Distributed Dynamic Channel Selection in Chaotic Wireless Networks

Matthias Ihmig and Peter Steenkiste

***Abstract*—Wireless networks in residential neighborhoods often show a suboptimal channel assignment which can result in unnecessary congestion and poor performance. This paper introduces a heuristic, decentralized algorithm for automatic channel assignment in 802.11 access points. Our algorithm optimizes the use of available bandwidth in unmanaged deployments without requiring manual configuration. Our evaluation shows that the algorithm always performs better than a single-channel static scenario and deviates less than 5% from a manually optimized channel layout.**

***Index Terms*—802.11 networks, self-managing networks, dynamic channel selection.**

I. Introduction

The use of wireless connections to access the Internet has been growing rapidly, fueled by the availability of inexpensive, 802.11-capable devices such as PDAs and laptops. This growth is dramatically changing the nature of 802.11 networks. Initial 802.11 networks were mainly campus or business deployments that were installed and managed by professional staff. In contrast, recent growth is mostly in residential neighborhoods and public areas (e.g. hotspots). Unlike campus networks, these new deployment tend to be small (one or a few access points) and are often installed by end-users. We will use the term *chaotic wireless networks* [1,2] for such deployments.

These chaotic deployments differ from traditional deployments in two ways [1]:

- *Unplanned*: APs are set up by individuals or independent organizations based on the location of network outlets and without taking existing wireless networks into account. Such deployment leads to highly variable AP densities.
- *Unmanaged*: Users are typically unfamiliar with wireless technology, so they struggle with the AP configuration process. As a result, people often use factory-set default values for key parameters such as SSID, channel selection, or security settings.

High AP densities combined with poor parameter settings can easily result in poor performance. As a result, we are exploring techniques to make chaotic wireless networks self-managing and self-optimizing, i.e. they adapt to both the physical environment and traffic load by automatically selecting appropriate configuration parameters.

This work was supported in part by the NSF through award CNS-0520192. Matthias Ihmig is with the Technische Universität München, Arcisstr. 21, Germany (e-mail: m.ihmig@mytum.de). The research was performed while he was a visiting student at Carnegie Mellon University. Peter Steenkiste is with Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213 USA, (e-mail: prs@cs.cmu.edu).

In this paper we address the problem of decentralized automatic channel selection. Earlier work has focused on optimizing channel assignment in centrally managed WLAN networks. In chaotic wireless networks, however, individual access points are typically owned by different people, so a centralized approach does not apply. In this paper, we introduce a per-AP channel selection algorithm that is sensitive to both interference from nearby APs and traffic load. We describe and compare different methods for measuring the runtime parameters used by the algorithm. We evaluate the performance of our approach for different scenarios and also evaluate how different forms of voluntary coordination across access points affect performance.

The remainder of this paper is organized as follows. In the next section we elaborate on the problem of automatic channel selection in chaotic wireless networks. We introduce our design in Section III and present an evaluation based on OPNET in Section IV. We discuss related work in Section V and summarize our results in Section VI.

II. Automatic channel selection in chaotic wireless networks

We elaborate on the characteristics of chaotic wireless deployment and define the channel selection problem.

A. Chaotic wireless network properties

[1] presents an analysis of a data set 28475 APs; the data was collected by the Intel Place Lab project in six US cities in June 2004 (http://www.placelab.org/). The analysis shows that more than half of the APs have three or more neighbors within transmission range. The maximum number of neighbors for one AP was 85 in Boston. Due to the still rising popularity of WLANs, we expect these densities to continue to increase.

We also conducted measurements in 73 locations in residential areas around Carnegie Mellon campus in Fall 2005. The histogram in Figure 1 shows that between 2 and 28 APs could be heard. AP density is clearly highly variable, and while the average density is well below the density on the Carnegie Mellon campus [8], this gap is likely to decrease over time.

Besides high, irregular access point density, chaotic wireless deployments also suffer from lack of proper AP configuration. For example, an analysis of channel distribution based on data from WifiMaps.com, which covers 302934 APs, shows that 41% of APs use channel 6, followed by 12% on channel 2 and 11% on channel 11. Many access points also use channels other than 1, 6 or 11. Our own measurements around CMU confirm this: we see a

similar distribution of APs across channels 1, 6, and 11, although we saw relatively few access points on other channels. In contrast, measurements on the CMU campus show a balanced distribution across channels 1, 6, and 11. This suggests that many people in residential areas do not understand the importance of configuring their AP.

