for IEEE 802.11 ad hoc wireless LANs. In *PIMRC*, 2003.
- [8] N. Jain, S. R. Das, and A. Nasipuri. A multichannel CSMA MAC protocol with receiver-based channel selection for multihop wireless networks. In *IEEE ICCCN*, 2001.
- [9] T. Luo, M. Motani, and V. Srinivasan. Cooperative asynchronous multichannel MAC: Design, analysis, and implementation. *IEEE Transactions on Mobile Computing*, 8(3):338–52, March 2009.
- [10] R. Maheshwari, H. Gupta, and S. R. Das. Multichannel MAC protocols for wireless networks. In *IEEE SECON*, 2006.
- [11] K. Ramachandran, M. M. Buddhikot, G. Chandranmenon, S. Miller, K. Almeroth, and E. Belding-Royer. On the design and implementation of infrastructure mesh networks. In *WiMesh*, 2005.
- [12] J. So and N. Vaidya. Multi-channel MAC for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. In *ACM MobiHoc*, 2004.
- [13] A. Tzamaloukas and J. Garcia-Luna-Aceves. Channel-hopping multiple access with packet trains for ad hoc networks. In *IEEE Device Multimedia Communications*, 2000.
- [14] S. Weber, V. Cahill, S. Clarke, and M. Haahr. Wireless ad hoc network for Dublin: A large-scale ad hoc network testbed. *ERCIM News*, 54, 2003.
- [15] J. Zhang, G. Zhou, C. Huang, S. H. Son, and J. A. Stankovic. TMMAC: an energy efficient multi-channel MAC protocol for ad hoc networks. In *IEEE ICC*, 2007.