Dynamic channel allocation in wireless ad-hoc networks

Shaan Mahbubani[#1], Anup Tapadia[*2], Liang Chen[#1]

[#1]*Computer Science and Engineering, University of California San Diego*

[#2]*Electrical and Computer Engineering, University of California San Diego*

[1]smahbuba@cs.ucsd.edu

[2]anup@ucsd.edu

[3]leoliangchen@ucsd.edu

***Abstract*—**

With the explosion of 802.11 network deployments, there exist co-located wireless networks in many densely populated areas (such as residential buildings, complexes, etc). Deployment of a mesh network in these locations would result in contention for the wireless spectrum from competing networks, leading to performance degradation.

The 802.11 standard specifies the use of several orthogonal channels which can be used to reduce interference in the spectrum. However, many wireless access points (APs) use channels that are statically allocated at the time of installation. When taking into consideration the fact that wireless traffic exhibits temporal and spatial locality, we contend that at different times/places, some of the channels are relatively underutilized.

We propose an ad-hoc network which contains monitor nodes at multiple locations. These monitor nodes observe and store data about the traffic in the vicinity. This data can be used, via our dynamic channel allocation algorithm, to determine the optimal channel to use at that place/time. We can therefore switch the channel used by a link in that vicinity to try and improve performance across the wireless link.

We observe several properties about the wireless channel, and correlate those with network performance. Using this correlation we propose a dynamic channel allocation scheme. We evaluate the effectiveness of this scheme for providing improved network throughput. In some cases this scheme can boost the performance up to 2 to 3 times the throughput of a statically allocated network.

I. INTRODUCTION

Wireless mesh networks are an attractive option for providing wireless connectivity in an enterprise or community setting due to relatively low deployment costs. Examples of such deployments and their benefits include Roofnet [10]. Such networks can be overlaid in areas where other wireless networks already exist. The recent explosion of 802.11 network usage has led the co-location of several wireless networks in an area to be a relatively common occurrence. For example, a community mesh network in the area of an apartment building may be co-located with an 802.11 network put up by each apartment resident.

However, the performance in these static mesh networks suffers due to interference from other devices using the wireless spectrum. The existence of co-located networks in the same physical location reduces the throughput of any network in that area. The sources of capacity degradation are several. One source occurs in an area in which multiple links on the same network overlap. In such an area contention for the channel on the same network causes transmission delays on at least one of the links. Another source of capacity degradation occurs from the wireless channel interference caused by transmission on co-located interfering networks.

To alleviate the contention on a given channel, as well as the interference caused by other wireless traffic on the channel, 802.11 provides for multiple orthogonal channels in a given spectrum. 801.11b allows 3 channels in the 2.4GHz spectrum, while 802.11a allows for 12 fully orthogonal channels in a 5GHz spectrum. At the time of deployment, the nodes in a static mesh network are configured to use a specific fixed channel. Unless specifically forced to, nodes will never change the channel they are transmitting on through the duration of their use. This may lead to physical locations where some channels are utilized to a greater extent than others.

In addition to the possibility of multiple nodes in a given area being configured to transmit on the same channel, the traffic in any given area at any given time on any channel exhibits temporal and spatial locality. Over long periods of time, the traffic will vary in both the number of users on each channel, and the amount of traffic they are generating. Thus, a mesh network that has the channels for each link statically allocated will at times face a greater level of interference and contention than at other times. However, a link that dynamically changes it's channel based on temporal and spatial traffic can attempt to achieve optimal performance by adapting to the current level of interference and contention on each channel. Therefore, to minimize the amount of interference and contention faced by the link between any two nodes at any given time, we can dynamically select which link each channel in the mesh network should be on.

The problem therefore becomes one of how to choose the best channel for any given link at any given time. Given that traffic exhibits temporal and spacial locality, we can analyze a short history of the traffic in an area, and use that to determine which channel to use for links in that area. Based on certain properties of the traffic on each channel at any given point in space/time, we want to predict which channel will offer us the best performance.

To know how to pick the best channel, we must first perform some experiments which correlate properties of the packets on the wireless channel. Hopefully there is some set of factors pertaining to the traffic on a channel which will able us to predict the performance of a transmission on that channel. This prediction can then be used to rank the channels available, and allow us to pick the best channel.

Once we have decided how to pick the best channel, we can then build an experiment that contrasts the performance achieved on a dynamically chosen channel against the performance achieved on a statically allocated channel. One important consideration that needs to be taken into account is the overhead and frequency of switching the channel. Our assumptions and goals can thus be summarized as follows.

Assumptions:

- Traffic characteristics on a channel vary over time.
- These characteristics exhibit some level of short term temporal and spatial locality.
- Some of these characteristics, given their variation and temporal/spatial locality, can be correlated with the obtainable performance on that channel.
- We can therefore measure these characteristics and rank a set of channels based on the performance we can achieve on each.

Goals:

- Determine which properties pertaining to the traffic on a channel are correlated with achievable performance on that channel.
- Implement a mechanism for choosing the channel of a link dynamically based on the appropriate channel characteristics.
- Evaluate the performance gains obtainable by deploying the dynamic channel allocation algorithm.

To achieve these goals, we propose, implement, and evaluate a wireless ad-hoc network with dynamically allocated channels. To sense and evaluate the current channel conditions, our network includes channel sensor nodes. The channel detection sensors monitor and record the traffic on each channel. Using this network, we can first perform some experiments to determine the best set of properties about each channel to be able to rank all available channels in terms of achievable performance. We can then build a channel allocation engine that will, according to the appropriate set of properties about the current channel utilization, choose and dynamically assign the best channel for the links in our wireless mesh network. We can then evaluate the performance gains offered by this allocation engine.

II. Related Work

The capacity of a wireless network is affected by the interference on the channel. The interference comes from contention on the channel from other wireless nodes, and the background noise in the environment. As noted in [1], one of the key factors restricting capacity in wireless networks is the need for all nodes in the domain to share the channel they are using with nodes in their local neighborhood.

Studies [2] have shown that links that use the same channel as neighboring links introduce interference, reducing total path throughput. This study also shows that using multiple radios can improve the end to end throughput in some cases. Routing metrics like ETX [5] have demonstrated that hop count is not the only metric which should be considered to build routing decisions. This leads to an exploration of whether dynamic channel selection can improve network throughput.