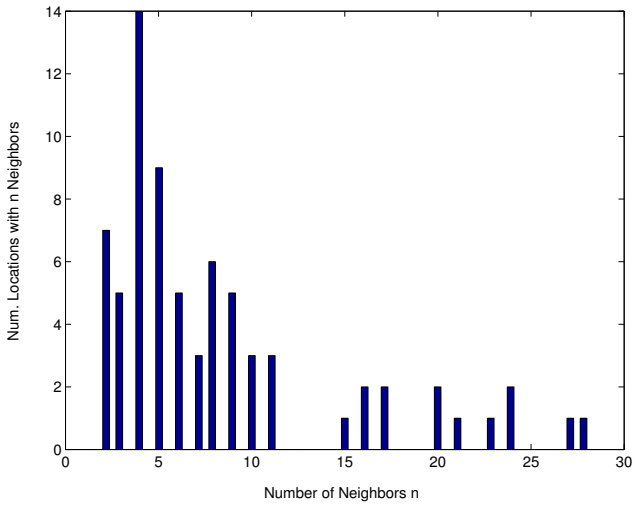

Figure 1. Number of APs at selected locations

B. Access Point Usage and Traffic Load

While the density of APs in residential neighborhoods has received a lot of attention, fewer studies have looked at the number of clients and the traffic profile. We captured packets in three locations, two in residential areas and one on campus, for a 24 hour period. Based on the captured packet headers, we looked at the number of APs and clients that could be heard at each location. Nodes that transmitted fewer than 100 packets were excluded from the analysis. Table I summarizes the results for the remaining nodes.

TABLE I: NUMBER OF CLIENTS AND APS

	Both in range			Hidden	
Area	APs	Clients	Cl/AP	AP	Client
Oakland	11	86	7.8	10	27
Shadyside	6	14	2.3	4	8
CMU	6	34	5.3	2	17

Columns 2 and 3 show the number of APs and clients for which we could hear both the AP and the client. We see that the Client/AP ratio (Column 4) differs substantially across the areas. It is quite high for Oakland (which has a lot of hot spots) but very low for Shadyside. Surprisingly, the ratio is quite low for the CMU location, probably because the measurements were collected in an office building; we would expect higher ratios, near, for example, a large auditorium. Column 5 (6) shows the number of hidden APs (clients), for which only the clients (AP) could be heard.

We also collected traffic profiles in the residential areas to better understand the traffic load. Figure 2 shows for example the load on an access point in Oakland. We see that the load is quite bursty with peaks around midnight and around the late morning. We saw a similar pattern in all locations, although the peaks varied greatly both in shape and time of day.

The observed traffic can be divided into three categories. First, we observed interactive traffic such as web browsing and mail, especially during the peaks. A second category consisted of wireless management frames like beacons, which add a load of about 1-5 kBytes/s to each channel. The remaining traffic consists of mainly peer-to-peer file sharing traffic, e.g. Bittorrent or Gnutella. This traffic accounts for most of the load, even though it is rate limited. Web, mail and other traffic is small with a daily average even below the WLAN management frames. Overall, the average load per AP is quite low, both because APs are used only intermittently and the Internet access links (mostly DSL or cable modem) limit throughput.

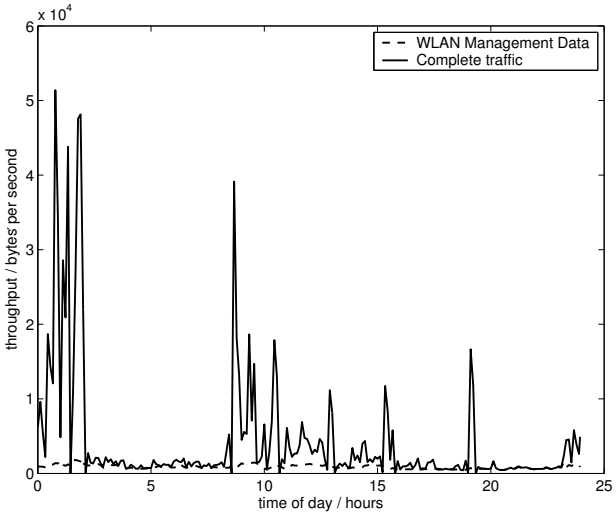

Figure 2. Traffic over 24h on a week-day

Our results are quite consistent with several previous studies that examined the characteristics of wireless network use in diverse environments [3,4,5]. Common observations are [6]:

- The concurrent and total number of clients associated with an access point varied widely over time [4,5];
- Data transfer rates in terms of throughput of users varies considerably [4];
- Usage characteristics vary widely across clients, resulting in little correlation between the traffic on APs and the number of associated users [3,4,5].

C. Automatic channel selection problem

In this paper, we look at the problem of automatic channel selection in chaotic wireless networks, such as those characterized above. Since we expect most home environments and hot spots (e.g. small coffee shops) to have a single AP, we will focus on single-AP service sets, i.e. there are no Extended Basic Service Sets (EBSS). This means that we must assume that APs will try to selfishly optimize their own performance, and solutions based on cross-AP optimization, e.g. based on a centralized server, are not appropriate. Note that our algorithm is likely to also pay off for the case of EBSS, since cooperation between the APs in an EBSS can only further improve performance.

