SMARTA: A Self-Managing Architecture for Thin Access Points

N. Ahmed and S. Keshav
School of Computer Science, University of Waterloo
Waterloo, ON, Canada, N2L 3G1
{n3ahmed, keshav}@cs.uwaterloo.ca

ABSTRACT

Optimally choosing operating parameters for access points in an enterprise wireless LAN environment is a difficult and well-studied problem. Unlike past work, the SMARTA self-managing wireless LAN architecture *dynamically* adjusts *both* access point channel assignments and power levels in response to *measured changes* in the wireless environment to optimize *arbitrary* objective functions, while taking into account the *irregular* nature of RF propagation, and working with *unmodified* legacy clients. We evaluate the SMARTA architecture through simulation and show that our solution is not only feasible, but also provides significant improvements over existing approaches. For example, in a realistic scenario, SMARTA can provide 50% more throughput and 40% lower mean per-packet delay than a hand-optimized configuration. Moreover, SMARTA can *automatically* reconfigure channels and power levels in response to both small and large changes in the RF environment due to client movement.

Categories and Subject Descriptors

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

General Terms

Management, Measurement, Performance

Keywords

Access Points, RF Interference, Channel Assignment, Power Control

1. INTRODUCTION

Wireless LAN management is surprisingly more complex than managing a wired LAN [5], and the impact of fundamental tuning parameters, such as channel allocation and power level, is not well understood even by professionals. Moreover, as the size of the network increases, the management complexity multiplies. Challenges in managing wireless LANs include:

- *RF Interference:* Co-channel RF interference can cause throughput reduction factors of up to $4\times$ [2]. Moreover, the interference range of an AP may be much larger than its transmission range.
- *Irregular Coverage Areas:* Wireless signals do not experience uniform path loss, as is commonly assumed in most of the wireless research literature. Therefore, the Euclidean distance of a point from an AP does not determine whether that point is in the AP's coverage area.
- *Dynamic Coverage Areas:* Coverage areas may change over time and over short distances due to shadowing and multi-path transmission. For instance, Reference [26] shows that signal strength varies significantly over a range of $\frac{\lambda}{4}$, where λ is the radio wavelength. At 2.4Ghz, this is just $3.7cm$. This effect cannot be captured by simplistic exponential radio decay models.
- *Asymmetric Channel Conditions:* Channel state may differ in the client-to-AP and AP-to-client directions even on the same link.
- *Conflicting Objectives:* The network operator may want to simultaneously optimize multiple system objectives, some of which may conflict with each other.
- *Inability to Make Client Modifications:* Ideally, existing client devices should not need to be modified when introducing a new system.

Several researchers have addressed the problem of wireless LAN management in the past [10, 22, 27]. However, these solutions ignore one or more of these fundamental constraints, for example, assuming uniform path loss models, or ability to make end-client modifications, making them infeasible in practice.

Contributions: We present SMARTA, an architecture that takes the above-mentioned challenges into account. Our infrastructure-based solution, targeted towards enterprise wireless LANs, does not require client-side modifications, allowing backwards compatibility. Utility functions provide a unified framework for capturing multiple and even conflicting performance objectives. Moreover, SMARTA makes no assumptions about RF propagation and uses dynamic optimization to address varying channel conditions.

At a high level, SMARTA uses active probes to build a *conflict graph* to accurately model the RF environment without making path loss assumptions. Utility functions are defined on the conflict graph to characterize network performance. Finally, a variety of operating parameters are used to optimize the computed utility.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents an overview of the SMARTA architecture and Sections 4 and 5 discuss the models we use to characterize performance. Section 6 discusses techniques for measuring network performance and Section 7 presents algorithms for optimizing performance. We evaluate the features of the SMARTA architecture in Section 8 and end with a discussion and some conclusions in Sections 9 and 10, respectively.

2. RELATED WORK

Static vs. Dynamic Optimization: Techniques that solve management problems for wireless LANs fall into two broad categories: static optimization and dynamic optimization.

CoNext'06 December 4-7th, 2006, Lisbon, Portugal.

Static optimization involves optimally placing and configuring access points at deployment time [15]. Due to the spatio-temporal variations of the wireless channel, static optimization performs poorly, at best. Dynamic wireless LAN optimization techniques, therefore, are popular and commercially available today [2, 3, 4]. Such systems fall into one of two categories: decentralized fat-access-point architectures, or centralized thin-access-point architectures. Decentralized fat-access points use built-in intelligence to sense the wireless environment and either unilaterally decide the best configuration for themselves, or coordinate with each other to agree on a globally optimal configuration [2]. Centralized-thin-access point architectures use a centralized controller (or switch) to connect all access points. Access points merely sense the environment and send reports to the central controller which then decides the best configuration for them. Centralized approaches are better-suited for enterprises due to the already present centralized management infrastructure (e.g. centralized authentication, authorization, and accounting (AAA)). Meru [4] and Extricom [3] are examples of commercial systems that adopt this approach. However, all these management solutions are customized for proprietary hardware and use proprietary algorithms to achieve their ends, making them both hard to validate and hard to compare with other algorithms.

In contrast, academic prototypes such as ECHOS [27], Cell Breathing [9] and MiFi [10] have also been proposed with similar ideas in mind. Additionally, Kauffman et al [20] take a decentralized coordinated approach to network management. However, these systems require either client-side modifications or use unrealistic propagation models to characterize the wireless environment, making them impractical in a real-world deployment. Our work seeks to extend this body of work to deal with real-world constraints.

Detecting Interference: Interference detection has been well-studied in the literature [8, 14, 18]. However, most of these techniques infer interference using higher layer (e.g. NET/MAC layer) statistics that are impacted by multiple physical layer RF phenomena [25]. Therefore, the accuracy of these approaches in detecting interference is limited. In contrast, Qiu et al. [24] adopt a trace-driven simulation approach in which they collect traces from the real environment and replay them in the simulator. The simulator acts as a controlled environment in which accurate root-cause analysis can be done. Similar to the ideas in this paper, Padhye et al. [23] discuss an approach of running controlled pairwise experiments to detect and quantify RF interference. Their approach, however, requires artificially injecting flows into the system and can take a considerable amount of time to run, making it infeasible for use in realtime scenarios. In contrast, we show that it is possible to run simple and efficient tests on-the-fly in order to accurately detect RF interference.

Channel Assignment: Common techniques for optimizing wireless LAN performance are to perform parameter tuning at access points and clients. The most common of these tuning parameters is the AP's channel. AP channel assignment has been studied extensively in the literature [12, 20, 22] and is a well-known NP-hard problem. A number of heuristics have been proposed for this problem [12, 22]. For example, Mishra et al. [22] use a randomized search algorithm that incorporates client interference in the channel assignment process. We adopt similar techniques in our architecture.

Transmit Power Control: Transmit power control also has a significant influence on the performance of a wireless LAN [21]. Optimal power-level assignment is similar in hardness to channel assignment, and, for the coverage planning problem, has been shown to be NP-complete [7]. Many techniques have been proposed in the literature for computing optimal power levels for access points [8, 9]. We also propose a heuristic that we show works well in optimizing the performance of our architecture. Note, although channel assignment and power control have each been studied independently in past work, *we are not aware of any other work, other than our own [6], that solves them both simultaneously*.

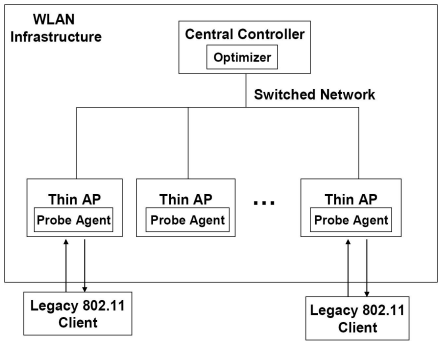


Figure 1: SMARTA System Architecture

3. SMARTA ARCHITECTURE

The SMARTA architecture is illustrated in Figure 1. The central controller coordinates the channels and power levels of the thin access points. The choices of channels and power levels are decided based on optimizing a utility function, whose value is computed using measurements performed by the access points. The controller periodically cycles through five phases: startup, channel assignment, annotation, power-level assignment, and refinement.

In the first or startup phase, the controller obtains the desired performance objective(s) from a network administrator. We assume that the administrator provides the parameters in the form of weights controlling a utility function. We expect manufacturers to provide carefully chosen defaults, so that, in practice, the network administrator could simply choose an objective such as 'maximize throughput' or 'minimize delay' instead of numerically choosing weights. This is akin to laptop users choosing verbal objectives such as 'maximize battery lifetime' or 'maximize performance', which are then translated into specific settings for disk spin-down timers and screen brightness.