Jangeun Jun and et al. [6] investigated the capacity of a wireless mesh network on a single channel by studying the traffic collision domain, defined with respect to the geographical area of the network. Their results showed that the throughput of each node decrease as O(1/n), where n is the total number of nodes in the network.

Richard Draves and et al. [9] refined a routing metric in multi-radio multi-hop wireless mesh networks. The multi-radio multi-hop wireless mesh networks allow us to select the optimal path by taking into consideration the diverse channel and path length. The idea of their paper is to find a high-throughput path between a source and a destination by a path metric, which depends on the loss rate and bandwidth of paths. They proposed a link metric, Expected Transmission Time (ETT), to evaluate the loss rate and the bandwidth of the link.

The individual link weights are combined into a path metric called Weighted Cumulative ETT(WCETT), which explicitly accounts for the interference among links that use the same channel. The WCETT metric is incorporated into the source routing protocol, so that the optimized route can be selected, resulting in improved throughput.

Roofnet[10] focuses on the unplanned wireless mesh network. The unplanned mesh network is easy to deploy, but may suffer from performance degradation. The roofnet may have multiple gateways to provide external access, but may lack connectivity to an isolated node island. The paper shows that Roofnet works well with a route selection algorithm that that considers loss rate and bandwidth, such as ETT.

To improve the capacity of wireless mesh networks, some work has been done in the area of community based multi-radio multi-channel wireless mesh networks. The researchers focus on how to improve the throughput with channel and route selection. A. Raniwala and et al.[3] addresses two issues in a multi-channel wireless mesh network. These issues include how to assign a channel in the mesh network in order to avoid intra-network interference from each link, and how packets should be routed through a multi-channel wireless mesh network. To optimize the throughput of their mesh network, the paper adopted the channels according to its traffic load and network topology.

K.N. Ramachandran et al. [4] proposed the multi-radio mesh AP. Each mesh AP has one radio to capture packets on each supported channel for a small duration. One radio always stays on the default channel, and one radio adapts to the optimal channel for the performance improvement. The

channel allocation algorithm depends on two separate channel rankings: one is the ranking of interfering radios and the other is the channel utilization. A channel assignment server uses the information from the mesh APs to optimize the channel assignment as well as the routing configuration, thereby optimizes the throughput of the mesh network.

Compared to the work of [4], we combine the monitor radio of each mesh AP into the Cognet database provided by Cailt2, which provides detailed channel information of each mesh AP. Since we have more knowledge on the interference of each channel, we expect to achieve a better channel assignment algorithm to fully utilize the multi-radio mesh network capacity.

III. IMPLEMENTATION

A. *CogNet Testbed*

We decided to use the CogNet testbed at Calit2 for our experiments. This testbed consists of several monitoring nodes spread around the building. These nodes collect data on 802.11 b/g channels. The nodes sample each 802.11 b/g channel for about 1 second in a round robin scheme. This data is updated to a database which can then be queried.

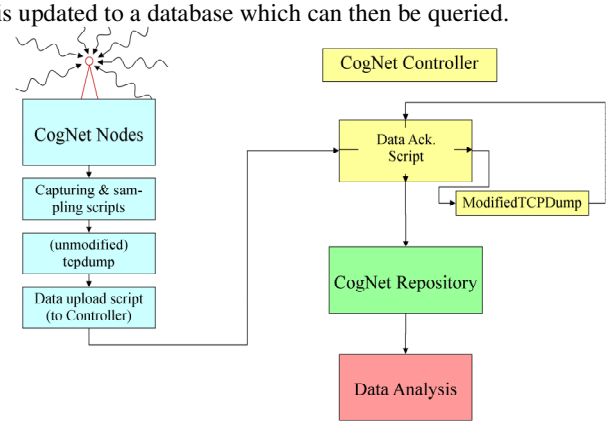

Figure 1: Flowchart of the operation of the CogNet Testbed. Components are grouped by color.

Figure 1 shows a high level flow of the way this system works. The CogNet nodes run a stripped down version of the Linux 2.6 kernel with madwifi driver on an AMD geode board. These nodes have an Atheros chipset wireless card which is configured to run in monitor mode. This enables the node to capture all the traffic from any node in the vicinity on the tuned channel. These nodes shift frequencies and sample all the 802.11 b/g channels under the direction of a shell script. Tcpdump captures all the packets into a file and this file is uploaded to the Cognet Controller every 1 minute via the wired network. The CogNet controller uses our modified version of tcpdump to parse these packets and insert data from the packet headers into the CogNet repository in a MySQL database.

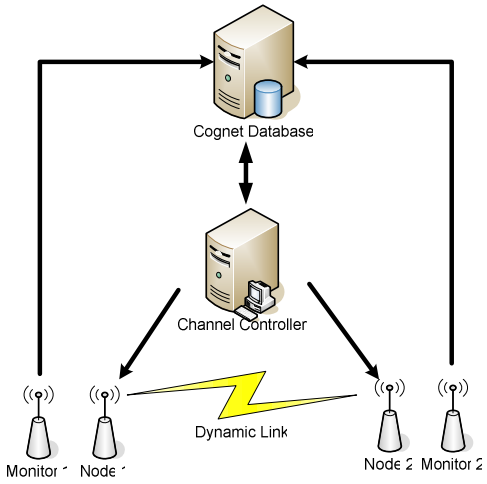

Figure 2: Ad-hoc network setup with CogNet

B. *Ad-Hoc setup*

For our experiments we decided to add two new CogNet nodes to the testbed which had the Atheros card configured in monitor mode. We added another Prism 2.5 802.11 b card for our throughput tests on this same node. This resulted in a single box which monitored as well as communicated over the wireless links. In order to avoid interference from our own traffic, the monitoring updates go over a wire instead of wireless. The Prism 2.5 cards were configured to run in ad-hoc mode with RTS-CTS switched off. Data ACK was turned on with a retry limit of 8. We changed the monitoring scripts to restrictively sample only channel numbers 1,6 and 11, for 1 second per channel. This meant that in a 60 second window, we get 20 seconds of sample data for each channel sampled; each channel was sample every 3rd second in this interval. These samples can be used to infer actual channel conditions and statistics. Figure 2 shows the ad-hoc setup we configured for all of our experiments. The channel controller is responsible for dynamic channel allocation based on statistical information queried from the CogNet database about the previous channel conditions.