We also assume that clients do not roam. Every client belongs to its specific BSS and does not associate with foreign access points. If the connection is lost, clients scan until they find their own AP with its unique BSS again. The increasing popularity of using WEP/WPA encryption enforces this behavior and supports this assumption.

Finally, we will focus on allocating the non-overlapping channels 1, 6, and 11. We will also only scan those three channels when assessing channel availability. This cuts back

significantly on the overhead of scanning, compared with scanning all 11 channels. This does not affect our algorithm: APs that use channels 2/3/4/5 and 7/8/9/10 will show up as interference.

III. Architecture and Algorithm

We now present a fully distributed algorithm for channel selection in chaotic wireless networks. We first describe our high-level design and we then evaluate different metrics for characterizing channel load. We also discuss a number of options of inter-AP and client-AP coordination.

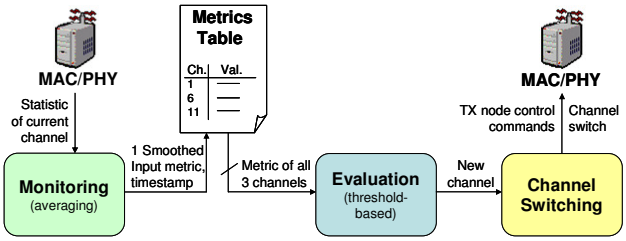

Figure 3. Function modules and data flow

A. Functional design

The functionality required for the dynamic channel switching algorithm can be split into three components (Figure 3). The *monitoring component* collects information on the load on channels 1, 6, and 11 and records this information in a load table. The *evaluation component* periodically checks this table to determine if the current channel is still a good choice or whether another channel should be used. The *channel switching component* switches channels if needed. We now elaborate on the operation of each component.

1) Monitoring module

The monitoring component gathers information about the channel load. While it can easily continuously measure the load on the current channel, several options exist for getting load information on other channels. One option is to rely on neighboring APs to announce the load they observe on their channel; however, since each AP is in a different physical location, the load observed by an AP may not be representative for its neighbors. An alternative is to have APs occasionally switch to other channels to measure performance. This is the approach we use, although we try to minimize channel switches to reduce disruption to clients, i.e. we only switch channels if there is reason to believe that this will improve performance, as is described in Section III.A.2. Moreover, APs always remain on a channel for at least t_{hold} seconds so they can get a reliable measurement of the traffic load.

When an AP switches to a new channel, it takes its associated clients with it, e.g. by announcing its channel switch (see Section III.B). This is important since it avoids interrupting active connections between the AP and its clients. Moreover, by moving the entire BSS to the new channel, we can measure the load on the new channel using regular traffic between the AP and its clients, i.e. we do not have to generate additional test traffic. The resulting measurements will also be more representative of normal traffic conditions.

Our algorithm is designed for off-the-shelf single-radio hardware. Multi-radio APs could further improve performance. For example, service sets with more than one client could distribute their clients over different channels. Alternatively, additional radios could be used to monitor the load on other channels, thus giving APs more up-to-date information on channels loads to guide channel selection.

We will evaluate different channel metrics in Section III.B, but independent of what metric is used, the channel load fluctuates significantly. In order to avoid unnecessary channel switches, we smooth the load using an exponentially weighted moving average [7], with $\alpha = f(\Delta t_{\text{sample}})$:

$$\bar{x}_k = \alpha \bar{x}_{k-1} + (1-\alpha) x_k$$

2) Evaluation module

The evaluation component regularly checks whether the current channel is still satisfactory, or whether another channel can potentially offer better performance. This is done by comparing the current channel load against a threshold. If it is crossed, a switch to another channel is scheduled.