The utility of a particular system configuration is determined jointly by the weights chosen by the administrator, the current workload, the current RF coverage, and the degree of interference between APs and clients in the system. To keep track of these parameters, the controller computes and periodically updates a "Conflict Graph" (or CG) [18], which is a graph where nodes are APs and there is an edge between two APs, if they interfere when assigned the same channel, *assuming they are transmitting at maximum power* (which is the worst case). In the second or channel assignment phase, the optimizer makes use of the CG to generate optimal channel assignments for the access points using the algorithm described in Section 7.1. At the end of this step, every AP is assigned a 'good' channel. We do channel assignment before power-level assignment because changing an AP's channel affects all clients associated with it. In contrast, changing its power level is not likely to significantly affect most clients. So, we assign channels at a slower time scale, and then refine power levels at a faster time scale.

In the third or annotation phase, the CG is augmented further to generate an *annotated conflict graph*, or ACG. This is similar in spirit to the conflict set ideas proposed in [22]. The annotated conflict graph adds clients to the conflict graph, which previously only contained access points. During ACG construction, access point channels may be re-assigned to reflect client information in the channel assignment process. This two-step channel assignment process is discussed in greater detail in Sections 7.1 and 7.2.

In the fourth or power-level assignment phase, SMARTA computes optimal power levels for access points. The power control algorithm used for this purpose is described in Section 7.3.

After this procedure completes, SMARTA moves to the fifth or refinement phase. In this phase, the power levels of access

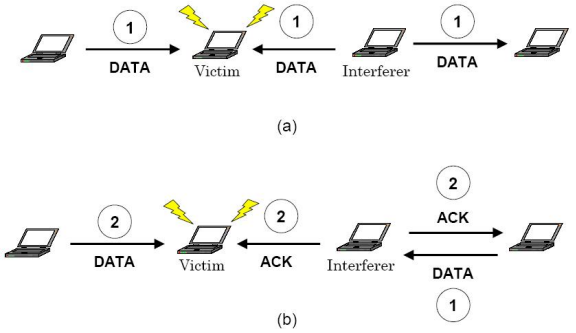


Figure 2: (a) illustrates a Data-Data collision scenario where *victim* is the node experiencing interference. (b) illustrates a Data-Ack collision scenario wherein the direction of traffic flow at the *interferer* is reversed. The steps that occur in each scenario are labeled accordingly.

points are altered to account for 'small' dynamic changes in the environment. This allows the system to evolve the configuration in response to changes in the environment. However, there may be circumstances where a large change in the environment is observed (e.g., a large number of users flock to a particular location) causing the current assignment of channels and power levels to be sub-optimal. This requires re-computing the configuration from scratch. Specifically, if the change in utility exceeds a significance threshold, the system discards the current ACG and starts the optimization process from the beginning, by returning to phase 2. Otherwise, it remains in the refinement phase.

The next sections describe each of these phases in greater detail. We first discuss the utility function model.

4. UTILITY MODEL

We use utility functions to characterize the benefit from a particular system configuration. The function is typically a linear combination of terms, where each term has a weight reflecting its importance to the network administrator. Note that this approach allows us to overcome the inherent problem of multi-objective optimization with conflicting objectives.

Utility functions can capture any type of performance objective and we discuss some common objectives next. Note that, although we are presenting some typical performance objectives, SMARTA is agnostic to the actual utility function chosen by the network administrator. Here we focus on objectives that maximize aggregate network throughput. Fairness criteria can also be captured in the utility function, and we leave their modeling to future work. SMARTA correctly chooses operating parameters to maximize the utility function *independent* of its form.

Let N be the total number of access points, p_1 to p_n represent the performance parameters to be captured, and w_1 to w_n be their respective normalized weights. Then, an example of a typical utility function for a wireless LAN deployment can be stated as follows:

$$U_{total} = \Sigma_{i=1}^{N} U_i \quad (1)$$

where,

$$U_i = w_1 p_1 + w_2 p_2 + ...w_n p_n \quad (2)$$

Equation 1 represents the aggregate utility of the wireless LAN, and Equation 2 represents the utility obtained by each of the access points (i representing a given access point). Next, we describe some example instantiations of p_i.

The Utility of Throughput

The utility gained from throughput depends on the nature of the client application. If it is real-time, then, as long as the throughput exceeds the required minimum value, full utility is achieved. On the other hand, for non-realtime applications, utility monotonically increases with increasing throughput. Suppose n non-realtime clients and m realtime clients are associated to the access point. Then, the aggregate utility provided to all clients is,

$$U_{clients} \quad = \quad n * U_{nrt} + m * U_{rt} \quad (3)$$

where U_{nrt} is a monotone function and U_{rt} is a clamped function, of the achieved throughput. The achieved throughput can be obtained by an AP by counting the number of packets sent to (or by) the client.

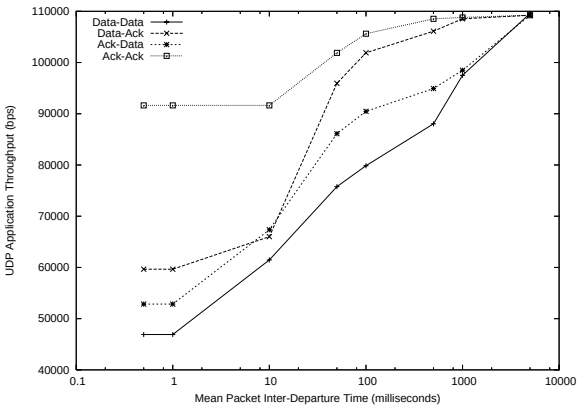


Figure 3: Throughput obtained by a node in the presence of interference. The x-axis indicates the mean delay between successive packets sent by the interferer (see Figure 2)

Effect of Interference on Utility

Suppose client a is associated with AP A and is potentially interfered with by AP B. How should this be modeled? Our intuition is that if B is mostly idle, then a is unaffected. However, if B is mostly busy, then a is likely to pay a price for this. Essentially, we want to map B's load to its expected effect on a, that is, the disutility to a due to the drop in its throughput.

Analytical models that quantify the effect of such interference are known, but they are quite complex even for very simple scenarios [14]. They are also limited in their ability to accurately model the impact of interference. Instead, we choose to empirically measure (to first order) the effects of interference on the throughput obtained by the interfered node, as follows.

We use the Qualnet [1] simulator and vary the sending rate of the interferer, which is transmitting UDP-based CBR traffic. The interfered node also transmits similar traffic at rates high enough to saturate the medium. This models the worst case by analyzing the impact of interference on high-throughput flows. We analyze four collision scenarios (Data-Data, Data-Ack, Ack-Data and Ack-Ack) using a simple four node topology, two of which are illustrated in Figure 2[1]. The results are shown in Figure 3. Packet inter-departure times at the interferer are independent and identically distributed using an exponential distribution, with a mean shown on the x-axis. Data-Data collisions have the greatest impact on the drop in throughput of the interfered node. These values obtained from simulation can thus be used to *quantify* the effect of interference by subtracting the carried load (shown in Figure 3) of the interfered node from its true offered load. Of course, this is by no means an exhaustive study, but our goal is to attempt to measure the degree of non-linearity in the effect of an interferer's load on the interfered node's throughput. As can be seen, for the most part, the effect

[1]For each scenario name, the first packet type corresponds to packets being received by the interfered node, whereas the second packet type represents packets interfering with the reception at the interfered node.

is log-linear, and we therefore model it with a simple log-linear model, using an empirically-derived slope. In particular, the effect of interference is a function of the load of the interfering source (represented by the value on the x-axis in Figure 3). Of course, more sophisticated models of this effect are possible, and we leave that to future work.

As described in more detail in Section 6, in reality four interference scenarios can occur in a wireless LAN deployment, based on nodes that are participating in the scenario (i.e. whether they are access-points or clients). These are inter-access-point interference (IAP), access-point-client interference (OAP/OC), and inter-client interference (IC) [2]. Thus, the total interference in the network is the sum of these individual interferences and can be expressed as:

$$\begin{aligned} U_{int} &= -(\Sigma_{i=1}^{N}\Sigma_{j=1}^{N} IAP_{ij} eff_i + \Sigma_{i=1}^{N}\Sigma_{v=1}^{K} OAP_{iv} eff_i \\ &+ \Sigma_{v=1}^{K}\Sigma_{i=1}^{N} OC_{vi} eff_v + \Sigma_{u=1}^{K}\Sigma_{v=1}^{K} IC_{uv} eff_u) \quad (4) \end{aligned}$$

where eff_i is the (assumed log-linear) effect of interference by access-point/client i on the throughput of the interfered access point or client. The functions IAP, OAP, OC, and IC are boolean functions that indicate the presence or absence of interference between pairwise nodes.

5. THE ANNOTATED CONFLICT GRAPH

A conflict graph succinctly represents the degree of interference between APs [18]. It is defined as a graph $G = (V, E)$, where V is the set of vertices and E the set of edges such that:

- $V = \{ap_1, ap_2, ap_3, \ldots, ap_n\}$, where ap_i is access point i.
- $E = \{(v, u) | f(ap_v, ap_u) \leq 0\}$
- $f(i, j) = -(IAP_{ij} eff_i)$,
  where, IAP_{ij} indicates the presence/absence of interference from access-point i on access point j and eff_i is the effect of interference on ap_j[3].