C. *Channel Controller*

The channel controller has been implemented as a set of java classes that run on a linux machine. These software modules include a query engine, and a main driver that includes decision logic. Together, these modules can query the database in real time (if required), make a decision on which channel to assign to a particular link, initiate a change channel message to any node(s), and fire command line performance testing tools such as ping or iperf.

The query engine establishes a connection to the mysql database with a JDBC driver. The mysql database tables are indexed by epoch time to make executing queries within a specific time range very fast. This connection via JDBC is maintained throughout the duration of the experiment (as

opposed to setting up and tearing down the connection every time a query needs to be run) to allow fast query execution.

Queries are only executed for our dynamic channel allocation scheme (the static and random schemes do not need any information from the database). An NTP client is run on the channel controller machine to allow time synchronization with the database and the wireless nodes. A typical query involves getting the current time (with java's Calendar class), subtracting an offset (based on the length of time upon which we want to choose our channel), and getting the average of a field (or count) of the packets within that time window. Fields include properties of the packets such as RSSI, 802.11 rate, etc. The results of the query is passed immediately back to the main driver. The fact that the JDBC connection is constantly maintained, the database tables are indexed by epoch time, and our queries are essentially range queries upon the packets within specified epoch times results in our queries running very fast, typically on the order of milliseconds.

The main decision engine will periodically run the required query upon each table in the database. Typically each sniffer node will have it's own table in the database. In our testbed, each data node is co-located with a sniffer node. Therefore, to choose a channel for a particular link in our network, the decision engine will query the two tables corresponding to the sniffer nodes co-located with the data nodes that are connected by the link we want to change the channel for. Upon receiving the results of the queries for each channel, the decision engine will average the data received by both sniffer nodes. The channels can then be ranked based on offered opportunity, and the optimal channel chosen.

After the optimal channel is chosen for a specific link, the decision engine will call the command line tool (written in C) to actually change the channel of the nodes. The change channel command is invoked on both of the nodes at the same time to ensure that they change as simultaneously as possible. After the change channel command is invoked and returns with the success or failure of the change, the decision engine will start a ping to check if both nodes have connected to each other on the new channel and the link is up. The time taken after the channel is changed and before the ping returns is timed so we can measure how long the link is unavailable for. As soon as one ping returns, the decision engine will invoke whichever iperf test we want to run, depending on the experiment. After the iperf test finishes and the command returns, we sleep for a certain period in order to allow the network to recover to it's natural state.

The decision engine will loop over each scheme we want to test (e.g.static, dynamic, random), changing the channel for the link then executing iperf to measure performance. This loop will be executed for many iterations over a long period of time to measure performance for each scheme over varying channel conditions.

D. Channel switcher

Figure 3 demonstrates the implementation of our mesh network. When the channel controller decides the channel allocation of the links in the mesh network, it calls a channel switcher; the switcher communicates with each AP over the wired link to instruct the nodes when to switch which radio and to which channel. The channel switcher contains two modules: a console based program to receive a shell command from the channel controller, and a connection manager to maintain TCP connections with the mesh APs. The console program and the connection manager communicate via local IPC. Because they are decoupled, the channel controller can call multiple mesh APs at the same time, which reduces avoid the latency of synchronous channel switching. The established TCP connections are expected to provide reliable low latency message transactions with mesh APs.

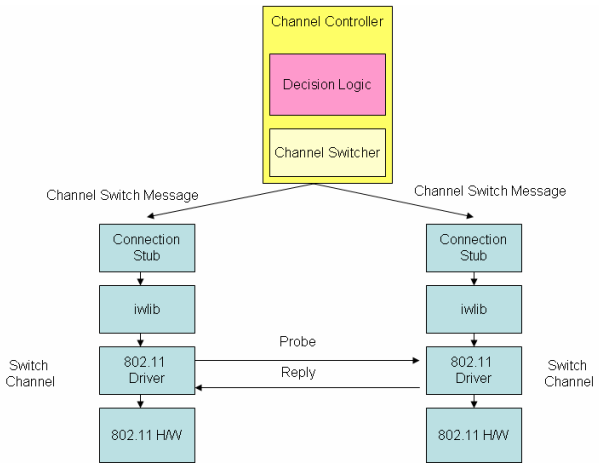

Figure 3: Channel Switcher

We implement a connection stub at each AP. The connection stub receives the channel switch command from channel switcher, and calls the iwlib library [7] to switch the channel at the given time. The iwlib will send IOCTL messages to the 802.11 driver, and instructs the driver to switch the channel, and the driver similarly instructs the 802.11 hardware.

When the channel controller switches the channel of a link, it sends control message to the two APs of the link. When the two APs switch their channels, they send probe packets and scan for the available 802.11 nodes within the same BSSID. When one AP hears the probe packets from an AP with same BSSID, it will reply to the AP and re-associate with the AP. After that, it will send ARP request packets to update its knowledge of the IP-MAC address binding of it's neighbors.

In our current implementation, the re-association takes considerable amount of time, about 2-12 sec. Because of this, the issue of synchronization at each AP is not a critical concern. To reduce the overhead of reassociation, we consider modifying the 802.11 MAC.

Because mesh networks have a static topology and routing among mesh APs, we can expect that the nodes connected by a link are always same after we switch the channel. Therefore, each mesh AP can use it's knowledge of it's neighbors before the channel switch. If the 802.11 MAC maintains this knowledge during the channel switch, we can remove the messages transaction for re-association.

Another concern in our implementation is the time synchronization among mesh APs. If the two APs can not switch their channel at the same time, the link will break until it is re-established. Our current implementation uses NTP to synchronize the clock among the mesh controller and all mesh APs. As reported in Jigsaw[8], NTP is not good enough to provide fine grained time synchronization; we can consider implementing a refined synchronization mechanism in future work.

IV. Algorithm Design

Figure 4 shows the hourly packet count over various channels received during a regular working day. Channel 6 seems to be the most utilized channel throughout the day. It can also be observed that the packet count of channel 1 and 11 are almost the same until 12:00pm. After 12:00pm, the utilization on channel 1 increases suddenly to 80000 packets, which is more than the packets received on channel 6. Channel 6 (the busiest channel until 12:00pm) seems to carry fewer packets during this time. Channel 11 still remains the least utilized channel until midnight. If these channels were utilized optimally, each channel would have carried an equal amount of traffic. Unfortunately, it is very hard to rely on an optimal static configuration, because the amount of traffic on each channel varies with time and space. This motivated an investigation into which factors are responsible for effective throughput on 802.11 networks.