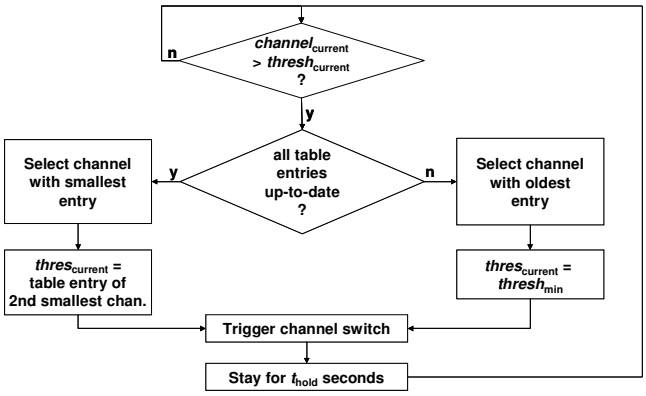

Figure 4. Flow chart algorithm for evaluation function

The flowchart in Figure 4 shows the selection algorithm. On startup, the current threshold $thresh_{\text{current}}$ is set to a minimum threshold $thres_{\text{min}}$, which will determine when the first switch takes place; the influence of $thres_{\text{min}}$ on the algorithm behavior is discussed in Section IV.E. The action, which is taken when the $thresh_{\text{current}}$ threshold is exceeded, depends on the age of the entries in the metric load table. We say that the load table is up-to-date, when the oldest entry is younger than two times the minimum time required to test all three channels for t_{hold} seconds each, which is $2 \cdot 3 \cdot 10s = 60s$ in most of our experiments. If the load table is up-to-date, the channel with the smallest metric value is selected as the new channel and the new value for $thres_{\text{current}}$ is set to the 2nd smallest channel metric value. This accounts for the now increased metric value and reduces the probability of another subsequent switch. However, if the table is *outdated*, we assume that load information of some channels is no longer valid. In that case, we initiate a switch to the channel for which we have the most outdated information so we can reassess its traffic load. We also reset $thresh_{\text{current}}$ to the minimum threshold $thres_{\text{min}}$.

3) Channel Switching module

The channel switching component receives switching requests and sets the transceiver hardware appropriately.

Channel switches in the access point are triggered by the Evaluation component, as described above. In clients, the simplest form of channel switching is based on the AP selection process that is part of the MAC layer. It is triggered automatically whenever the connection to the AP is lost. The delay associated with AP selection depends on several parameters, e.g. whether active or passive scanning is used, how many channels are scanned, and beacon interval. Typical overheads range from 2 to 10 seconds [8].

B. *Coordination across nodes*

In the architecture described above, we assume that there is no coordination among the wireless nodes. We now consider two forms of coordination, AP-client and AP-AP.

In the simplest case, APs simply switch channels when they detect a loaded channel. Unfortunately, this approach is disruptive to clients since they will lose connectivity and have to go through a time-consuming scanning and reassociation process; during that time, clients are disconnected from the network. A more attractive solution is to have the AP announce to its clients that it plans to switch channels, so they can reconnect more quickly, reducing disruption. This is supported in IEEE 802.11k through a management command that allows an AP to specify a new channel and a designated switching time to clients; this offers a standards-based method for efficient channel switching. OPNET, which we use in our evaluation, is based on the original IEEE 802.11 and does not support this feature. To support efficient switching in our simulations, we added a SWITCH command to the beacons. This reduces interruptions from up to several seconds to less than 1ms, similar to what can be achieved with 802.11k. If the beacon with a switch command gets lost, clients fall back to the basic expensive scanning mechanism.

Given our target environment we cannot assume cross-AP optimization of channel selection. However, we can consider some forms of voluntary coordination. APs can warn neighboring APs that they plan to switch channels. APs that hear this announcement may view this as a hint that they should delay switching channels, since the load on the channel they are currently using is likely to drop. This can potentially avoid oscillation in the case that multiple APs sharing a channel and seeing similar load conditions, decide on channel switches at the same time. In our OPNET implementation, the channel switch hint is implemented as a HOLD command that is included in the beacon. After receiving a HOLD command, APs stay on the same channel for at least one averaging window. A beacon can include both a SWITCH (for clients) and HOLD (for neighboring APs) command.

C. *Measuring the channel state*

The algorithm discussed in the previous assumes that there is a way of measuring the load on a channel. Earlier work on dynamic channel selection is often based on node layout: APs select a channel with minimal interference from nearby APs. However, this work targets chaotic wireless deployments, which, as described in Section II, have characteristics that differ substantially from those of campus-style deployments. Specifically, chaotic wireless networks have AP densities and traffic loads that are highly variable. This argues for channel selection algorithms that are sensitive to traffic load instead of node layout. There are many ways of measuring traffic load [8] – here, we focus on the three most promising ones: channel utilization, transmit queue length, and packet delay.

The most obvious metric to capture load is channel utilization, i.e. the percentage of time that the channel is busy. Channel utilization can for example be measured by the radio receiver on the AP. An alternative is to use the transmit queue length on the AP: one would expect the queue size to increase as the load increases. This metric has the advantage that it is very easy to measure on today's hardware. A final metric is the packet delay, such as MAC delay, the time between when a packet is handed to the MAC layer and when it is first transmitted; this delay captures both the transmit queuing time and the contention time for the medium.

To compare the performance of these three metrics, we ran a simulation (see Section IV for details) in which a set of clients executed an HTTP traffic load. We gradually increased the number of clients to increase the load in the network and measured the value of the different load metrics. Figure 5 and Figure 6 show the results for the MAC delay and utilization metrics with typical stochastic web traffic [10]; the results are 10s averages with their standard deviation boundaries.