A conflict graph is therefore a *directed graph* where each edge represents interference (or conflict) caused by an access point at which the edge originates, on an access point at which the edge terminates. Due to wireless channel characteristics, interference between access points may not be symmetric.

The conflict graph is used during channel assignment to minimize the number of conflicts that occur between access points. This reduces to a graph-colouring problem, which is NP-hard [17]. In Section 7.1, we discuss a heuristic for channel assignment based on this conflict graph.

To perform power control, it is necessary to extend the conflict graph to include clients and AP loads, similar to the approach discussed in [22]. This *annotated conflict graph* has two types of edges between a client and an access point. If a client is associated with an access point, an undirected *association edge* is added between them. If a client interferes with an access point to which it is not associated, or an access point interferes with a client to which it is not connected, a directed *interference edge* is added between them. Finally, if clients interfere with one another, an interference edge is added between them. Figure 4 shows an illustration of the ACG. Note that channels that had been assigned before the creation of the ACG may be refined during ACG construction. This is elaborated in greater detail in Section 7.2.

Interference edge weights are derived using techniques described in Section 4. Association edge weights correspond to the utility that clients receive from their access points.

We point out that the conflict graph models the maximum possible number of conflicts, which corresponds to all access points transmitting at maximum power and using the same channel.

[2] We do not consider external interference in our model.

[3] The function f(i, j) is only defined for access points that interfere with each other when transmitting at maximum power using the same channel, and not across all pairs of APs.

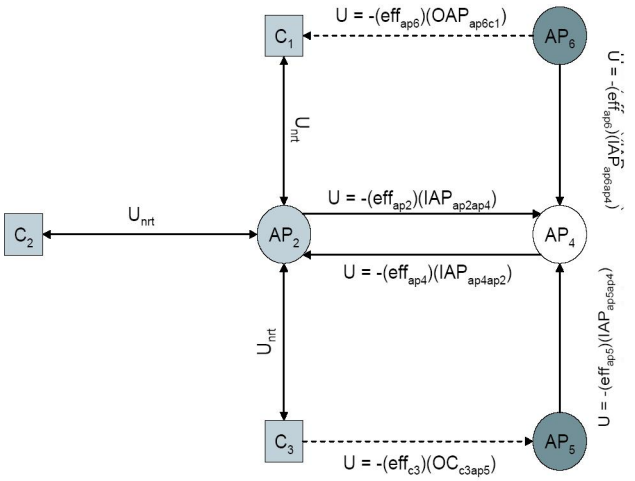


Figure 4: Annotated Conflict Graph. Circular vertices are access points and square vertices are clients. Basic conflict graph (used for channel assignment) contains only the circular vertices, shown in the figure. Clients have the same channel as their associated access point.

6. ACG COMPUTATION

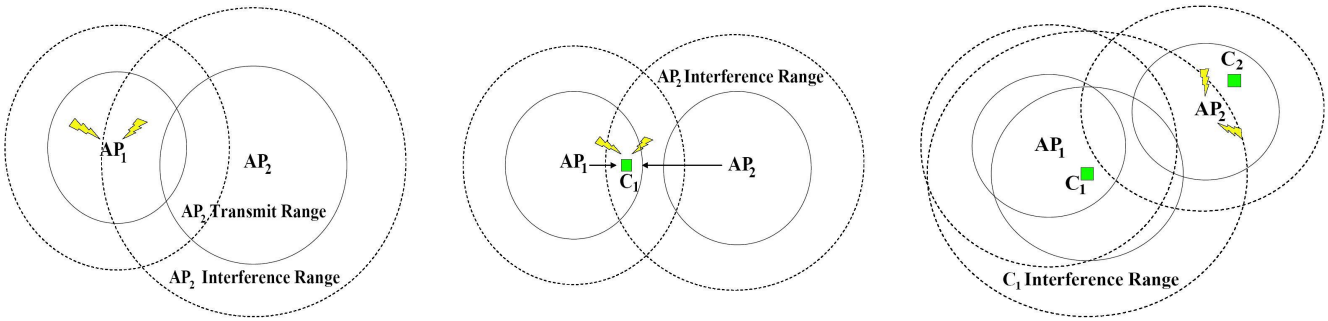
The annotated conflict graph requires a number of parameters to compute the utility of the system. This information has two parts; disutility corresponding to interference in the environment, and positive utility corresponding to utility that clients receive from the system. Interference disutility is captured by means of infrastructure-based testing using a probing agent, discussed next (contrasted with the client modifications required by [22]). Positive utilities are computed by passively observing statistics such as the number packets sent and received by each AP to and from each client (per observation interval), using techniques discussed in more detail in [6]. Due to space limitations, we only discuss the role of the probe agent.

As in [22], we classify interference scenarios in terms of the distance of the interference (in hops) from the infrastructure. For instance, inter-AP interference is zero-hops away from the infrastructure, since APs are directly connected to the wired backbone. The basic intuition is that as the interference moves further away from the infrastructure, it becomes progressively harder to detect and resolve. For each scenario, we prescribe a test to detect the existence of that scenario. In the sequel, the *Tester* is the entity that transmits the probe packet. It may also observe interference at nodes not capable of doing so themselves, e.g. legacy 802.11 clients. A *Sensor* is a node that checks to see if the *Tester* is interfering with it. All tests assume time synchronization; techniques to achieve synchronization within a few μs are described in [19]. Note, these tests do not require any underlying wireless propagation model for their operation, making them applicable to real-world scenarios.

6.1 Inter-AP (Zero-Hop) Interference

If the interference range of an access-point covers a neighbouring access-point, the overlapped access-point suffers interference from transmissions of the neighbouring access point (as shown in Figure 5(a)). Inter-access point interference is 'zero hop' interference because interference is experienced *zero hops* from the infrastructure.

The test for detecting zero-hop interference is as follows. One access-point acts as the tester while all other access-points act as sensors. The tester transmits m broadcast packets and the sensors listen for interference. During a broadcast, the sensor observes whether there is a change in the state of the channel, i.e., whether the channel transitions from *idle* to *busy*. If so, then with high like-



(a) Zero-Hop Interference (b) Single-Hop (OAP) Interference (c) Single-Hop (OC) Interference

Figure 5: Illustrations of *zero* and *one-hop* interference

lihood, the sensor is in interference range of the tester[4]. If it is also able to decode the packet, then it is in transmission range as well. The tester sends m broadcast packets for this test to increase confidence in the results. As illustrated in [6], a relatively small value of m, around 5, suffices for this purpose.

Each access-point performs this test. Therefore, the total number of tests required to detect zero-hop interference is bounded by $O(N)$, where N is the number of access-points.

6.2 AP-Client (One-Hop) Interference

We now describe two *one hop* interference scenarios that involve both clients and access points.

Overlapping Access Point (OAP)

Consider the case where an access-point's interference range covers a client connected to a neighbouring access-point. The client experiences interference from this access-point, from whom the client may or may not be hidden. If the client is hidden, packets being sent by it will be suppressed due to contention, and those being received will be susceptible to collision with packets transmitted from the interfering access-point. This is shown in Figure 5(b), where C_1 is associated to AP_1 and experiences interference from AP_2.

To detect this scenario, the following test is performed. The tester, which is the access-point to which the client is associated, transmits an RTS packet to the client, while the sensor which is the access-point that is interfering with the client simultaneously transmits a broadcast packet. Once RTS transmission is complete, the tester sets a timer equal to $(SIFS + Delay_{CTS} + Delay_{bcast})$, awaiting receipt of a CTS from the client, where $Delay_{CTS}$ is the propagation delay for a CTS packet and $Delay_{bcast}$ is the propagation delay for a broadcast packet[5]. If the broadcast packet and the RTS packet collide at the client, the client will not receive the RTS transmission correctly. Thus, it will not respond with a CTS, causing the tester to time out. The tester can then assume the RTS packet collided with the sensor's broadcast. The test completes after either the tester receives a CTS from the client or it times out in the process. This test is repeated m times. Since we need to perform this test for each client-AP pair and there are a total of C clients and N APs, the number of tests required to detect OAP interference is bounded by $O(NC)$. We realize that for large N and C, this is not scalable. We plan to look into techniques for scaling the OAP test in future work.

[4]We adopt a conservative approach in this test as well as the OC test described further by also classifying exposed terminals as sources of interference. Techniques for making this more accurate are possible, one of which is highlighted in the discussion section of the paper.

[5]We wait the additional broadcast propagation delay to ensure the client has sufficient time to reply with a CTS if it carrier senses the sensors broadcast but does not actually experience interference from it.

Overlapping Client (OC)

In this scenario, the client's interference range covers an access-point other than the access-point to which it is associated. If the access-point is hidden from the client, packets being sent by it will be suppressed due to contention, and those being received will be susceptible to collision with packets transmitted from the interfering client. This is shown in Figure 5(c), where C_1 is associated to AP_1 and causes interference on AP_2.