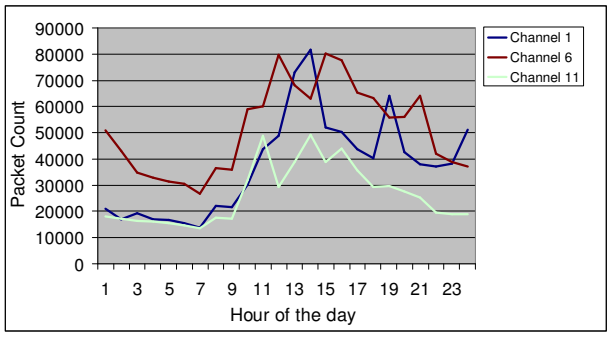

Figure 4: Traffic on Thursday 26th April 2007 at Calit2 6th floor

There are several factors which affect throughput on an 802.11 network. There has been previous work [11] which shows that throughput is also affected by interference from non 802.11 devices such as microwaves, Bluetooth devices and cordless phones. These interfering devices have a significant impact on 802.11 traffic performance, and can cause broken links or extremely low throughput.

There are other factors such as observed packet count, average RSSI of packets, and 802.11 rate which also have a considerable effect on the throughput. We decided to investigate these factors in our study. We do not consider external interference factors, as they are non deterministic and very difficult to detect.

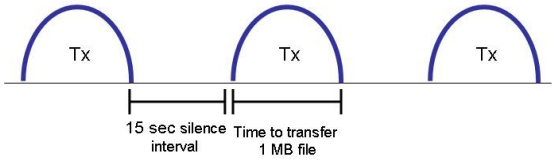

Figure 5: Parameter evaluation test timing

We did some initial tests to try and determine how different surrounding factors affect our traffic. The goal was to study which parameters should be considered when selecting the best channel to use. In this experiment we transfer a 1MB file from one wireless node to another using TCP. The nodes were locked on channel 6 during this experiment. The test was run several times, and each test result is recorded independently. Figure 5 shows the test timing diagram. The transmitting nodes are silent for 15 seconds before every transfer. We refer to this interval as the silence interval, and the packets captured during this interval as silence packets. In these tests, we correlate factors about packets observed during this interval with the time taken to transfer the file. We make the assumption that the channel conditions observed before the transfer starts continue through the duration of the transmission; this leads us to correlate the channel conditions before the transfer with the effect of those conditions of the transfer. To study the effect of various traffic loads on transmission performance, we programmed a random traffic generator which randomly bursts between 1kbps − 2000kbps for a random duration of time, ranging between 1−5 minutes.

Figure 6.1 shows how the transfer time for 1 MB file varies with respect to the packet count of the packets observed during the silence period. It can be observed that the lower the number of packets captured on the channel during the silence interval, the shorter the time required to transfer the file. With around 50 packets observed during the silence interval, the transfer time was around 2 seconds. The greater the number of packets observed during the silence interval, the greater the time required to transfer the file. Figure 6.2 compares the time required to transfer the file against the other packets observed in the air while we are transmitting. Like Figure 6.1, we observe that as the count of other packets increases, so does the transfer time. This shows that our transfer speeds are affected by the observed number of packets of the surrounding traffic. The variation in the packet count is on the order of 100%-400%. This large variation means that the packet count seems to be a sound predictor of how busy is the channel, and how our throughput may be affected if we use this channel. Therefore, we can infer that our throughput may be roughly inversely proportional to the packet count.

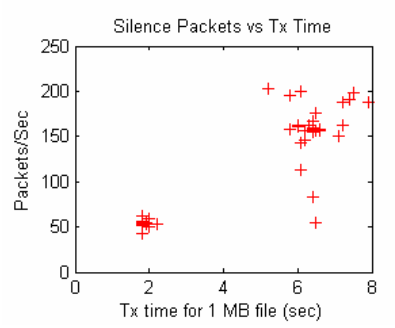

Figure 6.1

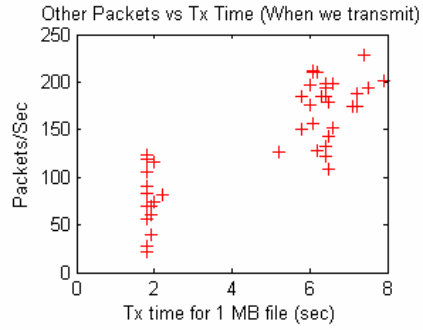

Figure 6.2

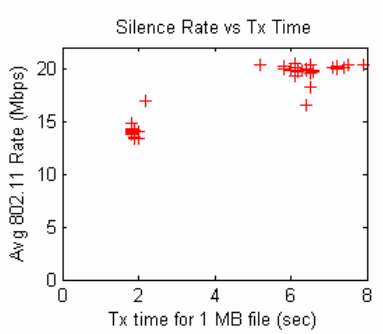

Figure 7.1

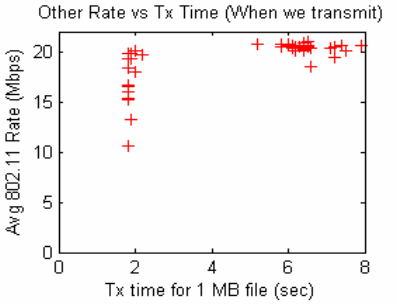

Figure 7.2

Figure 7.1 shows the average 802.11 rate of the observed packets during silence period. This rate is extracted from the Prism headers of the packets observed at the monitoring node. It can be observed that the average 802.11 rate of other packets during the silence period was slightly higher when the file took longer to transfer. However, the variation in rate is not very large, roughly only 25%. This would indicate that the observed 802.11 rate is not a strong predictor for the throughput on that channel. Even though the observed 802.11 rate could possibly be used as a basis for predicting the file transfer time, the low variation in rate leads us to believe that other factors with more variation would provide more accurate predictions. Figure 7.2, which compares transfer time against the observed 802.11 rate of other packets when we are transmitting also shows a similar trend where the variation is about 25%.

Figure 8.1 shows the average RSSI of the other packets in the air during the silence period. It can be observed that the average RSSI of other packets remains almost constant, between 40-50db, while our transfer time varies between 2 to 8 seconds. Figure 8.2 shows the average RSSI of other packets when our traffic is present. We can observe that the average RSSI of other packets while we transmit is higher compared to the silence period, when our transmit time was shorter. We hypothesize that our packets affected the other packets with lower RSSI, and therefore the average RSSI was higher in comparison to observed RSSI of other packets during the silence interval.