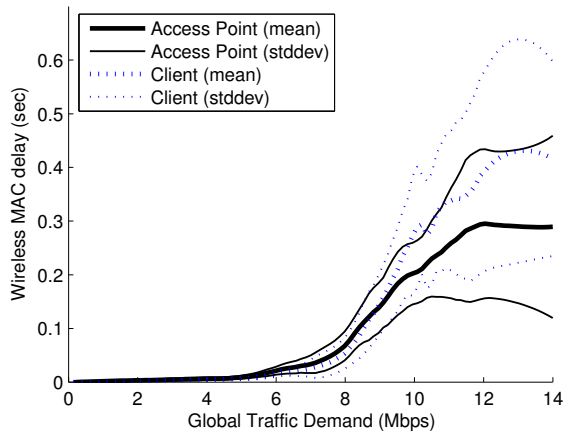

Figure 5. MAC delay versus traffic load

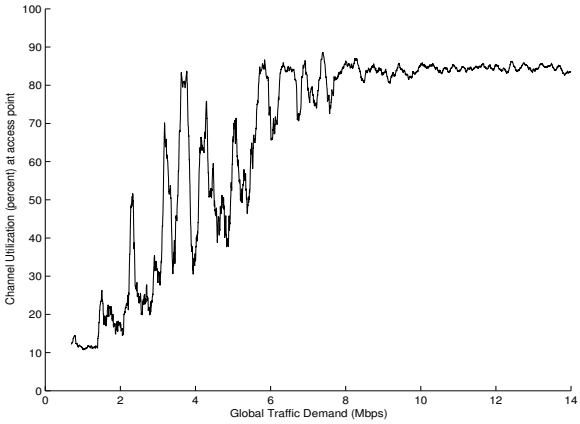

Figure 6. Channel Utilization versus traffic load

Our results show that MAC delay is the most attractive metric. In a non-congested state, it consistently has a low value without spikes that can cause false triggers. With higher loads, the MAC delay increases steadily, without

significant dips. In contrast, channel utilization is much more sensitive to the dynamics of the traffic load in the network: traffic burst on other APs can drive up the utilization and potentially trigger switches even if it has no or little impact on the local traffic. We also evaluated the transmit queue length as a load metric (see [8] for results) and found it to be unattractive: it is highly variable and can trigger channel switches even for a low traffic profile. Based on these results, we decided to use the MAC delay as our load metric.

We also found that the load information collected by APs is representative for their clients [8]. This should not be a surprise since APs and clients share the same transmission medium.

IV. Evaluation

We first describe our evaluation methodology. We then evaluate the performance of the basic algorithm for different traffic loads and assess the impact of the optimizations.

A. *OPNET simulation set up*

We implemented our dynamic channel selection algorithm in OPNET [11]. We also implemented the optimizations discussed in Section III.B by adding the SWITCH and HOLD commands to the beacons. For legacy clients, i.e. clients that do not recognize the SWITCH command, it takes on average about 7 seconds to reassociate with the AP after it has switched channels. This time includes detecting that connectivity is lost, scanning time using passive scanning, and reassociation delay.

Our evaluation scenarios model a residential area with a set of APs, each supporting a small number of clients (Section II.A). We collected results for different node densities, node dynamics, and traffic loads. We compare the performance of our algorithm with that of two static configurations: a single-channel setup in which all APs use the same channel and a manual setup in which APs are manually assigned to the 3 non-overlapping channels to maximize capacity.

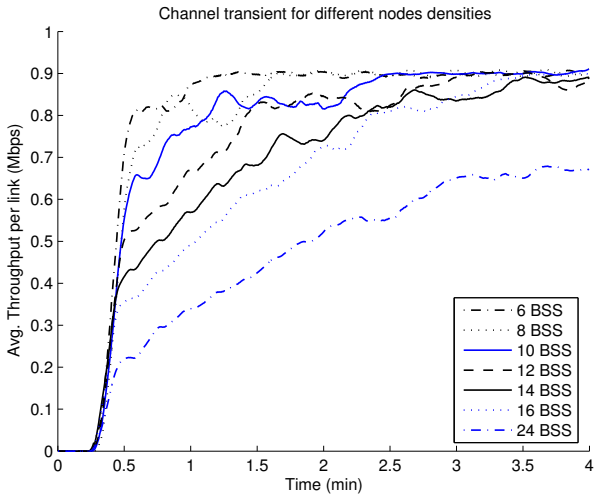

Figure 7. Average throughput as a function of increasing node density

B. *Algorithm Performance*

Figure 7 shows how dynamic channel selection using our dynamic algorithm improves performance in a dense scenario. Different numbers of nodes are placed in an 80m x 80m area, so since nodes have a transmission range of 115m, they form a full mesh. The traffic load is 1 Mbps of CBR download traffic per AP-client link and all nodes start to send traffic after 30 seconds on the same channel. The throughput has been averaged over a 10s window. We see that the system converges to a stable distribution, although the convergence time depends on the number of nodes. With 6 APs, it takes about one minute, while 16 APs take about 3 minutes. For 24 APs, the load is too high for the network capacity, so the requested throughput of 1 Mbps is never reached. The channel switching rate drops to zero once the system stabilizes.