In order to detect OC interference, the following test is performed. The tester, which is the access-point to which the client is associated, transmits an RTS packet to the client. Upon receiving the RTS, the client responds with a CTS. During the CTS transmission, the sensor which is the access-point that is experiencing interference from the client observes to see a change in the state of the channel. If the sensor detects a change, then client-access-point interference exists between the sensor and the client. Once the tester receives the CTS packet from the client, the test is complete. This process is also repeated m times to increase our confidence in the result.

Note that if the sensor also experiences inter-access-point interference from the tester, then it must ignore channel state changes during the transmission of the RTS. Thus the sensor ignores state changes for a duration equal to the propagation delay of the RTS packet, from the time at which the tester initiated the RTS transmission. In this test, all neighbouring APs can simultaneously act as sensors, effectively limiting the number of such tests that need to be performed. Because this test needs to be performed for each client in the network, and we have C clients in total, the total number of tests required to detect OC interference is bounded by $O(C)$.

6.3 Inter-Client (Two-Hop) Interference

Clients may also mutually interfere with each other. For this scenario, we are interested in the case where the interfering clients are associated with separate access points because clients connected to the same access point can mitigate interference using RTS/CTS. Note that clients interfere with each other only if their respective access points use the same channel for communication.

For this case, two scenarios can arise, one of which is shown in Figure 6. In this scenario, the client experiences interference from a neighbouring client while it is receiving data (C_1). Therefore it is not able to correctly decode packets from the sender. The second scenario corresponds to clients that mutually contend for the medium. This scenario is described in greater detail in [6].

The following test detects inter-client interference. The tester (any one of the APs) sends a dummy data packet to its client. Once transmission is complete, the sensor (second AP) begins its part of the test. After waiting a SIFS interval, it initiates transmission of a dummy data packet to its client. Once transmission is complete, the sensor awaits an acknowledgement of its data packet. If it receives an acknowledgement within a timeout period of $(SIFS + Delay_{ACK})$, where $Delay_{ACK}$ is the propagation delay for an ACK packet, then the tester's client does not interfere with the sen-

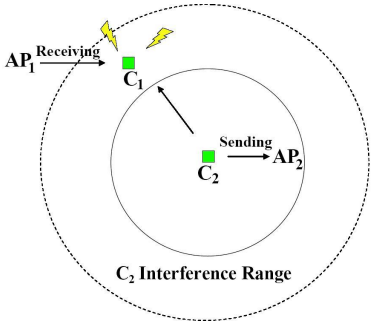


Figure 6: Illustration of *two-hop* interference scenario. C_2's data packets collide with data packets being received by C_1. AP_1 and AP_2 interference ranges have been elided for clarity.

sor's client.

The intuition for this test is the following. The sensor transmits its data packet when the tester's client is responding to the tester with an ACK. If the tester client's ACK collides with the data transmission being received by the sensor's client, the sensor's client will not be able to properly decode the data transmission. Therefore, it will not respond to the data packet with an ACK. Timing out of the sensor on the ACK is thus an indication of interference from the tester's client on the sensor's client. Interference detection in the reverse direction can also be done using a symmetric test. This test is also performed multiple times to reduce chances of a poor channel from affecting the results of the test. In the worst case, each client must perform such a test with all other clients, causing the overhead of this interference test to be bounded by $O(C^2)$.

Note that the interference tests we described must be conducted in a *'clean'* (i.e interference-free) environment. To arrange for this, the controller asks all APs to both stop their transmission and to force clients in their range to also stop transmission by broadcasting a CTS-to-self [6]. This generates the interference-free environment in which to conduct interference tests. We have analytically studied the overhead of conducting such tests along with techniques to mitigate it [6]. However, due to space limitations, we omit its discussion here.

7. OPTIMIZATION ALGORITHMS

We first discuss our approach to optimal channel assignment and then discuss the details of power control.

7.1 Channel Assignment

Channel assignment attempts to allocate orthogonal channels to nodes in the conflict graph that have an edge between them. Once completed, channels should be rarely changed because this disrupts service for clients. This is particularly important in the SMARTA architecture because legacy IEEE 802.11 clients cannot be instructed to change channels and are therefore disconnected if the AP frequently changes its channel.

To minimize channel changes, channel-assignment optimization is done on the basic conflict graph that deals only with access-point conflicts. Of course, we still need some way to deal with client conflicts and this is done during construction of the annotated conflict graph, discussed next. The algorithm to perform optimal channel assignment is called Randomized One-point optimization ($RanOp$) and bears some similarity to the approach described in [22].

The algorithm first assigns a random channel to each access point and computes the current total number of conflicts . Then, considering each access point (a_i) in turn it computes the gain in utility (in terms of reducing the total number of access-point conflicts) by switching that access point to a different channel. It computes the gain in utility for the access point on all channels and selects the channel C that yields the greatest gain for a_i. It then checks

Algorithm 1 wIR Power Control Algorithm

```
1: A = {a_1, a_2, ..., a_i} /* set of access points */
2: while true do
3:    u = ComputeTotalUtility(A)
4:    θ = MaxConflictAP(A)
5:    Z = {z_i | neighbour(θ, z_i) = true}
6:    for i = 1 ... | Z | do
7:       AdjustWeight(θ, z_i)
8:    end for
9:    γ = MaxConflictEdgeAP(θ, Z)
10:   ReducePowerLevel(γ)
11:   if u > ComputeTotalUtility(A) then
12:      IncreasePowerLevel(γ)
13:      Terminate.
14:   end if
15: end while
```

whether changing a_i to C yields an improvement in utility that is larger than the best utility gain seen in the iteration so far. If so, (a_i, C) is labeled as the best improvement seen so far. Because the algorithm performs this operation across all access points, it selects the access point and channel change that yields the largest gain in overall utility. This process repeats until we reach a configuration where any further one-point alterations do not yield a gain in utility. Because the solution of the algorithm may depend on the initial assignment of channels to access points, we perform multiple runs of the algorithm and choose the best solution (in terms of utility) among them.

7.2 Channel Refinement

In the second phase of channel-assignment, we refine channel allocations as an optimization of the assignment we computed previously. Note, for channel refinement we only consider optimization of assignments that keep the number of inter-AP conflicts constant. Inter-AP conflicts can be considered the most severe type of conflicts and those that are likely to persist over longer periods of time than conflicts involving clients. This is why we only consider them as part of the RanOp algorithm. For channel refinement, whenever we add a client to an AP (to construct the ACG), we try all other channels for that AP to see if we can reduce the total number of client conflicts, keeping the number of inter-AP conflicts constant. If such a channel is available (e.g. in the case of 802.11a), the access point is switched to that channel. If not, the access point remains on the same channel. This procedure is local to an access point as it does not require AP coordination to perform the channel search.

7.3 Power Control

Power control can be done quickly, even on a per-packet basis. However, two constraints make the power control problem challenging. First, power control needs to ensure that clients do not lose service by reducing an AP's power level too much. Second, every alteration to access-point power causes the underlying ACG to change. Therefore, we will need to re-compute (or refine) the ACG for every change in access-point power.

Our power control technique proceeds in two steps. First, we compute optimal power levels for all access points, taking the change in the ACG into account. Second, we refine access point power-levels to allow the system to adapt to changes in the environment. Due to space limitations, we only describe the first step of power control here.

The algorithm for computing optimal power-levels (called *weighted Iterative Reduction* (wIR) and shown as Algorithm 1) proceeds as follows. Initially, all access points are set to transmit at maximum power and we compute the total utility of this configuration ($ComputeTotalUtility(A)$). Note, the algorithm re-computes this utility in every iteration, before per-

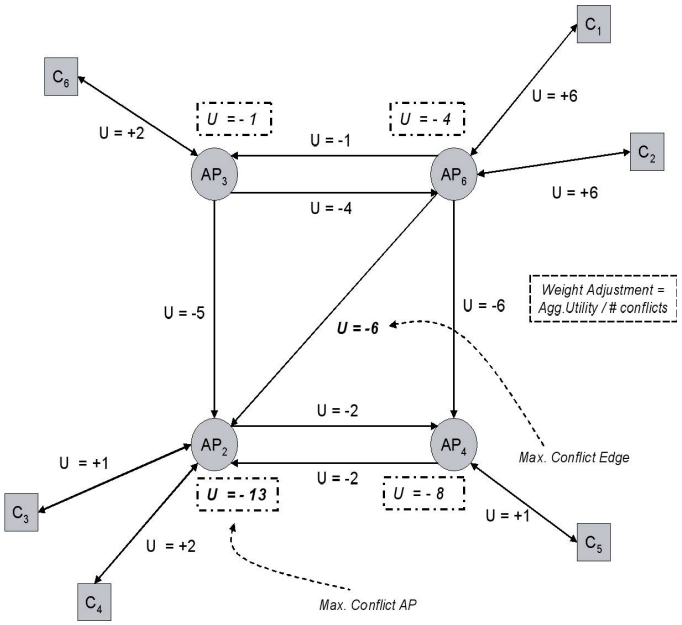


Figure 7: AP_2 is identified as the maximum conflict AP and the edge from AP_6 to AP_2 represents the maximum conflict edge, before edge re-weighting is done.