We did some further tests by varying the rate of our packets to see how our traffic affects the RSSI and packet count of the traffic around us. In these tests, one node sends UDP packets to another node in the presence of other interfering background traffic. Between every attempt, the node stays silent without transmitting any information for 15 seconds. This interval is known as the silence interval and the packets received during this interval are called silence packets. Figure 9 compares the average RSSI of packets during observed during the silence period, and that of the other packets during the time we are transmitting. It can be observed that the average RSSI remains almost constant in both these cases until we reach 3000kbps. After 3000kbps, the average RSSI of the other packets increases as our rate increases.

The reason for this may be as previously noted; our packets affect the packets with lower RSSI, causing them to drop, resulting in and increased average observed RSSI. Furthermore, the average RSSI variations are affected by many other factors such as multi-path, free space attenuation and interference from moving people. The variation in RSSI, like observed 802.11 rate, is low: about 10%-20%. The RSSI therefore does not seem to a very strong predictor for attainable throughput on the channel. From this study we can conclude that our packets mildly affect the RSSI of other packets received by the monitoring node, but the RSSI of received packets in not a good metric to use when making a channel selection (due to low variation).

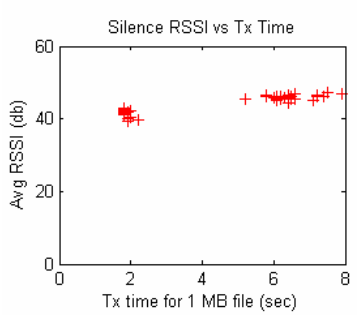

Figure 8.1

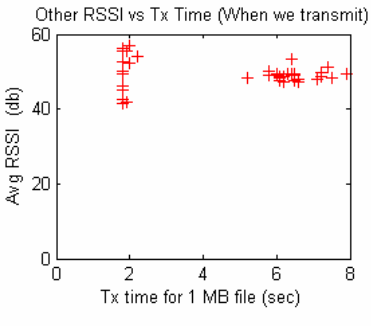

Figure 8.2

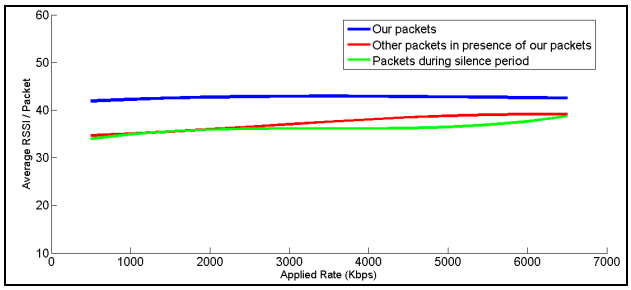

Figure 9. Effect of our transmission rate on others RSSI

These experiments made us believe that the observed RSSI and 802.11 rate of background traffic packets are not strong predictors for the achievable throughput. We therefore decided to compute a channel rank metric based on observed count of the packets observed from the surrounding traffic.

$$C_i = \frac{N_{ip} + N_{iq}}{2}$$

$$C_j = \min_{i=1,6,11} (C_i)$$

C_i is the rank of channel i. This rank is computed over a timing window of t seconds, counting the number of observed packets on that channel. N_{ip} and N_{iq} are the observed number of packets on channel i on nodes p and q respectively in the specified timing window.

We average these packets to compute the channel metric. C_j is the best channel based on rank of the channel metric. The lower the rank, the better the channel. Therefore, we select the channel with the minimum channel metric when choosing a channel with our dynamic channel allocation scheme.

V. Experiments and Results

A. Experiment

Having chosen to use the packet count per channel as our metric for choosing the best channel in our dynamic channel allocation scheme, we can make the two closely related claims. Firstly, we claim that the channel with the greatest number of packets on it will offer the worst performance. Conversely, the channel with the lowest number of packets will offer the best performance. We further propose that since the traffic on a channel exhibits temporal and spatial locality, we can sniff the packets on each channel for a short period of time, and pick the best channel with the assumption that the traffic characteristics of the channel will be maintained for the duration of our tests. To validate these claims, we devised an experiment to test the effectiveness of this scheme. We use a static channel assignment as the base case, where we fix the channel of the link to channel 6. We compare the performance of file transfers over the link using our dynamic scheme against the static assignment. We also include a comparison with a randomly selected channel, to test whether it is truly the optimality of the channel we have chosen rather than just a different channel from the statically allocated default that provides us with performance gains.

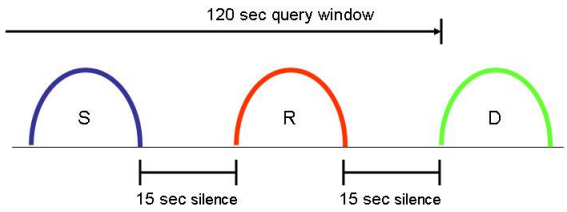

Figure 10: Experiment timing diagram

To pick our dynamically allocated channel, we queried the database over a 120 second window. We used 120 seconds because the sniffer nodes updated the database every minute, and therefore 120 seconds would provide the minimum time for both nodes in a link to update the database and provide as recent information as possible to our decision engine. We count the number of packets on each channel during the 120 second window at each node, and choose the channel with the lowest average number of packets. After changing the channel, we send a 3MB file across the link using iperf and measure the time taken to transfer the file. After the file transfers, we pause for 15 seconds to allow the network to revert back to it's original state. We then change the link to channel 6 and send the file over the link again, giving us the performance for the static scheme. We finish the test by pausing for 15 seconds, changing the link to a randomly chosen channel, and once again sending the file. The transfer times of these three iperf bursts are considered as one test. We repeat this test a number

of times over a long period of time to measure the performance of each scheme in the face of varying traffic patterns. Figure 10 summarizes the timing in this experiment.