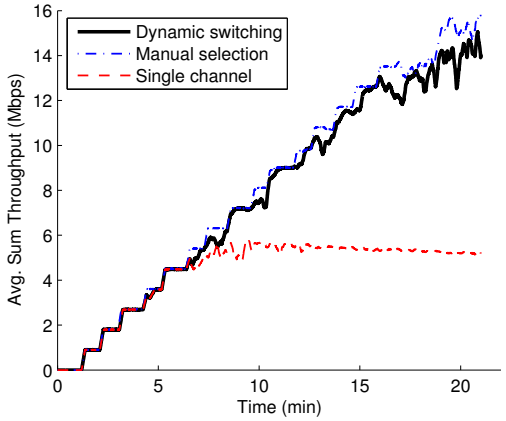

Figure 8. Throughput with increasing number of clients

Next we consider the dynamic case where clients are added incrementally: we have 8 APs in an area of 80m x 80m and every minute we add one client. Up to 20 clients are associated in a way that each AP serves between 1-4 clients. The results in Figure 8 show that in this dynamic case, the automatic switching algorithm has performance that is similar to that of manual assignment, although when more than 8 clients are in the system, there is a small delay associated with distributing the load across the non-overlapping channels.

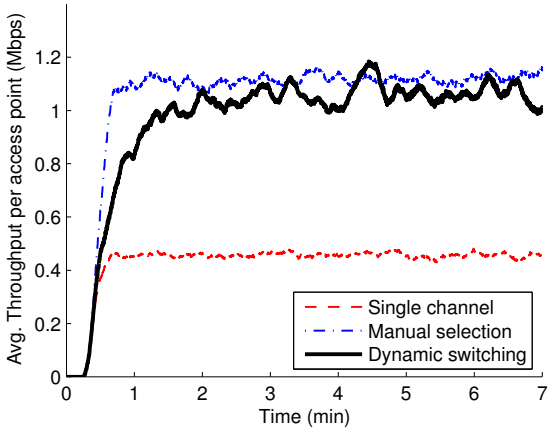

Figure 9. Averaged throughput for low-density scenario

Figure 9 shows the result for a low density scenario, where we distribute 20 APs and 40 clients in a 300m x 300m area. The nodes have a transmission range of 115m, so not all nodes can hear each other. We see that the overall throughput with dynamic channel selection is about 95% of the optimal throughput. Because APs only have a partial view of the environment, the channel switching rate does not drop to zero as in the dense scenario but converges to a rate

of 1.2 switches per AP per minute. It is possible to reduce this rate (e.g. by increasing t_{hold}) at the cost of reducing responsiveness.

We also ran a similar experiment with RTS/CTS enabled. The results are summarized in Table II: the RTS/CTS overhead reduces throughput by about 30%, but the overhead is the same for all three configurations.

TABLE II: EFFECT OF RTS/CTS ON LINK THROUGHPUT (IN MBPS)

Channel selection	Throughput no RTS/CTS	Throughput with RTS/CTS	degradation
Manual	0.56	0.397	70%
Dynamic	0.54	0.373	70%
Single	0.231	0.159	69%

When we used a randomized but static channel selection instead of the automatic switching algorithm, performance in the worst case dropped to 87% on average of the maximum throughput compared to an optimal channel selection. As expected, a random scheme can work well if the load on every AP is similar, but for variable load, dynamic channel switching reached better performance than random channel distribution.

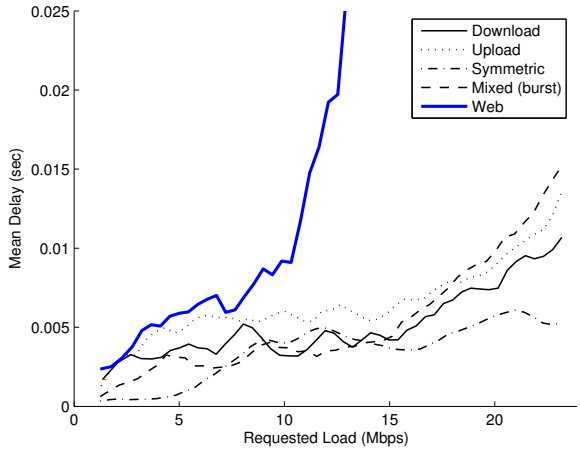

Figure 10. Average MAC delay for various traffic profiles and increasing load

C. *Different traffic loads*

We also ran simulations for diverse traffic loads such as download, upload, symmetric, mixed and web traffic. The download and upload profile transferred 100kByte objects every second, the symmetric-profile transferred two 1kByte objects every 16ms and the mixed profile contained a 1MByte download burst with symmetric traffic. The web profile was based on statistical descriptions from [10]. The impact on the MAC delay channel metric is shown in Figure 10. We see that the MAC delay metric works well: it generally increases with load and has relatively few peaks that could trigger unnecessary channel switches.