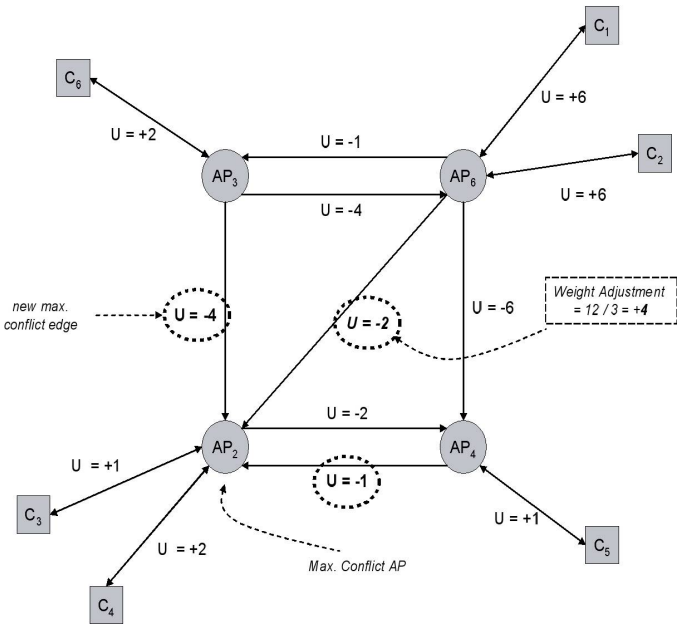


Figure 8: After edge re-weighting (shown in dashed circles), AP_3 is identified as the AP that has the maximum conflict edge to AP_2. AP_3 and AP_4 edge weights to AP_2 only change slightly because these APs provide very little utility to their clients.

forming the steps outlined further. In each iteration, the algorithm finds the access point that has the greatest number of conflicts ($MaxConflictAP(A)$). This is the AP whose sum total number of conflicts on all incoming edges from neighbouring APs is the greatest[6]. The algorithm then re-weights these incoming edges ($AdjustWeight(\theta, z_i)$) as follows: For each AP that interferes with the maximum-conflict AP, the incoming conflict edge's weight is increased proportional to the amount of utility that this AP provides to its clients (as shown in Figures 7 and 8). Thus, edge weights are adjusted by adding a $\frac{U}{E}$ positive value to the original weight, where U and E are the aggregate client utility provided and the total number of access point conflicts caused by the AP from which the edge emanates. After edge re-weighting, the algorithm selects the access point which induces the greatest conflict on the maximum conflict AP ($MaxConflictEdgeAP(\theta, Z)$), and instructs it to reduce its power level by one step ($ReducePowerLevel(\gamma)$). This repeats in successive iterations until there is no further improvement that can be made and a decrease is detected in the overall utility, at which point the algorithm terminates (after reversing the last power alteration). This approach rewards APs that have more active clients, so that they are less likely to have their power reduced.

8. EVALUATION

We now present an evaluation of our architecture, evaluating interference estimation, optimization, and the ability to dynamically re-configure in response to changes in the wireless environment. Due to space limitations, we do not present validation results for interference estimation, although we have studied this in detail in [6].

We first describe the simulation environment and network scenarios we considered in our evaluation and then discuss our results.

8.1 Methodology

8.1.1 Simulation Environment

We used the well-known QualNet simulator [1]. The central controller is emulated by means of a coordination component. Each thin access point houses two radios, and thus two MAC layers: A standard IEEE 802.11 compliant MAC layer and an *Environmental Sensing (ES)* MAC. The ES MAC supports the functionality for the probing agent. It periodically conducts the tests outlined in Section 6. Data is only sent on the 802.11 MAC. The clients implement the ES MAC in simulation for the sake of simplicity. In practice, a client does not require multiple interfaces/MACs, and, in fact, can be completely unmodified.

Every 5 minutes, the central controller recomputes the current utility of the system. If this drops by more than 20%, the controller instructs access points to re-initiate interference estimation tests. Using this information combined with statistics collected by passively sniffing traffic on the IEEE 802.11 MAC, the central controller re-runs the RanOp channel assignment and wIR power control algorithms. Once complete, the controller re-evaluates the utility of the system at the next scheduled time step.

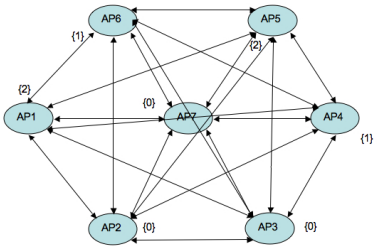
We have focused on specifying utility as the throughput that a client obtains from its access point, with the goal of maximizing aggregate network throughput. As described in Section 4, this does not take into account fairness criterion for clients, which we leave to future work. The statistics we captured (on the IEEE 802.11 MAC) in order to compute this metric include information on access point load and the number of packets sent/received per second from each client[7]. Unless otherwise indicated, each client also implements Auto-Rate Fallback (ARF) and thus the data rate will likely change during the course of the simulation. Interference is also modeled in the utility function, and is assumed to have a log-linear effect on the throughput received by clients, i.e. we use the load of the interfering source to compute the degree of interference. Both parameters, throughput and interference are assumed to carry equal weight in the utility function. We used the two-ray ground model in our simulations, and did not consider any fading. For each scenario, we initiated CBR traffic from access points to clients with 512 byte packets.

We evaluated two forms for our proposed optimization algorithms; one that only performs channel assignment (RanOp), and the other that also performs power control (RanOp-wIR). These were evaluated against the channel and static power configuration

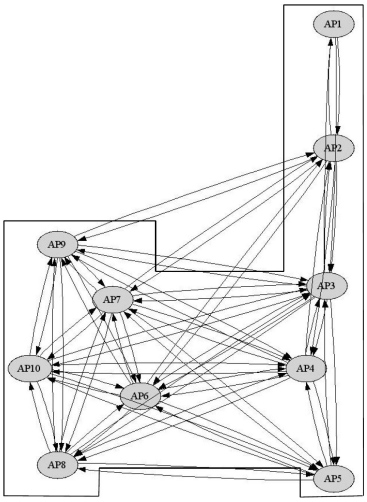
[6]This particular algorithm does not consider client conflicts, though more sophisticated versions of it can easily incorporate such information.

[7]Exponential averaging was used to smooth out abrupt changes to each metric.

(a) DC AP layout blueprint. Stars indicate AP locations.



(b) Circular (Star) topology conflict graph with channel assignment (using 3 channels) shown in curly brackets.



(c) DC AP Conflict Graph at transmit power of 30dbm.

Figure 9: Simulation Topologies

currently chosen by the network administrator for the building seen in Figure 9(a). Channels were assigned based on an extensive site survey that was carried out for the building. Moreover, to illustrate the benefits of our proposed centralized channel allocation algorithm, we compare it against a decentralized Least Congested Channel Search (LCCS) approach, discussed in [16]. LCCS is the current state-of-the-art algorithm for channel assignment and operates as follows: Each AP periodically observes data transmissions from other access points and clients on its channel. If the transmissions exceed a pre-specified threshold, it moves to a channel that is less congested. LCCS serves to show how well local tuning can perform in comparison to centralized channel assignment.

8.1.2 Simulation Scenarios

Our evaluation has three parts. In the first part, we present micro-benchmarks to illustrate the correct operation of SMARTA. In the second part, we simulate a large university building that we will call 'DC' (illustrated in Figure 9(a)). This allows us to gauge the effectiveness of SMARTA in a more realistic environment. For this scenario, we assume clients are stationary and are continuously sending traffic. Finally, we also present micro-benchmarks for client mobility. These micro-benchmarks allow us to observe the behavior of the SMARTA architecture in dynamic scenarios.

8.2 Results

We first discuss two micro-benchmarks to validate the correct operation of SMARTA.

8.2.1 Micro-benchmarks

Linear Topology: We first consider a simple linear topology with four APs. The transmit power of the APs is set such that an AP interferes both with adjacent APs and with neighbours of the adjacent APs. Clients are placed in between APs. Even if we consider just 3 channels, we can trivially produce a conflict-free colouring where AP channel assignment from left-to-right is given as (1, 6, 11, 1). This sequence can be repeated for an arbitrarily long AP chain, illustrating that linear topologies (typically found in hallways) are easier to address using just channel assignment, without power control. The result is shown in Figure 10(a). At $t = 120s$, when RanOp channel assignment is initiated, the aggregate network throughput improves significantly and remains steady thereafter.