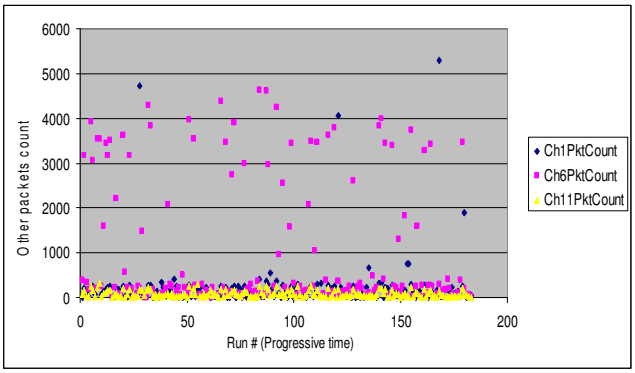

Figure 11: Variation in background packets during experiment

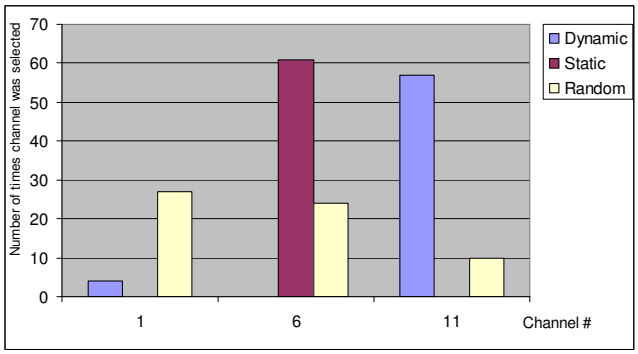

Figure 12: Selected channel histogram

Figure 11 shows how the number of packets on each channel varied over the course of the experiment. As shown in the graph, channel 6 is consistently highly utilized. According to our theory, this would have a negative impact upon the static scheme, which always set the channel to 6. The implications of this will be further discussed in the Analysis section. There are a few instances where channel 1 is more highly utilized, while channel 11 has consistently less packets than the other two channels. As we will see, this means that our dynamic scheme picked channel 11 most often.

Figure 12 shows how often each scheme picked each channel. The static scheme (by definition) always picked channel 6. As expected from observing Figure 11, the dynamic scheme picked channel 11 most frequently. A few times it picked channel 1, however channel 6 was never chosen. This conveniently provides a large amount of contrast to be drawn between the performance of the dynamic and static schemes since they always picked different channels. One further observation that can be made is that the random scheme picked all channels, however with a non uniform distribution. This may be due to several factors. The tests may not have been numerous enough for a more uniform distribution to occur. We had to exclude some results where the channel switching time was very long (this will be explained in more detail in the Analysis); it may have been the case that the results we excluded happen to be more frequent when the chosen channel was 11. Another possibility is that Java's Random function may not have been behaving truly randomly.

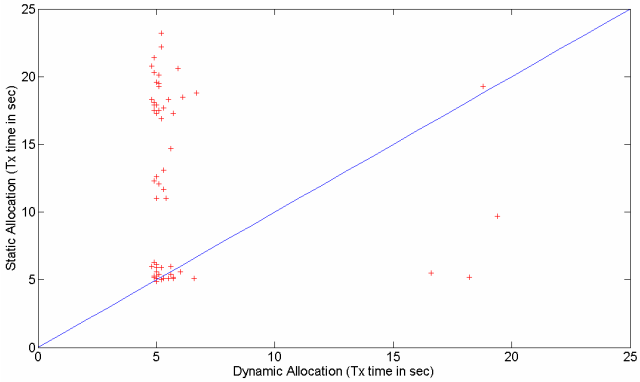

Figure 13: Dynamic allocation vs Static allocation performance

B. *Results*

Figure 13 compares the transfer times of the dynamic and static schemes in each test. The blue line is where transfer times are the same. Points clustered on the bottom left of the graph around the blue line indicate that for many of the tests, the transfer times using either scheme was low, around 5 seconds each. These points indicate tests where both channel 6 and the channel chosen by the dynamic channel provided equally good performance. The vast majority of the points are clustered near a time of 5 seconds with the dynamic allocation, indicating that most of the time the dynamically chosen channel allowed good performance. However, those same points are spread out over a range of 5 to 23 seconds with respect to the static allocation axis. This indicates that the performance achieved with the static scheme was sometimes as good as with the dynamic scheme, but was often much worse. There are a few points where the transfer took between 15 and 20 seconds with our dynamic allocation scheme, as well as a single point where both schemes performed poorly. Possible explanations for these results can be found in the Analysis section.

From these observations, we conclude that the dynamically chosen channel consistently offered good performance, whereas the performance on the statically chosen channel varied considerably. Our dynamic scheme is able to adapt to changing channel conditions and switch to a better channel.

Figure 14 contrasts the transfer times between the dynamic and random channel allocation schemes. The graph indicates by the color of the points the cases when the dynamic and random schemes picked the same channel, and when they picked different channels in a single test. This graph shares some characteristics with the graph contrasting the dynamic and static schemes. One similarity is that there is a cluster of several green points around the point (5,5). This cluster

indicates that while the two schemes often picked different channels, both picked channels that allowed good performance. Another similarity is that most of the points lie along the line where the transfer time with the dynamic scheme was around 5 seconds. The points are spread out over a range of the random scheme axis. This indicates that the channel chosen by the dynamic scheme consistently offered good performance. In contrast, the randomly chosen channel, while allowed good performance most of the time - the cluster of points around (5,5) also sometimes chose channels that offered poor performance. Also similar to the dynamic vs. static graph, there are a small number of points where performance was poor on the dynamic channel, but good on the random channel. These points are all cases where the random and dynamic channels were different. We defer possible explanations for these points to the Analysis section.