Figure 11 compares the performance of automatic, single-channel, and manual channel selection for download, file upload, mixed and web traffic in a dense node configuration with increasing numbers of nodes along the x-axis. We see the automatic selection works very well and even outperforms the manual selection for the web profile.

D. *Effects of coordination*

When an AP switches to a different channel, clients will be disconnected from the network until they reassociate with the AP. For legacy APs this takes on average about 5.4 seconds with passive scanning and between 90ms and 300ms with active scanning [9]. When we used the SWITCH command introduced in Section III.B, the switching time dropped to 5ms on average. We found that in our experiments, accelerated clients achieved slightly better performance than legacy clients: on average around 2% higher throughput and 7% lower delay. The differences are so small because under stable conditions, the number of channel switches is very low thus the performance loss is small.

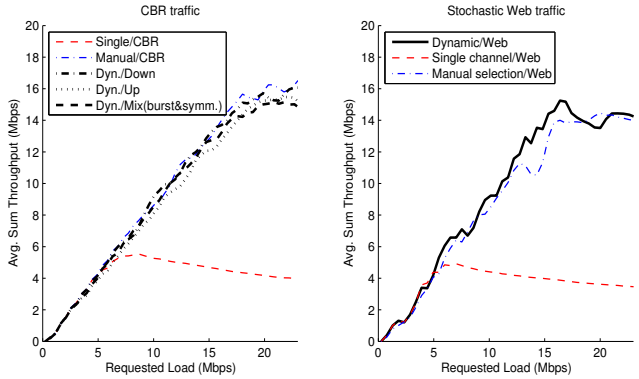

Figure 11. Link throughput for CBR and Web traffic with increasing load

We also evaluated the benefit of the HOLD command. We found only marginal improvements in performance (about 3% in global throughput) when the HOLD command was used with fast-switching clients. However, the HOLD command does cut back significantly on the number of channel switches. For example, in some high traffic load experiments, the number of channel switches was reduced by a factor of 2.8. This helps legacy clients that cannot process a SWITCH command to reduce disconnection time during scanning and reassociation. While the benefit will depend strongly on the channel switching rate, it can be as much as 50% at the maximum switching rate of one switch every t_{hold} seconds.

E. *Parameter sensitivity*

Finally, we evaluated the sensitivity of our results to the values of key parameters of the channel selection algorithm.

We looked at the impact of the averaging interval for the load metric. We average the load to avoid that small changes in load will trigger channel switches. We found that high averaging intervals (e.g. 60 seconds) result in smooth changes in the load metric but it significantly slows down convergence after a change in load conditions. A 10s interval generally works well, even for bursty web traffic. If clients support the SWITCH command, an averaging window as low as 1s works well since the cost of slow convergence is higher than the cost of extra channel switches. The hold time parameter t_{hold} has similar properties as the averaging window: higher values reduces the number of channel switches (which is important for legacy clients) but slows down convergence.

Another key parameter is the minimum threshold $thres_{min}$ which controls channel switching. For the MAC delay metric, we found that a value of less than 4.3ms, corresponding to a load of about 50% of the channel capacity, works well. Higher values increase convergence

times, as is illustrated in Figure 12: we show how long it takes for 10 APs and 10 clients to distribute across three channels, starting from channel 1. A lower minimum threshold results in unnecessary channel switches, which impacts performance for legacy clients.

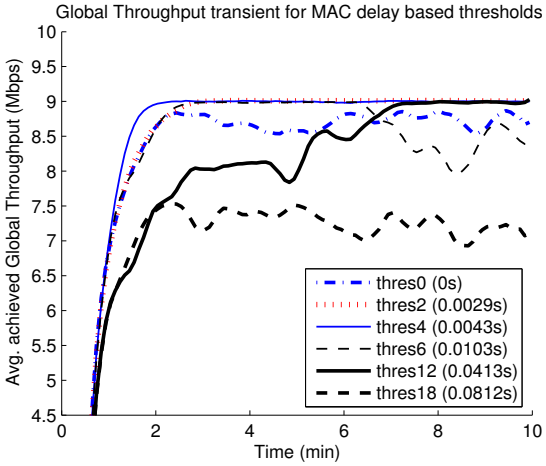

Figure 12. Throughput for varying MAC delay thresholds

V. Related Work

A number of vendors sell APs that do automatic channel selection, e.g. Cisco, AutoCell Laboratories Inc., and D-Link. While there is little known about the algorithms used in these products, reports suggest that they generally use number of adjacent APs [2] and signal strength information [12] for channel selection. In contrast, our algorithm is load sensitive.