Circular (Star) Topology: Next, we consider a circular topology where both channel assignment and power control prove to be useful in optimizing network throughput.

The circular topology we considered is illustrated in Figure 9(b). If we use only 3 channels, and have access points transmit at a nominal power of $20dbm$, a channel assignment for this topology will always yield solutions where at least two APs conflict with each other[8]. Thus, there are opportunities to improve network performance with the help of power control. This is illustrated in Figure 10(b). Improvements in throughput occur in identifiable stages where initially, all APs are transmitting using the same default channel. At $t = 120s$, SMARTA initiates channel assignment, producing the channel assignment shown in Figure 9(b). Note, because the topology considered here is a clique, a good channel assignment will equally partition APs across each of the channels, where the total number of conflicts is minimized. RanOp produces an assignment which maintains this property, resulting in a total of only 10 conflicts. This validates the ability of RanOp in finding good channel assignments for this topology.

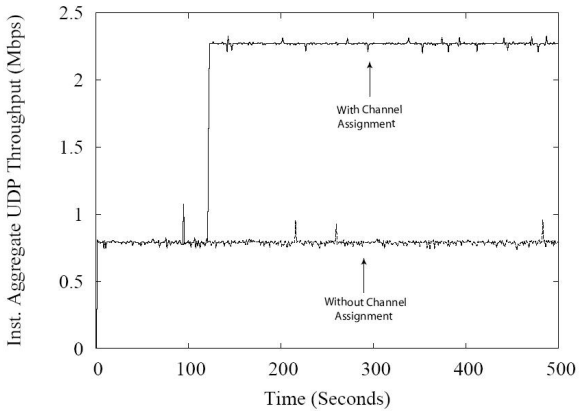
At time $t = 250s$, wIR power control begins. At $t = 380s$, we observe a significant increase in aggregate network throughput. While, RanOp produces an assignment that is almost 200% better than the original default assignment, wIR further improves performance by almost 25%.

Note that wIR terminates if changes in power levels do not produce observable improvements. This requires us to observe the network after each change. We find an observation window of 3s to be suitable. Of course, the accuracy of the observation is a function both of the length of the observation window and the agility of the wireless environment. This is a tuning parameter for the system and can be set based on the environment under consideration. Power control also requires an up-to-date ACG upon each iteration, which incurs an additional overhead. Optimizing the computation of the ACG is a challenging problem and a subject of future work.

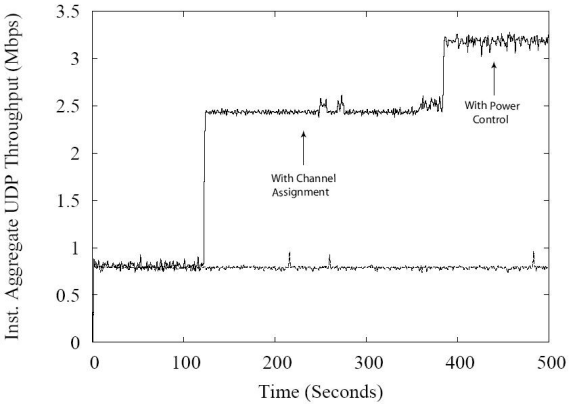
8.2.2 A More Realistic Scenario

We now discuss results of running SMARTA on the DC topology (Figure 9(a)). For these results, we randomly distribute clients within the coverage radius of each of the access points, whose size is determined by the transmit power of the access point. We analyze the performance of SMARTA on scenarios exhibiting a high degree of interference. Note that the degree of interference is affected by the transmit power of the APs/clients, the number of clients, and the

[8]Note, although RanOp may not produce the same assignment of channels to APs in each invocation of the algorithm, the sum total number of conflicts across all assignments should remain the same.



(a) Instantaneous aggregate client throughput on Linear topology. Improvement seen is a result of channel assignment.



(b) Instantaneous aggregate client throughput on Star topology. Initial improvement is because of channel assignment while the subsequent improvement comes as a result of power control.

Figure 10: Micro-benchmark results

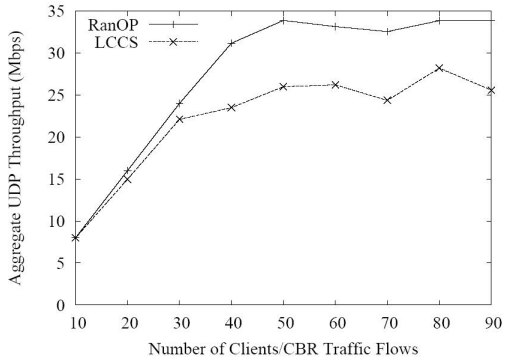


Figure 11: Aggregate client throughput at 30dbm using 12 channels

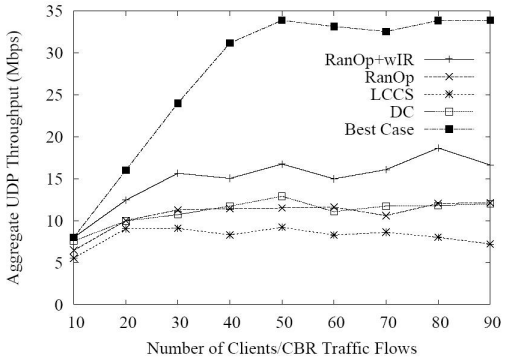


Figure 12: Aggregate client throughput at 30dbm using 3 channels

client distribution [8]. However, while the transmission power of access points is tunable and controllable, client density and distribution is not. Therefore, in order to independently study the effects of each, we decouple them in our simulations. For our results, we use the metrics of aggregate network throughput and per-packet delay to compare the different algorithms. We have also analyzed the distribution of flow throughput across clients [6]. However, due to space limitations, we omit these results.

In our scenarios, APs transmit at $30dbm$ and clients transmission power is set equal to that of the APs to facilitate connectivity even at coverage boundaries. We use $30dbm$ to stress test our system. We have analyzed the performance of the algorithms in low power scenarios (i.e. $20dbm$) as well and obtained promising results. However, due to space limitations, we also omit these results.

Referring to Figure 9(c), we see many access point conflicts. There are also client conflicts, not shown for clarity. With 802.11a (i.e 12 orthogonal channels), we can trivially eliminate all conflicts by assigning an independent channel to each AP. In this situation, the best possible solution is to assign a separate channel to each AP and setting each APs transmit power to maximum. We call this configuration 'best', and use it to benchmark solutions generated in other scenarios[9].

Throughput: Figure 11 shows aggregate client throughput against client density, for the case where we have 12 available channels. The 'best' curve corresponds to the case where we only perform channel assignment using RanOp channel assignment since we observed that our algorithm always produced conflict free assignments in this scenario. Because no power control is required in this scenario, RanOp-wIR performs identically to the best case, and thus we do not show it in the figure. We observe that, at high densities, due to its decentralized approach, even with 12 channels, LCCS is unable to optimally assign channels to access points. Of course, in low density environments, LCCS performs close to the best case because of the lower degree of interference.

Figure 12 shows aggregate client throughput for the 3 channel case. Not surprisingly, aggregate client throughput drops significantly for all the algorithms. However, observe that RanOp combined with wIR performs the best in this scenario. Because channel assignment cannot eliminate all conflicts, power control yields improvements. However, there is still a significant performance gap between the 'best' solution and our algorithms. Aside from the limited number of channels, this is because of the limitations of power control. Channel assignment has the ability to eliminate all types of conflicts (i.e. zero, one, and two-hop conflicts) whereas power control can only address OAP and zero-hop conflicts. This is because of the inability to adjust client powers. As a result, even a provably optimal power control strategy will ultimately be unable to eliminate all conflicts in such cases. Nevertheless, we still observe significant improvements over LCCS.

We also plot the performance curve for the hand-tuned DC channel configuration (i.e. the configuration decided by the network

[9]Strictly speaking, this may not be 'best' because we could potentially improve performance by carefully allocating clients to APs to evenly distribute load. However, load-balancing is not explored in this paper and is left to future work.

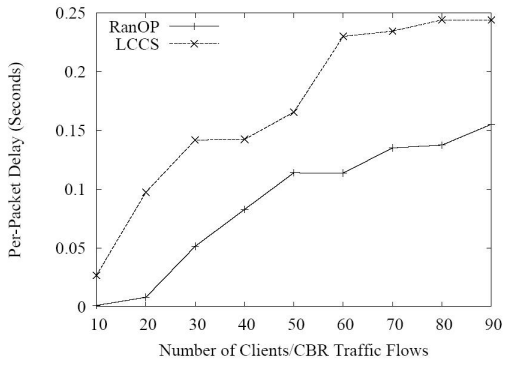


Figure 13: Per-packet delay at 30dbm using 12 channels

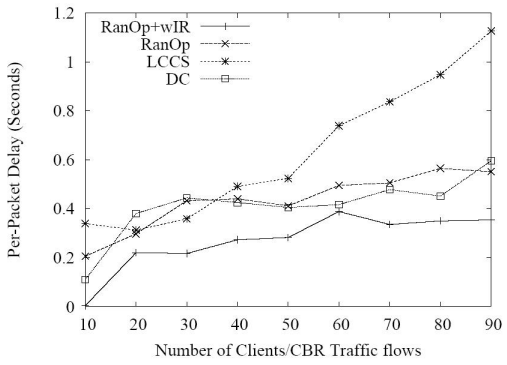


Figure 14: Per-packet delay at 30dbm using 3 channels

operator at our university). This configuration performs similar to the RanOp algorithm used in SMARTA. The reason that RanOp does not yield significant improvements over the hand-optimized DC assignment is because of the large number of conflicts. The number of conflicts significantly decreases the number of possibly good configurations that yield high throughput. Nevertheless, we still observe that in high power scenarios, RanOp is able to perform just as well as a carefully hand-optimized channel assignment and 50% better with the addition of transmit power control.