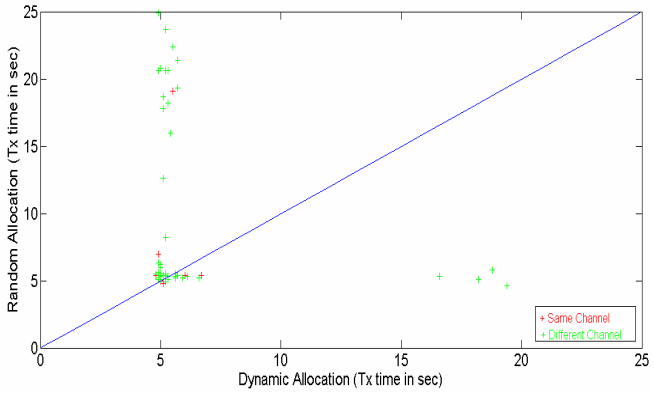

Figure 14: Dynamic allocation vs Random allocation performance

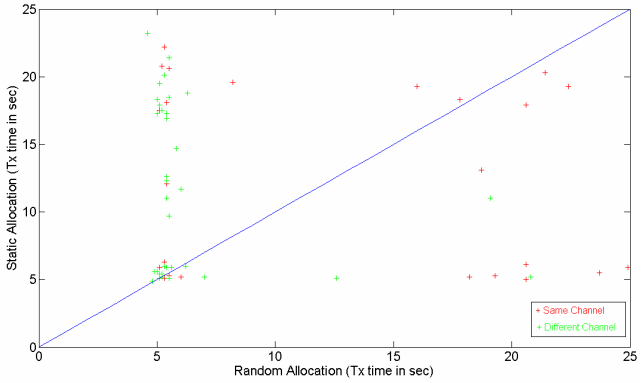

Figure 15: Random allocation vs static allocation performance

We include Figure 15 for completeness, comparing the transfer performance with a randomly chosen channel against that of the statically allocated channel. This graph has points scattered around 4 main regions. Once again, there is a large cluster of points around (5,5), indicating tests where both the randomly chosen channel and the static channel offered good performance. Most of these points are green, indicating the channel chosen by each scheme was different.

This would suggest that the static channel was not the only good channel, and some other channel during that test was also a good channel to use. There are more points near the 5 second mark on the random allocation axis than on the static allocation axis. This indicates that the static channel offered a large variation in performance, whereas the randomly chosen channel was often good.

Several red points lie in the upper right region of the graph. These suggest tests that occurred when the static channel was bad (since both schemes picked the same channel). One interesting observation is the small number of red points in the lower right and upper left corners of the graph. These are tests where both schemes picked the same channel, yet the transfer time took much longer when the channel is randomly chosen. These points will also be discussed further in the Analysis.

Scheme	**Dynamic**	**Static**	**Random**
Avg Rate	4042 kbps	2100 kbps	2742 kbps
Average Transfer Time (3MB)	6.08s	11.7s	8.96s

Table 1: Summary of allocation schemes

Table 1 summarizes our results for all of the tests. Transfers on the dynamically chosen channel achieved a much faster rate than the other two schemes, averaging nearly double the average rate of the static scheme. However, this does not take into account the overhead of changing the channel. Transfers on the randomly chosen channel also achieved a roughly 30% average rate over the static channel transfers. This indicates that changing the channel will avoid the bad performance suffered on a loaded static channel. However, as we will see, this may only be due to the fact that the static channel conditions were bad during the test (see Figure 11). The average transfer times produce identical comparisons, and are included for completeness (since they are simply the file size divided by the rate, and the file size was constant for these tests).

VI. Analysis

A. *Possible experiment criticisms:*

Our experiment made the following claims and assumptions. We claimed that the channel with the greatest number of packets will offer the worst performance. Conversely, the channel with the lowest number of packets will offer the best performance. Finally, the traffic patterns we observe during our sniffer period will continue through the duration of our file transfer.

As can be seen from Figure 11, channel 6 was the most heavily utilized during the time our tests were performed. Our static channel was also configured to use channel 6. According to our hypothesis, the channel with the highest number of packets is the channel where performance will be the most severely impacted. Our dynamic channel allocator never chose channel 6 during our tests, as can be seen from Figure 12. This provided a good contrast between transfers on

our dynamically chosen channel and the static channel. From the facts that our dynamic channel never chose the same channel as the static scheme, the static channel was heavily loaded, and transfers on our dynamically chosen channel were on average much faster than those on the static channel, we can infer that our hypothesis is valid. That is to say, we can infer that better performance can be achieved on a relatively unloaded channel as opposed to a channel with a very high number of packets being transmitted on it. This validates our first claim - that the channel with the greatest number of packets offers the worst performance. However, this may not strongly validate our claim that the channel with the lowest number of packets offers the best performance. For example, if our static channel had been configured to channel 11, the static performance may have been consistently good. In that experiment, our dynamic scheme would have chosen the same channel as the static scheme most of the time. If our dynamic channel occasionally chose a different channel and still offered better performance than the static channel, then we would have been able to more strongly infer that the channel with the lowest number of packets is the best channel. Put another way, since our static channel was relatively overloaded, it can be hypothesized that the static scheme performance was negatively impacted. Therefore, compared to this scheme, any other channel which was not so overloaded and on which performance was not negatively impacted would seem like a good channel.

B. Anomalous result explanations

There are several observations that can be made of our results which do not agree with our hypotheses. Here we propose possible explanations for these results, including possible violations of our assumptions and hypotheses. These anomalous observations are repeated here.

In our graph contrasting the dynamic and static schemes, there are several points on the lower right hand corner of the graph. These indicate tests where even though the dynamic scheme picked the channel with the lowest number of packets, the static scheme offered much better performance. There also exists a single point in the upper right region of the graph, indicating that both schemes chose channels on which performance was bad. Similarly, in the graph contrasting the dynamic and random schemes, there are several points where performance was good on the random channel, but bad on the dynamically chosen channel. Finally, in the graph contrasting the random and static schemes, there are red points both in the upper left and lower right regions of the graph that indicate occurrences where performance was bad with one of the schemes yet good on the other even though both schemes picked the same channel.

We stated in the Introduction the assumption that traffic exhibits a certain amount of temporal and spatial locality. In addition, we can measure the short term properties of a channel to predict performance on that channel in the near future. The traffic patterns may change between the time we sniff the channel and the time we transfer. The duration for which we sniff the channel may be too long or too short, i.e. the traffic locality may be changing faster or slower than our observation window. There may be a burst of traffic on a certain channel that adversely affects our transfer time, although that burst was not present during our query window. Any and all of these reasons are plausible explanations for the cases where performance on our dynamically chosen channel is worse than the random or static channels, or where tests where two schemes that picked the same channel have contradictory performance measurements.