Several researchers have looked at the problem of automatic channel selection for campus deployments. Examples include centralized and distributed approaches based on game theory [13], graph coloring [14], channel quality [15], and learning [16]. Similar work has been done in cellular networks, e.g. [17]. Researchers have also develop tools to help network managers with AP placement and channel selection [18,19]. These approaches typically minimize interference in campus deployments. In contrast, chaotic deployments have highly variable densities and traffic, so we use an approach that is not only sensitive to interference but also to load.

[20] uses channel hopping for channel selection in residential WLANs. The main difference is that our approach is load sensitive.

There has also been work on channel selection in mesh and ad hoc networks, e.g. [21]. In this context, channel selection is tied to multi-hop routing and a key challenge is to use a channel assignment that limits the effect of "self-interference" between the hops of a single path. This focus is very different from ours.

VI. Conclusion

In this paper, we explored a decentralized algorithm which measures current channel load and automatically switches to a less used channel to reduce resource congestion. We showed that using a delay based metric is suitable for our threshold-based algorithm. Using this approach, we can achieve at least 95% of the throughput compared to a hand-optimized scenario with equal load on each AP and even higher for highly varying usage profiles, such as in residential environments.

References

[1] Aditya Akella, Glenn Judd, Srinivasan Seshan, and Peter Steenkiste. Self-management in chaotic wireless deployments. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 185–199, 2005.

[2] Arunesh Mishra, Eric Rozner, Suman Banerjee, and William Arbaugh. Exploiting partially overlapping channels in wireless networks: Turning a peril into an advantage. In *IMC '05, 2005 Internet Measurement Conference*, pages 311–316, 2005.

[3] Anand Balachandran, Geoffrey M. Voelker, Paramvir Bahl, and P. Venkat Rangan. Characterizing user behavior and network performance in a public wireless lan. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 195–205, 2002. ACM Press.

[4] Magdalena Balazinska and Paul Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 303–316, 2003. ACM Press.

[5] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. In *MobiCom '02: 8th annual international conference on Mobile computing and networking*, pages 107–118, 2002.

[6] Godfrey Tan and John Guttag. Capacity Allocation in Wireless LANs. Number 973, Cambridge, MA, November 2004.

[7] Ming Tham, Dealing with Measurement Noise, http://lorien.ncl.ac.uk/ming/filter/filter.htm, 1998.

[8] Matthias Ihmig, Dynamic Channel Allocation for Self-Managing WLAN Access Points in Chaotic Wireless Networks, Diplomarbeit, Master's Thesis, Technische Universität München, Mar. 2006.

[9] Hector Velayos and Gunnar Karlsson. Techniques to reduce the IEEE 802.11b handoff time, In ICC 2004 – IEEE International Conference on Communications, vol. 27, no. 1, pages 3844-3848, 2004.

[10] Dirk Staehle, Karl Leibnitz and Phuoc Tran-Gia, Source traffic modeling of wireless applications. Technical Report 261, University of Würzburg, Jun 2000

[11] OPNET Technologies, Inc., http://www.opnet.com/

[12] AutoCell Laboratories Inc. Laboratories. March 2005, http://propagatenet.com/news/releases/ ac name 030105release.pdf.

[13] Magnus M. Halldorsson, Joseph Y. Halpern, Li (Erran) Li, and Vahab S. Mirrokni. On spectrum sharing games. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 107–114, 2004.

[14] Arunesh Mishra, Suman Banerjee, and William Arbaugh. Weighted coloring based channel assignment for WLANs. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(3):19–31, 2005.

[15] D.J. Leith and P. Clifford, A Self-managed Distributed Channel Selection Algorithm for WLANs, 4th Int. Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006.

[16] D.J. Leith and P. Clifford, Convergence of Distributed Learning Algorithms for Optimal Wireless Channel Allocation. Proc. IEEE Conf on Decision and Control, San Diego, 2006.

[17] L. Narayanan, Channel Assignment and Graph Multicoloring, Handbook of wireless networks and mobile computing, Wiley, 2002.

[18] T. Vanhatupa, M. Hannikainen, and T.D. Hamalainen, Frequency management tool for multi-cell WLAN performance optimization, 14th IEEE Workshop on Local and Metropolitan Area Networks, 2005.

[19] Alex Hills, Large-Scale Wireless LAN Design, IEEE Communications, 39(11):98-104, Nov 2001.

[20] Arunesh Mishra, Vivek Shrivastava, Dheeraj Agrawal, Suman Banerjee, Samrat Ganguly, Distributed Channel Management in Uncoordinated Wireless Environments, Mobicom 2006.

[21] Ashish Raniwala, Tzi-cker Chiueh "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network", in Proceedings of IEEE Infocom 2005.