Per-Packet Delay: We also analyze per-packet delay for each of the algorithms. Per-packet delay is a crucial metric for delay-sensitive applications such as voice and multimedia. Note that the results we discuss here use the same throughput maximization utility function as was used for the previous results. We expect a utility function catering specifically to per-packet delay to perform even better.

Figure 13 plots per-packet delay results against client density, using 12 channels. The results for RanOp and combined RanOp-wIR are identical. We observe a significantly lower per-packet delay for the RanOp algorithm than that for LCCS. Moreover, the delay values for RanOp are almost always below $150ms$ across the board. This is an interesting result since delay requirements for most voice applications fall within this range. Thus, we believe that the centralized RanOp algorithm is well suited to supporting such applications even in very dense scenarios characterized by a large degree of AP/client conflicts.

Figure 14 presents similar results for the 3 channel case. Not surprisingly, per-packet delays have increased over the 12-channel case due to increased MAC contention delays and a larger number collisions. However, we observe that RanOp-wIR provides the lowest per-packet delay primarily because power control reduces APs collision domains significantly, reducing both MAC contention and the probability of packet collisions. LCCS performs the worst in this case with per-packet delays of over one second in very dense environments, demonstrating its limitations in these scenarios.

8.2.3 Effect of Mobility

We analyze the impact of mobility on the SMARTA architecture. Recall, SMARTA triggers re-computation of access point configurations if the change in utility is significant, i.e. exceeds a predefined utility change threshold α. For the purposes of our simulation, we set this threshold to 20%.

We construct two scenarios to analyze the impact of mobility. In the first scenario, a client moves between a set of access points, as shown in Figure 15. This is typical for an employee that might periodically visit the break room from her office. We use this scenario to illustrate the stability of SMARTA in reacting to small-scale changes that may occur in the environment. In the second

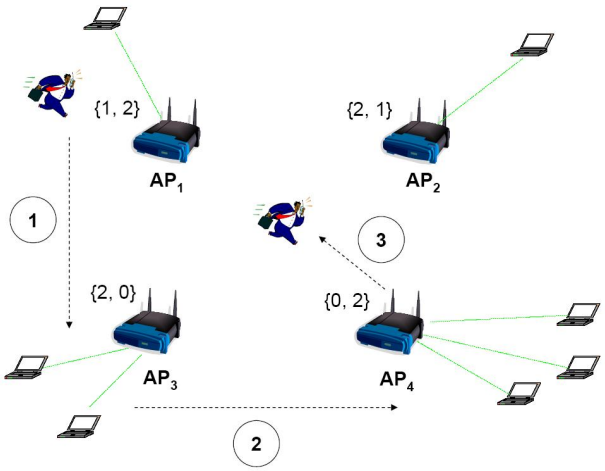


Figure 15: Micro-benchmark setup for analyzing impact of mobility. User mobility shown as dotted arrows with labels indicating steps followed by client. For large-scale scenario, channels are shown in curly brackets where the numbers depict assignments before and after large-scale change.

scenario, clusters of users move from different access points to a common access point. This is likely to occur in situations where groups of people gather together for a scheduled meeting and represents a large-scale change that SMARTA must cater for.

Small-Scale Scenario: Figure 15 illustrates the user mobility pattern considered in this scenario. A single user starts at AP_1 and moves between access points, finally ending up in between them. Note, the user disconnects and re-connects with AP_4 even during movement step 3. Changes in aggregate network throughput are illustrated in Figure 16. Before the initial move, at $t = 120s$, SMARTA computes optimal channel and power level configurations for the access points, causing the aggregate throughput to increase to approximately $6Mbps$. At time $t = 200s$, the user disconnects from AP_1 and re-connects to AP_3 at $t = 300s$. During this interval, the utility drops by approximately 16%, which is not below the change threshold and increases again at $t = 300s$. Thus, at $t = 420s$, when utility re-assessment is done, SMARTA does not initiate re-computation of channels and power levels. This process successively repeats without the utility change ever falling below the change threshold. In summary, we observe that SMARTA's use of utility-based triggers allows it to be resilient to oscillations that may occur as a result of small-scale changes in the RF environment. This is particularly crucial for legacy clients that may be affected by continuous changes in access point channels.

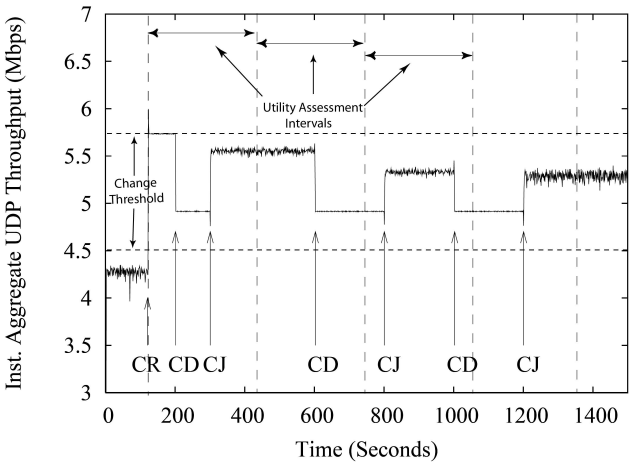


Figure 16: Instantaneous aggregate client throughput in small-scale scenario. Following events are shown: CR=Configuration Re-computation, CJ=Client Join, and CD=Client Disconnect. Utility assessment intervals shown as vertical dotted lines and change threshold shown as horizontal dotted lines.

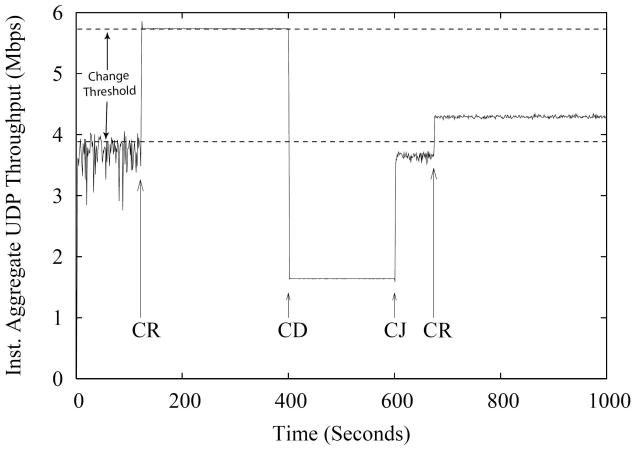


Figure 17: Instantaneous aggregate client throughput in large-scale scenario. The events shown are similar to those illustrated in Figure 16.

Large-Scale Scenario: For this scenario, we use the same setup as was used in the previous scenario. However, in this case, two groups of users move from separate access points to a common access point. Channel assignments based on initial user distribution are shown in Figure 15. At $t = 120s$, SMARTA performs optimal channel and power level assignment for all access points. At $t = 400s$, all clients from AP_1 and AP_4 disconnect and proceed to move towards AP_3. At $t = 600s$, all clients re-connect to AP_3, subsequently increasing its load. Utility re-assessment between the time when clients disconnect and reconnect is disabled to illustrate the effect of re-configuration after clients re-connect to the network. In reality, SMARTA would already account for this case during periods of disconnection as it will observe a large decrease in utility and re-assign channels and powers as a result to maximize utility for currently connected clients. At $t = 600s$, when clients re-connect, an increase in utility is observed. Note, SMARTA is only aware of the utility that was computed at $t = 120s$, during the last time re-configuration was performed. At $t = 680s$, a significant drop is observed and SMARTA re-initiates computation to improve system utility. Notably, the utility improvement is not very significant and in particular, does not match the utility of the configuration at $t = 120s$. This is due to the large number of clients connected to AP_3 and the excessive load on it. This reduces per-client throughput and contributes to the drop in aggregate client throughput even after the configuration is refreshed.

The scenarios outlined above provide insight into the ability of the SMARTA architecture to accurately determine the type of change that occurs in the environment. The utility change threshold is a tuning parameter for our architecture and can be set to suit the needs of the deployment environment.