Our hypothesis is based on the idea that we can use a certain property, namely the packet count on a channel, to predict the performance on a channel. However, as we can see from Figure 6.1, above a certain threshold of packets the performance achieved is sub optimal (around 100 in the upper graph). Therefore, it is possible that even though one channel has less packets than another, both channels may offer poor performance if both channels have enough packets on them to cross the threshold. This may explain results such as the point in the upper right region of Figure 13

$$t_d + t_c \le t_s$$
$$t_d \le t_s - t_c$$
$$t_d \le \frac{f}{r_s} - t_c$$
$$r_d \ge \frac{f}{\frac{f}{r_s} - t_c}$$

for $t_c = 5.7s$
$f = 3072kbits$
$r_s = 2100kbps$
$r_d \ge 4096kbps$

t_d = Tx time on dynamic channel
t_s = Tx time on static channel
t_c = Time to change
r_d = Rate on dynamic channel
r_s = Rate on static channel
f = Size of file

Figure 16: Time switching cost analysis

C. Switching time analysis:

Figure 16 shows what the rate on our dynamically chosen channel (the dynamic rate) must be in order for us to achieve a gain in switching channels. The dynamic rate depends on the amount of data left to transfer, the rate on the static channel, and the time taking to change channels. The smaller the file size, the faster our rate must be in order to overcome the transfer time lost in switching. In addition (and quite intuitively), the longer our switching time or the faster our static rate, the faster our dynamic rate must be. As shown by Figure 16, our switching time was 5.7s on average. With a file size of 3MB, our dynamic rate needed to have been faster than 4096kbps for us to achieve a gain from switching. However, as shown by Table 1, our average dynamic rate achieved was slightly slower than required, at 4042kbps. Observe that if our file time was any larger, the average achieved static rate any slower, or the average channel switching time any faster, then we would have achieved some gain in switching. This naturally leads to what further work can be done in this area.

D. Further experiments:

As shown in our switching time analysis, the minimum dynamic rate depends on the file size (or data left to transfer), the switching time, and the static (or current) rate. The most obvious optimization we can attempt to achieve is a reduction in switching time. In our experiment, there were a few tests where the linked failed to switch channels, or failed to converge on the new channel after switching for a long period of time (e.g. longer than the query window). We had to exclude these results because they would not make sense; for example, the measurement of transfer performance when the channel took longer than the query window to switch would mean that the 120 seconds prior to our transfer was completely disjoint from our query window and therefore not correlated to our observation of the channel conditions.

Given the time to more closely analyze the process involved in switching a channel, and optimizing firmware/driver/software components to minimize switching time would yield more accurate results, and place less demands on how fast the new channel must be. As can be inferred from the equation, as the switching time tends to 0, the dynamic time must simply be greater than the static time for us to achieve some gain in switching.

Our current dynamic predictor is very simple, taking into account simply the number of packets on each channel when making a decision. This can also be modified in several ways to optimize how we switch channels. The simplest modification would be to attempt to factor in other considerations when choosing a channel. As can be seen from the Figures 7.1, 7.2, 8.1 and 8.2, other factors such as RSSI and rate have some correlation with achieve rate. In this experiment, we chose to use packet count because it seemed to offer the most variation with regards to achieved rate; we considered packet count therefore to be the single most accurate factor out of all we had observed. If we attempt a more detailed analysis on how multiple factors (and their interaction) correlates with achieved performance, we can more accurately predict what our dynamic rate may be if we switch to a specific channel. In a further attempt at more accurately measuring how accurate our predictor is, we can build a cache of the values of various factors measured when taking our decision against achieved performance. This cache could be built up over a longer period of time in varying channel conditions in order to more concretely measure how accurate our predictions are. We can take all these optimizations further, and use them to actually predict not only what channel to switch to, but actually whether or not we should switch at all. Once we have achieved a more accurate prediction of exactly what our rate would be if we switched to a new channel, we can decide whether we would achieve a performance gain in switching, given our knowledge of what our average switching time is and the rate at which we are currently achieving.

In addition to these immediate optimizations which can be made to our current scheme, there are several additional experiments we can propose that would allow a more accurate prediction of which channel is optimal. For example, we can vary the size of the query window in order to determine how much locality is exhibited by traffic patterns at any given time/place. For example, would a 60 second or 300 second query window be better? We can dynamically change the weight given to multiple factors used in our decision based on observed traffic patterns at any given time. For example, if we observe that at a given time, the average RSSI on a specific channel is extremely high, even though the packet count is low, we can give more weight to RSSI when deciding channel optimality. We can use a more long term or moving average of factors, in conjunction with the average observed during the query window snapshot, in an attempt at following observed diurnal traffic patterns. All of these experiments, however, require significantly more involved consideration than we have described.

VII. CONCLUSIONS

We propose a dynamic channel allocation scheme that assigns to a link the channel on which the lowest number of packets is being transmitted. This scheme improves the throughput on an 802.11 ad-hoc network by a factor of up to 2 times the throughput achieved with a static channel allocation. Dynamic channel allocation provides an improvement only when the statically allocated channel suffers from a large amount of contention, and other channels are relatively underutilized. When the allocated static channel allocated is not overused, our dynamic allocation scheme would result in performance that is equivalent to that achieved with a static scheme. Taking into consideration the spatial and temporal variations in traffic over various channels throughout the day, our dynamic channel allocation strategy allows for a network to adapt to varying channel conditions; it can be adopted as an additional way to improve the throughput in mesh networks.

ACKNOWLEDGMENT

We wishes to thank Prof. Alex Snoeren for guiding us during this study. We also wish to thank Dr. B.S Manoj and Prof. Ramesh Rao for letting us use the CogNet infrastructure at Calit2 for our experiments. Parts of this study were funded by NSF SGER-CogNet grant.

REFERENCES

[1] Piyush Gupta and P. R. Kumar, The capacity of wireless networks, IEEE Transactions on Information Theory, Volume 46, Number 2. March 2000. Pages 388-404.

[2] Richard Draves, Jitendra Padhye, Brian Zill, Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. Proceedings of the International Conference on Mobile Computing and Networking, August 2004, Philadelphia, Pennsylvania. Pages 114-128.

[3] A. Raniwala, T. Chiueh, “Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network”, IEEE Infocom 2005.

[4] K.N. Ramachandran, E.M. Belding, K.C. Almeroth, M.M. Buddhikot, “Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks”, IEEE Infocom 2006.

[5] Douglas S.J. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, ACM Mobicom 2003.

[6] Jangeun Jun and Mihail L. Sichitiu, The Nominal Capacity of Wireless Mesh Networks, IEEE Wireless Communications, Oct 2003

[7] http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html

[8] Y.C. Cheng and et al., Jigsaw: Solving the Puzzle of Enterprise 802.11 Analysis, ACM SIGCOMM 2006.

[9] Richard Draves and et al., Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks, ACM Mobicom 2004.

[10] John Bicket and et al., Architecture and evaluation of an unplanned 802.11b mesh network, ACM Mobicom 2005.

[11] Yu-Chung Cheng and et al. Automating Cross-Layer Diagnosis of Enterprise Wireless Networks