9. DISCUSSION

The ideas and design goals embodying SMARTA lay the groundwork for a more sophisticated management solution for wireless LANs. We now highlight some enhancements and limitations of our approach.

9.1 Enhancements

While coordinated probe testing can identify the existence of interference, properly scheduling these tests is crucial in minimizing their impact on network performance. This becomes important for clients that only have a single radio and use the same channel (or set of virtual channels [13]) for both signalling and data. Due to space limitations, we are unable to present an analysis of this overhead as well as techniques that we propose for mitigating it. Additionally, though not affecting the correctness of our tests, very low carrier sensitivity threshold (CST) settings at nodes can cause SMARTA to be overly conservative in identifying the existence of interference (e.g. for the zero-hop test). Accurate tuning of the CST may prove useful [27], and is a subject of future work.

A key characteristic of our approach is the ability to support self-configuration entirely at the network end. While we have presented techniques for maximizing network performance, client association characteristics also largely impact the performance of the system. Thus, it becomes important for the network to determine the best association points for clients, in order to maximize performance for all clients. Infrastructure-directed association and load balancing techniques are thus required and can also be applied to the SMARTA architecture. Additionally, techniques for AP affiliation that involve clients implementing access point selection strategies are also possible and known [11, 28]. They may also be used in conjunction with the SMARTA architecture.

9.2 Limitations

One limitation of our work is that we measure client utility only from the perspective of an AP. Unfortunately, this does not allow us to measure what was actually received by a client: due to interference and other physical effects, what a client receives may not be what the AP sent. A more sophisticated approach to detect packet loss, or perhaps additional information from an agent running on a client, would improve matters [22].

A second limitation of our approach is the use of AP and client loads for modeling the effect of interference. However, load, by itself, may not be an adequate metric to model interference. For instance, a lightly loaded client transmitting at a low rate over the air due to ARF could actually cause more interference than a heavily loaded client transmitting at a higher rate. Therefore, it may be better to model interference as a function of the mean channel utilization: we are exploring this in future work.

A third limitation of our approach is that it serializes the process of channel assignment and power control for optimization. Jointly optimizing channel assignment and power control is a challenging problem, because of the tight coupling between these two parameters. A particular assignment of channels yields a new solution for power control, and vice versa. Our current work presents a point solution in a large solution space of possible approaches. Comparison with other types of techniques is left to future work.

Finally, a fourth limitation of our approach is the use of a centralized manager for coordinating access point configurations. A centralized solution may not scale to all scenarios (e.g. high-density deployments). We are investigating distributed solutions that parti-

tion the network into nearly-independent clusters to reduce the size of each subproblem.

10. SUMMARY

Modern-day wireless networks suffer from poor performance because of their inability to adapt to dynamic changes in their surrounding environment. We advocate the need for dynamic reconfiguration and propose SMARTA, a centralized architecture supporting measurement, optimization, and dynamic re-tuning of enterprise wireless LANs. SMARTA does not make any assumptions on the nature of the wireless environment, making it amenable to real-world deployments. We propose a set of probing tests for accurately measuring the existence of interference in real environments. These are used in an *annotated conflict graph* to model the utility from a particular system configuration. Two algorithms, RanOp for channel assignment, and wIR for power control, are proposed that make use of the ACG to identify opportunities for improving network performance. Finally, dynamic reconfiguration with the help of utility-based triggers allows the system to adapt to continuous changes in the environment. Through extensive evaluation, we illustrate the feasibility of these ideas and their wider application to self-configuration and tuning of wireless LANs for real-world deployments.

11. ACKNOWLEDGEMENTS

This research was supported thanks to grants from the National Science and Engineering Council of Canada, the Canada Research Chair Program, Nortel Networks, and Intel Corporation. We would also like to thank our shepherd, Augustin Chaintreau, and our anonymous reviewers for their valuable feedback and suggestions.

12. REFERENCES

[1] Scalable Networks Inc., QualNet Simulator version 3.7.

[2] Propagate Inc., AutoCell - The Self-Organizing WLAN. http://www.propagatenet.com/resources/index.html.

[3] Extricom Inc., Wireless LAN Switch DataSheet, 2005. http://www.extricom.com/imgs/Uploads/PDF/SWDataSheet.pdf.

[4] Meru Networks Inc., Virtual Cells: The Only Scalable MultiChannel Deployment. White Paper, 2005. http://www.merunetworks.com/pdf/Virtual_Cells_WP4_0705.pdf.

[5] A. Adya, P. Bahl, R. Chandra, and L. Qiu. Architecture and techniques for diagnosing faults in IEEE 802.11 infrastructure networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking (MobiCom)*, Philadelphia, PA, 2004.

[6] N. Ahmed. A self-management approach to configuring wireless infrastructure networks, Master's Thesis, University of Waterloo (UW). 2006. http://etd.uwaterloo.ca/etd/n3ahmed2006.pdf.

[7] N. Ahmed, A. Allavena, and S. Keshav. A mathematical model for optimal coverage planning in Wireless LANs. 2006. *Unpublished Manuscript*.

[8] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-management in chaotic wireless deployments. In *Proceedings of the 11th annual international conference on Mobile computing and networking (MobiCom)*, pages 185–199, Cologne, Germany, 2005.

[9] P. Bahl, M. Hajiaghayi, K. Jain, V. Mirrokni, L. Qiu, and A. Saberi. Cell breathing in wireless lans: Algorithms and evaluation. *IEEE Transactions on Mobile Computing*, 2006.

[10] Y. Bejerano and R. Bhatia. MiFi: A Framework for Fairness and QoS Assurance in Current IEEE 802.11 Networks with Multiple Access Points. In *23th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1229–1240, 2004.

[11] Y. Bejerano, S.-J. Han, and L. E. Li. Fairness and load balancing in wireless lans using association control. In *10th annual international conference on Mobile computing and networking (MobiCom)*, pages 315–329, 2004.

[12] R. Borndorfer, A. Eisenblatter, M. Grotschel, and A. Martin. Frequency assignment in cellular phone networks. In *Annals of Operations Research*, volume 76, pages 73–93, 1998.

[13] R. Chandra, V. Bahl, P. Bahl, and K. Birman. VirtualWiFi: Connecting to mutliple IEEE 802.11 networks with one WiFi card, http://research.microsoft.com/netres/projects/virtualwifi/default.htm.

[14] H. Chang and V. Misra. 802.11 link interference: A simple model and a performance enhancement. In *Proceedings of the 4th International IFIP-TC6 Networking Conference (NETWORKING 2005)*, pages 1330 – 1333, May 2005.

[15] S. J. Fortune, D. M. Gay, B. W. Kernighan, O. Landron, R. A. Valenzuela, and M. H. Wright. WISE design of indoor wireless systems: Practical computation and optimization. In *IEEE Computer Society - IEEE Computational Science and Engineering*, pages 58–68, 1995.

[16] J. Geier. Assigning 802.11b access point channels. 2002. http://www.wi-fiplanet.com/tutorials/article.php/972261.

[17] M. M. Halldorson, J. Y. Halpern, L. E. Li, and V. S. Mirrokni. On spectrum sharing games. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing (PODC)*, pages 107–114, St. John's, Newfoundland, Canada, 2004.

[18] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom)*, pages 66–80, San Diego, CA, USA, 2003.

[19] L. G. Jeremy Elson and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA., December 2002.

[20] B. Kauffman, F. Bacelli, A. Chaintreau, K. Papagiannaki, and C. Diot. Self-Organization of Interfering 802.11 Wireless Access Networks. Technical Report 5649, INRIA, 2005.

[21] V. Kawadia and P. R. Kumar. Principles and protocols for power control in ad hoc networks. *Special Issue on Ad Hoc Networks*, 1, 2005.

[22] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh. A client driven approach for channel management in wireless LANs. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Barcelona, Spain, 2006.

[23] J. Padhye, S. Agarwal, V. N. Padmanabhan, L. Qiu, A. Rao, and B. Zill. Estimation of link interference in static multi-hop wireless networks. In *ACM Internet Measurement Conference (IMC)*, pages 305–310, Berkeley, CA, 2005.

[24] L. Qiu, P. Bahl, A. Rao, and L. Zhou. Fault detection, isolation, and diagnosis in multi-hop wireless networks. Technical Report TR-2004-11, Microsoft, 2003.

[25] A. Sheth, C. Doerr, D. Grunwald, R. Han, and D. Sicker. MOJO: A Distributed Physical Layer Anomaly Detection System for 802.11 WLANs. In *4th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Uppsala, Sweden, 2006.

[26] D. Tse and P. Viswanath. *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

[27] A. Vasan, R. Ramjee, and T. Woo. ECHOS - Enhanced capacity 802.11 hotspots. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1–9, San Diego, CA, USA, 2005.

[28] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, and D. Towsley. Facilitating access point selection in ieee 802.11 wireless networks. In *ACM Internet Measurement Conference (IMC)*, Berkeley, CA, 2005.